UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Programa de Doctorado en Automática, Robótica e Informática Industrial
Instituto de Automática e Informática Industrial

Surface Registration Techniques Applied to Archaeological Fragment Reconstruction

Ph.D. Dissertation of: Carlos Sánchez Belenguer
Director and tutor: Eduardo Vendrell Vidal

July, 2015

To Ana and Eduardo.

Abstract

Reconstruction of broken archaeological artifacts from fragments is a very time-consuming task that requires a big effort if performed manually. In fact, due to budgetary limitations, this is not even attempted in countless sites around the world, leaving vast quantities of material unstudied and stored indefinitely.

This Thesis dissertation faces the application of surface registration techniques to the automatic re-assembly of broken archaeological artifacts from fragments. To efficiently do so, the reconstruction problem has been divided into two groups: 3 degrees of freedom and 6 degrees of freedom problems. This distinction is motivated for two major reasons: archaeological interest of the application and computational complexity of the solution.

First kind of problems (3 degrees of freedom) deal with 2D objects or with flat 3D objects, like ripped-up documents or frescoes, respectively. In both cases, the mural paintings and engravings on the fragments' surface are of huge importance in the field of Cultural Heritage Recovery. In this sense, archaeologically speaking, the value of the reconstruction is not the final model itself, but the information stored in the upper surface. In terms of computation complexity, the reduced solution space allows using exhaustive techniques to ensure the quality of the results, while keeping execution times low. A fast hierarchical technique is introduced to face this kind of problems. Starting from an exhaustive search strategy, the technique progressively incorporates new features that lead to a hierarchical search strategy. Convergence and correction of the resulting technique are ensured using an optimistic cost function. Internal search calculations are optimized so the only operations performed are additions, subtractions and comparisons over aligned data. All heavy geometric operations are carried out by the GPU on a pre-processing stage that only happens once per fragment.

Second kind of problems (6 degrees of freedom) deal with more general situations, where no special constraints are considered. Typical examples are broken sculptures, friezes, columns... In this case, computational complexity increases considerably with the extra 3 degrees of freedom, making exhaustive approaches prohibitive. To face this problems, an efficient sparse technique is introduced that uses a pre-processing stage to reduce the size of the problem: singular key-points in the original point cloud are selected

based on a multi-scale feature extraction process driven by the saliency of each point. By computing a modified version of the PFH descriptor, the local neighborhood of each key-point is described in a compact histogram. Using exclusively the selected key-points and their associated descriptors, a very fast one-to-one search algorithm is executed for each possible pair of fragments. This process uses a three-level hierarchical search strategy driven by the local similarity between key-points, and applying a set of geometric consistence tests for intermediate results. Finally, a graph-based global registration algorithm uses all the individual matches to provide the final reconstruction of the artifact by creating clusters of matching fragments, appending new potential matches and joining individual clusters into bigger structures.

Resumen

La reconstrucción de objetos arqueológicos fracturados a partir de fragmentos es una actividad que, si se realiza manualmente, supone un gran coste temporal. De hecho, debido a restricciones presupuestarias, esta tarea no llega a abordarse en incontables yacimientos arqueológicos, dejando grandes cantidades de material sin ser estudiado y almacenado indefinidamente.

La presente propuesta de tesis aborda la aplicación de técnicas de registro de superficies a el re-ensamblado automático de objetos arqueológicos fracturados a partir de fragmentos. Por motivos de eficiencia, el problema de la reconstrucción se ha dividido en dos grupos: problemas de 3 grados de libertad y problemas de 6 grados de libertad. Esta distinción está motivada por dos razones: (1) el interés arqueológico de la aplicación final de las técnicas desarrolladas y (2) la complejidad computacional de la solución propuesta.

El primer tipo de problemas (3 grados de libertad) se enfrenta a objetos bidimensionales o tridimensionales planos como documentos fragmentados y frescos, respectivamente. En ambos casos, los murales y grabados sobre la superficie de los fragmentos son de gran importancia en el ámbito de la conservación del patrimonio cultural. En este sentido, desde el punto de vista arqueológico, el valor de la reconstrucción final no radica en el modelo en sí, sino en la información almacenada sobre su superficie. En términos de complejidad computacional, el reducido espacio de soluciones permite emplear técnicas de búsqueda exhaustivas que garantizan la corrección de los resultados obtenidos con tiempos de ejecución acotados. La técnica propuesta para abordar este tipo de problemas parte de una estrategia exhaustiva y, progresivamente, incorpora nuevas optimizaciones que culminan con una técnica íntegramente jerárquica. La convergencia y corrección de la solución propuesta están garantizadas gracias a una función de coste optimista. Los cálculos internos durante las búsquedas han sido optimizados de modo que sólo son necesarias operaciones de adición/substracción y comparaciones sobre datos alineados en memoria. Todas las operaciones complejas asociadas a la manipulación de datos geométricos son realizadas por la GPU durante una etapa de pre-procesamiento que se ejecuta una sola vez por fragmento.

El segundo tipo de problemas (6 grados de libertad) se enfrenta a situaciones más generales, en las que ninguna restricción especifica puede ser asumida. Ejemplos típicos son esculturas fragmentadas, frisos, columnas... En este caso, la complejidad computacional incrementa considerablemente debido a los 3 grados de libertad adicionales por lo que el coste temporal de las estrategias exhaustivas resulta prohibitivo. Para abordar este tipo de problemas, se propone una técnica dispersa eficiente apoyada en una fase de preprocesamiento cuyo objetivo consiste en reducir la talla de los datos de entrada: a partir de las nubes de puntos originales, puntos clave singulares son identificados gracias a un proceso de extracción de características multi-escala apoyado en el valor de saliencia de cada punto. Mediante el cálculo de una versión modificada del descriptor PFH (Persistent Feature Histograms), el vecindario local de cada punto clave es descrito en un histograma compacto. Empleando únicamente estos puntos y sus descriptores asociados, un algoritmo de búsqueda uno-a-uno muy rápido se ejecuta sobre cada par de fragmentos. Dicho proceso emplea una estrategia de búsqueda jerárquica de tres niveles, dirigida por la similitud entre puntos clave y que aplica un conjunto de tests de consistencia geométrica sobre los resultados intermedios. Finalmente, un algoritmo de registro global toma como datos de entrada todas las correspondencias individuales para generar la reconstrucción final del objeto.

Resum

La reconstrucció d'objectes arqueològics fracturats a partir de fragments és una activitat que, si es realitza manualment, suposa un gran cost temporal. De fet, a causa de restriccions pressupostàries, esta tasca no arriba a abordar-se en incomptables jaciments arqueològics, deixant grans quantitats de material sense ser estudiat i emmagatzemat indefinidament.

La present proposta de tesi aborda l'aplicació de tècniques de registre de superfícies a l're-enssamblatge automàtic d'objectes arqueològics fracturats a partir de fragments. Per motius d'eficiència, el problema de la reconstrucció s'ha dividit en dos grups: problemes de 3 graus de llibertat i problemes de 6 graus de llibertat. Esta distinció està motivada per dues raons: (1) l'interès arqueològic de l'aplicació final de les tècniques desenvolupades i (2) la complexitat computacional de la solució proposada.

El primer tipus de problemes (3 graus de llibertat) s'enfronta a objectes bidimensionals o tridimensionals plans com documents fragmentats i frescos, respectivament. En tots dos casos, els murals i gravats sobre la superfície dels fragments són de gran importància en l'àmbit de la conservació del patrimoni cultural. En este sentit, des del punt de vista arqueològic, el valor de la reconstrucció final no es basa en el model en si, sinó en la informació emmagatzemada sobre la seva superfície. En termes de complexitat computacional, el reduït espai de solucions permet emprar tècniques de recerca exhaustives que garanteixen la correcció dels resultats obtinguts amb temps d'execució acotats. La tècnica proposada per abordar aquest tipus de problemes part d'una estratègia exhaustiva i, progressivament, incorpora noves optimitzacions que culminen amb una tècnica íntegrament jeràrquica. La convergència i correcció de la solució proposada estan garantides gràcies a una funció de cost optimista. Els càlculs interns durant les recerques s'han optimitzat de manera que només són necessàries operacions d'addició / substracció i comparacions sobre dades alineats en memòria. Totes les operacions complexes associades a la manipulació de dades geomètriques són realitzades per la GPU durant una etapa de pre-processament que s'executa una única vegada per fragment.

El segon tipus de problemes (6 graus de llibertat) s'enfronta a situacions més generals, en què cap restricció especifica pot ser assumida. Exemples típics són escultures fragmentades, frisos, columnes ... En este cas, la complexitat computacional s'incrementa considerablement a causa dels 3 graus de llibertat addicionals pel que el cost temporal de les estratègies exhaustives resulta prohibitiu. Per abordar este tipus de problemes, es proposa una tècnica dispersa eficient recolzada en una fase de pre-processament l'objectiu del qual consisteix a reduir la talla de les dades d'entrada: a partir dels núvols de punts originals, s'identifiquen punts clau singulars gràcies a un procés d'extracció de característiques multi-escala recolzat en el valor de saliència de cada punt. Mitjançant el càlcul d'una versió modificada del descriptor PFH (Persistent Feature Histograms), els veins locals de cada punt clau és descriuen en un histograma compacte. Emprant únicament estos punts i els seus descriptors associats, un algoritme de cerca un-a-un molt ràpid s'executa sobre cada parell de fragments. Aquest procés fa servir una estratègia de cerca jeràrquica de tres nivells, dirigida per la similitud entre punts clau i que aplica un conjunt de tests de consistència geomètrica sobre els resultats intermedis. Finalment, un algoritme de registre global pren com a dades d'entrada totes les correspondències individuals per generar la reconstrucció final de l'objecte.

Contents

List of Figures

CHAPTER 1

Introduction

Since the dawn of mankind, our ancestors have built man-made objects to assist in everyday life. As a consequence of numerous causes, like earthquakes, floods, wars and many more, these artifacts have been broken apart and, with the passing of time, fragments have been eroded, spread or even lost. In fact, in current archaeological sites, artifacts are rarely found intact. More often, what archaeologists and anthropologists find, are fragments of ancient relics that have to be re-assembled in an attempt to recompose the original artifact.

The study of recovered objects in archaeological sites provides a better understanding of our history and our ancestors life. In fact, ceramic pots (generally referred to as sherds) are often the most important source of information because they allow archaeologists to infer about the society that existed at a given location: chronology, the population's socioeconomic standards... By reconstructing mural paintings and mosaics, lot of information is revealed about the iconography of an age, stylistic and drawing tools developments. In the case of stone tablets and other artifacts that bear text inscriptions, reconstructions that allow epigraphists to read part of the entire text provide important information about the society's organization and the scientific and cultural evolution (e.g., poetry, drama), of ancient civilizations such as Greek, Persian, Egyptian, etc. For all these reasons, reconstruction of broken or torn artifacts of archaeological, historical or cultural importance is an indispensable tool in the hands of researchers in these fields.

1.1 Motivation

Reconstruction of ancient artifacts from fragments found at archaeological sites, is a tedious task that requires many hours of work from the archaeologist and restoration personnel. Historically, this reconstruction process has been manual, occupying a major proportion of the human effort at excavation sites. In fact, since the assembly work is so time-consuming and labor-intensive, reconstruction is not even attempted at countless sites around the world, leaving vast quantities of material unstudied and stored indefinitely.

Advancements in low-cost, high-volume acquisition devices and modern computer systems performance have provided a new tool for archaeologists to face the problem of reconstruction from fragments.

Nowadays, in the field of heritage restoration, 3D acquisition devices and techniques are used primarily for documentation tasks, virtual simulation, multimedia applications and monitoring and control tasks of the conservation status of the recorded material when morphological and/or texture changes happen. However, operating on digital models of fragments can rapidly and systematically consider many thousands of possible fragment alignments and combinations, improving this way the overall performance on the reconstruction stage.

The final goal of these techniques is reducing the amount of candidate matches between fragments, and providing an automatic or semi-automatic tool for archaeologists to recompose the original artifact efficiently. This way, the extra time spent in acquiring computer models is compensated by the time reduction in the matching stage and, additionally, ensures the integrity of the studied fragments and provides a digital database that can be easily shared with the rest of the research community for further studies.

The proposed Ph.D. Thesis dissertation is an extension of the work started in a research project granted by the Generalitat Valenciana to the Instituto de Automática e Informática Industrial (Robotics Group) in 2008. Under the designation “CATALOGARQ: Catalogación, Reconocimiento y Clasificación de Piezas Arqueológicas”. This project proposed a low cost semi-automatic procedure to catalogue mural painting fragments.

Starting from an upper and lower image of each fragment, a basic feature extraction was performed and information introduced by the user was associated. Afterwards, using a pattern matching technique, relative alignments between the upper and lower faces were corrected and an automatic 3D reconstruction of the fragment was generated. Data associated to each piece and extracted features were used to perform a classification of resources and to assist restoration personnel in the reconstruction of the original artifact.

The motivation of the project came from the variety of archaeological sites existing in the Comunitat Valenciana, from where lots of fragments are extracted every year. According to local legislation, these fragments have to be properly classified before being displayed in museums or stored for further interventions. However, archaeology has traditionally been a field with a very artisanal methodology, where Information Technologies have not been applied (or have been applied in a wrong manner). This way, the cataloguing process is generally manual, supported by more or less standardized technical notes, which takes lots of time to the archaeologist and restoration personnel that works in the archaeological sites.

The techniques described in this document continue the work started in 2008 and focus on the re-assembly problem. To do so, close collaboration has been stablished with expert personnel from museums and research groups from the field of restoration. The goal of this collaboration is to produce an efficient method that automatizes manual procedures involved in the re-assembly of archaeological artifacts and to create a closed-form solution that could be applied in the everyday's work of the museums.

As an alternative use of the developed registration algorithms, a different kind of problem has been faced to prove the applicability of the proposed techniques: indoor location in known environments. Self-localization techniques based on observations can be understood as an specific application of surface registration techniques: the observer's location can be inferred after calculating the optimal correspondence between a local observation and a ground-truth map of the known environment.

In this line of research, two approaches have been developed: a fast mobile robot self-localization algorithm for structured indoor environments and a system to allow nuclear inspectors from IAEA (International Atomic Energy Agency) to keep track of their location inside nuclear facilities and to automatically detect changes in the environment. First approach has been done in close collaboration with the robotics group of ai2 (Instituto de Automática e Informática Industrial, Universitat Politècnica de València), and the second one is the result of an internship in the Institute of Transuranium Elements, Joint Research Center (European Comission).

The development of this research is supported by the "Programa de Ayudas de Investigación y Desarrollo (PAID)" of the Universitat Politècnica de València and enclosed in a national research project granted by the Spanish government in 2012 as part of the "Plan Nacional de I+D+i 2008-2011" from the Ministerio de Economía y Competitividad, Project ID: HAR2012-38391-C02-02.

1.2 Objectives

The main goal of this research is providing an automatic computer-based solution to the problem of reassembling archaeological artifacts from fragments. To do so, all stages are considered: from acquisition of B-Rep models using 3D triangulation laser scanners to pairwise matching of fragments to the final reassembly of the original artifact.

Quality of achieved results has to be measured according to two different criteria: correction and performance. This way, an automatic solution has to be proven to find correct correspondences between neighbor fragments in the original artifact, and the overall performance of the system has to be greater than the manual procedure. Otherwise, it is not worth replacing traditional restoration processes.

First challenge to face is generating well-defined 3D B-Rep models from original fragments. Acquisition using 3D triangulation laser scanners is a relatively simple process, well documented and used in many fields (mostly industrial). However, acquiring objects whose surface present reflections and/or refractions make acquisition harder, because these kind of objects violate almost every assumption made in vision algorithms. For industrial solutions, whitening sprays are used to create a thin white opaque layer that allows acquisition. However, these sprays are hard to remove, and additional chemical or physical procedures have to be applied to clean the original object. Considering that archaeological artifacts are unique and fragile objects, rubbing the surface or applying chemical products

to them is not possible. To face this problem, the alternative use of a known product in restoration is proposed: cyclododecane. Thanks to its chemical stability and to the fact that it sublimes at room temperature leaving no residuals, a set of experiments have been developed in order to prove that it can be used as a whitening spray for acquisition of singular artifacts.

Once fragments have been acquired, the variety of topologies makes interesting to distinguish between two kinds of approaches: one to face 3 degrees of freedom problems and another to face 6 degrees of freedom problems.

First one deals with the reassembly from flat fragments, characterized by 2D contours or 3D B-Rep models. These kind of objects are very common in archaeological sites (like frescoes, mosaics or ripped-up documents), and the engravings in their surface provide lots of useful information to archaeologist. In this case, the reduced size of the solution space allows implementing efficient search strategies that ensure global correction, since a complete exploration of the solution space can be performed.

The second kind of problems, that consider 6 degrees of freedom, allow working with full 3D problems, like the re-assembly of sculptures, friezes... In this kind of problems, the combinatory explosion in the solution space make exhaustive approaches prohibitive, so a reduction in the problem size has to be applied. To do so, singular key-points are identified and, using descriptors, their surrounding geometry is characterized in a compact manner. This way, matching is performed using this reduced set of features instead of the whole original object.

In general terms, the re-assembly process is divided in two different contexts: one that faces one-to-one correspondences between fragments, and a more general one that takes these correspondences and tries to reconstruct the original artifact. For both cases, an NP-complete problem has to be faced: in one-to-one comparisons, matches between fragments are usually partial (the whole surface of a fragment rarely matches the whole surface of its neighbor).

Deciding which parts of one surface have a counterpart in the other fragment is a strong NP-complete problem. The same way, for the global reconstruction stage, where the puzzle has to be solved, deciding which potential matches are correct and which are not has also been characterized as a strong NP-complete problem, which becomes harder when the effect of erosion increases the uncertainty of correspondences.

First problem needs to be automatically solved, in order for the proposed system to be helpful to restoration personnel. By applying some simplifications (based on discretization or in feature extraction), an efficient solution has to be achieved. The second one is hardly formalizable and, performing an exhaustive search for the best solution may lead to an extremely slow result. This way, since puzzling is a problem humans can solve very fast if the proper help is provided, the final goal of the proposed research is developing a semi-automatic tool that offers the user potential matches of fragments, leaving him the last decision about which ones are globally consistent or not.

Given the hard computing needs of the proposed approach, and the geometric nature of the involved operations, exploiting the GPU (*Graphics Processors Unit*) computing capabilities has been considered during the implementation of the automatic re-assembly technique.

Additionally to the re-assembly from fragments technique presented in this dissertation and, considering that this research is enclosed in a national research project, results achieved have been applied in the same field to several other applications: an on-line database of scanned fragments has been generated, where users can visualize and download the 3D B-Rep models generated from a web page, an automatic approach for creating tailored packagings for archaeological artifacts has been developed taking advantage of GPUs and the use of 3D printers to create the missing parts of the finally reconstructed artifacts is being developed.

1.3 Document Organization

After this chapter, the document is organized as follows:

Chapter 2: Given that the re-assembly of broken artifacts is a specific application of a more general discipline called Shape Matching, this chapter provides a full background and a classification of the most common techniques in this field.

Chapter 3: Provides an overview to the most important automatic fragment reconstruction techniques, classifying them according to the kind of problems faced: from jigsaw puzzles to global reconstruction techniques.

Chapter 4: Presents the problem of fragment acquisition for 2D and 3D cases, paying special attention to the second one, where more complex situations have to be faced due to the physical limitations of acquisition devices when working with reflective / refractive surfaces.

Chapter 5: Introduces the proposed technique to face 3 degrees of freedom problems, where the reduced solution space is efficiently exploited and, with the support of the GPU, a hierarchical search strategy is presented.

Chapter 6: Introduces the proposed technique to face 6 degrees of freedom problems, where the input size of the problem is efficiently reduced using a feature extraction stage. Then, a fast matcher based on descriptors and geometrical consistency tests is proposed to address the alignment of the fragments. Finally, a global reconstruction algorithm is presented to exploit all the one-to-one alignments in order to produce the final re-assembly.

Chapter 7: Shows the application of surface registration techniques to address the self-localization problem, considering two different scenarios: (1) a resource-limited autonomous robot, where the structured nature of indoor environments is exploited and a sparse efficient registration is performed. (2) A general purpose application to address the self-localization problem for inspection and verification purposes. The goal is developing a technique that allows nuclear inspectors from the IAEA (International Atomic Energy Agency) orienting inside nuclear facilities and detecting changes with respect to previously acquired 3D models. In this case, since no special resource limitations are assumed, a dense approach is used.

Chapter 8: Presents the conclusions of this dissertation.

CHAPTER 2

Background on Shape Matching

Reconstruction of broken artifacts from fragments can be classified as a specific application of a more general discipline called *shape matching* (also referred in the bibliography as "shape correspondence", "shape registration", "shape alignment", or simply "matching"), which can be enclosed into many different fields: Computer Vision, Computer Graphis or Artificial Inteligence and Pattern Recognition. The main difference between the general approach and this particular application is that reconstruction from fragments adds an extra non-penetration constraint, that has to be satisfied for all pairs. Shape matching applications cover a wide range of fields:

Shape registration given a set of scans with partial overlaps, align them to reconstruct the targeted object. If shapes do not change during acquisition it is called rigid registration, otherwise is called non-rigid registration.

Shape interpolation morph one shape into another satisfying some aesthetic conditions.

Recognition and retrieval computing a correspondence between a query shape and the models contained in a dataset, which is one of the challenges in computer vision.

Statistical shape modeling generate models that describe the valid variations in the appearance and the size of a given shape. Very common in anatomical applications.

Change detection track changes on a shape over time.

This chapter gives a background on current shape matching techniques, and classifies them according to a set of different criterions.

2.1 Classification

Finding the best match between a set of shapes can be formulated in a general way as an optimization problem: "Given some input data (shapes) and a cost function, find the mapping/transformation between their elements that maximizes the quality of the match". When two elements of different shapes are related, we say that they match to each other. This relation may vary depending on the specific problem, so sometimes pairs are stablished as one-to-one relationships, whilst other times are one-to-many or many-to-many.

The proposed classification criteria followed in this chapter derives from the previous general problem statement:

- Input data: *"How are the input shapes represented?"*
- Resulting match: *"What kind of result is expected after calculating the mapping/ transformation function, and what properties does it possess?"*
- Cost function: *"How good is a given correspondence?"*
- Search strategy: *"How do we get the best correspondence?"*

Next sections in this chapter will expand this four classification categories. This way, Section 2.2 details the kinds of input data that shape registration algorithms work with. Section 2.3 covers the expected resulting matches. Section 2.4 deals with the different ways to evaluate the quality of a match and, finally, Section 2.5 analyses the most common search strategies.

2.2 Input data

2.2.1 Dimensionality

Classifying shape matching techniques according to their input data allows filtering them based on a dimensionality criteria. Normally, the most common cases are 2D and 3D domains. However, due to recent technological advances, an application that is gaining increasing importance is the reconstruction of shapes acquired over time, while moving and deforming.

When considering time, input data consists on a set of scans acquired on a fixed period and registration has to find correspondences in both objects and motion sequences [175] [91] [147] [44] [57] [165]. With this extra dimension new challenges appear due to big amounts of missing data that can present in each frame (occlusion problems) [130], or dataset captured without a fixed acquisition period [29] [194]. However, adding a temporal dimension might simplify the search process if kinematic constraints are assumed [111].

Datasets used for shape matching are obtained from multiple sources, depending on their dimensionality. 2D datasets are fast to acquire and do not present occlusion problems with flat scan beds or cameras. 3D datasets require an extra processing step due to topological problems, generally induced by self-occlusions. Common acquisition devices are 3D scanners based on laser triangulation, structured light, physical contact... but also more complex devices like magnetic resonance images (MRI) or computed tomographies (CT) are used, generally in medical applications [135] [156] [134].

This way, classifying shape matching techniques according to the input data dimensionality presents four main groups: 2D problems, 3D problems, and their variants including time (2D + time and 3D + time). However, in this chapter, time techniques will be ignored for being out of the scope of this document.

2.2.2 Shape representation

The way shapes are represented is a key aspect for designing a shape matching algorithm: intrinsic properties of the representation may accelerate computation considerably, and allow a technique to converge faster and more accurately to the global solution of the problem faced.

Shape representation and search strategies are very close: classic registration methods such as RANSAC [49], geometric hashing [184], pose clustering [117], and alignment [73] typically work with point sets. Recent methods based on articulated shapes [28], isometric surfaces [22] [95] deformation [72] [192] and graphic applications based on template matching [91] [4] [128] use surfaces as most common representation. Time-varying surfaces are typical on motion reconstruction of deforming surfaces [111] [175] [147] [194] and, finally, skeletons are a more general name for shape representations such as Reeb graphs, medial axis or M-reps [150] [37] [16] [9].

Instead of working directly on the original representation of the datasets, it is a very common practice to reduce the size of the input data by extracting some representative key points (features) and computing descriptors for these points. Descriptors are normally scalar values, or vectors of scalars, that capture some properties of the surface around the interest point. This way, similarity between datasets can be computed indirectly by comparing the similarity between their associated descriptors. Ideally, if two descriptors are similar, their corresponding points should also be similar. Alternatively, the descriptors can be used to guide the search for initial solutions, while the final verification of the correspondence quality is performed with the original dataset. Figure **??** graphically illustrates this concept.

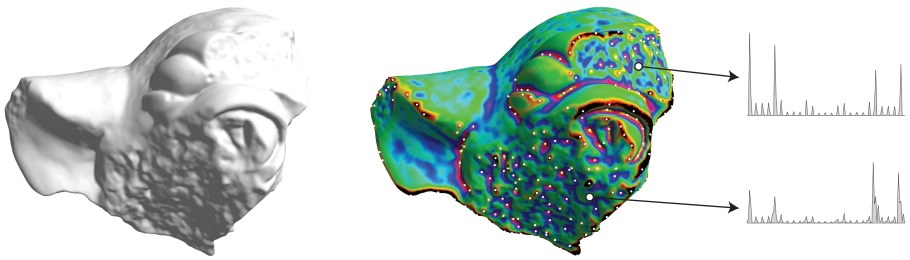

Figure 2.1: *Feature selection and descriptor computation. (left) original input model. (right) features extracted, marked with white points, and two histogram-based descriptors associated with the marked features.*

2.2.2.1 Feature detectors

Feature-based approaches became very popular in computer vision and image analysis applications, due to the works like [101] [154] [109]. In these approaches, an image was described as a collection of local features from a given vocabulary, resulting in a representation referred to as a *bag of features*. In shape analysis, feature-based approaches have been introduced more recently and are gaining popularity in registration applications.

Features can be classified as *global* or *local*. Global features are very common in the field of shape retrieval (given an input shape, find the most similar ones in a database) and object recognition. They focus on describing the whole shape with a unique feature that captures its topological properties. On the other side, local features attempt to find patterns in the shape which differs from its immediate neighborhood. They are normally associated with changes of a shape property or several properties simultaneously, although they are not necessarily localized exactly on this change. In this section, we focus on local features for being more related to the scope of this Thesis project.

Local features can be points, but also regions. Typically, some measurements are taken from a region centered on a local feature and converted into descriptors. It is the task of the feature detector (also called *extractor* in the bibliography) to identify these points/regions.

Local features typically have a spatial extent. Ideally, one would like such local features to correspond to semantically meaningful object parts. In practice, however, this is unfeasible, as this would require high-level interpretation of the scene content, which is not available at this early stage. Instead, detectors select local features directly based on the underlying geometric patterns.

Good features should have the following properties:

1. Repeatability: given two shapes of the same object, taken under different conditions, a high percentage of the features detected on the part visible in both shapes should be found in both shapes. This is arguably the most important property of all, can be achieved in two different ways: either by invariance or by robustness.

2. Distinctiveness/informativeness: patterns underlying the detected features should show a lot of variation, such that features can be distinguished and matched.

3. Locality: features should be local in order to reduce the probability of occlusion and facilitate the matching of shapes acquired under different conditions. Locality and distinctiveness are competing properties and cannot be fulfilled simultaneously: the more local a feature, the less information is available in the underlying pattern and the harder it becomes to match it correctly, especially in database applications where there are many candidate features to match to.

4. Quantity: the number of detected features should be sufficiently large, such that a reasonable number of features are detected even on small objects. However, the optimal number of features depends on the application. Ideally, the number of detected features should be adaptable over a large range by a simple and intuitive threshold. The density of features should reflect the information content of the image to provide a compact image representation.

5. Efficiency: the detection of features in a new shape should satisfy temporary requirements of the application.

A very common way to measure the quality of a feature detector is based on the repeatability of the extracted features. Assuming for each transformed shape Y in a dataset the ground truth dense correspondence to the null shape X is given by pairs of points $\mathcal{C}_0(X,Y) = \{(x'_k, y_k)\}_{k=1}^{|Y|}$, a feature point $y_k \in \mathcal{F}(Y)$ is said to be repeatable if a geodesic ball of radius ρ around the corresponding point $x'_k : (x'_k, y_k) \in \mathcal{C}_0(X,Y)$ contains a detected feature point $x_j \in \mathcal{F}(X)$. Repeatable features are

$$\mathcal{F}_\rho(Y) \quad = \{y_k \in \mathcal{F}(Y) : \mathcal{F}(X) \cap B_\rho(x'_k) \neq \emptyset, (x'_k, y_k) \in \mathcal{C}_0(X,Y)\} \tag{2.1}$$

where $B_\rho(x'_k) = \{x \in X : d_X(x, x'_k) \leq \rho\}$ and d_X denotes the geodesic distance function in X.

Similarly, for region detectors, a region $Y_l \in \mathcal{F}(Y)$ is repeatable if the corresponding region $X'_l \subset X$ has overlap larger than ρ,

$$\mathcal{F}_\rho(Y) \quad = \{Y_l \in \mathcal{F}(Y) : |X'_l \cap X_l|/|X'_l \cup X_l| \geq \rho\} \tag{2.2}$$

The repeatability of a feature detector is defined as the percentage $|\mathcal{F}_\rho|/|\mathcal{F}(Y)|$ of features that are repeatable, the definition being dependent of whether a point or region descriptor is used [21].

Some of the most common 3D feature detectors are:

Harris 3D [153] extends the 2D Harris corner detector [65] to work with 3D data. The algorithm suggests to determine a neighborhood around a vertex. Next, this neighborhood is used to fit a quadratic patch which is considered as an image. After applying a gaussian smoothing, derivatives are calculated which are used to calculate the Harris response for each vertex.

Mesh-DoG [186] considers the general setting of 2D manifolds $\mathcal{M}$ embedded in $\mathbb{R}^3$ endowed in with a scalar function $f : \mathcal{M} \to \mathbb{R}$, such as color or curvature. This represents a generalization of 2-D images, that can be viewed as a uniformly sampled square grid with vertices of valence 4. Operators, such as the gradient and the convolution are defined in this context. A scale-space representation of the scalar function f is build using iterative convolutions with a Gaussian kernel. Feature detection consists of two steps. Firstly, the extrema of the function's Laplacian (approximated by taking the difference between adjacent scales - Difference of Gaussian) are found across scales, followed by non-maximum suppression using a 1-ring neighborhood both spatially and across adjacent scales. Secondly, the detected extrema are thresholded (400 points). Mean and Gaussian curvature computed using [108] were the scalar functions used for current tests.

Mesh SIFT [104] detects scale space extrema as local feature locations. First, a scale space is constructed containing smoothed versions of the input mesh, which are obtained by subsequent convolutions of the mesh with a binomial filter. Next, for the detection of salient points in the scale space, the mean curvature H (Mesh SIFT-H) and the principal coordinates in curvature space KK (Mesh SIFT-KK), which are minimal and maximal curvature, are computed for each vertex and at each scale in the scale space (H_i and KK_i). Note that the mesh is smoothed and not the function on the mesh (H or KK). Scale space extrema in scale spaces of differences between subsequent scales ($dH_i = H_{i+1} - H_i$ for Mesh SIFT-H and $dKK_i = KK_{i+1} - KK_i$ for Mesh SIFT-KK) are finally selected as local feature locations.

Mesh-Scale DoG [41] following the work described in [186] that presented a Difference of Gaussians based feature points detector for mesh objects, a Gaussian filter on the mesh geometry is defined, and a set of filtered meshes are computed. Consecutive octaves are subtracted to compute the DoG function, and define the local maxima (both in location and scale) as feature points at that point and scale.

Some other interesting interesting approaches to choose the set of descriptors that gives the best correspondence results are derived from the machine learning discipline, and can be consulted in [64] [177] [79] [171].

2.2.2.2 Descriptors

Once features have been selected, a unique signature has to be computed for each one of them in order to characterize, as best as possible, the underlaying properties of the surface they enclose. This signature is what descriptors are.

In order to measure how similar two shapes are, distances between pairs of descriptors are computed using a dissimilarity measure (the term "similarity" is often used as a synonym in the bibliography, but dissimilarity corresponds better to the notion of distance: small distance means small dissimilarity).

A dissimilarity measure can be formalized by a function defined on pairs of descriptors indicating the degree of their resemblance. In a more formal way, a dissimilarity measure d on a set S is a non-negative valued function $d : S \times S \rightarrow \mathbb{R}^+ \cup \{0\}$. Function d may have some or all of the following properties:

1. Identity: $\forall x \in S, d(x, x) = 0$.
2. Positivity: $\forall x \neq y \in S, d(x, y) > 0$.
3. Symmetry: $\forall x, y \in S, d(x, y) = d(y, x)$.
4. Triangle inequality: $\forall x, y, z \in S, d(x, z) \leq d(x, y) + d(y, z)$.
5. Transformation invariance: for a chosen transformation group T, $\forall x, y \in S, t \in T, d(t(x), t(y)) = d(x, y)$.

The identity property says that a shape is completely similar to itself, while the positivity property claims that different shapes are never completely similar. These properties are sometimes too strong for high-level shape descriptors, and often not satisfied. Symmetry is not always wanted. Actually, it is a common situation that the dissimilarity between x and y is different than the opposite way. Also, triangle inequality is not always satisfied when partial matching situations happen. The only property that has to be completely satisfied is the transformation invariance: comparisons and shape descriptors extraction have to be independent of the location, orientation and (sometimes) scale of the shapes compared.

Some other desirable properties for shape descriptors are:

1. Discriminative power: a descriptor should capture properties that discriminate objects, or parts of objects, well.
2. Partial matching: in contrast to global matching, partial matching consists on finding a shape of which a part is similar to a part of other shape. This is specially common when registering partial views obtained with a 3D scanner.

3. Robustness and sensitivity: it is desirable that a shape descriptor is insensitive to noise and robust agains arbitrary topological degeneracies. This means that small changes in a shape should result in small changes in its associated descriptors. The same way, big changes in the shape should result in big changes in its descriptors. Otherwise, the descriptor is said to be not sensitive, which leads to poor discriminative abilities.

4. Pose normalization: in the (common) absence of prior knowledge, shapes are arbitrary oriented, positioned and, sometimes, scaled in space. Descriptors have to be invariant to these transformations or, in case they are not, shapes have to be normalized to be compared. Common normalization techniques are translating the center of mass of a shape to the origin of coordinates, using Principal Component Analysis (PCA) or more advanced techniques, like the one described in [31], for orientation, and scaling the shape so the average distance of its points to the center of mass is constant.

5. Efficiency: computing shape descriptors has to be fast enough to satisfy temporary requirements of a given application.

A variety of descriptors have been proposed in the literature. In general terms, they can be classified according to the type of dataset they work with. This way, shape context approaches [10] [87], describe the coarse arrangement of the shape with respect to a point inside or on the boundary of the shape using unoriented point datasets. When considering oriented point datasets, spin images [78] and multi-scale features [93] are very common in the literature. Finally, for describing local properties of surfaces, some typical approaches are curvature maps [59], integral invariants [105] [60], spherical harmonics and wavelets [53] [36] [82], salient geometric features [56], part-aware metrics [98] or heat kernel signatures [161].

More in detail, shape descriptors can be classified according to the dimensionality of the input data, and their philosophy. For 2D domains, the most common techniques can be split into three main categories: contour based descriptors, region based descriptors and hybrid 2D descriptors.

Contour based descriptors Contour based descriptors only consider the boundary of the shape ignoring the information contained in the interior [152]. These descriptors are very efficient at filtering out the results based on the boundary points because of their low computation complexity. However, they are not good at handling image noise and thus not accurate.

1. *Fourier Descriptor (FD)*: a Fourier descriptor represents the shape obtained after applying a Fourier transform on the coefficients of the shape signature (any 1D function used to represent 2D shapes or boundaries). The most frequently used shape signatures are centroid distance, complex coordinates, curvature function and cumulative angular function. Among them, Fourier descriptor method performs better using centroid distances [191]. Main advantages of Fourier descriptors are that they are simple to compute, simple to normalize, capture both local and global features and are insensitive to noise.

2. *Wavelet Descriptor (WD)*: Wavelet descriptor was proposed in [34], where the authors used wavelet transforms to describe the shape of planar closed curves. It is a multi-resolution approach which decomposes the shape into several components in multiple scales. In higher resolutions, components contain the global information whilst, in lower resolutions, information is more detailed in a local area. Their main advantages are that they are insensitive to noise, invariant, unique ad stable against boundary variations. They have been recently used in [83] in combination of Fourier descriptors.

3. *Curvature Scale Space (CSS)*: originally introduced in [8] this is one of the most widely used in content based image retrieval. The key idea below this algorithm consists on dividing the shape into convex and concave segments by identifying a set of inflection points. A two dimensional vector is associated with each inflection point, expressed as (s, l), where s is the amount of smoothing applied until to there are no zero curvature points (so the contour becomes convex) and l is the position of the point on the contour curve. The original algorithm is fully detailed in [14] [19], and improved in [141].

4. *Shape Context Descriptor (SCD)*: introduced in [11] shape context finds the correspondence between two shapes and finds out the dissimilarity measure between them. To find the correspondence, N points are sampled from the contour of the shape and a reference point is fixed. The points are sampled using an edge detector algorithm. Then a set of vectors are computed originating from the reference point to all the other sampled points. The shape context for each point is defined as a histogram of relative polar coordinates of the remaining sampled points. Shape context is a very robust shaped descriptor which is highly discriminative. It is also transformation invariant, robust against shape variations, and has few outliers [110].

Region based descriptors Region based descriptors take into account the boundary as well as the internal information of the image and are more robust against noise and other shape variations than countour based descriptors.

1. *Zernike moments descriptor*: this descriptor was introduced in [164]. It is one of the most commonly used region based descriptors and has been improved since it was first released. Zernike moments are continuous orthogonal moments derived from Zernike polynomials [190]. Main advantages of this descriptor are rotation invariance, robustness against small changes in shape, that it is insensitive to noise and highly expressive [27]. On the counterpart, its main disadvantages are related to the coordinate space normalization (image coordinate space must be transformed) and that the discrete approximation of continuous integrals lead to errors in the computations.

2. *Scale Invariant Feature Transform (SIFT)*: originally introduced in [100], this descriptor is based on the work published on [144] that stated that efficient object recognition could be achieved by using local image descriptors that could be sampled at a large number of repeatable locations. This way, the SIFT algorithm converts the image into a huge collection of location feature vectors invariant to scale, rotation and translation. In the first step of the algorithm,

extracts the scale invariant features in the image using the staged feature approach. These vectors are called SIFT keys, and used for indexing and identifying candidate object models by using a nearest neighbor search algorithm. Main advantages of SIFT descriptors are invariance to scale, rotation and translation, partially invariance to illumination changes, robust against occlusions and object degeneracies and insensitivity to noise.

3. *Angular Radial Transform (ART)*: originally proposed in [84] is a popular RBD which is used in the MPEG-7 standard [19]. It has been described as "the orthogonal unitary transform defined on a unit disk that consists of the complete orthogonal sinusoidal bases functions in polar coordinates" in [19].

Hybrid 2D shape descriptors By combining the two previous kinds of descriptors, some new techniques have been introduced. In [178] a two-component solution was proposed in which centroid distances, contour curvature and Zernike moments were selected as shape features, while a two-component strategy was applied in feature matching. In [152] a hybrid approach was proposed which combines Fourier (to extract local features) descriptor with ART (to extract global features). The authors performed experiments with Fourier descriptors + ART and Fourier descriptors + Zernike moments and showed that their techniques performed better than the two-component solution.

When working with 3D input data, the main categories are view based descriptors, histogram based descriptors, transform based descriptors, graph based descriptors and hybrid 3D descriptors. These descriptors have been extensively used by researchers since the 90's in 3D search engines and sketch based modeling systems.

View based descriptors View based descriptors use silhouette, greyscale or depth-buffer images extracted from multiple views of 3D objects. Their main advantages are that they do not require the explicit virtual model information, which makes the method robust to real practical applications, and that the view-based 3D model analysis methods can be benefited from existing image processing technologies which have been studied from several decades [97].

1. *Compatc Multi-View Descriptor (CMVD)*: this method was introduced in [39] [40]. It accepts multi-modal queries (3D images, sketches and 3D models). In a first step, a pose is estimated using PCA and visual contact area methods. Then, 24 sets of 3D image views are generated from 18 different viewpoints of a 32-hedron surrounding the 3D object. Two type of views are extracted: binary views (silhouettes) and depth views. After that, 2D descriptors are calculated, and 3D matching is achieved by computing the total dissimilarity of the 2D images generated from the 3D objects.
2. *Light Field Descriptor (LFD)*: Introduced in [30] is based on the idea that two 3D objects are similar if they look similar from all viewing angles. To compare objects, 10 silhouette images are taken from viewing angles distributed in a dodecahedron. Zernike moments and Fourier descriptors are the used to extract features, and dissimilarity is calculated from rotating the viewing spheres of one light field descriptor relative to the other light field descriptor. In [55] LFD has proven to perform better than Spherical Harmonics Descriptors.

Histogram based descriptors Histogram based descriptors work over the feature domain by collecting numerical values in bins that represent the features of a 3D shape.

1. *Point Feature Histograms (PFH)*: Point Feature Histograms (PFH) [139] descriptor's goal is to generalize both the surface normals and the curvature estimates. Given two points, p and q, a fixed reference frame consisting of the three unit vectors (u, v, w) is built centered on p as follows:

   - The vector u is the surface normal at p.
   - $v = u \times \frac{p-q}{d}$.
   - $w = u \times v$, where $d = \|p - q\|_2$.

   Using this reference frame, the difference between the normals at p (n_p) and q (n_q), can be represented by:

   - $\alpha = acos(v \cdot n_q)$.
   - $\phi = acos(\frac{u \cdot (p-q)}{d})$
   - $\theta = atan(w \cdot n_p, u \cdot n_p)$

   The angles α, ϕ, θ and the distance d are computed for all pairs in the k-neighborhood of point p. In fact, usually the distance d is dropped as it changes with the viewpoint, keeping only the 3 angles. These are binned into an 125-bin histogram by considering that each of them can fall into 5 distinct bins, and the final histogram encodes in each bin a unique combination of the distinct values for each of the angles.

2. *Fast Point Feature Histograms (FPFH)*: the Fast Point Feature Histograms [138] are a simplification of the PFH descriptor above that reduce the computational complexity of the PPF algorithm from $\mathcal{O}(nk^2)$ to $\mathcal{O}(nk)$. The first step is to compute the histogram of the three angles between a point p and its k-nearest neighbors (not between all pairs of neighbors) in the same way as in PPF. This produces the Simplified Point Feature Histogram (SPFH). Then, for each point p, the values of the SPFH of its k neighbors are weight by their distance $w_i = d$ to p to produce the FPFH at p:

   $FPFH(p) = SPFH(p) + 1/k \sum_{i=1}^{k} SPFH(i)/w_i$

   The three angles are binned into 1-bin histograms that are concatenated into a single 33-bin FPFH descriptor.

3. *Signature of Histograms of Orientations (SHOT)*: the SHOT descriptor [168] is based on obtaining a repeatable local reference frame using the eigenvalue decomposition around an input point. Given this reference frame, a spherical grid centered on the point divides the neighborhood so that in each grid bin a weighted histogram of normals is obtained. The descriptor concatenates all such histograms into the final signature. It uses 9 values to encode the reference frame and the authors propose the use of 11 shape bins and 32 divisions of the spherical grid, which gives an additional 352 values. The descriptor is normalized to sum 1.

4. *3D Shape Context*: this descriptor is was proposed in [52]. It uses a spherical grid on each of the features. The north pole of the grid is oriented as the

surface normal at the feature and the grid consists of bins along the radial, azimuth and elevation dimensions. The divisions along the radial dimension are logarithmically spaced. The number of bins can be set by the user. Each bin makes a weighted count of the number of points that fall into it. The weights used are inversely proportional to the bin volume and the local point density. Since the axes tangent to the surface are placed randomly, there is the need to extract as many versions of this descriptor per database object as there are divisions along the azimuth direction. All these versions of the descriptor need to be tried on a test cloud to find an object match.

5. *Unique Shape Context*: the Unique Shape Context [167] was proposed as an upgrade of the 3D Shape Context with the goal of avoiding the need to obtain as many versions of the descriptor as the number of azimuth bins. Consider a point p with a spherical neighborhood of radius R. A weighted covariance matrix M of the points in the neighborhood is computed as:
$M = \frac{1}{Z} \sum_{i=d_i \leq R} (R - d_i)(p_i - p)(p_i - p)^T$
where p_i is a point in the spherical neighborhood, $d_i = \|p_i - p\|_2$ and $Z = \sum_{i=d_i \leq R}(R - d_i)$. The eigenvector decomposition of M is used to obtain the 3 unit vectors of the local reference frame. The sign of the eigenvectors with the biggest and smallest eigenvalues is changed so that it is coherent with the majority of the vectors they represents. The sign of the third eigenvector is obtained from the other two considering that they must form an orthonormal base. The eigenvector with the smallest eigenvalue gives the normal direction. Apart from this reference frame determination process, the Unique Shape Context descriptor is obtained like the 3D Shape Context.

6. *Rotation Invariant Feature Transform (RIFT)*: the RIFT descriptor [89] was developed to generalize the SIFT descriptor [101]. A circular normalized patch is built at each input point. The circular patch is divided into 4 rings of equal width. For each ring, a histogram of gradient orientations with 8 bins is computed, thus producing a 32 value descriptor for each input point. The orientations of this histogram are obtained with respect to the radial outward direction at each point.

7. *Viewpoint Feature Histogram*: the Viewpoint Feature Histogram (VFH) [140] adds viewpoint variance to the above FPFH by using the viewpoint vector direction. It also produces only one descriptor for the input point cloud (it is a global descriptor). The process is the following:

   - Find the input cloud centroid, c.
   - For each point p in the cloud, build the the local reference frame (u, v, w) using $u = n$, $v = (p - c) \times u$, $w = u \times v$
   - Find the angles (α, ϕ, θ) as in the PFH, using this reference frame.

   Each of the three angles is binned into a 45-bin histogram. The angle $\beta = acos(n_p \cdot c/\|c\|)$ that the central viewpoint direction translated to each normal makes with each point's normal is also encoded in a 128-bin histogram.

8. *Clustered Viewpoint Feature Histogram*: the Clustered Viewpoint Feature Histogram (CVFH) descriptor for a given point cloud dataset containing normals, was proposed in [2]. Stable regions are obtained by first removing the points with high curvature and then applying a smooth region growing algorithm. The CVFH is obtained using the following steps:

   - Determine the set S of stable regions.
   - For each $s_i \in S$, find the centroid (c) and its normal (n_c).
   - Build a local reference frame (u_i, v_i, w_i) like in the VHF but using c and n_c instead of the centroid and respective normal for the whole input cloud.
   - Find the histograms of the angles $(\alpha, \phi, \theta, \beta)$ as in VHF (the first 3 coded as 45-bin histograms and β coded as a 128-bin histogram).
   - Find the Shape distribution Component (SDC) as $SDC = \frac{(c-p_i)^2}{max\{(c-p_i)^2\}}, i = 1, ..., |S|$.

   The CVFH is given by the concatenated histograms $(\alpha, \phi, \theta, SDC, \beta)$ which is a 308-bin histogram.

9. *Ensemble of Shape Functions*: this is a global shape descriptor proposed in [182] consisting of 10 concatenated 64-bin histograms resulting in a single 640 value histogram for a given input point cloud. It is based on three shape functions [119] describing distance (D2: distance between two randomly selected points), angle (A3: the angle enclosed by two lines created from 3 randomly selected points) and area (D3: area of the triangle formed by 3 randomly selected points) distributions. They also use an idea from [75] that is to classify each of the values into three classes based on where the connecting lines between points reside: on the object surface, off the surface and mixed (partly on and off). Before finding these distances the cloud is approximated by a voxel grid of side 64 and the match is done using L1-distance. The authors propose to optimize this descriptor by learning weights for the sub-histograms.

10. *3D Shape Spectrum Descriptor (3D SSD)*: a 3D shape spectrum descriptor is a shape descriptor that combines a shape index distributed over the entire mesh [187]. The index is defined as a local geometric feature of the shape, expressed as the angular coordinate of the polar representation of the principal curvature vector. This descriptor locally characterizes free form discrete polygon 3D meshes. Its main characteristics are generality, invariance to scale and Euclidean transforms, and robustness. Given that this descriptor is a simple local feature representation, it should be combined with some globe representation schemes.

11. *Generalized Shape Distributions (GSD)*: originally proposed in [99] this descriptor is commonly used in shape retrieval applications. GSD is based on local and global shape signatures / descriptors of a 3D model. Before generating the histogram, first steps involve the generation of a dictionary of local shape descriptors using spin images. A set of points are sampled on the surface of the shape and accumulated to create spin images. These images are then clustered into 1500 clusters using k-means, and each spin image is assigned an index base on the index of its nearest cluster. After this, a 3D histogram is created.

First dimension stores Euclidean distance of the 2 point pairs. while the other two dimensions store the index value of the two points. In [120] it has been proven the technique to be more accurate and efficient than shape distributions and bag of features descriptors.

12. *Bag-of-Features Histogram (BoF)*: this descriptor consists on accumulating the visual features of a 3D model in a histogram where thousands of visual features are extracted from range images. This technique has proved to be robust against articulated or non-rigid 3D models.

Transform based descriptors The theoretical foundations of transform based descriptors are in classical processing such as spherical harmonics and Fourier transform. Usually, all techniques of this kind have a first step for pose normalization using principal component analysis.

1. *Spherical Harmonics Descriptor (SHD)*: first approach using spherical harmonics to describe a 3D shape was [55]. With this descriptor, the 3D model is first binary voxelized and then, the voxel is placed under concentric spheres and decomposed into spherical functions. Next, a set of harmonic functions are computed from each concentric sphere and each one is represented as a histogram called the spherical signature. By combining these signatures a rotation invariant 3D shape descriptor is generated. Fig. **??** illustrates this process. Spherical harmonics descriptors are compared using Euclidean distances.
2. *PCA Spherical Harmonics Transform*: there has been a debate around whether to use PCA for pose normalization or not. In [173] [172] another spherical based shape descriptor was proposed that used PCA as its pose estimation step. This work differs from [55] in a way that this descriptor involves a generalized PCA step for pose estimation not only considering the vectors and coordinate axes, but also all the points on the mesh with equal weights. According to [173] this descriptor is slightly more expensive, but more accurate than the original approach.
3. *Spherical Trace Transform Descriptor (STTD)*: originally proposed in [188] this descriptor is an extension of the 'Trace Transform'. STTD does not employ PCA as its preliminary step but it uses rotation invariant spherical functions to produce a completely rotation invariant shape descriptor. First step is to achieve translation and scaling normalization by placing the 3D model inside a bounding cube and voxelizing. Then a set of initial 2D functions are applied to the model creating a set of concentric spheres. For similarity matching, weights are assigned to each descriptor.

Graph based descriptors This kind of descriptors represent the topology of a 3D shape in the form of a graph or a tree structure. These descriptors are easy to compute. However they are not computationally efficient. One of their best advantages is that they allow representation at multiple levels of detail and facilitate matching of local geometry.

1. *Skeletal Graph Descriptors*: the concept of skeletons was proposed in [18]. The main idea of these descriptors is to use a skeletal graph of a shape as its

descriptor. A skeleton in 2D is a medial axis, while in 3D is the medial surface. Several methods have been proposed to perform skeletonization such as distance transform [20], thinning [88], Voronoi-based methods [115] or curve skeletonization [47]. The skeletal graph stores the various entities obtained after skeletonization in a graph data structure. Advantage of these methods is that they are topology preserving. Hence, the ca be used for subgraph isomorphism at a ver low computational cost. Additionally, local part attributes can be stored for a more accurate comparison.

2. *Reeb Graph Descriptors*: defined by Reeb in [133], a Reeb graph is determined using a continuous scalar function on an object. Three types of scalar functions have been used: height function, curvature function, and geodesic distance. Geodesic distance has been used in many applications because it provides invariance against rotation and robustness against noise and small perturbations. The function is integrated over the whole body to make it invariant to the starting point and is also normalized to achieve scale invariance.

Hybrid 3D shape descriptors Hybrid approaches are combinations of the previous ones that improve, in some aspect, the quality of the 3D shape analysis.

1. *CMVD + STT*: in [40] a Compact Multi-View Descriptor was combined with Spherical Trace Transform, achieving better results than all the algorithms proposed before. This hybrid algorithm has been compared with Light Field Descriptors, SIFT + bag of features and Depth-Buffer + Silhouette + REXT, providing better precision-recall results.
2. *SIFT + Bag of Features (BF-SIFT)*: proposed in [116], this hybrid approach is based on extracting local visual features of a 3D model using SIFT algorithm and efficiently integrating the in a histogram using the Bag of Features approach. In this algorithm, several 2D range images are obtained from the 3D model. Then, SIFT algorithm is used to extract local features. Each feature is a vector quantized using a visual codebook. K-means learning is used to cluster the local features into a bag of visual words. Then a histogram is generated using the frequencies of visual words, which acts as the feature vector for the 3D model. Some advantages of BF-SIFT are 1) suitable for articulated models, 2) high discriminative power, 3) suitable for 2D image and sketch based queries, and 4) effective for partial matching.
3. *Depth-Buffer + Spherical Harmonics*: originally proposed in [121] this descriptor uses a depth buffer algorithm for extracting 2D features and spherical harmonics for encoding 3D features. Pose is normalized by using two methods named CPCA and NPCA. Performance achieved by this descriptor has been proven to be superior agains LFD.

2.3 Output match

Correspondence between shapes can be represented in different manners and can posses different properties that are exploited in the search strategy. This way, for example, shape retrieval approaches expect a query as input data (a 3D model, a sketch, a picture...), and search for similar objects. The kind of result expected here is only a correspondence (without a transformation), typically calculated on a global context. On the other hand, scan registration approaches take as input data partial views of the same scene, and return the transformation that minimizes the error on the overlapping area. Here registration is performed using partial correspondences (only a fraction of two point clouds are overlapping), and a dense correspondence is typically used (lots of points in both point clouds are used to perform the final registration).

This section characterizes shape registration techniques according to the properties and representations of the resulting output match.

2.3.1 Correspondence representation

A correspondence can be represented as a transformation applied to the shapes (i.e. registration of scanned point clouds) or simply as a relation between elements of the dataset (i.e. shape retrieval approaches).

When looking for a transformation, one of the most distinguishing factors is the type of transformations that are considered. These transformations can be ordered by increasing the number of degrees of freedom: translation, rigid transformation (translations and rotations), similarity transformation (includes isotropic scaling), affine transformation (adds shearing), and nonlinear deformation (includes nonlinear transformations):

Rigid transformations preserve distances between points, and are composed by translations, rotations and reflections. These set of transformations are very common when dealing with problems such as scan registration [137] [60].

Similarity transformations when incorporating the possibility of uniform scaling to rigid transformations, more complex problems can be faced, like matching patterns to limited portions of larger datasets [129].

Affine transformations if affine transformations are considered, the possibility of shearing is included in the search space. This can be used at a global [76] [1] or local [4] [158] level.

Non-rigid transformations in this case, it might be necessary to allow the shape elements to move freely in order to match with the corresponding dataset, which can be seen as assigning a nonlinear deformable transformation to each element [128].

It also should be distinguished whether the transformation is applied to the whole shape (a global transformation) or whether it is applied in a local manner (elements of the shape). Generally, the global case is in the domain of rigid alignments, whilst the local one is in the non-rigid alignments domain.

When looking only for a correspondence, the relation between pairs can be bijective (one-to-one mapping), injective (every element of one shape must be related to one or

multiple elements of the other shape) or full (many-to-many). Even more, restrictions one-to-one or one-to-many can be required for only a subset of elements.

Some methods allow the user to select which type of mapping is desired [90] where the final correspondence is obtained by filtering an initial result according to the mapping constraints. Other approaches build their search strategy by assuming that the focus of interest is only on a specific type of mapping [103] [13].

Correspondences can also be characterized by whether pairwise assignments are weighted based on their confidence. These weights can be discrete (either an assignment is part of the correspondence or not) or fuzzy (there is a degree of confidence). The type of confidence measure that is available depends on the correspondence algorithm. There are probabilistic approaches that return probabilistic weights [189], algorithms based on relaxation to the continuous domain that also return confidence weights attached to each assignment or methods [35] [90] that formulate the problem in terms of integer optimization and return binary weights [103] [13].

2.3.2 Full and partial correspondences

Some techniques are only suitable for contexts in which the whole shape is considered, while others can also compute partial correspondences. The partial case is more generic than the full one, so partial correspondence algorithms can be applied to full correspondence problems, but not necessarily the opposite way. In fact, it is considered that partial algorithms are a very important specialization of shape correspondence.

The problem of partial correspondence can be defined as finding a subset of shape elements for which a meaningful correspondence can be computed. This problem can be divided in two main tasks: finding the optimal subset of elements that match consistently, and finding the correspondence between these elements.

One way to find the best subset of elements is to evaluate the objective function looking for sharp increases in the alignment error, that happens when an outlier point is added to the set of matched points [60] [192]. Also, an estimation on the number of outliers can be provided to the optimization, limiting the number of points of the computed correspondence [103] [13]. This estimation can be derived from the data itself [118].

Another way to find the best subset of elements is to use a strategy based on voting [95] [9], where a set of candidate correspondences is computed and votes are cast on the pairwise assignment that constitute each candidate. When this voting ends, inlier assignments are easily distinguishable from outlier assignments because of their quantity of votes. This procedure acts as a group reinforcement, where only assignments are voted if they can be part of a consistent correspondence.

Partial correspondence can also be treated like a problem of matching two graphs. Feature points can be represented as nodes, connected with an edge proportionally weighted to some geometric quantity. Then, partial matching becomes the problem of subgraph isomorphism, known as a NP-complete problem. Given that different heuristics have been proposed to address this problem, they can be applied for shape matching as well. In [146] and [16] skeleton matching was faced using the notion of matching two graphs by finding a set of operations that transform one graph into the other by merging nodes [114].

2.3.3 Dense and Sparse correspondences

Sparse correspondences are characterized by considering a very reduced amount of elements. Their main advantage is that the complexity of the computation in both, time and space, is reduced considerably.

Some techniques are designed considering this factor, such as search-based methods described in [60] [192]. Despite their associated search space is exponential, these techniques can be used in practice by considering a sparse set of feature points extracted from the shapes.

There are other approaches whose complexity increases linearly with the size of the problem and so, they can be used for both sparse and dense problems.

Finally, a specialized set of techniques has been developed in order to work with dense correspondences from an initial sparse one [3] [4] [158] [128] [145], or techniques that progressively refine a sparse initialization into a dense one [143].

2.4 Cost function

The cost function gives a measure of how good a given correspondence is (or how far it is from the desired solution). Sometimes it is referred as the "error measure", the "objective function" or the "energy", in the case of methods that formulate the problem as the minimization of some energy function. Its formulation depends on the type of input dataset and also on the specific problem to be solved.

2.4.1 General distance metrics

In general terms, this similarity measure can be seen as an order relation between shapes that has to satisfy some basic properties. Suppose that there are three shapes P, Q and R and that the distance (cost function) between P and Q is represented as $d(P, Q)$. For d to be a valid cost function, the following properties have to be satisfied:

1. Metric properties:

   - Non-negativity: $d(P, Q) \geq 0$.
   - Identity: $dP, P = 0$.
   - Uniqueness: $d(P, Q) = 0 \leftrightarrow P = Q$.
   - Strong Triangle Inequality: $d(P, Q) + d(P, R) \geq d(Q, R)$.
   - Relaxed Triangle Inequality: $c(d(P, Q) + d(Q, R)) \geq d(P, R), c \geq 1$.
   - Symmetry: $d(P, Q) = d(Q, P)$.

   If a distance function obeys the identity, uniqueness and strong triangle inequality properties, it is referred to as a metric. If only identity and the strong triangle inequality are satisfied, it is called a semi-metric.

2. Continuity properties: with respect to similarity functions, robustness is considered a form of continuity. This property allows the distance function to be robust against the effects of discretization, noise or outliers.

3. Invariance properties: a distance function d is considered invariant under a group of transformation G if $\forall g \in G, d(g(P), g(Q)) = d(P, Q)$.

Some of the most common distance functions are:

Discrete Metric $d(P, Q) = 0$, if P equals Q. Otherwise $d(P, Q) = 1$. The main disadvantage with this metric is that if P is even minutely distorted (to form shape P') the discrete distance $d(P, P')$ will always be 1. If the discrete metric is used, computing the smallest $d(P, Q)$ over all transformations in a set G is equivalent to looking for a transformation $g \in G$ such that $g(P) = Q$. This is called the exact congruence matching.

Minkowski Distance (L_p Distance) The Minkowski distance is a metric on Euclidean space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance. For two points $x, y \in \mathbb{R}^k$, th L_p distance is defined as:

$$L_p(x, y) = \left(\sum\nolimits_{i=0}^{k} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Bottleneck Distance Assume A and B are two point-sets of size n. Let $d(a, b), a \in A, b \in B$ be the distance between two points a and b. The bottleneck distance, $F(A, B)$, is the minimum over all one-to-one correspondences f between A and B of the maximum distance $d(a, f(a))$.

Hausdorff Distance The Hausdorff distance, or Hausdorff metric, also called Pompeiu-Hausdorff distance, measures how far two subsets of a metric space are from each other. Two sets are close in the Hausdorff distance if every point of either set is close to some point of the other set. This way, the Hausdorff distance is the greatest of all the distances from a point in one set to the closest point in the other set. Let P and Q be two sets of points in $\mathbb{R}^d$. The directed Hausdorff distance from P to Q, denoted by $h(P, Q)$, is:

$$\max_{p \in P} \min_{q \in Q} \|p - q\|$$

The Haussdorf Distance between P and Q, denoted by $H(P, Q)$, can be defined as:

$$\max\{h(P, Q), h(Q, P)\}$$

Intuitively, the function $h(P, Q)$ finds the point $p \in P$ that is farthest from any point in Q and measures the distance from p to its nearest neighbor in Q. However, as is apparent from its definition, the Hausdorff distance is extremely sensitive to noise (outliers). A measure that seems to be less sensitive to noise is the partial Hausdorff distance, defined as:

$$H_k(P, Q) = \max\{h_k(P, Q), h_k(Q, P)\}$$

where $h_k(P, Q)$ is the k-th value in increasing order of the distance from a point in P to Q. Thus, $h_k(P, Q) = k^{th}_{p \in P} \ \min_{q \in Q} \ d(p, q)$. However, the partial Hausdorff distance is not a metric as it does not satisfy the triangle inequality property mentioned earlier.

Fréchet distance This is typically used to measure the similarity between curves, taking into account the location and ordering of the points along the curves. The Fréchet distance between two curves is defined as follows:

$$Fr(P,Q) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \|P(\alpha(t)) - Q(\beta(t))\|$$

where $P, Q : [0,1] \to \mathbb{R}^2$ are parametrizations of the two curves and $\alpha, \beta : [0,1] \to [0,1]$ range over all continuous and monotone increasing functions. In some variations, the monotonicity condition of the parametrization could be dropped. Partial matching could also be considered, where the goal is to find the section of one curve to which the other has the smallest Fréchet distance.

2.4.2 Rigid alignment

For the problem of rigid alignment between two or more datasets, the objective is normally defined in terms of the number of matching points, or given by a metric that quantifies how well the datasets align to each other after applying a rigid transformation.

Largest Common Pointset (LCP) here the interest is in finding a transformation that brings the largest number of points into correspondence [76] [1], given a threshold ϵ which indicates whether two points are close enough and can be considered as matching to each other. Therefore, the objective is to maximize the cardinality of the set of matched points. It can be expressed for two point sets P and Q as

$$\mathrm{LCP}(P,Q) = \sum_{p\in P} \mathrm{Match}(p,Q)$$

where

$$\mathrm{Match}(p,Q) = \begin{cases} 1, & \exists q \in Q, \|p-q\| < \epsilon \\ 0, & \text{otherwise} \end{cases}$$

for some distance measure $\|\ldots\|$

Geometric distance another common objective function does not rely on a parameter ϵ but minimizes the alignment error given by the sum of squared distances between points. That is, for each point in the transformed set, the closest point in the reference set is found and the distance between these two points is added to the error measure, expressed as:

$$\mathrm{Dist}(P,Q) = \sum_{p\in P} \mathrm{Dist}(p,Q)$$

where

$$\mathrm{Dist}(p,Q) = \min_{q\in Q} \|p-q\|$$

This is the common measure utilized in algorithms such as the Iterated Closest Point (ICP) [137]. Variants of this scheme can also be utilized, like adding orientation or surface information [32], where $\mathrm{Dist}(p,Q)$ is replaced by a more elaborate point-to-surface measures.

From the above, the LCP formulation has the advantage that partial matching can be directly handled by the objective function, since the largest set of matching points will correspond to the region of overlap between the two point sets. The sum of squared distances will necessarily consider all the points in the objective, unless the user provides the algorithm with an estimate of the amount of overlap between the point sets or a threshold to identify points that are too far away from each other [137].

2.4.3 Non-rigid alignment

In the case that the shapes are aligned to each other by a non-rigid deformation, the objective will have to incorporate terms to quantify when such a transformation is meaningful. That is, if each vertex can move freely according to its own transformation or displacement, some form of global consistency (regularization) has to be enforced. Such a regularization can be obtained by limiting the number of degrees of freedom of the transformations or by penalizing large deformations. For example, we can demand the transformations of neighboring vertices to be similar (which provides a smooth transition of transformations from one vertex to the other). Such a transformation similarity can be measured in a direct manner (e.g., by the norm between the matrix representations of the transformations [4] [158]) or according to derivatives [128]. Moreover, as in the case of similarity-based matching, the error measure in the non-rigid case also includes a quantification of how well the datasets are aligned. This can be given by a measure of geometric distance (similar to the rigid case discussed above) [158] [128] or point to plane distance in the case of surfaces [4].

2.4.4 Similarity-based correspondence

When the goal is to find a correspondence between two datasets without first aligning the shapes, shape descriptors and intrinsic measures to quantify the quality of the correspondence must be used. Therefore, for two shapes P and Q and a correspondence relation R, the objective takes the form

$$Obj(P, Q, R) = Sim(P, Q, R) + \alpha Distor(P, Q, R)$$

with a similarity term that is linear on the number of feature points and a distortion term that is usually quadratic on the number of feature points, since it commonly involves comparing properties of pairs of points. The weight a controls the influence of each term in the objective function. Automatically setting a to a value that reflects the user's goal can also be a challenging problem [25].

Similarity term this term encodes the similarity of the shape descriptors of points in correspondence. The descriptors can also include geometric attributes such as point normals or local frames, which can give an indication of whether the orientation of the points is coherent across the two matched point sets [6].

Distortion term this term quantifies how much the shapes would be deformed if their corresponding elements were brought into alignment. A common candidate for a distortion measure is the disparity in the distances between pairs of matched points.

The disparity is an approximate way of measuring the distortion introduced by the correspondence without having to first align the shapes. It can be expressed as

$$\text{Distor}(P, Q, R) = \sum \text{Dispar}(p_1, p_2, q_1, q_2), \;\; \{p_1, p_2\} \subset P, \{q_1, q_2\} \subset Q$$

where $(p_1, q_1) \in R$ and $(p_2, q_2) \in R$. The disparity term between two pairs $\{p_1, p_2\}$ and $\{q_1, q_2\}$ is given by the difference in the distances between the pairs of points

$$\text{Dispar}(p_1, p_2, q_1, q_2) = |\text{dist}_P(p_1, p_2) - \text{dist}_Q(q_1, q_2)|$$

Any appropriate distance measure can be used. Examples include Euclidean distance [28], or geodesic distance in the case of surfaces [6]. Alternatively, the compatibility between pairs of assignments can be evaluated with the intersection configuration distance [160], which utilizes fuzzy geodesics (a generaization of surface geodesics) to measure the similarity in the structural arrangement of points on the shapes.

Deformation a more elaborate form of quantifying distortion is to use a global deformation measure, such as described in [72] [192]. Once the matching of feature points is known, one shape can be deformed into the other so that the matched points are aligned. For this step, one of the recently proposed deformation methods can be utilized, as in the surveys [155] [159]. Then, measuring how much the surfaces had to deform to align to each other (an intrinsic rigidity energy) gives an indication of the distortion introduced by the correspondence. One advantage of using a surface-based deformation energy is that it is able to differentiate between correspondences that switch symmetric parts of the shape, which usually pass undetected when only pairwise distances are utilized. Note that the deformation can also be performed at the part level and with the aid of binary relations between parts [185].

2.5 Search strategy

Finding the best correspondences in a given dataset can be faced in several manners. In order to classify registration techniques based on their search strategy, two criterions are proposed: one based on particular properties of the existing approaches, and another based on terms of the solution paradigm. First classification criteria deals with the differentiation between automatic/semi-automatic techniques, global/local search strategies and pairwise/groupwise approaches. The second classification criteria deals with the distinction between methods that search for a transformation that aligns the shapes, methods that only consider the pairwise assignments between elements and find a solution using well-known optimization or search techniques, and methods that perform a hybrid search, alternating between alignment and correspondence computation.

2.5.1 Properties of existing search strategies

Fully-automatic/semi-automatic semi-automatic methods require user input, such as a proper initialization or a set of corresponding landmarks between the shapes. Fully-automatic methods do not require any user input besides a few parameter values.

Semi-automatic methods include the approaches for cross-parameterization [3] and methods that take markers as input [4] [158] [128] [145].

Although user input is required for the proper initialization of certain methods, this can also be seen as a necessary requirement when the semantics of the shapes cannot be easily inferred. Therefore, a track open for future research is that of methods based on a feedback loop, where the user gradually improves a correspondence based on his or her preferences. Ideally, such a method would minimize the amount of user interaction and provide hints on what in- formation is missing to refine the correspondence.

Global/local search : this distinction refers to the way the solution space is explored: in the case of global search strategies, the whole domain is considered in order to get a good solution (e.g., by performing an exhaustive search [60] [192]). On the other hand, when results of the method depend directly on its initialization, only a part of the solution space is considered. This is what is called local search strategies. The initialization can be given by a user, as with semi-automatic methods, or by a fully-automatic method, whose solution will be used as the starting point for the local search of the algorithm.

The most prominent example of the local search category is the ICP algorithm [137]. Since this iterative process follows a single path in the solution space, it can end up with a result that is a local minimum. Thus, the initial state clearly influences the final result of the algorithm, and therefore different forms of initialization have been proposed for this algorithm (which take the form of computing a pre-alignment between the shapes to be matched).

Another example of algorithms that perform local search are the methods for non-rigid alignment based on explicitly computing the transformation for each shape element. Since these transformations are computed with a method based on gradient descent or Newton's optimization, the initialization will also necessarily influence the final correspondence result [4] [158] [128].

Pairwise/groupwise methods for group correspondence appear predominantly in approaches that focus on object reconstruction from fragments and in the computational anatomy community [66], where a coherent correspondence between a group of shapes is important for the accurate construction of a statistical model. A successful class of methods for this case is based on the minimum description length approach, where quality criteria of the statistical model are used to guide the computation of the group correspondence and simultaneous construction of the statistical model [42].

Although the term group correspondence is not used in the field of time-varying reconstruction, a certain class of methods applied to this problem can also be seen as following this approach, since all scans are considered simultaneously in the registration. The difference to the case of anatomical shapes is that each scan can deform over time and there can be a significant amount of missing data between frames, while in the anatomy case the goal is typically a full correspondence between complete shapes, which are seen as variations from the same mean shape of an organ or bone. The time-varying reconstruction methods pose the problem as the reconstruction of a space-time surface [111] [163] [147], or obtain a skeleton that is coherent

for all the time frames [194]. The advantage of such formulations is that missing data can be filled in with data from frames that are further away in time.

2.5.2 Solution paradigm

Classifying registration techniques according to their solution paradigm is based on how do they obtain the correspondences. This way, the methods are primarily divided into those that search for an aligning transformation, those that search directly for a correspondence without performing alignment, and the ICP method, which works in a hybrid manner alternating between transformation search and correspondence search.

2.5.2.1 Transformation and alignment search

Methods enclosed in this category are the ones that first search for a transformation that aligns shapes and, then, derive the correspondences by proximity between facing points in the considered shapes. For these methods, a secondary classification criteria can be established, depending on the group of transformations considered.

Rigid transformations When considering rigid transformations, the final alignment can be inferred from a small set of sample points. For example, if the problem faced is three-dimensional, the transformation matrix can be obtained by simply matching 3 points from each shape.

After a candidate transformation has ben calculated, its correction has to be validated. To do so, two major approaches are used: one based on verification, and another based on voting.

Verification The most simple verification approach is a naive algorithm [73], which exhaustively samples three points from the first shape and three points from the second one. For each possible combination, a transformation matrix is computed, and some cost function is evaluated in order to quantify the quality of the achieved result. After testing all possible triplets, the best transformation is returned as result. The resulting time cost of this strategy is $\mathcal{O}(m^3n^2 \log n)$ for the 2D case, and $\mathcal{O}(m^4n^3 \log n)$ for the 3D case, considering that m and n are the sizes of the two considered point-sets.

In order to improve performance, some approaches use a Random Sample Consensus (RANSAC) strategy [49] that consists on using only a constant-sized set of random samples on one shape, reducing complexity by a factor of $\mathcal{O}(m^3)$ in the 3D case, or even applying this idea also to the verification step, reducing the complexity by another factor of $\mathcal{O}(m)$ [76].

Other approaches propose to explore geometric invariances maintained by the transformations. One such case is the ratio of distances between three coplanar points, which is preserved by rigid and affine transformations. Thus, the problem of searching for triplets of points that provide the optimal transformation can be transposed to that of finding four sets of coplanar points that share the same ratios [74]. By pre-processing these invariances and keeping them in appropriate data structures that allow for efficient retrieval, output-sensitive methods can be achieved [1], reducing the complexity of the alignment problem even further to $\mathcal{O}(n^2 + k)$, where k is the size of the reported output.

Voting Instead of sampling a transformation and evaluating its quality, the verification step can be replaced by a voting procedure. For this purpose, pose clustering utilizes an accumulation table [157] [117]. After selecting two triplets of points and deriving a transformation, a vote indexed by the parameters of the transformation is stored in the table. At the end of this $\mathcal{O}(m^3n^3)$ process, the cells with most votes correspond to the best candidate transformations that align the point sets.

Another technique based on the voting concept is geometric hashing. This method makes use of pre-processing to speed up the alignment [184]. Its main idea is to store in a hash table all the possible configurations of a group of reference point sets, so that when we seek the reference point set that best matches to a query point set, this search can be performed efficiently. This splits the previous $\mathcal{O}(m^3n^3)$ complexity of the naive enumeration into an $\mathcal{O}(m^3 \log n)$ preprocessing phase (which samples all the possible configurations of a reference set and stores them in the hash table) and an $\mathcal{O}(n^3 \log n)$ query phase (which samples all the possible configurations of the query set and accumulates votes in the hash table to allow the retrieval of the best matching reference set). The increase in speed in the query phase is gained at the cost of memory.

Piecewise transformation previous methods use one global transformation to compute the matching between one shape to another. This same idea can be generalized by applying transformations to local portions of the shapes. In [28], these transformations are applied in a piece-wise rigid manner to establish a correspondence between articulated shapes. The problem is formulated as labeling the vertices of the shapes with candidate transformations. Since now the vertices are restricted to possess a transformation from a pre-defined set, the solution search is greatly simplified. By adding a regularization term to the labeling optimization, a grouping of the vertices into rigid components is guaranteed. An alternative to this approach is to explicitly fit the shapes to a kinematic skeleton of articulated bones, so that the skeleton can be used to track the movement of the shape and also infer in which regions there is missing data [130] [29] [58].

Non-rigid alignment for the methods described in [4] [158] [128], different transformations are assigned to each vertex on the shape. The problem is formulated as finding the best transformation that brings each vertex in a reference shape close to its counterpart in the target shape, and a regularization term is added to enforce the similarity of transformations across neighboring vertices. The difficulty in this setting is avoiding solutions that are local minima. This is achieved by initializing the methods with a set of corresponding marker points and solving the optimization in a multi-level fashion. The optimization can be posed as a non-linear least squares problem and solved with a Newton-based method. This mainly involves the computation of derivatives of the high-dimensional problem and least-squares optimization at each step. Alternatively, the optimization can be performed by labeling a Markov random field [127].

Instead of computing local deformations or displacements for the vertices, in [145], the displacements are implicitly obtained by learning a function that warps one shape into the other. The warp is obtained by solving a convex optimization pro-

blem similar to learning a support vector machine classifier (which includes a form of regularization in its definition). Thus, global minima are avoided.

An extension of this methods which avoids the need of marker points is proposed in [92], where the alignment between two shapes is performed with two separated transformations: a global rigid transformation which roughly aligns the shapes, and per-vertex affine transformations that bring the non-rigid shapes into full alignment. A robust alignment can also be obtained by deforming one shape into the other in terms of a 3D optical flow [43] or a Laplacian deformation of the meshes [44].

2.5.2.2 Correspondence search

Correspondence search methods work primarily with the pairwise assignments between feature points, without searching for transformations that align the shapes. The correspondence problem is typically defined as optimizing an objective function of the form $Obj(P, Q, R) = Sim(P, Q, R) + \alpha Distor(P, Q, R)$. The objective is based on the quality of pairwise assignments and the compatibility between pairs of such assignments.

Most of the approaches in this category find the solution by using well-known discrete or continuous optimization methods: if the objective being optimized is only composed of a similarity term $Sim(P, Q)$, then the formulation becomes a Linear Assignment Problem (LAP). This simplified objective can be solved by the simplex algorithm, since it is a special case of a linear program [122]. However, if the correspondence is constrained to a one-to-one mapping, the problem becomes that of finding an optimal matching in a weighted bipartite graph [122]. On the other hand, if the objective comprises both the linear (pairwise assignments) and quadratic terms (compatibility between pairs of such assignment), we arrive at a Quadratic Assignment Problem (QAP), which is known to be NP-hard [126]. Several techniques have been proposed to compute approximate solutions to this problem. Examples include the softassign technique [62] (which iteratively normalizes rows and columns of an affinity matrix), concave programming [103], approximations based on linear programming [13], spectral clustering [90], or relaxation labeling [196]. It can also be formulated in probabilistic terms and solved as a convex optimization problem [189].

Another group of methods solves the problem in the discrete setting without resorting to the continuous domain. One common solution approach in the discrete case is to solve the problem by computing an optimal labeling of a graph, for example, the problem can be posed in terms of a Markov network where the set of labels corresponds to matching points on the target shape [6] [194]. Other methods make use of heuristics for combinatorial optimization, such as ant colony optimization [170]. One more option is to sample the space of correspondences in search of a solution, guided by geodesic distances and importance sampling [165].

A special group of methods in discrete optimization utilize a tree-based search to explore the solution space, such as branch-and-bound or priority search [60] [53] [192] [9] [185]. During the tree expansion, each node represents a partial solution. A full solution is found by following the path from the root of the tree to one of its leaves. Although the specific strategy in which the tree is expanded differs from method to method, these techniques usually involve three important steps: expanding a node that represents a new partial solution (branching), estimating how far the partial solution is from the optimum

solution (bounding), and eliminating nodes that will not lead to the optimum solution (pruning).

In the case of correspondence, solutions are mainly represented as collections of assignments between pairs of feature points, and the expansion step involves adding a new pairwise assignment to a given solution. Bounding and pruning can be performed by verifying the quality of the registration given by the current solution, either by aligning the shapes [60] or by deforming one shape into the other [192]. Other pruning methods include testing the compatibility between pairwise assignments, such as quantifying the distortion introduced in the Euclidean [60] [53] or geodesic distances [192] [9] between pairs of points, or testing the agreement in the spatial configuration of the shapes [9].

When a hierarchical or multi-resolution structure is extracted from the shape representation, this information can also be considered in the solution search. Skeletons are commonly represented as trees or graphs for which a tree can be easily extracted. Therefore, in this context, it is common to resort to search-based algorithms that take this hierarchy into account [162]. Methods of a more greedy nature can also benefit from such hierarchical [16] or coarse-to-fine representations of the shapes [67].

2.5.2.3 Hybrid search: ICP

The Iterative Closest Point (ICP) method [32] [15] can be considered as an hybrid search strategy, since it alternates between finding correspondences in the considered shapes (based on proximity), and calculating the transformation that better aligns them. It is a local search algorithm, whose correction depends on a initial guess for the relative rigid-body transform between two shapes that aligns them, computed with a different global search strategy. ICP is widely used for geometric alignment of three-dimensional models. In fact, almost every registration technique searching for a rigid alignment between shapes uses it in some of the matching stages.

Given two shapes, P and Q, the first step of the ICP algorithm consists on *selecting* which points $p \in P$ use for computing the correspondence. Original implementation of the algorithm [15] considered all the available points, and looked for the closest neighbor of each one of them on shape Q. In [169], a uniform subsampling of the available points was proposed, in order to accelerate the search process. A randomized approach using different sample points at each iteration was proposed in [107]. In special cases, when color or intensity is also considered, the alignment can be aided by selecting points with high intensity gradient. An alternative approach that selected points in both shapes was introduced in [61].

Once candidate points are selected, the next stage of ICP *matches* these points with their counterpart on the other shape. On the original algorithm, this search was exhaustively performed, increasing considerably the execution time of the registration. In [151] the use of a $k-d$ tree was proposed to accelerate the correspondence search. Another approach, that intersects in the destination shape a ray originating at the source point and in the direction of its normal was proposed in [32]. Some other approaches based on projections onto the destination mesh where proposed [17] [113] [12] [48] [179] [132].

For each pair of selected points, ICP computes a *weight* based on some criteria. Original approach had a constant value, but some alternatives have been suggested, like weighting proportionally to the distance, or the compatibility of the normals [61] or based on the expected effect of scanner noise on the uncertainty in the error metric.

Closely related to assigning weights to corresponding pairs is rejecting certain pairs entirely. The purpose of this is usually to eliminate outliers, which may have a large effect when performing minimization. Some common criterions are based on a maximum separation between pairs, rejecting the worst $n\%$ pairs based on a metric (in [131] a 10% value is given as a reference), rejecting pairs whose point-to-point distance is larger than some multiple of the standard deviation distances ([107] suggest 2.5 times), or rejecting pairs not consistent with neighboring pairs.

Finally, when potentially bad correspondences have been pruned, ICP minimizes a given error metric in order to find the best transformation that aligns the selected pairs. When considering sum of point-to-point squared distances, closed-form solutions exists for the minimization problem, such as singular value decomposition [7] quaternions [69], orthonormal matrices [70], and dual quaternions [174]. If considering point-to-plane distances, or plane-to-plane distances (being the planes defined by each point position and normal), no closed-form solutions are available. The least squares equations may be solved using a generic non-linear method (e.g. Levenberg-Marquardt), or by simply linearizing the problem (assuming that incremental rotations are small so $\sin\theta \approx \theta$ and $\cos\theta \approx 1$).

To perform the alignment search, ICP can repeatedly generate new sets of corresponding points using the calculated transformation in the previous step, and finding a new transformation that minimizes the error metric, or combine this idea with an extrapolated transform to accelerate convergence [15]. It can also start with several perturbations in the initial conditions and selecting the best result [151] to avoid local minima solutions, or perform the iterative minimization using various randomly-selected subsets of points, then selecting the optimal result using a robust metric [107] or, finally, perform a stochastic search using simulated annealing [17].

CHAPTER 3

Background on Automatic Fragment Reconstruction

In the previous chapter, the general problem of shape matching was introduced, and the most common techniques were explained and classified. This chapter focuses on the specific application of shape matching techniques to the automatic reconstruction of archaeological artifacts from fragments.

Reconstruction of ancient artifacts from fragments found at the archaeological sites, is a tedious task that requires many hours of work from the archaeologist and restoration personnel. Historically, this reconstruction process has been manual, occupying a major proportion of the human effort at excavation sites. In fact, since the assembly work is so time-consuming and labor-intensive, the reconstruction is not even attempted at countless sites around the world, leaving vast quantities of material unstudied and stored indefinitely.

Advancements in low-cost, high-volume acquisition devices and computer systems performance have provided a new tool for archaeologists to face the problem of reconstruction. Operating on digital models of fragments can rapidly and systematically consider many thousands of possible fragment alignments. The final goal is reducing the amount of candidate matches between fragments, and providing an automatic or semi-automatic tool to recompose the original artifact efficiently.

This chapter analyses the most remarkable techniques in this field following a chronological/complexity criteria: from 2D jigsaw puzzles to complex 3D multi-piece problems.

3.1 Jigsaw Puzzles

The problem domain of jigsaw puzzles is widely known to almost every person from childhood. Given n different non-overlapping pieces of an image, the player has to reconstruct the original image, taking advantage of both the shape and color information (when available). Although this game was proven to be an NP-complete problem [5] [46], it has been played successfully by children worldwide. Solutions to this problem might benefit the fields of biology [106], chemistry [176], speech descrambling [193], archeology [24], image editing [33] and the recovery of shredded documents or photographs [96] [94] [45].

The first approach in this discipline faced the problem of solving jigsaw puzzles [51]. Jigsaw puzzles considers a set of constraints that considerably simplify the correspondence search: each fragment is rectangular, with a flat outside edge, pieces often have exactly four neighbors, and they fit together via interlocking "indents" and "outdents". Although there may be measurement noise when digitizing pieces, erosion was not considered. These properties lead to a variety domain-specific algorithms that, while instructive, do not apply in a general fragment-matching context.

The solution proposed in [51] made no special assumptions about the type of surface match, except that most matches were complete matches between two edges. Each piece boundary was divided into edges at slope discontinuities, and global shape statistics of each edge were used to find candidate matches. Final matching determinations came from measuring the distance between aligned curve candidates. The puzzle was constructed piece-by-piece, spiraling outward from the starting point.

In [183] larger puzzles were solved by assuming four-sided pieces and efficiently computing the alignment error of every pair of sides. Edge pieces were detected separately, and an alignment sequence was searched among all pairwise alignments until a loop was closed. Then, starting at one corner of the frame and working across the puzzle, all pieces were tested for addition. At all times, the top 200 configurations were maintained, until the puzzle was completely assembled.

A more recent approach [63] eliminates the requirement that pieces have four sides. Instead of explicitly detecting the sides, this algorithm extracts indents, straight edges (to detect edge pieces), and outdents. Ellipses are fit to indents and outdents, and their centers are used as feature points to determine matches. After reconstructing the border, pieces are added to the puzzle which match with, at least, two already-placed pieces. New pieces are added in a greedy way, however a global re-alignment is performed on all pieces after each addition to redistribute error.

An interesting variation of purely geometrical approaches has gained increasing importance in the latest years: image puzzles, also called pictorial jigsaw puzzles (in contrast of the previous commented approaches, which are apictorial). For this type of puzzles, the shape information of individual pieces is normally disappeared, and replaced by chromatic information. This causes evaluating pairwise affinities critical. In this situation, the jigsaw puzzles problem becomes more challenging and compels the approaches to focus the efforts on image content.

The general methods in the recent works can be roughly divided into two steps: a measure for describing the proximity between pieces and a strategy for puzzle assembly. In the first step, only the image content consistency between adjoining pieces is considered. In the second step, the strategy of assembling the pieces bases on the previous measure.

However, the measure computed in the first step is not completely correct due to the locality of measure. Therefore, a self-correcting mechanism is necessary before the assembly. Some of the most recent approaches in this field can be found in [77] [149] [112] [195].

3.2 Contour Matching Techniques

Contour matching techniques provide solutions for more general problems, without distinguishing between specific edges or features. Applied to 2D domains, some of the most outstanding techniques can be found in [38] [86] [125] [68] and the references therein.

Approaches presented in [38] [86] both represent discretized contours using the curvature at each point. The matching cost between sequences of points on two contours is related to the difference in curvature at corresponding points and difference in length between the two sequences. The optimal matching sequence given a pair of starting points on each contour is found using a dynamic program. In the case of [38], a multi-scale approach is implemented in order to speed-up the search process, whilst in [86] only two different levels of detail are used. Despite being focused on 2D problems, [86] generalizes aspects of the proposed approach in order to extend the technique to certain 3D pieces.

In [125] an automatic process for reconstructing the wall paintings of Thera (Santorini) is proposed. Taking as input data photographies of each fragment, contours are automatically extracted and represented as discrete bitmaps. Then, for two input fragments, all possible pairings of fixed-length segments of edges are exhaustively searched. Alignment is computed based on the first pixel in each segment, and alignment quality is determined by a weighted measurement of both the open area between segments and their intersection area. A measure of each segment's curvature is used to prune the set of pairs to compare.

One of the most famous approaches for automatically reconstructing archaeological broken artifacts is the Stanford Digital Forma Urbis Romae Project [85]. Despite working with 3D models, this approach could not take advantage of the geometric information of the fragments because they were heavily eroded and adjacent fracture surfaces sometimes did not even touch each other. Instead, they used a contour matching strategy where the reconstruction was performed using the annotated incisions on the fragments.

The proposed automated boundary incision matching technique searches for incised topography that corresponds across the boundaries between two candidate adjacent fragments. A set of topographic feature types was selected from a hierarchy of over 150 possible labels which allowed annotations to encode varying levels of certainty about the topography depicted by particular incisions. The search process considered each possible pair of fragments, and then considered each reasonable alignment of annotated features between the fragment pairs. Candidate configurations of two fragments were scored based on the alignment of the corresponding feature types, being the highest scores the ones that had the highest number of strongly similar features aligned. Fragment pairings with highest scores were output in a ranked listing for further review by an archaeological expert.

A very common extension of contour matching techniques in archaeological fragment reconstruction are pottery re-assembly techniques. These are applied to revolution surfaces and can take advantage on the additional constraints of axial symmetry, torsion and curvature to guide the search process [81] [180] [86] [198]. Archaeologists commonly document pots based on their axis of rotation and profile curve (the pot's cross section in a plane through the axis of rotation) and many computational methods have been developed

to extract this information

The approach proposed in [80] starts with the estimation of the correct orientation of each fragment, taking advantage of the axial symmetry. This rotational axis is calculated using a Hough inspired technique. Next, the classification of the fragment based on its profile section allows the authors to decide to which class of object it belongs to. Since orientation of the candidate fragments is known the search space to work in is completely defined using two degrees of freedom. Then, a matching algorithm based on the point-by-point distance between facing outlines is used to perform the final reconstruction.

In [180], not only axial symmetry is exploited, but also contours are extracted to aid in the matching process. Starting with an initial estimate of the axis, a 6-th order polynomial is fit to the sherd for the profile curve. The axis is re-estimated, and the process iterates until convergence. Then, a probabilistic framework is used to find contour matches which are consistent with the axis of rotation and profile curve, which are re-estimated from all aligned fragments as the pot is reconstructed.

Algorithm described in [181] focuses on the problem of simultaneously estimating the geometric parameters of the unknown axially symmetric surface while aligning the pieces, assuming hat the correspondence between the piece break segments is already known. The method presented in this work solves the problem using a 2 step recursive algorithm. In step 1, the geometric constraint of axial symmetry is used to obtain an estimate of the unknown surface for a configuration of aligned pieces. Step 2 uses this surface estimation to obtain accurate estimates of the break segment and transformation parameters for a new piece added to the configuration.

3.3 Surface Matching

All of the previous matching contexts are two dimensional (embedded in three dimensions in the case of pottery matching, or in the Stanford Digital Forma Urbis Romae Project). Surface matching techniques are applied to more general problems, considering 3D input data, and providing solutions to problems with six degrees of freedom. In these kind of techniques the combinatory explosion in the solution space makes exhaustive searches prohibitive. Two ways to reduce the complexity: (1) grouping surfaces into facets, between which to search for matches, and (2) identifying features in all fragments to reduce the search space.

The first approach in this area was presented in [123] [124] as a semi-automatic system for the reconstruction of archaeological artifacts from their fragments. This approach exploited the underlying assumption that the fractured faces were nearly planar and they matched each other completely. Using a projective space, GPU depth maps where analyzed to reconstruct the original object. The proposed reconstruction scheme was divided into three main stages: the first stage dealt mesh segmentation (detecting independent regions, and classifying as “fracture facets” or “original facets”). In the second stage all pair of fragments and facets marked as “fracture” were compared looking for a pose that optimized the alignment. The third stage dealt with the full reconstruction by selecting those fragment combinations that minimized a global reconstruction error, equal the sum of matching errors of a given set of fragment pairs.

As happened with the previous approach, the technique described in [124] segments polygonal meshes into facets based on normal compatibility, and classifies them into orig-

inal or fracture surfaces based on roughness. Fracture surfaces are aligned with each other by first aligning their average normals, then using simulated annealing to minimize error. For fracture surfaces which border an original surface, contour matching on the boundary is used to quickly find candidate matches and alignments.

The technique described in [71] also segments fragments into faces to help restrict matching, but eliminates the requirement that faces completely match each other by computing local features to match instead. Volume descriptors giving the amount of the fragment inside a ball centered at each point, and the distance of each point in the ball from the surface, are computed everywhere. Features are computed with multiple-size balls, and are clustered by overlapping ranges of value and at multiple scales (so that larger clusters contain smaller ones). Each cluster is then represented by it center of mass, orientation (as defined by PCA), and size and anisotropy signatures computed from its singular values. Similar edge features are computed on face boundaries. Now fine-scale clusters with similar size and anisotropy signatures are considered matches provided all enclosing, larger-scale feature clusters also match. Additional geometric pruning finds sets of feature correspondences that lead to a consistent alignment. Starting from a rough initial alignment of pairs of clusters (using the PCA axes and singular values), surfaces are further aligned using a non-intersecting variant of ICP; any feature correspondence that leads to unstable alignment is rejected. Next, a forward search over all corresponding features between two faces computes the set of candidate alignments (there may be several corresponding to different subsets of correspondences), and a graph-based global optimization over all candidate fragment alignments extracts the final assembly. Final positions are optimized using a global, non-intersecting alignment algorithm.

Finally, [24] exploits the orientation constraints of flat fragments to achieve a simple, fast matcher based on edge geometry. The proposed technique analyzes exhaustively every possible alignment of a pair of fragments in a few seconds. To efficiently compute fragment matches, the proposed technique regularly resamples fragment edges into what authors call a "ribbon". A contour is extracted at a fixed offset from the front surface, then each sample is extruded vertically in a plane defined by the contour point's smoothed normal. Ribbon points are arranged in a grid, allowing efficient computation of correspondences. Results of the technique were evaluated using ground-truth fresco paintings, proving a very high success rate.

3.4 Multi-Feature Matching

One of the main problems faced during reconstruction of fractured archaeological artifacts is sorting through a large number of candidate matches between fragments to find the relatively few that are correct. Approaches commented so far in this chapter focus their effort on evaluating the quality of a match according to the geometric compatibility and, sometimes, color information. However, when artifacts have deteriorated over many years, ambiguity is highly increased, making extremely hard to distinguish between correct and incorrect correspondences. For example, some techniques may consider features such as color, which frequently have changed over time even among neighboring fragments. Alternatively, other techniques may operate exclusively on 3D geometry, which may not only have deteriorated, but is also challenging to acquire with the same fidelity and resolution as color images.

As suggested in [166], search process can be enriched by additional criteria, in addition to the geometric one. Thus, including color and texture information, may provide better results in certain cases [142] [50]. However, when considering multiple properties, a classifier is required to estimate the quality of a given alignment, according to all considered criteria [54] [148]. This way, in order to rank all the potential matches found, a multi-feature metric has to be used that, somehow, has to balance all the available information.

The approach described in [166] addresses the problem of reconstruction by considering multiple cues based on color, shape, and normal maps. The latter feature is a new source of information that had not been used before for matching purposes, and authors argue that it combines high data quality and resolution with high discriminability and robustness with respect to certain types of deterioration. As [24] proved, it is practical to use flatbed scanners to obtain normal maps of mostly-flat objects with 600 or 1200 dpi resolution. These normal maps reveal salient surface characteristics including string impressions, brush strokes, surface roughness, and fine cracks.

Using this information, a set of novel feature descriptors were introduced. Using only chromatic information, authors formalized features based on average color, saturation and variance and contour curvatures. Using only normal information, the following descriptors were introduced: average and variance of normals, normal discontinuities and dominant orientation. By combining chromatic and normal information new descriptors were suggested: color/normal variation, cracking and erosion. Finally, exploiting the scanned 3D model thickness and ribbon-matcher error and volume intersection derived from the approach presented in [24] were also included.

Once all the features were introduced, in order to combine them all in a metric relation, authors manually labeled sets of matching fragments, and selected some random non-matching pairs. Using this data as a training set, they trained four machine learning approaches: decision trees, random forests, support vector machines and logistic regression, and compared results in terms of performance, precision-recall and correction.

A similar technique was introduced in [166] that combined the ideas of the previous commented approach with the ones introduced in [148], where the properties of the matches in assembled frescoes were analyzed. The goal is to train a classifier that predicts the probability that a proposed match between two fresco fragments is correct. This way, large numbers of properties can be considered into a complex match scoring function.

3.5 Multi-Piece Matching

Multi-piece matching techniques are applied to the global reconstruction of the original object. Taking as input data pairwise matches, these techniques deal with the assembly of larger clusters. Considering the whole object adds new challenges to the reconstruction process, but can also help disambiguating pair-wise matches (since global consistency adds new constraints to the problem formulation). The main difficulty associated to this context is that small errors in alignment between adjacent fragments lead to gaps and interpenetration. To face this problem, most of the techniques apply a global relaxation step [63] that optimizes the global alignment. Even with global relaxation, however, error accumulation increases as the cluster grows, eventually rendering true and false matches indistinguishable. This effect limits the problem size automated assembly systems can

handle in practice.

The approach presented in [71] uses a graph-based global optimization over all candidate fragment alignments in order to extract the final assembly. Final positions are optimized using a global, non-intersecting alignment algorithm. The key idea proposed is to iteratively perform a local multi-piece registration and merge matched fragments into larger clusters until the original object is fully reassembled.

Applied to ripped-up documents, the approach presented in [197] performs a global reconstruction in two stages: first, it finds pair-wise candidate matches from document fragments using a contour matching approach. Second, the global disambiguation problem is formulated in a relaxation scheme in which the definition of compatibility between neighboring matches is proposed, and global consistency is defined as the global criterion. Initially, global match confidences are assigned to each of the candidate matches. After that, the overall local relationships among neighboring matches are evaluated by computing their global consistency. Then, these confidences are iteratively updated using the gradient projection method to maximize the criterion. This leads to a globally consistent solution and, provides complete document reconstruction.

For archaeological purposes, the technique described in [26] focuses on the problem of automatically agglomerating clusters of fragments from previously determined pairwise matches. By introducing two careful modifications on the traditional relaxation scheme, authors lift the limit imposed by error accumulation considerably. In contrast to previous work, global relaxation is integrated earlier, in the search phase of the assembly process. In addition, connections between assembled fragments are not fixed, but rather left flexible throughout the assembly. By modifying two representative assembly algorithms, authors demonstrate the effectiveness of the presented approach.

CHAPTER 4

Acquisition

Acquiring models of archaeological artifacts allows having detailed digital representations of them that can be used for analytical or dissemination purposes. Furthermore, operating on digital models prevents damaging original artifacts and facilitates the access to the research community. Acquisition stage is the only part in the entire process where original physical fragments are involved. The goal of this stage is to generate a vectorial representation of them, as accurate as possible, without compromising their safety.

In order to create these digital representations, acquisition devices are used to capture the topology of real objects. There are several technologies to perform such operation, like common 2D acquisition devices, such as digital cameras or flatbed scanners, contact scanners based on physical touch, photogrammetry approaches, time of flight cameras and triangulation laser scanners, amongst others. Last ones are the most common in heritage applications, given the balance between accuracy and cost that they offer.

Depending on the nature of the fragments, a 2D or 3D approach will be used: for thin fragments, or fragments with sharp fracture edges, a 2D approach is more convenient, since 2D acquisition devices are cheaper, faster, and provide very accurate results. Also, the search process using 2D images is faster than the one that considers 3D volumes. In case the 2D characterization does not provide enough information for the reconstruction process, 3D models will be acquired using a triangulation based 3D scanner, and B-Rep models will be generated to work with. Next sections cover the details of both processes, paying special attention to the 3D case, where more complex situations have to be faced.

Major contribution of this chapter is the introduction of Cyclododecane as whitening spray, allowing to perform the 3D acquisitions of reflective/refractive archaeological artifacts without compromising their integrity.

4.1 2D acquisition

Obtaining two-dimensional representations of fragments is quite simple: using a flatbed scanner or a digital camera, objects can be quickly acquired into high resolution bitmaps, with 600 or 1200 dpi. Together with the contour information, color data is retrieved and, as explained in [166], high resolution normal maps can be generated, revealing salient surfaces characteristics including brush strokes, surfaces roughness... However, since the proposed technique focuses only on geometry, all the extra information is discarded.

Given that the proposed search strategy finds the rigid transformation that better aligns fragments, scale transformations are not considered. This has to be taken into account during the acquisition stage in the sense that all fragments have to be represented proportionally to their original size. This way, an extra degree of freedom in the solution space is avoided, and the final search time is reduced.

Once discrete bitmaps are obtained, a vectorial representation of the contour has to be computed. To do so, an image segmentation stage is mandatory, in order to differentiate between the fragment and the background. Afterwards, a contour extraction has to be applied over the fragment's discrete representation.

Segmenting the image is straight forward, considering that it has been acquired in a controlled environment, and that a proper background for the acquisition has been chosen.

For the contour extraction stage, frontier pixels have to be identified. Given a fragment pixel $p = (x_p, y_p)$, it is considered to be a frontier pixel if any of its 4-connected neighbors is a background pixel. A pixel $q = (x_q, y_q) \neq p$ is a 4-connected neighbor of p if it verifies that $q = (x_p + 1, y_p) \curlyvee q = (x_p, y_p + 1)$. It is said that q is an 8-connected neighbor of p if it verifies that $|x_p - x_q| \leq 1 \curlywedge |y_p - y_q| \leq 1$ (Figure 4.1).

1	2	3
4	5	6
7	8	9

Figure 4.1: *Pixel neighborhood. For pixel number 5 (shaded with gray), pixels 1 to 4 and 6 to 9 are the 8-connected neighbors, and pixels 2, 4, 6, 8 are the 4-connected ones.*

Once frontier pixels are extracted, a random frontier pixel is selected as starting point and a line connecting it to one of its 8-connected neighbors is created. Iteratively, a new line starting from the last visited pixel is created with the same criteria, taking care of not visiting twice the same pixels. This process ends when the initial vertex is reached again.

Notice how this simple contour extraction technique expects fragments to be solid (without islands inside). If that were the case, inner holes would be discarded, considering only the outer perimeter. More complex contour extraction approaches can be found in the bibliography, since this kind of techniques are very common in the field of Computer Vision. Nevertheless, considering that contour extraction is out of the scope of this document and that the proposed solution is sufficient for the reconstruction purposes of this Thesis, this matter will not be discussed in more detail.

4.2 3D acquisition

Generating 3D models from the original fragments is much more complicated than the 2D acquisition. This is a consequence of two main reasons: self-occlusions and reflections/refractions. Self-occlusions happen because there is no way of observing (and thus acquiring) the whole surface of the fragment simultaneously. Reflections/refractions are related to the physical limitations of the acquisition devices, and the properties of the fragment's surface.

To face self-occlusion problems a set of partial views have to be acquired from different locations. After the acquisition, all views have to be registered in order to fully characterize the original object. To do so, two stages are involved: global registration and local refinement of the alignment. Global registration consists on identifying common parts in the acquired point clouds from two different views. This way the search space gets considerably reduced and an initialization is provided to the local refinement stage. The second step applies ICP from the provided rough alignment and refines the final result. In order to obtain quality results, it is very important that overlapping area between two different views are big enough to have as much correspondences as possible.

In order to accelerate the acquisition process, a turning plate connected to the scanner has been used. This way, by using a calibration chart that allows identifying the rotation axis of the plate, the global registration stage can be automatized: having the scanner in a fixed position, pointing towards the center of the plate, and the rotation axis properly identified in the scanner's reference frame, a set of equi-spaced angular views of the original object are acquired and globally aligned using the known orientations of the turning plate. Since local refinement is automatic, both stages are executed without user's intervention.

To face reflections/refractions the simplest way of proceeding will be to use an acquisition device whose technology allows working with reflective/refractive fragments. This way, contact scanners which probe the original object through physical touch, are good candidates. However, this technology has been discarded, given that fragments are sometimes extremely fragile and that acquisition times are longer.

Non-contact 3D scanners are mostly based on reflection: time-of-flight cameras probe distances to the object by timing the round-trip time of a pulse of light, triangulation based 3D laser scanners use calibrated cameras to identify the location of a laser beam reflected by the object's surface...

Transparent objects violate most of the fundamental assumptions made by vision algorithms. For instance, they cause the projection of a background scene to the image plane to be deformed. Furthermore, this projection can vary from one viewpoint to the next. Additionally the reflection of light by the surface complicates the reconstruction process.

To address this problem, industry solutions commonly use whitening sprays designed to create an opaque thin film over the object's surface. This way, acquisition using triangulation laser scanners can be performed and, if the film is thin enough, measurement errors introduced by the sprays can be ignored (since 3D scanners accuracy is orders of magnitude coarser).

The main problem with these sprays is that they are not suitable for archaeological fragments: after the acquisition has been performed, removing the spray layer requires lot of rubbing and the use of strong dissolvents, which may damage or alter the surface properties of the acquired object.

Given the unique nature of the fragments and their fragility, neither of these two procedures can be applied for obvious reasons. Also, chemical stability is a common requirement for manipulating these kind of artifacts and none of the available commercial spray satisfies this condition.

To provide a solution to the acquisition problem in the field of archaeology, a new application of a common conservation material is proposed: the use of cyclododecane as whitening spray. Cyclododecane (abbreviated as CDD) is a volatile cyclic alkaline ($C_{12}H_{24}$) that is solid at room temperature. For being non-polar and compound exclusively by carbon and hydrogen, it is an inert material whose most attractive characteristic is that it sublimes, eliminating additional chemical or physical treatment steps to remove it. Its physico-chemical properties include good film-forming capabilities, insolubility in water, solubility in organic solvents and low toxicity, which render the compound particularly useful in the field of cultural heritage. In fact, in the latest years it has become very common in this field as a temporary consolidant, sealant and hydrophobic protecting coating for fragile materials (paintings, ceramics, papers, textiles...).

Figure 4.2 shows the accelerated sublimation process, where the artifact has been oversprayed to better illustrate this concept.

Figure 4.2: *From left to right and top to bottom, the sublimation of cyclododecane applied on an artifact. Images are captured in 45 seconds intervals since the application of the spray (top-left picture). The process has been accelerated pointing a hair dryer towards the object.*

Cyclododecane can be applied in different manners but, for the proposed use, spray seems the most convenient since it is the one that creates the thinner film. After applying it to an object, it can be observed that its surface turns opaque and white, which are the most desirable conditions for laser scanning purposes. Immediately after the application, the thin layer starts to sublimate (pass from solid to gas state).

The average sublimation speed has been observed to be of $0.03mm$ each 24 hours but, there are additional factors that affect this behavior, like film thickness and density, substrate porosity, atmospheric temperature and pressure and air exchange over the surface of the film. This way sublimation can be accelerated by directing a hair dryer over the surface of the fragment, or retarded by reducing the airflow over the surface of the object.

4.2.1 Residuals Analysis

For CCD to be useful in acquisition it has to be proven that, after applying it, the fragment can be correctly acquired and that the error introduced by the thin layer created is smaller than the accuracy of the 3D scanner used. A set of experiments has been designed to demonstrate this and to compare CCD with common industrial whitening acquisition sprays: Helling Developer U 89 (referred as U89 in the tests), Helling Developer D 70 (referred as D70 in the tests) and OPN Developer White (referred as WHITE in the tests). To perform the proposed experiments, a Konica Minolta VIVID 900 has been used, with a $24mm$ lenses that provides an average accuracy of $600\mu m$.

First experiment consists on scanning a flat calibration chart and comparing the acquired point cloud with the optimal plane that better fits it. Residuals are expressed as signed distances between this plane and each point, and displayed together in an histogram. Average errors and standard deviations are calculated together with the two 2.5% intervals containing the worst samples (one for positive distances and another for negative ones).

The proposed experiment has been performed without applying any whitening spray (so the error distribution of the sensor can be characterized), with the 3 commercial whitening sprays commented before and with cyclododecane spray. For this experiment and the next ones, a colorimeter has been used in order to have fair comparisons: spray is applied on the surface, until an specific white tone is achieved.

Results of this experiment, shown in Figure 4.3, reveal no significant differences between using or not using any spray, and between cyclododecane and the other commercial solutions. As it can be appreciated, 95% of the samples are always in a range almost centered in 0, and with $600\mu m$ of size, which makes sense given the scanner's accuracy.

Second experiment uses more complex shapes to evaluate each sprays' performance. Given that, in this case, no optimal surface can be approximated, residuals are calculated in a different way: starting from an opaque object, a ground-truth model is acquired. Without touching the object and leaving the scanner focus fixed, the spray is applied until the desired level of white is achieved. Then, the model is acquired again, and both point-clouds are registered (because, even not focusing the scanner again, the focal length changes between acquisitions). Residuals are expressed as point-to-point distances between closest neighbors in both point clouds. To prevent measurement errors, each object has been scanned three times with each spray, and no object has been re-used after each test. This way, 12 identical replicas of each object have been used: after the acquisition with one spray, the object is discarded for further tests, so residuals remaining in the surface do not alter next measurements. Figures 4.4, 4.5 and 4.6 show the achieved results.

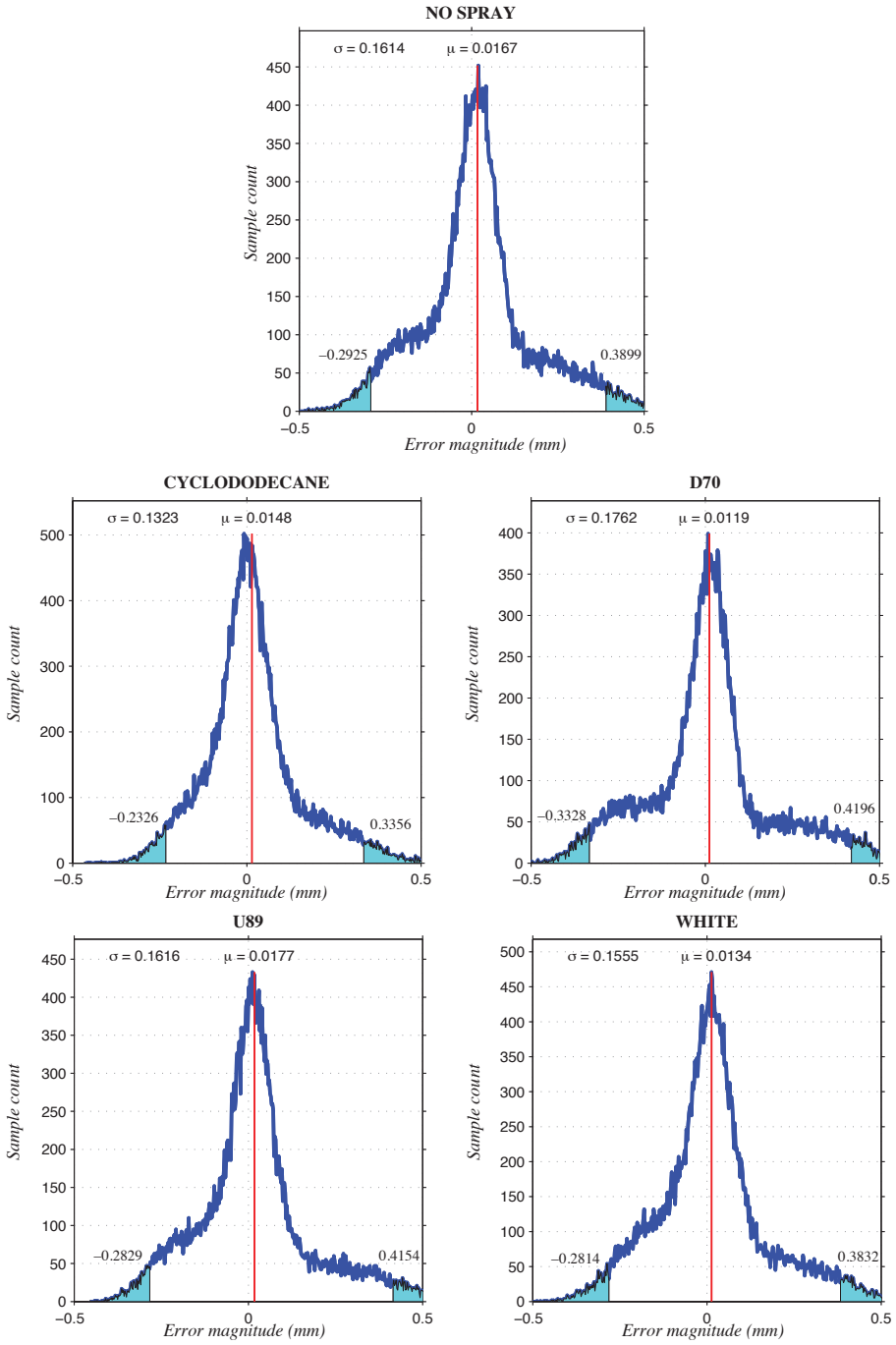

Figure 4.3: *Residuals for the calibration chart after applying each spray, with respect to the optimal plane. The red line indicates the signed average error. Blue intervals on both sides represent (each one) a 2.5% of the total acquired samples.*

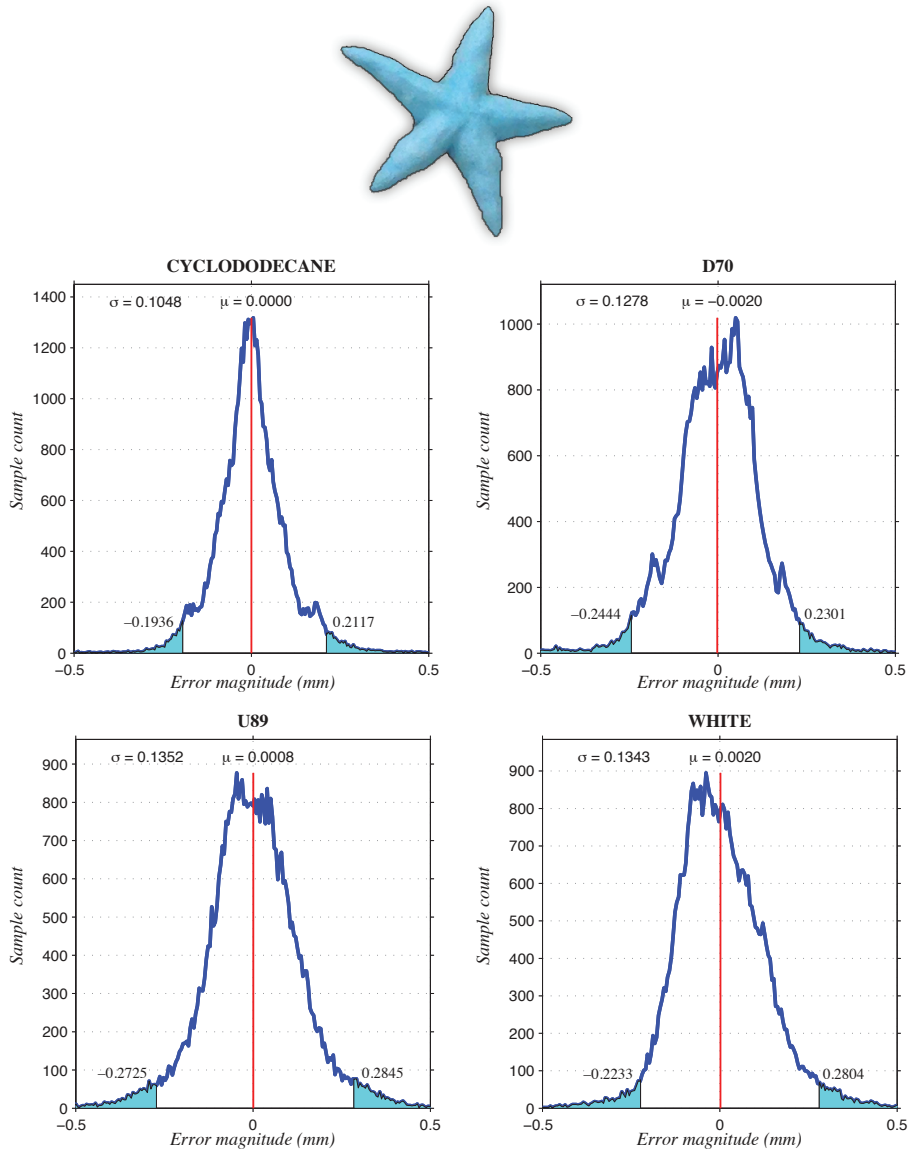

Figure 4.4: *Residuals for model ES after applying each spray, with respect to the same model/pose before appliying the spray, and properly registered. The red line indicates the signed average error. Blue intervals on both sides represent (each one) a 2.5% of the total acquired samples. This model presents an irregular shape with no curvature on the surface. As it can be noticed, all four sprays fall into the* $600\mu m$ *range of the scanner and cyclododecane performs even better than the others.*

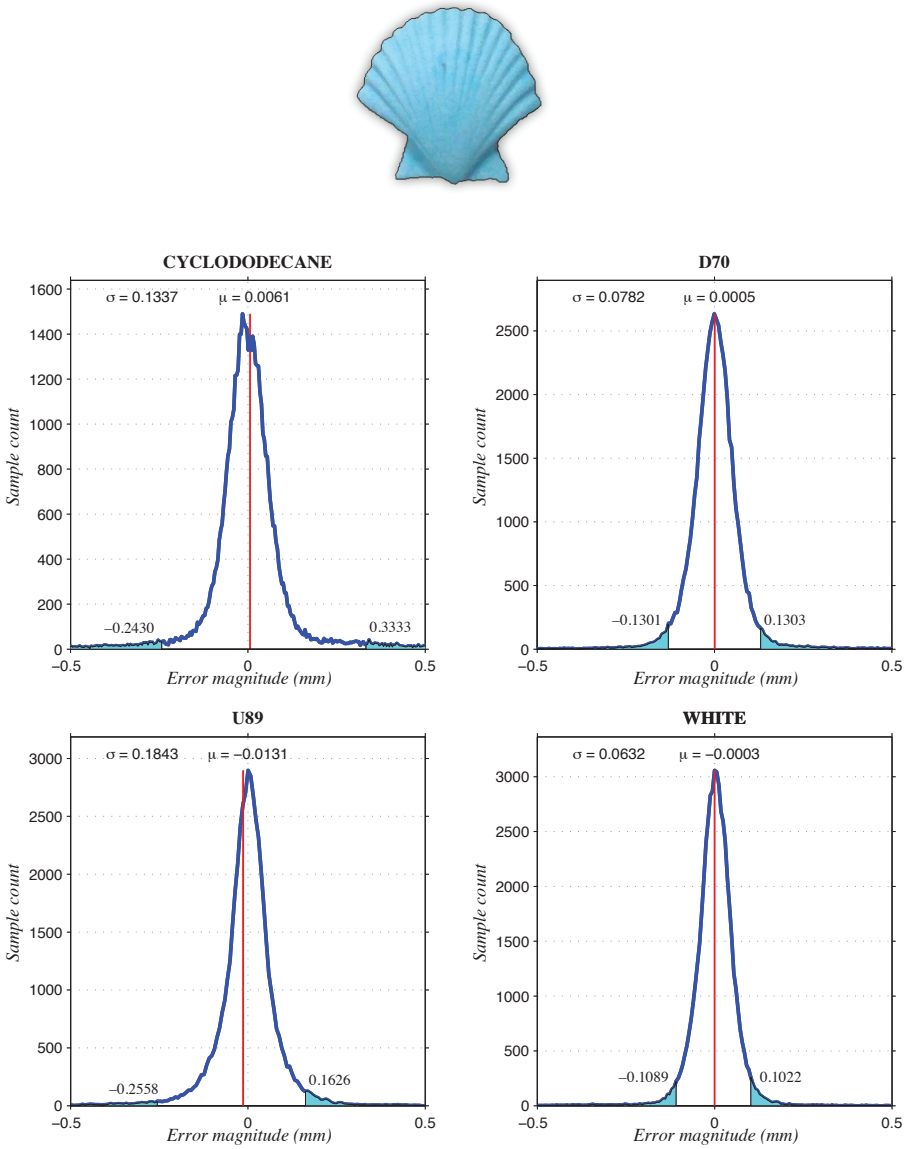

Figure 4.5: *Residuals for model CO after applying each spray, with respect to the same model/pose before appliying the spray, and properly registered. The red line indicates the signed average error. Blue intervals on both sides represent (each one) a 2.5% of the total acquired samples. This model presents an irregular shape with some smooth curvature on the surface. In this case results are more accurate than in the previous one, being cyclododecane slightly worst than the alternatives, but always inside the scanner's accuracy range.*

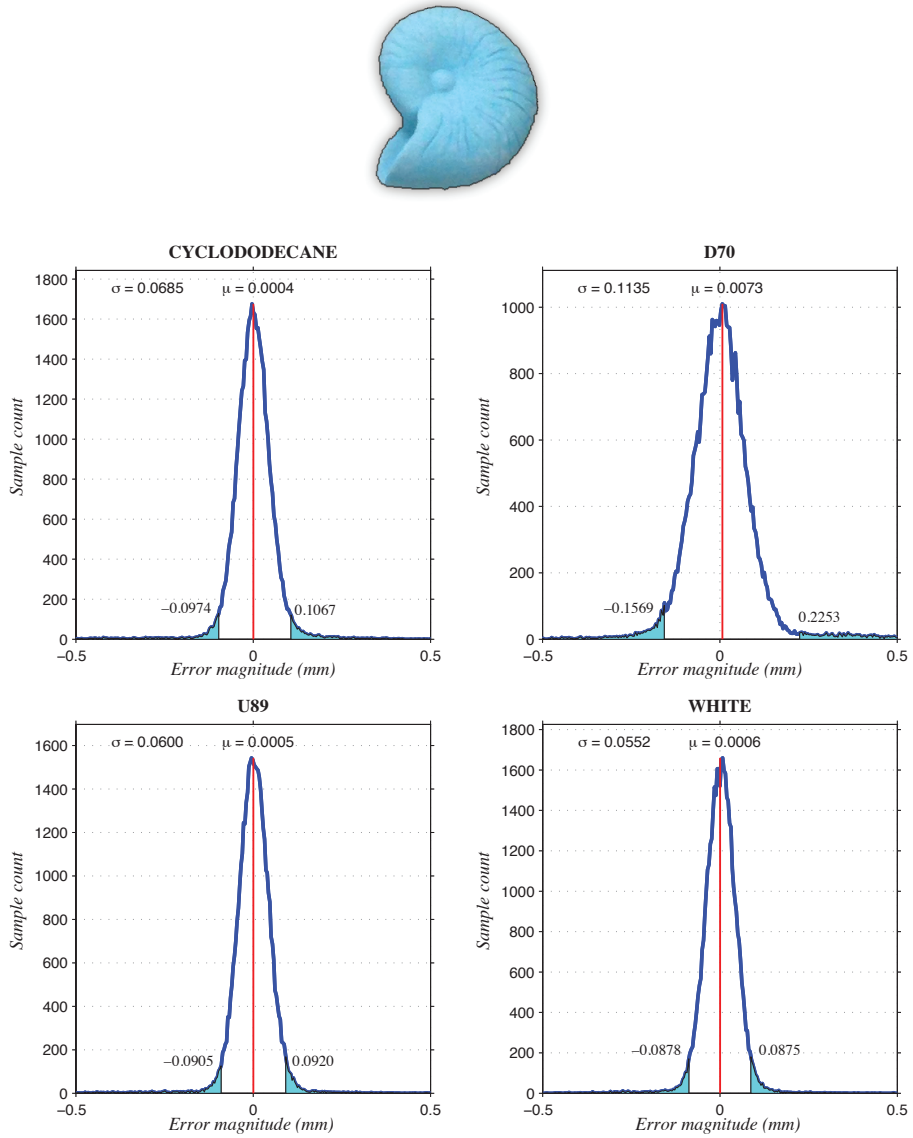

Figure 4.6: *Residuals for model MO after applying each spray, with respect to the same model/pose before appliying the spray, and properly registered. The red line indicates the signed average error. Blue intervals on both sides represent (each one) a 2.5% of the total acquired samples. This model presents an irregular shape with strong incisions on the surface. As happened with previous objects, all four sprays present an error distribution very similar, and inside the scanner's accuracy range.*

4.2.2 Electronic Microscope Analysis

As previous results suggest, all four compared products are suitable for acquiring 3D models of archaeological fragments. However, a simple visual inspection of the surface of the fragment after applying the spray reveals differences between cyclododecane and the other products: the white film created by cyclododecane shows more irregularities than the other sprays, suggesting a bigger particle size.

To confirm this, and measure the differences, a set of pictures have been taken using an electronic microscope. In Figure 4.7 it can be appreciated how cyclododecane particles are considerably bigger than the other products. However, a closer look to the 5.000 augments picture of cyclododecane (top-right image) clearly shows that the particle size is around $10\mu m$, which is one order of magnitude smaller than current laser triangulation scanners' accuracy. This observation confirms previous experiment results, and explains the empirical visual differences between cyclododecane and the other sprays.

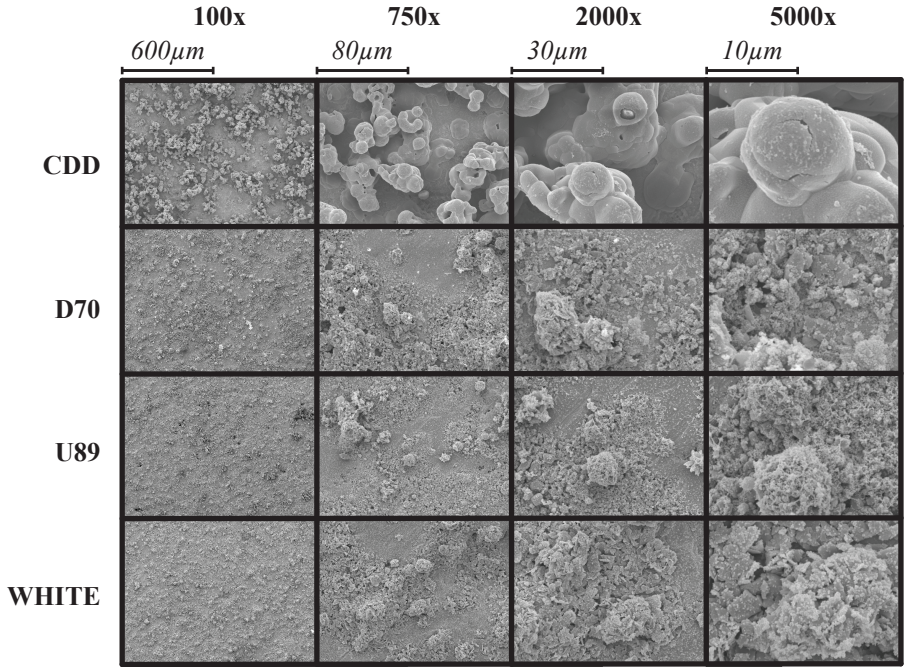

Figure 4.7: *Electronic microscope comparison of whitening sprays at different scales. Notice how, even having a bigger particle size than the other sprays, cyclododecane is still small enough to not interfere with laser scanners.*

4.2.3 Sublimation speed

In order to estimate the effective time in which CCD can be used as opacifier for acquisition purposes, two objects have been selected: a big flat glass with some roughness on its surface and a glass pot with more pronounced reliefs and irregularities. Both objects have been sprayed and scanned several times in a fixed pose.

Taking as reference model the point cloud acquired immediately after applying the spray, subsequent acquisitions have been compared against it, focusing on two different estimators for the sublimation speed: the number of valid measures returned by the scanner and the average deviation with respect to the reference model.

On average, the total amount of CCD applied to each object in order to ensure a good acquisition has been estimated as $7,998mg/cm^2$.

Results presented in Fig. 4.8 show how, after the first hour since the spray was applied, the total number of valid points do not change significantly (98.44% of the original points are still present in the 'Pot' case and 98.41% in the 'Flat glass' case), whilst average acquisition deviations are always very close to 0 (0.0526mm for the 'Pot' and 0.0052mm for the 'Flat glass'). During the next four days, the average error does not change significantly, whilst the number of valid measurements gradually decreases, remaining present 77.79% of the original points for the 'Pot' and 59.44% for the 'Flat glass' at the end.

It can be noticed how sublimation speed changes considerably from one case to the other: in the case of the 'Flat glass', since the air exchange over a big flat surface is greater than in an irregular object as the 'Pot', the evaporation process gets considerably accelerated. However, both objects were perfectly scanned during the first 75 minutes after the application of the spray which, from our experience, provides plenty of time to perform the acquisition.

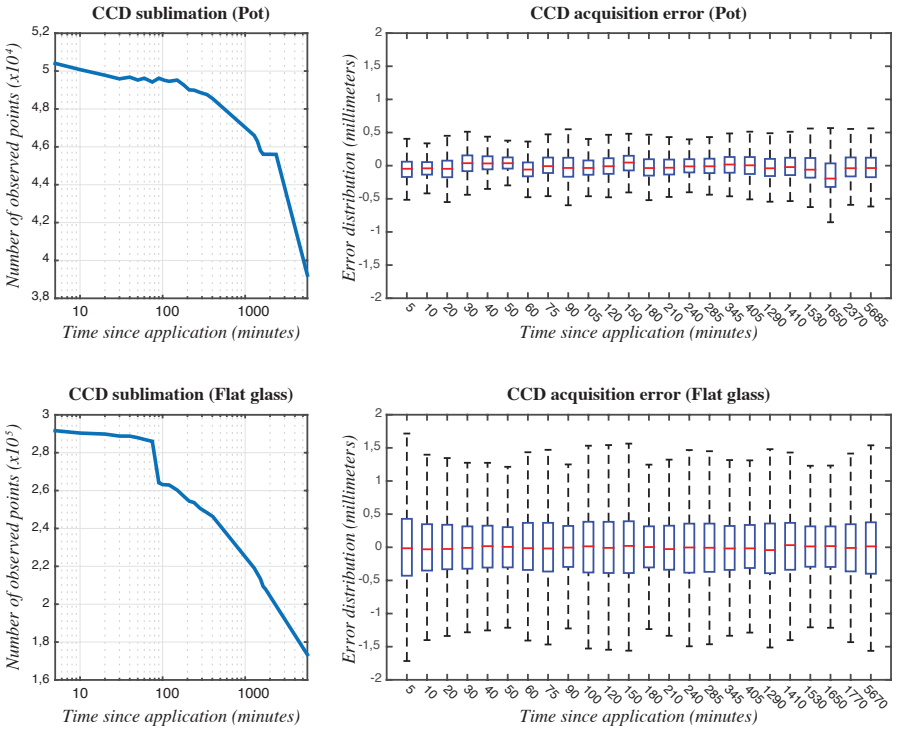

Figure 4.8: *Evaluation of the sublimation speed of CCD. Left charts show the number of valid measures over time with respect to the reference model (acquired immediately after applying the spray), whilst right charts show the error distribution over time.*

4.2.4 Conclusion

According to all previous experiments, the advantages arising from cyclododecane's chemical stability and the fact that it sublimes at ambient temperature make it a perfect candidate for the acquisition of reflective/refractive archaeological artifacts, even taking into account that its particle size is bigger. Also, it is interesting considering that it is a relatively cheap product (cheaper than whitening sprays), and that its toxicity is very low.

Figure 4.9 shows the results achieved when scanning a set of glass artifacts using and not using CCD as a whitening spray.

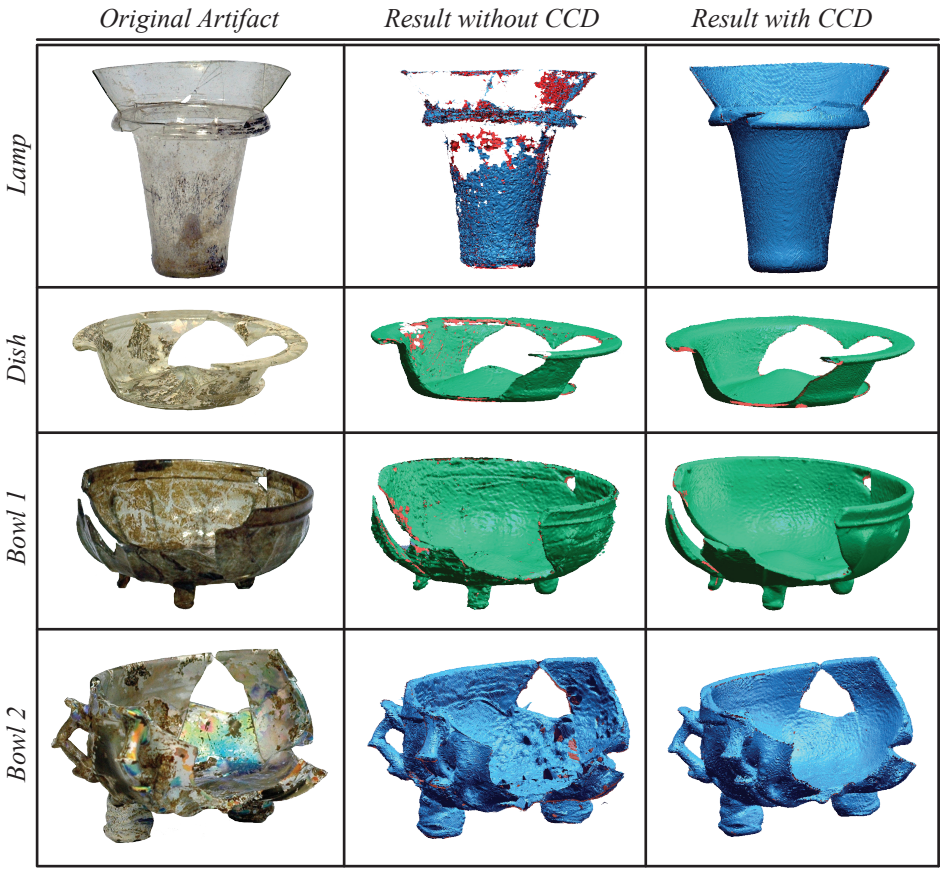

Figure 4.9: *Results achieved using and not using CCD as a whitening spray.*

In Figure 4.9 it can be appreciated how, for the topmost two artifacts ("Lamp" and "Dish"), transparent areas are not visible to the scanner when CCD is not used and, consequently, the resulting model presents lots of holes. Also, in the "Lamp" model and in "Bowl 1" and "Bowl 2", the areas opaque enough to be acquired present lot of noise, due to the reflections/refractions that happen when the laser beam hits the surface. Notice how the use of cyclododecane allows capturing the whole model and reduces considerably the noise introduced by this phenomenon.

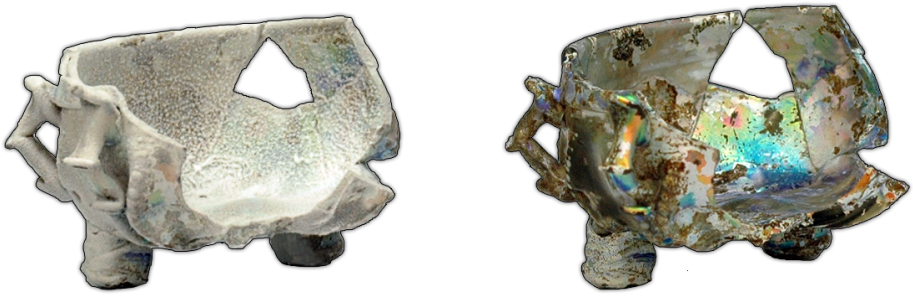

Figure 4.10: *The artifact "Bowl 2" with cyclododecane applied (left), and after the (unaccelerated) sublimation process (right). Notice how no trace of the spray remains on the surface of the object. The sublimation process took 36 hours after acquisition.*

Figure 4.10 shows one of the artifacts sprayed with cyclododecane for its acquisition, and the same artifact after the white film has completely sublimated, without rubbing or using solvents to remove it.

The development of these experiments has been done in collaboration with restoration experts from the IRP, (Universitat Politècnica de València) and it is enclosed inside the national research project supported by the "Plan Nacional de I+D+i 2008-2011" from the Ministerio de Economía y Competitividad of Spain, Project ID: HAR2012-38391-C02-01, HAR2012-38391-C02-02.

CHAPTER 5

3 Degrees of Freedom Approach

This chapter deals with the automatic reconstruction of flat archaeological artifacts. These are very common in archaeological sites and have the advantage of constraining the search space to three degrees of freedom: one associated to orientations, and two to translations. One of the most typical examples in this kind of problems are frescoes, whose mural paintings and surface engravings are of huge importance in the field of Cultural Heritage.

The proposed technique takes as input data two 2D/3D digital models of flat fragments and outputs the rigid transformation that maximizes the contact area between their surfaces. Results achieved always ensure that, for the calculated alignment, there are no penetrations between fragments and that the solution corresponds to the global minima.

Of course, depending on erosion, the best match is perhaps not the real solution. In this case, it is up to the user to discard the alignment and ask the system to return alternative matches. This way, the goal of the proposed technique is not solving the global puzzle, but providing a fast/robust algorithm to perform the necessary intermediate comparisons between pairs. Once these pairwise alignments have been calculated, an interactive semi-automatic tool uses them to assist the restoration personnel in the final reconstruction.

The reason for not facing the global problem is that it is known to be NP-complete [5] [46], which gets even harder when considering the extra ambiguity introduced by erosion. Also, error accumulation when re-assembling the original object increases as the number of fragments grows (even using relaxation techniques). This, eventually makes true and false matches indistinguishable, limiting the problem size fully-automated assembly systems can handle in practice [26].

Major contributions of this chapter are:

- A discrete characterization of fragments and alignments that allows performing all heavy geometric operations by the GPU on a pre-processing stage, and ensures data alignment.
- A hierarchical characterization of fragments.
- A hierarchical search strategy with a very high performance.
- An optimistic cost function that ensures convergency and global correction of the hierarchical search strategy, that operates on the hierarchical characterization of the fragments.

5.1 Overview

The proposed technique faces all the stages involved in the reconstruction process (Fig. 5.1). The final goal is to provide the potential matches between pairs to an interactive semi-automatic tool that will represent them graphically, assisting the restoration personnel in the decision making process, and accelerating the re-assembly of the original artifact.

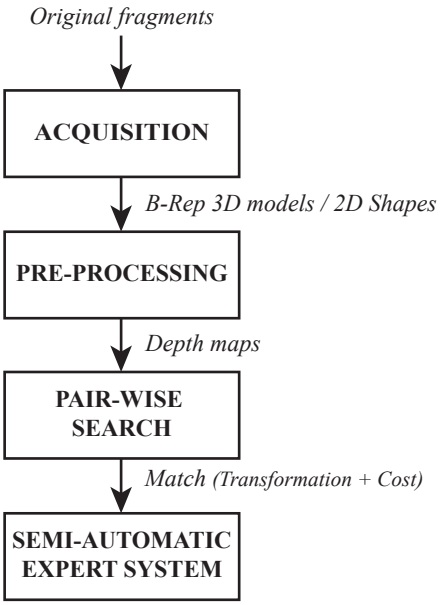

Figure 5.1: *3DOF technique overview*

Starting from the original fragments, first stage deals with the acquisition into a digital format. Depending on fragments' nature and, mostly, their thickness, a 2D or 3D acquisition device will be used. For thick fragments, a 3D B-rep model will be generated and, for thin ones, a vectorial 2D contour shape. Previous chapter covered this stage.

Taking as input data these vectorial representations of the fragments, a set of discrete GPU-computed hierarchical depth maps will be generated in the pre-processing stage (Section 5.3). The discrete and hierarchical nature of this alternative characterization will accelerate the search process in the next stage.

From the hierarchical depth maps of two fragments generated previously, the pair-wise search stage finds the rigid transformation that maximizes the contact area between them while preventing penetration. The result of this stage is the transformation matrix that stores this information, together with a quality measure. Section 5.4 covers this process, starting from an exhaustive approach and ending into a fully hierarchical one.

To accelerate the final expert system, Subsection 5.4.5 introduces a small modification in the one-to-one hierarchical search algorithm that allows taking advantage of the proposed search strategy in many-to-many comparisons.

In Section 5.5 achieved results with each technique are presented and compared and, in Section 5.6, major conclusions are highlighted and future improvements are introduced.

5.2 Cost Function and Solution Space

To compute the cost function of a given alignment, and for the proposed search strategy, fragments are assumed to have their upper and lower faces aligned to the $X - Z$ plane, being the lower face laying on the $Y = 0$ plane. This alignment is normally performed manually during the data acquisition stage, but it can also be automatized by segmenting data as explained in [71], and applying least squares to infer the optimal planes that contain the upper and lower faces.

The quality of an alignment is evaluated by using the LCP (*Largest Common Pointset*) metric, which is widely extended in matching techniques and can be defined as:

$$LCP(P,Q) = \sum_{p \in P} Match(p,Q) \tag{5.1}$$

$$Match(p,Q) = \begin{cases} 1, & \exists q \in Q, \|p-q\| < \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

where P and Q are two fragments, p is a sampled point in P, q is a sampled point in Q, and ϵ is a tolerance given by the user that indicates if two points are close enough to be considered as matching. The value of ϵ depends on how eroded the compared fragments are, and has to be empirically estimated using known matches. Normal values are in the range of $[0.1\%..2\%]$ of the average fragment size.

To efficiently calculate the corresponding point q that is aligned to a point p, both fragments are sampled uniformly using a parallel projection plane, Π, which is perpendicular to the $X - Z$ plane. This way, the two degrees of freedom associated to translations are expressed according to the cartesian system (u, v) defined by the projection plane. Distances between facing samples in both fragments can be easily calculated as the sum of their individual distances (v values) to the projection plane. Fig 5.2 illustrates this concept.

Despite being a three degrees of freedom problem, the non penetration constraint that has to be satisfied makes that, given a relative rotation between fragments, only u translations are independent: the value of v can then be estimated as the translation that has to be applied to fragment Q, so both fragments' surfaces get in contact.

In order to keep samples aligned, the total amount of discrete displacements over axis u is determined by the sampling resolution (n) of the fragments, whilst the value of v remains continuous. With respect to orientations, the solution space is discretized into a set of m equispaced intervals, being $m = 2^k, k \in \mathbb{N}$. This assumptions leave a solution space with two discrete axes (orientations and u displacements) and a continuous one (v displacements) that is dependent on the other two dimensions.

The proposed solution space make this approach suitable to solve the same problems faced in [24]. The main difference with this technique is that here the total alignment error is optimized by minimizing a cost function that considers the entire edge of the fragment, instead of a local patch. This allows performing a hierarchical search strategy that increases performance, and that would not yield into a big speedup in Brown's.

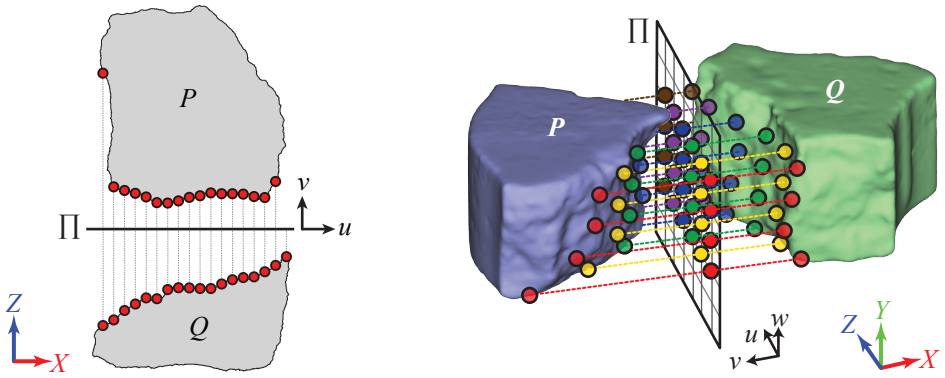

Figure 5.2: *Proposed cost function for 2D problems (left) and 3D problems (right). Dots represent the samples on the fragments' surface, colored in the 3D version to facilitate the understanding of the ilustration. Lines connecting samples represent the distance between the projection plane and the surface of the fragment. 3D models are courtesy by Vienna University of Technology.*

5.3 Pre-Processing

During the search process lots of comparisons between pairs of fragments have to be done, so it is convenient the evaluation cost to be minimum. Such cost is associated to the intrinsic operations performed, that can be classified as:

- Geometric: translations, rotations and projections.
- Visibility: self-occlusions in fragments.
- Discretization: uniform sampling of the fragments.
- Comparison between samples.

Notice how geometric, visibility and discretization operations depend only on the topology of each fragment, whilst comparison operations depend on which pair of fragments are compared and the specific alignment evaluated. It is also important to notice

that, while the first kind of operations require hard computing (including geometric transformations, visibility tests and discretization operations) the second can be performed using simple additions, as commented before.

These ideas lead to including a pre-processing stage to deal with all these costly operations that only depend on each fragment. Results achieved in this stage can be used in as many comparisons as needed in further searches. This way, the search process can be performed by using only simple operations over pre-processed aligned data.

To pre-calculate the projective distances of fragments, the proposed technique uses GPU *(Graphic Processor Unit)* computing capabilities, which makes it very similar to the approach presented in [123]. The main difference with this technique is that, in this case, fractured faces are not assumed to be nearly planar and match each other completely. This way, a major number of cases have to be considered, making the proposed approach suitable for more general problems.

Since nVidia first introduced T&L (Transform Clipping and Lightning) technology in 1999, graphics hardware have specific units embedded in its pipeline that efficiently perform all the geometric transformations needed. The use of depth buffers, combined with stencil buffers, allows GPUs performing visibility tests in screen-space with hardware acceleration and, given the discrete nature of generated images, the rasterization unit performs all the heavy calculations needed to get uniform samples over the surface of the fragment. Figure 5.3 shows the hardware architecture of modern GPUs.

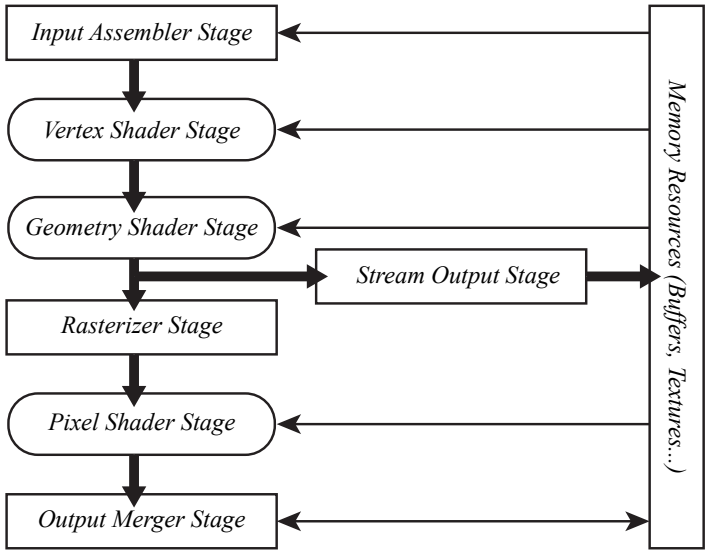

Figure 5.3: *GPU architecture for the Shader Model 4.0 standard. Rounded boxes are programmable units. Vertex Shader Stage deals with T&L operations. The rasterizer stage deals with discretization and depth tests are evaluated before rasterizer stage and after pixel shader stage.*

Given these advantages, the way to measure distances to the surface of the fragment lies in calculating the orthographic projection matrix defined by the angle of the plane, and rendering the depth buffer into a frame buffer object. The resolution of the depth

buffer used determines the number of samples calculated, and the precision of measures is considerably high (up to 32 bit float values of depth, distributed linearly on the visibility frustum). Figure 5.4 shows the result of measuring distances using a GPU-accelerated parallel projection.

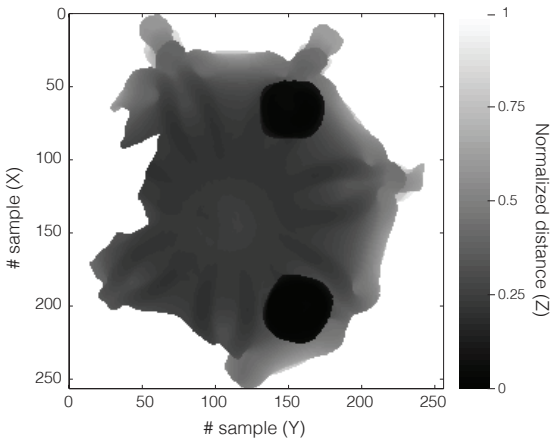

Figure 5.4: *GPU computed depth-map. Depth values obtained using the GPU, expressed in normalized distances with respect to the projection plane. Points outside the 3D model are represented with distance 1.*

For 2D approaches, the contour extracted from the acquired image is extruded along the Y axis and the vertical resolution of the computed depth buffer is one pixel. For 3D approaches, the acquired B-Rep model is used, and the depth buffer resolution ratio is forced to one (with respect to the object's dimensions), so surfaces are sampled with the same frequency in both axes. Figures 5.5 and 5.6 illustrate the measuring process carried out in the pre-processing stage for both, 2D and 3D cases.

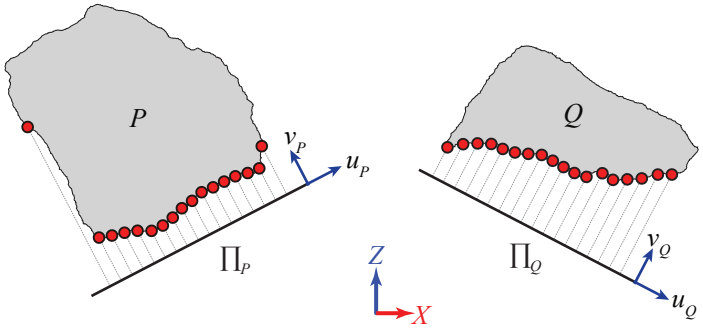

Figure 5.5: *2D Pre-Processing of the fragments compared in Fig. 5.2 (left) for two given orientations (represented as the projection plane orientation).*

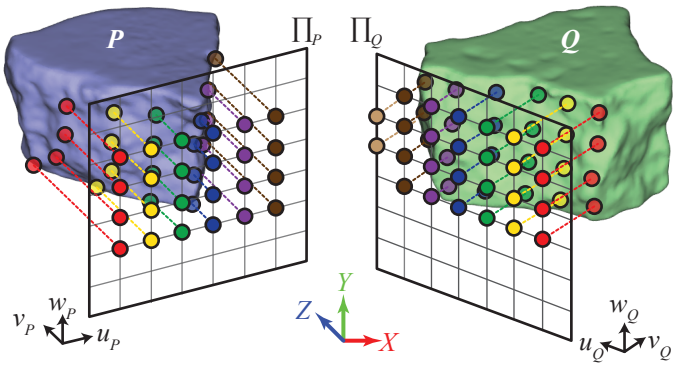

Figure 5.6: *3D Pre-Processing of the fragments compared in Fig. 5.2 (right) for two given orientations (represented as the projection plane orientation).*

5.4 Search Strategy

Having a proper cost function defined and the solution space characterized, the search stage deals with finding, as fast as possible, the best solution amongst all.

Subsection 5.4.1 introduces a basic exhaustive strategy whose main disadvantage is performance. To speedup the search process, a hierarchical strategy is introduced in subsections 5.4.2, 5.4.3 and 5.4.4, which faces the main bottlenecks of the exhaustive approach: orientations, displacements and cost function evaluation, respectively.

Using exhaustive search results as ground truth, hierarchical techniques accelerate the process preventing correction loss and ensuring that the search always converges to the optimal solution without stopping in local minima.

5.4.1 Exhaustive Search

Exhaustive approach, also known as *naive search*, is the most basic approach. Its main advantage is that it always produces the correct results, avoiding local minima. On the other hand, its main disadvantage is related to performance: it has to evaluate all possible alignments between fragments to find the best match.

An *Alignment*, A, and its associated cost function, $c(A)$ are defined as:

$$A(\theta_P, \theta_Q, \delta_u) \tag{5.3}$$

$$c : \mathbb{R} \times \mathbb{R} \times \mathbb{Z} \rightarrow \{x \in \mathbb{N} : x > 0\} \tag{5.4}$$

where θ_P and θ_Q in (5.3) correspond to the orientation of both fragments, and δ_u corresponds to the relative displacement along the u axis, defined by the projection plane, and illustrated in Fig. 5.2. The other displacement, δ_v, will be handled when deriving the matching between two projective planes, as explained below and in Fig. 5.7. The cost function detailed in (5.4) takes as input data an alignment, and returns a natural number

greater than 0 that corresponds to the amount of matching samples in both fragments that are close enough to be considered as matching.

Given that all possible alignments have to be compared, the total amount of evaluations of $c(A)$ can be expressed as:

$$\mathcal{O}(m,n) \approx m^2 * (2n-1) \tag{5.5}$$

where m is the number of orientations considered, and n is the number of samples.

The $2n-1$ term in (5.5) represents the total amount of displacements that can be performed, in order to keep samples from both fragments aligned. The m^2 term in (5.5) means that, for every orientation in fragment Q, every orientation in fragment P has to be compared. However, notice how there are lots of pairs of orientations which produce the same results ($\theta_P = \theta_Q + b$).

To reduce the number of combinations calculated, it can be assumed that one fragment (P), is only studied for k equi-spaced orientations, whilst the second (Q) is studied for the whole m orientations, being $k < m$. The value of k has to be great enough to fully characterize P, but low enough to simplify the search process. This way, k is not related to the size of the fragment, but to the *convexity* of its surface: the more self-occlusions on the surface of the fragment, the bigger value of k. In practice, values for k in the range $[6..16]$ have provided results equivalent to $k = m$.

The final amount of alignments to be evaluated can, then, be expressed as:

$$\mathcal{O}(m,n,k) \approx (m * k) * (2n-1) \tag{5.6}$$

To speedup the cost function evaluation, each fragment P is characterized as a set of distance matrices $P_{\theta_k} = [p_{i,j}]_{|u|\times|w|}$, where θ_k is the orientation of the projection plane, $|u|$ and $|w|$ represent the sampling resolution over axes U and W, respectively, and each element $p_{i,j}$ stores the calculated distance between the $(i,j)^{th}$ sample and the projection plane Π_{θ_k}. Orientation values of θ_k are defined as: $\theta_k = k * 2\pi/m$, being m the discretization resolution on rotations.

To generalize equations for both, 2D and 3D cases, it can be considered that, in the 2D case, $|w| = 1$, so the third dimension of the bidimensional fragment is discarded. This will be assumed during the entire chapter.

Since alignment between samples is ensured by the proposed characterization, given two fragments (P, Q) and a discrete displacement of fragment Q over U axis (δ_u), the sample facing $p_{i,j}$ is $q_{i-\delta_u,j}$.

$$c(A) = \sum_{i=0}^{|u-1|} \sum_{j=0}^{|w-1|} Match(p_{i,j}, q_{i-\delta_u,j}) \tag{5.7}$$

$$\delta_v = Min(p_{i,j} + q_{i-\delta_u,j}) \tag{5.8}$$

$$Match(p_{i,j}, q_{k,j}) = \begin{cases} 1, & p_{i,j} + q_{k,j} - \delta_v \leq \epsilon \\ 0, & \text{Otherwise} \end{cases} \tag{5.9}$$

To calculate the cost function of an alignment (5.7), the maximum displacement δ_v of fragment Q over V axis that produces contact between the surfaces of fragments (5.8) has to be calculated firstly. Then, given the calculated δ_v, for each pair of aligned samples

$p_{i,j} \in P$ and $q_{i-\delta_u,j} \in Q$, it has to be evaluated if they are close enough to be considered as matching (5.9), where ϵ is the tolerance introduced by the user.

Notice how the distance between the projection plane and the surface of the fragment does not affect the technique, given that δ_v ensures that contact between both fragments always happens. Figure 5.7 illustrates the calculation of δ_v and $c(A)$.

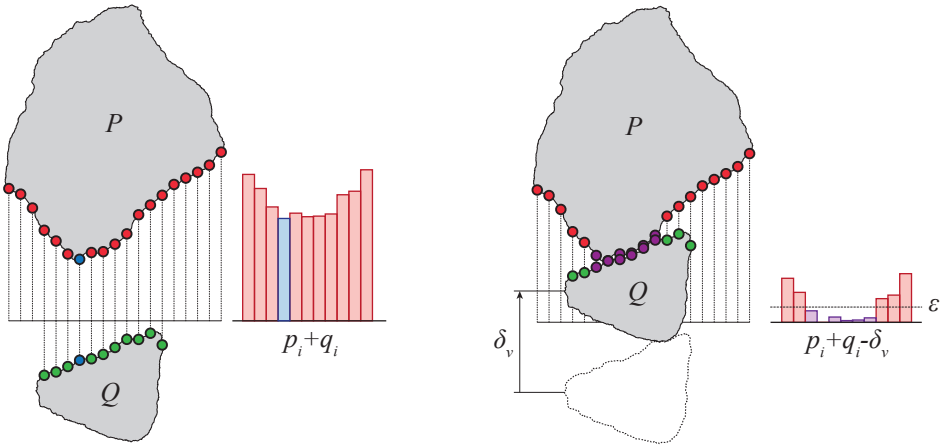

Figure 5.7: *Cost function evaluation for two centered fragments ($\delta_u = 0$). (left) δ_v calculation stage. Minimum distance is marked as blue samples. (right) Cost function calculation applying the δ_v displacement. Matching samples, according to ϵ, are marked in purple.*

Algorithm 1 shows the pseudo-code to perform an exhaustive search.

Algorithm 1 Exhaustive search

```
1:  best ← null;
2:  for θ_P ← 0 to 2π step 2π/k do
3:      for θ_Q ← 0 to 2π step 2π/m do
4:          for δ_u ← −|u| + 1 to |u| − 1 step 1 do
5:              A ← Alignment(θ_P, θ_Q, δ_u);
6:              if c(A) > c(best) then
7:                  best ← A;
8:              end if
9:          end for
10:     end for
11: end for
12: return best;
```

As Algorithm 1 shows, in order to return the best alignment, all possible solutions have to be evaluated and compared with the previous ones. This forces to explore the whole solution space, making the approach very inefficient. Fig. 5.8 shows all the values for the cost function during the exhaustive search over the alignment displayed in Fig. 5.16,

considering that fragment P (represented in blue) is fixed in the orientation displayed, and fragment Q (represented in green) is the one that moves.

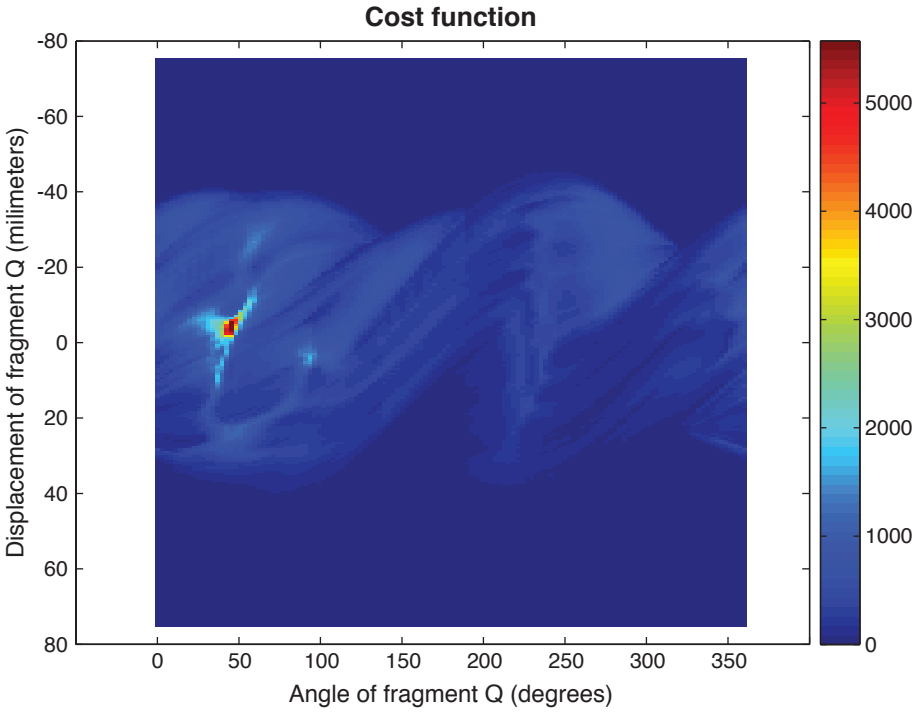

Figure 5.8: *Solution space and cost function results for all posible orientations and displacements of the alignment represented in Fig. 5.16.*

5.4.2 Hierarchical Orientations

The goal of this optimization is to reduce the computational cost associated to the $(m * k)$ term shown in (5.6), that is caused by the exhaustive exploration of orientations in the solution space. By introducing a binary hierarchical search, it is intended to achieve an execution cost of $\mathcal{O}(m) \approx log_2(m)$ for the best case execution scenario.

A hierarchical fragment characterization is proposed, based on the concept of *Levels Of Detail* (LOD), that can be defined as:

1. Each fragment is characterized as a set of $log_2(m) + 1$ LODs.
2. Each LOD_x covers the full 2π orientations, and is made up of 2^x distance matrices.
3. Each distance matrix $P_{[\theta_1..\theta_2[} = [p_{i,j}]_{|u|\times|w|}$, of a given LOD_x covers $2\pi/2^x$ orientations.
4. Each element of the distance matrix $p_{i,j}$ stores two values: the maximum and minimum projective distance of the $(i,j)^{th}$ sample in the covered angular range, which we notate as $\lceil p_{i,j} \rceil$ and $\lfloor p_{i,j} \rfloor$, respectively.

Fig. 5.9 illustrates the proposed characterization for a given orientation interval.

This way, it can be said that LOD_1 refines LOD_0 with two distance matrices $P_{[0..\pi[}$ and $P_{[\pi..2\pi[}$, which are called *children* of $P_{[0..2\pi[} \in LOD_0$, and provide a complete and disjoint partition of their *father*. Fig. 5.10 illustrates the hierarchical refinement proposed. Notice how, since sampled data stores the maximum and minimum distance from the projection plane to the surface of the fragment in the covered orientations, children representations always present lower or equal maximum distances and greater or equal minimum distances than the parent representation. This way, the amount of uncertainty decreases as the representation of the fragment is refined.

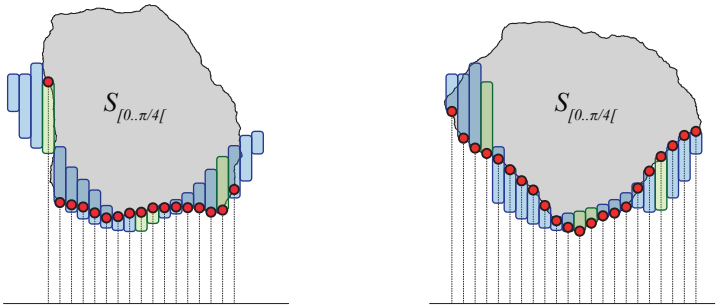

Figure 5.9: *Graphic representation of a distance matrix for a fragment in a coarse orientation LOD. Red points represent the calculated distances in the orientation shown. Blue and green boxes represent the minimum and maximum distance for each sample in current orientation interval. Boxes marked in green have in (left) and (right) the samples that produce the minimum and maximum distance. In (left) is shown the minimum angle of considered interval, whilst in (right) is shown the maximum one.*

This idea is particularly useful for the memory issue on exhaustive approach: the coarsest LODs are the most time consuming to calculate, since they need to compute all possible orientations contained in the covered range. However, these LODs store lots of information in a few distance matrices. As the search evolves, uncertainty decreases, and only several orientations present a potential match. This way, pre-calculations can be done only for the first k LODs. The potentially useful distance matrices of the finest LODs are calculated on the fly by the search algorithm and, considering the advantage that they contain only a few possible orientations, these calculations are executed extremely fast.

The proposed hierarchical search starts by comparing coarse representations of fragments and, recursively, refines the most promising ones until the finest Level Of Detail is reached. When this happens, the search stops returning the resulting alignment as the result. For the technique to converge to the global solution, avoiding local minima, the cost function of each alignment in an intermediate LOD has to be evaluated in an optimistic way. This makes necessary to extend equations (5.3) (5.4) (5.8) (5.9) as follows:

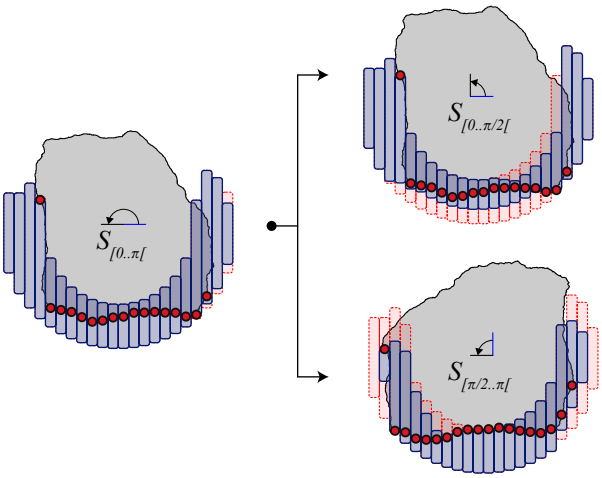

Figure 5.10: *Hierarchical orientation refinement. In (left) is shown the distance matrix $S_{[0..\pi[} \in LOD_1$, whilst in (right) are shown the two refined children, $S_{[0..\pi/2[}$ and $S_{[\pi/2..\pi[} \in LOD_2$. Marked in red, in (right), are the samples calculated in the parent representation. Notice how uncertainty always decreases as the resolution increases.*

$$A\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u\right) \tag{5.10}$$

$$c : \mathbb{R} \times [\mathbb{R} \dots \mathbb{R}] \times \mathbb{Z} \rightarrow \{x \in \mathbb{N} : x > 0\} \tag{5.11}$$

$$\delta_v = Min\left(\lceil p_{i,j} \rceil + \lceil q_{i-\delta_u,j} \rceil\right) \tag{5.12}$$

$$Match(p_{i,j}, q_{k,j}) = \begin{cases} 1, & \lfloor p_{i,j} \rfloor + \lfloor q_{k,j} \rfloor - \delta_v \leq \epsilon \\ 0, & \text{Otherwise} \end{cases} \tag{5.13}$$

From (5.10) it can be appreciated that now, we compare a unique orientation of fragment P with a range of orientations of fragment Q, which correspond to a distance matrix of a given LOD of Q. The cost function (5.11), as it did before, evaluates how many samples are close enough to be considered as matching. To do so, first it calculates δ_v using maximum projection distances (5.12) and then, it evaluates the distance between facing samples using minimum projection distances (5.13).

As it can be appreciated in Fig. 5.11, samples in *children* distance matrices always have minimum distances greater or equal than their *father*, and maximum distances smaller or equal than their *father*. This means that the value of δ_v decreases as the representation is refined, since it uses maximum distances. In the other hand, since the value of the $Match()$ function depends on δ_v and the minimum distances, as the representation is refined the value returned by this function is monotonically decreasing. According to this property of the metric, it can be ensured that the search process converges to the optimal solution without stopping in local minima.

The search process starts by initializing a heap with k alignments: one for each k orientation of fragment P, facing fragment Q represented in LOD_0. The heap stores all evaluated alignments sorted in descending order according to their cost function. Each

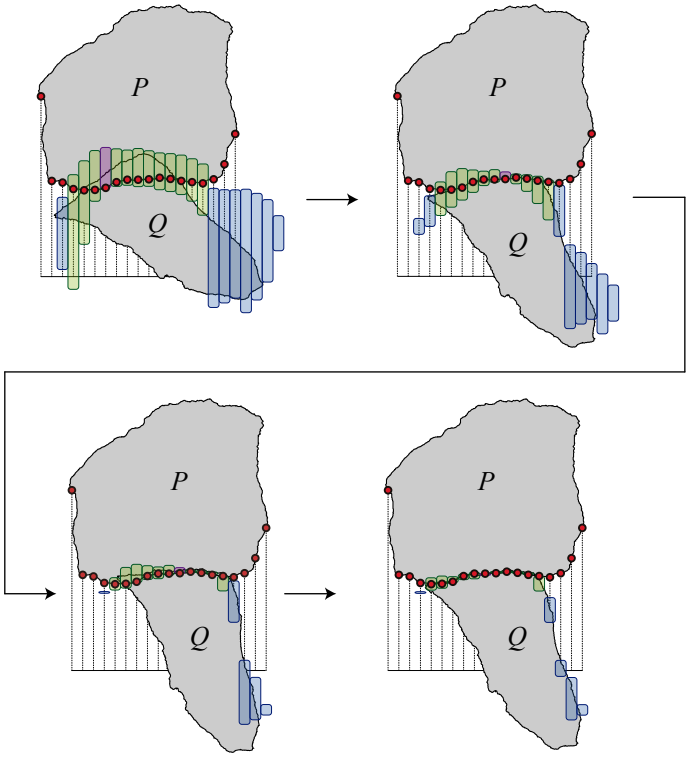

Figure 5.11: *Angular search process. Fragment Q is represented in the middle orientation of each distance matrix rotational interval. Purple boxes show the most restrictive distances in each case (which determines the value of δ_v), green boxes show matching samples, and blue boxes show non-matching samples. Notice how, in four iterations, ambiguity is highly reduced and the solution converges to the optimal one. Also notice how the value of δ_v decreases in each iteration, and how some previously matching samples fail the $Match()$ test in later alignments.*

iteration, the alignment (A) in the root of the heap (the one with the highest cost function value) is popped and refined into another two alignments in the next LOD as follows:

$$A\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u\right) \rightarrow \begin{cases} A_1\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{((b-a)/2)}\right], \delta_u\right) \\ A_2\left(\theta_P, \left[{\theta_Q}^{((b-a)/2)} \dots {\theta_Q}^{(b)}\right], \delta_u\right) \end{cases} \tag{5.14}$$

where $A \in LOD_x : A_1, A_2 \in LOD_{x+1}$

For each one of the two new alignments the cost function has to be evaluated in all possible displacements. The maximum score achieved is the one we use to insert each new alignment into the heap. This process continues until the root of the heap is an alignment in the finest LOD. In this case, the alignment is returned as the result of the problem. Fig. 5.11 shows a graphical representation of the proposed search process.

5.4.3 Hierarchical Displacements

In this case, the goal is to reduce the computational cost associated to the $(2n-1)$ term shown in (5.6), caused by the exhaustive exploration of displacements in the solution space, given the orientation of the fragments. By introducing a binary hierarchical search, an execution cost of $\mathcal{O}(n) \approx log_2(n)$ is intended to be achieved, for the best case scenario.

As happened with orientations, Levels of Detail are used to characterize fragments (Fig. 5.12) as follows:

1. Each distance matrix is represented as a set of $log_2(n)+1$ displacement LODs, being $n = Max(|u|, |w|)$.
2. Each LOD_x is a distance matrix of size $2^x \times \left\lceil \frac{2^x|w|}{|u|} \right\rceil$, if $|u| \geq |w|$, or $\left\lceil \frac{2^x|u|}{|w|} \right\rceil \times 2^x$ if $|w| > |u|$.
3. Each element $p(i,j)$ of the distance matrix in LOD_x stores three values: the maximum projective distance $\lceil p(i,j) \rceil$ of samples $\lceil p(k,l) \rceil \in LOD_{x+1}$, being $i*2 \leq k \leq i*2+1$ and $j*2 \leq l \leq j*2+1$, the minimum projective distance $\lfloor p(i,j) \rfloor$ of samples $\lfloor p(k,l) \rfloor \in LOD_{x+1}$, and the total amount of samples $|p(i,j)|$ contained in $|p(k,l)| \in LOD_{x+1}$.

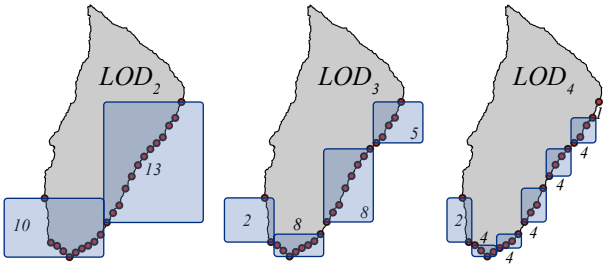

Figure 5.12: *Hierarchical displacements. A fragment represented in three diferent LODs. Red dots show samples at maximum resolution, whilst blue boxes represent samples in the illustrated LOD. Associated numbers to each box correspond to the value of* $|p_{i,j}|$.

Searching strategy for displacements is very similar to the one for orientations, and the same restrictions in the cost function have to be applied to ensure convergency: it has to be monotonically decreasing as LOD increases. To do so, equations (5.10) (5.11) (5.12) (5.13) have to be extended as follows:

$$A\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u, LOD_\delta\right) \tag{5.15}$$

$$c : \mathbb{R} \times [\mathbb{R} \dots \mathbb{R}] \times \mathbb{Z} \times \mathbb{Z} \rightarrow \{x \in \mathbb{N} : x > 0\} \tag{5.16}$$

$$\delta_v = Min\left(\lceil p_{i,j} \rceil + \lceil q_{i-\delta_u,j} \rceil\right) \tag{5.17}$$

$$Match(p_{i,j}, q_{k,j}) = \begin{cases} |p_{i,j}|, & \lfloor p_{i,j} \rfloor + \lfloor q_{k,j} \rfloor - \delta_v \leq \epsilon \\ 0, & \text{Otherwise} \end{cases} \tag{5.18}$$

In this case, notice how an alignment (5.15) now indicates the displacement, δ_u, and also the LOD in which this alignment is represented. As happened in (5.12), in (5.17) maximum distances are used to calculate δ_v whilst in (5.18), instead of adding single samples for each comparison in the $Match()$ function, $|p_{i,j}|$ is used.

Search process starts with a heap containing only one alignment, in which both fragments are represented in LOD_0. The heap stores all evaluated alignments sorted in descending order according to their cost function. Each iteration, the alignment (A) in the root of the heap is popped and refined into another three alignments in the next displacement LOD as follows:

$$A\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u, LOD_\delta\right) \rightarrow \tag{5.19}$$

$$\rightarrow \begin{cases} A_1\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u * 2 - 1, LOD_{\delta+1}\right) \\ A_2\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u * 2 + 0, LOD_{\delta+1}\right) \\ A_3\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u * 2 + 1, LOD_{\delta+1}\right) \end{cases} \tag{5.20}$$

This means that, when refining an alignment in a given LOD, three *children* alignments are inserted into the heap (Fig. 5.13): a centered one, a left displaced one and a right displaced one. The heap is then sorted according to the cost function of each alignment, and the process continues until the root element is an alignment in maximum LOD. In this case, this is returned as the result.

In Fig. 5.13 it can be appreciated how convergency for cases B, C, D, E, F, H, J and K can be demonstrated as it was done for hierarchical orientations: all facing samples have already been evaluated in the *parent* alignment, in which minimum distances were smaller, maximum distances were greater, and the amount of contained samples were also greater.

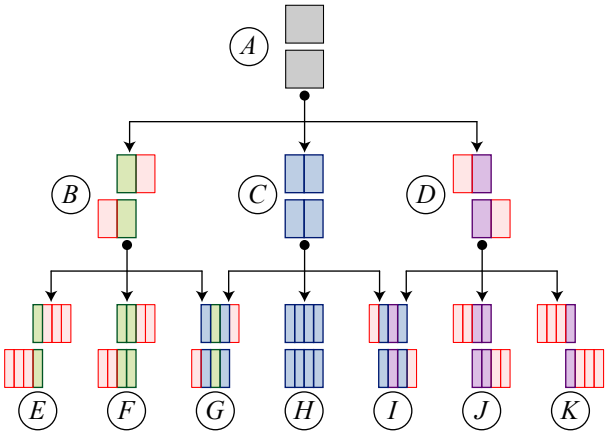

Figure 5.13: *Hierarchical displacements search process. Search process from LOD_0 to LOD_2. Each box represents a sample. Marked in green are the samples compared in case B, in blue the ones compared in case C, and in purple the ones compared in case D. Marked in red are the samples without a corresponding one in the other fragment.*

Cases G and I present an extra ambiguity given that they have two *parents* and, in none

of them, all the facing samples have been compared. However, it can be demonstrated that a *child* alignment with two *parents* will always have a smaller or equal cost function than one of them, which is sufficient to ensure convergency in the proposed approach.

Considering that both cases (G and I) are symmetrical, case G is used to demonstrate this property. Fig. 5.14 shows in detail this case, and illustrates the nomenclature used for the next equations.

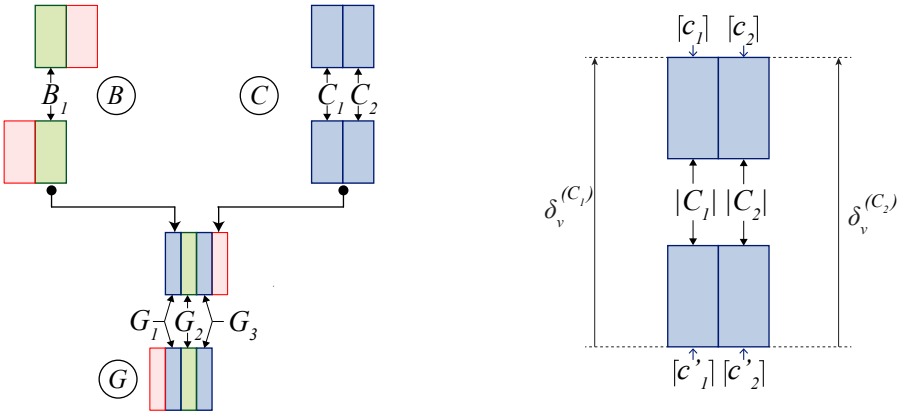

Figure 5.14: *Demonstration of convergency for case G. (left) both parents, B and C of case G, where matching samples colored with the same criteria as Fig. 5.13. (right) nomenclature used for the demonstration, taking as exmple case C: C_i represents two corresponding samples, c_i and c'_i. $|C_i|$ is the total number of matching samples. $\delta_v^{(C_i)}$ represents the distance between the maximum projective distances $\lceil c_i \rceil$ and $\lceil c'_i \rceil$.*

The first thing to prove is that the value of δ_v decreases as the alignment is refined. For this, facing samples from case C are considered and compared with the ones from case G as follows:

$$(\lceil c_1 \rceil \geq \lceil g_1 \rceil) \wedge (\lceil c'_1 \rceil \geq \lceil g'_1 \rceil) \rightarrow \lceil c_1 \rceil + \lceil c'_1 \rceil \geq \lceil g_1 \rceil + \lceil g'_1 \rceil \tag{5.21}$$

$$(\lceil c_2 \rceil \geq \lceil g_3 \rceil) \wedge (\lceil c'_2 \rceil \geq \lceil g'_3 \rceil) \rightarrow \lceil c_2 \rceil + \lceil c'_2 \rceil \geq \lceil g_3 \rceil + \lceil g'_3 \rceil \tag{5.22}$$

From these equations it can be said that:

$$\delta_v^{C_1} \geq \delta_v^{G_1} \tag{5.23}$$

$$\delta_v^{C_2} \geq \delta_v^{G_3} \tag{5.24}$$

which leads to the conclusion

$$Min(\delta_v^{C_1}, \delta_v^{C_2}) \geq Min(\delta_v^{G_1}, \delta_v^{G_2}, \delta_v^{G_3}) \tag{5.25}$$

$$\delta_v^{C} \geq \delta_v^{G} \tag{5.26}$$

no matter which value takes $\delta_v^{G_2}$, as shown below:

$$\delta_v^C = \begin{cases} \delta_v^{C_1} & \rightarrow \delta_v^C \geq Min(\delta_v^{G_1}, ...) \quad , \delta_v^{C_1} \geq \delta_v^{G_1} \\ \delta_v^{C_2} & \rightarrow \delta_v^C \geq Min(\delta_v^{G_3}, ...) \quad , \delta_v^{C_2} \geq \delta_v^{G_3} \end{cases} \tag{5.27}$$

Once proven that the value of δ_v decreases as the alignment is refined, to prove that the cost function value also decreases, we distinguish between two cases: 1) $|C_1| \neq 0$, and 2) $|C_1| = 0$. The first one is easier to demonstrate because, if C_1 has matching samples, it can be said that:

$$|G_1| + |G_2| \leq |C_1| \tag{5.28}$$

$$|G_3| \leq |C_2| \tag{5.29}$$

For the second one, where $C1$ has no matching samples, it can be demonstrated that:

$$|G_1| = 0 \tag{5.30}$$

$$(|G_2| + |G_3| \leq |C_2|) \vee (|G_2| + |G_3| \leq |B_1|) \tag{5.31}$$

considering that the fragment is a closed 3D model, without discontinuities.

5.4.4 Hierarchical Search

Given the previous two optimizations, it makes sense to combine both to perform a full hierarchical search. The technique proposed here is an orientation-driven search that, in an outer loop hierarchically explores the orientation of both fragments and, for each alignment in a given LOD_θ, an inner loop hierarchically explores displacements. The goal of the inner loop is to calculate the value of δ_u that maximizes the cost function of the alignment passed by the outer loop, whilst the goal of the outer loop is to find the global solution to the stated problem.

Algorithm 2 Hierarchical orientation-driven search

1: $H \leftarrow Heap()$:
2: **for** $\theta_P \leftarrow 0$ **to** 2π **step** $2\pi/k$ **do**
3: $\quad H.Push(Alignment(\theta_P, [0..2\pi], 0, 0))$;
4: **end for**
5: **while** $true$ **do**
6: $\quad A\left(\theta_P, \left[{\theta_Q}^{(a)} \ldots {\theta_Q}^{(b)}\right], \delta_u, 0\right) \leftarrow H.Pop()$;
7: $\quad$ **if** $A.LOD_\theta = LOD_{\theta_{max}}$ **then**
8: $\quad\quad$ **return** A;
9: $\quad$ **else**
10: $\quad\quad A_1 \leftarrow Alignment\left(\theta_P, \left[{\theta_Q}^{(a)} \ldots {\theta_Q}^{((b-a)/2)}\right], \delta_u, 0\right)$;
11: $\quad\quad A_2 \leftarrow Alignment\left(\theta_P, \left[{\theta_Q}^{((b-a)/2)} \ldots {\theta_Q}^{(b)}\right], \delta_u, 0\right)$;
12: $\quad\quad H.Push(BestDisplacement\,(A_1))$;
13: $\quad\quad H.Push(BestDisplacement\,(A_2))$;
14: $\quad$ **end if**
15: **end while**

Each iteration of the outer loop (Algorithm 2) pops an alignment from the heap, and inserts two new alignments. Each child alignment is passed to the inner loop (Algorithm 3) in order to compute its maximum value of the cost function and, according to this value,

is stored in the sorted heap of active alignments in the outer loop. When the best active alignment is in the maximum orientation LOD, it is returned as the solution to the problem.

By combining these two optimizations, the excessive amount of evaluations of the cost function proposed in the exhaustive approach $\mathcal{O}(m,n,k) \approx (m*k)*(2n-1)$ gets reduced to $\mathcal{O}(m,n) \approx log_2(m) * log_2(n)$ in the best execution case scenario. However, the third bottleneck associated to the evaluation of the cost function (which has to perform $|u| * |w|$ calculations per alignment) is still not solved.

To do so and without loss of correction, when evaluating displacements of an alignment in the inner loop, instead of searching until the maximum LOD search is performed until $LOD_\theta = LOD_\delta$. This allows performing very fast evaluations of the cost function when the uncertainty of the representation is high, and progressively achieve more precise results, as the process approaches to the global solution. Given that the proposed cost function is optimistic, convergency is ensured and, despite that the lack of precision in coarse representations may lead to perform some unnecessary comparisons, the increase of global performance fully justifies the proposed optimization, as it will be shown in the next section.

Algorithm 3 Hierarchical displacement search $BestDisplacement(A)$

1: $H \leftarrow Heap()$:
2: $H.Push(A)$;
3: **while** $true$ **do**
4: $A\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u, LOD_\delta\right) \leftarrow H.Pop()$;
5: **if** $A.LOD_\delta = A.LOD_\theta$ **then**
6: **return** A;
7: **else**
8: $A_1 \leftarrow Alignment\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u * 2 - 1, LOD_{\delta+1}\right)$;
9: $A_2 \leftarrow Alignment\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u * 2 + 0, LOD_{\delta+1}\right)$;
10: $A_3 \leftarrow Alignment\left(\theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u * 2 + 1, LOD_{\delta+1}\right)$;
11: $H.Push(A_1, c(A_1)); H.Push(A_2, c(A_2)); H.Push(A_3, c(A_3))$;
12: **end if**
13: **end while**

5.4.5 Many-to-Many Search

In order to create a semi-automatic tool to assist restoration personnel in the re-assembly of the original artifact, the one-to-one comparison search strategy proposed should compare each fragment with all the rest and, then, sort the resulting alignments in descending order to offer first the most promising matches. The main problem with this strategy is that, in a large database, lots of results will be discarded, since one fragment only matches with a very reduced set of neighbors.

However, with a simple modification in the proposed technique this process can be considerably accelerated: if the orientation-driven search illustrated in Algorithm 2 is initialized with all the possible pairs of fragments in the coarsest LOD, the first result achieved will be the best pair among all possible ones. Given that each iteration of the

hierarchical algorithm the most promising alignment is selected and refined, potentially bad alignments are automatically discarded.

As happened with previous hierarchical optimizations, first match found will ensure to be the best match of all the considered dataset, and next results will be also ordered according to the cost function, allowing the tool to offer first solutions with less ambiguity.

To do so, an alignment should be defined as:

$$A\left(P,\ Q,\ \theta_P, \left[{\theta_Q}^{(a)} \dots {\theta_Q}^{(b)}\right], \delta_u, LOD_\delta\right) \tag{5.32}$$

$$c : \mathbb{N} \times \mathbb{N} \times \mathbb{R} \times [\mathbb{R} \dots \mathbb{R}] \times \mathbb{Z} \times \mathbb{Z} \to \{x \in \mathbb{N} : x > 0\} \tag{5.33}$$

where P and Q are two indices pointing to the compared fragments in the alignment.

The final search algorithm according to this new strategy is shown in Algorithm 4.

Algorithm 4 Hierarchical N-to-N search

```
1:  H ← Heap() :
2:  for i ← 0 to Fragments.Count step 1 do
3:      P ← Fragments[i];
4:      for j ← i + 1 to Fragments.Count step 1 do
5:          Q ← Fragments[j];
6:          for θ_P ← 0 to 2π step 2π/k do
7:              H.Push(Alignment(P, Q, θ_P, [0..2π], 0, 0));
8:          end for
9:      end for
10: end for
11: while true do
12:     A(θ_P, [θ_Q^(a) ... θ_Q^(b)], δ_u, 0) ← H.Pop();
13:     if A.LOD_θ = LOD_θmax then
14:         return A;
15:     else
16:         A_1 ← Alignment(P, Q, θ_P, [θ_Q^(a) ... θ_Q^((b−a)/2)], δ_u, 0);
17:         A_2 ← Alignment(P, Q, θ_P, [θ_Q^((b−a)/2) ... θ_Q^(b)], δ_u, 0);
18:         H.Push(BestDisplacement(A_1));
19:         H.Push(BestDisplacement(A_2));
20:     end if
21: end while
```

From the code below, it is important to notice that, the $return$ instruction in line 14 should not stop the execution and the $while$ loop in line 11 should not be taken as in infinite loop. The implementation of this algorithm is done using an independent thread that continuously searches in an asynchronous manner for new results. Each time an alignment in the highest LOD is found, it is "returned" to the main thread without interrupting the search loop, which only ends when the application is closed, or all matches have been found. As this process continues refining alignments, the interface offers the user the already found matches so he does not have to wait until the whole database has been compared to start the re-assembly process.

5.5 Results

This section empirically evaluates the correction and performance of the proposed technique. First, a set of ceramic fragments is used to compare the efficiency of each one-to-one algorithm, from the exhaustive approach to the fully hierarchical one. Later, a singular fresco database from Brown's Ph.D. Thesis [23] is used to compare results. Finally, an example implementation of the many-to-many technique is presented.

5.5.1 Performance evaluation

To evaluate the efficiency of the proposed technique, a set of flat ceramic fragments from the seventeenth century has been used. As it can be seen in Fig. 5.16 (left) fragments are very eroded, and the fractured edges do not match between them perfectly.

It is important to notice that comparisons are performed considering only pairs of fragments, and results are expressed as the average search time for each possible combination. The goal is, then, to prove the performance and correction for pairwise comparisons, no matter how big the dataset of fragments is.

All tests have been executed in a 3.4 GHz Intel Core i7 computer with 4GB of RAM. The 3D models of the fragments have been acquired using a Konica Minolta Vivid 9i laser scanner and a turning plate. The average acquisition time, including partial view registration and global remeshing of the scanned point-clouds, has been around an hour per fragment. Resulting 3D files present an average triangle count of about 100K triangles.

An example of the 3D models used for evaluation purposes is shown in Fig. 5.15.

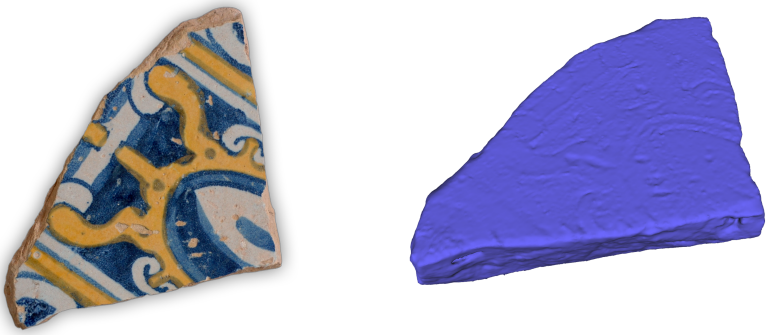

Figure 5.15: *A real fragment (left) and the acquired 3D model (right).*

In order to compare within the different proposed techniques, 36 different searches between all pairs of fragments have been executed. Each search is characterized by the amount of considered orientations (from 2^5 to 2^{10}) and the sampling resolution on the fragment's surface over axis U (from 2^5 to 2^{10}).

Each fragment has been pre-processed once per each resolution of the evaluations performed. The maximum pre-processing time has always been below 20 seconds, when calculating 1024 orientations with 1024 sample points in the U axis and 16 points in the W axis. Considering that, in a normal execution, one fragment has to be pre-processed

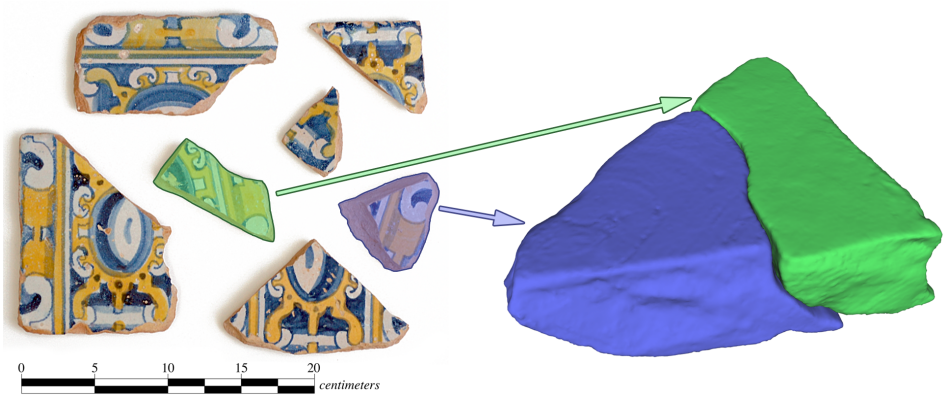

Figure 5.16: *Evaluation with real fragments. (left) Some of the fragments used to evaluate the proposed technique. (right) One of the results achieved during the search process.*

only once and that the acquisition time is much bigger, these times can be considered as not significant.

The total amount of memory required to store a fragment depends on the number of considered orientations (m), and on the sampling resolution for axes U and W ($|u|$ and $|w|$, respectively), which gives an overall cost of $\mathcal{O}(m, |u|, |w|) \approx m * |u| * |w|$.

More in detail, and considering $|u| = |w|$, the total size can be expressed as:

$$\mathcal{O}(m, |u|, |w|) \approx \frac{1 - 4^{nLods_d}}{1 - 4} * (2^{nLods_a} - 1) \tag{5.34}$$

being $nLods_d = log_2(|u|) + 1$, and $nLods_a = log_2(m) + 1$.

For the simplest search ($m = 32$, $|u| = 32$ and $|w| = 16$) using the hierarchical approach, the total amount of memory used was 404 KB, whilst for the most detailed search ($m = 1024$, $|u| = 1024$ and $|w| = 16$) 532.8 MB were required.

Given that the goal of this approach is to solve the pairwise matching of fragments, and not to solve the global puzzle, a result is considered to be correct if the transformation returned is the global minima of the stated cost function. Global correction considering all the fragments is out of the scope of this technique, and is highly conditioned by the ambiguity of the set of fragments considered: a higher level technique to reconstruct the original artifact should discard the matching pairs that contradict the common goal.

During the tests, all techniques have returned exactly the same results for each couple of fragments compared. This was expected given that it has been formally demonstrated that the hierarchical strategy always find the global minima solution.

Local minima results only exist when the sampling resolution is too low, given the irregularities of fragments' surface. In the evaluation set used, the maximum fragment size was smaller than 20 centimeters meaning that, with a sampling resolution over U axis of 1024, the separation between samples was always smaller than 0,019 millimeters (which is similar to a high resolution scanner accuracy). When a local minima solution appears, due to under-sampling, it appears in all proposed techniques. This way, after performing all the tests, the only difference between proposed techniques was the execution time.

Fig. 5.17 shows achieved results for exhaustive approach. Notice how execution time is linear to the amount of orientations considered, and spacial resolution affects more the speed of the technique. This happens because resolution affects both the evaluation the cost function and the amount of displacements to consider. In the other hand, since exhaustive approach has to consider all possible alignments, the increase of execution time is always proportional to the increase of the complexity of the problem.

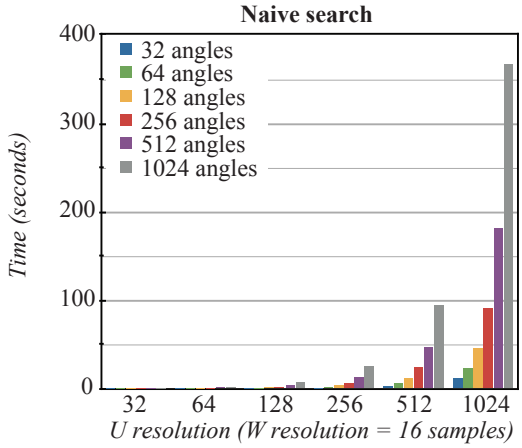

Figure 5.17: *Time results using exhaustive search.*

Fig. 5.18 shows achieved results using hierarchical displacements. In this case, the effect of spacial resolution in execution time is reduced, since the amount of displacements to consider are drastically reduced. This makes the total execution time to be arround 6.5% with respect to the exhaustive approach. However, since the amount of samples to operate with while evaluating the cost function remains the same, results can be still improved.

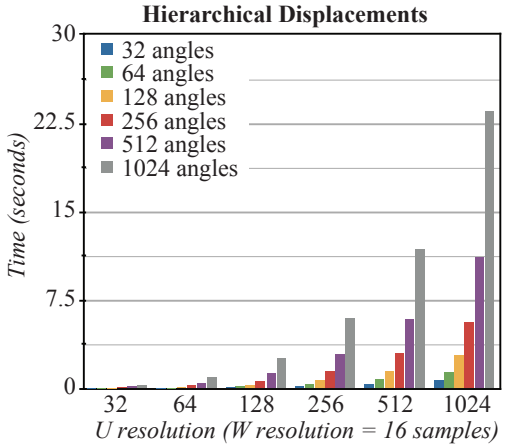

Figure 5.18: *Time results using hierarchical displacements.*

Fig. 5.19 shows achieved results using hierarchical orientations. In this case, the amount of orientations considered does not affect linearly to execution time. This is a consequence of the fact that each characterization of fragments in an orientation range provides new information to work with, and allows to discard lots of potentially bad alignments. As the search process evolves, and ambiguity is reduced, the result converges very fast to the optimal solution.

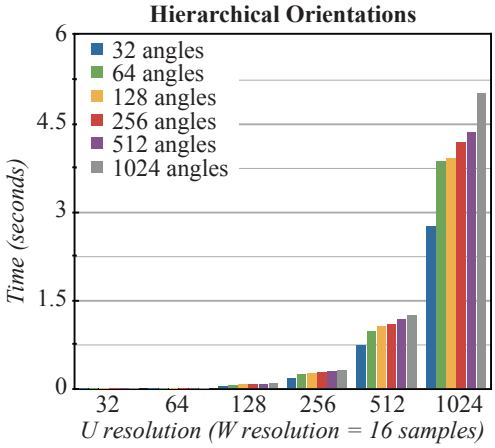

Figure 5.19: *Time results using hierarchical orientations.*

Fig. 5.20 shows achieved results combining both hierarchical techniques together. As it can be appreciated, both improvements are also combined and the overall execution time behaves almost linearly to the global size of the problem (notice that the X axis of the charts uses an exponential scale, and so does the rotation resolution). The most complex search is now about 1.400 times faster than with the exhaustive approach.

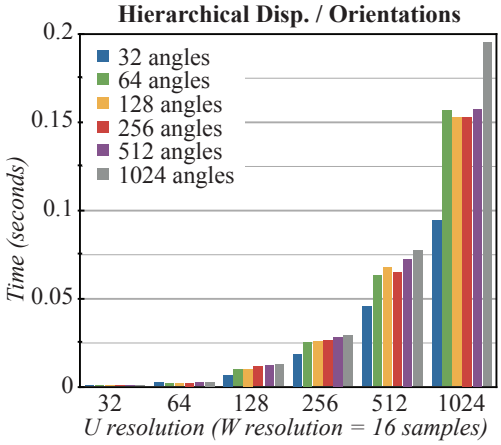

Figure 5.20: *Time results using the hierarchical approach.*

Finally, if the cost function evaluation is optimized by stopping the inner displacement search when $LOD_\theta = LOD_\delta$, the final hierarchical technique is obtained (Fig. 5.21), which speeds-up the naive results more than 20.000 times in the most complex problem, and provides the technique an execution cost linear to the size of the problem. Given that, in the proposed hierarchical technique, the displacement and orientation searches evolve simultaneously, only problems with the same resolution in both parameters are efficiently evaluated. If only these combinations are considered (right chart), it can be seen how the resulting technique behaves linearly to the size of the problem: $\mathcal{O}(n) \approx n$, being n the resolution of angles/displacements considered.

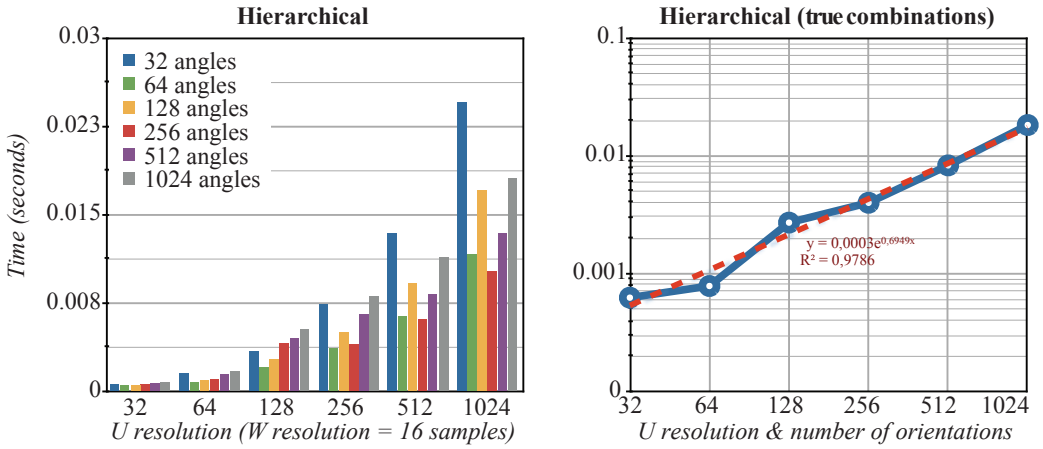

Figure 5.21: *Time results using the full hierarchical approach.*

5.5.2 Griphos dataset

To evaluate the performance of the proposed algorithm with a different dataset, the Griphos Fresco fragments have also been tested. These fragments were originally published in [24], and were used in Brown's Ph.D. Thesis [23] to to evaluate the performance of the ribbon matcher technique. 3D models used in this section are courtesy of Dr. Tim Weyrich (University College London), Prof. Szymon Rusinkiewicz (Princeton University) and Dr. Benedict Brown (University of Pennsylvania).

The Griphos Fresco dataset includes 263 fragments of wall paintings from the site of Akrotiri on the volcanic island of Thera (modern-day Santorini, Greece). These fragments have been exhaustively scanned, catalogued and textured.

One of the most interesting aspects of the Griphos dataset is that matches are already known, so correction of automatic re-assembly techniques can be evaluated. Fig 5.22 shows one of the biggest cluster of fragments found, together with the fragments' numbers and the matches found using the ribbon matcher technique.

Results achieved with the proposed technique using this dataset were always correct (the search process never stopped in a local minima solution). Amongst the matches found using fragments known to be neighbors, 92% corresponded to the real solution. The other 8% corresponded to alignments with a better cost, despite not being the correct match.

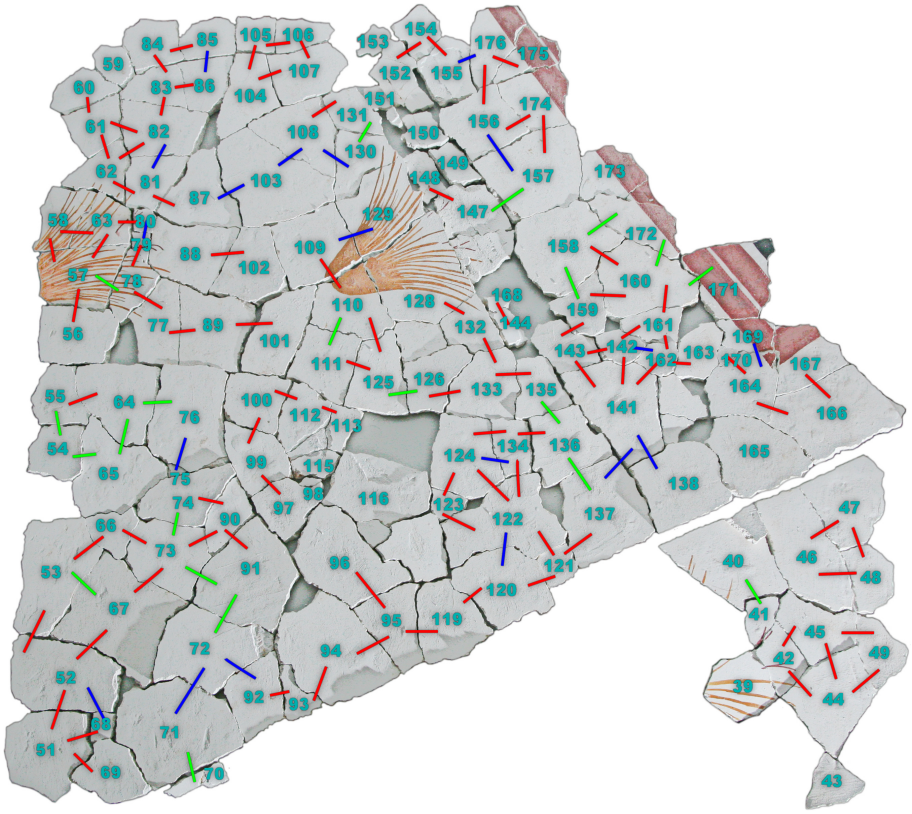

Figure 5.22: *Griphos Fresco dataset. Red lines indicate matches found using the ribbon matcher with a 25 mm strip width. Blue links indicate additional matches found with a 12.5 mm strip width, and green links indicate further matches found with a 50 mm strip width. Only numbered fragments have been scanned*

Performance in this case was slightly slower than in the previous dataset, probably because the erosion and the shape of the breaks created more ambiguity. However, the acceleration rate introduced by the hierarchical optimizations was similar.

One of the advantages of the proposed search algorithm is that, using always the same search settings (256 orientations and 256 * 16 samples for axes U and V, respectively), all matches were found. In the case of the ribbon matcher, as Fig. 5.22 shows, different configurations had to be used to find different matches. Figs. 5.23, 5.24 and 5.25 show some of the matches found that the ribbon matcher could find using a 12.5mm, 25mm and 50mm strip width, respectively. Fig. 5.26 shows some matches found that the ribbon matcher could not find. Numbers of the fragments in these figures correspond to numbers in Fig. 5.22 and search times using the full hierarchical approach are displayed below each match. The other great advantage of the proposed technique is that alignments between fragments are computed extremely fast (with an average search time of 0.015 seconds), whilst the ribbon matcher took an average search time of 2 seconds.

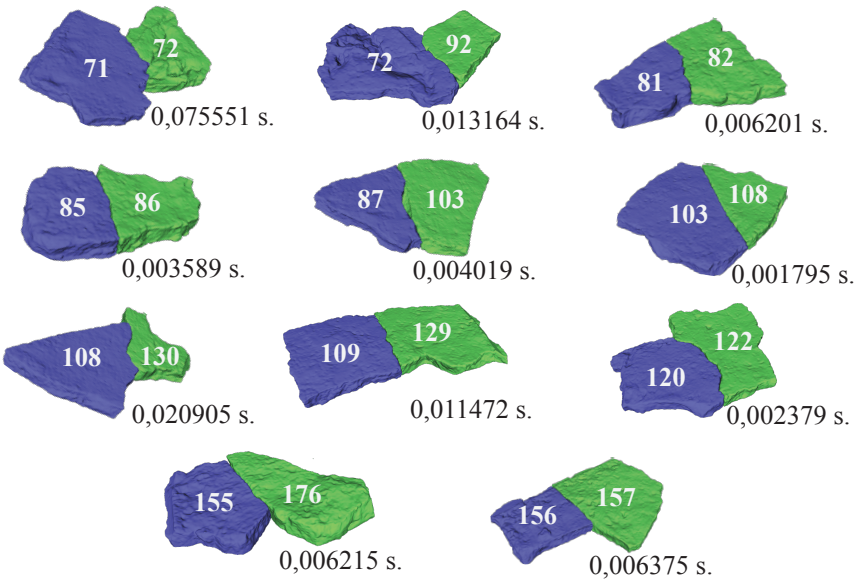

Figure 5.23: *Some alignments found using the Griphos Fresco dataset. This set of alignments corresponds to some that the ribbon matcher found using a 12.5mm strip width.*

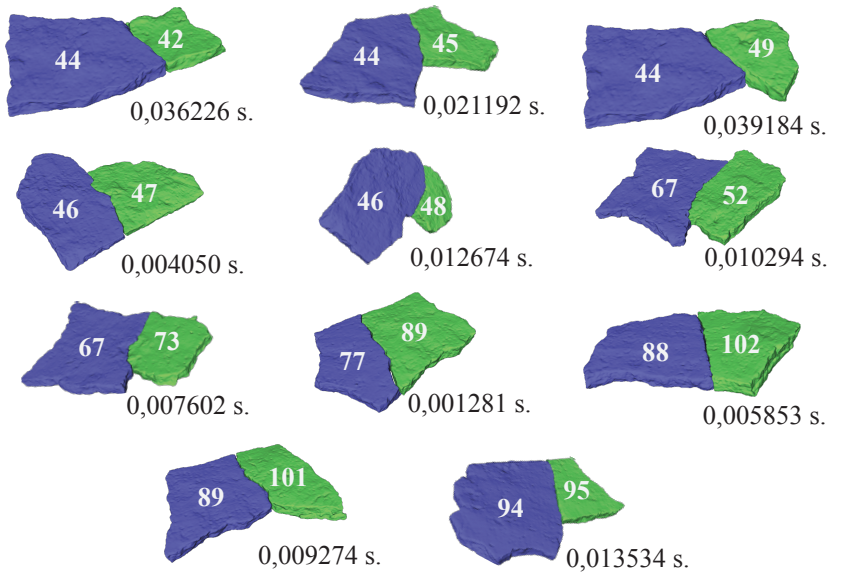

Figure 5.24: *Some alignments found using the Griphos Fresco dataset. This set of alignments corresponds to some that the ribbon matcher found using a 25mm strip width.*

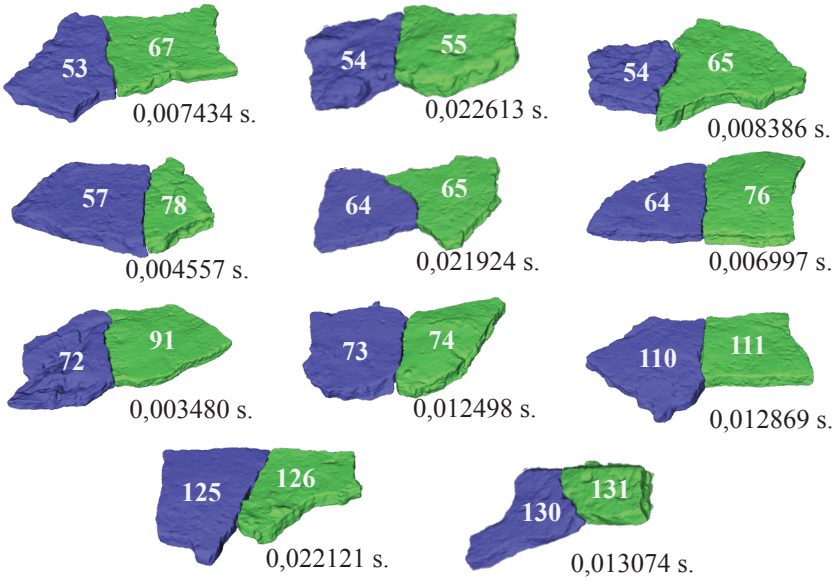

Figure 5.25: *Some alignments found using the Griphos Fresco dataset. This set of alignments corresponds to some that the ribbon matcher found using a 50mm strip width.*

Figure 5.26: *Some alignments found using the Griphos Fresco dataset that the ribbon matcher could not find.*

5.5.3 Many-to-many sample implementation

In order to prototype a semi-automatic tool that implements the proposed many-to-many search algorithm and to model the interaction with the end-user, an implementation of the 2D reconstruction tool has been developed. Fig. 5.27 shows a screenshot of the user interface during the re-assembly process.

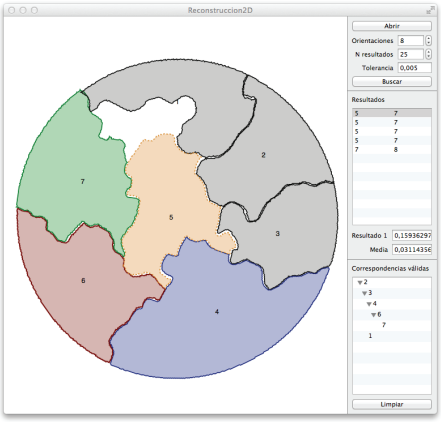

Figure 5.27: *Semi-automatic tool implementation for 2D re-assembly.*

The proposed application performs a global search by introducing into the heap all possible combinations between fragments and, hierarchically, searches for the most promising matches. Since the search process is performed asynchronously, after selecting the dataset to work with, best results are offered almost instantly. Thanks to the demonstrated optimist cost function, first results ensure to be the global best ones, so the re-assembly process can be started as soon as first matches are returned.

Potential alignments are offered to the user according to the ones he has previously selected, allowing to pre-visualize them in real-time (displayed as an orange fragment with a dashed outline in Fig. 5.27).

A tree structure representing selected alignments always provides a feed-back to allow reproducing the re-assembly with the real fragments, and contextual information for each fragment is shown graphically by clicking on the pre-visualization area (in the case shown in Fig. 5.27, fragment 6 has been selected, marked in red, and its parent and child fragments are displayed, marked in blue and green, respectively).

Thanks to the many-to-many hierarchical search proposed, the sample dataset illustrated (with 8 fragments pre-processed with 512 orientations and 512 samples over the U axis) returns the best match amongst the 28 possible pairs of fragments in just 0.159 seconds and it takes an average time of 0.031 seconds to find all best matches.

Of course, the bigger the dataset the longer it will take to compute all matches but, considering the hierarchical nature of the proposed technique and the achieved execution times, the use of this tool improves in orders of magnitude the efficiency of manual interventions.

5.6 Conclusions and Future Work

This chapter has presented an automatic process for re-assembling ancient artifacts from fragments found at archaeological sites. Focusing on flat topologies, the relatively low dimensionality of the solution space has been exploited in order to create an algorithm that efficiently solves the puzzling problem.

Starting from the acquisition stage, a solution for scanning fragile fragments has been introduced: the use of cyclododecane as a whitening spray. Thanks to its chemical stability and to the fact that it sublimes at room temperature leaving no residuals, CCD is a perfect candidate to solve reflection/refraction issues during the scanning process. A set of experiments has been presented, proving that the thin layer created on the surface of fragments do not interfere with the scanner accuracy thanks to the reduced particle size.

An easy to compute cost function and a detailed characterization of the solution space has been presented. Taking advantage of a parallel projection plane, distances between fragments in a given alignment are very fast to compute, since the only operations involved are simple additions, subtractions and logical comparisons.

In order to speed-up the search stage, a pre-processing stage has been introduced taking advantage on the GPU rendering capabilities. This way, all complex geometrical transformations, visibility tests and discretization operations are carried out using wired units embedded in the GPUs' architecture pipeline. Results achieved in pre-processing stage depend only on fragments, but can be used in as many alignments between different pairs as necessary.

A set of search algorithms has been proposed to find the best alignment between a given pair of fragments. Starting from a naive approach that explores exhaustively the complete solution space, a set of hierarchical optimizations have been progressively applied to improve performance. During the acceleration of the search technique, it has been formally demonstrated that none of the hierarchical algorithms loses correction with respect to the exhaustive approach. As result of these improvements, a linear cost with respect to the size of the problem has been reached.

In order to improve results in the final many-to-many search, a small modification has been applied to the fully hierarchical search strategy, in order to consider all fragments of a given dataset at the same time. This way, best matches amongst all possible pairs of fragments are found in the first place, automatically discarding potentially bad alignments.

A set of evaluation experiments has been presented, paying special attention to the empirical confirmation of the technique's correction and the use of resources (CPU and memory) for different problem sizes. A singular dataset of frescoes from the Thera island (the Griphos dataset) has also been used to compare achieved results with the most similar technique in the recent bibliography: the ribbon matcher proposed by Dr. Benedict Brown in his Ph.D. Thesis dissertation. Results using this dataset have provided new matches that could not be found with the previous approach, and search times have been considerably reduced. Also, the proposed technique has been proven to be less parametrized, avoiding the need of performing different searches over the same fragments with different settings.

Finally, a semi-automatic tool has been implemented to illustrate the potential of the full process in 2D problems. Taking advantage of the many-to-many hierarchical search strategy, and presenting the user an easy-to-use interface, efficiency with respect to manual interventions has been proven to be improved.

For further improvements in future approaches, several aspects of the process can be optimized. Perhaps, the most important one, that will make this technique feasible for museums to work with, is not related to the software aspect but to the hardware itself: the acquisition stage is the main bottleneck because the need to use a general purpose 3D scanner makes it to time-consuming. This problem could be faced in two ways: (1) by designing a specific acquisition hardware (probably based on laser triangulation and a turning plate) that efficiently exploits the flat nature of the fragments, or (2) by creating an acquisition software that automatically registers all partial views (scanned with the fragment laying on both the upper and lower face), and generates a triangulated B-Rep mesh to directly feed the pre-processing stage.

On the software side, during the experiments it has been observed that the main bottleneck in the search process is due to memory bandwidth. Since processor operations are so simple, computing the cost function and deciding the next alignment to refine is straight forward. However, for each cost function evaluation, the whole pre-computed projection matrix has to be loaded into memory. Even using a single threaded application (all results shown in this chapter only use one of the eight cores of the test computer), memory accesses keep the processor idle most of the time. Probably, trading between CPU and memory usage to find a solution that better balances the requirements of the proposed technique would yield into an enormous improvement. For this matter, using a compressed representation of the distances matrices would be an interesting way to explore.

Having the memory bottleneck solved, the next logical step would be to paralelize the search strategy. This way, using the same computer, a theoretical maximum acceleration of eight times could be achieved. Even better, by parallelizing the search strategy into the GPU, faster results could be obtained and the pre-processing stage could be partially included into the search stage, since all processing will be done in the same graphic context. This way, fragments could be only pre-processed until some coarse LOD, leaving the finest representations to be calculated on-demand during the search process. Keeping in mind that only a few finest LODs are reached in a given alignment, this improvement would yield into saving lots of memory, in exchange for some extra computing time.

Some other improvements would involve considering extra properties besides geometry in the cost function, or providing some “intelligence” to the semi-automatic tool, so collisions between fragments could be avoided, or a global optimization process could be applied to register big clusters of fragments considering all paired samples.

CHAPTER 6

6 Degrees of Freedom Approach

This chapter deals with the automatic reconstruction of generic archaeological artifacts. The main difference with previous approach is related to the dimensionality of the solution space to search in: since a flat topology is no longer assumed, the total amount of degrees of freedom to consider duplicates (three for positions and another three for orientations). This fact forces to change the search strategy from a dense to a sparse one, given that exhaustively exploring all possible alignments renders to prohibitive execution costs.

Some typical examples of this kind of problems are broken sculptures, friezes, columns... whose importance in Cultural Heritage Recovery is considerable.

The proposed technique takes as input data a set digital models of fragments, characterized as unstructured point clouds, and outputs the rigid transformation that maximizes the contact area between their surfaces. Unlike the previous approach, since the whole solution space cannot be explored within a reasonable time, results achieved cannot ensure global correction. To mitigate this drawback, a global reconstruction strategy is proposed in order to consider the whole original artifact. This allows to properly identifying good correspondences, and rejecting potentially incorrect ones in a more general context.

As commented in the previous chapter, global reconstruction in a NP-complete problem so, defining a deterministic solution may lead to an unbound execution time. However, exploiting highly potential alignments to disambiguate more doubtful ones can provide to an automatic system a great advantage over a simple one-to-one matcher. Of course, in certain cases, the human interpretation will be necessary. However, even for these cases, having an automatic tool that suggests potential correspondences can lead into an enormous performance boost.

The radical change in the search strategy introduced here attempts to mitigate the excessive cost of a dense registration approach. Even considering the hierarchical algorithm commented in the previous chapter, it has to be taken into account that the increase of performance achieved was a consequence of the progressive reduction of uncertainty. When facing 3-DOF problems, ambiguity was highly reduced exploring relatively high LODs. This way, until the hierarchical search produced an almost complete exploration of the 5th/6th LOD, high ambiguity situations were still present. Since only 3 degrees of freedom were considered, fully exploring LOD_5 meant exhaustively evaluating 32 orientations with their corresponding 65 displacements over the U axis (2080 alignments, with $32 \times |w|$ facing samples).

For the 6-DOF approach, in order to fully explore LOD_5, it will be necessary to exhaustively evaluate 32.768 orientations (2^{5^3}) with their corresponding 3.969 translations ($((2 \times 2^5) - 1)^2$), giving a total number of 130.056.192 alignments with 1024 facing samples (2^{5^2}) each one of them. At this point, ambiguity would be expected to reduce so the hierarchical strategy could filter only potentially good results. The computing requirements to reach this relatively low level of detail (only 32 orientations per axis considered) are extremely high, making the approach unsuitable for this dimensionality.

Also memory requirements and pre-processing times are extremely high. Fully characterizing a fragment with 8 LODs, for example, means computing 2^8 orientations per axis, which leads to $2^{8^3} = 16.777.216$ distance matrices with $2^{8^2} = 65.536$ samples each one. This means storing 1.099.511.627.776 float values, which takes 4TB (tera bytes), assuming a 4 byte float representation.

This computational and memory costs are simply prohibitive. This way, instead of performing an exhaustive search based on a dense representation of the correspondence, a sparse one based on descriptors is introduced.

Major contributions in this chapter are:

- A multi-scale feature extraction technique and a keypoint selection strategy driven by saliency.
- The extension of the Persistent Feature Histogram descriptor to reconstruction approaches.
- A fast one-to-one registration algorithm based on a three-level hierarchical search that exploits a set of formalized geometrical constraints.
- A graph based global reconstruction algorithm that uses individual matches to perform the final reassembly of the original artifact.

6.1 Overview

The proposed 6-DOF technique faces all the stages involved in the reconstruction process (Fig. 6.1). The final goal is to exploit all the individual matches between pairs in a many-to-many search algorithm that automatically attempts to reconstruct the original artifact. Redundant matches between adjacent fragments are used in a global optimization algorithm to refine the individual poses and highly probable matches are used to disambiguate more doubtful ones.

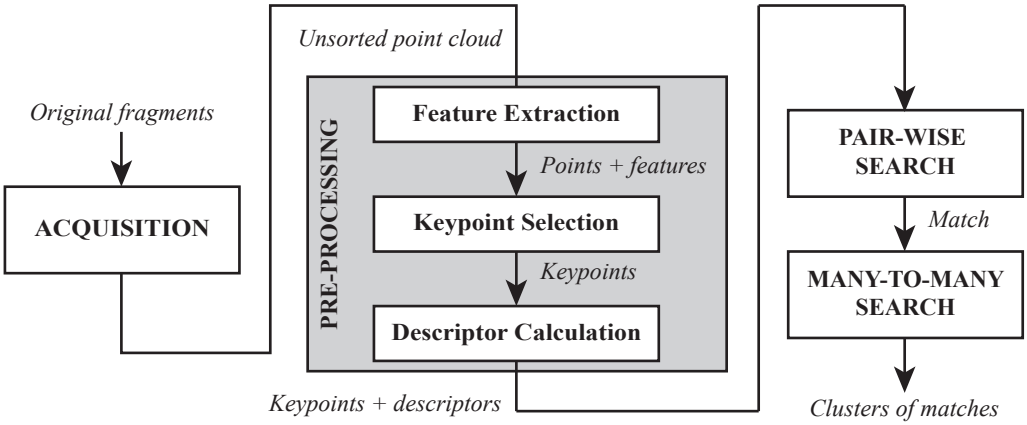

Figure 6.1: *6DOF technique overview*

Starting from the original fragments, first stage deals with the acquisition into a digital format. Unlike the previous approach, where a well defined B-Rep representation was needed to fully characterize the fragments, the proposed 6-DOF technique relaxes this requirement, being only necessary unstructured point clouds.

As commented before, the absence of planar constraints in the general problem forces to simplify the representation of fragments in order to achieve reasonable execution times. This process is carried out by the pre-processing stage, detailed in Section 6.2. In a first step, for each point in the original point cloud, a set of *features* is calculated in order to characterize them according to their local neighborhood. Using this features, only a subset of points called *keypoints* are selected. This selection is based on their uniqueness in their local domain, according to the previously calculated features. Once keypoints are selected, a set of histogram-based *descriptors* are calculated for each one of them trying to capture, in a compact representation, the topology of their surrounding neighbors. This way, from the pre-processing stage to the next ones, the original object representation is no longer necessary, and all algorithms operate over the keypoints and their associated descriptors.

Taking as input data the resulting keypoints and descriptors of two fragments (Section 6.3), the pairwise search algorithm tries to compute the rigid transformation that maximizes the number of keypoints in correspondence from both fragments. This way, the new cost function is defined over keypoints instead of uniformly sampled points in the fragment's surface. To efficiently do so, fast searches based on the descriptors are preformed.

Finally, the global reconstruction algorithm (Section 6.4) performs all the one-to-one combinations between fragments in a dataset to attempt to reconstruct the original artifact. Resulting individual matches are sorted according to their cost function and a graph based search is executed to build clusters of matching fragments. Ideally, if all the fragments considered fully define an artifact, the result of this last stage will be a single cluster that contains all the fragments properly registered. In the case of incomplete datasets, results will be sets of clusters partially defining parts of the object.

6.2 Pre-processing

This stage deals with the alternative sparse representation of the fragments. Taking as input data an unstructured point cloud, results are expected to be a set of strategically selected keypoints that reduce the dimensionality of the registration algorithm.

To accelerate the correspondence search process, each keypoint is supported by a compact representation of the topology of its local neighborhood provided by a descriptor. This descriptor has to be invariant to rigid transformations, in order to ensure that a random pose in the input fragments do not affect the quality of the achieved results.

Also, keypoint selection has to be robust against surface deformations, introduced by random noise during the acquisition stage, data discretization and erosion of the fragments.

The main idea of the proposed keypoint selection is inspired by the first jigsaw puzzle approaches: by properly identifying *indents* and *outdents*, the registration algorithm attempts to match pairs of complementary keypoints from two fragments to achieve the final registration.

6.2.1 Feature extraction

In order to efficiently identifying indents and outdents in the fragment's surface, a feature that represents the saliency of each point in the original point cloud has to be computed.

The saliency for a given point p, that belongs to a locally defined surface Φ is defined according to its surrounding neighbors. This way, using different search distances, r, different values for the saliency of p, $S^r(p)$, can be estimated.

To distinguish between indents and outdents, proposed saliency values are signed, being positive saliences associated to outdents and negative saliences associated to indents. Also, for bounding salience values into a uniform range, it is desirable a normalization that ensures that $-1 \leq S^r(p) \leq 1$.

Intuitively, the salience feature should be 0 when a point lays over a flat surface, it has to be a positive value when the point is a local maximum, and a negative value when the point is a local minimum.

Figure 6.2 illustrates these three cases for three different points p_1, p_2 and p_3, with their associated local surfaces, Φ_1, Φ_2 and Φ_3 respectively and the same search radius r. Notice how, according to the intuitively explained concept of saliency, it should be verified that $S^r(p_3) \leq S^r(p_1) \leq S^r(p_2)$ being $S^r(p_1) = 0$. This way, it will be immediate to identify p_2 as an outdent, p_3 as an indent and, more important, p_1 as a point with no relevant information for matching purposes, according to the introduced strategy.

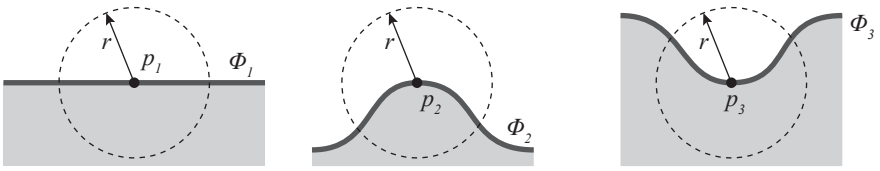

Figure 6.2: *Saliency of a point. Three different cases with different saliencies with a local neighborhood defined using the same radius r.*

To estimate the value of $S^r(p)$ for all the points in the original point cloud, the signed distance with respect to the best fitting plane for a given search radius, Π_p^r, is used. To compute this plane defined by a point, π, and a normal vector, n_Π, the covariance matrix, H, of all the points enclosed inside the sphere defined by r and centered on p is estimated as follows:

$$\mu_\Phi = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{6.1}$$

$$H = \sum (p_i - \mu_\Phi)(p_i - \mu_\Phi)^T \tag{6.2}$$

where each point p_i verifies that $p_i = \|p_i - p\| \leq r$ and N corresponds to the total number of points that satisfy this condition.

After computing H, which is a 3×3 matrix, a Singular Value Decomposition (SVD) is performed, so three 3×3 matrices are calculated: $[U, S, V] = \mathrm{svd}(H)$. The third column of the U matrix, $U^{(3)}$ corresponds to the third eigenvector with the lowest associated eigenvalue, and it can be said that:

$$\pi = \mu_\Phi \tag{6.3}$$

$$n_\Pi = U^{(3)} \tag{6.4}$$

However, since the signed distance to the best fitting plane is required in order to distinguish between indents and outdents, and the orientation of the third eigenvector does not ensure to be the correct one, it may result from the previous calculation that the estimated normal is mirrored with respect to the plane. To prevent this from happening, it has to be ensured that its projection with respect to the normal associated to p is always a positive value. Otherwise, n_Π should be inverted. More formally:

$$n_\Pi = \begin{cases} U^{(3)}, & \langle U^{(3)}, n_p \rangle \geq 0 \\ -U^{(3)}, & \langle U^{(3)}, n_p \rangle < 0 \end{cases} \tag{6.5}$$

where $\langle a, b \rangle$ represents the dot product between the vectors a and b, and whose value when considering normalized vectors is the cosine of the angle they define. Figure 6.3 illustrates these two cases.

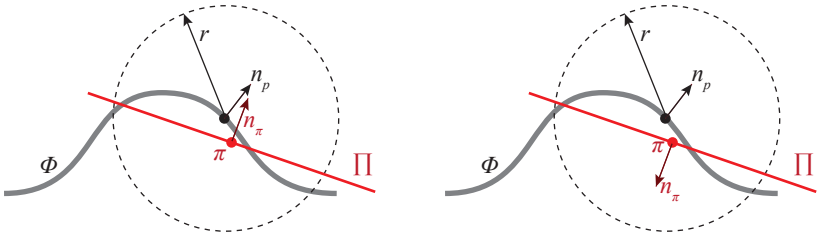

Figure 6.3: *Best fitting plane calculation. (left) A best fitting plane with the correct normal estimation $\langle n_\pi, n_p \rangle \geq 0$. (right) The same best fitting plane with the normal inverted $\langle n_\pi, n_p \rangle < 0$*

Notice how, for estimating the normal associated to p, which is the one used to disambiguate the orientation problem, the process is identical to the proposed one. However, since the original point cloud has been acquired using a 3D laser scanner, a viewing ray, v_p, from the scanner sensor to p is available, so n_p can be oriented in order to verify that $\langle v_p, n_p \rangle \leq 0$.

Once the best fitting plane is calculated for the given point p and considering the local neighborhood with a radius r, the signed distance between p and Π, is computed as:

$$\|p, \Pi\| = \langle n_\pi, (p - \pi) \rangle \tag{6.6}$$

Finally, since the proposed saliency feature representation has to be bound in the interval, $S^r(p) \in [-1 \dots 1]$, and considering that the absolute value of $\|p, \Pi\|$ is always smaller than the search radius used to compute the plane Π, the normalization that leads to the final feature formulation is defined as follows:

$$S^r(p) = \frac{\|p, \Pi\|}{r} \tag{6.7}$$

In order to select keypoints in the next stage, this saliency feature has to be calculated for all the points in the original representation, leading to a characterization as the one presented in Figure 6.4. Notice how upper flat face is shaded in cyan (denoting 0 saliency for the vertices that make it up), while outdents and indents are easy to distinguish.

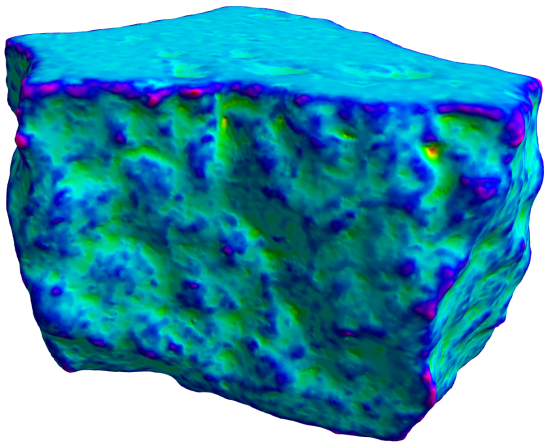

Figure 6.4: *Saliency feature for a 3D fragment. Cyan represents a close to 0 saliency value. The darker the blue, the more saliency, and the greener, the less.*

The proposed descriptor provides a good characterization for the points in the original dataset, but it presents a major drawback: which value for r should be used to locally characterize the saliency of a given point?. Using small search distances will yield into a very dense keypoint representation, where random noise introduced in the model could seriously affect the robustness of the search process. On the other hand, using large search distances may lead into a too sparse representation, where only sharp edges will be distinguishable from the rest of the object. Figure 6.5 illustrates this concept.

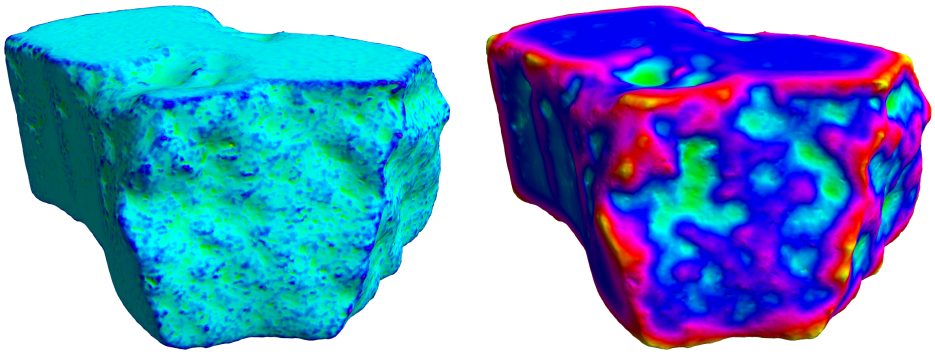

Figure 6.5: *Saliency feature for diferent search radius. Left image shows the saliency values calculated using a small search radius, whilst right image shows the same model with the saliencies associated calculated using a very big radius.*

In order to mitigate the effect that a wrong search radius selection could introduce into the final registration results, and to provide a more robust keypoint selection, the proposed saliency feature $S^r(p)$ is extended in order to consider multiple simultaneous scales, as proposed in [102].

The main idea of this improvement consists on applying the concept of scale-space for saliency estimation using a surface variation estimation. Given that the proposed feature is normalized with respect to the search distance, r, increasing the size of the local neighborhood is similar to applying a smoothing filter. This concept is more clear by analyzing the impact of the scale in the covariance matrix previously calculated: since H is defined as the sum of squared distances from the neighborhood's centroid, if r increases, each individual point contributes less to the surface variation estimate. This way, high-frequency variations are attenuated as a standard low-pass filter would do.

According to this, the saliency for each point p is defined as:

$$S(p) = \frac{1}{N} \sum_{i=1}^{N} S^{r_i}(p) \tag{6.8}$$

where $r_i = r_{min} + i \cdot (r_{max} - r_{min}) / N$. Values for minimum search distance, r_{min}, and maximum search distance, r_{max}, can be specified by the user. However, considering that this concepts are not very intuitive, an automatic selection can be performed according to the average radius of the fragments in a given dataset and to the resolution of the original point cloud or, even better, well known combinations can be defined as presets to work with different kinds of materials (stone fragments normally present higher roughness surfaces than clay fragments, so smaller search radius can be used to capture this information). The value of N represents the amount of scales to consider in the feature extraction process and, according to the empirical results presented in next sections, $N = 10$ has proved to provide a good compromise between calculation efficiency and results quality.

Figure 6.6 shows the results of applying the proposed multi-scale salience feature for the same fragment shown in Figure 6.5, and evaluating 10 different scales, linearly distributed between the radius illustrated in the left image and the one in the right image.

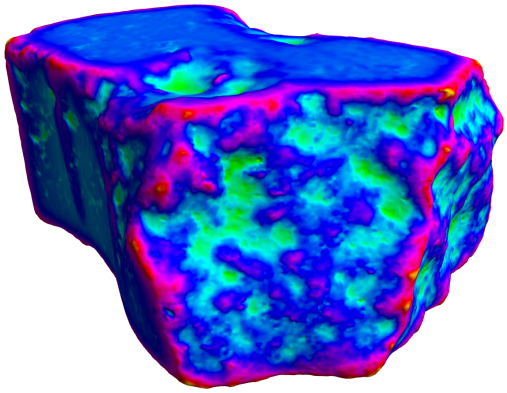

Figure 6.6: *Multi-scale salience feature.*

Notice how indents and outdents are more precisely characterized than in the previous case when a big search radius was used, and how high frequency details are added, allowing to identify new keypoints. With respect to the results achieved in the previous figure when using a small search radius, notice how the effect of high frequency noise is considerably mitigated, and how relevant points for further searches are more clearly distinguished from the rest of meaningless points.

Taking advantage on the proposed multi-scale feature, and with the aim of providing more information to the keypoint selection algorithm, a differential roughness descriptor is also computed in order to be able to discard smooth areas that probably correspond to the outer surface of the fragment.

In this sense, unlike other reconstruction approaches like [71] [123] [124], the proposed pre-processing stage does not perform a full segmentation of the fragment into sets of fracture faces and original ones. Instead, it computes for each single point the probability of being part of a fracture. This way, the search strategy gets more complicated, since the solution space cannot be divided into disjoint sets of potentially matching keypoints, but also more robust (the whole process cannot be compromised by misinterpreting a cluster of vertices) and a more general solution to the reconstruction problem is provided (having highly eroded edges can make impossible to detect the boundaries of facets).

The proposed roughness descriptor, $R^r(p)$, for a given point, p, is expressed as the variation of the signed distance previously calculated with respect to the distance, for a given local neighborhood enclosed inside a sphere centered in p and with radius r.

Considering that $S(p)$ has been already calculated with a multi-scale strategy, the roughness descriptor is directly computed using a search distance equal or smaller than the previous one, and considering the final multi-scale descriptor, $S(p)$, instead of a singular one, $S^r(p)$. This way, the value of the roughness descriptor can be expressed as:

$$R^r(p_i) = \frac{1}{k} \sum_{j=1}^{k} \frac{S(p_i) - S(p_j)}{\|p_i - p_j\|} \tag{6.9}$$

where $i \neq j$, all considered points verify that $\|p_i - p_j\| \leq r$ and k is the number of points that satisfy this condition. Figure 6.7 illustrates this concept.

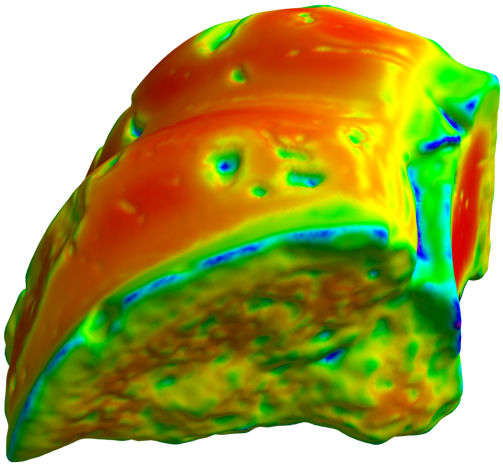

Figure 6.7: *Roughness feature. Red color indicates low values for the proposed descriptor. The more blue, the more roughness.*

Notice how, the points that belong to the original faces of the fragment present very low roughness values, whilst the ones that correspond to outdents and indents present very high values. In this sense, it is important to notice that, since the keypoint extraction algorithm, explained in the next section, is driven by the saliency, the roughness descriptor does not need to be calculated for all the points: only selected singular points will have the descriptor associated, and a threshold based rejector will discard those who present low values. This way, pre-processing execution costs are considerably reduced.

6.2.2 Keypoint selection

Once salience features are computed for all the points, the next stage in the proposed execution cycle selects the most meaningful ones in order to reduce the size of the data to work with in the searching stage. The selection strategy proposed in this section is based on the concept of dominancy of points with respect to their local neighborhood.

Given a point p_i and a search radius r, p_i is said to be *dominant* if one of the two conditions described below is verified:

$$\forall j \; S(p_i) > S(p_j), \quad \|p_j - p_i\| \leq r \wedge S(p_i) > 0 \tag{6.10}$$

$$\forall j \; S(p_i) < S(p_j), \quad \|p_j - p_i\| \leq r \wedge S(p_i) < 0 \tag{6.11}$$

In other words, if p_i is an outdent (saliency bigger than 0), it is considered as dominant if its associated saliency is bigger than all the saliences in his local neighborhood and, if p_i is an indent (saliency smaller than 0) it is considered as dominant if its associated saliency is smaller than all the saliences in his local neighborhood.

The bigger the size of the local neighborhood, the less keypoints will be selected. On the other hand, if the value of r is small, more new keypoints will be selected.

Regarding this property, it is important to notice that a dominant point using a search distance r_1 will always be dominant using search distances $r_2 \leq r_1$. This way, reducing

the local neighborhood during the keypoint selection stage can be understood as refining the model characterization. Experimental results show that a good compromise between efficiency and correction is setting the value of of r as the average value of the set of radius used for saliency estimation. However, if fragments are highly eroded or made of soft materials (like clay, for example), using a more reduced local neighborhood helps during the matching stage, since more keypoints are available to check the geometric consistency of the calculated alignment.

Figure 6.8 graphically illustrates this concept. In the left case, the search radius for the keypoint selection algorithm is the same as the one used in the highest scale for the feature extraction algorithm, $r = r_{max}$. The middle case shows the proposed average search radius, $r = (r_{max} - r_{min})/2$, whilst the right case shows the results achieved when using a very small search radius, $r = (r_{max} - r_{min})/4$. Notice how, in the three images, a dominant keypoint is highlighted. Given that this point was dominant when using the biggest radius, its dominancy is kept for all the illustrated cases. The same concept applies to all the keypoints shown in the left image.

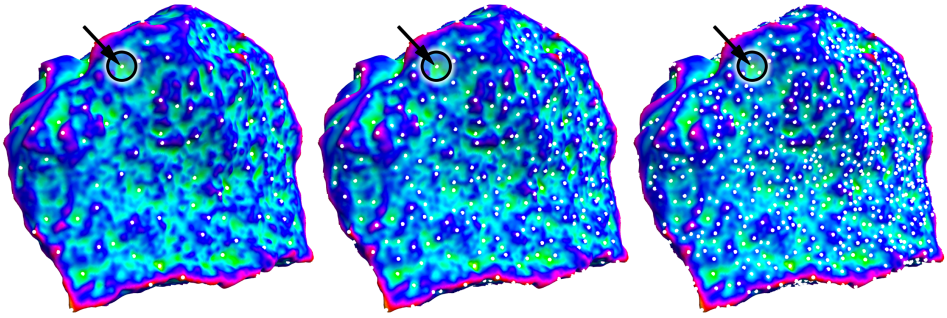

Figure 6.8: *Keypoint selection with multiple search distances. Three different searching radius (in decreasing order from left to right) over the same feature extraction. Keypoints are represented as white dots, and the fragment is shaded according to salience values, being green associated to indents and blue/magenta to outdents.*

Once keypoints are selected, the previously introduced roughness descriptor is computed only for the meaningful points. Introducing this descriptor in this stage covers two basic functions: (1) discarding keypoints that potentially lay over the original faces of the fragment and (2) removing keypoints in the fracture faces whose dominancy over their local neighborhood is very weak. To do so, a threshold based rejector evaluates all the selected points, invalidating those whose associated roughness is below to a given tolerance. As happened with the other parameters, the proper value for this one depends mostly on the kind of material considered, being higher for rock-made fragments, where roughness in the fractured faces is higher, and lower for fragments made with soft materials or highly eroded.

Figure 6.9 shows the results after performing the proposed prune of keypoints. Notice how in the top of the fragment, which is an original face, all keypoints have been rejected, reducing considerably the size of the input data for the matching algorithm. Also notice how, in the fracture faces, some weak keypoints have also been removed, contributing this way to reduce execution costs and improving search robustness.

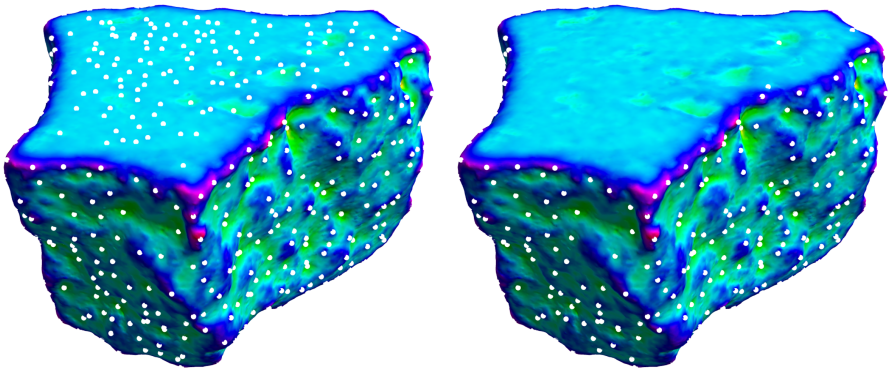

Figure 6.9: *Keypoint rejection using the roughness descriptor. (left) Keypoint selection results without considering the roughness descriptor. (right) Keypoint selection results considering the roughness descriptor.*

6.2.3 Descriptor calculation

Once keypoints are selected, the last part of the pre-processing stage deals with the descriptor extraction. The main idea is to compute, in a compact way, a local characterization of the surrounding neighborhood for each selected point to provide extra information to the search stage. This information will be helpful to reduce the number of potential correspondences between keypoints: only pairs with similar descriptors will be used to evaluate potential matches.

The selected descriptor for this purpose is a modified version of Persistent Feature Histograms (PFH). This descriptor was introduced in [139] as an alternative characterization of keypoints for point cloud alignment calculation. Given that originally it was intended to help in the registration process of partial views acquired over the same static environment, the original formulation has to be extended to support the indents and outdents concept.

To compute the descriptor associated to a given point, p_i, considering a neighborhood of size r, fist stage selects all points p_j that verify $\|p_i - p_j\| < r$. Then, for each pair of points p_i and p_j, where $i \neq j, j > i$, a source and target points are identified (p_s and p_t, respectively), being the source the one that has the smallest angle between its associated normal and the line connecting the two points:

$$p_s = \begin{cases} p_i, \ \langle n_i, p_j - p_i \rangle \leq \langle n_j, p_i - p_j \rangle \\ p_j, \ \text{otherwise} \end{cases} \tag{6.12}$$

being n_i and n_j the normals associated to p_i and p_j, respectively. This way, for each pair of points a source is uniquely defined.

Given a source point and a target point, and their associated normals, an orthonormal Darbux reference frame is built centered in the source point, as illustrated in Figure 6.10:

$$u = n_s \tag{6.13}$$

$$v = (p_t - p_s) \times u \tag{6.14}$$

$$w = u \times v \tag{6.15}$$

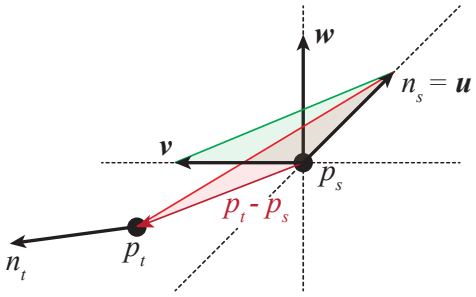

Figure 6.10: *Construction of the PFH reference frame given a source point p_s and a target point p_t and their associated normals n_s and n_t, respectively. Red triangle is the plane defined by the vectors $(p_t - p_s)$ and $n_s = u$, whilst the green one is defined by $(p_t - p_s) \times n_s = v$ and $n_s = u$*

Using the proposed frame as a reference, all surrounding points are expressed as a quadruplet of values $(\alpha, \phi, \theta, d)$ calculated as detailed below and illustrated in Figure 6.11:

$$\alpha = \langle v, n_t \rangle \tag{6.16}$$

$$\phi = \frac{\langle u, p_t - p_s \rangle}{\|p_t - p_s\|} \tag{6.17}$$

$$\theta = \text{atan}(\langle w, n_t \rangle, \langle u, n_t \rangle) \tag{6.18}$$

$$d = \|p_t - p_s\| \tag{6.19}$$

This way, for each pair of points, instead of storing 12 values (position and normal 3D coordinantes), a more compact representation is achieved, being only necessary 4 values: α, ϕ, θ and d.

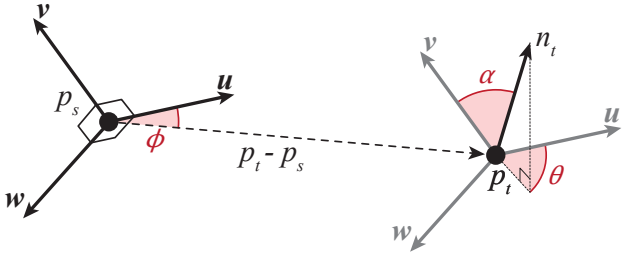

Figure 6.11: *Construction of the PFH reference frame given a source point p_s and a target point p_t and their associated normals n_s and n_t, respectively. Red triangle is the plane defined by the vectors $(p_t - p_s)$ and $n_s = u$, whilst the green one is defined by $(p_t - p_s) \times n_s = v$ and $n_s = u$*

The final PFH representation takes all the points around the selected keypoint inside a given search radius and generates the proposed alternative representation for each pair of them (Figure 6.12). Then, a histogram representation is built by spitting each value

of α, ϕ, θ and d into n bins, and counting the number of occurrences for each case. For example, if each feature is divided into into $n = 5$ correlated intervals, a histogram of 4^5 bins is created. In this space, the value stored in a bin will be incremented only if a given point has the proper value for all its 4 features.

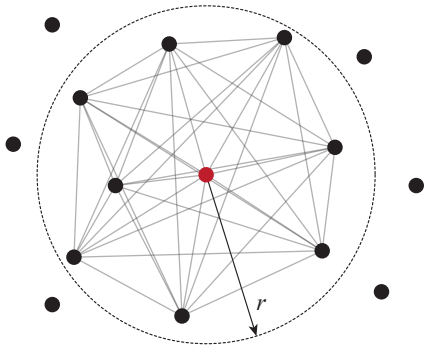

Figure 6.12: *PFH pairs between points around a selected keypoint (represented in red) with a given search radius, r.*

Finally, in order to extend the proposed descriptor to the reconstruction problem, where the indents from one fragment are supposed to be matched with outdent of another, the formulation has to be modified. To do so, for each fragment and its associated keypoints two descriptors are calculated: one using the standard PFH formulation, and another one considering $n_p = -n_p$, for all the points.

In other words, for indents to be properly matched with their corresponding outdents, one of the two descriptors used for the matching needs to be computed with the normals flipped. This way, the compact topological representation of the neighborhood around a given keypoints behaves identically as the input data PFH was designed for: partial views of the same model.

Figure 6.13 illustrates this concept with two fragments that match each other.

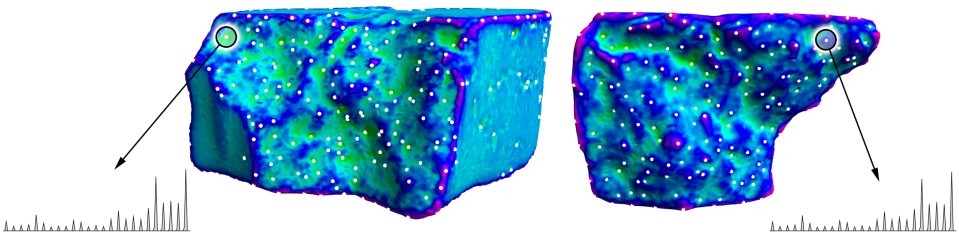

Figure 6.13: *PFH descriptors extended for reconstruction purposes. Two corresponding keypoints are marked in two different fragments (one indent and one outdent). Notice how their associated PFH histograms are practically identical, when computing the descriptor associated to the fragment shown in the left with the normals flipped.*

6.3 One-to-one search strategy

This stage takes as input data the keypoints calculated in the previous stage and their associated descriptors and outputs a rigid transformation (a rotation matrix R and a translation vector t) that better aligns both fragments. In order to be able to compare the quality of the alignment with respect to other one-to-one matches, an estimation of the quality is also provided. These results will be used in the next stage to perform the final reconstruction of the original object.

In order to be able to decide if a couple of keypoints (one from each fragment) are in correspondence, two basic conditions have to be satisfied: (1) their local neighborhood must be similar and (2) geometric consistence has to be granted with respect to other matching keypoints.

6.3.1 Local similarity

In order to measure the local similarity between a couple of keypoints, the information contained in the previously calculated descriptors is exploited. This way, given two fragments P and Q, with their associated keypoints p_i and p_j, respectively, and the descriptors associated to each keypoint (p_i^+, p_i^-) and (q_i^+, q_i^-), respectively, where $^+$ and $^-$ distinguish between the descriptors calculated using regular or inverted normals, respectively, a kd-tree of 4^n dimensions is built for each fragment using the inverted-normals descriptor (where n is the number of bins used for the PFH descriptor, and 4 corresponds to the α, ϕ, θ and d features).

For each keypoint $p_i \in P$ a k-nearest neighbor search is performed over the kd-tree created for fragment Q. Results for this search are a set of keypoints $q_j \in Q$, together with their associated distances, defined in the 4^n dimensional space, and calculated as:

$$d = \sqrt{\sum_{b=1}^{4^n} \left(p_{i_b}^+ - q_{i_b}^-\right)^2} \tag{6.20}$$

where $p_{i_b}^+$ stands for the b-th bin of the positive PFH descriptor of keypoint p, and $q_{i_b}^-$ for the corresponding b-th bin of the negative PFH descriptor of keypoint q.

Once potential correspondences between keypoints according to their local similarity are calculated, all of them are stored in a sorted array according to their associated distance. The rest of potential pairs between keypoints in both fragments will not be considered in the next search stages.

In order to reduce the number of possible combinations between the selected correspondences, this sorted list can be pruned so only the first k best matches are considered. Even if this prune operation is not performed, the total number of matches to evaluate is considerably reduced taking advantage of the previously calculated descriptor.

Figure 6.14 shows the result after this stage. Notice how, even having a powerful descriptor and a good selection of keypoints, the number of matches to consider is still very high. The main task of the search algorithm will be, then, discarding all the potential correspondences according to their geometric properties, and selecting only the ones that provide geometrically consistent results.

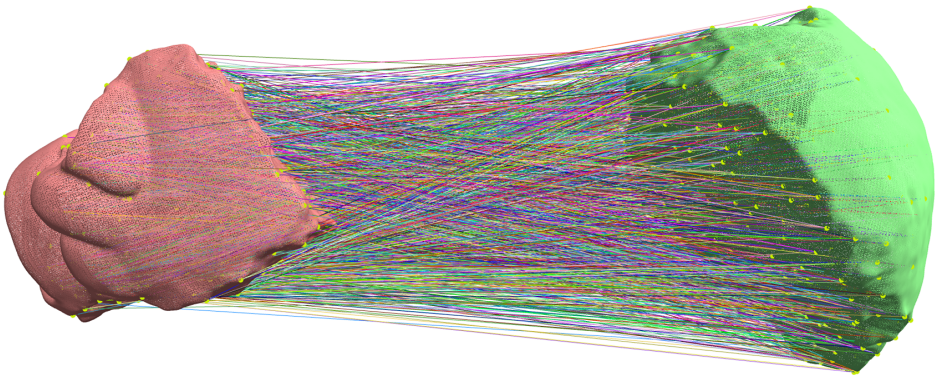

Figure 6.14: *Potential matches between two fragments considering only local similarity between keypoints.*

6.3.2 Geometric consistence

Starting from the results achieved by the local similarity search, the geometric consistence tests proposed here are divided in three major cases: (1) given a pair of matching keypoints (called *pivot* from here on), discard all the ones that make no sense in terms of geometric consistence, (2) given two pivots, perform the same operation and (3) given a roto-translation matrix that aligns fragment Q with fragment P, decide which keypoints are geometrically consistent.

Notice how, for the first proposed case, having a pivot fixed, $\pi = (p_i, q_j)$ where $p_i \in P$ and $q_j \in Q$, the three degrees of freedom associated to the relative translation that puts Q in contact with P are locked, while the three degrees of freedom associated to rotations remain unlocked. In the second case, having two pivots fixed (π_1 and π_2), two extra degrees of freedom associated to rotations get locked, leaving only the possibility to rotate with respect to the line defined by $\overline{\pi_1 \pi_2}$. Finally, in the third case, all degrees of freedom are locked, so a final rigid transformation can be computed.

In order to evaluate which pairs of keypoints (called *candidates* from here on, c_i) are geometrically consistent with a given pivot, the next conditions have to be satisfied:

$$|\|p_1 - p_2\| - \|q_1 - q_2\|| \leq \epsilon_t \tag{6.21}$$

$$|\langle n_{p_1}, n_{p_2} \rangle - \langle n_{q_1}, n_{q_2} \rangle| \leq \epsilon_r \tag{6.22}$$

$$\left| \frac{\langle p_1 - p_2, -n_{p_2} \rangle}{\|p_1 - p_2\|} - \frac{\langle q_1 - q_2, n_{q_2} \rangle}{\|q_1 - q_2\|} \right| \leq \epsilon_r \tag{6.23}$$

being $\pi_1 = (p_1, q_1)$, $c_1 = (p_2, q_2)$, n_{p_1} and n_{p_2} the normals associated to p_1 and p_2, respectively, n_{q_1} and n_{q_2} the normals associated to q_1 and q_2, respectively, ϵ_t a given tolerance for position errors and ϵ_r a given tolerance for normal orientation errors. Notice from the previous two equations that it is stated that relative distances between points in both fragments and relative orientations of the normals have to be similar.

In Equation 6.23 it is stated that the angle between the normals and the line connecting the pivot and the candidate has to be similar. Notice how the sign of n_{p_2} has been inverted.

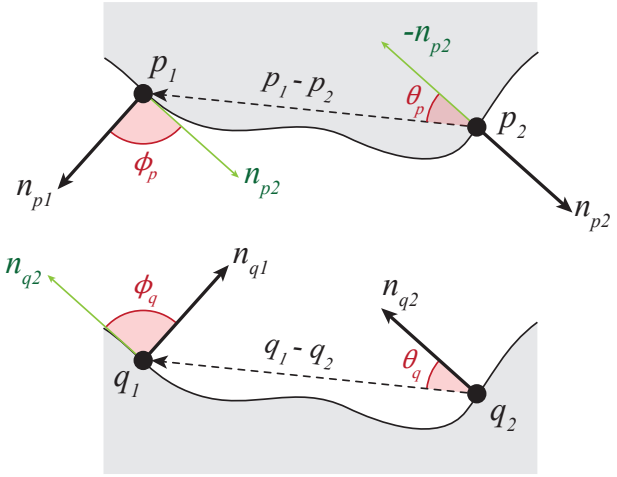

Figure 6.15: *Geometric constraints for one fixed pivot, $\pi = (p_1, q_1)$. Similarity has to be preserved for relative distances between points, relative orientations between normals (represented as ϕ_p and ϕ_q), and angles between the normals of the candidates and the line connectig each candidate with its associated pivot (represented as θ_p and θ_q).*

This condition is necessary for mathematical correction (see Figure 6.15) and, attempts to avoid penetration between fragments by forcing normals to be in opposite directions.

When two pivots are fixed, $\pi_1 = (p_i, q_j)$ and $\pi_2 = (p_k, q_l)$, $i \neq k$ and $j \neq l$, candidate correspondences are considered as valid if the next conditions are satisfied:

$$
\begin{aligned}
|\|p_1 - p_3\| - \|q_1 - q_3\|| &\leq \epsilon_t \\
|\|p_2 - p_3\| - \|q_2 - q_3\|| &\leq \epsilon_t \\
|\langle n_{p_1}, n_{p_3}\rangle - \langle n_{q_1}, n_{q_3}\rangle| &\leq \epsilon_r \\
|\langle n_{p_2}, n_{p_3}\rangle - \langle n_{q_2}, n_{q_3}\rangle| &\leq \epsilon_r \\
\left| \frac{((p_3 - p_1) \times (p_3 - p_2)).\text{norm}()}{(p_2 - p_1).\text{norm}()} - \frac{((q_3 - q_1) \times (q_3 - q_2)).\text{norm}()}{(q_2 - q_1).\text{norm}()} \right| &\leq \epsilon_t \\
|\langle (p_3 - p_1) \times (p_3 - p_2).\text{normalized}(),\ p_1 - (p_3 + n_{p_3})\rangle - & \\
\langle (q_3 - q_1) \times (q_3 - q_2).\text{normalized}(),\ q_1 - (q_3 - n_{q_3})\rangle| &\leq \epsilon_r
\end{aligned}
$$

being $\pi_1 = (p_1, q_1)$, $\pi_2 = (p_2, q_2)$, $c_1 = (p_3, q_3)$, the operator norm() returns the distance of the preceding vector and the operator normalized() expresses that the preceding vector has module 1.

First two conditions ensure that relative distances between the candidate points and their corresponding pivots are similar. Next two conditions preserve the similarity between the normals of the candidates and the normals of their corresponding pivots. Fifth condition compares the distance of the candidate with respect to the line that connects its two corresponding pivots. Last condition compares the normal of the candidate with respect to the plane defined by the two pivots and the candidate (see Figure 6.16).

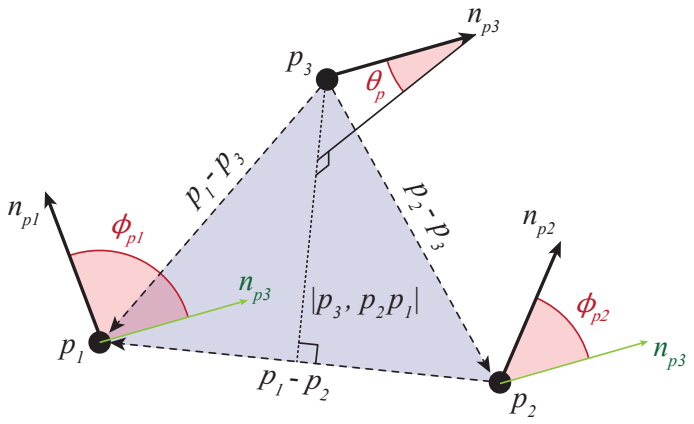

Figure 6.16: *Geometric constraints for two fixed pivots, $\pi_1 = (p_1, q_1)$ and $\pi_2 = (p_2, q_2)$. Similarity has to be preserved for relative distances between points, relative orientations between normals (represented as ϕ_{p_1} and ϕ_{p_2}), the distance between the candidate and the line connecting the two pivots, and for the relative orientation of the candidate's normal with respect to the normal of the plane defined by the two pivots and the candidate (represented as θ_p).*

Finally, when three pivots are fixed, $\pi_1 = (p_i, q_j)$, $\pi_2 = (p_k, q_l)$ and $\pi_2 = (p_m, q_n)$, , $i \neq k \neq m$ and $j \neq l \neq n$, all degrees of freedom are locked, so the only conditions that have to be satisfied for each candidate correspondence are:

$$
\begin{aligned}
&|\|p_1 - p_4\| - \|q_1 - q_4\|| \leq \epsilon_t \\
&|\|p_2 - p_4\| - \|q_2 - q_4\|| \leq \epsilon_t \\
&|\|p_3 - p_4\| - \|q_3 - q_4\|| \leq \epsilon_t \\
&|\langle n_{p_1}, n_{p_4}\rangle - \langle n_{q_1}, n_{q_4}\rangle| \leq \epsilon_r \\
&|\langle n_{p_2}, n_{p_4}\rangle - \langle n_{q_2}, n_{q_4}\rangle| \leq \epsilon_r \\
&|\langle n_{p_3}, n_{p_4}\rangle - \langle n_{q_3}, n_{q_4}\rangle| \leq \epsilon_r
\end{aligned}
$$

being $\pi_1 = (p_1, q_1)$, $\pi_2 = (p_2, q_2)$, $\pi_3 = (p_3, q_3)$ and $c_1 = (p_4, q_4)$.

6.3.3 Search strategy

The proposed search strategy for one-to-one alignments starts by reducing the search space using the explained local similarity function. When all potential matches between keypoints have been found, a set of alignments is created as follows: for each match between keypoints, a new alignment is created fixing it as a pivot and adding all the rest of correspondences as candidates (if they satisfy all the geometrical consistence conditions explained before for one pivot).

In order to sort alignments according to their potential quality, a cost function is defined by adding the number of pivots and the number of candidates. This way, a max-heap is built with all the possible combinations, where the top of the tree is the alignment with

better cost function (in the initialization case, the one that has more correspondences geometrically consistent with the selected pivot).

Iteratively, the most promising alignment is popped from the heap, and refined into a set of *child* alignments. To compute these new alignments, the proposed search strategy proceeds as follows:

Algorithm 5 GetChildren($Alignment\ A$)

```
1: result ← List(Alignment);
2: for i ← 0 to A.candidates.Count step 1 do
3:    child ← Alignment();
4:    child.pivots ← A.pivots + A.candidates[i];
5:    child.candidates ← A.candidates[i + 1 ... A.candidates.Count];
6:    child.RemoveInconsistentCandidates();
7:    result.Add(child);
8: end for
9: return result;
```

The function $RemoveInconsistentCandidates()$ associated to an alignment takes in consideration the total amount of fixed pivots and applies the proper set of geometric consistence tests.

Once child alignments have been calculated, they are added to the heap of active alignments, and the process continues until the top of the heap (the alignment with more correct candidates) reaches three fixed pivots. When this happens, all degrees of freedom get locked, and a transformation that aligns fragment Q with P is computed as follows:

$$\mu_P = \frac{1}{3}\sum_{i=1}^{3} p_i \quad , \quad p_1 \in \pi_1,\ p_2 \in \pi_2,\ p_3 \in \pi_3 \tag{6.24}$$

$$\mu_Q = \frac{1}{3}\sum_{i=1}^{3} q_i \quad , \quad q_1 \in \pi_1,\ q_2 \in \pi_2,\ q_3 \in \pi_3 \tag{6.25}$$

$$H = \frac{1}{3}\sum_{i=1}^{3} (p_i - \mu_P)\,(q_i - \mu_Q)^T \tag{6.26}$$

$$[U, S, V] = svd(H) \tag{6.27}$$

$$R_{3\times 3} = VU^T \tag{6.28}$$

$$t_{3\times 1} = -R \times (\mu_P + \mu_Q) \tag{6.29}$$

The result of these operations are a rotation matrix, R, and a translation vector, t, that align pivots in fragment Q with their counterpart in P when $q'_i = R * q_i + t$.

After performing this operation considering only the three fixed pivots, the set of geometric consistence tests for a given roto-translation are applied to all the candidates. After removing the ones that make no geometrical senses, a new set of rotation and translation matrices are calculated again, as explained in the previous formulation, but considering all the pivots and candidates that remain valid in the alignment.

Then, the resulting alignment is pushed back into the heap, promoting all the candidate correspondences as pivots. The iterative search process ends when the top of the heap is

an alignment with more than 3 pivots. In this case, the alignments is returned as the result of the process, together with the calculated roto-translation, and considering as cost function the total number of correspondences marked as pivots (notice that, at this point, no candidate correspondences are available since they have been promoted to pivots if they made geometrical sense or deleted if not). The general search algorithm is detailed below:

Algorithm 6 One-to-OneSearch($Keypoints\ P$, $Keypoints\ Q$)

1: $KdTree$ **kd** $\leftarrow Q.buildKdTreeOfDescriptors()$;
2: $List(Correspondence)$ **correspondences** $\leftarrow$ **kd**$.kNeighbors(P.descriptors)$;
3: **correspondences**$.sortOnDescriptorDistance()$;
4: **correspondences**$.removeLast(n)$;
5: $Heap$ **alignments** $\leftarrow Heap(Alignment)$;
6: **for** $i \leftarrow 0$ **to correspondences**$.Count$ **step** 1 **do**
7: $Alignment$ **A** $\leftarrow Alignment()$;
8: **A**$.pivots =$ **correspondences**$[i]$;
9: **A**$.candidates =$ **correspondences**$[i+1 \ldots$ **correspondences**$.Count]$;
10: **alignments**$.push($**A**$)$;
11: **end for**
12: **while alignments**$.top.pivots.Count < 4$ **do**
13: $Alignment$ **current** $\leftarrow$ **alignments**$.removeTop()$;
14: **if current**$.pivots < 3$ **then**
15: **alignments**$.push($**current**$.getChildrenAndEvaluateConsistence()$;
16: **else**
17: **alignments**$.push($**current**$.computeRotoTranslation()$;
18: **end if**
19: **end while**
20: **return alignments**$.top()$

This way, a three-level hierarchical search is performed, where not all possible combinations of three correspondences have to be exhaustively evaluated. Considering that correspondences are sorted with respect to their euclidean distance in the descriptor space (line 3 on the previous algorithm), and that pivots are fixed in potential alignments in this order, a normal execution of the proposed search strategy converges very fast to the global solution, being only necessary to evaluate the first two-to-six alignments created in the loop shown in line 6.

Figure 6.17 shows the results using the proposed search strategy for the same problem illustrated in Figure 6.14. Notice how, for the thousands of possible correspondences between keypoints considering only local similarity, only a subset of 19 of them make geometrical sense, according to the stated restrictions.

6.4 Many-to-many search strategy

Last stage in the global reconstruction proposed execution cycle deals with the many-to-many searches between the fragments of a given dataset. Input data are a set of alignments calculated using the one-to-one search strategy explained before and the result is a set of clusters of matching fragments.

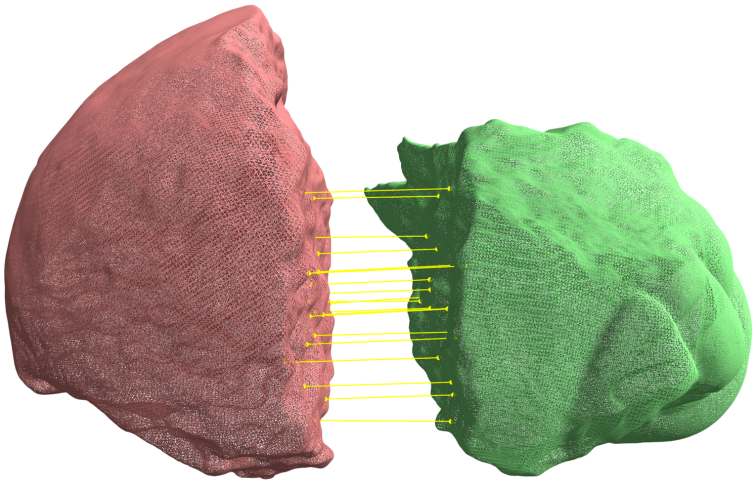

Figure 6.17: *Results of the one-to-one search strategy.*

Each of the clusters considered during this global reconstruction stage can be seen as a graph, where each node represents a fragment with its associated pose, and each edge represents a pair of matching keypoints connecting two fragments.

Considering that the previously proposed one-to-one strategy performs a three-level hierarchical search, in order to solve the many-to-many correspondence problem, the heap of the active alignments could be initialized considering all the one-to-one pairs of fragments. This approach would be similar to the one presented in the 3-DOF problem, but with one major difference: unlike the former solution, the 6-DOF hierarchical search only considers 3 different levels (one pivot, two pivots and three or more pivots). Trying to solve all the one-to-one correspondences at once would not yield into a great performance boost since, at least, the first level on the hierarchy must be exhaustively evaluated. Also, the major drawback of a pure hierarchical search is that, until the most promising alignment has not been refined into its child alignments, it cannot be decided which is the next most potential alignment. This way, the problem cannot be divided into independent tasks, so no parallelization can be performed.

As an alternative way of solving all the one-to-one correspondences, and considering the hardware architecture of modern CPUs, the proposed search algorithm starts by creating a list of one-to-one alignments to compute, and splitting this list into a set of n tasks, where n corresponds to the number of logic processors in the machine that is executing the algorithm. This way, since each one-to-one search is independent from the others, n threads are created and n simultaneous one-to-one alignments are solved at the same time.

Once all the searches have finished, results are collected in a list, where the alignments are sorted according to their cost function (number of pivots) in descending order. Then, the global reconstruction process begins.

In order to perform the final re-assembly of the original fragment, a set of operations with clusters are defined: *create* operations, where new clusters are defined, *append* operations, where new correspondences between fragments are added to an existing cluster and *merge* operations, where two existing clusters are combined into a single one.

First operation in this stage is a *create* operation, since no clusters exist during the initialization. Basically, the alignment with higher cost function is popped from the sorted list of one-to-one alignments, and a cluster is created with two nodes: one for the first fragment P, with an identity transformation matrix associated, and another for fragment Q, with a transformation matrix, $M_{4\times4}^{(Q)}$ created by composing the rotation matrix, R, and the translation vector, t, calculated in the alignment:

$$M_{4\times4}^{(Q)} = \begin{bmatrix} R(1,1) & R(1,2) & R(1,3) & t(1,1) \\ R(2,1) & R(2,2) & R(2,3) & t(2,1) \\ R(3,1) & R(3,2) & R(3,3) & t(3,1) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.30}$$

Then, a set of attributed edges connecting nodes P and Q are defined, where each edge contains the index of the two keypoints matching in the nodes connected. Figure 6.18 shows the graphical representation of a cluster just created. In the left image the complete representation is shown, whilst in the right image, a compact alternative representation is shown, where all edges are condensed into a single one, pondered with the number of matching keypoints. From this point, this is the representation that will be used in figures.

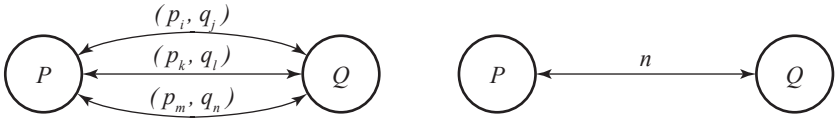

Figure 6.18: *Cluster representation as a graph. Resulting cluster after performing a create operation. Left image shows the complete representation, whilst right image shows the compact representation.*

After the first cluster has been created, two possibilities exist: (1) the next best match does not include none of the two fragments enclosed in the first cluster, so a new cluster has to be created, or (2) the next best match includes one of the previously enclosed fragments, so an *append* operation has to be performed.

Append operations add new correspondences to existing clusters. There are two different cases for this kind of operations: (1) just one fragment is already present in the cluster or (2) both fragments are already present in the fragment, but not previously connected.

In the first case, a new node is created, containing the newly added fragment, R, and connections between R and its matching fragment, P, are defined. In order to compute the transformation matrix of R, two possibilities exist:

1. R is the moving fragment in the alignment that connects it with P, so its transformation matrix is defined as $M_R = M_P * R + t$, where R and t are the rotation matrix and translation vector of the alignment, respectively, and M_P is the transformation matrix of P.

2. R is the fixed fragment in the alignment that connects it with P, so its transformation matrix is defined as $M_R = M_P * [Rt]^{-1}$, where $[Rt]^{-1}$ is the inverse of the 4×4 matrix built by appending t to R and adding [0 0 0 1] to the last row.

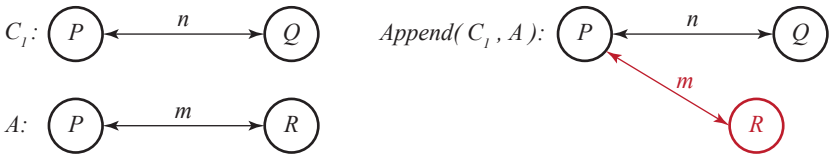

Figure 6.19: *Cluster append operation with one fragment of the alignment inside the considered cluster. Given one cluster, C_1, and one alignment, A, (left), result of the append operation when only one of the fragments considered in A exists in C_1 (right).*

Figure 6.19 illustrates this case for the append operation.

In the case of both fragments considered in the new alignment being part of a given cluster, a geometric consistency test is performed to evaluate if the new alignment makes sense with the previous poses of the fragments involved. This way, by fixing the pose of one of the fragments of the alignment to the one it has in the cluster, and by computing the new pose for the other fragment according to the criteria previously explained, a new transform is computed. If this transform is not too different for the one that already had inside the cluster, a new set of edges connecting both fragments are created. Otherwise, the alignment is considered as incorrect and, thus, rejected.

Notice how, when this situation happens, the pose of the moving fragment was already calculated using matches with higher associated cost functions (since fragments are added according to this criterion). This way, global consistency is evaluated in a very fast way, and potential incorrect correspondences can be identified and rejected. Figure 6.20 illustrates this particular case of the append operation.

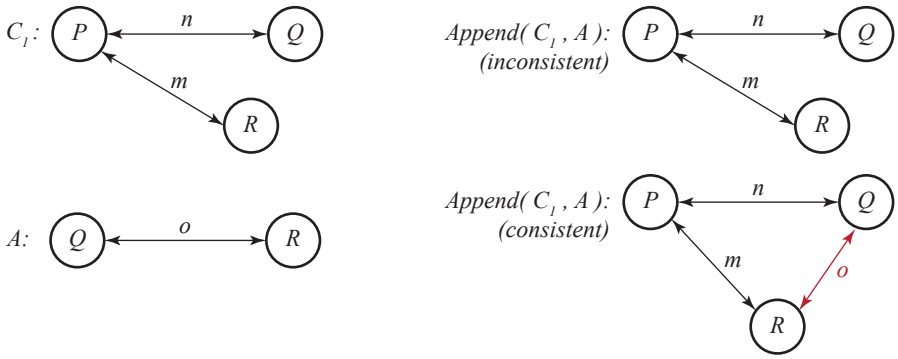

Figure 6.20: *Cluster append operation with both fragments of the alignment inside the considered cluster. Given one cluster, C_1, and one alignment, A, (left), result of the append operation when both fragments considered in A exists in C_1. (right-top) Results when the consistence test is not passed. (right-bottom) Results when the consistence test is passed.*

Finally, the *merge* operation is performed when the most promising alignment in the sorted list of one-to-one matches, includes one fragment from one cluster and another fragment from other cluster. In this case, both previously defined clusters are removed,

and a new one is created containing the union of the nodes and edges defined in its parent clusters, together with the newly added alignment. To update the transformation matrices of all the fragments, one of them is fixed and the others are updated in cascade according to the criterion already commented.

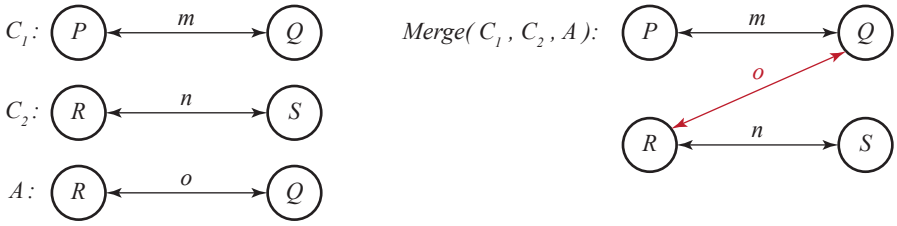

Figure 6.21: *Cluster merge operation. Given two clusters, C_1 and C_2, and one alignment, A, whose fragments are present in C_1 and C_2 (left), result of the merge operation (right).*

Given these operations, the global reconstruction process iterates through all the potential one-to-one alignments creating, appending and joining clusters until no alignment is left. This means that, for the first set of alignments (the ones that have a higher cost function), lots of changes are introduced in the final reconstructed object. When the best alignments have been processed, most of the subsequent results are discarded, since they contradict the previously computed poses.

Notice that $(n-1)!$ one-to-one searches are produced, being n the number of fragments in the dataset, and most of them will provide "good" results (alignments with more than three pivots in correspondence). By evaluating the global correction, these incorrect results are very easily identified and rejected.

6.5 Results

This section empirically evaluates the correction and performance of the proposed technique. To do so, the fragments presented in [71] are used. These datasets are courtesy by the Vienna University of Technology, and available on-line for research purposes. All tests presented here have been executed in a 3.4 GHz Intel Core i7 computer with 4GB of RAM.

In next subsections, results achieved in each stage of the proposed execution cycle are analyzed: from the pre-processing of the original point clouds, to the one-to-one search strategy, to the final reconstruction algorithm.

6.5.1 Pre-processing

In order to measure the performance of the pre-processing stage, all fragments in the considered datasets have been processed with the same parameters: a 10 level multi-scale feature extraction, with the neighbor search distances linearly distributed between 0.1 to 1.0 centimeters, a keypoint extraction based on the proposed dominancy value, $(r_{max} - r_{min})/2 = 0.5$ centimeters and a PFH descriptor where distances are ignored (since models are so dense and uniformly sampled, distances between pairs of points do

not provide much information) and each relative orientation with respect to the reference frame is splitted into 5 bins, so the final descriptor is a histogram of $5^3 = 125$ float values.

Next tables show the achieved results, where the first column, $Frag$, corresponds to the file name of the fragment, second column, $\#Vert$, is the total number of points in the original point cloud, third column, $\#nn$, is the average number of nearest neighbors for each point when using the maximum search distance (which provides an estimation on the model density of points), fourth column, $knn(sec)$, is the time spent in computing all the nearest neighbors searches, fifth column, $Feat(sec)$, is the time spent in extracting the proposed multi-scale feature, sixth column, $Key(sec)$, corresponds to the time spent in extracting keypoints, seventh column, $Rough(sec)$, is the time required to compute the roughness of the selected keypoints and filter the ones that are below the given tolerance, eighth column, $Desc(sec)$ corresponds to the time spent in calculating the proposed PFH descriptors for both characterizations (with regular normals and inverted normals) and, finally, last column is the total pre-processing time of the fragment.

Frag	#Vert	#nn	knn (sec)	Feat (sec)	Key (sec)	Rough (sec)	Desc (sec)	Total (sec)
1.obj	185833	389,8	7,463	2,573	0,097	0,087	2,076	12,296
2.obj	80116	430,2	3,487	1,234	0,049	0,049	1,098	5,916
3.obj	69443	412,0	2,733	0,976	0,035	0,030	0,699	4,474
4.obj	54197	505,6	2,641	1,001	0,032	0,037	0,895	4,606
5.obj	18441	529,3	0,935	0,340	0,012	0,009	0,202	1,499
6.obj	36229	519,2	1,858	0,630	0,021	0,019	0,373	2,900
7.obj	20381	614,2	1,170	0,418	0,013	0,012	0,306	1,919

Table 6.1: *Pre-processing times for the venus dataset.*

Frag	#Vert	#nn	knn (sec)	Feat (sec)	Key (sec)	Rough (sec)	Desc (sec)	Total (sec)
01.obj	138694	840,1	11,697	3,464	0,102	0,090	2,604	17,958
02.obj	146464	958,0	13,245	4,211	0,116	0,102	3,164	20,837
03.obj	154838	809,1	11,642	3,811	0,113	0,105	2,971	18,641
04.obj	90789	354,0	3,266	1,098	0,042	0,039	0,866	5,310
05.obj	135535	986,6	12,925	3,970	0,110	0,093	2,861	19,959
06.obj	149336	748,4	10,624	3,411	0,107	0,098	2,293	16,533
07.obj	112363	520,8	5,577	1,873	0,066	0,056	1,040	8,611
08.obj	135055	608,0	8,335	2,550	0,091	0,075	1,560	12,611
09.obj	149837	597,0	8,990	2,834	0,099	0,084	1,516	13,523
10.obj	136698	988,7	12,589	3,991	0,112	0,096	2,864	19,652
11.obj	98084	622,8	5,926	1,947	0,065	0,056	1,196	9,190

Table 6.2: *Pre-processing times for the cake dataset.*

Frag	#Vert	#nn	knn (sec)	Feat (sec)	Key (sec)	Rough (sec)	Desc (sec)	Total (sec)
1.obj	174014	711,7	12,047	3,987	0,119	0,116	2,608	18,877
10.obj	111879	578,0	6,122	2,122	0,072	0,079	1,739	10,134
11.obj	132920	544,1	7,044	2,393	0,083	0,086	1,791	11,396
12.obj	122982	2.130,4	24,142	8,259	0,188	0,194	6,616	39,399
13.obj	72461	1.651,2	10,826	3,617	0,090	0,077	2,639	17,249
14.obj	141851	495,7	6,725	2,370	0,077	0,079	1,515	10,766
15.obj	126422	784,3	9,159	3,071	0,092	0,090	2,159	14,570
16.obj	101445	1.065,5	10,436	3,216	0,089	0,087	2,484	16,311
17.obj	77285	852,7	6,298	2,113	0,058	0,058	1,530	10,057
18.obj	124089	579,3	7,147	2,322	0,080	0,076	1,433	11,058
19.obj	161356	811,8	12,508	3,979	0,118	0,112	2,692	19,410
20.obj	67317	1.306,3	8,690	2,635	0,071	0,056	1,515	12,967
21.obj	120157	6.890,5	228,059	41,417	11,659	3,181	15,826	300,141
22.obj	131421	2.475,0	31,613	10,100	0,235	0,217	7,650	49,816
23.obj	81085	3.156,2	23,807	7,996	0,192	0,153	5,267	37,414
24.obj	66150	2.861,7	17,525	5,851	0,134	0,100	3,595	27,205
25.obj	94084	6.469,2	85,541	24,866	0,416	0,351	15,658	126,832
2A.obj	145973	938,1	12,593	4,104	0,113	0,122	3,351	20,283
2B.obj	118270	1.082,6	11,911	3,775	0,106	0,100	3,281	19,174
2C.obj	147204	919,0	12,498	4,031	0,111	0,105	3,097	19,842
3.obj	146258	627,4	8,665	2,830	0,097	0,099	2,657	14,349
4.obj	106489	396,8	4,191	1,391	0,052	0,052	1,174	6,860
5A.obj	146906	353,2	5,220	1,747	0,065	0,075	1,702	8,809
5B.obj	92887	1.557,3	13,345	4,267	0,110	0,122	4,583	22,426
6A.obj	87836	838,9	6,837	2,230	0,067	0,069	2,177	11,381
6B.obj	152881	726,7	10,804	3,334	0,108	0,108	2,443	16,797
7.obj	128676	566,7	7,193	2,280	0,082	0,085	2,013	11,653
8.obj	121254	894,2	10,116	3,191	0,092	0,091	2,568	16,058
9.obj	111897	578,4	6,069	2,000	0,072	0,075	1,682	9,898

Table 6.3: *Pre-processing times for the gargoyle dataset.*

Frag	#Vert	#nn	knn (sec)	Feat (sec)	Key (sec)	Rough (sec)	Desc (sec)	Total (sec)
1.obj	295487	917,0	27,468	8,876	0,244	0,258	7,180	44,025
2.obj	171707	1.687,4	27,205	9,642	0,217	0,186	6,213	43,463
3.obj	180248	1.477,5	26,305	9,088	0,210	0,208	6,734	42,545
4.obj	274032	1.366,9	41,763	14,558	0,305	0,313	10,853	67,791
5.obj	280763	1.234,4	34,986	11,864	0,288	0,277	8,972	56,387
6.obj	167468	1.617,9	25,325	8,709	0,206	0,208	7,070	41,519

Table 6.4: *Pre-processing times for the brick dataset.*

Frag	#Vert	#nn	knn (sec)	Feat (sec)	Key (sec)	Rough (sec)	Desc (sec)	Total (sec)
1.obj	141271	1.500,7	22,478	6,413	0,163	0,116	3,166	32,336
10.obj	104749	1.098,8	10,661	3,575	0,095	0,067	1,718	16,116
11.obj	102985	2.147,6	21,100	6,863	0,158	0,119	4,386	32,625
12.obj	87634	2.791,3	23,228	7,718	0,173	0,108	3,560	34,787
13.obj	104944	2.945,1	32,762	10,161	0,219	0,138	5,145	48,426
14.obj	96881	912,6	8,564	2,793	0,074	0,049	0,954	12,433
15.obj	119678	1.010,9	11,462	3,659	0,099	0,061	0,898	16,178
2.obj	110021	1.189,7	12,409	3,911	0,108	0,093	2,249	18,770
3.obj	115932	1.126,2	12,360	3,892	0,108	0,077	1,897	18,335
4.obj	161158	1.210,0	18,799	5,901	0,161	0,116	2,845	27,822
5.obj	113270	991,0	10,548	3,313	0,093	0,070	1,607	15,631
6.obj	133121	887,3	11,932	3,531	0,099	0,066	1,287	16,916
7.obj	79554	660,5	5,088	1,623	0,053	0,043	0,767	7,574
8.obj	89634	744,6	6,288	2,022	0,064	0,055	1,046	9,475
9.obj	95271	729,1	6,601	2,136	0,067	0,058	1,165	10,026

Table 6.5: *Pre-processing times for the sculpture dataset.*

According to the previous tables, the average pre-processing time per fragment is 25.305 seconds, distributed as follows: 66,22% of the time is consumed in computing the nearest neighbors for each point, 19.53% of the time is dedicated to extracting features at 10 different scales,1.12% of the time is spent in extracting keypoints, 0.58% of the time is consumed in computing the roughness of the selected keypoints and removing the incorrect ones and, finally, the remaining 12.54% of the time is dedicated to computing the PFH descriptors. Figure 6.22 illustrates this time distribution.

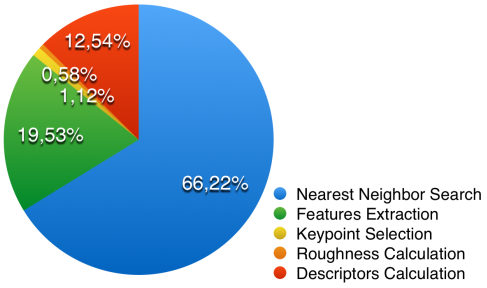

Figure 6.22: *Pre-processing time distribution.*

As it can be noticed, searching for neighbors is the most time-consuming task. This search is performed over an optimized kd-tree and simultaneously for all the points in the fragment. This setting provides faster results than individual searches, but makes parallelization very inefficient. In order to reduce the impact of the local searches in the pre-processing stage, the proposed implementation starts by caching all the neighbors for each

point in a local list, according to the higher search radius. This way, the search process has to be performed only once and results are re-used for smaller search radius. Nevertheless, despite of this optimization, local searches are still the main bottleneck.

From previous tables, it can be noticed how the total pre-processing time is linearly affected by the total amount of points in a given fragment. However, in cases like fragment $21.obj$ from the gargoyle dataset, extremely high values of density of points (6.890 neighbors using a 1 centimeter search radius) produce a radical increment on pre-processing times (300 seconds). These cases should be identified in a prior step, so a point reduction algorithm could be executed to prevent this from happening.

This way, a new linear relationship can be identified: pre-processing times also depend on the density of the input point cloud.

Figure 6.23 shows the relationship between the number of points in a fragment and the total pre-processing time (left) and the point density and the total pre-processing time (right). To compute the tendency line, all cases have been considered. However, to facilitate the lecture of the figure, the vertical axis is clamped to a maximum value of 80 seconds, so the two most time-consuming fragments are not displayed.

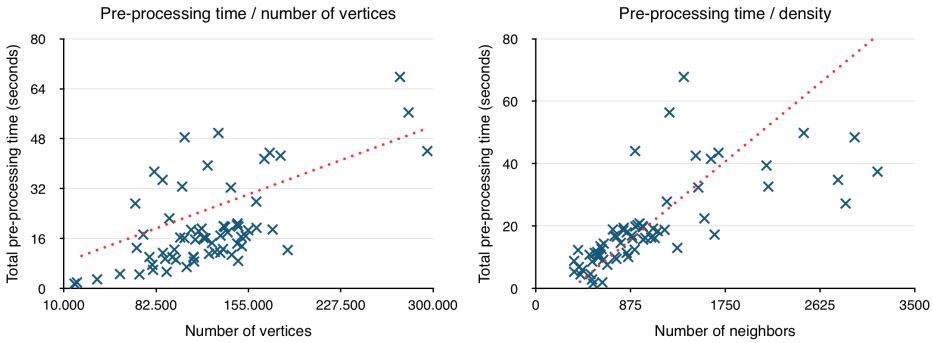

Figure 6.23: *Pre-processing time respect to the number of points (left) and density (right).*

6.5.2 One-to-one search

To evaluate the one-to-one search algorithm, the previous datasets have been used and, for each one, each fragment has been aligned with the rest in the same dataset. Performance of the search process is measured using two scalar values: the time spent on finding the best registration and the number of matching samples between both fragments.

In order to evaluate the impact of the parameters in the search algorithm (number of neighbors considered, position tolerance...), three search presets have been used (fast, medium and precise). Table 6.6 shows the values for these presets, where the column $\#neighbors$ corresponds to the number of k-nearest neighbors calculated for each keypoint, $\#correspondences$ corresponds to the number of best matches between keypoints that are used as pivots, ϵ_t is the maximum separation between two facing keypoints to be considered geometrically consistent and ϵ_r is the maximum normal orientation difference between two facing keypoints to be considered geometrically consistent.

Despite only $\#correspondences$ are used as pivots to evaluate the potential alignments, all correspondences between keypoints are used to evaluate their associated score.

Preset	#neighbors	#correspondences	ϵ_t (cm)	ϵ_r (º)
Fast	5	100	2	3,6
Medium	10	300	2	3,6
Precise	15	1000	2	3,6

Table 6.6: *One-to-one search presets.*

Figures 6.24, 6.25, 6.26, 6.27, 6.28 show the results achieved during the one-to-one search process for all the considered fragment pairs, using the three introduced search presets. In order to facilitate the interpretation of the results, search times and scores are displayed as density matrices, where element $m_{i,j}$ represents the search time / score of comparing fragment i with fragment j, where $i \neq j$. Complete tables of results can be found in Appendix A. Table 6.7 shows the average execution times and scores for all the compared fragments in each dataset.

Notice how increasing the number of potential matches considered as pivots has an exponential impact in the search time. However, results also show that one-to-one alignments are computed extremely fast (less than a second in the worst case), and that the final score density for a given dataset presents a similar distribution as the precision increases. Taking as example the brick dataset (Figure 6.24), no new significative one-to-one matches appear as the search quality is increased. This means that, using a very fast search preset, correspondences between fragments 1-{2, 3, 4}, 2-5, 4-{5, 6} and 5-6 can be identified without the need to increase precision, since no new significative matches appear.

However, for more complex cases like the gargoyle dataset or the venus dataset, higher precision might be needed. It has been empirically observed that the $medium$ preset provides a good balance between correction and efficiency.

It is also important to notice that the score function increases as the search precision does. This is an interesting aspect for the final registration since, the more matching keypoints are found, the more information is available for computing the final roto-translation matrix that puts both fragments in contact.

	Fast time	Medium time	Precise time	Fast score	Medium score	Precise score
Brick	0,0233	0,0675	0,2051	5,5333	16,3333	19,7333
Cake	0,0371	0,094	0,3111	6,8909	16,8909	19,2364
Gargoyle	0,0136	0,0362	0,135	1,0148	5,8547	7,1108
Sculpture	0,0062	0,0276	0,0985	3,0571	8,7333	9,9429
Venus	0,0124	0,0272	0,0884	2,4286	7,5714	9,381

Table 6.7: *One-to-one average searching times (in seconds) and scores for all datasets.*

Notice that one-to-one searches take one or two orders of magnitude below the second (depending on the selected preset), and how the score function increases as the search precision does. Also notice how, the bigger the dataset, the smallest the average score function is. This is a consequence of the combinatory explosion that happens when lots of fragments are considered since, in real cases, one fragment only has valid correspondences with a very reduced set of neighbors (typically between 3 and 5 fragments). In this sense, the gargoyle dataset is a good example (Figure 6.26).

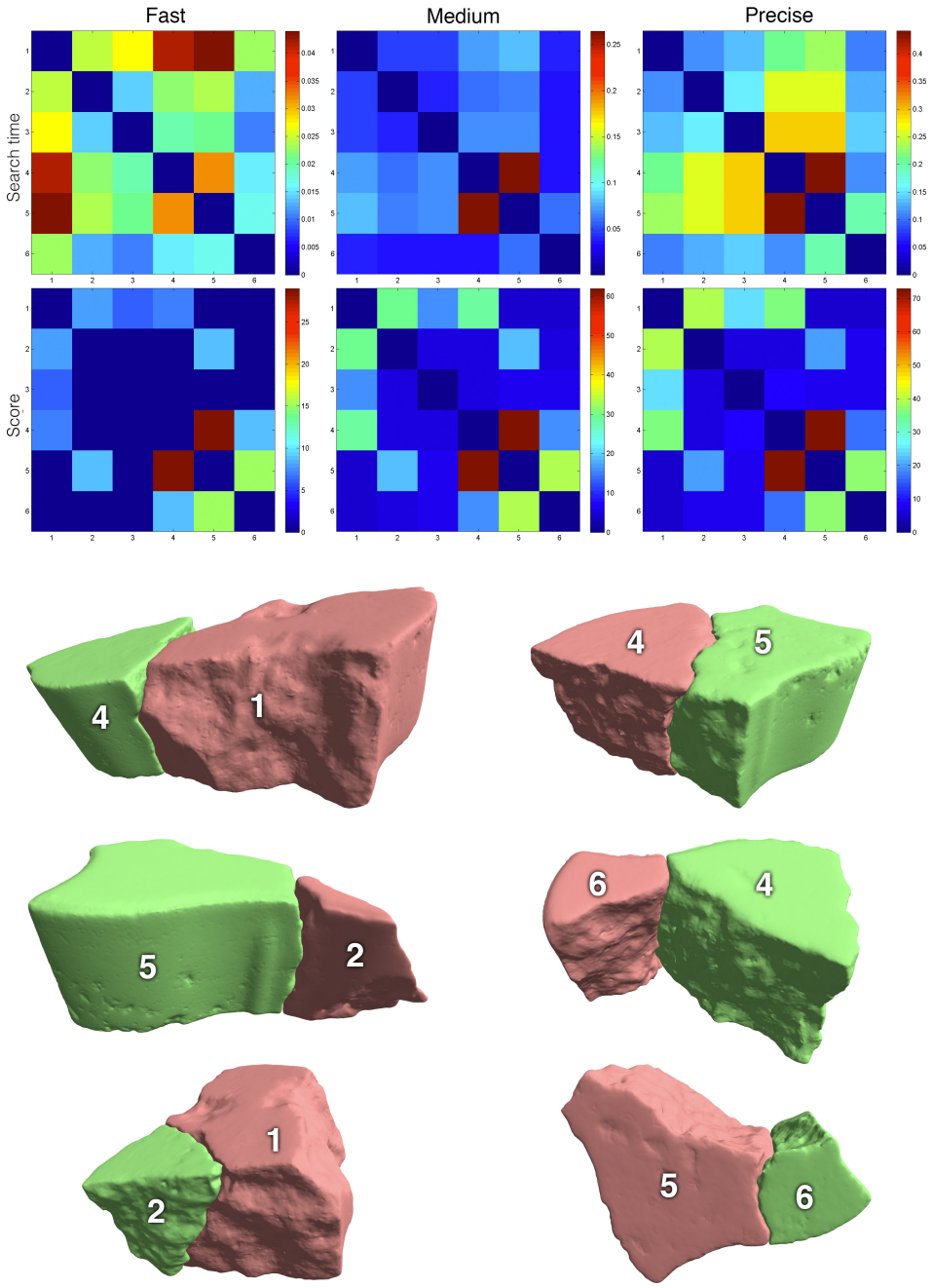

Figure 6.24: *One-to-One search results for the brick dataset. Top row shows the search times (in seconds) for each pair of fragments, whilst bottom row shows the score of the final alignment found. Bottom images show some of the best correspondences found, together with the fragment numbers.*

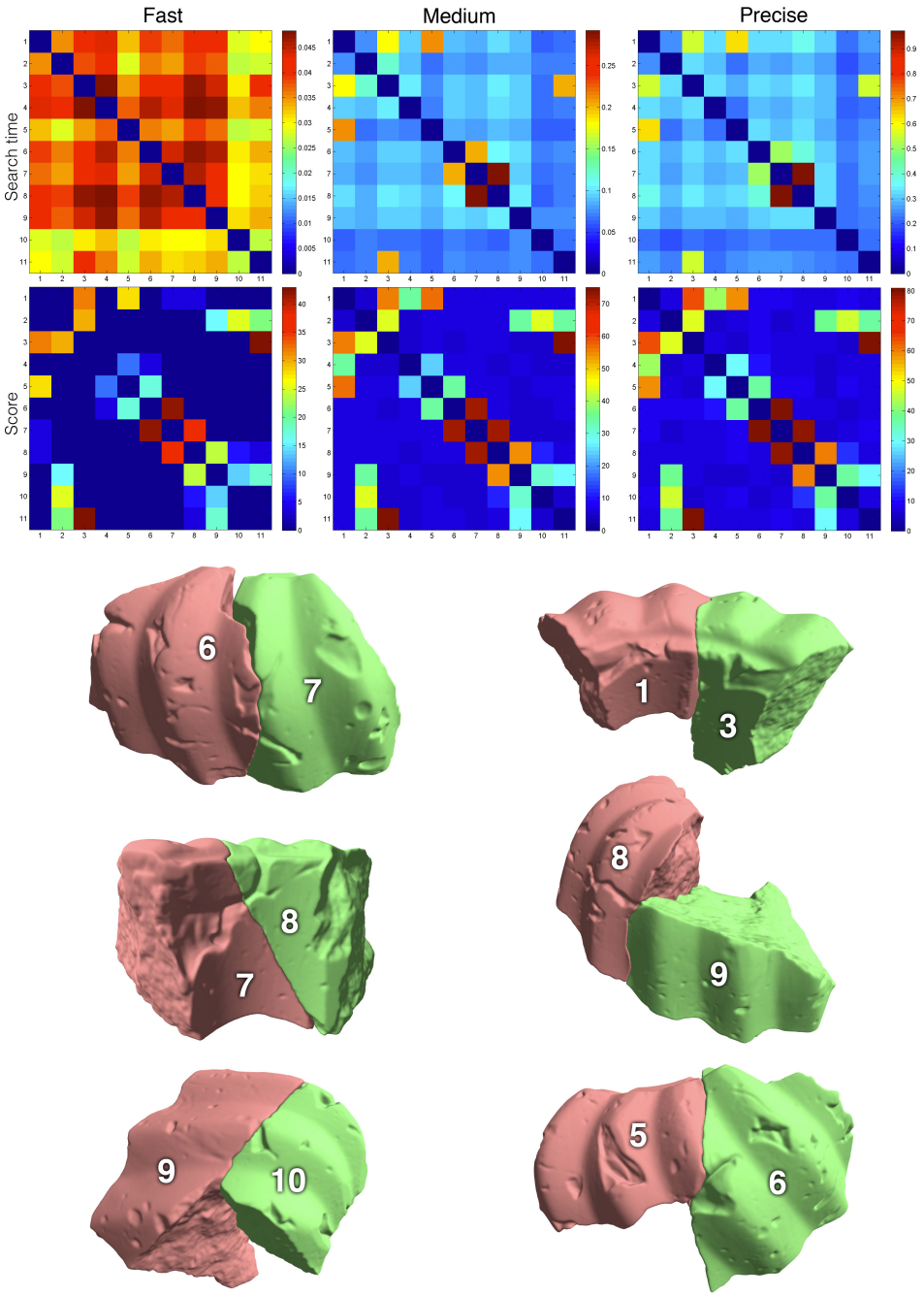

Figure 6.25: *One-to-One search results for the cake dataset. Top row shows the search times (in seconds) for each pair of fragments, whilst bottom row shows the score of the final alignment found. Bottom images show some of the best correspondences found, together with the fragment numbers.*

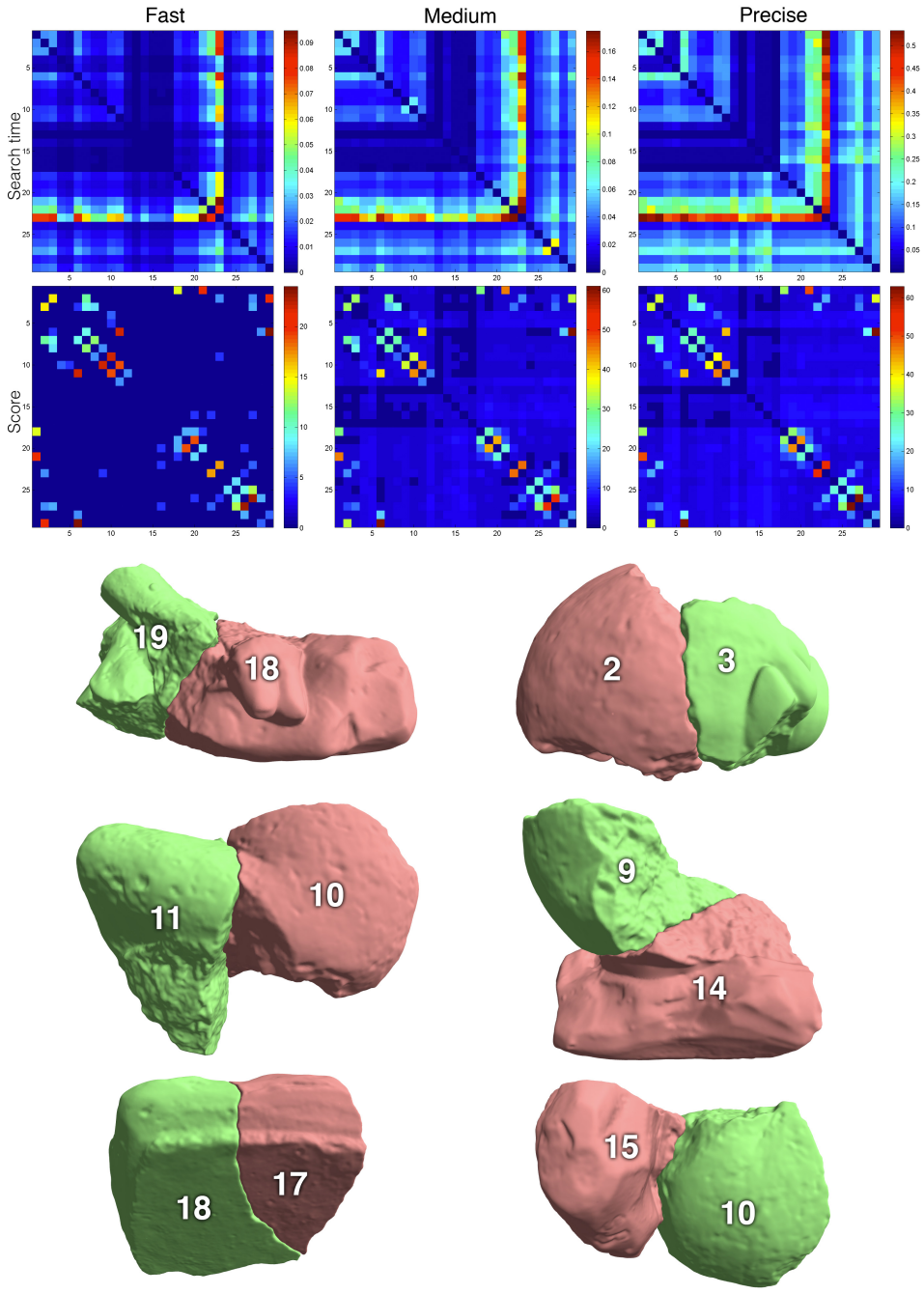

Figure 6.26: *One-to-One search results for the gargoyle dataset. Top row shows the search times (in seconds) for each pair of fragments, whilst bottom row shows the score of the final alignment found. Bottom images show some of the best correspondences found, together with the fragment numbers.*

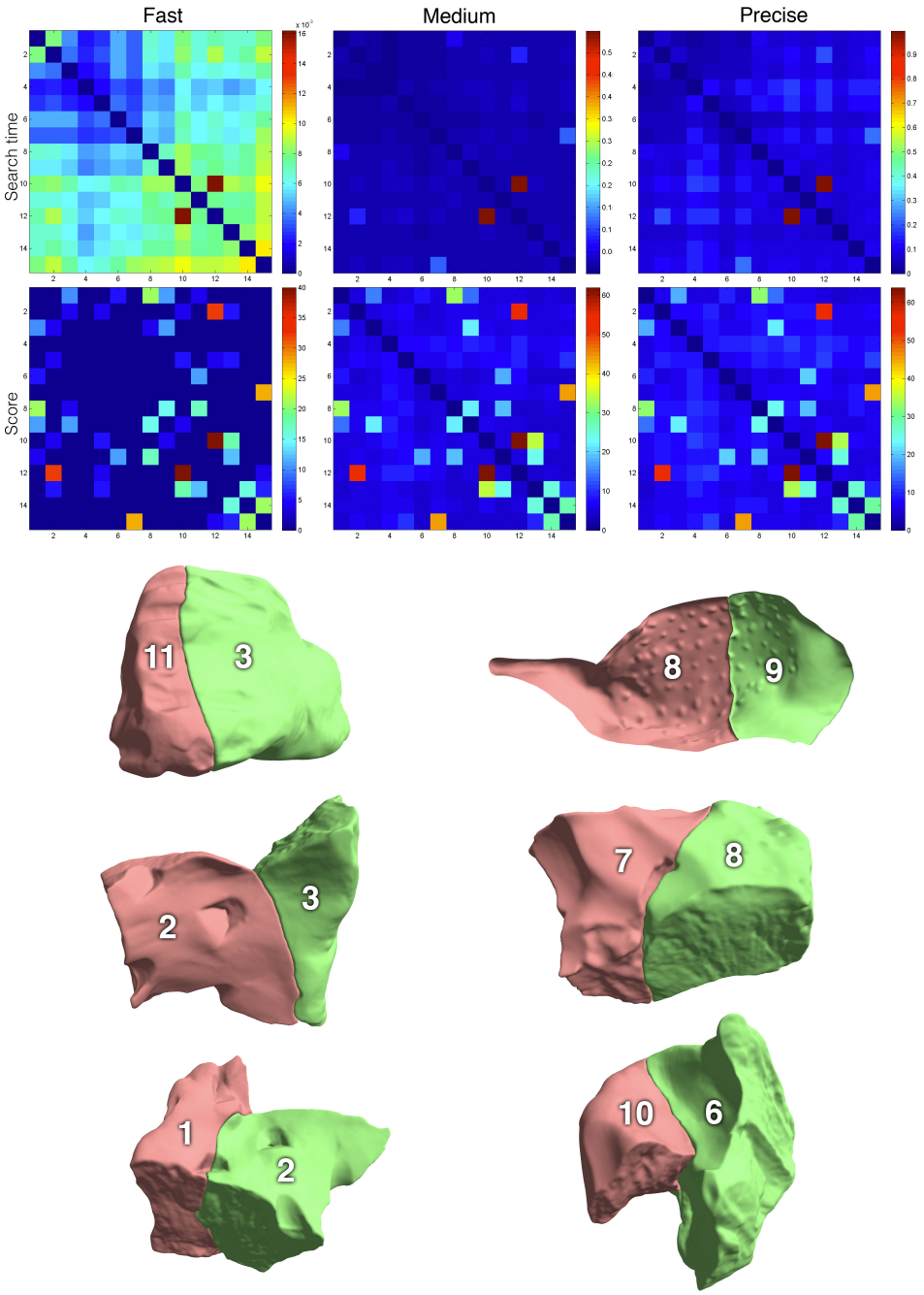

Figure 6.27: *One-to-One search results for the sculpture dataset. Top row shows the search times (in seconds) for each pair of fragments, whilst bottom row shows the score of the final alignment found. Bottom images show some of the best correspondences found, together with the fragment numbers.*

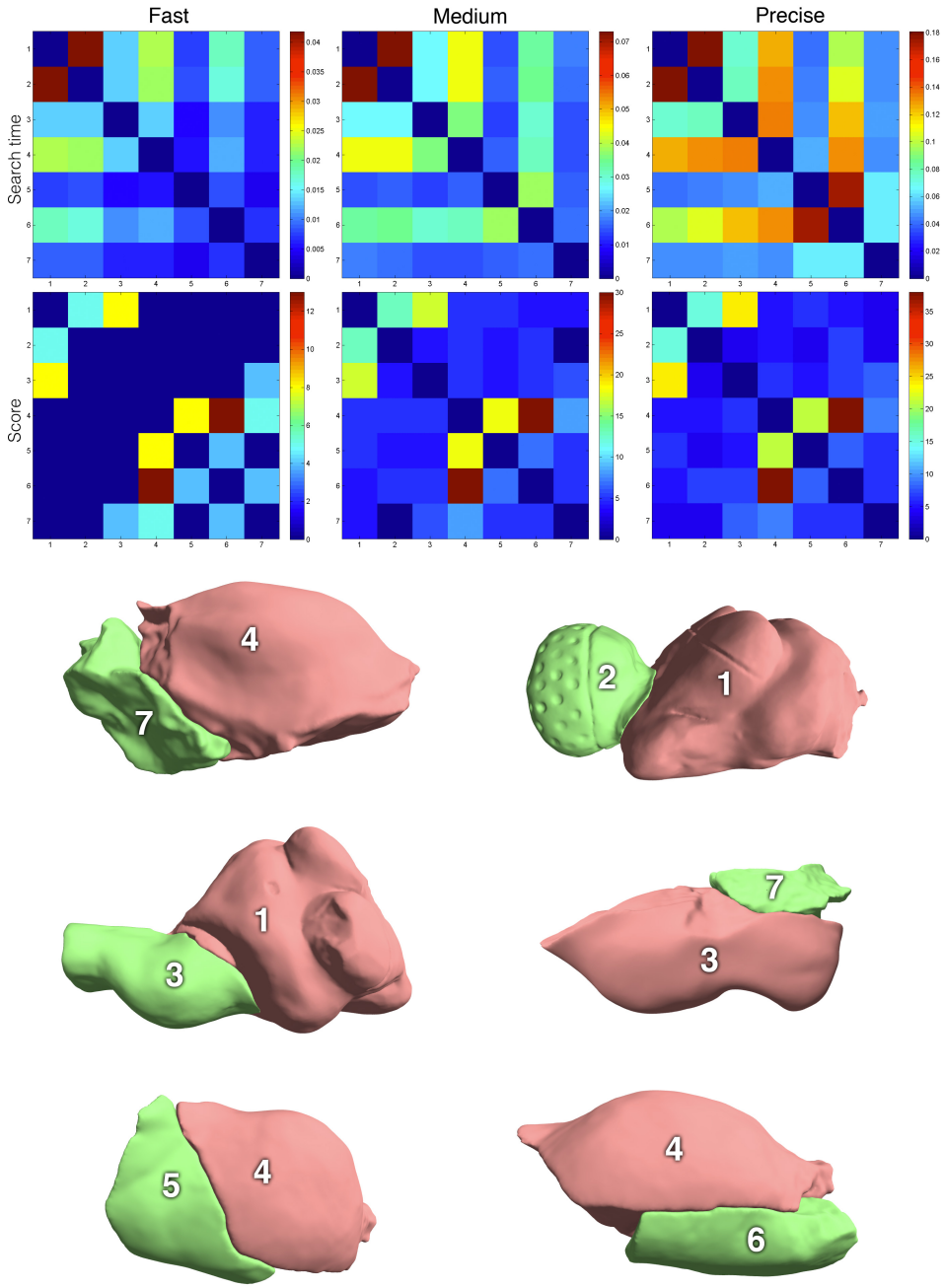

Figure 6.28: *One-to-One search results for the venus dataset. Top row shows the search times (in seconds) for each pair of fragments, whilst bottom row shows the score of the final alignment found. Bottom images show some of the best correspondences found, together with the fragment numbers.*

6.5.3 Many-to-many search

In order to evaluate the global reconstruction algorithm, all the previous datasets have been evaluated using the three search presets commented before. Results can be found in Figure 6.29 and in Table 6.8.

Notice in Figure 6.29 how, for the brick, cake and venus datasets, global reconstruction results are apparently identical using any parameter preset: the final assembly is complete and correct. However, given that the more precise the search is performed, the more matching keypoints are found, final alignments are expected to be more exact with higher search parameters (more correspondences to compute the transformation matrices).

On the other hand, notice how the gargoyle dataset returns two separate clusters using the $fast$ preset, since no valid correspondences are found between the head and the body of the object. Also, correspondences between the right eyebrow and the left foot are not found with these settings. Similarly, the sculpture dataset has a misplaced fragment using the $fast$ preset in the lower left corner.

Execution times have proven to be extremely fast in all the cases, with a maximum searching time of 14.34 seconds for the gargoyle dataset, and a global average time of 2.11 seconds. Comparing these results with the ones published in [71] shows a performance boost of many orders of magnitude: the only timing result provided was for the brick dataset, which took around 15 seconds (whilst it only took 0.36 seconds with the proposed $medium$ search preset). Extrapolating this relationship, the gargoyle dataset would take them 2 minutes and 15 seconds, which is considerably more than the 3.25 seconds achieved with the proposed technique.

It is also important to remark that the search technique presented in [71] uses the whole point cloud to compute the final reconstruction, whilst the one presented here uses only a reduced set of keypoints. This way, memory requirements are very reduced, being the average file size for each fragment 209 KB. This fact allows processing very large datasets without stressing the computer's memory.

	Brick	Cake	Gargoyle	Sculpture	Venus
Fast	0,072668	0,384117	1,179188	0,126020	0,056107
Medium	0,363549	1,238172	3,250621	1,109079	0,131010
Precise	0,810381	4,504372	14,345377	3,080175	0,431054

Table 6.8: *Many-to-many searching times (in seconds)*

From Table 6.8 is important to remark that one-to-one searching times presented in the previous sub-section are measured by looking for the best alignment between two fragments in a single thread. However, as explained before, the many-to-many search strategy exploits the hardware architecture of modern computers to parallelize these individual tasks. This explains the extra acceleration achieved in the global reconstruction time.

According to the results presented in this section, the $medium$ preset seems to provide a good balance between correction and performance, providing always correct results, while keeping searching times very reduced: 1.22 seconds on average.

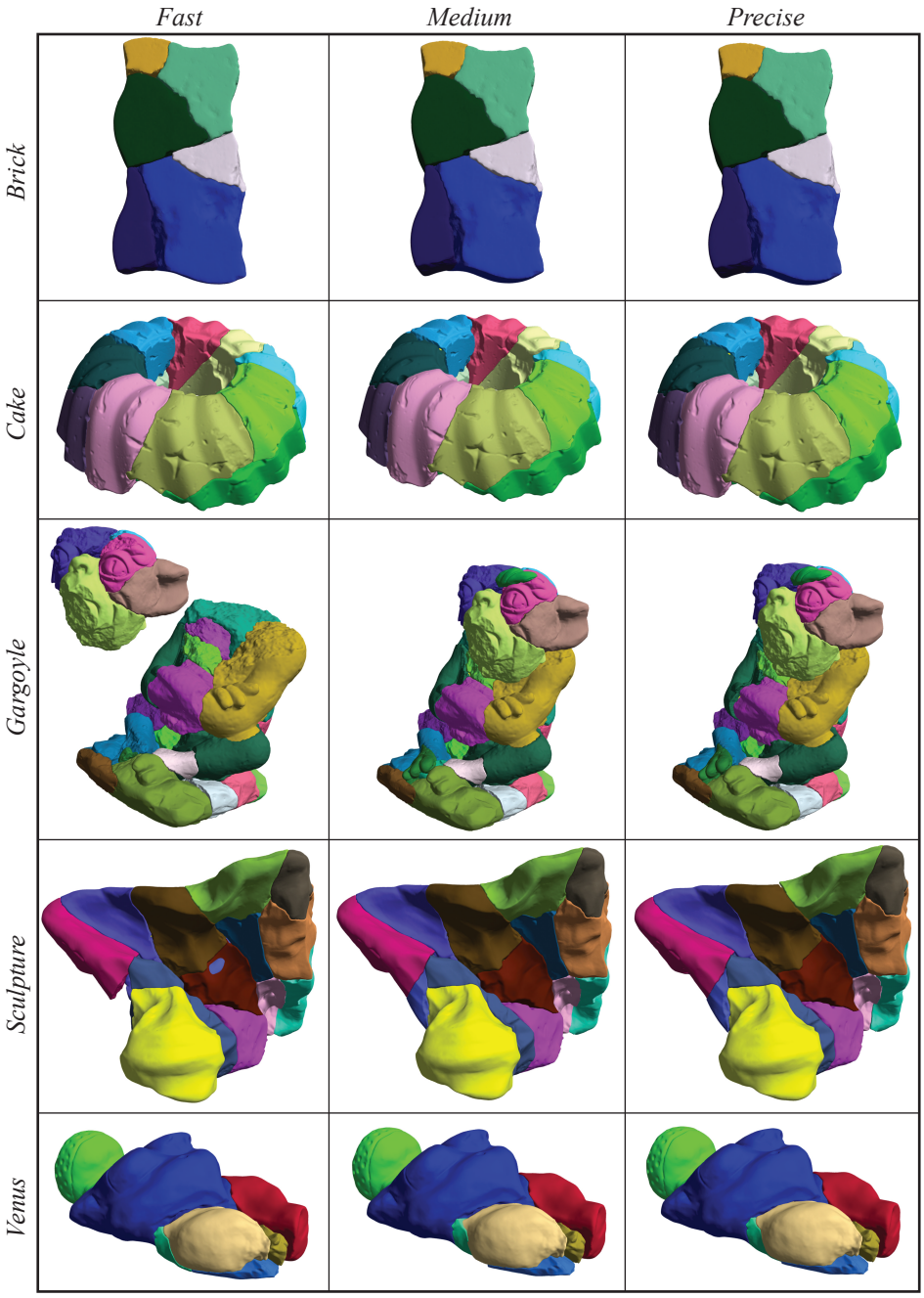

Figure 6.29: *Many-to-many search results for all the datasets using the three proposed presets.*

6.6 Conclusions and Future Work

In this chapter, an automatic technique for re-assembling archaeological artifacts from fragments has been presented. Providing a general solution for 6 degrees of freedom problems, an efficient execution cycle has been introduced in order to work with an alternative characterization of fragments.

A pre-processing stage has been proposed, where the size of the problem is considerably reduced: singular keypoints in the original point cloud are selected based on a multi-scale feature extraction process driven by the saliency of each point, together with a roughness estimation. Computing a modified version of the PFH descriptor, the local neighborhood of each keypoint is described in a compact histogram. All this geometric and topological information is stored in a very small file, that contains all the required data to perform the final reconstruction.

Using exclusively the selected keypoints and their associated descriptors, a very fast one-to-one search algorithm is executed for each possible pair of fragments. This process uses a three-level hierarchical search strategy driven by the local similarity between keypoints, and applying a set of geometric consistence tests for intermediate results.

Finally, a graph-based global registration algorithm uses all the individual matches to provide the final reconstruction of the artifact by creating clusters of matching fragments, appending new potential matches and joining individual clusters into bigger structures. This global registration algorithm exploits modern computer's hardware architecture, by performing simultaneous parallel one-to-one searches.

Achieved results have proven the technique to be very fast and accurate: it has been shown how, in a few seconds (or even less), complex reconstruction problems with millions of 3D acquired points have been correctly solved. It has also been evidenced that the proposed technique is scalable, and can be applied to very large datasets: fragment data is very reduced and individual searches are independent, parallelizable and very fast to execute.

Future works on this technique will focus on exploiting the extra information returned by the one-to-one matcher in order to perform a global registration of fragments considering loop-closures. Also, to improve robustness, a penetration detection algorithm will be implemented to enrich the detection of incorrect alignments between fragments/clusters. This last aspect is indirectly evaluated by the one-to-one geometric consistence constraints proposed, but may provide very useful information if considered in the many-to-many search strategy, allowing to detect incorrect results.

CHAPTER 7

Applications to Self-localization problems

This chapter applies surface registration techniques to a different field: self-localization problems in known environments. The two cases proposed face the same basic problem (inference of the observer's location) using two different approaches: a sparse one, based on a feature extraction process (like the proposed 6DOF technique) and a dense one, driven by the exploration of the solution space (like the proposed 3DOF technique).

7.1 Fast Indoor Localization for Mobile Robots

This section focuses on applying geometric registration techniques for mobile robot self-localization in structured indoor environments.

Self-localization techniques based on robot's observations can be understood as a specific application of surface registration techniques: by calculating the best correspondence between a given observation of the robot and a previous one (or a well-known map of the environment) the relative position of the observer can be obtained by simple triangulation

The final goal is to implement a fast algorithm that can be executed, in real-time, in a mobile robot with very limited computational resources. To do so, the proposed method takes advantage of the available structural information of a ground-truth map to perform a geometrical matching with the measurements collected by a laser rangefinder. In contrast to other global self-localization algorithms like Monte Carlo or SLAM, the proposed algorithm provides a linear cost with respect the number of measures collected, making it suitable for resource-constrained embedded systems.

Given the limited computational resources of the platform targeted, a dense correspon-

dence approach cannot be considered. Instead of this, a keypoint extraction algorithm has to be implemented that simplifies the observed data into a set of singular points that will be matched against the map. To efficiently do so, the structured nature of indoors environment is exploited. This structure is normally introduced by the presence of straight walls and, consequently, this work focuses its effort on developing a fast line extraction algorithm in order to detect them. However, the proposed algorithm can be easily extended to recognize any other type of geometric primitives using the same basic idea.

7.1.1 Overview

The self-localization technique proposed in this section runs in three different stages: calibration, segmentation and localization. Calibration stage is executed before the robot can move autonomously, and its results can be used in subsequent stages without re-calibrating the sensor. The goal of this stage is to calculate the error function $e(d)$ associated to the distance measurements d_i returned by the sensor.

Once the calibration has been performed, the execution cycle of the robot starts by reading the measures obtained by the laser, and converting the set of local 2D points obtained to a much simpler representation. This task is performed in the segmentation stage, where points s_i are inferred into lines l_i, filtered, and intersected between themselves, in order to calculate a set of corners c_i.

The most defined corner c is then passed to the localization stage, where it is aligned with all the similar corners in the known map m_i. Each alignment produces one possible location for the robot $(\theta_{robot}, x_{robot}, y_{robot})$, which is evaluated using a cost function based on squared errors. The alignment with the smallest error is then returned as the location of the robot in the known map. Fig. 7.1 shows the complete execution cycle of the proposed technique that is explained in this section.

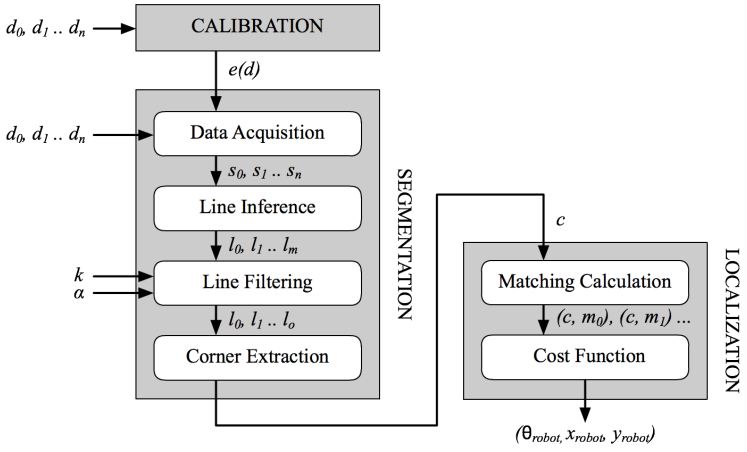

Figure 7.1: *Execution cycle of the proposed technique.*

Main contributions of this subsection are (1) a fast line extraction algorithm that can be executed in a resource-limited system providing a linear cost with respect to the sensor

resolution and (2) an indoor localization pipeline that exploits the structured nature of indoor environments to address the problems of global localization and kidnapped solving.

7.1.2 Calibration

Calibration stage deals with the estimation of the sensor's measurement error. The result of this stage is critical, since line inference algorithm uses this value to establish if a given sample belongs to the line defined by the previous ones or if it belongs to a new line.

The measurement error $e(d)$ is a function that indicates, for each distance d_i, the uncertainty of the returned value. Its value can be a constant, or a function that depends on the measured distance and/or the orientation of the laser beam in local coordinates.

To calibrate the sensor, the robot is placed in front of a straight wall and, using least squares, the equation of the line that best fits the returned measurements, is calculated. The error associated to each distance is calculated as the geometric distance between the point defined by the distance measurement returned by the sensor, and the point defined by the intersection between its projection line and the optimal line calculated using all samples. Fig. 7.2 illustrates this concept.

During the empirical tests with the laser, the error remains constant, no matter the measured distance or the projection angle. This way, a error function as a constant value for all measures has been set, being $e(d) = 2cm$, which corresponds to the maximum error obtained in the calibration stage.

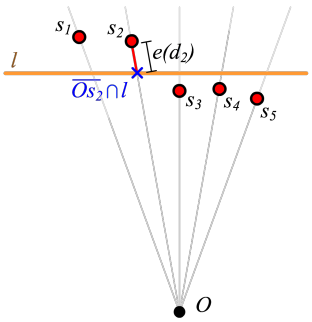

Figure 7.2: *Calculation of the error associated to each measurement. In orange is shown the line calculated using least squares.*

7.1.3 Segmentation

The goal of the segmentation stage is to reduce input data to a much simpler representation that allows comparing it with the known map in an efficient and robust way. For this, noise introduced by the sensor and real world objects imperfections has to be filtered using the error estimation calculated in the calibration stage. To speed-up localization calculations, the input set of distance measurements returned by the sensor is inferred into a set of straight lines. These lines are intersected with their neighbors to compute a set of corners. The most accurate estimated corner will be used in the localization stage to evaluate the position of the robot in the known map.

7.1.3.1 Data acquisition

A sensor is defined as $S(f, n, e(d))$ where f represents the FOV (Field Of View), n is the number of distance samples returned and $e(d)$ is the error function that estimates the measurement error associated to each distance d_i.

Values of f and n are specified by the sensor manufacturer, whilst value of $e(d)$ is calculated in the calibration stage.

The set of distance measurements obtained by the sensor $D = (d_1, d_2 \dots d_n)$ is an ordered array of decimal values. Projection angle φ_i of distance d_i can be calculated as $\varphi_i = f * \left(\frac{i}{n} - \frac{1}{2}\right)$, and its local coordinates (u_i, v_i) as follows:

$$u_i = sin(\varphi_i) * d_i \tag{7.1}$$

$$v_i = cos(\varphi_i) * d_i \tag{7.2}$$

Given a set of distances D, a set of samples $S = (s_1, s_2 \dots s_n)$ are calculated in this stage, where each sample s_i is defined by three points in the sensor's local coordinates:

$$s_i = (sin(\varphi_i) * d_i, cos((\varphi_i) * d_i) \tag{7.3}$$

$$\lfloor s_i \rfloor = (sin(\varphi_i) * (d_i - |e(d_i)|), cos(\varphi_i) * (d_i - |e(d_i)|)) \tag{7.4}$$

$$\lceil s_i \rceil = (sin(\varphi_i) * (d_i + |e(d_i)|), cos(\varphi_i) * (d_i + |e(d_i)|)) \tag{7.5}$$

where $e(d_i)$ is the associated measurement error for each sample.

This way, each sample is characterized by its measured distance to the sensor, s_i, its minimum possible distance, $\lfloor s_i \rfloor$, and its maximum possible distance, $\lceil s_i \rceil$, according to the error estimation performed in the calibration stage.

These samples are used in next stages to perform several calculations. If the required frequency for the proposed technique cannot be ensured, due to computational restrictions, an interesting simplification can be performed at this point: each calculated sample can represent a set of distance measures. Thus, by averaging distances, the amount of samples to process in further stages can be reduced, and some random noise can be filtered.

7.1.3.2 Line inference

The goal of this stage is to calculate a set of straight lines $L = (l_1, l_2 \dots l_m)$ that best fits the set of samples calculated during the acquisition stage, considering the measurement error. The main difficulty associated to these calculations is to establish if a given sample si belongs to the line obtained considering previous samples s_{i-j}, or if it belongs to a new line. To solve this problem, an estimator called *line visibility* is associated to each line, l_k, defined by two scalar values $V_{l_k} = (d_{min}, d_{max})$. This estimator specifies the range of projective distances in which sample s_i will be considered as part of the line defined by previous samples. This way, the following can be established:

$$s_i \in l_k \leftrightarrow [\lfloor s_i \rfloor \dots \lceil s_i \rceil] \cap [d_{min} \dots d_{max}] \neq \emptyset \tag{7.6}$$

being $[d_{min} \dots d_{max}]$ the range of distances specified by V_{l_k}, and l_k the line defined by previous samples $(s_{i-1}, s_{i-2}, \dots)$. Values of $[\lfloor s_i \rfloor \dots \lceil s_i \rceil]$ represent the uncertainty associated to the position of s_i, considering its associated measurement error. Fig. 7.3 illustrates this concept.

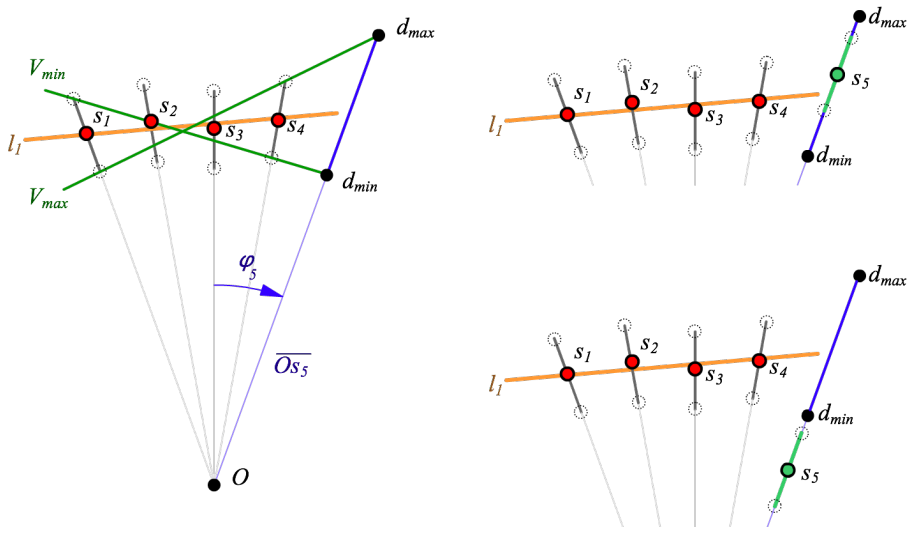

Figure 7.3: *Use of the line visibility concept. Red circles correspond to s_i values, dashed circles correspond to $\lfloor s_i \rfloor$ and $\lceil s_i \rceil$ values. Dark gray lines represent the uncertainty associated to each sample $[\lfloor s_i \rfloor \dots \lceil s_i \rceil]$. (Left) Graphical representation of V_{l_1} considering all illustrated samples. (Right-up) A new sample s_5, represented in green, that satisfies Eq.* (7.6) *and, thus, belongs to the line l_1. (Right-down) A new sample s_5 that does not satisfy Eq.* (7.6)*, so it does not belong to the line l_1.*

Given a set of consecutive samples $S = (s_{i-j} \dots s_{i-2}, s_{i-1})$, to compute values of d_{min} and d_{max} for the next sample s_i, it is necessary to find the minimum/maximum slope lines, V_{min} and V_{max} respectively, that intersects all ranges $[\lfloor s_k \rfloor \dots \lceil s_k \rceil], s_k \in S$.

The intersection points between these lines and the line $\overline{Os_i}$, defined by φ_i and marked in blue in Fig. 7.3 (left), correspond to d_{min} and d_{max}, being $d_{min} = V_{min} \cap \overline{Os_i}$ and $d_{max} = V_{max} \cap \overline{Os_i}$. This way, if a sample s_i fails Eq. (7.6), it means that it cannot exist one line that intersects all ranges $[\lfloor s_k \rfloor \dots \lceil s_k \rceil], s_k \in (S \cup s_i)$ and, consequently, the sample must belong to a new line.

To infer L, all samples are iterated in order, creating new lines when it is necessary. Algorithm 7 illustrates this process.

If a sample si passes Eq. (7.6), defined by the line that considers previous samples $(s_{i-j} \dots s_{i-2}, s_{i-1})$, it is then included in the line, and values of d_{min} and d_{max} are updated considering the new sample.

If s_i fails the test established in Eq. (7.6), a new line is created that only includes s_i, and the equation of the previous line is calculated using least squares for fitting, according to the projective distance measured for each included sample (s_i).

This way, by using $\lfloor s_i \rfloor$ and $\lceil s_i \rceil$, the line visibility estimator that splits the original sample set into smaller clusters of aligned samples is calculated. Then, by using s_i distances, the line that best fits all samples inside each cluster is calculated too and this set of lines is the result of this stage.

Algorithm 7 proposed clustering algorithm to infer the set of lines L that best fits the set of samples S.

```
1: L ← ∅;
2: l ← new Line(∅);
3: for all Sample s_i ∈ S do
4:    d_min ← V_{l_min};
5:    d_max ← V_{l_max};
6:    if [⌊s_i⌋ ... ⌈s_i⌉] ∩ [d_min ... d_max] ≠ ∅ then
7:       l.AddSample(s_i);
8:       l.RecalculateV_l();
9:    else
10:      l.LeastSquareFit();
11:      L.Add(l);
12:      l ← new Line(∅);
13:   end if
14: end for
15: l.LeastSquareFit();
16: L.Add(l);
17: return L;
```

One of the advantages of this technique is that it first produces the set of aligned samples, and then, it calculates the equation of the optimal line that best fit them. This is important, since Ordinary Least Squares (or Linear Least Squares), minimizes the sum of squared vertical distances between observations and the responses predicted by linear approximation. Thus, the more horizontal samples are distributed, the more accuracy. In order to optimize the results of this technique, the set of samples of a cluster can be rotated, then approximated, and then the optimal line is rotated back again.

To calculate values of d_{min} and d_{max}, minimum and maximum slope lines that intersect all $[\lfloor s_i \rfloor \dots \lceil s_i \rceil]$ ranges have to be calculated. Given that the way of calculating these lines is symmetrical, the calculations are focused on finding out the value of V_{max}. The first thing to consider when updating V_{max} consists in differentiating two different cases, according to the new sample (s_j) position: (A) if $[\lfloor s_j \rfloor \dots \lceil s_j \rceil]$ intersects V_{max} no update is necessary, since V_{max} is already the maximum slope line. (B) If not ($\lceil s_j \rceil < d_{max}$), V_{max} is no longer the maximum valid slope line, and has to be recalculated. In this case, it is important to notice that the new maximum slope line is defined by two points: $\lfloor s_p \rfloor$ and $\lceil s_q \rceil$, which are the most restrictive projective distances. Fig. 7.4 illustrates this concept.

Calculating the value of $\lceil s_q \rceil$ is immediate ($\lceil s_q \rceil = \lceil s_j \rceil$), since $\lceil s_j \rceil < d_{max}$. To calculate the value of $\lfloor s_p \rfloor$, it is necessary iterate through each sample included in the current line, looking for the point $\lfloor s_j \rfloor$ that minimizes the angle between the line $\overline{\lceil s_j \rceil \lfloor s_i \rfloor}$, and the projection line defined by $\overline{Os_j}$.

Considering that the addition of samples to the line is an iterative process, an interesting optimization can be performed: once a sample has been identified as the most restrictive one, none of the previous samples needs to be considered in subsequent iterations. This way, by storing the index of the most restrictive sample, a lot of unnecessary evaluations can be avoided, ensuring a correct result. Algorithm 8 illustrates this process.

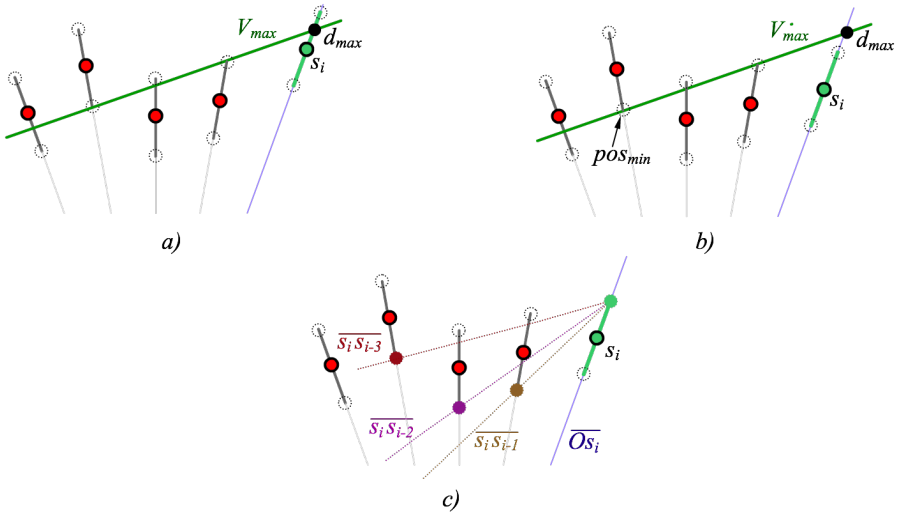

Figure 7.4: *Updating of V_{max}: a) update case A, where no computation is required. b) Update case B, where V_{max} has to be recalculated. c) Calculation of the new value for V_{max} . In the proposed example, V_{max} is the line defined by $\overline{\lceil s_i \rceil \lfloor s_{i-3} \rfloor}$.*

In terms of computational complexity, the worst-case scenario for the proposed algorithm consists in a set of samples perfectly aligned: since all of them will pass the visibility test, there will only be one line. When updating V_{min} and V_{max}, all new samples will force to recalculate the lines, and the most restrictive sample will always be the first one, so the final execution cost is expected to be $\mathcal{O}(n^2)$, being n the number of samples. However, the worst-case scenario is extremely unusual to happen, even impossible when using a sensor with FOV values similar or greater than 180º. In the best-case scenario, each sample included in a line will not force to update values of V_{min} and V_{max} so, the execution cost is expected to be $\mathcal{O}(n)$, being n the number of samples.

7.1.3.3 Line filtering

The goal of this stage is to add robustness to the proposed technique, considering new uncertainties not contemplated when calculating the error function in the measures of the sensor $e(d)$. These uncertainties are mainly related to physical imperfections in both, the sensor and the environment.

During the empirical tests, these imperfections have been classified in two cases: (a) interferences in the laser range finder due to the topology and material of the measured area and, (b) imperfections in the walls that lead to split them in several separated lines.

Interferences in the scanner happen when the laser beams intersect reflective/refractive surfaces. The resulting distance measures are unpredictable, and add a lot of noise to the input data. Also, this precision decreases significantly when scanning a sharp corner.

Imperfections in the walls make theoretically planar surfaces to be curved. This affects the line inference technique by making that, eventually, one sample of the same wall fails the visibility test defined by the previous ones. The result is that a wall is then characterized

Algorithm 8 pseudo-code to update value of V_{max}. Consider that pos_{min} is a global variable that keeps its value after each invocation of the function.
CalculateV_{max}(s_j : Sample, V_{max} : Line)

```
1:  d_max ← V_max ∩ Os_j;
2:  if ⌈s_j⌉ < d_max then
3:     angle_min ← ∞;
4:     for pos ∈ [pos_min ... j − 1] do
5:        if Angle(⌈s_j⌉⌊s_pos⌋, Os_j) < angle_min then
6:           angle_min ← Angle(⌈s_j⌉⌊s_pos⌋, Os_j);
7:           pos_min ← pos;
8:        end if
9:     end for
10:    V_max ← ⌈s_j⌉⌊s_pos_min⌋;
11: end if
12: return V_max;
```

by more than one line.

By detecting and properly filtering these imperfections, the quality of results improves significantly. Fig. 7.5 illustrates these two kinds of interferences using real data, some examples of them, and the result of applying the proposed filtering.

To filter interferences in the scanner, the fact that an anomalous measure seriously penalizes the line visibility estimator it is considered. Thus, a line containing a bad sample has a very limited amount of samples in it (cases A and B in Fig. 7.5), so all lines containing less than k samples are discarded ($|l_i| < k$), and so are the samples within them. The value of k is a parameter specified by the user, and empirical tests have proven that values in the range $[4 \dots 6]$ provide good results.

After removing the lines created by interferences in the scanner, imperfections in the walls are filtered using a second parameter, α, that indicates the maximum tolerance for wall deviations. All consecutive lines with an angular deviation smaller than α, are combined and recalculated considering all samples.

The result of this stage is a new set of lines $L' = (l'_1, l'_2 \dots l'_o)$ that improves the segmentation obtained in $L = (l_1, l_2 \dots l_m)$, and that verifies that $o \leq m$.

7.1.3.4 Corner extraction

To efficiently compare the resulting set of lines L' with the known map, a global registration between the two datasets is necessary. One of the most common techniques to perform this operation is to simplify the representation of both datasets into simpler ones. This way, alignment calculations are considerably accelerated.

Instead of working directly with lines, the use of *corners* is proposed to perform this operation. A corner is defined by the intersection between two consecutive lines, and characterized by its inner angle (β), its position (x, y) and its orientation (θ).

To obtain the set of corners $C = (c_1, c_2 \dots c_p)$ defined by $L' = (l'_1, l'_2 \dots l'_o)$, all lines are intersected between them, storing only the intersections that are close enough to the endings of both lines. All calculated corners maybe do not exist in the real world but they

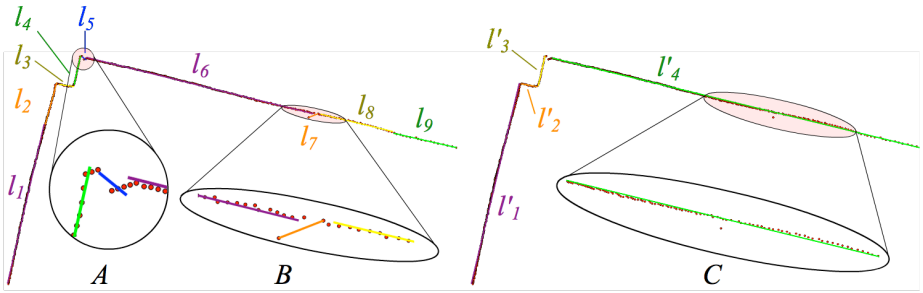

Figure 7.5: *Imperfections in the sensor measures that affect the line inference calculations. (Left) Result of the previous stage without considering these imperfections. (Right) Result after filtering. Case A shows the error of the sensor when scanning a sharp corner. Case B the consequences of reflections in the line inference algorithm. Case C shows a wall that deviates in both directions.*

can be very helpful in the location algorithm (e.g. this allows to avoid interferences caused by small pillars like the one shown in Fig. 7.5). Given the interest in accelerating global registration, instead of using all corners, only the best characterized is used: c. To find this one among all C, the quality of a corner is defined as $|c_i| = min(|l_1|, |l_2|)$, being l_1 and l_2 the lines that generate it, and $|l_1|$and $|l_2|$ the amount of samples of each line, respectively.

$$c = \quad max(|c_i|), c_i \in C \tag{7.7}$$

$$|c_i| = \quad min(|l_1|, |l_2|), l_1 \in c_i, l_2 \in c_i \tag{7.8}$$

7.1.4 Localization

The localization stage takes, as input data, the resulting corner obtained in the segmentation stage and a pre-calculated map, where all the corners have been identified. By aligning these corners with the one obtained in the previous stage, all possible locations for the robot are inferred. For each location, a measure of correctness is calculated, so they can be sorted according to the quality of achieved results. The most correct result is expected to be the robot localization in the real world.

7.1.4.1 Matching calculation

This stage deals with the calculation of all possible locations of the robot in the real world. To do so, all corners in the known map with a similar inner angle to the one obtained in the segmentation stage are aligned, producing each alignment a localization for the robot.

In order to accelerate calculations, the known map has been pre-processed so all corners have been obtained and sorted accord- ing to their inner angle. This way, finding similar angles computational cost is reduced to $\mathcal{O}(log_2(n))$, being n the total number of corners in the map. In case the map is too big and, to simplify this search, once the location of the robot has been calculated only corners around the last known position have to be checked.

Given an alignment between corners, the robot orientation (θ_{robot}) is calculated as follows:

$$\theta_{robot} = \frac{\pi}{2} + \theta_m - \theta_c - \frac{(\beta_m - \beta_c) * |l_{1_c}|}{|l_{1_c}| + |l_{2_c}|} \tag{7.9}$$

where θ_m is the orientation of the selected corner in the map (in global coordinates), θ_c is the orientation of the corner calculated in the segmentation stage (in local coordinates), β_m is the inner angle of the selected corner in the map, β is the inner angle of the corner calculated in the segmentation stage, and $|l_{1_c}|$, $|l_{2_c}|$ are the amount of samples contained in the lines that define the corner calculated in the previous stage.

By calculating θ_{robot} as proposed, the difference between inner angles is compensated proportionally to the amount of samples of each line that define the corner c. It is expected that, the more samples a line contains, the more accurate the estimation is.

Then, the robot position (x_{robot}, y_{robot}) is inferred by simple triangulation, as shown below:

$$x_{robot} = -x_c * cos(\theta_{robot}) + y_c * sin(\theta_{robot}) + x_m \tag{7.10}$$

$$y_{robot} = -x_c * sin(\theta_{robot}) - y_c * cos(\theta_{robot}) + y_m \tag{7.11}$$

where x_m and y_m are the global coordinates of the intersection point defined by the selected corner in the map, and x_c and y_c are the local coordinates of the intersection point defined by the corner calculated in the segmentation stage. Fig. 7.6 illustrates these concepts.

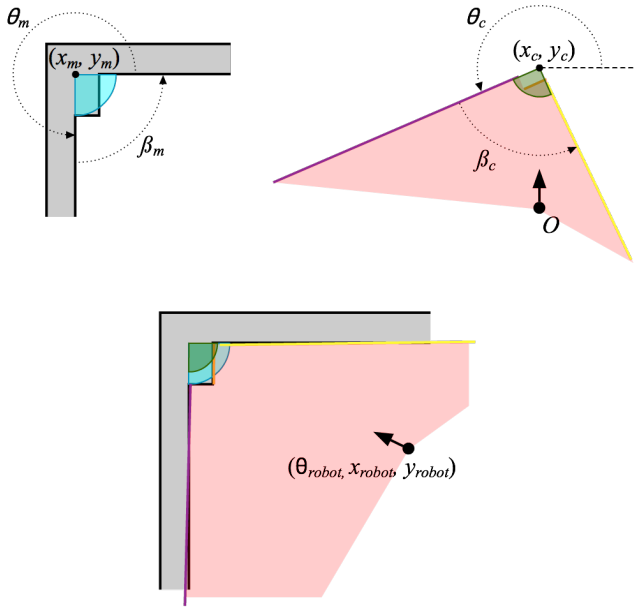

Figure 7.6: *Alignment between corner c and a corner from the known map m. (Top-left) Characterization of m, in global coordinates. (Top-right) Characterization of c in local coordinates. (Bottom) Correspondence between both corners and robot pose calculation.*

7.1.4.2 Cost function

This is the last stage in the proposed technique, and it is responsible to evaluate the quality of each pose calculated in the previous one. The final result of the localization algorithm is a sorted list of possible poses, where the first element is expected to be the real world's robot position. To quantify the quality of each possible pose calculated using the known map, the set of measures $D' = (d'_1, d'_2 \dots d'_n)$ that the sensor should have produced in ideal conditions (no measurement error), given the position and orientation calculated. The total error of a given orientation can then be calculated as:

$$error = \sum_{i=0}^{n} (d_i - d'_i) i^2 \tag{7.12}$$

being d_i the original measures returned by the sensor.

In case the computational cost of these calculations does not satisfy the execution frequency required, as it occurs at some robots with limited computational resources only a subset of equispaced distances can be used.

7.1.5 Results

In order to verify the correct functionality of the proposed algorithm, it has been implemented and analyzed in an experimental platform. This platform is based on a Robotino mobile robot equipped with a laser range finder sensor.

Robotino is a FESTO mobile robot system with three omnidirectional drives. The robot controller consists of an embedded PC running with a Linux operating system installed on a compact flash card. In order to improve its functionality, Robotino is also equipped with several types of sensors, like infrared distance measuring sensors, incremental encoders, anti-collision sensor, analog inductive proximity sensor, camera module, etc.

Robotino is driven by 3 independent, omnidirectional drive units. They are mounted at an angle of 120º to each other. The drive units allow for motion in all directions (forward, backward and sideways) and the robot can be turned on the spot as well. Each of the 3 drive units consists of a DC motor, a gear unit, an all-way roller, a toothed belt and an incremental encoder unit. Actual motor speed can be compared with desired speed by means of the incremental encoder, and can then be regulated with a PID controller. The controller unit consists of 3 components:

- PC 104 processor, compatible with MOPSlcdVE, 300 MHz, and Linux operating system with real-time kernel, SDRAM 128 MB and Ethernet, USB and VGA interfaces.
- Compact flash card that contains the operating system, the functions libraries (C++ API) and the included programs for controlling Robotino.
- Wireless LAN access point.

In order to read the measures for the proposed automatic localization algorithm, the Hokuyo URG-04LX-UG01 has been used. It is a laser sensor for area scanning. The light source of the sensor is an infrared laser, and the scan area is a 240º semicircle with a maximum radius of 5600mm. Pitch angle is 0.36º and sensor outputs the distance measured

at every point (maximum: 684 steps). Principle of distance measurement is based on calculation of the phase difference, due to which it is possible to obtain stable measurements with minimum influence from object's color and reflectance.

The Hokuyo laser range finder is connected via USB to the controller unit of Robotino. For software integration, a library that provides the basic methods of connection and receiving laser measurements has been used. The connection is made using the COM port assigned to the laser, and measures are asynchronously received with a period of 400ms. per event. At each event 684 measures are stored and processed. See Fig. 7.7.

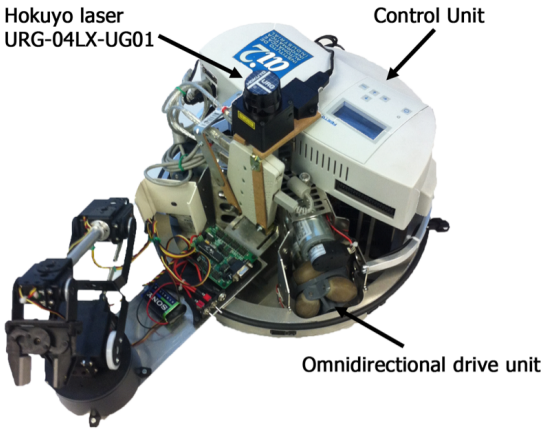

Figure 7.7: *Robotino with Hokuyo laser rangefinder installed.*

Several tests with the Hokuyo with Robotino were performed to calculate the actual computational cost of the algorithm based on the number of samples collected by the laser. Fig. 7.8 shows the relation between the time used by the algorithm according to the number of measures processed by Robotino's processor and the global correction of the technique respect to the sampling ratio, in an outlier-free environment.

As it can be seen, the computation cost is linearly related to the number of samples. In the worst case, using 668 measures from the laser, the algorithm takes 0.058s to run, which is not a problem for Robotino processor and it allows to locate the robot in each iteration of the control loop (control loop period is set at 100ms). Also, global correction of the technique decreases, as the number of samples considered is reduced.

To analyze in depth the computational cost of each phase of the algorithm, Fig. 7.9 shows the cost distribution of all stages for each test performed according to the number of samples. For example, for 668 samples, the total time is 0.058s. The data acquisition stage occupies 0.00464s (8% of the total time), the lines inference algorithm 0.0377s (65%), filtering, corner detection and localization 0.00116s (2%) and the cost function occupies the rest: 0.0145s (25%). However, for 66 samples the total time is 0.0044s, with a 0.00136s for the cost function (31%), 0.00015s for filtering, corner detection and localization (3.5%), 0.00242s for line inference (55%) and 0.00046s for acquisition (10.5%).

Analyzing the complete graph, the line inference stage is the heaviest one, with 65% of time on average. The cost function stage occupies 25% and the acquisition 10%. The filtering, corner detection and localization phases are the fastest, with 2% of the time.

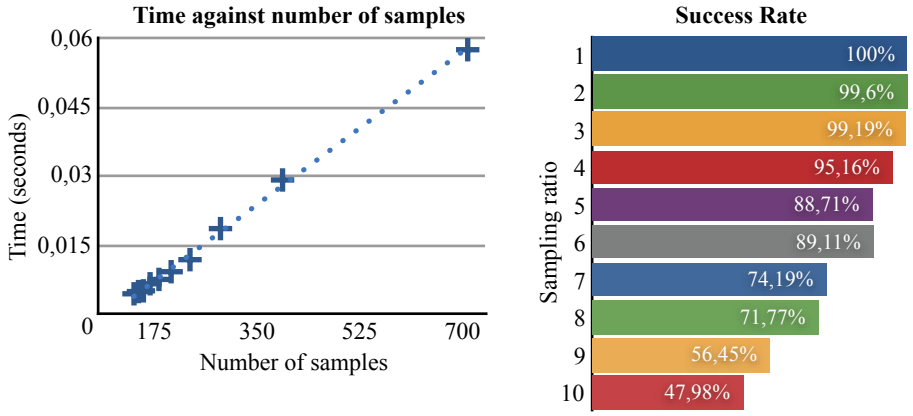

Figure 7.8: *(Left) Several tests to show the relation between the execution cost of the algorithm in Robotino processor and the number of samples collected by the Hokuyo sensor. (Right) Global correction of the achieved results, depending on the sampling ratio used in an outlier-free environment.*

To estimate the accuracy of the proposed technique, an offline registration algorithm that uses as many ICP iterations, has been applied using the same data collected by the robot in actual testing. This algorithm iterates over itself until it converges to the actual position of the robot. By checking that all paired points of the ICP algorithm in the last iteration are correct, it ensures that the achieved solution is the global optima, so it can be used as ground truth to evaluate the accuracy of the technique. Table 7.1 shows the actual accuracy in different scenarios calculated by this technique. These scenarios include an outlier-free environment, an environment with moving people in front of the robot, a scenario with some pieces of furniture, not included in the ground truth map and, finally, a scenario with moving flat outliers.

Test scenario	Success rate	Accuracy
Without outliers	100%	3.16
Moving people	97.67%	7.87
Furniture	88.66%	5.13
Flat outliers	95.99%	5.75

Table 7.1: *Results achieved for the different evaluation scenarios with sampling ratio =1..*

On the other hand, to evaluate the robustness of the proposed algorithm, the impact of outliers in several scenarios where the ground truth map does not fully correspond with the sensor readings, has been tested. Achieved results show that the effect of outliers can be classified in two major groups: (1) the presence of an outlier "breaks" a wall, so the total amount of samples used to infer the line is reduced causing several non-consecutive segments. (2) If the outlier is a flat surface it can be considered as a false wall leading to a misinterpretation of the environment. Fig. 7.10 shows both cases.

Post-processing the extracted lines and merging together aligned segments can attenu-

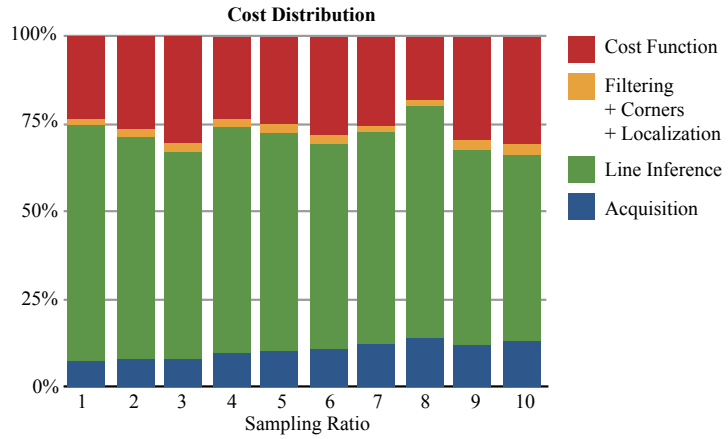

Figure 7.9: *Measures about the cost distribution of the algorithm for different number of samples collected by Hokuyo.*

ate the effects of case 1. This, of course, will require extra computing time. However, empirical tests have shown that results of the proposed technique without this post-processing stage are very robust when using a full sampling ratio as Table 1 shows.

The tests performed to evaluate this case include static and dynamic outliers, with the robot in static and dynamic locations. It is important to notice that, since time is not considered (each iteration previous data is ignored), there is no distinction between moving and static outliers.

For case 2, the effect of misinterpreting a flat outlier as a wall is very dependent on several factors: the portion of the environment visible for the sensor, the size and distance of the outlier respect to the sensor and the ambiguity introduced by the location of the outlier: if the outlier is isolated in the middle of an empty space, or if it makes an angle with other line segment that does not exist in the ground truth map, the risk of misinterpretation is very low.

Table 7.1 shows the results achieved for the different evaluation scenarios, considering all the samples collected by the sensor (sampling ratio=1). Success Rate measures the ro-

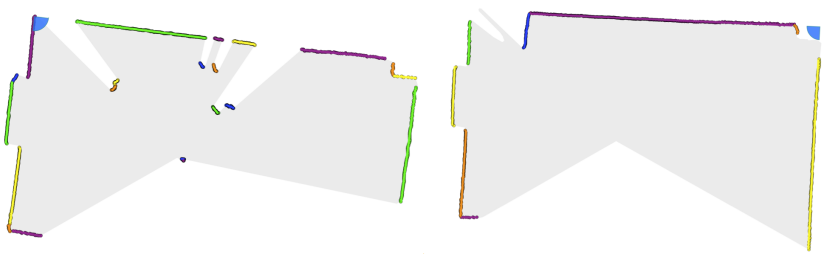

Figure 7.10: *(Left) Three people standing in front of the robot. Notice how the longest wall is split into four different segments. (Right) A box, represented with a blue line, leaning against a wall and misinterpreted as a wall.*

bustness of the technique, since a result is only considered successful if the paired corners are correct. Accuracy is the Euclidean distance between the estimated location, and the correct one. Only correct results are considered for estimating the accuracy.

It must be taken into account that the proposed algorithm is not using any feedback. Each iteration the previous estimated position is ignored and a new one is calculated. Thus, the algorithm does not accumulate any error between iterations and it does not need to know any previous estate. Considering previous estimations might accelerate the execution, evaluating only the part of the map close to the latest known position.

7.1.6 Conclusions and Future Works

A global self-localization fast algorithm for resource-limited systems has been proposed. The algorithm needs a laser or any other environment analysis sensor for auto-locate within a known map. Contrary to other algorithms as Monte Carlo, which uses particle filters hardly implementable in embedded systems, or SLAM algorithm which requires many resources to run, the proposed algorithm allows to self-localize analytically by a geometric matching, achieving a linear cost in relation to the number of measurements taken.

Given the low computational cost, and the fact that the proposed technique does not need any initialization or external feedback, this algorithm can be used cyclically in a separate thread or it can be launched by events, for example, when the positioning error estimated by odometry is too high.

Although the algorithm gives, in most cases, a unique pose as a result, there are some situations where ambiguity between multiple solutions appears. For this cases, future works aim to merge the proposed technique with odometry systems in order to exploit the benefits of time integration to face the global localization problem. Also, extending the line extraction algorithm to other parametrizable primitives is considered as a promising future line of work.

7.2 Indoor Localization for Inspection and Verification

This chapter focuses on applying the surface registration techniques to solve the localization problem in indoor environments. The goal is to use the 3D data provided by a Microsoft Kinect sensor and/or a Velodyne lidar sensor to localize a person inside of a known map, and track changes from previous acquisitions.

This ongoing project is oriented to assist nuclear inspectors from the International Atomic Energy Agency (IAEA) in their verification and change detection tasks inside nuclear facilities. The key idea is that, once the inspector is properly localized, 3D data acquired from the sensor is compared against previous scans of the same facility in order to automatically detect changes in sensitive areas.

The proposed technique assumes to have a *Ground Truth* point cloud of the environment (with no color information), and a continuous feed of partial views provided by the sensor. From these partial views only depth information is used (discarding color information provided by Kinect), so the technique can be applied to a big variety of sensors.

Since the sensor is supposed to be handheld, no specific motion models are used to approximate the observer's pose. The only cinematic constraints considered are non-correlated maximum displacements and rotational speeds.

Also, in order to provide a generic solution for the localization problem, the environment is assumed to be completely unstructured and with the presence of several outliers (topological variations introduced after generating the point cloud used as ground truth), which can be dynamic or static.

7.2.1 Overview

The execution cycle proposed starts by grabbing a new frame from the sensor. If there were no previous known locations, a kidnapped robot solver is executed in order to estimate the observer's position. This process continues until the current observation produces some potential locations. When potentially correct locations have been found, for each one of them, a local update based on a tracking algorithm is executed. After a pose has been updated, its correction is evaluated. In case the new location is classified as correct, it has to wait until the next frame produced by the sensor to be updated again. Otherwise, it has to be decided if it is worth spending computing resources in relocalizing it. In case the test succeeds, a relocalization algorithm is executed from the last known correct position, performing a local search in the surroundings, to attempt to recover it. On the contrary, if the test fails, the location is deleted and no longer tracked. Figure 7.11 shows the complete process.

The key idea of the proposed execution cycle is to have a kidnapped solver that populates the set of active locations using a single observation. Disambiguation of potentially good solutions is integrated in the execution cycle in a generic way. This way, there is no need to distinguish between different specific states, and the resulting technique provides a general solution for the tracking problem.

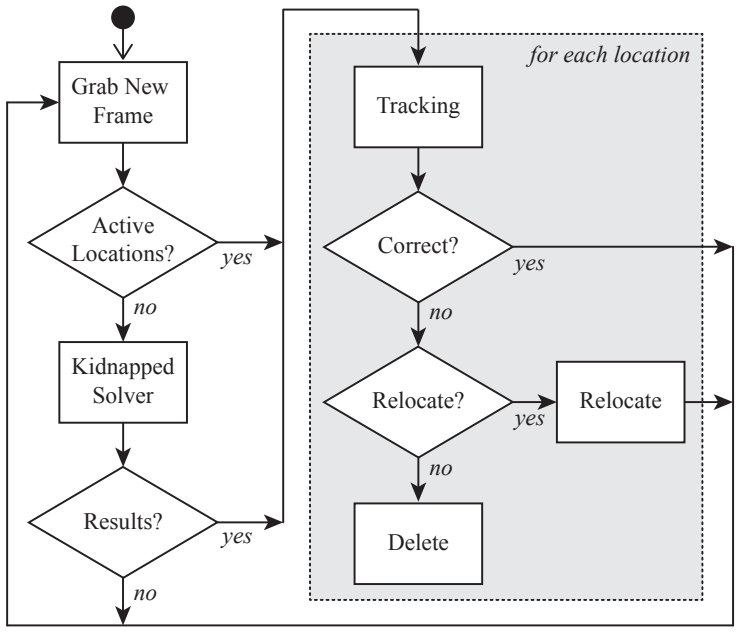

Figure 7.11: *General overview of the proposed execution cycle.*

Major contributions of this section are:

- A complete workflow to perform the continuous tracking of the sensor inside a known environment: from the kidnapped solving to the tracking and relocation, with well defined transitions between states.
- An efficient kidnapped robot solver supported by an optimization stage that efficiently filters ambiguous and incorrect poses.
- An efficient voxel-based representation of the ground truth point cloud that allows accelerating the registration process and achiving realtime results.

7.2.2 The Sensors

7.2.2.1 The Kinect Sensor

The Kinect sensor was released by Microsoft in November 2010. This low-cost range sensor was primarily designed for a natural interaction in a computer game environment. However, the characteristics of the data captured by Kinect and, specially, its reduced prices have attracted the attention of researchers from other fields, including mapping and 3D modeling.

This sensor captures depth and color images simultaneously at a frame rate of up to 30 Hz. The integration of depth and color data results in a colored point cloud that contains about 300000 samples in every frame.

The underlying technology of Kinect consists of an infrared laser emitter, an infrared camera and an RGB camera. The laser source emits a single beam which is split into multiple beams by a diffraction, creating a constant pattern of points projected onto the scene. This pattern is captured by the infrared camera and is correlated against a reference one. The reference pattern is obtained by capturing a plane at a known distance from the sensor, and is stored in the memory of the sensor. When a speckle is projected on an object whose distance to the sensor is smaller or larger than the reference plane, the position of the speckle in the infrared image is shifted in the direction of the baseline between the laser projector and the perspective center of the infrared camera. These shifts are measured for all points by a simple image correlation procedure, which yields a disparity image.

Figure 7.12 shows the resulting range image from the projected pattern.

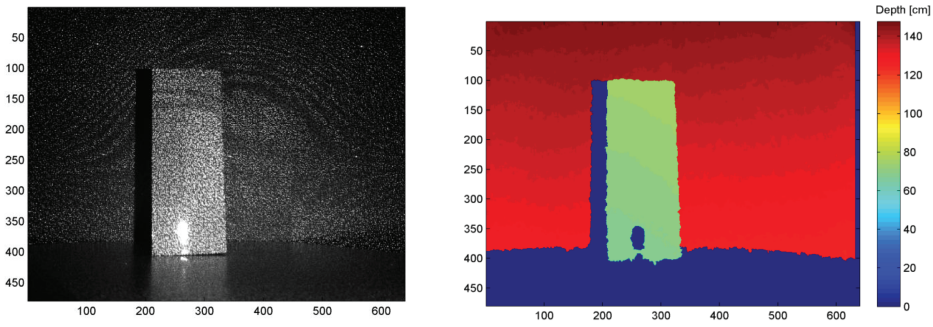

Figure 7.12: *Kinect IR projected pattern (left) and its corresponding range image (right).*

Figure 7.13 illustrates the relation between the distance of an object point k to the sensor relative to a reference plane and the measured disparity d. To express the 3D coordinates of the object points we consider a depth coordinate system with its origin at the perspective center of the infrared camera. The Z axis is orthogonal to the image plane towards the object, the X axis perpendicular to the Z axis in the direction of the baseline b between the infrared camera center and the laser projector, and the Y axis orthogonal to X and Z making a right handed coordinate system.

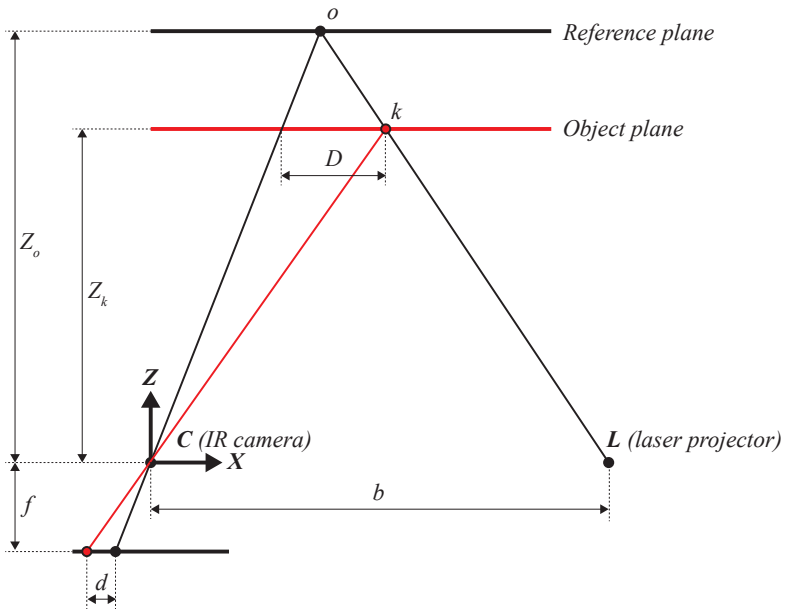

Figure 7.13: *Relation between relative depth and measured disparity.*

Assume that an object is on the reference plane at a distance Z_o to the sensor, and a speckle on the object is captured on the image plane of the infrared camera. If the object is shifted closer to (or further away from) the sensor the location of the speckle on the image plane will be displaced in the X direction. This is measured in image space as disparity d corresponding to a point k in the object space. From the similarity of triangles we have:

$$\frac{D}{b} = \frac{Z_o - Z_k}{Z_o} \tag{7.13}$$

and:

$$\frac{d}{f} = \frac{D}{Z_k} \tag{7.14}$$

where Z_k denotes the distance (depth) of the point k in object space, b is the base length, f is the focal length of the infrared camera, D is the displacement of the point k in object space, and d is the observed disparity in image space. Substituting D from Equation 7.13 into Equation 7.14 and expressing Z_k in terms of the other variables yields:

$$Z_k = \frac{Z_o}{1 + \frac{Z_o}{fb} d} \tag{7.15}$$

Equation 7.15 is the basic mathematical model for the derivation of depth from the observed disparity provided that the constant parameters Z_o, f, and b can be determined by calibration. The Z coordinate of a point together with f defines the imaging scale for that point. The planimetric object coordinates of each point can then be calculated from its image coordinates and the scale:

$$X_k = -\frac{Z_k}{f}(x_k - x_o + \delta x) \tag{7.16}$$

$$Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta y) \tag{7.17}$$

where x_k and y_k are the image coordinates of the point, x_o and y_o are the coordinates of the principal point, and δx and δy are corrections for lens distortion.

Measurement errors and imperfections in the depth data acquired by the Kinect sensor are mostly related to three main reasons: (1) the sensor, (2) the measurement setup and (3) specific properties of the object surface.

Sensor errors refer to inadequate calibration and inaccurate measurement of disparities. Inadequate calibration and/or error in the estimation of the calibration parameters lead to systematic error in the object coordinates of individual points. Such systematic errors can be eliminated by a proper calibration.

Errors caused by the measurement setup are mainly related to the lighting condition and the imaging geometry. The lighting condition influences the correlation and measurement of disparities. In strong light the laser speckles appear in low contrast in the infrared image, which can lead to outliers or gap in the resulting point cloud. The imaging geometry includes the distance to the object and the orientation of the object surface relative to the sensor. The operating range of the sensor is between 0.5 m to 5.0 m according to the specifications. Also, depending on the imaging geometry, parts of the scene may be occluded or shadowed.

The properties of the object surface also impact the measurement of points. This way, smooth and shiny surfaces may appear overexposed in the infrared image, impeding the measurement of disparities, and resulting in gaps in the point cloud.

The resolution of the infrared camera, or more precisely the pixel size of the disparity image, determines the point spacing of the depth data on the XY plane. Since each depth image contains a constant 640×480 pixels, the point density will decrease with increasing distance of the object surface from the sensor. Considering the point density as the number of points per unit area, while the number of points remains constant, the area is proportional to the square distance from the sensor. Thus, the point density on the XY plane is inversely proportional to squared distance from the sensor.

The depth resolution refers to the minimum depth difference that can be measured, and is determined by the number of bits per pixel used to store the disparity measurements. The Kinect disparity measurements are stored as 11-bit integers, where one bit is reserved to mark the pixels for which no disparity is measured, so-called no-data. Thus, a disparity image contains 1,024 levels of disparity. Since depth is inversely proportional to disparity, the resolution of depth is also inversely related to the levels of disparity. This way, it can be said that the depth resolution is also a quadratic function of depth, and decreases with increasing distance from the sensor.

7.2.2.2 The Velodyne Sensor

The Velodyne is a 3D sensor produced by Velodyne Lidar, specially designed for obstacle detection and navigation of autonomous ground vehicles and marine vessels. Its durability, 360º field of view and very high data rate makes it ideal for the most demanding perception applications as well as 3D mobile data collection and mapping applications. There are two different models: HDL64e and HDL32e.

Velodyne HDL64e scans a 360º horizontal field of view (FoV) and 26.8º vertical (FoV) at 5 to 15 Hz, providing over 1.3 million points per second. Data is acquired by means of 64 lasers mounted to specific vertical angles, with the entire unit spinning to cover the horizontal field of view. Velodyne HDL32e was introduced in 2010 with the same horizontal FoV as the HDL64e sensor but with a bigger vertical FoV (40º) covered by 32 lasers and providing about 0.7 million of points per seconds. HDL32e sensor is a rugged sensor, having the Ingress Protection rating of 67 (IP67, the highest possible), thus it can be used in almost all environments. HDL32e is superior to HDL64e regarding vertical field of view, size, design and robustness, but it has a less dense angular resolution in the vertical field of view. Both versions of the sensor provide an operating range up to 100 meters with an accuracy of 2cm.

Respect to the Kinect sensor, Velodyne offers a full 360º horizontal field of view and a larger operating range with higher accuracy. However, acquisition rates are slower (5-15Hz vs 30Hz), vertical resolution is smaller (32/64 planes vs 480), no color information is provided and, the most important difference, since the acquisition of data using the Velodyne sensor is performed continuously as the sensor spins, resulting point clouds suffer from the *warping* effect.

When reading data from the sensor it is necessary to consider that each consecutive distance sample in the same plane has been measured at a different time. This means that a continuous motion model is required in order to project each single sample into its world position with respect to a fixed reference frame. This process is called *unwarping* and, if not performed, resulting point clouds may be highly distorted, as shown in Figure 7.14.

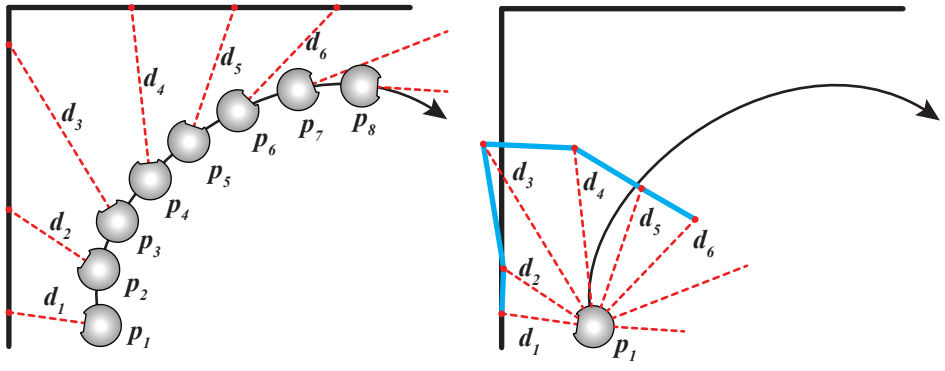

Figure 7.14: *Warping effect when acquiring data with Velodyne. (left) Distances measured in time (d_n) as the sensor moves through different positions (p_n). (right) Resulting reconstruction of the environment when not considering motion but a fixed acquisition position (p_1 in the proposed example).*

7.2.3 Kidnapped Robot Solver

Solving the kidnapped robot problem is a special global localization issue in mobile robotics that consists on localizing the observer inside the known environment with no prior knowledge on his pose.

Classical approaches facing this problem apply probabilistic Bayesian models over a set of random-initialized particles (which represent potential locations) over the entire map. Using a time-spaced set of observations and odometry data, these techniques attempt to converge to the real position by pruning potentially bad particles and populating zones around potentially good ones. This kind of techniques easily solve the problem for bi-dimensional representations of the environment, where only three degrees of freedom exist. However, when considering a three-dimensional environment with six degrees of freedom, the combinatory explosion in the solution space forces to use huge amounts of particles and increases considerably the execution and memory costs. This fact makes the approach unsuitable in lots of cases and, specially, when considering very large environments (like nuclear facilities in the proposed case).

In order to prevent the excessive cost of exhaustively exploring the whole solution space, and considering that no odometry is available for the proposed hand-held application, an alternative solution is suggested: using the known environment point cloud, a synthetic training dataset is built taking advantage of the GPU computing capabilities. This dataset consists on simulating the readings that the sensor would produce from different random poses spread all around the known map, as shown in Figure 7.15.

For each one of these poses, a set of fast-to-compute descriptors are calculated and stored for further searches queried with the real readings of the sensor (produced online). When a search is performed, most similar training observations are selected as potential locations and injected into the set of active poses proposed in the execution cycle of the application (Figure 7.11). Each one of these poses is then tracked using subsequent observations of the sensor. In case a potential pose were wrong, tracking will be lost and the pose will be deleted. Otherwise tracking will continue and, when all but one potential poses are deleted, the problem will be solved.

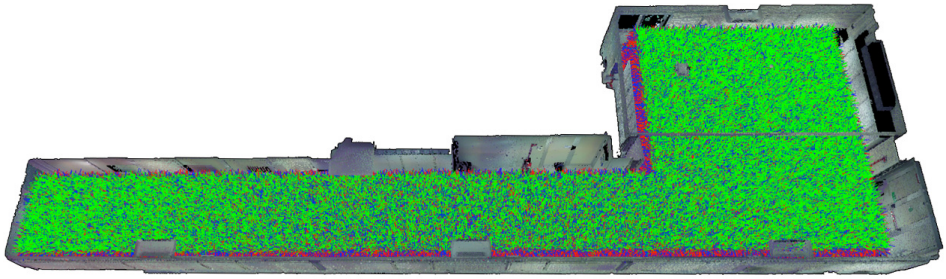

Figure 7.15: *Randomly distributed training poses (500K) inside a lab, where each pose is represented as a three-color coordinate system.*

To compute the synthetic readings of the sensor, a virtual camera that mimics the sensor's field of view and resolution is placed in a random position inside the known map. Using an openGL Frame Buffer Object (FBO), an off-screen depth map is calculated extremely fast. Values stored in this depth map correspond to the undistorted range map that the sensor would produce under ideal working conditions.

Taking as input data each one of these range maps, an easy and fast-to-compute descriptor has to be calculated. It is important to notice that, during the training process, it is not critical the descriptor to be fast to compute, since this process is performed off-line. However, since further searches will be queried with it (extracted using data provided by the sensor) and real-time results are expected, computing costs have to be kept low.

For a Kinect based kidnapped solver algorithm a matrix of 4×3 median distances is proposed, stored as a 12 bin histogram. Advantages of this descriptor is that is very simple to compute, efficiently removes noise and that it is not very sensitive to outliers. Figure 7.16 shows a synthetic range map used for the training dataset and the proposed criteria for computing descriptors.

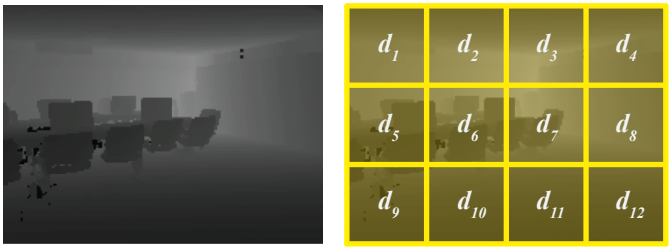

Figure 7.16: *Proposed Kinect descriptor. (left) A synthetic range map computed by the GPU using the ground truth point cloud. (right) Proposed criteria for computing the descriptor: each square is the set of values used to compute each median distance (d_i).*

In order to reduce the size of the training dataset and to speed-up the final technique, a last optimization stage is proposed. The goal of this stage is to remove all poses in the training set that may produce incorrect or ambiguous results.

Incorrect results are produced when the location of the sensor violates its operating

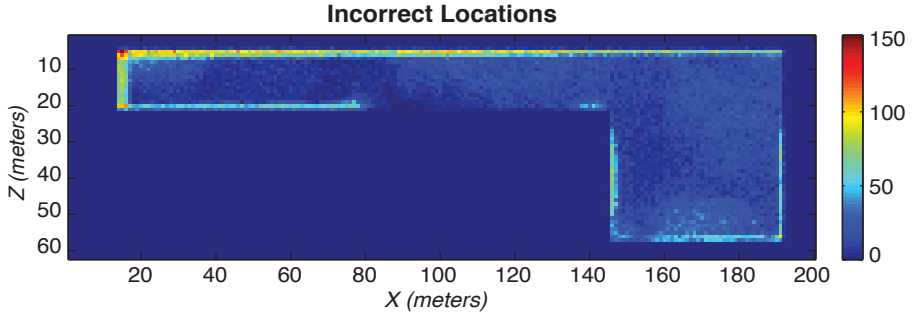

Figure 7.17: *Density of incorrect training poses for the Kinect. Notice how most of them are located by the walls, where minimum distances are not respected.*

distances (e.g. the observer is too close to a wall and the minimum distance is not respected). These cases are easy to detect in the training dataset (resulting depth maps can be clamped to the operating range of the sensor) and in the real data (too much null measures). Removing them from the training dataset makes it smaller and accelerates further searches. Figure 7.17 shows the density of incorrect training poses in the dataset illustrated in Figure 7.15.

Ambiguous results are produced by symmetries on the known map: if the partial view retrieved by the sensor has multiple correspondences in the environment, multiple results will be returned. This is not a wrong result, but may compromise the performance of the final algorithm: each time the sensor provides a new range map the tracking algorithm has to update all the potential locations. If, as a result of the kidnapped robot solver, there are too many of them, the execution cost will increase, leading to a frame-loss situation. To prevent this from happening, ambiguous poses should be detected and the kidnapped solver should only provide results when the current observation is not ambiguous.

Figure 7.18 shows some cases with different decreasing levels of ambiguity. Notice how for the first example, where the sensor is pointing to a wall, lots of possible results are produced. It will make no sense to track all of them. In the second case, when pointing to a corner, the number of possible results is considerably smaller, all being around similar parts of the map. However, in the last case, since the location is singular, only a set of results are returned, all of them around the right position.

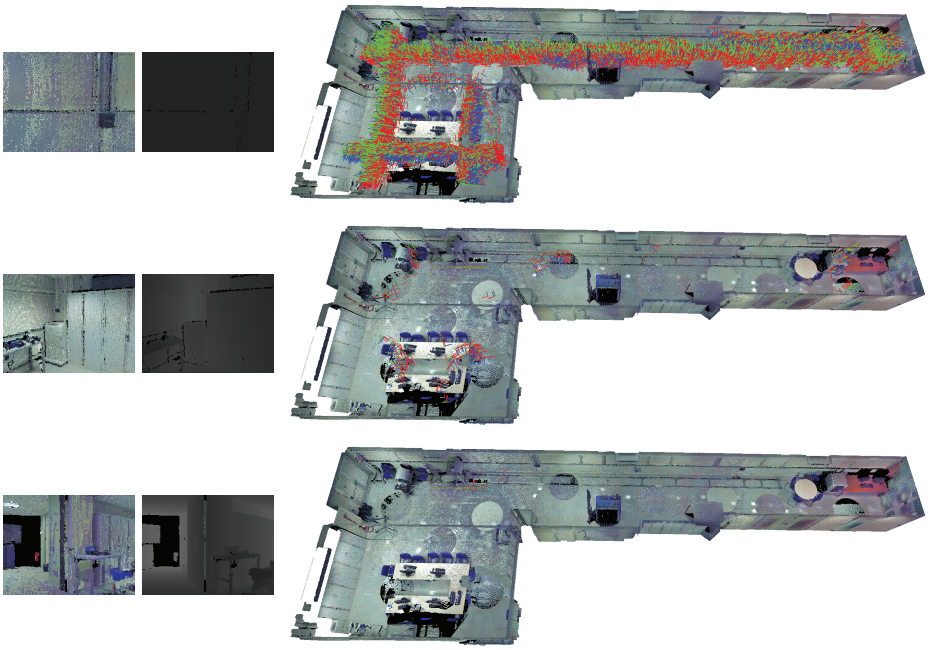

Figure 7.18: *Some poses in decreasing order of ambiguity. Left side of the figure shows the readings from the sensor (RGB and Depth channels). Right side of the figure shows the results produced by the kidnapped solver considering all training poses.*

To perform such optimization, a kd-tree is built using all the training data (except for

the previously deleted incorrect poses). For each entry, a neighbor search is performed using an hyper-sphere of radius r. Only poses with less than k neighbors will be used for solving the kidnaped problem, deleting all the rest.

As Figure 7.19 shows if, after computing the number of neighbors for each pose, poses are sorted according to this value, it can easily be distinguished which poses are ambiguous and which are not. For the given example, the first 300.000 training poses show low ambiguity, whilst all the rest are highly ambiguous.

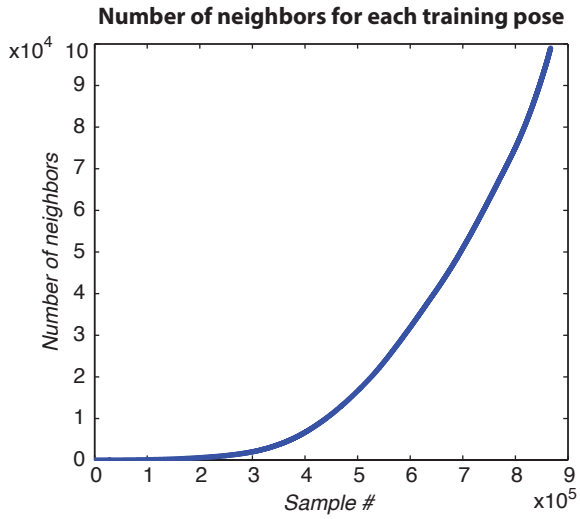

Figure 7.19: *Number of neighbors for each training pose.*

Figure 7.20 shows the density distribution of ambiguous poses. Notice how close to the walls there are few ambiguous poses, since they have already been deleted for being incorrect. The corridor presents high ambiguity due to the lack of distinctive elements and the center of the room shows low ambiguity values thanks to the furniture and variety of elements that reduce symmetries.

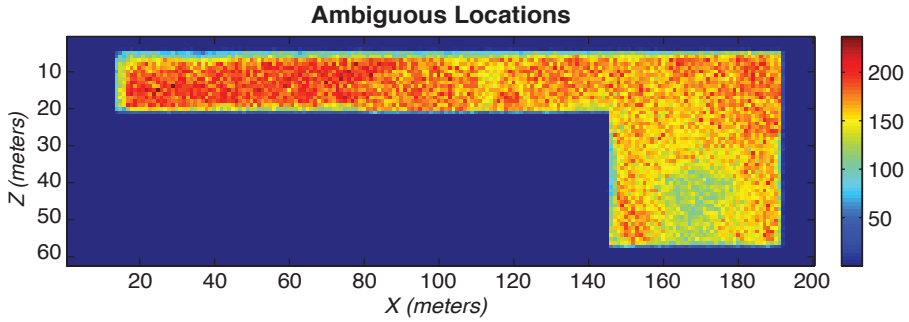

Figure 7.20: *Density of ambiguous training poses for the Kinect.*

After removing ambiguous and incorrect locations, the resulting training set is ready

to be used in the real-time application. Figure 7.21 shows the density of correct training poses remaining after the optimization process. Notice how best positions to be localized are in the center of the room and places nearby the intersection between the corridor and the room.

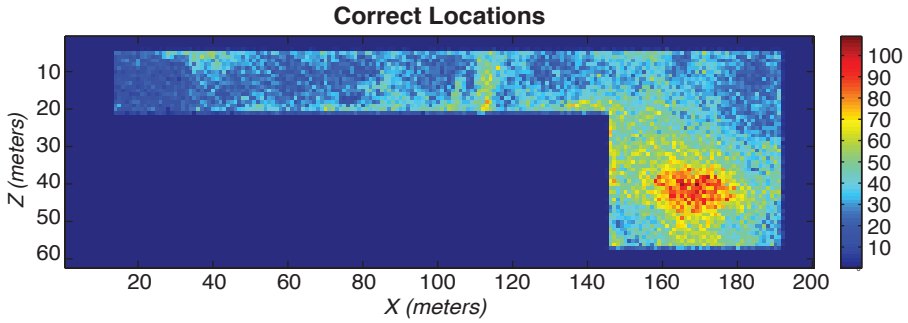

Figure 7.21: *Density of correct training poses for the Kinect.*

Once the training set is calculated, results are stored in a 12th dimensional kd-tree containing the values of the proposed descriptor, together with the poses from which they were taken from. During the on-line execution of the proposed technique, each time the sensor produces a new rangemap and no active locations exist, the proposed descriptor is computed from the depth values and used for a radius search on the training dataset. Most similar training poses are then marked as active locations and inserted into the main application loop. Since ambiguous poses have been filtered, the number of results is constrained, avoiding frame-loss situations. Figure 7.22 summarizes the entire process.

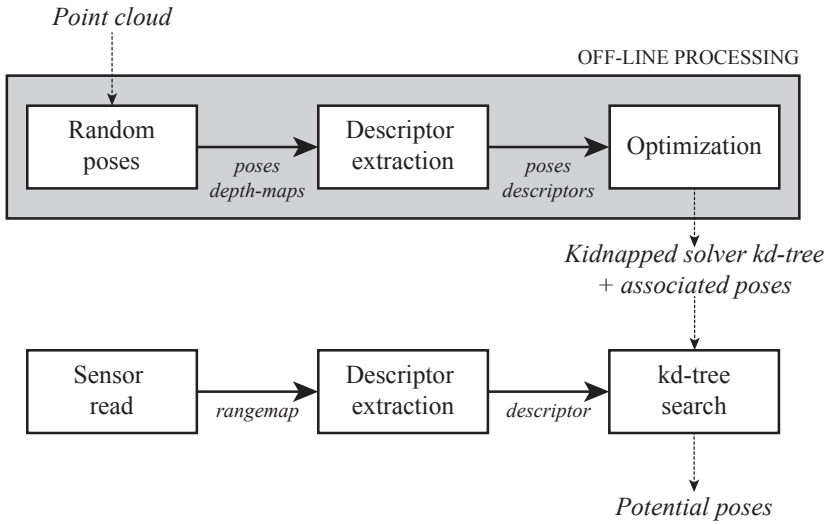

Figure 7.22: *Kidnapped solver overview.*

7.2.4 Tracking and Relocalizing Algorithms

According to the proposed execution cycle, once active poses exist, they have to be tracked as the observer moves inside the known environment. This tracking process is performed as a surface registration approach: if the sensor's observation can be registered with the known environment, the observer's pose can be inferred by simple triangulation.

Given that the approximate pose of the observer has been provided by the kidnapped solver algorithm, no global registration is needed. Instead, the tracking algorithm has to compute the total displacement from the last known position (local registration). To do so, the ICP algorithm is used.

7.2.4.1 Real-time ICP algorithm

The way ICP works makes it an hybrid registration technique: first, it searches for point correspondences and then it calculates a transformation that minimizes an energy function defined over these correspondences. This process is iteratively repeated until a termination criteria is satisfied (typically expressed as a number of iterations, maximum execution time, minimum residual...). It is important to notice that, since each iteration a transformation is applied to one cloud, correspondences between points change and have to be re-computed for subsequent iterations.

The proposed ICP algorithm for tracking finds point correspondences according to their distance. This way, given a point p from the range image acquired by the sensor, its correspondent q, is the closest point to p in the point cloud that represents the known environment. The energy function to minimize is the point-to-point squared distance, which is not a very powerful approach, but can be easily solved by using closed-form solutions like singular value decompositions, quaternions or orthonormal matrices.

In the proposed implementation, given two point-sets P and Q, where P is defined over the range image returned by the sensor and in world coordinates according to the observer's pose, Q is defined over the point-cloud that represents the known environment and the correspondent point to $p_i \in P$ is $q_i \in Q$, the update of each ICP iteration, defined as a translation vector t and a rotation matrix R, is computed as:

$$\mu_P = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{7.18}$$

$$\mu_Q = \frac{1}{N}\sum_{i=1}^{N} q_i \tag{7.19}$$

$$H = \frac{1}{N}\sum_{i=1}^{N} (p_i - \mu_P)(q_i - \mu_Q)^T \tag{7.20}$$

$$[U, S, V] = svd(H) \tag{7.21}$$

$$R = VU^T \tag{7.22}$$

$$t = -R \times \mu_P + \mu_Q \tag{7.23}$$

where N is the total number of points in both datasets, μ_P and μ_Q are the centroids of both point clouds, H is the 3×3 covariance matrix, $svd()$ performs the singular value decomposition of the H matrix, and R and t are the rotation and translation matrices,

respectively, that applied to P minimize the square distance between the considered corresponding points in both datasets.

Calculation costs for the centroids, covariance matrix, SVD decomposition and final roto-translation matrices are not very high in terms of CPU usage. However, finding corresponding points in both point clouds is.

To compute this correspondence, common registration techniques build a kd-tree over the ground truth point cloud, Q, and use it to find the closest point to each sample in the point cloud retrieved by the sensor, P. However, considering that the application is supposed to provide real-time results, and that the number of samples returned by Kinect in a second is around 7,680,000 and for Velodyne is around 700,000, search times using a big ground truth point cloud with so many query points would yield into very slow results.

A common solution to this problem is performing a RANSAC based selection of points for each sensor reading, so high sub-sampling rates are used and faster results are achieved. This solution, however, leads to a precision loss and, eventually, if the sub-sampling rate is too high, to incorrect results. To accelerate this neighbor-search process, an alternative solution is proposed exploiting the fact that the ground truth point cloud is already known: a discrete characterization of the map based on a voxel representation.

A regular voxel representation consists on a discrete 3D matrix, where each cell is a box marked as *full* if it contains some point inside and marked as *empty* otherwise. The proposed voxel representation adds some extra features: for each full cell a vector representing the centroid of all contained samples is stored. For each empty cell, a reference to the closest centroid is stored if the total distance between the box center and the centroid is smaller than a given maximum.

This way, the proposed voxel structure can be understood as a discrete Voronoi diagaram, or a pre-calculated signed distance function, making immediate the selection of the closest neighbor for a given point in space. For example, assume we want to find the closest neighbor in the known map for a given point $p = (p_x, p_y, p_z) \in \mathbb{R}^3$. Its corresponding voxel cell index, $idx = (idx_x, idx_y, idx_z) \in \mathbb{N}^3$ will be calculated as:

$$idx_x = \lfloor p_x / cellSize \rfloor \tag{7.24}$$

$$idx_y = \lfloor p_y / cellSize \rfloor \tag{7.25}$$

$$idx_z = \lfloor p_z / cellSize \rfloor \tag{7.26}$$

where $cellSize$ is the edge length of a single cell, and assuming a positive-defined point cloud. The data structure containing the voxel, accessed using the calculated indices, will return a $null$ value, in case there is no point closest than the given maximum distance, or the position of the closest neighbor otherwise.

Thanks to this characterization, two different functions are indirectly executed: (1) the nearest neighbor of a given point is calculated in a constant time ($\mathcal{O}(n) \approx k$), no matter how big the ground truth point cloud is, or how small voxel cells are. (2) An implicit outlier rejection technique is executed, since points with their correspondent nearest neighbor too far are automatically detected and rejected.

Figure 7.23 shows a partial view of the known environment represented as a point cloud, as a voxel with full cells storing the centroid of the contained samples and as the proposed voxel structure, where empty cells point to their closest neighbor in case the resulting distance is small enough.

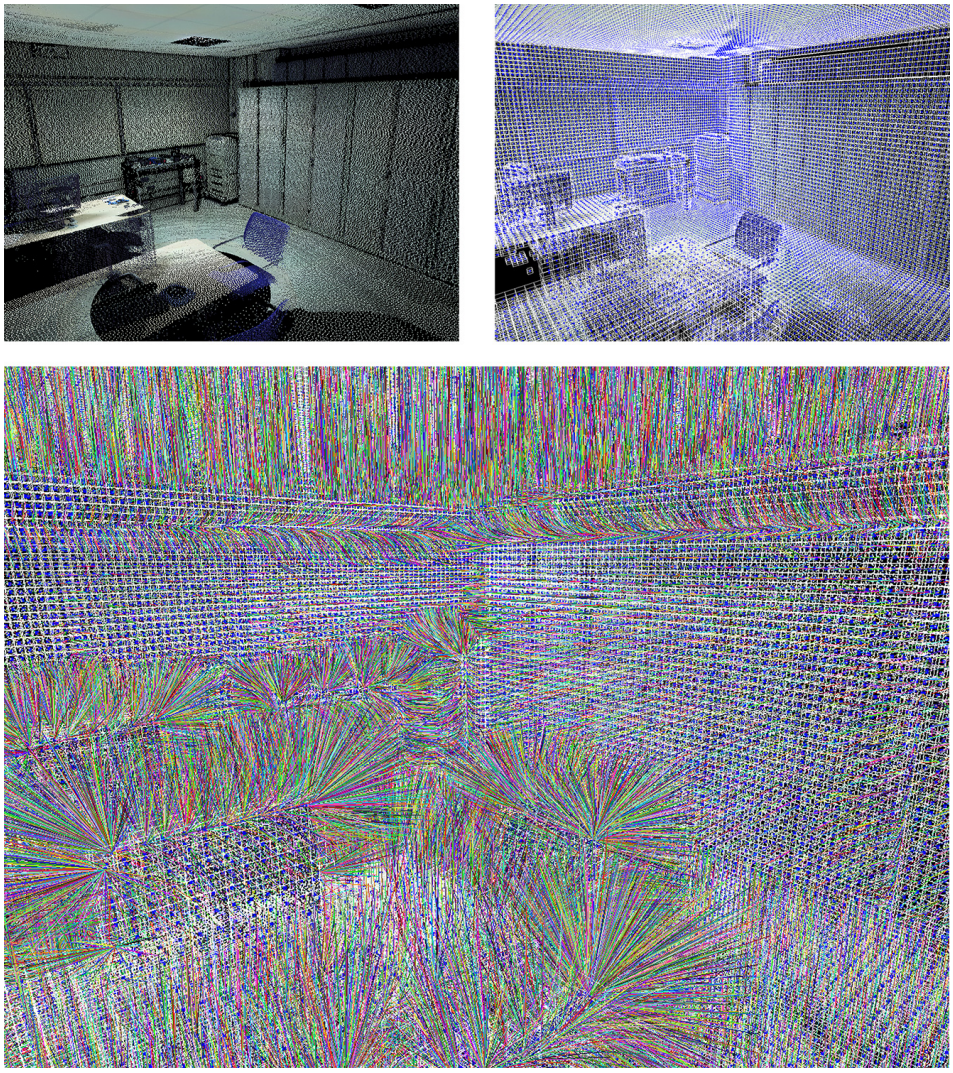

Figure 7.23: *Voxel representation. (top-left) Original point of the known environment. (top-right) Voxel with 5cm box length computed from the original point cloud, and showing the centroid of each cell in blue. (bottom) proposed voxel representation, where empty cells are pointing to the closest centroid in case it is not further than 25cm. Colors are only used to facilitate the reading of the figure.*

In order to calculate the proposed voxel structure from the original point cloud, a jump flooding algorithm like the one proposed in [136] is used, properly adapted to 3D cases. Main advantages of this technique is that it can be easily parallelized, taking advantage of the GPU computing capabilities, and that it provides results at constant execution time with respect to the number of full cells in the voxel.

JFA algorithm computes a discrete Voronoi diagram by performing a set of searches from each voxel cell to its neighbors, with different decreasing search radius. Given a search radius $r \in \mathbb{N}$ and a voxel cell $v = (v_X, v_Y, v_Z) \in \mathbb{N}^3$, another voxel cell w is considered to be a neighbor of v with radius r if, and only if, $w = (v_X + r, v_Y, v_Z)$ or $w = (v_X, v_Y + r, v_Z)$ or $w = (v_X, v_Y, v_Z + r)$. Given this neighborhood function, and a maximum search radius (expressed as a power of two), the algorithm works in different steps, as illustrated on Figure 7.25:

1. The voxel is initialized with all cells marked as null pointers, and populating the full ones with pointers to themselves. The search radius is initialized with the maximum search distance.
2. Perform an initial grow iteration to prevent "crossing triangle" errors (JFA+1).
3. For each cell, a neighbor search is performed with the current search distance.
4. From all the not-null neighbors found from each cell, the distance to their reference is calculated and a reference to the closest one is stored.
5. If the search radius was one, the diagram is complete, so return. Otherwise, the search radius is divided by two and back to step 3.

After the execution of the algorithm, every cell in the voxel has a reference to its closest full cell, or a null reference if the distance is greater than the maximum value indicated by the user. Figure 7.24 shows a graphical representation of the resulting Voronoi diagram.

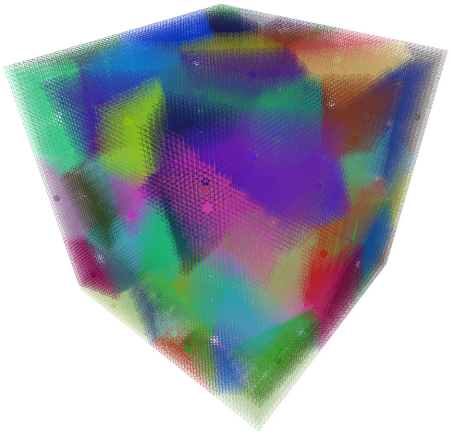

Figure 7.24: *JFA discrete Voronoi diagram. Each color filled cell is a $full$ one and each wireframed cell is colored according to its closest full cell, whose distance is smaller than 8 times the side of a single cell.*

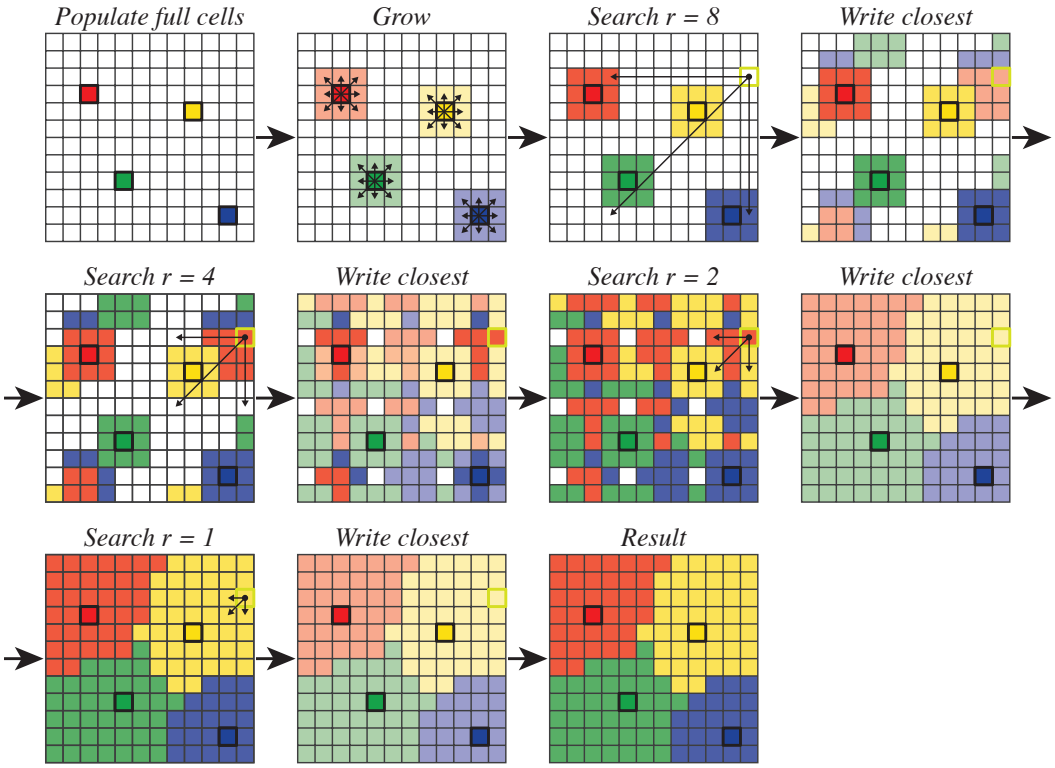

Figure 7.25: *JFA+1 algorithm 2D example, with a maximum search distance of $r = 8$, and with four initial full cells. Each time a search is performed, a sample cell (marked in green) is illustrated. Each time an update is performed (labeled as "Write closest") cells evaluating new neighbors are representes with a light version of the color of their reference.*

7.2.4.2 Tracking

Thanks to the proposed characterization of the environment, ICP registration is performed extremely fast, allowing around 500 iterations per Kinect frame with an average subsampling ratio of 8 (which provides the capability of registering around $60 * 10^6$ points per second, depending on the machine used).

Thanks to this performance, an interesting optimization can be applied to mitigate the effect of the weak point-to-point cost function minimized and to provide more robustness when having lots of outliers: after a new range map is acquired by the sensor, the local registration is initialized assuming that the new pose is a linear extrapolation of the two previous ones (constant speed motion model). However, instead of having only one initialization, a set of k potential poses are considered around the extrapolated one and registered with a high subsampling ratio. After a number of iterations, residuals for each pose are computed and the lowest one is selected for a fine refinement (with more ICP iterations and lower subsampling ratios).

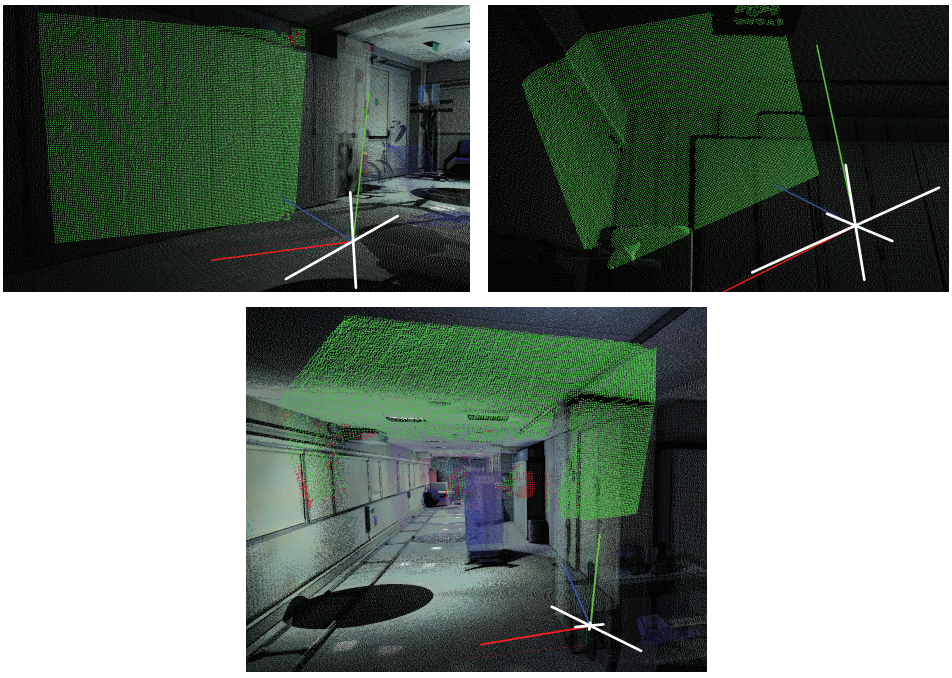

Figure 7.26: *Eigenvector and eigenvalue analysis of ambiguity. Images are captured during real-time tracking. The estimated pose is represented as a three-colored axis, the eigenvectors are represented in white and scaled according to their corresponding eigenvalues.*

To efficiently distribute these poses around the extrapolated one, an analysis of the eigenvectors and eigenvalues of the acquired range map is performed. The goal is to use the eigenvalues to measure the uncertainty associated to the orthogonal axis defined by their correspondent eigenvectors, and distribute the potential poses proportionally to this ambiguity.

Intuitively, this concept is very simple: if the sensor is pointing to a flat surface, ambiguity is distributed over the plane that better fits this surface (Figure 7.26 top-left), leaving two degrees of freedom associated to position and one to rotation poorly defined. If the sensor is pointing to a corner, and with a partial view of the floor/ceiling, ambiguity is equally distributed in all the axes (Figure 7.26 top-right). If the sensor is facing a corridor, ambiguity is distributed along the longitudinal axis of the corridor (Figure 7.26 bottom), leaving one degree of freedom associated to the position poorly defined.

Adding this feature to the tracking algorithm improves robustness considerably but produces a side-effect: initializing each tracking iteration with a set of random poses distributed as commented adds some high frequency noise to the final estimation of the pose. To attenuate this effect in the pose extrapolation algorithm and to prevent bad initializations, a low pass filter is added.

7.2.4.3 Relocalization

During the tracking of a pose it may happen that the achieved registration is considered as incorrect, leading to a tracking lost situation. This can be a consequence of several factors: the number of outliers is too high because the ground truth environment has changed too much since it was acquired, the previous poses of the sensor had high ambiguity (i.e. facing a wall) which lead to a great drift in the pose estimation, the sensor moved too fast or, simply, the initial pose for the tracking algorithm was not correct and registration results are inconsistent with the ground truth map.

In any of these cases, if it is decided that the pose has to be re-localized (see Figure 7.11), a local search starting from the last known good position has to be performed, in order to recover the correct sensor's location. Also, after the kidnapped solver provides a set of potential poses, and in order to initialize the tracking with a better estimation, a re-localization for each one of them has to be performed, taking as the last known position the pose returned by the kidnapped solver.

To do so, and exploiting the proposed voxel characterization and ICP algorithm, a set of random poses are distributed around the last known position. Given that no specific motion model is assumed and that the only cinematic constraints considered are non-correlated maximum displacements and rotational speeds, the potential poses are distributed according to these values and to the time elapsed since the tracking was lost. However, since normally rotational speeds in a hand-held application can be so high, a full 360^{o} uncertainty over each axis is considered.

For each one of these poses, an ICP iteration is performed and a cost function is computed as follows:

$$\|P, V\| = \sum_{p \in P} \|p, V\|^2 \tag{7.27}$$

$$\|p, V\| = \underset{v \in V}{\operatorname{argmin}} \|p, v\| \tag{7.28}$$

being P the point cloud acquired by the sensor, projected in world coordinates according to a given pose, and V the voxel representing the ground truth map.

Poses are then sorted in increasing order according to this sum of residuals, and a set of refinement iterations are executed as follows: worst poses are discarded while best poses are kept for the next iteration together with a new set of poses defined as the component-independent random linear combination of the best poses. After each iteration, the number of potential poses is progressively reduced. This way, a genetic-like algorithm is executed that accelerates convergence to the global minimum.

Figure 7.27 illustrates an example case of re-localization, where the initial ambiguity value is very high (notice the radius of the sphere in the top image), and 5000 random locations are created around the last known position. For initial guesses also notice how extremely high subsampling values are used and progressively refined when the number of potential poses gets reduced.

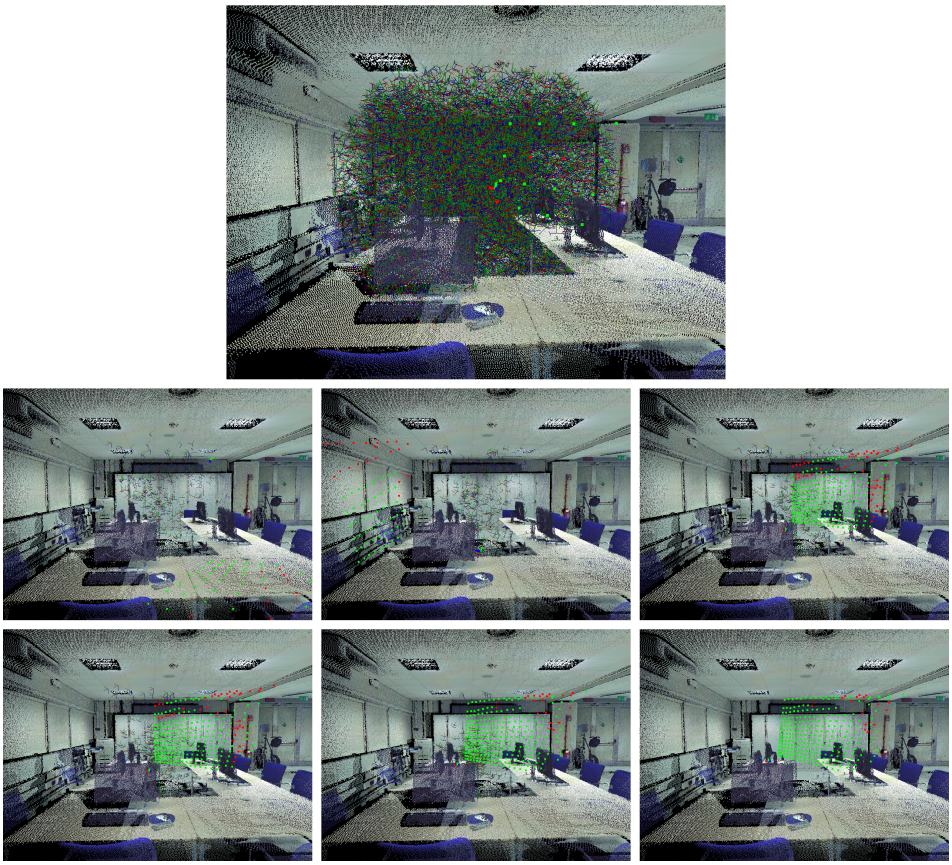

Figure 7.27: *Relocalization algorithm. Top image shows the potential poses used for initialization. Next images, from left to right and from top to bottom show the partial result of each iteration, illustrating the projected point cloud returned by the sensor from the pose that minimizes the proposed cost function. Notice how convergence is very fast (the third iteration has almost found the correct solution).*

7.2.5 Parameter Optimization

In previous sections, a set of parametric algorithms have been proposed to face the indoor localization problem in known environments. Together with these algorithms, Figure 7.11 proposes two conditions which have to be evaluated for each active pose and that have not yet been commented: *"is a pose correct?"* and if not, *"shall it be re-localized?"*.

This section deals with the optimization process of all the parameters involved in the global execution cycle, first by analyzing the factors that have to be considered in the previous questions, and then presenting the optimization framework proposed to tune all the parameters.

7.2.5.1 Tracking lost detection

In order to decide if a potential pose has been successfully tracked or if, on the contrary, the tracking has been lost and the estimated location cannot be considered as correct, three factors are considered: (1) its geometric consistence with respect to the known environment, (2) its geometric consistence with respect to the previous observations projected from their previous estimated poses and (3) if the cinematic constraints are respected.

First condition is very simple to evaluate, and has been previously defined in equations 7.27 and 7.28: to measure the consistency of the projected rangemap from a given pose with respect to the known environment, the squared sum of the residuals is used. This way, if a maximum value is exceeded, the tracking will be considered as lost. Figure 7.28 shows illustrates the outlier detection taking advantage of the propose voxel representation.

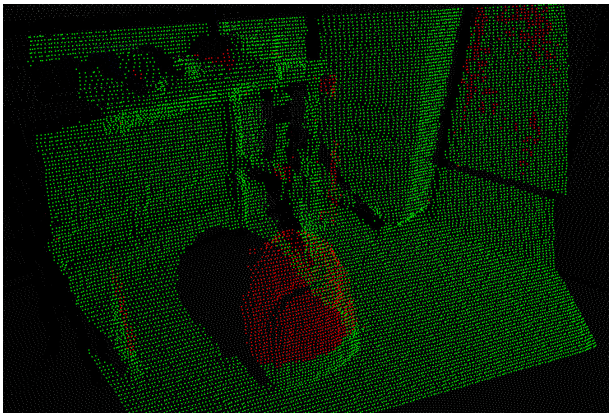

Figure 7.28: *World consistency test. The presence of an outlier represented as a discrete distance function. Samples colored in green are closer than 10cm to the reference point cloud, and samples colored in red are further.*

Second condition locally evaluates the quality of the tracking, ignoring the ground truth model. To do so, given a projected rangemap P_t with respect to the currently tracked pose, and another previously projected rangemap P_{t-k} from its estimated pose, points from P_{t-k} are inversely re-projected into the the current rangemap's image plane, and depth values are between corresponding points are compared in order to compute the square sum of residuals. Figure 7.29 illustrates this concept.

However, since not all the points have its counterpart (due to the observer's movement), the resulting distance has to be divided by the number of corresponding points to have a fair comparison. It is also important to notice that, since re-projection from one rangemap to the other operates on coordinates defined in $\mathbb{R}^3$, the local re-projected (u, v) coordinates in the current range map will be also defined in $\mathbb{R}^2$, so a interpolation between depth values has to be performed. Finally, in order to improve the quality of the estimation, and considering a high frequency acquisition, it is interesting to have a relatively big time separation between compared observations, in order to prevent numerical errors and to better filter the sensor's noise.

Finally, the last condition is satisfied if the total angular and linear speeds computed according to the current pose are inside the range defined by the maximum speeds.

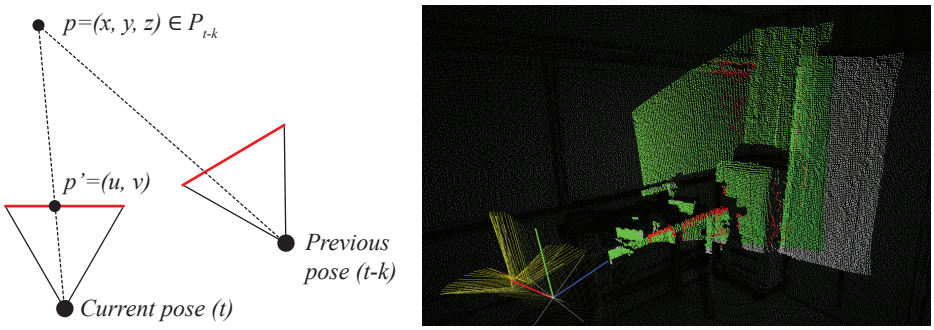

Figure 7.29: *Self error consistency test. Left image illustrates the re-projection of a point observed in a previous instant into the image plane of the currently tracked pose. Right image shows the self error test performed during real-time tracking. The more green points are, the smaller the error is. The more red, the bigger. Gray points are those that do not have counterpart, because of the sensor's movement.*

7.2.5.2 Relocalization test

In order to decide if it is worth spending computational resources in re-localizing a lost pose after a tracking failure, four factors are considered:

1. The tracking was lost not too long ago, so the uncertainty around the last known location is small enough to perform a local search in its surroundings. In Figure 7.11 it is not specified but, when the relocalization is performed, the corrected pose is kept in the list of potentially valid poses for the next iteration. However, only until the tracking succeeds updating this pose with new observations, the pose keeps marked as invalid. This way, before deciding to delete a wrong pose, several relocalization attempts may happen.

2. The lifetime of the pose makes worth spending computational resources in relocalizing it. This way, poses returned by the kidnapped solver have less priority for relocalization, whilst poses that were alive during a long time (tracking had succeeded using many observations) are more probable to be correct, and have to be relocalized when possible.

3. The quality of the observation when the tracking was lost was poor. This way, if a pose has been lost while facing a wall, or having too many null distances because the operating range of the sensor is not respected, the tracking fail can considered as normal and not a consequence of a bad pose. However, if the quality of the observation is good and the tracking has been lost, it is more likely that the pose were wrong.

4. The quality of the current observation is good: it is only worth spending computational resources in relocalizing a pose if the quality of the data used to do so is good. Otherwise, it is better to wait until the sensor produces a point cloud that allows performing such task.

Given that two of the previous conditions are defined according to the quality of the data, a formal way to measure it is defined according to two different criteria:

1. The quality of a range map retrieved by the sensor is proportionally related to the percentage of points defined inside its operating range. Null distances and points outside this range are assumed to be outliers and contribute to degrade the data.

2. The quality of a range map retrieved by the sensor is also proportionally related to the geometric detail of the resulting cloud. This way, an analysis based on the principal components of the cloud (eigenvectors and eigenvalues) is performed, and a scalar descriptor of this quality is defined as the relationship between the first (biggest) and third (smallest) eigenvalues.

7.2.5.3 Optimization framework

In order to set proper values to the parameters involved in the execution of the proposed algorithms, and to perform an optimization over them, two factors have to be considered: (1) a cost function has to be defined that considers the accuracy of the achieved results together with the execution time and (2) a set of ground truth datasets with known motions has to be acquired in order to perform an off-line optimization that can compare achieved results with the reference ones to compute the proposed cost function.

For the cost function computation, it is important to notice that there are two objectives to optimize, accuracy and performance, and that the two of them can be correlated: looking for excessively accurate results may lead to a critical performance loss, which will also render into losing sensor readings, which will produce tracking fails and, eventually, would also affect to accuracy. This way, a proper balance between both objectives has to be achieved. In the proposed (ongoing) optimization strategy, the way these two objectives are combined in a single cost function consists on calculating the total number of sensor readings that produce a $correct$ pose, considering that a pose is qualified as correct if its inside of a given distance range from the correct one.

For this cost function, it is also important to notice that the optimization is highly dependent on the processing capabilities of the computer that is executing the application. This way, instead of a generic calibration, the proposed optimization process has to be executed once for each different computer.

For the ground truth motion estimation, the need of evaluating performance and accuracy when moving inside large environments adds an extra difficulty: no motion detection facilities or robotic arms can be used to create reference models. Instead of this, and considering that the proposed tracking algorithm provides extremely accurate results when using the Velodyne sensor (as next section shows), a sensor holder has been designed and 3D printed in order to use simultaneously Kinect and Velodyne sensors (see Figure 7.30).

Given that, before recording the datasets, both sensors have been calibrated one with respect to the other, Velodyne sensor is used to compute the ground truth trajectory (properly transformed into the Kinect global coordinates), and the Kinect sensor readings are used as input data for the calibration process.

Figure 7.30: *Design of the sensor holder used for generating training datasets. (left) 3D model of the holder. (right) 3D printed version with both sensors attached.*

Once the training dataset is recorded, a grid search optimization process is executed with some manually predefined ranges for some parameters (like subsampling ratios during the ICP registration, maximum number of registration iterations, maximum number of random poses for the relocalization...). The goal is then to find the values for the parameters that minimize the proposed cost function as follows:

$$\text{minimize}_{\Theta=\theta_1,\theta_2\ldots\theta_n} \frac{1}{2m}\sum_{1}^{m}\left(f_\Theta\left(x^{(i)}\right)-y^{(i)}\right)^2 \tag{7.29}$$

being $\Theta = \theta_1, \theta_2 \ldots \theta_n$ the set of n parameters to optimize, m the number of sensor readings of a given training motion, $f_\Theta\left(x^{(i)}\right) = (x, y, z, \alpha, \beta, \gamma)^{(i)}$ the i-th pose returned by the proposed algorithm using the set of parameters evaluated and $y^{(i)} = (x', y', z', \alpha', \beta', \gamma')^{(i)}$ the i-th ground truth pose calculated with the Velodyne, and transformed into the Kinect reference frame.

7.2.6 Results

In order to perform the proposed parameter optimization for the Kinect sensor, a ground truth trajectory is needed to compare with. To do so, as proposed, the Velodyne sensor is being used properly attached and calibrated with the Kinect sensor. However, given that the result of the tracking using the Velodyne is assumed to be correct, its accuracy has to be evaluated.

To do so, a set of experiments has been performed in both, structured and unstructured environments. The proposed structured environment is the interior of a building (shown in previous figures), where there lots of elements have moved since the ground truth model was acquired. The unstructured environment is a rocky tunnel very similar to some nuclear

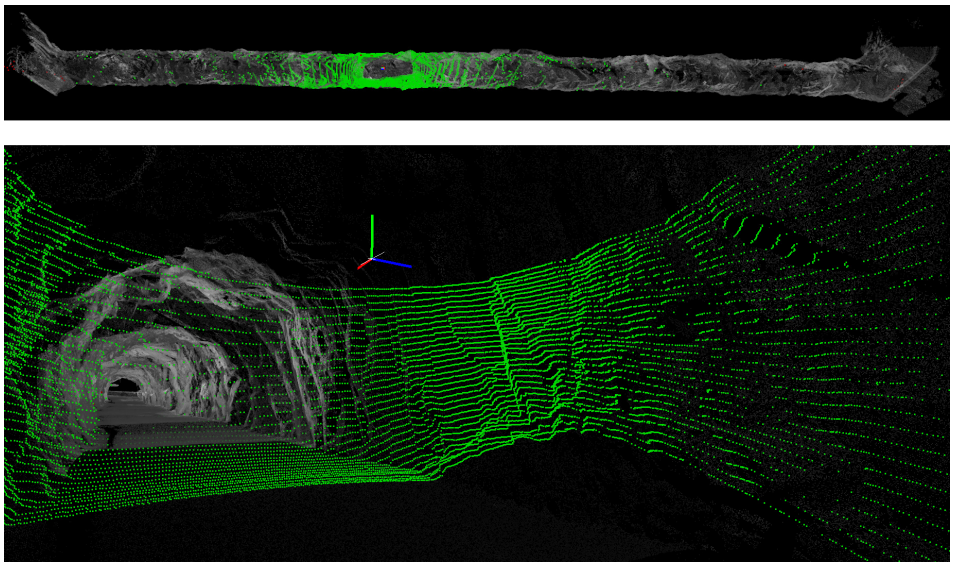

Figure 7.31: *Evaluation tunnel used for the Velodyne sensor. The top image povides a general overview of the tunnel (120 meters long), whilst the bottom one shows the interior shape of the walls. Both images represent the same data during one of the evaluation tests. Notice how no red dots are represented, meaning that all points from the sensor have a counterpart in the ground truth point cloud. This is a consequence of both, the absence of outliers and a correct tracking.*

facilities where the proposed system will be used. For this environment, hand-held and car-mounted tests have been performed at different speeds. Figure 7.31 shows the acquired point cloud for the tunnel.

In the tests performed in the tunnel environment, the reference point cloud was acquired just before the data retrieval, so no outliers exist. For each test, the tracking algorithm has been executed with a voxel cell size of 10cm. In an offline process, a fine registration of the data recorded and the ground truth point cloud has been performed in order to compare the real-time results using the voxel against it. Table 7.2 and Figures 7.32, 7.33, 7.34 and 7.35 show the achieved results.

#Track	Motion	Avg. speed	Distance	Avg. error	Std. dev.
1	hand-held	0.87 m/s	226.06 m	2.42 cm	3.60 cm
2	car-mounted slow	1.42 m/s	113.64 m	1.74 cm	2.08 cm
3	car-mounted fast	4.30 m/s	75.75 m	3.71 cm	3.50 cm
4	car-mounted backwards	1.25 m/s	106.05 m	1.66 cm	1.76 cm

Table 7.2: *Results of the evaluation tests performed in the tunnel with the Velodyne sensor.*

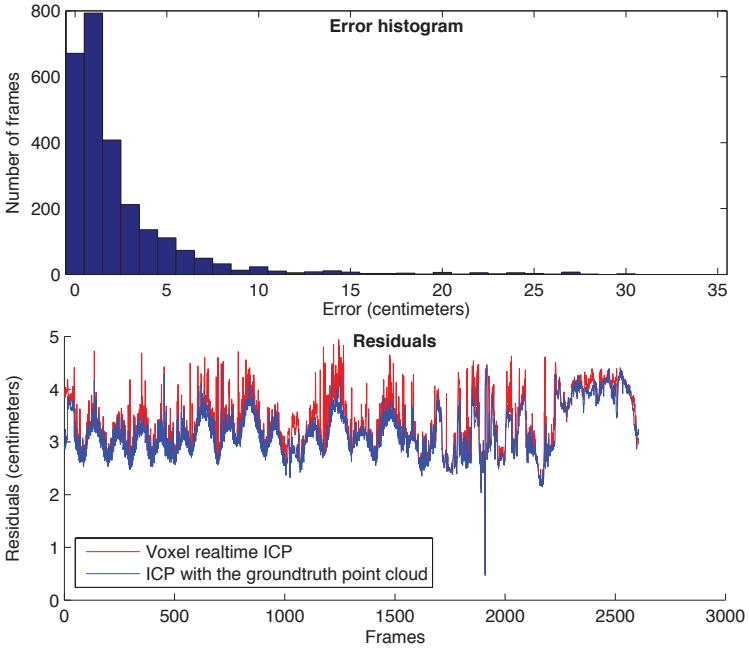

Figure 7.32: *Velodyne tracking results for track 1 (hand-held).*

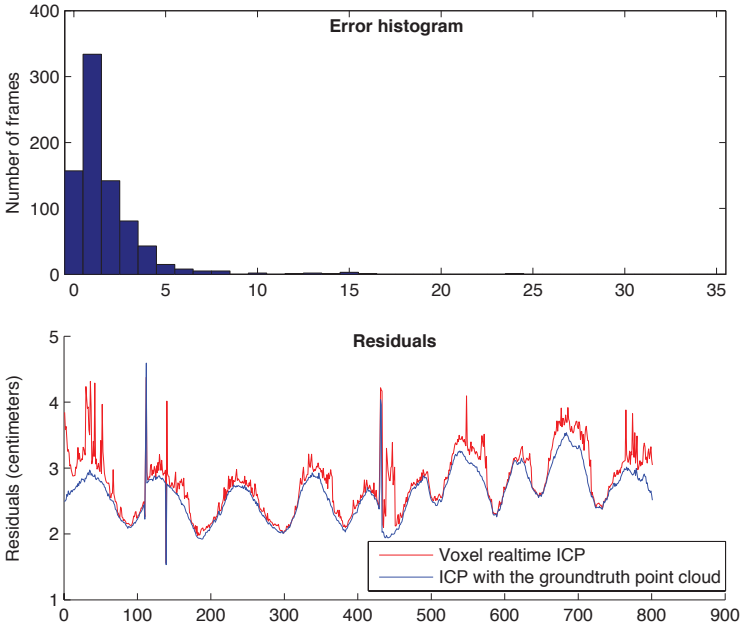

Figure 7.33: *Velodyne tracking results for track 2 (car-mounted, slow movement).*

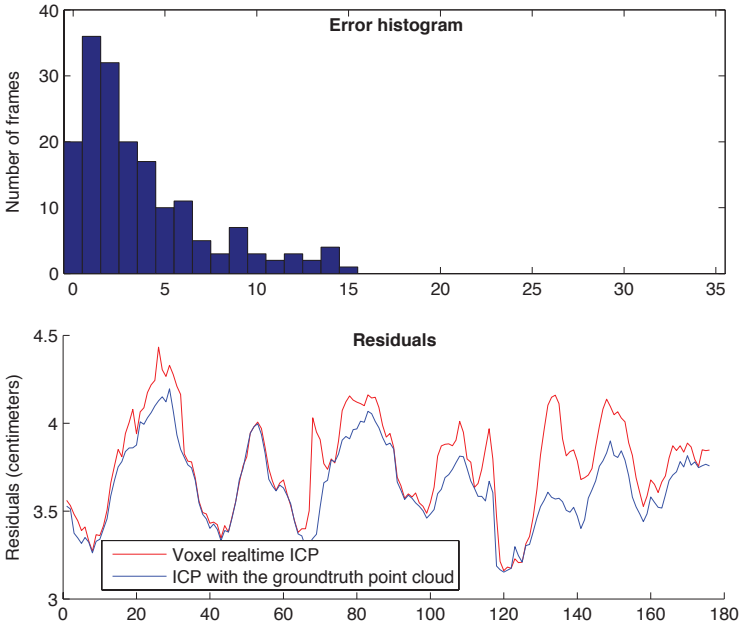

Figure 7.34: *Velodyne tracking results for track 3 (car-mounted, fast movement).*

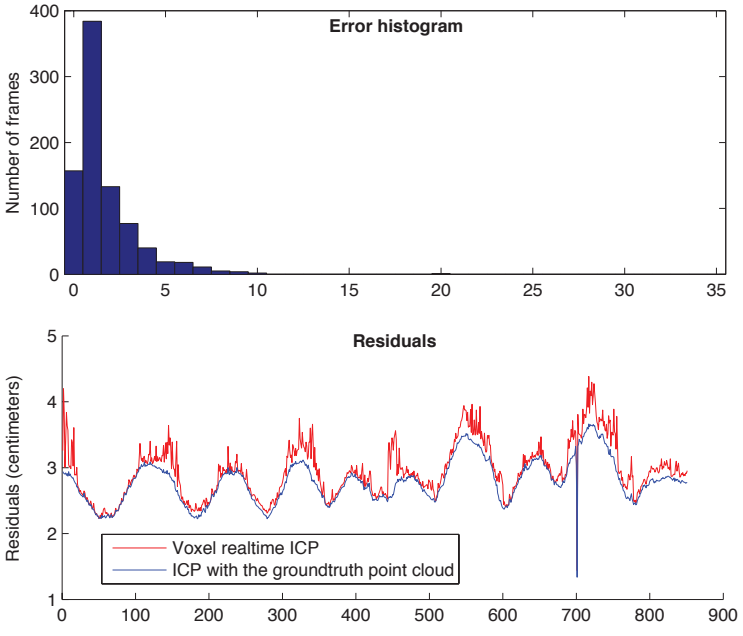

Figure 7.35: *Velodyne tracking results for track 4 (car-mounted, backwards).*

From Table 7.2 it can be appreciated how speed affects directly on the average error and standard deviation. This is, mostly, a consequence of the previously commenter unwarping effect with the Velodyne sensor and does not necessarily mean that the tracking algorithm is failing. In the case of the hand-held track, it can be noticed how the standard deviation increases considerably, as a consequence of the shaking produced in this kind of motion.

In the previous figures, it can be seen how the position error histograms always peak in the 1 centimeter error bin, having most of the samples inside the $0...5$ centimeters range. In the residuals plots, first thing to notice is how error periodically increases and decreases in both, real-time and off-line results. This is not a consequence of a bad tracking, but of a bad ground truth point cloud: given that the scans were taken each 15 meters and that the tunnel is extremely narrow, areas close to the acquisition locations are extremely well defined and areas in between two acquisition positions are poorly defined.

As a consequence of this factor, residuals are low when the environment is well defined because the nearest neighbor for each point acquired by the sensor can be better estimated. On the other side, in areas poorly defined, the closest neighbor for a sensor point do not represent properly the nearest point in the tunnel, so residuals increase. For this reason, as an estimation of the error produced, local minimum values should be used as a reference.

As Table 7.2 showed, residuals depend on the speed, having low values in tracks 2 and 4, and higher values in tracks 1 and 3. In the first track (hand-held), higher residuals around frame 1200 and in the end are a consequence of the sensor getting out of the tunnel, where no ground truth model exists, and less correspondences can be used for tracking. Also in the last stage of the track, to test for robustness, the person holding the sensor made fast movements and jumped. The tracking was able to locate the sensor in all the stress test, but a bit of precision was lost.

Regarding to the data representation, tests performed in the structured indoor environment showed the results detailed in Table 7.3. It is important to notice how, the more general the problem is, the less memory is required to store the data necessary to solve it. This is very important in the case of considering a big environment: in order to solve the kidnapped robot problem, the whole environment has to be considered, so its spatial requirements have to be small enough to fit in memory. For tracking and re-localization purposes, only the part of the facility surrounding the last known pose has to be considered, so a higher detail representation ca be afforded.

Representation	# samples	Size (MB)
Point cloud	15,104,800 points	345
Voxel (5cm cell size)	18,668,538 cells	213
Voxel (10cm cell size)	2,347,380 cells	26.8
Training dataset raw	500,000 poses	34.3
Training dataset optimized	165,585 poses	11.3

Table 7.3: *Memory requirements to store the propsed data structures*

7.2.7 Conclusions and Future Works

In this chapter, a general technique has been proposed in order to face the particularities of the localization in indoor environments using 3D sensors. Three major situations have been identified: the kidnapped robot problem, the tracking problem and the re-localization problem.

The kidnapped robot problem refers to estimating the pose of the observer with no prior information. To do so, and considering that he can be located in any point of the facility, a global search strategy has been proposed that simplifies the range map acquired by the sensor into a set of descriptors. Using this signature, a fast k-dimensional search is performed over pre-computed training data, obtained using the ground truth map of the facility, to find the most similar potential locations. An optimization stage has been also proposed to prune from the training dataset ambiguous and incorrect potential poses.

Once the observer is localized inside the map, a tracking algorithm is executed using next sensor's observations in order to update his pose. These updates are preformed using a local search strategy that only considers the part of the environment that surrounds the sensor and, as a result, provide a local displacement from the previous known pose. To efficiently do so, a voxel-based representation of the ground truth point cloud has been proposed. This representation allows accelerating the registration process between the partial view observed by the sensor and the known environment, providing real-time results.

In the eventual case of a tracking-lost situation as a consequence of several factors, like a fast movement of the inspector or the presence of too many outliers, a re-localization strategy has been proposed. This algorithm performs a local search around the last known position of the observer, distributing potential locations distributed according to an uncertainty model and using a genetic-like strategy to accelerate convergence. Taking advantage of the proposed characterization of the environment, this search is performed extremely fast and allows to be integrated in the final real-time application.

A complete execution cycle has also been introduced, and a procedure to calibrate the parameters involved in each stage has been presented.

Since this is an ongoing project, major improvements are being implemented in almost every stage. Firstly, for the general ICP algorithm that is used in tracking and relocalization stages, a point-to-plane distance function is being minimized instead of a point-to-point. Advantages of this minimization is that it converges faster than the point-to-point one and, for some situations, it is able to solve registrations that the simplest approach cannot. In order to keep the computational cost of this operation as low as possible, normals will be calculated on the ground truth model (so no extra pre-processing time is required for the range maps acquired by the sensor), and orientations will be linearized in order to analytically solve the minimization problem.

To improve the kidnapped robot solver algorithm, and to avoid populating with excessive number of potential locations the proposed execution cycle, a local odometry estimation algorithm ignoring the ground truth map is being implemented. The main idea consist on executing the proposed kidnapped solver in order to get potential locations but, instead of tracking them directly using the ground truth map, a particle filter fed with the motion estimation is proposed. This way, after several observations, the kidnapped solver is executed again and only particles in poses compatibles with the new set of potential locations are kept alive. This process is iteratively repeated until only one potential location exists.

Thanks to this optimization, global execution performance is expected to improve and

the memory requirements are expected to be considerably lower: since only one potential pose exists, only the part of the voxel surrounding it has to be loaded into memory, allowing to work with bigger ground truth environments.

Finally, in order to create a portable tool that can be used in real situations, a backpack is being designed in order to fix the sensor and batteries to the inspector and a portable hand-held display is being designed to provide useful information in real time.

CHAPTER 8

Conclusions

In this document surface registration techniques have been applied to solve the automatic reconstruction of broken archaeological artifacts from fragments.

For archaeological and computational complexity reasons, the problem has been divided into two different sub-problems and treated independently: a 3 degrees of freedom approach to face problems that consider flat fragments and a 6 degrees of freedom approach to provide a general solution to the reconstruction problem.

To work with digital representations of fragments an initial acquisition stage, common to both approaches, has been studied. Working with fragile and singular objects imposes physical restrictions when scanning reflective/refractive surfaces. To face them, an alternative use of cyclododecane has been proposed. Thanks to its good film forming capabilities, its chemical stability and to the fact that it sublimes at room temperature leaving no residuals, CCD is a perfect candidate to solve reflection/refraction issues during the scanning process. A set of experiments has been presented, proving that the thin layer created on the surface of fragments do not interfere with the scanner accuracy thanks to the reduced particle size.

To face 3 degrees of freedom problems, the reduced size of the solution space has been exploited. Starting from an exhaustive approach that evaluates all possible discrete alignments, a set of optimizations have been proposed to increase the overall performance while keeping the correction of results. It has been formally and empirically demonstrated that the use of the proposed optimistic cost function estimator for intermediate results ensures convergency and speeds up considerably the search process. Also, by taking advantage of modern GPUs, all heavy computations are executed in specifically designed wired units, so the final algorithm performs the registration using only addition and comparison oper-

ators. Combining these advantages with an alternative characterization of fragments that ensures data alignment, achieved execution times have shown a performance boost of the proposed technique with respect to previous approaches.

To face 6 degrees of freedom problems, the combinatory explosion that happens in the solution space when adding three extra dimensions has been proven to make exhaustive approaches prohibitive in terms of execution times and memory requirements. This fact makes mandatory a paradigm change: instead of looking for a dense correspondence using an exhaustive technique, a sparse registration algorithm has been proposed. Taking advantage of a pre-processing stage, an alternative characterization for fragments has been introduced. Using a key-point selection algorithm based on a multi-scale saliency feature and a modified version of the PFH descriptor, a compact and descriptive representation of fragments has been proposed. A fast one-to-one three-level hierarchical search strategy has been presented to exploit local similarity between keypoints and geometrical consistence tests during the registration process. Then, using these results, a many-to-many search algorithm performs the final reconstruction taking advantage of modern computers' architecture and using a graph-based strategy. Empirical results have proven the technique to be very fast and suitable for solving large problems.

To show alternative applications of surface registration techniques to other fields, two localization techniques have been presented. First one exploits the structured nature of indoor environments to solve the self-localization problem for mobile robots with very limited computational resources. In order to keep execution times low, a sparse registration technique is proposed that takes advantage of a very fast line inference algorithm. Second one allows nuclear inspectors to localize and detect changes in known environments for verification and inspection purposes. A complete execution cycle for the self-localization problem has been presented where, the fact of having a ground truth map of the environment is effectively exploited: by pre-computing a training dataset of synthetically generated points of view, an efficient kidnapped solver has been introduced. Also, a voxel-based representation of the ground truth point cloud has been proposed. This representation allows accelerating the registration process between the partial view observed by the sensor and the known environment, providing real-time results.

Bibliography

[1] Dror Aiger, Niloy J. Mitra, and Daniel Cohen-Or. 4pointss congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 85:1–85:10, New York, NY, USA, 2008. ACM.

[2] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R.B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 585–592, Nov 2011.

[3] Marc Alexa. Recent advances in mesh morphing. *Comput. Graph. Forum*, 21(2):173–196, 2002.

[4] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22(3):587–594, July 2003.

[5] Tom Altman. Solving the jigsaw puzzle problem in linear time. *Appl. Artif. Intell.*, 3(4):453–462, January 1990.

[6] Dragomir Anguelov, Praveen Srinivasan, Hoi cheung Pang, and Daphne Koller. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *In TR-SAIL-2004-100*, pages 33–40, 2004.

[7] K.S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(5):698–700, Sept 1987.

[8] H Asada and M Brady. The curvature primal sketch. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):2–14, January 1986.

[9] Oscar Kin-Chung Au, Chiew-Lan Tai, Daniel Cohen-Or, Youyi Zheng, and Hongbo Fu. Electors voting for fast automatic shape correspondence. *Comput. Graph. Forum*, 29(2):645–654, 2010.

[10] S. Belongie and J. Malik. Matching with shape contexts. In *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on*, pages 20–26, 2000.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, April 2002.

[12] R. Benjemaa and F. Schmitt. Fast global registration of 3d sampled surfaces using a multi-z-buffer technique. In *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, pages 113–120, May 1997.

[13] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 26–33, Washington, DC, USA, 2005. IEEE Computer Society.

[14] F. Berrada, D. Aboutajdine, S.E. Ouatik, and A. Lachkar. Review of 2d shape descriptors based on the curvature scale space approach. In *Multimedia Computing and Systems (ICMCS), 2011 International Conference on*, pages 1–6, April 2011.

[15] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.

[16] Silvia Biasotti, Simone Marini, Michela Spagnuolo, and Bianca Falcidieno. Sub-part correspondence by structural descriptors of 3d shapes. *Computer-Aided Design*, 38(9):1002 – 1019, 2006. Shape Similarity Detection and Search for CAD/CAE Applications Shape Similarity Detection and Search for CAD/CAE Applications.

[17] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):820–824, August 1995.

[18] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[19] M. Bober. Mpeg-7 visual shape descriptors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):716–719, Jun 2001.

[20] Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321 – 345, 1984.

[21] E. Boyer, A. M. Bronstein, M. M. Bronstein, B. Bustos, T. Darom, R. Horaud, I. Hotz, Y. Keller, J. Keustermans, A. Kovnatsky, R. Litman, J. Reininghaus, I. Sipiran, D. Smeets, P. Suetens, D. Vandermeulen, A. Zaharescu, and V. Zobel. Shrec 2011: Robust feature detection and description benchmark. In *Proceedings of the 4th Eurographics Conference on 3D Object Retrieval*, EG 3DOR'11, pages 71–78, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association.

[22] Alexander M. Bronstein, Michael M. Bronstein, Alfred M. Bruckstein, and Ron Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *Int. J. Comput. Vision*, 84(2):163–183, August 2009.

[23] Benedict J. Brown. *Registration and Matching of Large Geometric Datasets for Cultural Heritage Applications*. PhD thesis, Princeton University, June 2008.

[24] Benedict J. Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 84:1–84:9, New York, NY, USA, 2008. ACM.

[25] T.S. Caetano, J.J. McAuley, Li Cheng, Quoc V. Le, and A.J. Smola. Learning graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(6):1048–1058, June 2009.

[26] A. G. Castañeda, B. J. Brown, S. Rusinkiewicz, T. Funkhouser, and T. Weyrich. Global consistency in the automatic assembly of fragmented artefacts. In *Proceedings of the 12th International conference on Virtual Reality, Archaeology and Cultural Heritage*, VAST'11, pages 73–80, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association.

[27] M. Emre Celebi and Y. Alp Aslandogan. A comparative study of three moment-based shape descriptors. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I - Volume 01*, ITCC '05, pages 788–793, Washington, DC, USA, 2005. IEEE Computer Society.

[28] Will Chang and Matthias Zwicker. Automatic registration for articulated shapes. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1459–1468, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[29] Will Chang and Matthias Zwicker. Range scan registration using reduced deformable models. *Comput. Graph. Forum*, 28(2):447–456, 2009.

[30] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003.

[31] Jiun-Hung Chen and L.G. Shapiro. Groupwise pose normalization for craniofacial applications. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 248–255, Jan 2011.

[32] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, April 1992.

[33] Taeg Sang Cho, Moshe Butman, Shai Avidan, and William T. Freeman. The patch transform and its applications to image editing. In *CVPR*. IEEE Computer Society, 2008.

[34] G.C.-H. Chuang and C.-C.J. Kuo. Wavelet descriptor of planar curves: theory and applications. *Image Processing, IEEE Transactions on*, 5(1):56–70, Jan 1996.

[35] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.*, 89(2-3):114–141, February 2003.

[36] Moo K. Chung, Richard Hartley, Kim M. Dalton, and Richard J. Davidson. Encoding cortical surface by spherical harmonics. *Statistica Sinica*, 18(4):1269–1291, 2008.

[37] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, May 2007.

[38] H. C. da Gama Leitão and J. Stolfi. A multiscale method for the reassembly of two-dimensional fragmented objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:1239–1251, September 2002.

[39] P. Daras and A. Axenopoulos. A compact multi-view descriptor for 3d object retrieval. In *Content-Based Multimedia Indexing, 2009. CBMI '09. Seventh International Workshop on*, pages 115–119, June 2009.

[40] Petros Daras and Apostolos Axenopoulos. A 3d shape retrieval framework supporting multimodal queries. *Int. J. Comput. Vision*, 89(2-3):229–247, September 2010.

[41] T. Darom and Y. Keller. Scale-invariant features for 3-d mesh models. *Image Processing, IEEE Transactions on*, 21(5):2758–2769, May 2012.

[42] Rhodri Davies, Carole Twining, and Chris Taylor. *Statistical Models of Shape: Optimisation and Evaluation*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[43] E. de Aguiar, C. Theobalt, C. Stoll, and H. P Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[44] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 98:1–98:10, New York, NY, USA, 2008. ACM.

[45] A. Deever and A. Gallagher. Semi-automatic assembly of real cross-cut shredded documents. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 233–236, Sept 2012.

[46] Erik D. Demaine and Martin L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graph. Comb.*, 23(1):195–208, February 2007.

[47] Gabriella Sanniti di Baja and Stina Svensson. A new shape descriptor for surfaces in 3d images. *Pattern Recognition Letters*, 23(6):703 – 711, 2002. Discrete Geometry for Computer Imagery.

[48] Chitra Dorai, Gang Wang, Anil K. Jain, and Carolyn Mercer. Registration and integration of multiple object views for 3d model construction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):83–89, January 1998.

[49] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[50] M. Fornasier and D. Toniolo. Fast, robust and efficient 2d pattern recognition for re-assembling fragmented images. *Pattern Recogn.*, 38:2074–2087, November 2005.

[51] H. Freeman and L. Garder. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *Electronic Computers, IEEE Transactions on*, EC-13(2):118–127, April 1964.

[52] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In Tomás Pajdla and Jiri Matas, editors, *ECCV (3)*, volume 3023 of *Lecture Notes in Computer Science*, pages 224–237. Springer, 2004.

[53] T. Funkhouser and P. Shilane. Partial matching of 3d shapes with priority-driven search. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 131–142, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[54] T. Funkhouser, H. Shin, C. Toler-Franklin, A. G. Castañeda, B. J. Brown, D. Dobkin, S. Rusinkiewicz, and T. Weyrich. Learning how to match fresco fragments. *J. Comput. Cult. Herit.*, 4(2):7:1–7:13, November 2011.

[55] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Trans. Graph.*, 22(1):83–105, January 2003.

[56] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25(1):130–150, January 2006.

[57] Jürgen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *2009 IEEE Conference on Computer Vision and Pattern Recognition : CVPR 2009*, pages 1746–1753, Miami, USA, 2009. IEEE.

[58] Jürgen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *2009 IEEE Conference on Computer Vision and Pattern Recognition : CVPR 2009*, pages 1746–1753, Miami, USA, 2009. IEEE.

[59] Timothy Gatzke, Cindy Grimm, Michael Garland, and Steve Zelinka. Curvature maps for local shape comparison. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, SMI '05, pages 246–255, Washington, DC, USA, 2005. IEEE Computer Society.

[60] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[61] Guy Godin, Marc Rioux, and Rejean Baribeau. Three-dimensional registration using range and intensity information, 1994.

[62] Steven Gold and Anand Rangarajan. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 2:2–4, 1995.

[63] David Goldberg, Christopher Malon, and Marshall Bern. A global approach to automatic solution of jigsaw puzzles. *Comput. Geom. Theory Appl.*, 28(2-3):165–174, June 2004.

[64] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.

[65] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[66] Tobias Heimann and Hans-Peter Meinzer. Statistical shape models for 3d medical image segmentation: A review. *Medical Image Analysis*, 13(4):543 – 563, 2009.

[67] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 203–212, New York, NY, USA, 2001. ACM.

[68] K. Hori, M. Imai, and T. Ogasawara. Joint detection for potsherds of broken earthenware. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2440–2445, 1999.

[69] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[70] Berthold K. P. Horn, H.M. Hilden, and Shariar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOURNAL OF THE OPTICAL SOCIETY AMERICA*, 5(7):1127–1135, 1988.

[71] Q. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graphics*, 25(3):569–578, 2006.

[72] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1449–1457, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[73] Daniel P. Huttenlocher and Shimon Ullman. Recognizing solid objects by alignment with an image. *Int. J. Comput. Vision*, 5(2):195–212, November 1990.

[74] D.P. Huttenlocher. Fast affine point matching: an output-sensitive method. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 263–268, Jun 1991.

[75] Cheuk Yiu Ip, Daniel Lapadat, Leonard Sieger, and William C. Regli. Using shape distributions to compare solid models. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, SMA '02, pages 273–280, New York, NY, USA, 2002. ACM.

[76] Sandy Irani and Prabhakar Raghavan. Combinatorial and experimental results for randomized point matching algorithms. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry*, SCG '96, pages 68–77, New York, NY, USA, 1996. ACM.

[77] Sou-Young Jin, Suwon Lee, N.A. Azis, and Ho-Jin Choi. Jigsaw puzzle image retrieval via pairwise compatibility measurement. In *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on*, pages 123–127, Jan 2014.

[78] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, May 1999.

[79] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. *ACM Trans. Graph.*, 29(4):102:1–102:12, July 2010.

[80] M. Kampel and R. Sablatnig. On 3d mosaicing of rotationally symmetric ceramic fragments. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 265–268 Vol.2, Aug 2004.

[81] A. Karasik and U. Smilansky. 3d scanning technology as a standard archaeological tool for pottery analysis: practice and theory. *Journal of Archaeological Science*, 35(5):1148–1168, 2008.

[82] Khaled Khairy and Jonathon Howard. Spherical harmonics-based parametric deconvolution of 3d surface images using bending energy minimization. *Medical Image Analysis*, 12(2):217–227, 2008.

[83] Suchitra Khoje and Shrikant Bodhe. Article: Performance comparison of fourier transform and its derivatives as shape descriptors for mango grading. *International Journal of Computer Applications*, 53(3):17–22, September 2012. Published by Foundation of Computer Science, New York, USA.

[84] Whoi-Yul Kim and Yong-Sung Kim. A region-based shape descriptor using zernike moments. *Signal Processing: Image Communication*, 16(1–2):95 – 102, 2000.

[85] D. Koller and M. Levoy. *Computer-aided reconstruction and new matches in the Forma Urbis Romae*, volume Supplement, pages 103–125. 2006.

[86] W. Kong and B. B. Kimia. On solving 2d and 3d puzzles using curve matching. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:583–590, 2001.

[87] Marcel Körtgen, G. J. Park, Marcin Novotni, and Reinhard Klein. 3d shape matching with 3d shape contexts. In *The 7th Central European Seminar on Computer Graphics*, April 2003.

[88] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):869–885, September 1992.

[89] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1265–1278, August 2005.

[90] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1482–1489, Washington, DC, USA, 2005. IEEE Computer Society.

[91] Hao Li, Bart Adams, Leonidas J. Guibas, and Mark Pauly. Robust single-view geometry and motion reconstruction. *ACM Trans. Graph.*, 28(5):175:1–175:10, December 2009.

[92] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1421–1430, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[93] Xinju Li and Igor Guskov. Multi-scale features for approximate alignment of point-based surfaces. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[94] Huei-Yung Lin and Wen-Cheng Fan-Chiang. Reconstruction of shredded document based on image feature matching. *Expert Syst. Appl.*, 39(3):3324–3332, February 2012.

[95] Yaron Lipman and Thomas Funkhouser. Möbius voting for surface correspondence. *ACM Trans. Graph.*, 28(3):72:1–72:12, July 2009.

[96] Hairong Liu, Shengjiao Cao, and Shuicheng Yan. Automated assembly of shredded pieces from multiple photos. *Multimedia, IEEE Transactions on*, 13(5):1154–1162, Oct 2011.

[97] Qiong Liu. A survey of recent view-based 3d model retrieval methods. *CoRR*, abs/1208.3670, 2012.

[98] Rong Liu, Hao Zhang, Ariel Shamir, and Daniel Cohen-Or. A part-aware surface metric for shape analysis. *Comput. Graph. Forum*, 28(2):397–406, 2009.

[99] Yi Liu, Hongbin Zha, and Hong Qin. The generalized shape distributions for shape matching and analysis. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on*, pages 16–16, June 2006.

[100] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[101] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[102] M. Gross M. Pauly, R. Keiser. Multi-scale feature extraction on point-sampled surfaces. In *Eurographichs 2003*, 2003.

[103] João Maciel and João P. Costeira. A global solution to sparse correspondence problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):187–199, February 2003.

[104] C. Maes, T. Fabry, J. Keustermans, D. Smeets, P. Suetens, and D. Vandermeulen. Feature detection on 3d face surfaces for pose normalisation and recognition. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*, pages 1–6, Sept 2010.

[105] Siddharth Manay, Daniel Cremers, Byung-Woo Hong, Anthony J. Yezzi, and Stefano Soatto. Integral invariants for shape matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1602–1618, October 2006.

[106] William Marande and Gertraud Burger. Mitochondrial DNA as a Genomic Jigsaw Puzzle. *Science*, 318(5849):415+, October 2007.

[107] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3-d model construction. In *Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I - Volume 7270*, ICPR '96, pages 879–, Washington, DC, USA, 1996. IEEE Computer Society.

[108] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. pages 35–57. Springer-Verlag, 2002.

[109] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.

[110] Yang Mingqiang, Kpalma K. Idiyo, and Ronsin Joseph. A Survey of Shape Feature Extraction Techniques. *Pattern Recognition, Peng-Yeng Yin (Ed.) (2008) 43-90*, pages 43–90, November 2008.

[111] Niloy J. Mitra, Simon Flöry, Maks Ovsjanikov, Natasha Gelfand, Leonidas Guibas, and Helmut Pottmann. Dynamic geometry registration. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 173–182, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[112] Debajyoti Mondal, Yang Wang, and Stephane Durocher. Robust solvers for square jigsaw puzzles. In *Proceedings of the 2013 International Conference on Computer and Robot Vision*, CRV '13, pages 249–256, Washington, DC, USA, 2013. IEEE Computer Society.

[113] P.J. Neugebauer. Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In *Shape Modeling and Applications, 1997. Proceedings., 1997 International Conference on*, pages 130–139, Mar 1997.

[114] Michel Neuhaus and Horst Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[115] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *in Proc. Conf. on Computer Vision and Pattern Recognition*, pages 63–69, 1992.

[116] R. Ohbuchi, K. Osada, T. Furuya, and T. Banno. Salient local visual features for shape-based 3d model retrieval. In *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on*, pages 93–102, June 2008.

[117] Clark F. Olson. Efficient pose clustering using a randomized algorithm. *Int. J. Comput. Vision*, 23(2):131–147, June 1997.

[118] C. Olsson, O. Enqvist, and F. Kahl. A polynomial-time bound for matching and registration with outliers. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[119] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3d models with shape distributions. In *Proceedings of the International Conference on Shape Modeling & Applications*, SMI '01, pages 154–, Washington, DC, USA, 2001. IEEE Computer Society.

[120] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, October 2002.

[121] P. Papadakis, I. Pratikakis, T. Theoharis, G. Passalis, and S. Perantonis. 3d object retrieval using an efficient and compact hybrid shape descriptor. In *IN EUROGRAPHICS 2008 WORKSHOP ON 3D OBJECT RETRIEVAL*, 2008.

[122] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[123] G. Papaioannou, E. Karabassi, and T. Theoharis. Virtual archaeologist: Assembling the past. *IEEE Computer Graphics and Applications*, 21:53–59, 2001.

[124] G. Papaioannou and E. A. Karabassi. On the automatic assemblage of arbitrary broken solid artefacts. *Image and Vision Computing*, 21(5):401–412, 2003.

[125] C. Papaodysseus, T. Panagopoulos, M. Exarhos, C. Triantafillou, D. Fragoulis, and C. Doumas. Contour-shape based reconstruction of fragmented, 1600 bc wall paintings. *Signal Processing, IEEE Transactions on*, 50(6):1277–1288, June 2002.

[126] Panos M. Pardalos, Franz Rendl, and Henry Wolkowicz. The quadratic assignment problem: A survey and recent developments. In *In Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, volume 16 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–42. American Mathematical Society, 1994.

[127] RasmusR. Paulsen and KlausB. Hilger. Shape modelling using markov random field restoration of point correspondences. In Chris Taylor and J.Alison Noble, editors, *Information Processing in Medical Imaging*, volume 2732 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2003.

[128] Mark Pauly, Niloy J. Mitra, Joachim Giesen, Markus Gross, and Leonidas J. Guibas. Example-based 3d scan completion. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[129] Mark Pauly, Niloy J. Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J. Guibas. Discovering structural regularity in 3d geometry. *ACM Trans. Graph.*, 27(3):43:1–43:11, August 2008.

[130] Yuri Pekelny and Craig Gotsman. Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum*, 27(2):399–408, 2008.

[131] Kari Pulli. Multiview registration for large data sets. In *Proceedings of the 2Nd International Conference on 3-D Digital Imaging and Modeling*, 3DIM'99, pages 160–168, Washington, DC, USA, 1999. IEEE Computer Society.

[132] Kari Pulli and Linda G. Shapiro. Surface reconstruction and display from range and color data. *Graphical Models*, 62(3):165 – 201, 2000.

[133] Georges Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus Acad. Sciences*, 222:847–849, 1946.

[134] T. Rohlfing. Image similarity and tissue overlaps as surrogates for image registration accuracy: Widely used but unreliable. *Medical Imaging, IEEE Transactions on*, 31(2):153–163, Feb 2012.

[135] K. Rohr and S. Worz. An extension of thin-plate splines for image registration with radial basis functions. In *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*, pages 442–445, May 2012.

[136] Guodong Rong and Tiow-Seng Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06, pages 109–116, New York, NY, USA, 2006. ACM.

[137] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[138] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, pages 1848–1853, Piscataway, NJ, USA, 2009. IEEE Press.

[139] R.B. Rusu, N. Blodow, Z.C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391, Sept 2008.

[140] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162, Oct 2010.

[141] Farzin Mokhtarian Sadegh, Sadegh Abbasi, and Josef Kittler. Robust and efficient shape indexing through curvature scale space. In *In Proceedings of British Machine Vision Conference*, pages 53–62, 1996.

[142] M. S. Sagiroglu and A. Ercil. A texture based matching approach for automated assembly of puzzles. *Pattern Recognition, International Conference on*, 3:1036–1041, 2006.

[143] Y. Sahillioglu and Y. Yemez. Coarse-to-fine combinatorial matching for dense isometric shape correspondence. *Computer Graphics Forum*, 30(5):1461–1470, 2011.

[144] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.

[145] Bernhard Schölkopf, Florian Steinke, and Volker Blanz. Object correspondence as a machine learning problem. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 776–783, New York, NY, USA, 2005. ACM.

[146] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):550–571, May 2004.

[147] Andrei Sharf, Dan A. Alcantara, Thomas Lewiner, Chen Greif, Alla Sheffer, Nina Amenta, and Daniel Cohen-Or. Space-time surface reconstruction using incompressible flow. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 110:1–110:10, New York, NY, USA, 2008. ACM.

[148] H. Shin, C. Doumas, T. Funkhouser, S. Rusinkiewicz, K. Steiglitz, A. Vlachopoulos, and T. Weyrich. Analyzing fracture patterns in theran wall paintings. In *Proceedings of the 11th International conference on Virtual Reality, Archaeology and Cultural Heritage*, VAST'10, pages 71–78, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[149] Dror Sholomon, Omid David, and Nathan S. Netanyahu. A genetic algorithm-based solver for very large jigsaw puzzles. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 1767–1774, Washington, DC, USA, 2013. IEEE Computer Society.

[150] Kaleem Siddiqi and Stephen Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[151] David A. Simon. *Fast and Accurate Shape-based Registration*. PhD thesis, Pittsburgh, PA, USA, 1996. AAI9838226.

[152] Chandan Singh and Pooja. Local and global features based image retrieval system using orthogonal radial moments. *Optics and Lasers in Engineering*, 50(5):655 – 667, 2012.

[153] Ivan Sipiran and Benjamin Bustos. A robust 3d interest points detector based on harris operator. In *Proceedings of the 3rd Eurographics Conference on 3D Object Retrieval*, EG 3DOR'10, pages 7–14, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[154] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1470–, Washington, DC, USA, 2003. IEEE Computer Society.

[155] Olga Sorkine. Differential Representations for Mesh Processing. *Computer Graphics Forum*, 25(4):789–807, December 2006.

[156] A. Sotiras and N. Paragios. Discrete symmetric image registration. In *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*, pages 342–345, May 2012.

[157] George Stockman. Object recognition and localization via pose clustering. *Comput. Vision Graph. Image Process.*, 40(3):361–387, December 1987.

[158] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, August 2004.

[159] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[160] Jian Sun, Xiaobai Chen, and Thomas Funkhouser. Fuzzy geodesics and consistent sparse correspondences for deformable shapes. *Computer Graphics Forum (Symposium on Geometry Processing)*, 29(5), July 2010.

[161] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1383–1392, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[162] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Proceedings of the Shape Modeling International 2003*, SMI '03, pages 130–, Washington, DC, USA, 2003. IEEE Computer Society.

[163] Jochen Sussmuth, Marco Winter, and Günther Greiner. Reconstructing animated meshes from time-varying point clouds. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1469–1476, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[164] Michael Reed Teague. Image analysis via the general theory of moments. *J. Opt. Soc. Am.*, 70(8):920–930, Aug 1980.

[165] Art Tevs, Martin Bokeloh, Michael Wand, Andreas Schilling, and Hans-Peter Seidel. Isometric registration of ambiguous and partial data. In *CVPR*, pages 1185–1192. IEEE, 2009.

[166] C. Toler-Franklin, B. J. Brown, T. Weyrich, T. Funkhouser, and S. Rusinkiewicz. Multi-feature matching of fresco fragments. *ACM Trans. Graph.*, 29(6):185:1–185:12, December 2010.

[167] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3d data description. In *Proceedings of the ACM Workshop on 3D Object Retrieval*, 3DOR '10, pages 57–62, New York, NY, USA, 2010. ACM.

[168] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ECCV'10, pages 356–369, Berlin, Heidelberg, 2010. Springer-Verlag.

[169] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 311–318, New York, NY, USA, 1994. ACM.

[170] Oliver van Kaick, Ghassan Hamarneh, Hao Zhang, and Paul Wighton. Contour correspondence via ant colony optimization. In *Proc. 15th Pacific Conference on Computer Graphics and Applications (PG'2007)*, pages 271–280, 2007.

[171] Oliver van Kaick, Andrea Tagliasacchi, Oana Sidi, Hao Zhang, Daniel Cohen-Or, Lior Wolf, , and Ghassan Hamarneh. Prior knowledge for part correspondence. *Computer Graphics Forum (Proc. Eurographics)*, 30(2):553–562, 2011.

[172] D.V. Vranic. An improvement of rotation invariant 3d-shape based on functions on concentric spheres. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III–757–60 vol.2, Sept 2003.

[173] D. V. Vranić, D. Saupe, and J. Richter. Tools for 3d-object retrieval: Karhunen-loeve transform and spherical harmonics. In *IEEE MMSP 2001*, pages 293–298, 2001.

[174] Michael W. Walker, Lejun Shao, and Richard A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Underst.*, 54(3):358–367, October 1991.

[175] Michael Wand, Philipp Jenke, Qixing Huang, Martin Bokeloh, Leonidas Guibas, and Andreas Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 49–58, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[176] C. Wang. *Determining Molecular Conformation from Distance Or Density Data*. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2000.

[177] AaronD. Ward and Ghassan Hamarneh. Statistical shape modeling using mdl incorporating shape, appearance, and expert knowledge. In Nicholas Ayache, Sébastien Ourselin, and Anthony Maeder, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, volume 4791 of *Lecture Notes in Computer Science*, pages 278–285. Springer Berlin Heidelberg, 2007.

[178] Chia-Hung Wei, Yue Li, Wing-Yin Chau, and Chang-Tsun Li. Trademark image retrieval using synthetic features for describing global shape and interior structure. *Pattern Recognition*, 42(3):386 – 394, 2009.

[179] S. Weik. Registration of 3-d partial surface models using luminance and depth information. In *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, NRC '97, pages 93–, Washington, DC, USA, 1997. IEEE Computer Society.

[180] A. R. Willis. *Stochastic 3d geometric models for classification, deformation, and estimation*. PhD thesis, Providence, RI, USA, 2004. AAI3134376.

[181] Andrew Willis and David B. Cooper. Alignment of multiple non-overlapping axially symmetric 3d datasets. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4 - Volume 04*, ICPR '04, pages 96–99, Washington, DC, USA, 2004. IEEE Computer Society.

[182] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2987–2992, Dec 2011.

[183] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan. Solving jigsaw puzzles by computer. *Ann. Oper. Res.*, 12(1-4):51–64, February 1988.

[184] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Comput. Sci. Eng.*, 4(4):10–21, October 1997.

[185] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. *ACM Trans. Graph.*, 29(6):184:1–184:10, December 2010.

[186] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. Surface feature detection and description with applications to mesh matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 373–380, June 2009.

[187] Titus Zaharia and Francoise J. Preteux. 3d-shape-based retrieval within the mpeg-7 framework, 2001.

[188] Dimitrios Zarpalas, Petros Daras, Apostolos Axenopoulos, Dimitrios Tzovaras, and Michael G. Strintzis. 3d model search and retrieval using the spherical trace transform. *EURASIP J. Appl. Signal Process.*, 2007(1):207–207, January 2007.

[189] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[190] Von F. Zernike. Beugungstheorie des schneidenver-fahrens und seiner verbesserten form, der phasenkontrastmethode. *Physica*, 1(7-12):689–704, May 1934.

[191] Gang Zhang, Zong-min Ma, Lian-qiang Niu, and Chun-ming Zhang. Modified fourier descriptor for shape feature extraction. *Journal of Central South University*, 19(2):488–495, 2012.

[192] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1431–1439, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[193] Yu-Xiang Zhao, Mu-Chun Su, Zhong-Lie Chou, and Jonathan Lee. A puzzle solver and its application in speech descrambling. In *Proceedings of the 2007 Annual Conference on International Conference on Computer Engineering and Applications*, CEA'07, pages 171–176, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[194] Qian Zheng, Andrei Sharf, Andrea Tagliasacchi, Baoquan Chen, Hao Zhang, Alla Sheffer, and Daniel Cohen-Or. Consensus skeleton for non-rigid space-time registration. *Computer Graphcis Forum (Special Issue of Eurographics)*, 29(2):635–644, 2010.

[195] Xiangtao Zheng, Xiaoqiang Lu, and Yuan Yuan. Image jigsaw puzzles with a self-correcting solver. In *Virtual Reality and Visualization (ICVRV), 2013 International Conference on*, pages 112–118, Sept 2013.

[196] Yefeng Zheng and D. Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):643–649, April 2006.

[197] L. Zhu, Z. Zhou, and D. Hu. Globally consistent reconstruction of ripped-up documents. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30:1–13, January 2008.

[198] G. Üçoluk and I. H. Toroslu. Automatic reconstruction of broken 3-d surface objects. *Computers and Graphics*, 23(4):573–582, 1999.

APPENDIX A

6 DOF One-to-One results

A.1 Brick dataset

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d
1.3d		0.0247s 8	0.0273s 6	0.0415s 7	0.0440s	0.0230s
2.3d			0.0141s	0.0222s	0.0238s 9	0.0126s
3.3d				0.0196s	0.0210s	0.0104s
4.3d					0.0318s 29	0.0156s 9
5.3d						0.0172s 15
6.3d						

Table A.1: *One-to-one search results for the brick dataset using the 'Fast' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d
1.3d		0.0458s 30	0.0486s 16	0.0737s 29	0.0801s 4	0.0397s 4
2.3d			0.0397s 5	0.0590s 5	0.0632s 19	0.0332s 5
3.3d				0.0685s 5	0.0665s 6	0.0336s 6
4.3d					0.2647s 62	0.0351s 16
5.3d						0.0609s 33
6.3d						

Table A.2: *One-to-one search results for the brick dataset using the 'Medium' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d
1.3d		0.1109s 39	0.1367s 24	0.2081s 36	0.2308s 5	0.1097s 5
2.3d			0.1546s 6	0.2561s 6	0.2561s 20	0.1304s 7
3.3d				0.2959s 8	0.2935s 7	0.1423s 7
4.3d					0.4410s 73	0.1167s 16
5.3d						0.1932s 37
6.3d						

Table A.3: *One-to-one search results for the brick dataset using the 'Precise' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

A.2 Venus dataset

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d
1.3d		0.0419s 5	0.0135s 8	0.0223s	0.0078s	0.0179s	0.0089s
2.3d			0.0131s	0.0218s	0.0081s	0.0170s	0.0091s
3.3d				0.0137s	0.0050s	0.0105s	0.0060s 4
4.3d					0.0055s 8	0.0120s 13	0.0062s 5
5.3d						0.0081s 4	0.0042s
6.3d							0.0068s 4
7.3d							

Table A.4: *One-to-one search results for the venus dataset using the 'Fast' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d
1.3d		0.0730s 13	0.0269s 17	0.0441s 5	0.0148s 5	0.0336s 4	0.0173s 4
2.3d			0.0266s 4	0.0443s 5	0.0152s 4	0.0345s 5	0.0169s
3.3d				0.0360s 5	0.0127s 4	0.0307s 5	0.0142s 6
4.3d					0.0148s 18	0.0314s 30	0.0142s 8
5.3d						0.0380s 7	0.0153s 5
6.3d							0.0164s 5
7.3d							

Table A.5: *One-to-one search results for the venus dataset using the 'Medium' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d
1.3d		0.1808s 15	0.0760s 24	0.1243s 5	0.0421s 6	0.0979s 5	0.0465s 4
2.3d			0.0777s 4	0.1302s 5	0.0433s 4	0.1064s 7	0.0477s 4
3.3d				0.1339s 6	0.0467s 5	0.1217s 7	0.0487s 8
4.3d					0.0512s 21	0.1320s 38	0.0478s 9
5.3d						0.1745s 8	0.0644s 6
6.3d							0.0623s 6
7.3d							

Table A.6: *One-to-one search results for the venus dataset using the 'Precise' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

A.3 Cake dataset

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d	8.3d	9.3d	10.3d	11.3d
1.3d		0.0353s	0.0424s 32	0.0437s	0.0327s 28	0.0390s	0.0366s 4	0.0415s 4	0.0405s	0.0285s	0.0298s
2.3d			0.0381s 30	0.0409s	0.0284s	0.0356s	0.0343s	0.0389s	0.0375s 16	0.0268s 25	0.0278s 21
3.3d				0.0484s	0.0340s	0.0425s	0.0405s	0.0469s	0.0442s	0.0306s	0.0413s 43
4.3d					0.0376s 10	0.0459s 4	0.0432s	0.0483s	0.0474s	0.0329s	0.0355s
5.3d						0.0358s 17	0.0343s	0.0387s	0.0366s	0.0267s	0.0274s
6.3d							0.0442s 42	0.0470s	0.0459s	0.0304s	0.0334s
7.3d								0.0460s 37	0.0410s	0.0302s	0.0313s
8.3d									0.0423s 24	0.0296s 6	0.0317s 4
9.3d										0.0310s 14	0.0337s 18
10.3d											0.0266s
11.3d											

Table A.7: *One-to-one search results for the cake dataset using the 'Fast' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d	8.3d	9.3d	10.3d	11.3d
1.3d		0.0734s 5	0.1805s 56	0.0950s 34	0.2115s 57	0.0872s 6	0.0846s 6	0.0988s 7	0.0872s 6	0.0619s 6	0.0672s 6
2.3d			0.1229s 44	0.0875s 5	0.0644s 7	0.0802s 6	0.0775s 6	0.0863s 6	0.0835s 35	0.0718s 44	0.0713s 35
3.3d				0.1007s 5	0.0750s 6	0.0932s 5	0.0887s 6	0.1009s 6	0.0920s 5	0.0669s 5	0.2010s 75
4.3d					0.0720s 24	0.0919s 7	0.0873s 6	0.0975s 5	0.0935s 6	0.0647s 5	0.0701s 5
5.3d						0.0839s 34	0.0799s 7	0.0885s 6	0.0828s 5	0.0610s 6	0.0635s 6
6.3d							0.2021s 72	0.1065s 6	0.0978s 5	0.0689s 5	0.0760s 6
7.3d								0.2921s 72	0.0916s 5	0.0657s 5	0.0694s 5
8.3d									0.1035s 55	0.0628s 11	0.0705s 6
9.3d										0.0767s 31	0.0762s 28
10.3d											0.0649s 5
11.3d											

Table A.8: *One-to-one search results for the cake dataset using the 'Medium' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d	8.3d	9.3d	10.3d	11.3d
1.3d		0.2496s 7	0.5521s 64	0.3257s 42	0.6241s 59	0.3267s 8	0.3244s 8	0.3773s 7	0.3056s 7	0.2239s 8	0.2634s 7
2.3d			0.3324s 47	0.2945s 6	0.2267s 7	0.2738s 8	0.2725s 6	0.2961s 7	0.2715s 38	0.1917s 46	0.2327s 37
3.3d				0.3261s 7	0.2747s 6	0.3267s 8	0.3115s 7	0.3495s 7	0.2962s 7	0.2200s 6	0.5481s 80
4.3d					0.2133s 30	0.2896s 11	0.2780s 7	0.3090s 7	0.2951s 6	0.2010s 7	0.2158s 7
5.3d						0.3281s 37	0.2979s 8	0.3338s 8	0.2845s 6	0.2251s 8	0.2450s 7
6.3d							0.5152s 81	0.3803s 7	0.3070s 7	0.2354s 6	0.2644s 7
7.3d								0.9731s 79	0.3298s 6	0.2248s 7	0.2604s 8
8.3d									0.3265s 60	0.2201s 13	0.2580s 7
9.3d										0.2276s 37	0.2292s 30
10.3d											0.2244s 7
11.3d											

Table A.9: *One-to-one search results for the cake dataset using the 'Precise' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

A.4 Sculpture dataset

	1.3d	10.3d	11.3d	12.3d	13.3d	14.3d	15.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d	8.3d	9.3d
1.3d		0.0080s	0.0043s 10	0.0022s	0.0026s	0.0046s 5	0.0034s	0.0067s 21	0.0057s 11	0.0079s	0.0064s 4	0.0072s	0.0067s	0.0066s	0.0080s
10.3d			0.0038s 4	0.0024s	0.0031s 4	0.0047s	0.0035s	0.0064s	0.0058s	0.0082s 4	0.0064s	0.0088s 32	0.0067s 5	0.0074s	0.0089s
11.3d				0.0018s	0.0024s	0.0046s	0.0034s	0.0056s 4	0.0052s 11	0.0071s	0.0060s	0.0065s	0.0061s	0.0064s	0.0075s
12.3d					0.0020s	0.0032s	0.0027s	0.0044s	0.0040s	0.0060s	0.0050s	0.0051s	0.0045s	0.0050s	0.0060s
13.3d						0.0038s	0.0030s	0.0047s	0.0044s	0.0062s 5	0.0052s	0.0060s 4	0.0054s 5	0.0056s	0.0065s
14.3d							0.0032s	0.0055s	0.0051s	0.0073s	0.0060s 11	0.0067s	0.0060s	0.0065s	0.0075s
15.3d								0.0053s	0.0047s	0.0070s	0.0058s	0.0065s	0.0057s	0.0063s	0.0086s 28
2.3d									0.0069s 15	0.0086s	0.0078s 17	0.0073s	0.0068s	0.0074s	0.0087s
3.3d										0.0085s 4	0.0069s	0.0076s	0.0069s	0.0075s	0.0091s
4.3d											0.0080s	0.0162s 40	0.0087s 18	0.0081s	0.0098s
5.3d												0.0077s	0.0073s 12	0.0074s	0.0090s
6.3d													0.0078s	0.0077s	0.0089s 4
7.3d														0.0081s 16	0.0091s 6
8.3d															0.0103s 21
9.3d															

Table A.10: *One-to-one search results for the sculpture dataset using the 'Fast' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	10.3d	11.3d	12.3d	13.3d	14.3d	15.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d	8.3d	9.3d
1.3d		0.0208s 5	0.0119s 15	0.0066s 4	0.0087s 5	0.0130s 8	0.0113s 5	0.0539s 32	0.0174s 14	0.0227s 5	0.0181s 5	0.0205s 5	0.0177s 4	0.0193s 5	0.0224s 4
10.3d			0.0149s 5	0.0083s 6	0.0110s 7	0.0127s 4	0.0138s 5	0.0215s 6	0.0191s 6	0.0260s 5	0.0217s 6	0.0824s 52	0.0193s 5	0.0223s 5	0.0267s 6
11.3d				0.0079s 5	0.0112s 6	0.0121s 5	0.0137s 5	0.0201s 6	0.0184s 22	0.0238s 6	0.0207s 5	0.0221s 6	0.0174s 5	0.0209s 5	0.0245s 5
12.3d					0.0217s 7	0.0140s 7	0.0253s 8	0.0208s 7	0.0267s 9	0.0332s 8	0.0213s 6	0.0307s 9	0.0196s 6	0.0250s 6	0.0268s 7
13.3d						0.0126s 6	0.0139s 6	0.0228s 8	0.0233s 6	0.0341s 7	0.0222s 7	0.0344s 9	0.0191s 5	0.0238s 6	0.0262s 10
14.3d							0.0107s 4	0.0157s 5	0.0150s 4	0.0215s 5	0.0200s 18	0.0197s 7	0.0170s 4	0.0186s 4	0.0217s 4
15.3d								0.0185s 5	0.0199s 5	0.0270s 6	0.0200s 6	0.0284s 6	0.0188s 5	0.0225s 6	0.1189s 44
2.3d									0.0313s 22	0.0234s 5	0.0305s 19	0.0215s 4	0.0179s 4	0.0208s 5	0.0243s 4
3.3d										0.0248s 6	0.0201s 5	0.0232s 5	0.0183s 4	0.0205s 4	0.0252s 4
4.3d											0.0207s 5	0.5489s 62	0.0271s 34	0.0220s 6	0.0255s 4
5.3d												0.0214s 5	0.0235s 24	0.0202s 4	0.0237s 4
6.3d													0.0188s 4	0.0211s 5	0.0254s 4
7.3d														0.0342s 25	0.0239s 9
8.3d															0.0314s 28
9.3d															

Table A.11: *One-to-one search results for the sculpture dataset using the 'Medium' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	10.3d	11.3d	12.3d	13.3d	14.3d	15.3d	2.3d	3.3d	4.3d	5.3d	6.3d	7.3d	8.3d	9.3d
1.3d		0.0731s 6	0.0464s 18	0.0234s 5	0.0314s 5	0.0477s 9	0.0458s 5	0.1022s 32	0.0643s 14	0.0891s 6	0.0724s 6	0.0825s 5	0.0698s 4	0.0781s 5	0.0903s 5
10.3d			0.0674s 7	0.0343s 7	0.0435s 9	0.0496s 5	0.0567s 7	0.0920s 8	0.0790s 7	0.1143s 8	0.0873s 8	0.1980s 53	0.0764s 6	0.0921s 6	0.1155s 7
11.3d				0.0326s 6	0.0476s 8	0.0493s 6	0.0600s 6	0.0933s 8	0.0752s 22	0.1087s 7	0.0861s 7	0.1032s 8	0.0721s 6	0.0892s 6	0.1042s 6
12.3d					0.1147s 8	0.0632s 9	0.1400s 9	0.1030s 10	0.1465s 9	0.1719s 11	0.1101s 8	0.1645s 10	0.0950s 9	0.1428s 7	0.1510s 8
13.3d						0.0564s 6	0.0755s 7	0.1288s 9	0.1090s 8	0.1654s 9	0.1034s 8	0.1681s 11	0.0907s 7	0.1098s 7	0.1235s 11
14.3d							0.0436s 4	0.0617s 6	0.0616s 5	0.0852s 6	0.0736s 20	0.0822s 7	0.0657s 5	0.0740s 6	0.0865s 5
15.3d								0.0823s 6	0.0956s 7	0.1251s 7	0.0876s 6	0.1466s 7	0.0801s 6	0.1028s 8	0.2287s 45
2.3d									0.0787s 24	0.0913s 8	0.0951s 19	0.0858s 5	0.0674s 5	0.0795s 8	0.0919s 6
3.3d										0.0948s 6	0.0758s 5	0.0902s 7	0.0685s 5	0.0794s 5	0.0945s 6
4.3d											0.0776s 5	0.9962s 64	0.0842s 34	0.0813s 7	0.0935s 6
5.3d												0.0829s 7	0.0749s 25	0.0788s 5	0.0916s 5
6.3d													0.0715s 6	0.0862s 5	0.0999s 6
7.3d														0.0856s 26	0.0901s 9
8.3d															0.0996s 29
9.3d															

Table A.12: *One-to-one search results for the sculpture dataset using the 'Precise' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

A.5 Gargoyle dataset

	1.3d	10.3d	11.3d	12.3d	13.3d	14.3d	15.3d	16.3d	17.3d	18.3d	19.3d	20.3d	21.3d	22.3d	23.3d	24.3d	25.3d	2A.3d	2B.3d	2C.3d	3.3d	4.3d	5A.3d	5B.3d	6A.3d	6B.3d	7.3d	8.3d	9.3d
1.3d		0.0248s	0.0265s	0.0051s	0.0040s	0.0250s	0.0128s	0.0086s	0.0086s	0.0150s	0.0157s	0.0039s	0.0009s	0.0047s	0.0020s	0.0024s	0.0019s	0.0138s 14	0.0110s	0.0129s	0.0286s 21	0.0418s	0.0768s	0.0074s	0.0144s	0.0183s	0.0256s	0.0126s	0.0204s
10.3d			0.0259s 15	0.0054s	0.0043s	0.0250s	0.0138s 11	0.0092s	0.0092s	0.0162s	0.0172s	0.0041s	0.0010s	0.0052s	0.0021s	0.0026s	0.0022s	0.0149s	0.0117s	0.0136s	0.0285s	0.0429s	0.0769s 6	0.0080s	0.0152s	0.0190s	0.0263s 6	0.0134s	0.0216s 20
11.3d				0.0059s	0.0049s	0.0279s 4	0.0150s 9	0.0104s 8	0.0102s	0.0177s	0.0192s	0.0046s	0.0012s	0.0057s	0.0024s	0.0030s	0.0024s	0.0159s	0.0128s	0.0158s	0.0324s	0.0462s	0.0848s	0.0089s	0.0169s	0.0211s	0.0291s 7	0.0148s	0.0233s
12.3d					0.0021s	0.0138s	0.0075s	0.0051s	0.0047s	0.0087s 5	0.0097s	0.0020s	0.0006s	0.0026s	0.0011s	0.0011s	0.0009s	0.0082s	0.0065s	0.0075s	0.0151s	0.0194s	0.0336s	0.0043s	0.0078s	0.0102s	0.0141s	0.0071s	0.0115s
13.3d						0.0127s	0.0067s	0.0045s	0.0042s	0.0081s 4	0.0085s	0.0018s	0.0005s	0.0024s	0.0009s	0.0010s	0.0007s	0.0074s	0.0058s	0.0068s	0.0138s	0.0176s	0.0308s	0.0037s	0.0073s	0.0097s	0.0132s	0.0065s	0.0105s
14.3d							0.0147s 9	0.0099s	0.0098s	0.0172s	0.0190s 22	0.0045s	0.0011s	0.0054s	0.0023s	0.0028s	0.0023s	0.0157s	0.0125s	0.0145s	0.0309s	0.0465s	0.0855s	0.0086s	0.0162s	0.0213s	0.0290s	0.0144s 6	0.0238s 24
15.3d								0.0077s 12	0.0073s	0.0131s	0.0141s	0.0033s	0.0008s	0.0041s	0.0017s	0.0019s	0.0015s	0.0123s	0.0098s	0.0111s	0.0233s	0.0333s	0.0601s	0.0065s	0.0122s	0.0156s	0.0223s	0.0111s	0.0177s
16.3d									0.0059s 6	0.0109s	0.0116s	0.0025s	0.0007s	0.0033s	0.0014s	0.0015s	0.0011s	0.0100s	0.0081s	0.0091s	0.0189s	0.0256s	0.0454s	0.0052s	0.0098s	0.0126s	0.0174s	0.0090s	0.0143s
17.3d										0.0110s 22	0.0115s	0.0025s	0.0007s	0.0033s	0.0013s	0.0014s	0.0011s	0.0100s	0.0080s	0.0093s	0.0188s	0.0251s	0.0431s 4	0.0051s	0.0096s	0.0124s	0.0172s	0.0088s	0.0140s
18.3d											0.0140s 19	0.0034s	0.0007s	0.0040s	0.0017s	0.0020s	0.0015s	0.0119s	0.0095s	0.0111s	0.0235s	0.0340s	0.0618s	0.0066s	0.0122s	0.0165s	0.0223s	0.0109s	0.0176s
19.3d												0.0034s 6	0.0008s	0.0041s	0.0018s	0.0021s	0.0016s	0.0120s	0.0098s	0.0110s	0.0239s	0.0348s	0.0641s	0.0066s	0.0124s	0.0156s	0.0218s	0.0111s	0.0179s
20.3d													0.0005s	0.0025s	0.0009s	0.0009s	0.0007s	0.0075s	0.0059s	0.0068s	0.0137s	0.0174s	0.0304s	0.0037s	0.0069s	0.0092s	0.0128s	0.0065s	0.0104s
21.3d														0.0015s	0.0006s	0.0006s	0.0004s	0.0046s	0.0036s	0.0043s	0.0085s	0.0104s	0.0182s	0.0023s	0.0042s	0.0056s	0.0080s	0.0040s	0.0065s
22.3d															0.0010s	0.0010s	0.0008s	0.0076s	0.0061s	0.0070s	0.0145s	0.0192s	0.0340s	0.0041s	0.0074s	0.0097s	0.0138s	0.0068s	0.0111s
23.3d																0.0007s	0.0005s	0.0055s	0.0044s	0.0051s	0.0102s	0.0129s	0.0228s	0.0028s	0.0052s	0.0072s	0.0103s	0.0055s	0.0085s
24.3d																	0.0006s	0.0061s	0.0046s	0.0054s 4	0.0109s	0.0134s 4	0.0233s	0.0029s	0.0053s	0.0069s	0.0100s 4	0.0051s	0.0081s
25.3d																		0.0051s	0.0039s	0.0047s	0.0095s	0.0117s	0.0204s	0.0025s	0.0047s	0.0062s	0.0089s	0.0043s	0.0076s
2A.3d																			0.0091s 7	0.0102s 7	0.0225s 4	0.0326s	0.0589s	0.0059s	0.0110s	0.0142s	0.0200s	0.0099s	0.0161s
2B.3d																				0.0104s 19	0.0217s	0.0323s	0.0575s	0.0057s	0.0111s	0.0141s	0.0203s	0.0099s	0.0160s
2C.3d																					0.0218s 8	0.0318s	0.0572s	0.0060s	0.0114s	0.0143s 4	0.0198s	0.0101s	0.0161s
3.3d																						0.0510s	0.0947s	0.0097s	0.0179s	0.0226s	0.0305s	0.0159s	0.0252s
4.3d																							0.0776s 17	0.0102s	0.0188s	0.0238s	0.0323s 5	0.0169s	0.0272s
5A.3d																								0.0152s	0.0274s	0.0334s 4	0.0448s	0.0227s	0.0360s
5B.3d																									0.0094s 8	0.0125s	0.0165s	0.0087s	0.0134s
6A.3d																										0.0161s 9	0.0223s 13	0.0113s	0.0175s
6B.3d																											0.0250s 23	0.0130s 7	0.0203s
7.3d																												0.0143s	0.0228s
8.3d																													0.0165s 5
9.3d																													

Table A.13: *One-to-one search results for the gargoyle dataset using the 'Fast' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	10.3d	11.3d	12.3d	13.3d	14.3d	15.3d	16.3d	17.3d	18.3d	19.3d	20.3d	21.3d	22.3d	23.3d	24.3d	25.3d	2A.3d	2B.3d	2C.3d	3.3d	4.3d	5A.3d	5B.3d	6A.3d	6B.3d	7.3d	8.3d	9.3d
1.3d		0.0548s 4	0.0586s 4	0.0111s	0.0089s 4	0.0596s 4	0.0303s 4	0.0195s 4	0.0196s 4	0.0347s 5	0.0369s 4	0.0090s 5	0.0019s	0.0103s	0.0045s	0.0045s	0.0033s	0.0344s 30	0.0252s	0.0298s	0.0923s 45	0.0840s 4	0.1507s 4	0.0166s	0.0318s 4	0.0427s 4	0.0591s 4	0.0296s 4	0.0482s 4
10.3d			0.0593s 32	0.0119s 4	0.0091s 4	0.0587s 4	0.0321s 28	0.0204s 4	0.0197s 4	0.0364s 4	0.0392s 4	0.0087s 4	0.0020s	0.0108s 4	0.0046s 4	0.0049s 4	0.0035s 4	0.0341s 4	0.0260s 4	0.0307s 4	0.0638s 4	0.0837s 4	0.1478s 16	0.0176s 4	0.0326s 4	0.0434s 4	0.0595s 11	0.0307s 4	0.0570s 34
11.3d				0.0123s 4	0.0094s 4	0.0611s 5	0.0315s 19	0.0213s 23	0.0213s 4	0.0377s 4	0.0407s 4	0.0091s 4	0.0020s	0.0113s 4	0.0047s 4	0.0051s 4	0.0048s 4	0.0344s 4	0.0267s 4	0.0320s 4	0.0677s 4	0.0881s 14	0.1571s 4	0.0182s 4	0.0352s 4	0.0459s 14	0.0624s 4	0.0312s 4	0.0502s 4
12.3d					0.0093s 4	0.0497s 5	0.0268s 5	0.0199s 5	0.0193s 5	0.0340s 9	0.0321s 5	0.0096s 6	0.0015s	0.0104s 5	0.0040s 4	0.0040s 4	0.0026s 4	0.0289s 5	0.0230s 5	0.0280s 6	0.0524s 6	0.0586s 5	0.1188s 5	0.0167s 6	0.0287s 6	0.0373s 5	0.0473s 5	0.0259s 5	0.0393s 5
13.3d						0.0493s 8	0.0285s 5	0.0211s 7	0.0226s 6	0.0333s 6	0.0318s 9	0.0110s 6	0.0016s 4	0.0117s 6	0.0046s 5	0.0047s 4	0.0025s	0.0294s 5	0.0237s 6	0.0301s 6	0.0531s 6	0.0643s 6	0.1005s 6	0.0168s 5	0.0306s 6	0.0376s 6	0.0486s 6	0.0264s 6	0.0422s 6
14.3d							0.0337s 26	0.0219s	0.0213s 7	0.0393s 4	0.0451s 40	0.0094s 4	0.0020s	0.0114s 6	0.0048s 5	0.0051s 4	0.0039s	0.0365s 4	0.0280s 4	0.0332s 4	0.0719s 4	0.0934s 4	0.1664s 4	0.0188s	0.0355s 6	0.0482s 4	0.0652s 20	0.0321s 61	0.0743s 4
15.3d								0.0222s 27	0.0181s 4	0.0335s 5	0.0351s 4	0.0081s 5	0.0015s	0.0099s 4	0.0041s 4	0.0041s 4	0.0029s	0.0318s 4	0.0243s 4	0.0287s 4	0.0578s 4	0.0738s 4	0.1321s 4	0.0160s 4	0.0303s 4	0.0399s 4	0.0551s 4	0.0281s 4	0.0441s 4
16.3d									0.0169s 13	0.0307s 5	0.0329s 5	0.0083s 4	0.0015s 4	0.0096s 4	0.0038s 4	0.0037s 4	0.0027s	0.0297s 4	0.0228s 4	0.0270s 5	0.0540s 5	0.0640s 4	0.1113s 4	0.0152s 5	0.0278s 4	0.0360s 5	0.0505s 5	0.0261s 6	0.0398s 6
17.3d										0.0607s 35	0.0351s 5	0.0091s 4	0.0016s 4	0.0099s 4	0.0040s 4	0.0038s 4	0.0026s	0.0312s 4	0.0247s 5	0.0292s 5	0.0548s 5	0.0702s 5	0.1134s 5	0.0159s 4	0.0304s 5	0.0389s 5	0.0519s 5	0.0278s 5	0.0422s 5
18.3d											0.0423s 44	0.0085s 4	0.0017s 4	0.0099s 4	0.0042s 4	0.0042s 4	0.0034s	0.0310s 4	0.0239s 5	0.0287s 5	0.0593s 5	0.0742s 4	0.1310s 5	0.0161s	0.0308s 4	0.0411s 4	0.0557s 4	0.0278s 4	0.0445s 4
19.3d												0.0081s 15	0.0016s 4	0.0099s 4	0.0040s 4	0.0043s 4	0.0034s	0.0312s 4	0.0236s 4	0.0281s 4	0.0633s 4	0.0754s 4	0.1348s 4	0.0155s 4	0.0301s 4	0.0397s 4	0.0548s 5	0.0273s 4	0.0444s 4
20.3d													0.0015s 5	0.0111s 5	0.0041s 4	0.0040s 4	0.0026s 4	0.0320s 4	0.0237s 5	0.0302s 6	0.0566s 6	0.0646s 6	0.1083s 6	0.0163s 6	0.0312s 5	0.0408s 7	0.0514s 7	0.0283s 6	0.0427s 6
21.3d														0.0087s 4	0.0045s 5	0.0040s 5	0.0023s 4	0.0236s 5	0.0189s 4	0.0207s 5	0.0387s 4	0.0440s 6	0.0755s 4	0.0125s 4	0.0213s 5	0.0278s 4	0.0368s 5	0.0203s 4	0.0309s 5
22.3d															0.0048s 4	0.0044s 4	0.0025s 4	0.0287s 5	0.0227s 6	0.0268s 5	0.0494s 5	0.0561s 5	0.0969s 6	0.0163s 6	0.0296s 6	0.0354s 5	0.0471s 6	0.0288s 5	0.0382s 5
23.3d																0.0058s 5	0.0029s 4	0.0287s 6	0.0240s 8	0.0287s 6	0.0490s 5	0.0547s 6	0.0938s 5	0.0185s 7	0.0287s 6	0.0363s 6	0.0454s 6	0.0271s 6	0.0393s 6
24.3d																	0.0034s 4	0.0359s 6	0.0275s 7	0.0346s 6	0.0540s 6	0.0604s 6	0.1132s 7	0.0224s 6	0.0357s 7	0.0389s 7	0.0516s 6	0.0300s 6	0.0460s 6
25.3d																		0.0236s 5	0.0190s 5	0.0223s 4	0.0399s 5	0.0485s 4	0.0793s 5	0.0121s 4	0.0222s 4	0.0288s 5	0.0379s 6	0.0209s 5	0.0314s 6
2A.3d																			0.0230s 26	0.0285s 14	0.0555s 9	0.0712s 4	0.1247s 7	0.0147s 4	0.0280s 4	0.0379s 4	0.0548s 4	0.0260s 4	0.0419s 4
2B.3d																				0.0477s 42	0.0542s 12	0.0706s 4	0.1224s 4	0.0146s 5	0.0284s 4	0.0374s 5	0.0513s 4	0.0262s 4	0.0426s 5
2C.3d																					0.0563s 24	0.0728s 8	0.1262s 4	0.0155s 4	0.0299s 4	0.0388s 4	0.0537s 4	0.0279s 4	0.0438s 5
3.3d																						0.0954s 5	0.1686s 4	0.0192s 4	0.0362s 4	0.0481s 4	0.0653s 4	0.0333s 4	0.0520s 4
4.3d																							0.1749s 46	0.0199s	0.0383s 5	0.0493s	0.0667s 11	0.0335s 4	0.0554s 4
5A.3d																								0.0256s 4	0.0465s 5	0.0600s 13	0.0813s 7	0.0412s 5	0.0656s 5
5B.3d																									0.0272s 24	0.0357s 6	0.0488s 4	0.0253s 4	0.0401s 4
6A.3d																										0.0393s 21	0.0598s 31	0.0288s 4	0.0436s 6
6B.3d																											0.1070s 54	0.0291s 15	0.0465s 4
7.3d																												0.0302s 4	0.0493s 4
8.3d																													0.0418s 17
9.3d																													

Table A.14: *One-to-one search results for the gargoyle dataset using the 'Medium' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

Table A.15: *One-to-one search results for the gargoyle dataset using the 'Precise' preset. Each cell in the matrix shows the total search time (up) and the number of matching keypoints (down).*

	1.3d	10.3d	11.3d	12.3d	13.3d	14.3d	15.3d	16.3d	17.3d	18.3d	19.3d	20.3d	21.3d	22.3d	23.3d	24.3d	25.3d	2A.3d	2B.3d	2C.3d	3.3d	4.3d	5A.3d	5B.3d	6A.3d	6B.3d	7.3d	8.3d	9.3d
1.3d		0.1753s 5	0.1939s 6	0.0388s 4	0.0295s 4	0.2699s 5	0.1088s 5	0.0658s 4	0.0649s 5	0.1239s 5	0.1285s 4	0.0209s 5	0.0329s 4	0.0357s 4	0.0107s 4	0.0105s 4	0.0065s 33	0.1698s 0.0863s 4	0.1058s 4	0.2606s 50	0.2628s 5	0.5325s 9	0.0560s 5	0.1129s 5	0.1477s 5	0.2063s 5	0.1693s 5	0.1631s 5
10.3d			0.2233s 36	0.0377s 4	0.0296s 4	0.2395s 5	0.1007s 5	0.0670s 4	0.0642s 5	0.1208s 5	0.1282s 4	0.0228s 5	0.0031s 4	0.0338s 4	0.0111s 4	0.0109s 4	0.0065s 5	0.1148s 0.0877s 4	0.1072s 5	0.2126s 5	0.2130s 5	0.2539s 19	0.5330s 4	0.0567s 4	0.1094s 4	0.1447s 4	0.1980s 15	0.1035s 4	0.1623s 37
11.3d				0.0371s 6	0.0294s 4	0.2055s 5	0.1005s 21	0.0660s 24	0.0628s 7	0.1215s 4	0.1287s 4	0.0209s 4	0.0442s 4	0.0328s 4	0.0032s 4	0.0148s 4	0.0116s 4	0.0112s 4	0.0070s 4	0.1138s 0.0874s 4	0.1075s 4	0.2130s 5	0.2260s 6	0.2457s 7	0.4146s 6	0.0742s 6	0.1345s 4	0.1694s 4	0.1447s 4	0.1980s 15	0.1038s 4	0.1767s 4
12.3d					0.0426s 6	0.2177s 6	0.1261s 6	0.0924s 6	0.0928s 7	0.1458s 13	0.1395s 6	0.0568s 6	0.0031s 4	0.0472s 6	0.0139s 5	0.0127s 5	0.0065s 4	0.1323s 6	0.1035s 6	0.1326s 6	0.2260s 6	0.2457s 7	0.4146s 6	0.0742s 6	0.1345s 5	0.1054s 7	0.1754s 6	0.2081s 7	0.1188s 8	0.1767s 8
13.3d						0.2191s 8	0.1312s 7	0.0932s 7	0.1003s 8	0.1493s 7	0.1436s 9	0.0568s 7	0.0031s 4	0.0513s 6	0.0155s 5	0.0156s 6	0.0065s 4	0.1384s 7	0.1130s 6	0.1396s 7	0.2514s 7	0.2634s 6	0.4311s 7	0.0907s 6	0.1425s 7	0.1754s 6	0.2146s 7	0.1204s 7	0.1895s 7
14.3d							0.1142s 27	0.0745s 4	0.0706s 4	0.1382s 4	0.1469s 43	0.0310s 5	0.0034s 4	0.0374s 6	0.0123s 5	0.0121s 6	0.0075s 4	0.1268s 4	0.0977s 4	0.1182s 4	0.2405s 4	0.2910s 4	0.5088s 5	0.0609s 4	0.1210s 4	0.1658s 4	0.2251s 5	0.1093s 23	0.2151s 63
15.3d								0.0659s 27	0.0659s 5	0.1223s 5	0.1272s 4	0.0296s 5	0.0029s 4	0.0350s 5	0.0108s 4	0.0104s 4	0.0060s 4	0.1144s 4	0.0893s 4	0.1091s 5	0.2144s 5	0.2509s 4	0.4298s 5	0.0564s 4	0.1118s 4	0.1463s 5	0.2089s 5	0.1030s 5	0.1669s 5
16.3d									0.0659s 14	0.1239s 5	0.1294s 5	0.0339s 5	0.0028s 5	0.0379s 5	0.0112s 4	0.0104s 4	0.0059s 4	0.1185s 6	0.0939s 5	0.1108s 6	0.2051s 6	0.2367s 5	0.4027s 6	0.0604s 5	0.1137s 6	0.1436s 6	0.1944s 6	0.1063s 5	0.1668s 5
17.3d										0.1347s 39	0.1395s 6	0.0393s 5	0.0028s 5	0.0388s 5	0.0120s 4	0.0110s 4	0.0056s 5	0.1244s 5	0.1000s 5	0.1197s 6	0.2291s 6	0.2618s 6	0.4301s 6	0.0668s 6	0.1076s 6	0.1626s 6	0.2149s 6	0.1159s 6	0.1755s 6
18.3d											0.1314s 47	0.0302s 4	0.0032s 4	0.0355s 4	0.0110s 4	0.0109s 4	0.0062s 5	0.1158s 5	0.0939s 5	0.1073s 5	0.2150s 5	0.2526s 5	0.4337s 5	0.0558s 5	0.1108s 5	0.1465s 5	0.2011s 5	0.1035s 5	0.1601s 5
19.3d												0.0306s 17	0.0029s 4	0.0347s 4	0.0106s 4	0.0103s 4	0.0066s 5	0.1130s 5	0.0864s 4	0.1064s 5	0.2099s 5	0.2459s 5	0.4272s 5	0.0551s 5	0.1105s 5	0.1428s 5	0.2013s 5	0.1012s 5	0.1593s 5
20.3d													0.0027s 6	0.0489s 5	0.0140s 5	0.0136s 5	0.0066s 4	0.1475s 7	0.1166s 7	0.1645s 6	0.2761s 6	0.2908s 7	0.4714s 7	0.0831s 6	0.1505s 7	0.1990s 7	0.2555s 8	0.1431s 7	0.2144s 8
21.3d														0.0351s 5	0.0126s 4	0.0111s 5	0.0059s 4	0.1013s 5	0.0815s 5	0.0970s 5	0.1772s 5	0.2033s 5	0.3480s 4	0.0554s 4	0.0989s 5	0.1236s 5	0.1658s 6	0.0932s 5	0.1376s 5
22.3d															0.0156s 5	0.0133s 5	0.0064s 4	0.1348s 7	0.1118s 7	0.1264s 7	0.2165s 7	0.2312s 5	0.3925s 6	0.0808s 6	0.1392s 7	0.1570s 7	0.2069s 7	0.1302s 7	0.1692s 8
23.3d																0.0193s 6	0.0080s 4	0.1409s 7	0.1226s 8	0.1587s 7	0.2524s 7	0.4298s 7	0.0887s 6	0.1637s 7	0.1911s 7	0.2396s 7	0.1548s 7	0.2036s 7
24.3d																	0.0091s 5	0.1841s 7	0.1568s 8	0.1844s 7	0.2856s 8	0.4580s 8	0.1226s 7	0.1835s 8	0.2077s 8	0.2687s 8	0.1796s 8	0.2726s 8
25.3d																		0.1083s 5	0.0867s 7	0.1038s 6	0.1852s 6	0.2116s 6	0.4568s 6	0.0660s 5	0.1031s 6	0.1312s 6	0.1743s 7	0.0961s 6	0.1458s 6
2A.3d																			0.0863s 32	0.1077s 7	0.2143s 6	0.2430s 6	0.4216s 6	0.0539s 5	0.1064s 6	0.1399s 5	0.1930s 7	0.1012s 6	0.1566s 6
2B.3d																				0.1183s 46	0.2039s 13	0.2425s 5	0.4121s 6	0.0562s 5	0.1092s 5	0.1413s 5	0.1931s 6	0.1003s 5	0.1556s 5
2C.3d																					0.1977s 26	0.2423s 12	0.4138s 5	0.0552s 4	0.1086s 5	0.1414s 5	0.1943s 5	0.1038s 5	0.1551s 5
3.3d																						0.2642s 9	0.4641s 7	0.0560s 4	0.1122s 4	0.1456s 5	0.2045s 5	0.1029s 4	0.1641s 4
4.3d																							0.4954s 54	0.0558s 4	0.1110s 5	0.1464s 5	0.2072s 12	0.1028s 5	0.1657s 5
5A.3d																								0.0622s 4	0.1182s 5	0.1547s 18	0.2159s 8	0.1098s 4	0.1523s 6
5B.3d																									0.1020s 25	0.1391s 6	0.1848s 5	0.0989s 5	0.1521s 7
6A.3d																										0.1366s 23	0.1985s 32	0.1024s 5	0.1579s 6
6B.3d																											0.2102s 63	0.0994s 15	0.1610s 5
7.3d																												0.1010s 4	0.1592s 4
8.3d																													0.1533s 18
9.3d																													