UNIVERSIDAD POLITECNICA DE VALENCIA

DEPARTAMENTO DE INGENIERÍA GRÁFICA

TESIS DOCTORAL

Aplicación para reconocimiento de bocetos basada en sistemas multi-agente

Presentada por:

D. Daniel García Fernández-Pacheco

Dirigida por:

Dra. Dña. Nuria Aleixos Borrás
Dr. D. Julián Conesa Pastor

SEPTIEMBRE 2010

AGRADECIMIENTOS

A Nuria Aleixos,

por haber sido una magnífica directora y una persona muy cercana que ha sabido sacar en todo momento lo mejor de mí, apoyándome siempre en los momentos difíciles.

A Julián Conesa,

por su ayuda y colaboración, así como por haberme introducido en esta línea de investigación.

A Fco. Albert Gil,

por el excelente trabajo que ha aportado a esta tesis, y por haber sido un gran compañero y amigo.

A José Blasco,

por su apoyo y comprensión durante todo el período de realización de la tesis.

A Manuel Contero,

por sus ánimos y colaboración durante el trascurso de esta tesis.

Al Departamento de Ingeniería Gráfica de la Universidad Politécnica de Valencia,

por su acogida y amabilidad durante mis estancias en el departamento. A todos mil gracias.

A los revisores externos,

por su dedicación y sus sugerencias de mejora.

Al Departamento de Expresión Gráfica de la Universidad Politécnica de Cartagena,

por su comprensión y apoyo, por el esfuerzo personal que algunos han dedicado para que yo pudiera dedicar tiempo a este trabajo, por poner a mi disposición todos los recursos necesarios, siempre os estaré agradecido.

A mi esposa, familia y amigos,

por sobrellevar mi ausencia. Os he tenido abandonados en estos años, espero recuperar el tiempo perdido.

ÍNDICE
DE CONTENIDOS

ÍNDICE
DE FIGURAS

ÍNDICE
DE TABLAS

LISTADO DE ACRÓNIMOS

1. Acrónimos de términos generales:

- **ABLE**: **A**gent **B**uilding and **L**earning **E**nvironment
- **ADF**: **A**gent **D**escription **F**ile
- **ASME**: **A**merican **S**ociety Of **M**echanical **E**ngineers
- **BDI**: **B**eliefs **D**esires **I**ntentions
- **BUGG**: **B**rown **U**niversity **G**raphics **G**roup
- **CAD**: **C**omputer **A**ided **D**esign
- **CAE**: **C**omputer **A**ided **E**ngineering
- **CAM**: **C**omputer **A**ided **M**anufacturing
- **CAS**: **C**omputer **A**ided **S**ketching
- **CCSL: C**ornell **C**omputational **S**ynthesis **L**ab
- **CDL**: **C**omputational **D**esign **L**ab
- **CGGTUI**: **C**omputer **G**raphics **G**roup of the **T**echnical **U**niversity of **I**lmenau
- **CIGRO**: **C**alligraphic **I**nterface for **G**eometric **R**econstructi**o**n
- **CSCW**: **C**omputer **S**upported **C**ollaborative **W**ork
- **CVCUAB**: **C**entro de **V**isión por **C**omputador de la **U**niversidad **A**utónoma de **B**arcelona
- **DCR**: **D**irection **C**hange **R**atio
- **DFT**: **D**iscrete **F**ourier **T**ransform
- **DLL**: **D**ynamic-**L**ink **L**ibrary
- **FFT**: **F**ast **F**ourier **T**ransform
- **FIPA**: **F**oundation for **I**ntelligent **P**hysical **A**gents
- **GEGROSS**: **Ge**sture & **G**eometric **R**econstructi**o**n based **S**ketch **S**ystem

- **GNU-LGPL**: **G**NU's **N**ot **U**nix - **L**esser **G**eneral **P**ublic **L**icense
- **GUIR**: **G**roup for **U**ser **I**nterface **R**esearch
- **IBM**: **I**nternational **B**usiness **M**achines
- **IDE**: **I**ntegrated **D**evelopment **E**nvironment
- **IEEE**: **I**nstitute of **E**lectric and **E**lectronic **E**ngineers
- **IMMI**: **I**ntelligent **M**ulti**M**odal **I**nterfaces Group
- **IPL**: **I**mage **P**rocessing **L**ibrary
- **J2ME**: **J**ava **2** **M**icro **E**dition
- **J2SE**: **J**ava **2** **S**tandard **E**dition
- **JADE**: **J**ava **A**gent **De**velopment
- **JADE-LEAP**: **J**ava **A**gent **De**velopment – **L**ightweight **E**xtensible **A**gent **P**latform
- **JADEX**: **J**ava **A**gent **De**velopment E**x**tended
- **JASON**: **J**ava-based **A**gentSpeak interpreter used with **S**aci for multi-agent distribution **o**ver the **n**et
- **JAVA**: **J**ust **A**nother **V**irtual **A**pproach
- **JDK**: **J**ava **D**evelopment **K**it
- **JNI**: **J**ava **N**ative **I**nterface
- **JWT**: **J**ava **W**ireless **T**oolkit
- **KSE**: **K**nowledge **S**haring **E**ffort
- **MCRPC**: **M**icrosoft **C**enter for **R**esearch on **P**en-Centric **C**omputing
- **MIT**: **M**assachusetts **I**nstitute of **T**echnology
- **NDDE**: **N**ormalized **Di**stance between **D**irection **E**xtremes
- **OMG**: **O**bject **M**anagement **G**roup

- **PDA**: **P**ersonal **D**igital **A**ssistant
- **PHP**: **P**HP **H**ypertext **P**reprocessor
- **RBF**: **R**adial **B**asis **F**unctions
- **SA**: **S**imulated **A**nnealing
- **SBIM**: **S**ketch-**B**ased **I**nterfaces and **M**odelling
- **SBIMW**: **S**ketch-**B**ased **I**nterfaces and **M**odelling **W**orkshop
- **STL**: **S**tandard **T**emplate **L**ibrary
- **SWIG**: **S**implified **W**rapper and **I**nterface **G**enerator
- **TCVD**: **T**angent **C**orner **V**ertices **D**etection
- **UIRG**: **U**ser **I**nterface **R**esearch **G**roup
- **UML**: **U**nified **M**odeling **L**anguage
- **UMPC**: **U**ltra **M**obile **P**ersonal **C**omputer
- **UPV**: **U**niversidad **P**olitécnica de **V**alencia
- **WIMP**: **W**indows, **I**con, **M**enu, **P**ointing device.
- **XML**: E**x**tensible **M**arkup **L**anguage

2. Acrónimos de los Agentes implementados:

- **ANA**: **An**gle **A**gent
- **ARA**: **Ar**c **A**gent
- **ARCA**: Artistic **Arc A**gent
- **AWA**: **A**rro**w A**gent
- **AXIA**: **Axi**s **A**gent
- **BAAM**: **B**roker **A**gent - **A**nnotation **M**odule
- **BACC**: **B**roker **A**gent – **C**entral **C**ore
- **BAM**: **B**roker **A**gent – **M**odule
- **BARM**: **B**roker **A**gent - **R**estriction **M**odule
- **BASM**: **B**roker **A**gent - **S**hape **M**odule
- **BATM**: **B**roker **A**gent - **A**rtistic **M**odule
- **CIA**: **C**ircle **A**gent
- **CONA**: **Con**centric **A**gent
- **CRA**: **C**ross-out **A**gent
- **DIAA**: **Dia**metral Dimension **A**gent
- **DIMA**: Lineal **Dim**ension **A**gent
- **EQUA**: **Equ**al Dimension **A**gent
- **EXTA**: **Ext**rusion **A**gent
- **FA**: **F**eature **A**gent
- **GPA**: **G**eneral **P**olygon **A**gent
- **HORA: Hor**izontal **A**gent
- **IA**: **I**nterface **A**gent

- **INA**: **In**telligent **A**gent
- **LIA**: **Li**ne **A**gent
- **LINA**: Artistic **Lin**e **A**gent
- **PA**: **P**re-processing **A**gent
- **PARA**: **Par**allel **A**gent
- **PERA**: **Per**pendicular **A**gent
- **POA**: **P**oint **A**gent
- **RADA**: **Rad**ial Dimension **A**gent
- **REVA**: **Rev**olution **A**gent
- **ROA**: **Ro**und Arrow **A**gent
- **SECA**: **Sec**tion **A**gent
- **SICA**: **S**ketch **I**nterpretation/**C**lassification **A**gent
- **TANA**: **Tan**gency **A**gent
- **VERA**: **Ver**tical **A**gent

RESUMEN DE LA TESIS

Resumen de la tesis

El empleo de los interfaces naturales o basados en bocetos no se ha extendido aún debido a los problemas que presentan en la actualidad, tales como el de la robustez, repetibilidad o fiabilidad, los cuales no están garantizados en este tipo de interfaces de usuario. Los reconocedores actuales que se pueden utilizar en línea sobre aplicaciones interactivas tampoco ofrecen soluciones idóneas. Estos suelen ser poco flexibles y su porcentaje de acierto en la clasificación decrece según aumenta el número de gestos o símbolos que admite el sistema. Este tipo de interfaces es muy conveniente, ya que permitirían soportar las fases iniciales del diseño (diseño conceptual) que actualmente se trabajan únicamente con lápiz y papel, debido fundamentalmente a que la funcionalidad que ofrece lo que existe hoy en día no supera en nada a dichas herramientas tradicionales, todo lo contrario. La idea pues, es la creación de un interfaz que permita los bocetos paramétricos por ordenador y su posterior reutilización en aplicaciones CAD para la creación de geometría, que es la motivación principal de esta tesis.

En la presente tesis se ha diseñado un paradigma de reconocimiento de bocetos fiable y robusto que puede dar soporte al usuario en las primeras etapas del diseño conceptual, y en último término se ha desarrollado un interfaz caligráfico natural (que permite la generación espontánea de ideas en papel), adaptativo (que permite al usuario definir geometría imprecisa e incompleta) y transparente (que el usuario puede dibujar una línea a trazos o un rayado de un área sin anunciar dicha intención a la aplicación por medio de algún menú).

Para la implementación del interfaz diseñado se ha recurrido a una plataforma multi-agente, pues se ha comprobado que los sistemas basados en agentes son válidos para aquellas aplicaciones que necesitan de reglas de decisión guiadas por el conocimiento, donde es importante la información y conocimiento disponible del contexto para tomar decisiones, pudiendo cambiar dichas decisiones en función de diferentes posibilidades, permitiendo que el usuario pueda dibujar con total libertad sin importar lo que dibuje, el número de trazos o la secuencia de introducción de los mismos.

A su vez, se ha dotado al interfaz de un carácter modular, permitiendo así añadir nuevos gestos/símbolos al diccionario con el mínimo intrusismo sobre el sistema diseñado y facilitando la flexibilidad del sistema.

Son varias las aportaciones novedosas que se han hecho al interfaz implementado, entre ellas la incorporación en el reconocedor del “interspersing” (interrupción de la actividad que realiza el usuario para realizar una acción concreta y proseguir luego en el punto en que se quedó), el “overtracing” (trazos superpuestos para creación de geometría imitando el bocetado “artístico” en papel) y el cambio de modo automático (no es necesario atender a menús para

cambiar de modo “geometría” a modo “comando” o a modo “introducción de restricciones u otros símbolos del boceto”, reconociéndose automáticamente la intención del usuario).

Además, en el reconocedor se ha recurrido al desarrollo e implementación de un nuevo método de segmentación del boceto en el que se detectan además de los vértices esquinas, los vértices tangentes, con un porcentaje de acierto cercano al 100%, lo que mejora en gran medida los métodos de segmentación de bocetos existentes en la literatura hasta el momento.

Por último, se ha recurrido al empleo de un marco para la optimización que permite el entrenamiento automatizado del sistema, entrenamiento que actualmente se lleva a cabo fuera de línea en la mayoría de aplicaciones que necesitan de este entrenamiento previo (consumiendo mucho tiempo), y que permite además el ajuste inicial de los parámetros de dicho sistema para que éste alcance la mejor solución posible en el reconocimiento.

En general, los desarrollos llevados a cabo en esta tesis incluyen el conocimiento profundo del **análisis de imagen**, de técnicas de **segmentación**, de **sistemas basados en agentes**, de **clasificadores / reconocedores** y de técnicas de **optimización**.

Resum de la Tesi

L'ocupació dels interfícies naturals o basats en esbossos no s'ha estés encara a causa dels problemes que presenten en l'actualitat, com ara el de la robustesa, repetibilitat o fiabilitat, els quals no estan garantits en este tipus d'interfícies d'usuari. Els reconeixedors actuals que es poden utilitzar en línia sobre aplicacions interactives tampoc oferixen solucions idònies. Estos solen ser poc flexibles i el seu percentatge d'encert en la classificació decreix segons augmenta el nombre de gestos o símbols que admet el sistema. Este tipus d'interfícies és molt convenient, ja que permetrien suportar les fases inicials del disseny (disseny conceptual) que actualment es treballen únicament amb llapis i paper, degut fonamentalment a que la funcionalitat que oferix el que existix hui en dia no supera en res a les dites ferramentes tradicionals, tot el contrari. La idea és la creació d'un interfície que permeta els esbossos paramètrics per ordinador i la seua posterior reutilització en aplicacions CAD per a la creació de geometria, que és la motivació principal d'esta tesi.

En la present tesi s'ha dissenyat un paradigma de reconeixement d'esbossos fiable i robust que pot donar suport a l'usuari en les primeres etapes del disseny conceptual, i en últim terme s'ha desenrotllat un interfície cal·ligràfic natural (que permet la generació espontània d'idees en paper), adaptatiu (que permet a l'usuari definir geometria imprecisa i incompleta) i transparent (que l'usuari pot dibuixar una línia a traços o un ratllat d'una àrea sense anunciar la dita intenció a l'aplicació per mitjà d'algun menú).

Per a la implementació de l'interfície dissenyat s'ha recorregut a una plataforma multi-agent, perquè s'ha comprovat que els sistemes basats en agents són vàlids per a aquelles aplicacions que necessiten regles de decisió guiades pel coneixement, on és important la informació i coneixement disponible del context per a prendre decisions que guien el reconeixement, podent canviar les dites decisions en funció de diferents possibilitats, permetent que l'usuari puga dibuixar amb total llibertat sense importar el que dibuixe, el nombre de traços o la seqüència d'introducció dels mateixos.

A més, s'ha dotat a l'interfície d'un caràcter modular, permetent així afegir nous gestos/símbols al diccionari amb el mínim intrusisme sobre el sistema dissenyat i facilitant la flexibilitat del sistema.

Són diverses les aportacions que s'han fet a l'interfície implementat, entre elles la incorporació en el reconeixedor del “interspersing” (interrupció de l'activitat que realitza l'usuari per a realitzar una acció concreta i prosseguir després en el punt en què es va quedar), el “overtracing” (traços superposats per a creació de geometria imitant el dibuixat “artístic” en paper) i el canvi de manera automàtic (no cal atendre a menús per a canviar de mode

"geometria" a mode "ordre" o a mode "introducció de restriccions o altres símbols de l'esbós", reconeixent-se automàticament la intenció de l'usuari).

A més, en el reconeixedor s'ha recorregut al desenrotllament i implementació d'un nou mètode de segmentació de l'esbós en què es detecten a més dels vèrtexs cantons, els vèrtexs tangents, amb un percentatge d'encert pròxim al 100%, la qual cosa millora en gran manera els mètodes de segmentació d'esbossos existents en la literatura fins al moment.

Finalment, s'ha recorregut a l'ocupació de un marc per a l'optimització que permet l'entrenament automatitzat del sistema, entrenament que actualment es du a terme de forma off line en la majoria d'aplicacions que necessiten este entrenament previ (consumint molt de temps), i que permet a més l'ajust inicial dels paràmetres del dit sistema a aconseguir la millor solució possible.

A la fi, els desenrotllaments duts a terme en esta tesi inclouen el coneixement profund de **l'anàlisi d'imatge**, de tècniques de **segmentació**, de **sistemes basats en agents**, de **classificadors / reconeixedors** i de tècniques d'**optimització**.

Summary of the Thesis

The use of natural or sketch based interfaces is not extended at all due to the problems they present at the moment, like the lack of robustness, repeatability and reliability, which are not guaranteed in this kind of user interfaces. The actual recognizers used on line in interactive applications neither offer ideal solutions. They use to be rigid and the success ratio in the classification decreases when the number of gestures or symbols admitted by the system increases. This kind of interfaces are much convenient, so they could support the initial stages of design (conceptual design), in which the traditional pen and paper continues being used. On the contrary, the modern tools don't improve the functionality of these traditional tools. So the idea is to create an interface that permits to draw parametric sketches on a computer and later use the sketches in CAD applications to create geometry, which is the principal aim of the thesis.

In the present thesis a reliable and robust sketch recognition paradigm that supports the user in the first conceptual design stages has been designed. In last term, this has allowed to develop a natural (it permits the spontaneous generation of ideas on paper), adaptative (it permits to define imprecise and incomplete geometry) and transparent (the user can change his activity without previous advertisement –no menus available- and his intention is captured) caligraphic interface.

In order to implement the designed interface, a multi-agent platform has been selected. It has been verified that agent based systems are valid for applications that need knowledge guided rules, where the context information is important to take decisions, and these decisions can be changed in terms of different possibilities, offering the user the possibility to draw freely no matter what he sketches, the number of strokes or the order they are introduced.

Besides, the interface has been endowed with a modular structure, permitting to add new gestures / symbols to the dictionary with the minimal intrusion in the designed system and making the system more flexible.

There are several innovative contributions that have been implemented on the interface, like the inclusion to the recognizer of the "interspersing" (the act of interrupting an action and continuing later in the same point), the "overtracing" (superimposed strokes for creating geometry imitating the artistic sketching on paper), and the automatic change mode (it's not necessary to use menus or buttons for changing between the "geometry", "command" or "introduction of restrictions" or another sketching symbols) recognizing the user's intention automatically.

Besides, concerning the recognizer, a new segmentation method of the sketch has been developed and implemented. In the proposed method, apart from the vertex corners, the

tangents corners are detected with success ratios close to 100%, what supposes an improvement over the segmentation methods that can be found nowadays in literature.

Finally, a frame for the optimization that permits the automated training of the system has been used. This training is being carried out by processes off line in the majority of applications that need of this previous workout (consuming long time). The developed optimization in this thesis permits the automated initial adjustment of the system parameters in order to reach the best possible solution.

Summarizing, the developments accomplished in this thesis include the intensive knowledge of the **image analysis** techniques, **segmentation** techniques, **agent-based systems**, **classifiers / recognizers**, and **optimization** techniques.

Capítulo 1:
INTRODUCCIÓN

1. INTRODUCCIÓN

Esta tesis hace aportaciones en el ámbito de los interfaces naturales para crear dibujos de ingeniería a mano alzada, y más concretamente, para ofrecer soporte en la fase de diseño conceptual.

Diseñar mediante bocetos es una técnica vigente de diseño conceptual en ingeniería. Sin embargo, el proceso de construcción automática de los modelos CAD tridimensionales a partir de los bocetos es un problema aún sin resolver. El problema es importante, dado que los modelos CAD tridimensionales son necesarios para el resto del proceso de diseño, y su construcción a partir de los bocetos produce retrasos y suele introducir errores.

La interacción mediante técnicas WIMP (Windows, Icon, Menu, Pointing device), esto es, entornos basados en ventanas con menús e iconos, no es la más apropiada para el proceso de diseño conceptual. No ayuda a los diseñadores a hacer lo que realmente desean (desarrollar su pensamiento creativo); si no que les fuerza a alterar su forma natural de trabajar (donde dominan las ideas) en aras de ajustarse a la forma rígida de trabajar en un ordenador. Por tanto, es necesario encontrar métodos de interacción mejor adaptados al proceso de diseño conceptual.

En este capítulo se introduce el problema y se pone en situación al lector, aportándose argumentos para justificar la vigencia actual del diseño conceptual de ingeniería mediante bocetos, y posteriormente revisando el estado actual del ámbito denominado interacción y modelado basado en bocetos (SBIM, por sus siglas en inglés). La revisión es muy somera, puesto que únicamente trata de justificar que se trata de un problema no resuelto por diferentes motivos, y para centrar el ámbito de actuación de la tesis. Después se mencionan algunos de los problemas actuales de la interacción y el modelado basado en bocetos y por último se enuncian las aportaciones de la presente tesis, las cuales están encaminadas a proponer soluciones a dichos problemas.

En el capítulo segundo se pasa a detallar los objetivos de la presente tesis. En el capítulo tercero se exponen los trabajos más relevantes encontrados en el estado del arte más relacionados con el tema de la tesis. A continuación, en el cuarto capítulo, se exponen los materiales y métodos propuestos para la consecución de la misma, detallándose éstos en el quinto capítulo. En el capítulo sexto se propone un algoritmo de optimización de parámetros para el algoritmo de segmentación propuesto en los capítulos anteriores, y es en el séptimo y octavo capítulos donde se ofrecen los resultados de los diversos test realizados, así como las conclusiones finales que justifican la utilidad de esta tesis.

1.1 El boceto en el proceso de diseño

1.1.1 Utilidad del boceto en el proceso de diseño

Un estudio promovido por la *Engineering Design Graphics Division* de la *American Society for Engineering Education* [Bar04], concluye que la "habilidad para crear modelos

sólidos 3D por ordenador" y la "habilidad para bocetar objetos de ingeniería a mano alzada" son las dos destrezas más valoradas como competencia de los estudiantes de ingeniería en el ámbito de la comunicación gráfica. Otros estudios similares como el promovido por la *American Society Of Mechanical Engineers* ASME [Ros05] confirman estas conclusiones. Y autores como [Tve02, PA02] han analizado el importante papel que desempeña el uso de bocetos durante el proceso de desarrollo de nuevos productos industriales. Tal como destaca Jonson [Jon02] los puntos fuertes del uso de bocetos radican en su economía de medios (bajo coste), inmediatez (herramienta de interfaz sencilla) y facilidad de corrección y revisión (sobreescritura y borrado).

La llegada del CAD afectó profundamente a otras fases del proceso de diseño, pero no alteró prácticamente a la fase de diseño conceptual, en la que se ha mantenido el uso de los bocetos realizados con lápiz y papel. La práctica habitual es que después, y sólo después de que se haya obtenido un boceto final y la fase de diseño conceptual haya terminado, es cuando el diseñador comienza a crear un modelo CAD 3D desde cero.

La marginación de la fase de diseño conceptual respecto a la incorporación de herramientas asistidas por ordenador se debe a las carencias que tienen las herramientas CAD disponibles tanto a nivel comercial como académico. Cuando se han desarrollado herramientas con una funcionalidad adecuada, éstas han mostrado su efectividad. Por ejemplo, se ha demostrado que el uso de las herramientas CAS (*Computer Aided Sketching*) es, al menos, tan útil como el uso del lápiz y papel convencional para desarrollar las habilidades de visión espacial en estudiantes de ingeniería noveles [CNC05]. Sin embargo, la razón por la que las herramientas CAS no se han desarrollado de forma adecuada se debe en primer lugar a que el hardware necesario para su implementación no ha estado disponible hasta la reciente aparición en el mercado de los Tablet PC y UMPC (*Ultra Mobile Personal Computer*) y, en segundo lugar, por una limitada funcionalidad de dichos dispositivos que no ha mejorado las prestaciones del bocetado tradicional sobre papel.

Las herramientas CAD "paramétricas" disponibles comercialmente (con alguna capacidad de "pseudo-bocetado") distan mucho de ser una alternativa real para satisfacer todas las necesidades de bocetado, ya que están orientadas hacia el diseño de detalle y no hacia el diseño conceptual [Ott98]. Además, dichas herramientas fuerzan a los ingenieros en direcciones equivocadas tal como señala Jonson [Jon02]: "el ordenador impulsa al usuario a ir directo hacia el producto final, sin el periodo de pensamiento crítico y creativo". En otras palabras, tales herramientas CAD introducen al diseñador en un mundo de acciones (dibujo) en lugar de en un mundo de ideas (diseño). Esto se debe a que dichas herramientas están orientadas para obtener la mayor eficiencia computacional, y no orientadas a conseguir la eficiencia para el diseñador.

Para desarrollar entornos digitales de bocetado que sustituyan el lápiz y el papel, dichos entornos deben proporcionar un proceso "natural" que no entorpezca al usuario. Existen dos aspectos clave para proporcionar un entorno digital de bocetado efectivo. El primer aspecto clave está relacionado con la funcionalidad proporcionada por la aplicación de bocetado. El objetivo es ofrecer un "papel aumentado" al usuario, para ofrecerle la misma

funcionalidad que aporta el papel real. Posteriormente, y como segundo aspecto, se presenta la ventaja de poder aportar funciones "extra" a dicho papel aumentado que proporcionaría el software de la aplicación. Esto se debe llevar a cabo a la vez que se garantiza, al menos, la misma libertad ofrecida por el papel real, por lo que también se tendrá en cuenta la usabilidad del software [PA02]. De acuerdo con el IEEE, se entiende que la usabilidad es la facilidad con la que un usuario puede aprender a operar, preparar entradas e interpretar las salidas de un sistema o componente. En el contexto de una aplicación CAS, una elevada usabilidad significa que la aplicación sea tan cómoda de manejar como el lápiz y el papel reales.

Resumiendo, los requerimientos de usabilidad de una aplicación CAS implican que debe ser tan amigable y transparente como lo son el lápiz y el papel. Los requerimientos de funcionalidad básica o "estándar" significan que es necesario proporcionar unas prestaciones adicionales al lápiz y papel convencionales, como facilitar la exploración rápida de diferentes alternativas de diseño y facilitar al diseñador la transferencia de su idea/diseño final a las herramientas CAD/CAE/CAM que se utilizarán en las posteriores fases del diseño. Esta funcionalidad más avanzada también justifica la importancia de la creación de modelos a partir de bocetos.

En suma, el reto de sustituir el bocetado convencional con lápiz y papel por un entorno digital de bocetado existe; este nuevo entorno debe diseñarse para favorecer un proceso "natural" que no entorpezca al usuario y que ofrezca como salida un modelo digital de diseño que se podrá reutilizar en el resto de fases del proceso de diseño. La arquitectura que se propone e implementa en esta tesis está pensada para este propósito.

1.1.2 Tipos del boceto en el proceso de diseño

Para centrar el ámbito de actuación de la tesis se hace necesario caracterizar los diferentes tipos de bocetos que el ingeniero/diseñador utiliza en su actividad creativa. Aunque existen otras clasificaciones, en esta tesis se va a utilizar la clasificación de bocetos propuesta por Ferguson [Fer92]. En este sentido, distinguimos entre "bocetos de ideación" (*thinking sketches*) utilizados para centrar y guiar el pensamiento no verbal, "bocetos de discusión" (*talking sketches*) empleados para dar soporte a los razonamientos sobre el diseño, y "bocetos prescriptivos" (*prescriptive sketches*) aplicados para dar instrucciones al delineante, el cual está encargado del acabado final del plano de ingeniería.

Tal como se indica en Company et al. [CCV09], la mayor parte de la actividad investigadora en el desarrollo de herramientas CAS se ha centrado en los últimos años en el campo de los bocetos de ideación, con el desarrollo de aplicaciones que transforman de forma automática bocetos en modelos 3D. Así, podemos distinguir dos aproximaciones principales:

1. Modelado basado en reconstrucción (p.e. [CNC05]), donde se interpreta una o varias vistas esbozadas y se genera un modelo 3D plausible automáticamente.

2. Modelado basado en gestos (p.e. [CNJ03]) donde se utilizan operaciones de modelado codificadas a través de gestos para transformar secciones 2D esbozadas en modelos 3D.

Con respecto a los bocetos de discusión, existen algunas aportaciones dentro del ámbito del *Computer Supported Collaborative Work* (CSCW), pero orientadas a la creación colaborativa y al uso compartido de bocetos 2D. Sin embargo, en la práctica se ha avanzado muy poco en el campo de su interpretación automática y posterior conversión a modelos 3D digitales. Esta situación es similar en la interpretación de bocetos prescriptivos para la generación de modelos 3D a partir de estos mismos [HCE97, JK84].

1.2 Modelado a partir de bocetos

El campo del modelado y los interfaces basados en bocetos SBIM (*Sketch-Based Interfaces and Modelling*) es un campo de investigación emergente [NCC07]. Prueba de ello es que a nivel europeo el foro especializado en este campo tan sólo tiene seis años de antigüedad. Se trata del *workshop* SBIM [SBIMW], cuyos objetivos actuales son todavía muy variados, siendo uno de los campos de trabajo más activos la creación de modelos 3D a partir de bocetos de ideación. En esta línea de investigación también cabe destacar a otros grupos nacionales e internacionales, como [CDL, GUIR, CGGTUI, BUGG, CCSL, IMMI, UIRG, CVCUAB y MCRPC], entre otros.

Tal y como se ha descrito en el apartado anterior, existen dos aproximaciones al problema de transformar un boceto de ideación en un modelo 3D en el contexto del desarrollo de productos industriales [PLD04]: las basadas en técnicas de reconstrucción geométrica y las denominadas de modelado gestual.

El método de reconstrucción geométrica de modelos a partir de dibujos lineales o bocetos, basado en percepción de señales visuales apareció como consecuencia del trabajo de Marill [Mar91] en el que se planteaba la “optimización de regularidades”. En breve, el método proponía obtener un modelo tridimensional añadiendo profundidad (coordenada z) a cada uno de los vértices de la figura plana original. El criterio para asignar profundidades apropiadas era elegir aquellas que optimizaban ciertas propiedades geométricas del objeto que se pretende reconstruir. A dichas propiedades se las denominó “regularidades”. El problema residía en encontrar las regularidades del modelo antes de reconstruirlo. Fueron Leclerc y Fischler [LF92] quienes primero plantearon un método para detectar propiedades que el modelo podía tener, a partir de señales perceptivas que la figura tenía. Es decir, que describieron un método práctico para relacionar ciertas características del dibujo original, con propiedades probables del modelo buscado. En concreto, formularon una señal perceptiva (planicidad de caras) como una función de mérito para un proceso de optimización matemática. En el posterior trabajo de Lipson y Sphitalni [LS96] se amplió el catálogo de señales perceptivas convertidas en regularidades, y también Company et al. [CCC04] y Piquer et al. [PMC04] realizaron contribuciones en este campo. La contribución más reciente es debida a Yuan et al. [YTJ08].

Los fundamentos teóricos de la relación de las señales perceptivas con la percepción visual se pueden encontrar en Palmer [Pal99], mientras que en el libro de Hoffmann [Hof00] se puede encontrar una aproximación menos detallada, pero con la ventaja de incluir criterios que se aproximan bastante a criterios de diseño de un sistema informático orientado hacia la percepción visual artificial.

El otro enfoque se basa en la interacción con el usuario mediante gestos que son reconocidos como comandos generadores de sólidos a partir de secciones 2D (ver Figura 1.1).

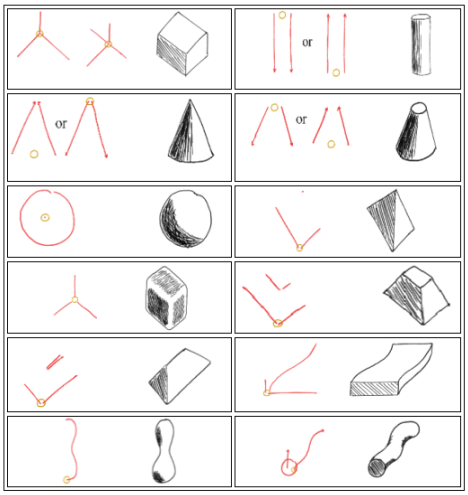


Figura 1.1. Interfaz del sistema SKETCH con bocetos de ideación

Hay que tener en cuenta que en este enfoque muchas veces se simplifica el reconocimiento limitando al usuario a dar la entrada de una determinada manera, ya que parece lógico que una acción tenga sentido si se realiza siguiendo una determinada secuencia. Si lo ilustramos con un ejemplo, no es lo mismo trazar una línea de abajo hacia arriba que de arriba hacia abajo, ya que es posible que queramos extruir en un sentido u otro, sin embargo, el resultado en pantalla es el mismo: una línea vertical. Muchos trabajos que basan su reconocimiento de gestos en este principio limitan la forma de operar del usuario, proporcionando un interfaz rígido y poco intuitivo.

Como ejemplos de esta línea que denominamos de “interfaz gestual” tenemos el sistema SKETCH [ZHH96]. Esta aplicación está orientada básicamente a formas arquitectónicas, donde el proceso de modelado de la geometría queda definido a través de un código de gestos y convencionalismos basados en el dibujo de simples líneas. El gesto más básico consiste en trazar tres líneas, alineadas con los tres ejes coordenados del dibujo, que inciden en un punto. Basado en este sistema, SKETCH-N-MAKE [BZ98] pretende facilitar la tarea de mecanizado por control numérico de piezas sencillas modeladas mediante un interfaz de tipo gestual. El sistema TEDDY [IMT99] permite modelar mediante un interfaz muy sencillo superficies tridimensionales de forma libre. El procedimiento consiste en trazar un boceto que represente la silueta del objeto, y el sistema propone automáticamente una superficie a través

de una malla poligonal que encaje en esa silueta. Basado en este sistema, en 2004, Alexe et al. [AGB04] presentaron otro modelador de superficies de forma libre basado en algunos algoritmos desarrollados por Igarashi [IMT99].

El sistema GIDeS [PJB00] permite la entrada a partir de una única proyección en perspectiva para construir diseños precisos a partir de dibujos ambiguos. El sistema dispone de un alfabeto de gestos que identifican un conjunto básico de primitivas de modelado. Por contra, el reconocimiento dinámico de estos gestos de modelado muestra al usuario una serie de iconos contextuales, que le obligan a confirmar su intención de diseño en caso necesario.

Finalmente, podemos señalar que existe un tercer tipo de sistema que podríamos denominar híbrido, que combina los dos enfoques ya comentados. Por ejemplo el sistema CIGRO [CNJ03], derivado del trabajo desarrollado en REFER [CCP04], soporta la reconstrucción de objetos poliédricos ortogonales (normalones) y cuasi-ortogonales (cuasi-normalones), incorporando un interfaz caligráfico avanzado que permite realizar en tiempo real el proceso de reconstrucción a partir de una perspectiva axonométrica, empleando para ello un motor de inflado axonométrico. Posteriormente se desarrolló un sistema de bocetado basado en la reconstrucción geométrica y gestual, denominado GEGROSS [CCP04, NCA07a], el cual integra un interfaz de bocetado paramétrico en dos dimensiones denominado ParSketch [ANC07, NCA07b y NCA07c]. Este sistema GEGROSS posee dos limitaciones importantes: 1) El criterio principal para distinguir entre los modos gestual y geométrico viene determinado por el nivel de presión del lápiz empleado para introducir el trazo; y 2) Las entidades formadas por varios trazos se deben introducir siguiendo un orden concreto. Estas dos condiciones, como más adelante se concretará, se han resuelto en la presente tesis, permitiendo un cambio de modo automático (el sistema detecta automáticamente la intención del usuario), y dotando al usuario de total libertad a la hora de introducir el boceto, sin importar la secuencia de los trazos.

Otro ejemplo de sistema híbrido es SMARTPAPER [SC04], que combina comandos gestuales con el enfoque de reconstrucción desarrollado por Lipson [LS96].

Una revisión exhaustiva del estado del arte en el marco de la reconstrucción geométrica y modelado a partir de gestos se describe en [CPC05 y OSS09].

Aunque los avances realizados en el campo de la aproximación basada en reconstrucción no solventan todos los problemas que presenta el enfoque basado en optimización para la reconstrucción 3D, la aplicación de gestores de restricciones algebraicas que resuelven las restricciones extraídas de los esbozos (restricciones que están contenidas en las regularidades derivadas de los indicios o pistas, también llamados aquí '*cues*'), sí que puede ofrecer importantes avances al tema de la generación de modelos a partir de bocetos. Parte de esos problemas son abordados por Zou y Lee [ZL07], los cuales "arreglan" modelos 3D (reconstruidos a partir de bocetos 2D) previo a su uso en aplicaciones CAD. Los modelos 3D reconstruidos presentan inexactitudes debido a posibles errores en los datos de entrada proporcionados, al método de reconstrucción y a las inexactitudes propias del método. Para la corrección del modelo proponen un método que detecta restricciones geométricas presentes en el modelo reconstruido (tales como paralelismo y perpendicularidad entre caras), y

seleccionan un subconjunto de ellas que restrinja el modelo de forma suficiente y consistente, lo que hacen mediante un algoritmo basado en la optimización quasi-Newton que prioriza las restricciones detectadas según su tipo y elimina las redundantes e inconsistentes. Así pues, el futuro más prometedor se basaría en el paradigma propuesto de percepción de indicios incompletos y contradictorios (que podemos llamar pistas o '*cues*') y su conversión en restricciones de un entorno gestor de restricciones algebraicas, el cual generará como salida un modelo 3D perceptualmente válido.

Centrándonos en este campo de actuación, en la interpretación de bocetos se incluyen tanto los bocetos en sí mismos (conjunto de trazos que definen una geometría) como los símbolos y demás gestos necesarios para definir y manejar la información gráfica bocetada. Un ejemplo sería las restricciones geométricas o dimensionales (símbolos de paralelismo, concentricidad, dimensiones, ejes de simetría, etc.), como otros gestos esbozados que permitirían el manejo de la información y la ejecución de comandos (como por ejemplo los comandos de borrado, extrusión, edición de dimensiones, encuadre, etc.). En la Figura 1.2 se muestra un ejemplo de un modelo 3D (izquierda) que podría ser generado a partir del boceto de la derecha (conjunto de líneas de geometría, contorno exterior, eje de simetría, rayado de la sección) y de otra información complementaria también bocetada (dimensiones, restricciones geométricas, gesto de revolución, información de acanaladuras, roscas, etc.).

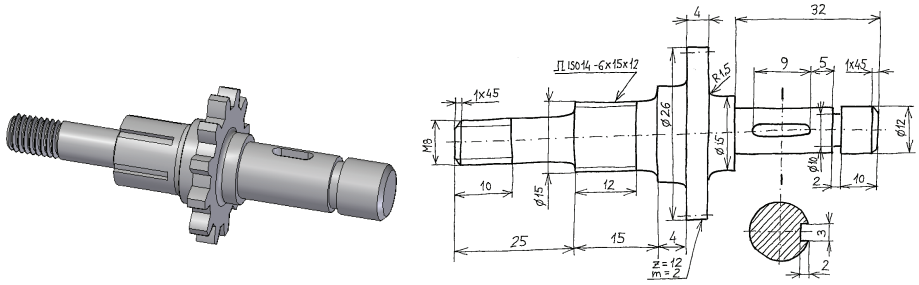


Figura 1.2. Modelo 3D de un eje (izquierda) y su correspondiente boceto prescriptivo (derecha)

Estos bocetos incluyen información gráfica como cotas, tolerancias y toda la simbología propia de los planos de ingeniería. En su contexto original estos bocetos servían para transmitir la definición de un diseño a un delineante para la generación de los planos finales de detalle. Existen trabajos sobre reconocedores de simbología propia de los planos de ingeniería, donde la información de partida no es un mapa de bits procedente de un escáner sino la información vectorial proporcionada en la tarea de bocetado. En este contexto el trabajo de Ablameyko et al. [ABG02] proporciona un marco adecuado respecto a la obtención de modelos paramétricos que compatibilicen la geometría proporcionada con el boceto, con la indicada por su acotación. El trabajo de Iyer y Mills [IM06] revisa trabajos previos en la transformación de planos de ingeniería en modelos 3D. Por último, Dimri y Gurumoorthy [DG04] se centran en el reconocimiento de las secciones.

Resumiendo, variados trabajos de investigación se han centrado en construir modelos 3D a partir de bocetos, ya sea basándose en métodos de optimización sobre una sola vista del

objeto, como basándose en bocetos de ideación que, combinados con un catálogo de comandos gestuales, permiten la ulterior creación de geometría en el espacio.

En este trabajo se propone una arquitectura que posibilita la implementación de un reconocedor que interprete bocetos, y cuyo resultado serviría como entrada a un sistema CAD basado en restricciones, el cual generaría como salida un modelo 3D perceptualmente válido. Esto permitirá dotar de una nueva funcionalidad a las aplicaciones CAD paramétricas comerciales, como es un interfaz natural. Dicho interfaz natural permitirá transformar un boceto bidimensional en un dibujo 2D vectorial y paramétrico, controlando mediante gestos las restricciones geométricas y dimensionales del mismo, de modo que sirva para generar posteriormente un modelo 3D.

Con este motivo entramos a tratar el tema de los reconocedores para el reconocimiento e interpretación de los bocetos (*sketches*), siendo aún éste un tema por resolver.

1.3 Los reconocedores en la actualidad y sus limitaciones

La nueva tecnología que incorpora lápices o pantallas ópticas se está imponiendo como una herramienta estándar que permite el desarrollo de nuevos interfaces de usuario, también conocidos como Interfaces Caligráficos, creados para sustituir las tradicionales herramientas de lápiz y papel. Estos interfaces poseen muchos inconvenientes, de hecho, el reconocimiento de símbolos dibujados a mano es un campo de investigación todavía muy activo. Hay que tener en cuenta que el reconocimiento de dichos símbolos o gestos de forma automática es una tarea compleja, ya que existe mucha variabilidad en el trazado de un usuario a otro, tanto en forma, como en orientación o tamaño.

Debido a estos inconvenientes, muchos algoritmos de reconocimiento no son lo bastante robustos o presentan ambigüedad en sus resultados. Una forma de minimizar el impacto de estos problemas es aplicar técnicas de procesado de imágenes. Sin embargo, la descripción de forma a través de características como la longitud, la anchura, el perímetro, el área, los momentos de inercia, el *bounding box*, etc., presenta el inconveniente de que se obtienen resultados similares para distintos tipos de formas, siendo además dependientes del tamaño y la orientación de las mismas. En su lugar, se puede recurrir a características invariantes como los descriptores de Fourier, o los momentos de Hu, entres otros, haciendo de esta forma al reconocedor independiente tanto al tamaño como a la orientación.

A lo largo del tiempo se han utilizado diversas técnicas para el reconocimiento de bocetos, ya sean símbolos, diagramas, formas geométricas u otro tipo de comandos gestuales, requiriendo la mayoría de una etapa previa de entrenamiento que posibilite al reconocedor clasificar el boceto introducido en función de la base de datos creada. Esto dificulta el empleo del reconocedor al tener cada usuario que pasar previamente por dicha etapa de entrenamiento.

Estas técnicas a su vez reducen la libertad con la que puede dibujar el usuario, limitando el número de trazos bocetados y exigiendo una secuencia de introducción de los

mismos. En algunos interfaces que poseen varios modos de funcionamiento, como por ejemplo un modo “geometría” para introducción de geometría, un modo “comando” para ejecutar comandos gestuales, o un modo “restricciones” para fijar restricciones a los elementos bocetados, se requiere de algún tipo de entrada explícita como el cambio de presión en el lápiz o la selección de un botón u orden de un menú para realizar dicho cambio de modo. Sin embargo, dicho cambio de modo debería realizarse de forma automática, reconociéndose automáticamente la intención del usuario.

Uno de los pasos más importantes en el proceso de reconocimiento e interpretación de bocetos a mano alzada es la segmentación del trazo esbozado para descomponerlo en primitivas más simples. Esta segmentación requiere a su vez de un algoritmo de detección de vértices que sea preciso, para la posterior extracción de las primitivas. Sin embargo, los trabajos encontrados en la literatura hasta la fecha presentan algunos inconvenientes que se deben solucionar, como por ejemplo la detección de vértices tangentes tan útiles en este tipo de aplicaciones.

En muchos campos de la ciencia se han empleado diferentes técnicas de optimización para mejorar los resultados esperados, aunque en el campo de los reconocedores su uso no está muy extendido, encontrándose únicamente en la literatura trabajos que aplican técnicas de optimización a la elección y optimización de parámetros en diferentes aspectos de la visión por computador, como por ejemplo en los sistemas de iluminación, entrenamiento, calibración del sistema de visión, etc. Aun así, el uso de este tipo de técnicas podría aportar una mayor eficiencia y eficacia en los resultados alcanzados en el proceso de segmentación de cualquier reconocedor.

A todo lo mencionado hay que añadir que los reconocedores existentes poseen el inconveniente de que son poco robustos, no muy eficientes, rígidos y muy dirigidos a lo que el reconocedor tiene que ser capaz de encontrar, fijando el dominio de trabajo previamente. Por esta razón se hace imprescindible un cambio en la metodología de los reconocedores para que éstos diferencien y clasifiquen los dibujos como lo haría un ser humano, esto es, utilizando información del contexto, siendo flexible en las decisiones intermedias y finales, pudiendo llegar a contradicciones que obliguen a cambiar la decisión a través de negociaciones consensuadas. Este tipo de comportamiento se le puede atribuir al reconocedor mediante el uso de sistemas expertos, de sistemas basados en objetos, y de sistemas basados en agentes, entre otros. De entre estos, la tecnología basada en agentes se ha empleado con éxito en muchas ocasiones para simulación y control de procesos, organización, control de tráfico, navegación, etc. aunque poco a poco su uso se está empezando a extender al proceso de reconocimiento en interfaces naturales, donde los resultados obtenidos demuestran que se trata de una opción válida que ofrece un avance importante en este campo.

1.4 Aportaciones de la tesis

A partir de los problemas mencionados que actualmente poseen los reconocedores de bocetos, la presente tesis propone una arquitectura que posibilita la implementación de un interfaz de bocetado natural (permite la generación espontánea de ideas en papel), adaptativo

(permite al usuario definir geometría imprecisa e incompleta) y transparente (el usuario puede dibujar una línea a trazos o un rayado de un área sin necesidad de anunciar dicha intención a la aplicación mediante algún icono o menú).

Para ello, son varios los aspectos que trata y mejora:

1. En primer lugar, se ha desarrollado un método muy sencillo y efectivo para la segmentación del trazo, esto es, la descomposición del trazo en las diferentes primitivas que lo componen. Este método basa la detección de los vértices en el cálculo de la función del radio del trazo, y considera tanto los vértices esquina (existentes entre dos rectas, y recta y arco) cómo los vértices tangentes (existentes entre recta y arco, y entre dos arcos).

   Se han implementado tres métodos recientes y relevantes encontrados en la literatura y se ha probado la superioridad del método desarrollado en esta tesis, sobre todo en lo que a vértices tangentes se refiere, donde no existe ninguna técnica de segmentación hasta el momento que se muestre efectiva.

   Este aspecto es muy importante, ya que el futuro reconocimiento se basa en la segmentación del trazo, lo que hace que éste mejore notablemente si la segmentación es buena.

2. En segundo lugar, se ha empleado la técnica de optimización Simulated Annealing para determinar los mejores valores de los parámetros utilizados en la fase de segmentación del reconocedor. Aunque su uso no está muy extendido en el ámbito de la tesis, este tipo de técnicas ha demostrado de manera firme aportar más eficiencia y eficacia en los resultados alcanzados en cualquier tipo de sistema al que han sido aplicados, tal y como se contrasta tras la aplicación de la misma al segmentador de la presente tesis.

3. Y en tercer lugar, se ha propuesto una arquitectura multi-agente para implementar el reconocedor de bocetos. El desarrollo de trabajos previos, como los descritos en el capítulo tercero, así como los resultados obtenidos en la presente tesis demuestran la viabilidad de este tipo de sistemas aplicados al campo del reconocimiento de bocetos. El empleo de una plataforma multi-agente ha posibilitado a su vez las siguientes aportaciones:

   a. Se explota la información del contexto, cosa que no suele hacerse en los reconocedores existentes, mirando éstos únicamente los aspectos más locales.

   b. Se utilizan técnicas de razonamiento similares a las utilizadas por los sistemas expertos que permiten cambiar las decisiones en función de diferentes posibilidades, al contrario que las técnicas existentes de reconocimiento que suelen ser rígidas en su comportamiento. Este elemento junto con el primero posibilitan que el usuario pueda

dibujar con total libertad sin importar el número de trazos o la secuencia de introducción de los mismos.

c. Se ha conseguido dotar al interfaz de un cambio de modo automático, esto es, no es necesario recurrir a un icono o botón para cambiar de modo "geometría" a modo "comando" o a modo "introducción de restricciones u otros símbolos del boceto", reconociéndose automáticamente la intención del usuario. Los sistemas actuales necesitan de algún tipo de entrada explícita como el cambio de presión en el lápiz o la selección de un botón u orden de un menú para realizar dicho cambio de modo.

d. Se ha diseñado un sistema fácilmente extensible, dotando al interfaz de un carácter modular, permitiendo añadir nuevos gestos/símbolos al catálogo con el mínimo intrusismo sobre el sistema diseñado.

e. Una vez introducido un trazo, se buscan todas las soluciones posibles basándonos en los trazos existentes, sin que éstos se hayan tenido que introducir de una manera concreta y posibilitando al usuario interrumpir el trazado de un gesto determinado para continuarlo posteriormente (termino denominado '*interspersing*').

f. También se ha definido un módulo artístico que permite el uso de varios trazos para representar una única línea o un único arco (conocido como '*overtracing*').

g. En el caso de que los trazos introducidos no conformen ningún símbolo del diccionario, estos se considerarán geometría o 'no gesto', evitando de esta manera los falsos positivos en el reconocimiento.

Capítulo 2:
OBJETIVOS

2. OBJETIVOS

Como se ha visto en el capítulo anterior, en la actualidad se sigue manteniendo una desconexión evidente entre las herramientas utilizadas durante la fase de diseño conceptual del proceso de desarrollo de un producto, y las herramientas CAD utilizadas en las fases siguientes. Los diseñadores e ingenieros mantienen en esta fase inicial la utilización de bocetos realizados a mano alzada como el mecanismo principal de comunicación y razonamiento geométrico sobre el producto. Teniendo en cuenta que durante el 20% inicial del proceso de desarrollo del producto se condiciona el 80% restante del coste final del mismo, la incorporación de nuevas herramientas de ayuda al diseño en estas fases iniciales constituye un ámbito de trabajo de gran interés, habida cuenta de la repercusión directa de cualquier mejora realizada en esta fase de diseño sobre el resultado final del mismo.

Por todo ello, se considera de sumo interés disponer de herramientas que permitan la definición geométrica del producto a partir de la información gráfica contenida en los bocetos, tanto de ideación como de los bocetos prescriptivos. El interés proviene de aceptar como hipótesis de partida que se puede acortar el ciclo de desarrollo del producto si se produce una integración entre la definición geométrica a partir de bocetos y la utilización posterior de herramientas CAD de modelado 3D.

Otro aspecto importante a tener en cuenta es el de la optimización. En muchos problemas de visión, como es el que nos ocupa, el rendimiento del proceso de segmentación depende en gran medida del algoritmo implementado y de su parametrización. Trabajos de investigación presentados en el capítulo sexto prueban sobradamente que la utilización de técnicas de optimización de los parámetros utilizados en la segmentación permite alcanzar mejores resultados que en aquellos casos en los que no se utilizan dichas técnicas.

De todo lo expuesto anteriormente, los objetivos generales que se proponen son los siguientes:

- Diseñar un paradigma de reconocimiento de bocetos fiable y robusto que dé soporte a las primeras fases del diseño.
- Diseñar un algoritmo de segmentación que sea capaz de detectar tanto los vértices esquina (existentes entre dos rectas, y recta y arco) cómo los vértices tangentes (existentes entre recta y arco, y entre dos arcos).
- Diseñar un marco para la optimización del sistema que permita el ajuste de los parámetros del algoritmo de segmentación diseñado.
- Estudiar la viabilidad del uso de sistemas basados en agentes para soportar el reconocedor.

Para la consecución de estos objetivos generales, se hace necesaria además la consecución de otros objetivos más concretos tal y como se detalla a continuación:

- Diseñar una arquitectura basada en sistemas multi-agente que soporte el reconocimiento de entidades gráficas, convencionalismos gráficos y símbolos multi-icónicos, esto es, reconocimiento de entidades y símbolos empleados en los bocetos de ideación y prescriptivos, tales como la detección de restricciones geométricas o dimensionales, comandos gestuales, secciones, tipos de línea, etc., para su posterior conversión a un modelo 3D. Esta arquitectura debe de soportar bocetos o *sketches* multi-*stroke*.
- Establecer las pistas (también denominadas *'cues'*) básicas de los trazos como las características relevantes que se pueden extraer de un trazo simple, como por ejemplo el perímetro, la circularidad, etc.
- Establecer las *'cues'* de relación en los trazos compuestos como las características de relación entre los trazos simples, como son el paralelismo, concentricidad, tipos de línea, dimensiones, etc.
- Establecer las reglas sintácticas y semánticas que guiarán los criterios que determinen el reconocimiento. Este objetivo implica establecer todos los indicios o pistas de relación entre trazos (posiciones relativas, contexto, etc.) para comprobar si es un posible rayado, tipo de línea o cualquier otro símbolo compuesto.

La arquitectura propuesta deberá ser validada para comprobar su robustez y demostrar que cumple con los requerimientos esenciales de un entorno de bocetado digital, realizándose varias pruebas específicas que demuestren la viabilidad de la arquitectura propuesta y sus resultados.

En definitiva, se pretende desarrollar métodos y algoritmos novedosos que permitan proporcionar a un entorno de modelado paramétrico basado en bocetos la posibilidad de introducir la geometría y controlar la parametrización de forma directa, manuscrita y sin interferencia de menús. La idea se ilustra en la Figura 2.1 donde se muestra un ejemplo de modelado a partir de bocetos de ideación, en el que tras introducir un boceto (Figura 2.1a y Figura 2.1b) éste es reconocido automáticamente. Si posteriormente el usuario edita el boceto (Figura 2.1d) o dibuja un símbolo correspondiente a una restricción geométrica (Figura 2.1b, Figura 2.1c y Figura 2.1e), o a una restricción dimensional o cota (Figura 2.1f), estos símbolos o comandos gestuales son interpretados, capturándose así la intención de diseño del usuario y llevándose a cabo la acción correspondiente en ausencia de menús. Si el usuario desea posteriormente crear geometría tridimensional, podrá directamente dibujar los comandos de extrusión, revolución o barrido que serán interpretados y ejecutados como se muestra en el ejemplo de la Figura 2.2.

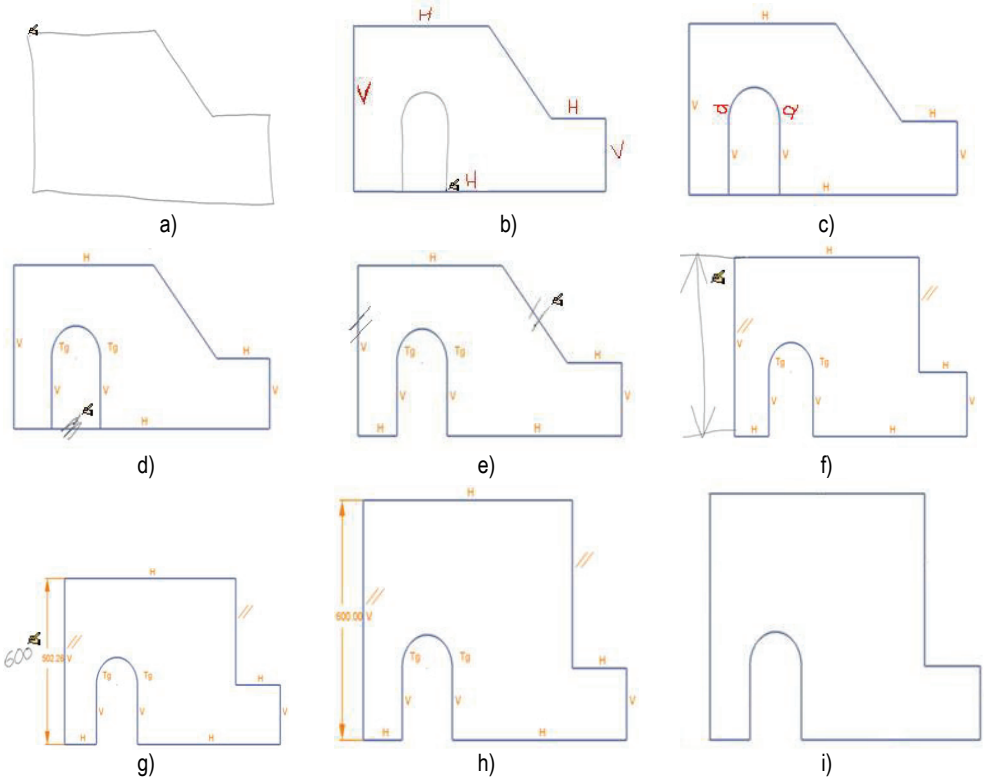


Figura 2.1. Simulación de reconocimiento de bocetos de ideación en un interfaz caligráfico

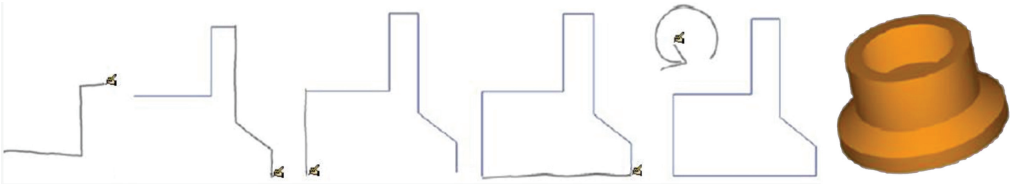


Figura 2.2. Simulación de creación de geometría de bocetos de ideación en un interfaz caligráfico

Capítulo 3:
EL ESTADO DEL ARTE

3. EL ESTADO DEL ARTE

En este capítulo se describen los trabajos más recientes o destacados en el campo del reconocimiento, en el campo de los algoritmos de detección de vértices para la segmentación de bocetos o *sketches*, en el campo de los sistemas basados en agentes aplicados a diversos procesos de visión por computador relacionados con el tema de la presente tesis, y en el campo de las aplicaciones de bocetado o *sketching* existentes actualmente. Este estado del arte se ha estructurado de esta manera para mostrar las carencias actuales de los algoritmos desarrollados o de las técnicas de programación utilizadas hasta el momento, y las soluciones que se proponen en esta tesis.

En primer lugar se muestran trabajos donde los algoritmos desarrollados, muchas veces basados en técnicas de análisis de imagen, para la clasificación de símbolos o gestos y su posterior interpretación concluyen con resultados no satisfactorios y faltos de precisión o repetibilidad. En segundo lugar se trata el tema de la segmentación, entendiéndose por segmentación la aproximación de una forma bocetada a las primitivas que la componen, siendo éste un tema clave ya que de ella dependen los procesos posteriores del análisis del boceto. En tercer lugar se analizan otras técnicas de programación, ya que se ha demostrado que los métodos tradicionales son poco flexibles y fiables en lo que a resultados de reconocimiento se refiere. Con los métodos tradicionales las decisiones son rígidas y poco adaptativas, lo que muchas veces provoca errores en la decisión de los algoritmos diseñados. Es aquí donde se contempla la programación con agentes, ya que, por la propia definición de agente, estos permiten implementar algoritmos más complejos de toma de decisiones consensuada y adaptativa teniendo en cuenta otra información del contexto que por métodos tradicionales resultaría más complejo o casi imposible. En cuarto y último lugar, se han buscado aplicaciones o herramientas comerciales de *sketching* para mostrar el estado actual de las mismas y hasta donde llegan actualmente, probando que no existen aplicaciones de *sketching* como tales que den servicio al diseñador en las primeras fases del diseño, y menos aún que lo bocetado pueda reutilizarse para fases posteriores del diseño, existiendo una desconexión completa entre este tipo de herramientas y las aplicaciones CAD 2D o 3D, y dando sentido a lo que se plantea acometer en esta tesis.

3.1 Aplicaciones de reconocimiento

Como ya se ha comentado previamente en el capítulo de introducción, la mayoría de las aplicaciones CAD interactúan con el usuario a través de lo que se ha dado a conocer como el paradigma WIMP (Windows, Icons, Menus, Pointing Device). Sin embargo este tipo de interfaz resulta muy estricto para las primeras fases del diseño conceptual, por lo que el papel sigue constituyendo el medio más utilizado actualmente por los ingenieros, arquitectos y otros diseñadores para sus bocetos iniciales, tal y como demuestra el estudio realizado por Company et al. [CCN06].

Los nuevos dispositivos portátiles en los que se dispone de una pantalla digitalizadora y un lápiz, como los Tablet-PCs o los asistentes personales PDAs, han surgido como

instrumentos de dibujo estándares y han supuesto un importante avance en el desarrollo de nuevos interfaces de usuario, conocidos también como interfaces caligráficos. Muestra de ello es el sistema CIGRO propuesto por Contero et al. [CNJ03] donde se aporta un interfaz caligráfico de instrucciones reducidas para la creación de objetos poliédricos mediante bocetos.

Las mejoras que ofrece el empleo de estos interfaces posibilita el desarrollo de aplicaciones muy diversas. Por ejemplo, Lopes et al. [LCS09] proponen una solución para la creación de comics online como si se realizara en formato papel, combinando grandes posibilidades de edición, técnicas de dibujo asistido por ordenador y un interfaz caligráfico que permite reutilizar elementos introducidos previamente. De igual manera, Sharf et al. [SLS07] introducen un método de reconstrucción de superficies en función de su topología que requiere la intervención del usuario mediante ciertos trazos para tomar las decisiones correctas en regiones donde la topología del modelo no se puede deducir de forma automática con un grado de confianza razonable. Por otro lado, Wu et al. [WTB07] presentan una sencilla aplicación para definir formas en 3D a partir de una única vista empleando "paletas de forma", esto es, se dibuja una primitiva 2D en un vista bi-dimensional y se especifica posteriormente su orientación 3D dibujando la primitiva correspondiente en la paleta de forma. Otros autores como Mori e Igarashi [MI07] desarrollan un sistema interactivo, denominado Plushie, que permite a usuarios inexpertos diseñar sus propios muñecos de trapo mediante bocetos, implementando comandos gestuales que permiten recortar o alargar partes del modelo existentes. Nealen et al. [NIS07] también presentan su sistema FiberMesh para el diseño de superficies de forma libre mediante curvas 3D bocetadas.

Sin embargo, uno de los usos más importantes de los que podemos dotar a un interfaz caligráfico es el de proveer a un sistema CAD de la capacidad de reconocer de forma automática ciertos símbolos bocetados a mano para utilizarlos como comandos de un software concreto, en lugar de tener que recurrir al uso de menús complejos y poco intuitivos.

El reconocimiento de símbolos resulta una tarea muy compleja si tenemos en cuenta que un mismo símbolo se puede dibujar de forma muy diferente según el usuario que lo bocete, variando incluso la orientación y el tamaño. Además, el orden seguido por cada usuario durante la fase de bocetado puede influenciar en el resultado final del reconocimiento. Puntos finales de línea que no coincidan con los puntos iniciales de otra línea, o líneas que terminan interceptando con otras líneas pueden ser algunos de los problemas principales que podemos encontrarnos. Debido a estos inconvenientes muchos algoritmos de reconocimiento resultan poco robustos o presentan ambigüedades en sus decisiones, desventajas que quedan recogidas en Hong-Kong et al. [HLL02].

Para minimizar la influencia de estos problemas se puede recurrir al empleo de técnicas basadas en el procesamiento de imágenes digitales, tales como filtros de convolución u operaciones morfológicas. Sin embargo, la descripción de forma a través de características como la longitud, la anchura, el perímetro, el área, los momentos de inercia, el *bounding box*, etc., presenta el inconveniente de que se obtienen resultados similares para distintos tipos de formas, siendo además dependientes del tamaño y la orientación de las mismas [BAM03, KS05]. Por ello, para solucionar problemas de escala u orientación en el trazado de los

símbolos deben aplicarse técnicas basadas en características invariantes [KS02], como los descriptores de Fourier [GW02 y KSP95], que han sido usados frecuentemente para la comprensión de información en imágenes, la clasificación de formas regulares [MA05] o el reconocimiento de caracteres [CBK05]. Otro ejemplo de característica invariante son los momentos geométricos, donde podemos encontrar el conjunto de descriptores invariantes obtenidos por Hu [Hu62]. Este conjunto de momentos geométricos invariantes han sido ampliados a juegos más grandes [WS99] y a otras formas [DBM77, LP98 y HN04]. Como ejemplo del uso de estos momentos citar el algoritmo de procesamiento de imágenes desarrollado por Zion [ZSK99] para clasificar imágenes de pescados de especies diferentes y empleado en piscifactorías de agua dulce.

De esta manera, Zang et al. [ZL02] emplean descriptores de Fourier para recuperar imágenes de bases de datos, consiguiendo tanto una buena representación como una normalización sencilla, siendo a su vez invariantes al escalado y a la rotación. Los autores prueban y comparan diferentes descriptores de Fourier usando para ello una base de datos de formas típicas, concluyendo que la distancia al centroide es la mejor característica de forma en términos de robustez, complejidad de cómputo y velocidad de convergencia de sus series de Fourier.

Otros ejemplos de métodos que recurren al empleo de los descriptores de Fourier son la detección del movimiento de la mano del usuario en un sistema de realidad aumentada [LS04], la clasificación de huellas digitales mediante un análisis discriminante no-lineal [PP05], o el reconocimiento de la trayectoria de gestos de la mano [HE04].

Así pues, el método de reconocimiento propuesto en la presente tesis está basado en la extracción y el análisis de descriptores morfológicos de forma, como el perímetro, la circularidad y los momentos de Hu, y de descriptores invariantes frente a la posición, escala y orientación, como los descriptores de Fourier de dos características de forma del gesto/símbolo: la distancia de cada punto al centroide del gesto [KL07] (conocida también como "*radius signature*") y el ángulo entre dos puntos consecutivos [KSS86 y Wol90] (conocido como "*cumulative versus turning angle signature*") de la forma bocetada.

Las siguientes tablas muestran un resumen de los trabajos mencionados y de los descriptores empleados por cada uno.

Tabla 3.1. Tabla resumen de los principales descriptores empleados y su aplicación (I)

Referencia	*Descripción*	*Descriptores*	*Clasificación*
Hu [Hu62]	Reconocimiento de patrones visuales mediante momentos invariantes	Momentos de Hu	-
Dudani et al. [DBM77]	Reconocimiento de modelos de aviones en imágenes digitales	Varios momentos invariantes	Análisis discriminante Bayesiano
Kalvin et al. [KSS86]	Técnica para reconocimiento 2D de objetos solapados	Atributos característicos del objeto que conforman una 'huella'	Template-matching
Wolfson [Wol90]	Se presentan dos algoritmos encargados de encontrar la subcurva común más larga entre dos curvas 2-D, esto es, tratan de encajar una curva en la otra como si de un engranaje se tratara.	Varias firmas de forma, entre ellas el ángulo entre dos puntos consecutivos	String-matching
Kauppinen et al, [KSP95]	Comparación de los métodos de clasificación de forma basados en modelos de autoregresión y en los descriptores de Fourier del contorno cerrado, usando una base de datos de imágenes de letras y aviones.	Curvatura, distancia al centro, coordenadas complejas...	Clasificador no paramétrico (kNN)
Liao & Pawlak [LP98]	Análisis detallado de la precisión de los momentos de Zernike en cuanto a errores de discretización y poder de reconstrucción	Momentos de Zernike (ZMs)	-
Wong & Siu [WS99]	Aproximación que permite implementar una estructura de filtros digitales para acelerar el cálculo de momentos	Transformada Z de las series de Maclaurin	-
Zion et al. [ZSK99]	Algoritmo para procesamiento de imágenes capaz de distinguir entre 3 especies de peces dentro de una piscifactoría	Momentos invariantes junto con consideraciones geométricas	Red neuronal
Kan & Srinath [KS02]	Reconocedor invariante de caracteres alfanuméricos de distinto tamaño	Momentos de Zernike (ZMs), pseudo-Zernike (PZMs), y momentos ortogonales de Fourier-Mellin (OFMMs)	Vecino más proximo (NN), mínima distancia media (MMD), perceptron multicapa (MLP) y red neural probabilística (PNN)
Zang & Lu [ZL02]	Estudio de varios descriptores de Fourier aplicados a la extracción de la forma	Descriptores de Fourier de varias firmas de la forma (distancia al centroide, área, longitud de la cuerda,etc)	-

Tabla 3.2. Tabla resumen de los principales descriptores empleados y su aplicación (II)

Referencia	*Descripción*	*Descriptores*	*Clasificación*
Blasco et al. [BAM03]	Estimación online de la calidad de naranjas, melocotones y manzanas mediante técnicas de visión artificial según unos atributos de calidad, como tamaño, color, posición del tallo, etc.	Área, longitud, centroide, color	Análisis discriminante Bayesiano
Hse & Newton [HN04]	Método para reconocimiento online de símbolos bocetados basado en los momentos de Zernike	Momentos de Zernike (ZMs)	Vecino más próximo (NN), Mínima distancia media (MMD) y "Support Vector Machines" (SVM)
Harding & Ellis [HE04]	Aplicación basada en los descriptores de Fourier para el reconocimiento de trayectorias de gestos de la mano	Descriptores de Fourier aplicados a las coordenadas de la trayectoria de la mano	Red neuronal probabilística (PNN)
Licsar & Sziranyi [LS04]	Sistema de reconocimiento de posturas de la mano basado en visión	Descriptores de Fourier modificados	Vecino más próximo (NN)
Kara & Stahovich [KS05]	Reconocedor de símbolos dibujados a mano basado en el esquema de reconocimiento multi-capa	Coordenadas polares de los puntos	Template-matching
Mokhtarian & Abbasi [MA05]	Se afronta el problema de seleccionar el número mínimo de vistas de representación de los objetos de una base de datos de manera que posteriormente se pueda reconocer dicho objeto a partir de cualquiera de sus vistas seleccionadas.	Técnica CSS, momentos invariantes y desciptores de Fourier	Clasificador basado en el CSS
Chen et al. [CBK05]	Reconocedor de patrones que propone un descriptor invariante a la rotación	Ridgelets, wavelet cycle-spinning y transformada de Fourier	Se clasifica el patrón a la clase que posea la menor distancia entre todas las clases en la base de datos.
Park & Park [PP05]	Aproximación para clasificar huellas dactilares	Transformada discreta de Fourier y filtros direccionales	Análisis discriminante nolineal
Kunttu & Lepisto [KL07]	Sistema industrial de inspección de superficies que detecta defectos y guarda una imagen en una base de datos para su posterior tratamiento	Descriptor de Fourier de la firma polar (distancia de cada punto al centroide)	No clasifica, tan sólo detecta los defectos

Son múltiples las técnicas usadas en el reconocimiento de bocetos para detectar símbolos, diagramas, formas geométricas y otros comandos gestuales. A través de un discriminador clásico lineal, Rubine [Rub91] calcula características relacionadas con el ángulo inicial del gesto, la longitud y el ángulo de la diagonal de la caja que encierra dicho gesto, el perímetro, y otros para clasificar bocetos de un solo trazo, como dígitos, letras y comandos básicos. También basándose en características similares, Apte et al. [AVK93] desarrollaron un algoritmo capaz de reconocer formas geométricas dibujadas a mano alzada (rectángulos, triángulos, círculos, elipses, rombos y líneas) a partir de bocetos compuestos por varios trazos introducidos de forma consecutiva. El sistema de reconocimiento que implementan está basado en lo que denominan "filtros", que son ratios de características como el área, el perímetro, el convex-hull, etc. Por otra parte, Jorge et al. [PJB04] utilizan porcentajes para los ratios del área y el perímetro para detectar formas como círculos, rectángulos, triángulos, líneas, etc.

Cabe mencionar el trabajo realizado por Hammond et al. [PH08 y HEP08], donde se implementa el reconocedor PaleoSketch capaz de clasificar entre ocho formas primitivas, así como la combinación de estas, empleando con éxito para ello la distancia normalizada entre direcciones extremas (NDDE) y el ratio de cambio de dirección (DCR).

Gross [Gro94] describe un prototipo para el reconocimiento de grafos que puede ser interpretado en el contexto de distintos dominios de diseño, como planos de edificios, dispositivos mecánicos o circuitos eléctricos. El reconocedor emplea una matriz de 3x3 para buscar esquinas y líneas que se corten, así como otros ratios que posteriormente se comparan con varias tablas de símbolos.

Otras características que permanecen invariantes con la rotación, como el convex-hull, el perímetro y varios ratios del área, son empleadas por Fonseca et al. [FJ00]. Xiangyu et al. [XWJ02] reconocen formas geométricas simples (como cuadriláteros, elipses, triángulos, pentágonos y hexágonos) y las regularizan (las ajustan a la intención del usuario), realizando previamente una etapa de pre-procesado de la forma bocetada para suprimir el ruido en los puntos introducidos. La tarea de clasificación se lleva a cabo mediante un algoritmo iterativo, que establece un número de vértices de una forma poligonal predeterminada, y calcula la distancia media desde los vértices de la forma predeterminada a los vértices del trazo introducido.

La siguiente tabla comparativa muestra un resumen de las aportaciones de los principales reconocedores que se han tratado.

Tabla 3.3. Tabla comparativa de los principales reconocedores de bocetos

Referencia	*Descripción y características*	*Pre-procesado*	*Segmentación*	*Clasificación*	*Secuencia de bocetado estricta*	*Trazos Múltiples*	*Otros*
Rubine [Rub91]	Reconocedor de trazos simples que extrae algunas características del trazo y realiza una clasificación lineal con la base de datos creada previamente	NO	NO	Discriminador clásico lineal	SI	No	
Apte et al. [AVK93]	Reconocedor multi-trazo que permite detectar las entidades: rectángulo, elipse, círculo, rombo, triángulo y línea. Para ello usa unos filtros calculados sobre los puntos del trazo introducido.	NO	NO	Se calcula el valor de varios filtros y en función de unos umbrales se determina la entidad	No	SI	No se puede dibujar la entidad en un único trazo, sino que hay que hacerlo en varios
Gross [Gro94]	Reconocedor de grafos aplicable a varios dominios, como diagramas de arbol y pollíneas	NO	Detección de esquinas donde exista concentración de puntos y baja velocidad	Emplea una matriz de 3x3 sobre el trazo para compararlo con la base de datos	SI	SI	Requiere un entrenamiento entre 2 y 4 horas por usuario
Fonseca et al. [FJ00]	Mejora del reconocedor de Apte [AVK93] que además de detectar primitivas multi-trazo (rectángulo, círculo, rombo, triángulo, etc...) reconoce comandos gestuales realizados en un único trazo (borrar, copiar, deshacer, mover y línea ondulada).	NO	NO	Se calculan varios filtros basados en el convex-hull que encierra el trazo introducido, comparándolos posteriormente con una base de datos creada	En primitivas NO, en comandos SI	En primitivas SI, en comandos NO	Se emplea lógica difusa para abordar aquellos casos que presentan imprecisión o ambigüedad
Xiangyu et al. [XWJ02]	Reconocedor de formas geométricas simples (triángulo, cuadrilátero, elipse, pentágono y hexágono) que regulariza el trazo a la intención del usuario	SI	NO	Tras obtenerse la forma convexa del trazo introducido se combinan los vértices próximos y en función del número definitivo de estos se determina la clasificación (triángulo 3, pentágono 5,etc...)	NO	NO	Una vez reconocida la forma geométrica, redibuja el trazo introducido para que coincida con la forma detectada
Jorge et al. [PJB04]	Mejora del reconocedor de Rubine [Rub91], soportando ahora multi-trazo y otras características	SI	NO	Discriminador clásico lineal que ahora usa también probabilidades y otras características, como la distancia de Mahalanobis	No	SI	Para evitar ambigüedades ofrece una ventana para elegir la intención del usuario
Hammond et al. [PH08] y [HEP08]	Reconocedor que detecta 8 tipos de primitivas, permitiendo dibujar varias seguidas pero en un único trazo. Implementa mejoras sobre Rubine [Rub91]	SI	Detecta esquinas donde la curvatura sea elevada y divide el trazo en sub-trazos	Emplea dos características nuevas (la NDDE y el DCR) para clasificar cada sub-trazo entre las 8 primitivas posibles	NO	NO	Para evitar ambigüedades ofrece una ventana para elegir la intención del usuario

3.2 La segmentación del trazo

La detección de esquinas supone un componente fundamental a la hora de crear interfaces caligráficos. Dado que se usa a menudo en la segmentación del trazo esbozado para descomponerlo en primitivas más simples, es uno de los pasos más importantes en el proceso

de reconocimiento e interpretación de bocetos a mano alzada. Muestra de ello es el método de comparación propuesto por Company et al. [CVP09] para evaluar los algoritmos de segmentación y su capacidad de emular la percepción humana de los bocetos, minimizando las diferencias entre lo que el algoritmo detecta y lo que el usuario realmente percibe.

Así pues, la detección de vértices se emplea también en el reconocimiento de gestos más complejos, tales como el borrado de un trazo mediante el dibujo de un garabato, el redondeo de una expresión matemática escrita a mano para invocar un reconocedor [LZ04], o como parte de un conjunto de características en un algoritmo de aprendizaje automatizado [LJZ07]. Dada la utilidad de encontrar los vértices en los interfaces caligráficos y el hecho de que encontrar dichos vértices de forma precisa en muchos casos ayudará a determinar la precisión global de un reconocedor de bocetos, es esencial disponer de una técnica precisa para la detección de vértices.

Hasta la fecha son varios los algoritmos desarrollados para la detección de vértices en reconocedores de bocetos. Una de estas aproximaciones busca los valores de curvatura y velocidad que sobrepasan un umbral definido, tomando estos puntos como vértices del trazo [Sta04]. En el caso de Sezgin et al. [SSD01] buscan aquellos vértices donde se da la máxima curvatura y la mínima velocidad. Una vez el sistema combina el conjunto de esquinas candidatas a partir de los datos de curvatura y velocidad, se busca una serie de ajustes híbridos para detectar las esquinas o vértices reales.

Un caso aparte es el de Qin et al. [QWJ01], que estudian los valores de la curvatura, la velocidad y la aceleración. Emplean la velocidad y la aceleración para normalizar el trazo antes de ser procesado en la etapa de segmentación. Posteriormente, buscando en la dirección del trazo, se marcan los puntos de esquina para finalmente aproximar cada parte del trazo en función de su circularidad en líneas, arcos o splines.

Kim y Kim [KK06] proponen una nueva medida de la curvatura en su algoritmo de detección de vértices. Evitan la necesidad de calcular la longitud de los arcos ya que realizan un re-muestreo de la entrada inicial, de modo que los puntos re-muestreados contiguos poseen una distancia constante. Esta simplificación permite definir el cálculo de la curvatura como el cambio de la dirección en un punto dado. Otra técnica empleada para la detección de esquinas es la aproximación multi-escala, esto es, se calculan los descriptores del trazo introducido empleando varios niveles de suavizado. Rattarangi y Chin [RC92] suavizan los puntos del *stroke* mediante una escala Gausiana variable, eliminando el ruido y mejorando así el proceso de detección de esquinas. Este trabajo lo mejoran Lee et al [LSC95] al recurrir a la transformada wavelet en su detector de vértices multi-escala. Sezgin [SD06] también recurre a esta técnica para mejorar la detección de vértices implementada anteriormente [SDD01], aplicando varios filtros de suavizado a los datos de la curvatura y escogiendo como esquinas aquellos puntos que mantienen los valores máximos en todas las escalas.

En la detección de puntos de esquina se ha llegado a combinar la segmentación y el reconocimiento de primitivas de forma conjunta. Por ejemplo, Yu [Yu03] lleva a cabo el reconocimiento empleando en este caso la técnica de realzado Mean Shift a las gráficas de dirección y curvatura del trazo introducido. Todo esto unido a una clasificación recursiva de las

entidades entre los vértices hallados (denominado tramo o '*stretch*') asegura un reconocimiento correcto de los trazos.

Así pues, todos los algoritmos mencionados dependen de un conocimiento matemático avanzado. Sin embargo, en 2008 Wolin et al. [WEH08] propusieron ShortStraw, un sencillo y eficiente algoritmo de detección de vértices que demostró ser altamente preciso tanto en el número de esquinas detectadas correctamente como en las pruebas de "todo o nada" de precisión de vértices (esto se conoce como "*all-or-nothing accuracy*", que implica que si la forma está correctamente segmentada en su totalidad se cuenta como acierto y en otro caso como fallo), introduciendo el concepto de "*straws*", que dependen de una ventana de tamaño constante para examinar las partes contiguas de un trazo, lo cual está en contraste con el detector de vértices de Teh y Chin[TC89], que usa una ventana variable para cada punto durante la detección de las esquinas. Posteriormente, el algoritmo ShortStraw fue mejorado por Xiong y La Viola [XL09], que presentaron un nuevo algoritmo de detección de vértices, IStraw, que corrige algunas limitaciones intentando no aumentar la complejidad computacional. Además, extiende ShortStraw para tratar con trazos que contengan curvas. Los resultados obtenidos mostraron mejoras significativas en la detección de "todo o nada" de vértices esquina comparado con ShortStraw para trazos formados únicamente por líneas rectas. Las pruebas también mostraron que IStraw tenía mayor precisión "todo o nada" detectando vértices que si se usaba ShortStraw sólo o en combinación con la extensión de búsqueda de vértices en formas que contuvieran curvas.

Aunque los dos algoritmos previos [WEH08, XL09] ofrecen buenos resultados en la detección de vértices, y han sido presentados como los dos métodos más precisos y sencillos en comparación con el resto de métodos existentes, todavía presentan algunos inconvenientes que se deben solucionar, como por ejemplo la ausencia de detección de los vértices tangentes. Este inconveniente lo intentan solucionar Pu y Gur [PG09], los cuales aplican un método de forma razonablemente robusta usando la función del radio, y aunque alcanzan buenos resultados en la detección de este tipo de vértices, el número de falsos positivos es muy elevado (alrededor del 25%), además de no distinguir entre vértices esquina y tangentes en los resultados. Este es uno de los puntos clave que se ha trabajado en esta tesis, aportando un nuevo método de segmentación al que se ha llamado TCVD (*Tangent Corner Vertices Detection*), el cual sigue siendo simple computacionalmente, esto es, permite trabajar sobre una aplicación en línea y mejora la detección de vértices esquina, a la vez que incorpora de manera muy efectiva la detección de vértices tangentes tan útiles en este tipo de aplicaciones y hasta ahora no detectados correctamente en los trabajos encontrados en la literatura.

Las siguientes tablas muestran a modo de resumen las principales características de los trabajos mencionados.

Tabla 3.4. Tabla comparativa de los principales segmentadores de bocetos (I)

Referencia	*Descripción*	*Pre-procesado*	*Características analizadas*	*Segmentación*	*Clasificación*	*Trazos Múltiples*	*Otros*
Teh y Chin[TC89]	Detector de vértices en trazos simples que usa una ventana de tamaño variable para calcular de forma local la curvatura	SI	Curvatura	Detecta esquinas en aquellos puntos que poseen la máxima curvatura local, calculada mediante una ventana variable aplicada en cada punto	No clasifica, tan sólo detecta vértices	No	Realiza una revisión y se compara con los principales detectores de vértices hasta 1989
Rattarangsi and Chin [RC92]	Detector de vértices basado en el espacio de escala Gausiano, esto es, el valor máximo de los valores absolutos de la curvatura en varias escalas	SI	Curvatura y escala gausiana variable	Detecta esquinas en puntos que mantienen la máxima curvatura en la mayoría de las escalas	No clasifica, tan sólo detecta vértices	No	
Lee et al. [LSC95]	Detección de vértices multi-escala que recurre a la transformada wavelet	SI	Orientación (dirección en cada punto)	Tras obtener la transformada wavelet de la orientación de los puntos del contorno, se consideran esquinas aquellos puntos con valor máximo	No clasifica, tan sólo detecta vértices	No	Mejora el trabajo de Rattarangsi and Chin [RC92]
Qin et al. [QWJ01]	Segmentador de bocetos en líneas rectas, arcos circulares y splines	SI	Curvatura, velocidad y aceleración	Detección de esquinas donde exista concentración de puntos y baja aceleración	Mediante umbrales de linealidad y circularidad	No	
Sezgin et al. [SSD01]	Segmentador de bocetos en líneas y arcos	SI	Curvatura y velocidad	Detección de esquinas en puntos con máxima curvatura y mínima velocidad	Aproximación por mínimos cuadrados	No	También reconoce algunas formas básicas (círculos, rectángulos, etc)
Yu [Yu03]	Segmentador que intenta aproximar el trazo a una primitiva (línea, arco, círculo o elipse) y si falla lo divide en dos, aplicando de forma recursiva el proceso hasta que todos los subtrazos sean clasificados	SI	Dirección y Curvatura	Se aplica la técnica de realzado Mean Shift a las gráficas de dirección y curvatura, y aquellos puntos que cumplan determinadas condiciones son marcados como vértices	Estudia la pendiente de la dirección y varios ratios entre el trazo y la primitiva propuesta	No	Primer método que intenta, sin éxito, calcular vértices tangentes

Tabla 3.5. Tabla comparativa de los principales segmentadores de bocetos (II)

Referencia	*Descripción*	*Pre-procesado*	*Características analizadas*	*Segmentación*	*Clasificación*	*Trazos Múltiples*	*Otros*
Stahovich [Sta04]	Segmentador de bocetos en líneas y arcos	SI	Curvatura y velocidad	Detección de esquinas en puntos con máxima curvatura y mínima velocidad	Se selecciona la primitiva (línea o arco) que mejor se adapte al tramo	SI	Tras clasificar se refunden las entidades similares contiguas
Kim & Kim [KK06]	Detector de vértices en trazos simples a partir de una estimación de la curvatura	SI	Estimación de la curvatura mediante convexidad local y monotonicidad local	Detección de esquinas en puntos con máxima curvatura, tomando las distintas estimaciones de la curvatura en lugar de la clásica	No clasifica, tan sólo detecta vértices	No	
Sezgin and Davis [SD06]	Detector de vértices que usa la teoría del escalado múltiple, sin recurrir a ningún tipo de umbral	SI	Curvatura y velocidad	Se aplican varios filtros de suavizado a los datos de la curvatura y se escogen como esquinas aquellos puntos que mantienen los valores máximos en todas las escalas	No clasifica, tan sólo detecta vértices	No	No funciona con curvas, sólo con trazos formados por líneas rectas
Wolin et al. [WEH08]	Detector de vértices en trazos lineales que usa una ventana de tamaño fijo para calcular la distancia 'straw' de los puntos remuestreados	SI	Posición	Detecta como vértices aquellos puntos cuya distancia 'straw' esté por debajo de un determinado valor. Posteriormente hace un refinado para eliminar falsos positivos	No clasifica, tan sólo detecta vértices	No	No funciona con curvas, sólo con trazos formados por líneas rectas
Xiong y LaViola [XL09]	Detector de vértices que mejora considerablemente el propuesto por Wolin et al. [WEH08], soportando ahora también curvas	SI	Posición y velocidad	Detecta como vértices aquellos puntos cuya distancia 'straw' (redefinida) esté por debajo de un determinado valor (también redefinido). Posteriormente se ejecutan varios refinamientos para eliminar falsos vértices	No clasifica, tan sólo detecta vértices	No	Acepta trazos con curvas
Pu y Gur [PG09]	Detector de vértices que estudia los valores de radio en cada punto, sin recurrir a la velocidad ni la curvatura	SI	Radio	Trata de obtener el número mínimo de vértices que minimizan la función de energía de una curva de ajuste implícito	No clasifica, tan sólo detecta vértices	No	25% de falsos positivos en vértices tangentes

3.3 Desarrollos con sistemas basados en agentes

Nuestro objetivo actual viene definido por un nuevo paradigma que: a) ayude a la interpretación de los bocetos realizados por los diseñadores de productos durante las etapas creativas del diseño, y b) sea capaz de ofrecer una solución escalable. Sin embargo, la mayoría de los reconocedores implementados poseen el inconveniente de que son poco robustos, no muy eficientes y rígidos. Es por esta razón que se hace imprescindible un cambio en la metodología para que estos reconocedores diferencien y clasifiquen los bocetos como lo haría un ser humano, esto es, utilizando información del contexto y siendo flexible en las decisiones intermedias y finales. De ahí que nuestros objetivos se definan a partir de una estructura basada en agentes, para servirnos de su flexibilidad y autonomía.

Nuestra estructura está definida en tres niveles: a) un nivel inferior donde actúan los agentes básicos, encargados de clasificar los trazos simples introducidos y distribuir algunas tareas de reconocimiento, b) un nivel intermedio donde los agentes primitivos usan los resultados obtenidos para encontrar el significado sintáctico a cada trazo, y c) un nivel superior que contiene a los agentes combinados encargados de la obtención del significado semántico del conjunto de trazos.

Según Wooldridge & Jennings [WJ95] estos agentes deben ser autónomos (actuar por sí mismos sin intervención humana), reactivos (percibir su entorno y responder a tiempo a los cambios que se produzcan en él), pro-activos (deben poder tomar la iniciativa y marcarse unos objetivos) y sociales (deben interactuar con otros agentes para completar juntos un problema común, así como ayudar al resto en sus tareas).

Aunque los agentes se han utilizado sobre todo en aplicaciones como la simulación de procesos, y gestión y control de procesos, esta técnica se está utilizando cada vez más en procesos que intervienen en entornos de visión artificial, análisis de imagen y tareas de reconocimiento, como es el caso de la interpretación de bocetos para soportar interfaces caligráficas. Existen varias propuestas en cuanto a la estructura que un sistema multi-agente debería implementar, como Rogers et al. [RRS00] que proponen el sistema colaborativo “Impact” para la creación de aplicaciones basadas en agentes, o Hammond y Davis [HD05] que definen el lenguaje de bocetado “LADDER” para desarrolladores de interfaces de usuario, así como Julian et al. [JCR02] que presentan la arquitectura multi-agente a la que llaman “ARTIS”, la cual garantiza la respuesta entre agentes satisfaciendo las restricciones temporales críticas de un entorno en tiempo real.

Son varios los trabajos de investigación donde se ha recurrido al empleo de Agentes, abarcando un campo de aplicación muy amplio desde algoritmos para el trazado óptimo de recogida de basura [PM09], hasta reconocedores faciales [CS06] y aplicaciones software para gestión de reservas en el área turística mediante Iphone [PAG09], aunque poco a poco su uso se está empezando a extender al proceso de reconocimiento en interfaces naturales.

Juchmes et al. [JLA05] justifican el empleo de los sistemas multi-agente para la interpretación de bocetos dibujados a mano basándose en tres motivos:

1) El boceto pertenece a un entorno dinámico e impredecible: los agentes desconocen si el siguiente trazo que se dibuje pertenecerá a la secuencia actual o por el contrario se trazará en una parte completamente diferente del boceto, o si será un trazo, un comando gestual o simplemente una selección de algún elemento existente.

2) Un sistema multi-agente permite descomponer el problema existente: la interpretación de un boceto supone un objetivo muy complejo, el cual se puede dividir en otros sub-objetivos más sencillos. El reconocimiento de los símbolos de forma individual resulta menos tedioso, surgiendo la complicación cuando existen varios elementos que a su vez poseen distintas relaciones entre ellos.

3) Un sistema multi-agente siempre está dispuesto al cambio: la arquitectura de un sistema multi-agente está diseñada para adaptarse fácilmente a los cambios. Como el número de elementos gráficos a reconocer es prácticamente ilimitado, se requiere de una arquitectura software modular que permita conectar o desconectar los distintos elementos sin que esto repercuta en el sistema completo.

A su vez, presentan el sistema EsQUIsE como una herramienta de interpretación de bocetos en la etapa conceptual del diseño arquitectónico, empleando un interfaz caligráfico (lápiz y tableta digital). El sistema es capaz de capturar líneas e interpretarlas en tiempo real, componiendo de forma progresiva los modelos tecnológico y funcional del edificio que se está diseñando. El sistema presenta una estructura compuesta de una etapa de reconocimiento gráfico, seguida de un análisis semántico de los elementos detectados.

Por otro lado, Achten & Jessurum [AJ02] presentan un modelo teórico de plataforma multi-agente capaz de reconocer unidades gráficas en un dibujo técnico. La estructura propuesta consta de agentes singulares especializados en el reconocimiento de unidades gráficas, los cuales se comunican y establecen mecanismos de negociación que permitirán resolver casos de ambigüedad. Sin embargo, tan sólo proponen la estructura del sistema, sin llegar a implementarla.

Una estructura similar es presentada por Azar et al. [ACD06], que proponen e implementan un interfaz multimodal para la interpretación de bocetos arquitectónicos basado en una arquitectura multi-agente. En este caso el sistema está compuesto por varios agentes gráficos encargados del reconocimiento de elementos gráficos básicos (como líneas y círculos), y de otros más complejos (como puertas, escaleras, etc.), característicos del diseño arquitectónico. También implementan una serie de agentes “vocales” encargados de reconocer las dimensiones y comandos introducidos mediante un micrófono.

Otros autores como Mackenzie et al. [MA03] optan por aplicar una arquitectura multi-agente a un clasificador de bocetos de animales dibujados por niños, tales como un perro, un gato, un pez, etc. Dicho clasificador consta de dos etapas principales: la primera, donde se reconocen los elementos gráficos según su geometría y se clasifican como partes del cuerpo animal (cabeza, pata, cola, etc.) según una base de datos previamente establecida; y una segunda etapa donde se analizan las relaciones entre las entidades detectadas y se clasifica el boceto introducido. Para ello se establece una “arena” donde se van colocan las características

extraídas del dibujo y desde donde cada agente busca una serie de características concretas que le permitan alcanzar su objetivo, definiendo así un nivel de confianza por cada agente.

En el caso de Flasinski et al. [FJM09], presentan un sistema multi-agente basado en una aproximación a un reconocedor de patrones sintáctico, empleado en esta ocasión para clasificar posturas de la mano en el lenguaje por signos polaco. Dada la gran variedad de formas de realizar dichos gestos en función del usuario, es necesario construir un alfabeto que pueda dividirse en varios sub-alfabetos y poder distribuir y asignar cada uno a varios agentes encargados de detectar si el gesto introducido les pertenece y clasificarlo.

Para obtener un buen reconocimiento resulta evidente que es crucial extraer la información del contexto, pues en muchas ocasiones nos permite además resolver ambigüedades. Este es el caso precisamente del reconocedor de diagramas UML bocetados propuesto por Casella et al. [CDM08], basado en una arquitectura multi-agente. El proceso de reconocimiento es llevado a cabo por una serie de "agentes inteligentes" que se encargan de reconocer los símbolos bocetados y de coordinarse entre sí con el fin de obtener una interpretación eficiente y precisa del boceto introducido. Cabe mencionar que precisamente la arquitectura propuesta por Casella et al. fue tomada como referencia para las implementaciones iniciales de la presente tesis.

Aparte de los reconocedores de bocetos también se recurre a los sistemas multi-agente para implementar otro tipo de reconocedores, como el caso de Rodin et al. [RBG04] que presentan un sistema de procesado de imágenes de ejecución paralela mediante el uso de agentes aplicado al campo de la Biología. Los agentes implementados los definen como "reactivos", esto es, cada agente posee un comportamiento muy sencillo que le permite tomar una decisión (encontrar un borde, una región, etc.) según su posición en la imagen y la información que le rodea. Estos comportamientos son asignados a los agentes de forma rápida y sencilla mediante un lenguaje de programación propio denominado oRis.

Chetty & Sharma [CS06] introducen una propuesta de sistema multi-agente para tratar el problema que plantea el reconocimiento facial. Dicha propuesta consta de un modelo estructural y funcional en varios niveles donde residen varios agentes que llevan implementados algoritmos de extracción de características, posibilitando así porcentajes de éxito más elevados incluso en condiciones del entorno algo complejas (cambios de luz, de fondo, exposición, etc.).

También cabe destacar la labor de Alhajj et al. [AE05], que desarrollan un sistema multi-agente destinado a tratar el problema de separar dígitos conectados por un mal bocetado, y el trabajo de Saeed [Sae00] que implementan mediante agentes una aplicación para reconocer caracteres arábigos, así como el sistema multi-agente "AgenTrac" desarrollado por Bryll et al. [BRQ05] basado en visión artificial y destinado al seguimiento de gestos en conversaciones (concretamente de las manos y la cabeza).

Las siguientes tablas muestran a modo de resumen una descripción de las distintas propuestas de sistema multi-agente que se han mencionado.

Tabla 3.6. Tabla resumen de los sistemas multi-agente mencionados (I)

Referencia	*Descripción*	*Observaciones*
Rogers et al. [RRS00]	Se propone IMPACT como una plataforma software común donde los usuarios puedan crear agentes desde diversas localizaciones, interactuando a su vez con otros agentes del sistema	La plataforma permite crear agentes, describir los tipos de datos tratados, las funciones del interfaz de aplicación, las acciones que los agentes pueden tomar, las restricciones de integridad que el agente debe poseer y el código que condiciona cuando el agente está obligado, permitido o prohibido a ejecutar una acción.
Saeed [Sae00]	Aplicación para reconocimiento de caracteres arábigos mediante agentes	El sistema consta sólo de tres agentes
Julian et al. [JCR02]	Se presenta la arquitectura SIMBA basada en agentes ARTIS como componente principal para el desarrollo de sistemas multi-agente en tiempo real	La arquitectura tiene en cuenta las restricciones temporales de cada agente a la hora de realizarse la comunicación entre estos, recurriendo para ello el estandar de comunicación FIPA
Achten & Jessurum [AJ02]	Se presenta un modelo teórico de plataforma multi-agente capaz de reconocer unidades gráficas en un dibujo técnico	Tan sólo se propone la estructura del sistema, sin llegar a implementarla
Mackenzie et al. [MA03]	Clasificador de bocetos de animales dibujados por niños implementado sobre una arquitectura multi-agente	El clasificador primero clasifica las partes del cuerpo animal (cabeza, pata, cola, etc.) según una base de datos previamente establecida, para luego analizar las relaciones entre las entidades detectadas y clasificar así el boceto introducido
Rodin et al. [RBG04]	Sistema de procesado de imágenes de ejecución paralela mediante el uso de agentes aplicado al campo de la Biología	Se implementa un lenguaje de programación propio denominado oRis que adjudica a los agentes implementados un comportamiento "reactivo", el cual le permite tomar una decisión (encontrar un borde, una región, etc.) según su posición en la imagen y la información que le rodea
Hammond & David [HD05]	El lenguaje LADDER permite a partir de la descripción de un dominio transformar dicha descripción automáticamente en un interfaz de reconocimiento de bocetos para dicho dominio.	Varios autores recurren al lenguaje LADDER para definir sus algoritmos de reconocimiento
Juchmes et al. [JL05]	Se presenta el sistema EsQUIsE como una herramienta de interpretación de bocetos en la etapa conceptual del diseño arquitectónico, mediante el empleo de un interfaz caligráfico	El sistema es capaz de capturar líneas e interpretarlas en tiempo real, componiendo de forma progresiva los modelos tecnológico y funcional del edificio que se está diseñando

Tabla 3.7. Tabla resumen de los sistemas multi-agente mencionados (II)

Referencia	*Descripción*	*Observaciones*
Alhajj et al. [AE05]	Sistema multi-agente destinado a tratar el problema de separar dígitos conectados por un mal bocetado	El sistema consta únicamente de dos agentes
Bryll et al. [BRQ05]	Sistema multi-agente basado en visión artificial y destinado al seguimiento de gestos en conversaciones (concretamente de las manos y la cabeza)	El sistema puede ser reconfigurado para seguir todo tipo de objetos en las secuencias de video
Azar et al. [ACD06]	Se propone e implementa un interfaz multimodal para la interpretación de bocetos arquitectónicos basado en una arquitectura multi-agente	El sistema está compuesto por varios agentes encargados del reconocimiento de elementos gráficos básicos (como líneas y círculos), y de otros más complejos (como puertas, escaleras, etc.), característicos del diseño arquitectónico
Chetty & Sharma [CS06]	Se introduce una propuesta de sistema multi-agente para tratar el problema que plantea el reconocimiento facial	Se plantea un modelo estructural y funcional en varios niveles donde residen varios agentes que llevan implementados algoritmos de extracción de características, posibilitando así porcentajes de éxito más elevados incluso en condiciones del entorno algo complejas (cambios de luz, de fondo, exposición, etc.)
Casella et al. [CDM08]	Reconocedor de diagramas UML bocetados basado en una arquitectura multi-agente	Se define una serie de "agentes inteligentes" encargados de reconocer los símbolos bocetados y coordinarse entre sí para obtener una interpretación eficiente y precisa del boceto introducido
Flasinski et al. [FJM09]	Sistema multi-agente para clasificar posturas de la mano en el lenguaje por signos polaco	Dada la gran variedad de formas de realizar dichos gestos en función del usuario, es necesario construir un alfabeto que pueda dividirse en varios sub-alfabetos y poder distribuir y asignar cada uno a varios agentes
Pérez & Matos [PM09]	Aplicación que calcula los caminos óptimos que debe seguir un camión de reciclaje de basura por las calles de Toro, en la provincia de Zamora	Se recurre al algoritmo ACS (Ant Colony System), afrontando el problema como si de una colonia de hormigas obreras se tratara
Palanca et al. [PAG09]	Aplicación software para la gestión de reservas en el área turística mediante Iphone (búsqueda de restaurantes según unos criterios establecidos, etc...)	Implementada sobre una plataforma multi-agente con soporte RDF (Resource Description Framework)

Como se ha visto, las aplicaciones basadas en agentes, esto es, que se han implementado sobre arquitecturas que contienen agentes, proporcionan ventajas obvias sobre las plataformas tradicionales de programación como son la flexibilidad, la organización de la información, la jerarquización, la autonomía del código implementado, la adaptatividad del código, el debate y consenso de las tomas de decisiones, la facilidad de valorar el contexto, etc. Es por estos motivos tan importantes por lo que se ha decidido implementar los métodos y técnicas de la arquitectura propuesta en esta tesis sobre una plataforma basada en agentes.

3.4 Aplicaciones comerciales y gratuitas de *sketching*

Actualmente existen varias aplicaciones en el mercado que permiten la creación de modelos simples en dos y tres dimensiones. Sin embargo muchas de ellas hacen uso del término "*sketching"* para designar el proceso de desarrollo conceptual, empleando una serie de botones e iconos que posibilitan las distintas acciones del programa. Este concepto no es del todo correcto, pues el término de "*sketch*" se atribuye normalmente a un boceto realizado a mano de forma rápida sin necesidad de botones ni restricciones que dificulten el plasmado de la idea que se posee de un determinado modelo o forma.

Así pues son diversas las aplicaciones comerciales que se pueden encontrar para el desarrollo de piezas y geometría tanto en dos como en tres dimensiones, como puede ser el caso del software Autocad [AutCad82], Pro/Engineer [ProEng88], Solidworks [SldWks95], Sketch Up! [SketUp00] y Catia [Catia77], entre otros, que por medio de botones y menús permiten la creación de cualquier modelo. Estas aplicaciones sin embargo distan mucho de la posibilidad de crear un boceto de forma rápida e intuitiva, requiriendo unos tiempos de aprendizaje elevados que conllevan que el usuario finalmente recurra al lápiz y papel de siempre durante la etapa conceptual del diseño. Cabe destacar que en el caso del software Catia de la empresa Dassault Systems, se detecta si se está ejecutando la aplicación en un Tablet PC y ofrece en este caso la posibilidad de usar una lista concreta de comandos gestuales bastante reducida. Sin embargo es frecuente que se requiera trazar el gesto varias veces hasta que lo reconozca correctamente, por lo que resulta en ocasiones más costoso que usar directamente el botón de la acción correspondiente.

Cabe destacar también el empleo de la librería Microsoft.Ink [JS02] para implementar en varios lenguajes de programación aplicaciones diversas que aprovechen las características de un Tablet PC, como la captura del puntero lápiz, la distinción de presión del lápiz sobre la pantalla, y otros parámetros interesantes.

Por otro lado, varios programadores desarrollan software de bajo coste, e incluso en ocasiones gratuito, que ofrecen la posibilidad de realizar bocetos en un Iphone [Iphon07], un Tablet PC [TablPc01], e incluso un ordenador mediante el empleo del propio ratón.

Este es el caso de la herramienta comercial "Sketches" [Sket2iP08] diseñada y programada para el iPhone y el iPod-Touch que permite dibujar y decorar imágenes con figuras de diversos tipos, así como escribir notas y diagramas de forma táctil.

Otra aplicación para el iPhone desarrollada por Autodesk es “SketchBook Pro” [SkeBook07], que permite dibujar de forma artística en un iPhone consiguiendo resultados bastantes espectaculares.

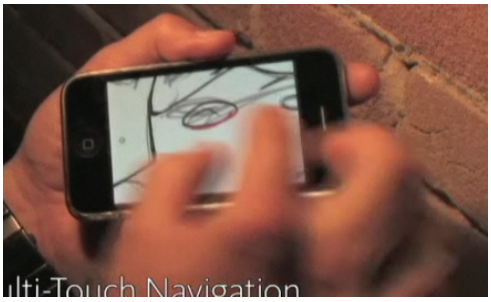


Figura 3.1. Ejemplo con “SketchBook Pro”

Un ejemplo sencillo de aplicación gratuita de bocetado puede encontrarse en la página web [SusApp07], donde se ofrece un interfaz para introducir un boceto que posteriormente queda almacenado en una base de datos, posibilitando que cualquier visitante de la página pueda consultarlo posteriormente.

En el caso de “Java Sketch” [JavSke97], se trata de otra aplicación web gratuita fruto del artículo [ZHH96]. Dicha aplicación web permite la creación de modelos en tres dimensiones mediante el dibujo de cubos y líneas.

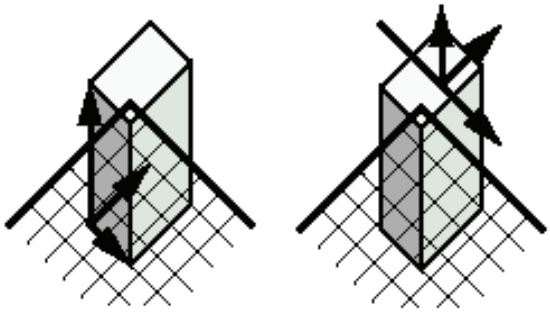


Figura 3.2. Ejemplo realizado con “Java Sketch”

Dado el carácter famoso adquirido por FaceBook en los últimos años, también existen aplicaciones como “FaceBook Sketch Me” [FBSkeMe09] que permite transformar una fotografía digital en un boceto para insertarlo posteriormente en el perfil de FaceBook.

Figura 3.3. Ejemplo en "FaceBook Sketch Me"

Otro tipo de aplicación aplicado a la fotografía es el "Sketch Master" [SkeMast05], que transforma las fotografías introducidas en bocetos dibujados a mano de forma bastante realista.

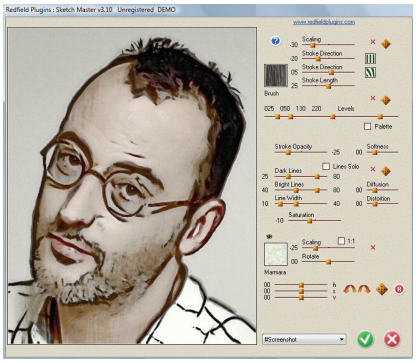


Figura 3.4. Ejemplo en "Sketch Master"

Hasta ahora, las aplicaciones comentadas ofrecen la posibilidad de crear un boceto de forma sencilla y táctil, pero dicho boceto es tratado como una mera imagen de píxeles, sin entrar en ningún tipo de detección de entidades, relación entre los elementos detectados, etc. Por este motivo el MIT llevó a cabo el desarrollo de la aplicación de bocetado sobre una pizarra digital denominada "Magic Paper" [MITSke07], donde los distintos elementos son reconocidos y asignados con unas propiedades físicas determinadas. Un ejemplo de dicho interfaz se puede apreciar en la Figura 3.5.

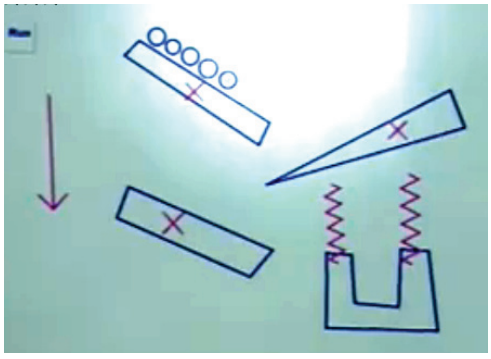


Figura 3.5. Ejemplo con "Magic Paper"

Paralelamente se ha desarrollado el software de bocetado de formas curvas "ILoveSketch" [ILovSke08], basado en [BBS08]. En esta aplicación se recurre para el trazado a la técnica del "*overtracing"*, siendo necesario dibujar varias veces la misma línea curva que se desea introducir. Los resultados obtenidos hasta la fecha por la aplicación son bastante prometedores, pudiendo visualizar varios videos y demos en la página web oficial del proyecto. Un ejemplo de las capacidades del programa se puede apreciar en la Figura 3.6.

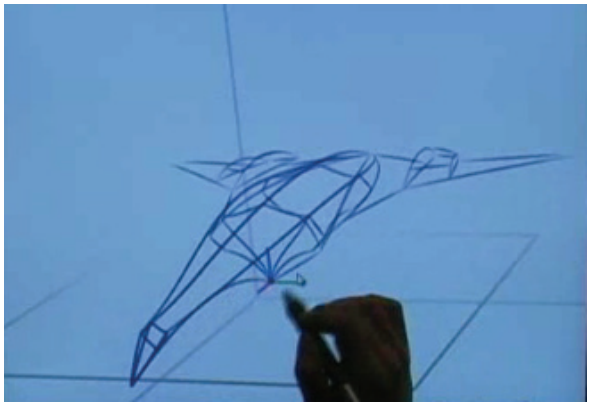


Figura 3.6. Ejemplo con "ILoveSketch"

A pesar de que ambas aplicaciones ofrecen un interfaz intuitivo que ayuda al usuario en la fase conceptual del diseño, su campo de aplicación se encuentra bastante delimitado. El trabajo, por ejemplo, realizado por el MIT permite dibujar entidades variadas que componen un sistema físico y al cual se le aplica una cierta gravedad a los elementos constituyentes en función del volumen encerrado por cada forma, ofreciendo una simulación del proceso. Sin embargo no se realiza en ningún momento el diseño de una pieza o elemento. En el caso de la aplicación ILoveSketch sí se lleva a cabo un diseño, empleando para ello una serie de líneas curvas que posteriormente podrán ser exportadas a otra aplicación para su tratado. Sin embargo no se lleva a cabo ningún reconocimiento, limitándose únicamente a generar las curvas en función de los puntos trazados.

Es precisamente esta la razón por la cual se ha optado por desarrollar una arquitectura multi-agente que permita el reconocimiento de bocetos dibujados a mano, como parte de un software de más alto nivel, detectando tanto entidades (líneas, arcos, círculos, curvas, elipses, etc.) como comandos gestuales para su tratamiento (extrusión, revolución, simetría y borrado) y restricciones geométricas/dimensionales (acotación, concentricidad, paralelismo, perpendicularidad, etc.). El empleo de agentes permite a su vez ampliar el campo de aplicación de forma sencilla mediante la creación de nuevos agentes, que se pueden incorporar siguiendo una estructura modular y dotando así al sistema de cierta flexibilidad.

La siguiente tabla muestra a modo de resumen las aplicaciones comerciales y gratuitas mencionadas.

Tabla 3.8. Tabla resumen de las aplicaciones mencionadas

Referencia	*Aplicación*	*Descripción*
[Catia77]	Catia	Herramienta CAD/CAM/CAE muy potente que dispone de varios módulos específicos para la industria aeronáutica, del automóvil, etc....
[AutCac82]	Autocad	Software de diseño asistido por ordenador para dibujo en 2D y 3D, actualmente desarrollado por la empresa Autodesk
[ProEng88]	Pro/Engineer	Software de diseño paramético para CAD/CAM/CAE, muy popular entre diseñadores mecánicos y más económico que CATIA
[SldWks95]	Solidworks	Programa de diseño asistido por ordenador para modelado mecánico que permite modelar piezas y conjuntos y extraer de ellos tanto planos como otro tipo de información necesaria para la producción
[JavSke97] [ZHH96]	Java Sketch	Aplicación web gratuita que permite la creación de modelos en tres dimensiones mediante el dibujo de cubos y líneas
[SketUp00]	Sketch Up!	Software de diseño y modelado en 3D para entornos arquitectónicos e ingeniería civil que permite conceptualizar rápidamente volúmenes y formas arquitectónicas. Desarrollado y publicado por Google.
[TablPc01]	Tablet PC	Interfaz caligráfico consistente en un ordenador portátil provisto de pantalla tactil y puntero. Existen algunos modelos que posibilitan rotar la pantalla y cerrarlo, convirtiéndose en un portafolios digital.
[JS02]	Microsoft.ink	Librería software para implementar en varios lenguajes de programación aplicaciones diversas que aprovechen las características de un Tablet PC
[SkeMast05]	Sketch Master	Aplicación que transforma las fotografías introducidas en bocetos dibujados a mano de una forma muy realista.
[Iphon07]	Iphone	Teléfono móvil multimedia de última generación provisto de pantalla táctil (con tecnología multitactil) y una interfaz de hardware minimalista.
[SkeBook07]	SketchBook Pro	Aplicación desarrollada por Autodesk que permite dibujar de forma artística en un iPhone consiguiendo resultados bastantes espectaculares
[SusApp07]	Simple Sketch App	Web que posee un interfaz para introducir un boceto que posteriormente queda almacenado en una base de datos, posibilitando que cualquier visitante de la página pueda consultarlo posteriormente
[MITSke07]	Magic Paper	Aplicación de bocetado sobre una pizarra digital donde los distintos elementos son reconocidos y asignados con unas propiedades físicas determinadas
[Sket2iP08]	Sketches	Herramienta comercial diseñada y programada para el iPhone y el iPod-Touch que permite dibujar y decorar imágenes con figuras de diversos tipos, así como escribir notas y diagramas de forma táctil
[ILovSke08] [BBS08]	ILoveSketch	Software de bocetado de formas curvas que recurre para el trazado a la técnica del "overtracing", siendo necesario dibujar varias veces la misma línea curva que se desea introducir
[FBSkeMe09]	FaceBook Sketch Me	Aplicación que permite transformar una fotografía digital en un boceto para insertarlo posteriormente en el perfil de FaceBook

Capítulo 4:
MATERIALES Y MÉTODOS

4. MATERIALES Y MÉTODOS

Esta tesis se enmarca dentro de un proyecto del ministerio (Ref.: DPI2007-66755-C02-01) que tiene como uno de sus objetivos principales el implementar un interfaz robusto e intuitivo que permita la creación de modelos a través de bocetos. La idea principal es que se reconozca la geometría de los bocetos (fundamentalmente primitivas como rectas, arcos, círculos), así como los símbolos que lo acompañan (para parametrizarlo o porque pertenezcan a él) y gestos que lo controlan (de visualización, edición, borrado u otros comandos para la generación de geometría tridimensional).

Previamente a la ejecución de esta tesis, se ha participado en el desarrollo de dos reconocedores basados en técnicas tradicionales de clasificación utilizando características invariantes extraídas a los bocetos introducidos mediante el análisis de imagen. Estos reconocedores tratan de ajustar el boceto introducido con una geometría determinada e interpretan los símbolos y comandos gestuales establecidos, permitiendo parametrizar la geometría y editar el modelo. Ambos reconocedores se describen con más detalle en el Anexo II.

Sin embargo, estos dos trabajos previos se basan en técnicas tradicionales poco flexibles respecto a la ampliación del alfabeto de símbolos/gestos, y muy rígidas en cuanto a la clasificación de los mismos, sin posibilidad de no clasificación de símbolo/gesto inexistente en el alfabeto, esto es, siempre clasifica el esbozo introducido, aún cuando éste no pertenezca a la colección recogida en el alfabeto definido, lo que plantea un serio problema. Otro problema que presentan es que en cuanto aumenta el número de símbolos del alfabeto, el porcentaje de éxito disminuye considerablemente aunque se entrene el sistema para aceptar los nuevos símbolos. Estos inconvenientes, añadidos a la falta de información del contexto a la hora de decidir la clasificación y la estructura secuencial de su organización hacen que sea poco efectivo y no muy eficiente, lo que nos conduce plantearnos profundizar en otros posibles campos más adecuados para este tipo de aplicaciones.

Por estos motivos, y atendiendo a los objetivos propuestos, se ha optado por elegir una plataforma multi-agente que soporte la arquitectura que se pretende diseñar y en la que se implementen los métodos y algoritmos necesarios vía agentes para la implementación del reconocedor. Dicha plataforma debe soportar la comunicación entre agentes, las estructuras de datos necesarias y la ejecución de librerías en lenguaje C. Otra cuestión que se comenta en los objetivos es la del establecimiento de las diferentes *cues* del trazo y otra información extraída del contexto que determinarán el reconocimiento, cuya extracción se deberá implementar a través de los agentes de la plataforma.

En este capítulo se describe inicialmente el concepto de agente y su funcionalidad dentro de una arquitectura multi-agente. Posteriormente se plantean varias plataformas que posibilitan la implementación de la arquitectura multi-agente, haciendo una breve comparación. A continuación se detalla la plataforma seleccionada. Un dato importante a tener en cuenta es que las plataformas que soportan programación con agentes no utilizan el lenguaje C de programación, sin embargo, muchas de las librerías estándar de visión están desarrolladas en

C o C++, con lo que se hace necesario la creación de un *Shell* que permita la ejecución de ciertas librerías externas a la plataforma. Atendiendo a esto último, también se describe la metodología empleada para ejecutar código nativo de C++ en el entorno de programación de la plataforma seleccionada. A continuación se detalla un estudio de viabilidad de este tipo de sistemas basado en agentes sobre uno de los trabajos previos a esta tesis expuesto en el Anexo II. Luego se presenta el paradigma de reconocimiento propuesto y el sistema diseñado con la arquitectura multi-agente implementada y, por último, se establecen las *cues* básicas que posteriormente se usarán para la correcta clasificación de los bocetos, así como el método de segmentación desarrollado para la detección de vértices de esquina y tangentes.

4.1 La plataforma multi-agente

4.1.1 Los agentes

Previamente a la elección de la plataforma donde implementar la arquitectura multi-agente, cabe definir qué se entiende por agente y cuáles son sus características principales.

Existen muchas definiciones de qué es un agente inteligente [Woo02], siendo la que mejor se adapta la dada por Pattie Maes, investigadora del MIT: "Un agente es un sistema computacional que vive en un entorno complejo y dinámico. El agente puede sentir ese entorno y actuar en consecuencia, teniendo un conjunto de objetivos o motivaciones que intenta conseguir a través de dichas acciones".

Algunas de las características principales que debe poseer un agente son: autonomía, cooperación, movilidad, adaptabilidad, veracidad, reactividad y pro-actividad.

Las tareas o servicios que un agente lleva a cabo se especifican mediante comportamientos (también denominados "*behaviours"*), que hacen referencia a las funcionalidades que incorpora el agente.

En el caso, por ejemplo, de la plataforma JADE, estos comportamientos se agrupan en varios grupos tal y como se describe en [BCG07 y BPR00], siendo los tres principales:

- **Comportamientos one-shot**: tipos de comportamiento los cuales se ejecutan de manera casi instantánea, y solamente una vez.
- **Comportamientos cíclicos**: son aquellos que nunca son sacados del conjunto de comportamientos del agente y cuyo método nunca finaliza.
- **Comportamientos genéricos**: algo más complejos, el código que se ejecuta en ellos depende del estatus del agente, y eventualmente finalizan su ejecución basándose en condiciones definidas.

En la Figura 4.1 se muestra la organización jerárquica de la clase Comportamiento de los agentes.

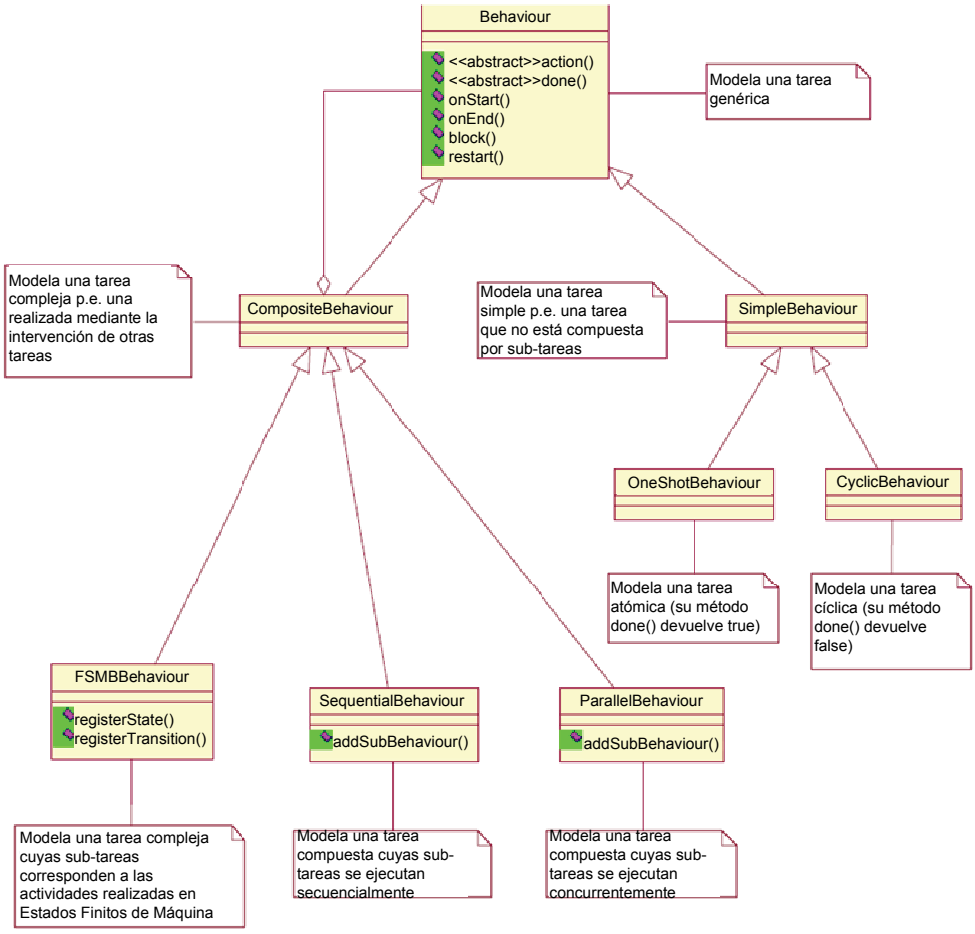


Figura 4.1. Diagrama de bloques de la jerarquía de la clase Comportamiento

4.1.2 El estándar

El creciente desarrollo de grupos de investigación en torno a los sistemas multi-agente fomentó la aparición de nuevas formas para el desarrollo de esta tecnología, lo cual conllevó a que cada uno de estos grupos presentara soluciones diferentes e independientes.

Algunos problemas que se presentaron fue la carencia de una definición estándar de sistema multi-agente e incompatibilidad, lo que conllevó a una incapacidad para satisfacer los fuertes requisitos de las empresas e industrias actuales. Esto promovió la creación de diversos organismos, los cuales se preocuparían de desarrollar una definición estándar para la construcción de sistemas multi-agente. Algunos de estos organismos son OMG (Object Management Group), KSE (Knowledge Sharing Effort) y FIPA (Foundation for Intelligent Physical Agents). La siguiente tabla muestra un resumen de sus objetivos y aportaciones:

Tabla 4.1. Tabla resumen de las principales organizaciones y sus aportaciones

Estándar	*Objetivos*	*Aportaciones*
OMG	Reutilización, portabilidad e interoperabilidad de sistemas distribuidos de componentes software orientados a objetos	- UML (Unified Modeling Language) - CORBA (Common Object Request Broker Architecture) - MASIF (Mobile Agent System Interoperability Facility)
KSE	Facilitar la compartición y reutilización de bases y sistemas basados en conocimiento	- Sintaxis: KIF (Knowledge Interchange Format) --> Lenguaje para el intercambio de conocimiento - Semántica: Ontolingua --> Lenguaje para la definición de Ontologías - Pragmática: KQML (Knowledge Query and Manipulation Language) --> Lenguaje para la comunicación entre agentes --> Interoperabilidad entre agentes en un entorno distribuido
FIPA	Promoción de la industria de los agentes inteligentes mediante el desarrollo de especificaciones que soporten la interoperabilidad entre agentes y aplicaciones basadas en agentes. Cubre todos los aspectos de un entorno de agentes - Aplicaciones - Arquitectura - Comunicación - Protocolos de Interacción, Actos comunicativos y Lenguajes de Contenidos - Gestión de Agentes - Transporte de mensajes	- FIPA-ACL --> Basado en la Teoría de los Actos del Habla --> Un mensaje en FIPA ACL representa la intención de realizar alguna acción (acto comunicativo) --> Un mensaje en FIPA ACL tiene una sintaxis similar a un mensaje en KQML --> Modelo de Comunicación basado en la asunción de que 2 agentes que quieren conversar han de compartir una ontología común que describa el universo de discurso

De todos estos estándares, FIPA es el que más relevancia ha tomado en estos últimos años, pues cubre todos los aspectos que se requieren en un entorno basado en agentes:

- Mecanismos de comunicación
- Manejo de directorio de agentes
- Manejo de directorio de servicios
- Lenguaje de comunicación entre agentes

Todas estas características son tratadas con mayor detalle en el Anexo I, donde se define dicho estándar FIPA de forma más amplia.

4.1.3 Las plataformas

Son varias las plataformas que actualmente posibilitan la implementación de una arquitectura multi-agente, destacando los siguientes entornos de programación:

- ABLE (Agent Building and Learning Environment).

  Está desarrollado por IBM, y no está orientado inicialmente a crear sistemas multi-agente. Permite crear agentes en forma de AbleBeans o AbleAgents, que son clases en JAVA desde las que, utilizando herencia, se desarrollan los agentes necesarios. Haciendo uso de un editor de agentes se obtiene una visión gráfica del conjunto completo de la aplicación. Otra de las ventajas de este entorno es la gran facilidad para incorporar diversos mecanismos de inteligencia artificial a los propios agentes, tales como redes neuronales, árboles de decisión, teorema de Bayes, etc.

- JADE (Java Agent Development).

  Su desarrollo actual se lleva a cabo en el Telecom Italia Lab., aunque gracias a su licencia libre (Lesser General Public License) permite que realmente sea desarrollado por una comunidad global muy activa. Esto le da otra de sus grandes ventajas, como es la documentación generada por toda esa comunidad y una realimentación continua. Es un entorno probado y con múltiples proyectos en producción. Está orientado a la creación de sistemas multi-agente desde una perspectiva interna al agente. Se distingue entre el agente en sí y sus comportamientos, que son intercambiables y, lógicamente, es posible basarse en cualquiera de ellos para desarrollar un comportamiento nuevo. JADE cumple las especificaciones de la FIPA (Foundation for Intelligent Physical Agents), encargada de desarrollar estándares para la interoperabilidad entre plataformas de agentes software.

- JADEX (Java Agent Development Extended).

  Se trata de una plataforma en modo gráfico basada en JADE para crear una nueva forma de desarrollar agentes. JADEX sigue el modelo BDI (Beliefs Desires Intentions), es decir, cada agente tiene sus creencias, deseos e intenciones. Resumiendo: cada agente tiene un fichero XML llamado ADF (Agent Description File) donde entre otras cosas se definen los datos que conocemos, los objetivos que queremos alcanzar y los planes para alcanzar los objetivos. Por otra parte, los planes que definimos, así como los datos que conocemos del entorno, son métodos/objetos Java.

- JASON (Java-based AgentSpeak interpreter used with Saci for multi-agent distribution Over the Net).

  Jason es un intérprete para una versión extendida de AgentSpeak, basándose en la arquitectura BDI (Beliefs, Desires, Intentions). Proporciona la plataforma para el desarrollo de sistemas multi-agente con muchas características ajustables. JASON está disponible como Open Source y se distribuye bajo licencia GNU-LGPL.

- JADE-LEAP (Java Agent Development – Lightweight Extensible Agent Platform).

  LEAP es una plataforma reducida de JADE, diseñada para ser usada en dispositivos móviles tales como PDA's o móviles (LEAP es a JADE como J2ME es a J2SE). Para ejecutarla, se necesita la versión micro de Java en vez de JDK y debería usarse con JWT (Java Wireless Toolkit). Hay que tener en cuenta que muchas funciones están ausentes, dado que estamos usando dos versiones limitadas (LEAP y J2ME). Por ejemplo, no podremos usar Awt o el Sniffer típico de JADE, usado para monitorizar a los agentes cuando estos se están ejecutando.

La siguiente tabla comparativa muestra las ventajas y desventajas que posee cada una de las plataformas, aplicadas al caso concreto de la implementación de una arquitectura multi-agente para el reconocimiento de bocetos.

Tabla 4.2. Tabla comparativa de las principales plataformas multi-agente

Plataforma	*Ventajas*	*Desventajas*
ABLE	1 - Haciendo uso de un editor de agentes se obtiene una visión gráfica del conjunto completo de la aplicación 2 - Facilidad para incorporar diversos mecanismos de inteligencia artificial a los propios agentes, tales como redes neuronales, árboles de decisión, teorema de Bayes, etc	1 - No está orientado inicialmente a crear sistemas multi-agente
JADE	1 - Licencia gratuita LGPL (Lesser General Public License) 2 - Desarrollado por una comunidad global muy activa 3 - Amplia documentación generada por dicha comunidad, contando con una realimentación continua 4 - Entorno probado y con múltiples proyectos en producción 5 - Orientado a la creación de sistemas multi-agente desde una perspectiva interna al agente 6 - Se distingue entre el agente en sí y sus comportamientos 7 - Cumple las especificaciones de la FIPA, encargada de desarrollar estándares para la interoperabilidad entre plataformas de agentes software	
JADEX	1 - Plataforma en modo gráfico basada en JADE para crear una nueva forma de desarrollar agentes	1 - Sigue el modelo BDI (Beliefs Desires Intentions), es decir, cada agente tiene sus creencias, deseos e intenciones. Sin embargo este tipo de modelos son usados normalmente para la implementaciones de redes sociales, sin tener mucha utilidad en el ámbito de la presente tesis
JASON	1 - Disponible como Open Source y distribuido bajo licencia GNU-LGPL	1 - intérprete para una versión extendida de AgentSpeak, basándose en la arquitectura BDI (Beliefs Desires Intentions), por lo que tampoco tiene mucha utilidad en el ámbito de la presente tesis
JADE-LEAP	1 - Plataforma diseñada para ser usada en dispositivos móviles tales como PDA's o móviles	1 - Para ejecutarla se necesita la versión micro de Java en vez de JDK y debería usarse con JWT (Java Wireless Toolkit) 2 - Muchas funciones están ausentes, dado que se está usando dos versiones limitadas (LEAP y J2ME) 3 - No se puede usar el Sniffer típico de JADE, usado para monitorizar a los agentes cuando estos se están ejecutando

Así pues, y atendiendo a la tabla comparativa expuesta, de entre todas las posibles plataformas se ha elegido JADE para implementar la arquitectura multi-agente para reconocimiento de bocetos, proporcionando facilidades como el desarrollo extremadamente rápido de agentes dentro de un sistema multi-agente, desde una perspectiva de la interoperabilidad total tanto entre distintas plataformas que cumplan con las especificaciones FIPA, como entre los propios agentes del sistema independientemente de la ubicación, dispositivo o distancia a la que se encuentren. Adicionalmente a esta justificación, cabe mencionar el empleo de esta plataforma por parte de los grupos más fuertes en el uso de Agentes (como por ejemplo el grupo de Vicente Botti de la U.P.V., o el grupo de la Universitat Jaume I de Castellón), así como de otros grupos cercanos que comparten conocimientos y avances.

4.1.4 El entorno de programación

Son varios los entornos de programación que posibilitan la programación en JAVA, en el cual se desarrolla JADE. Sin embargo, dentro de este amplio abanico, tan sólo existen dos que posibilitan la implementación de la plataforma multi-agente JADE: Eclipse y Netbeans.

A continuación se muestra una tabla comparativa de ambos entornos de programación.

Tabla 4.3. Tabla comparativa entre Eclipse y Netbeans

Eclipse	*Netbeans*
Interfaz más amigable y rápido que Netbeans	Requiere gran cantidad de memoria para funcionar correctamente
En poco minutos se puede realizar un nuevo proyecto	La curva de aprendizaje es elevada
Ofrece un muy buen soporte de refactorización (proceso de llevar a cabo cambios en el código sin afectar el comportamiento de la aplicación)	Netbeans carece de soporte de refactorización (solo hay un pequeño módulo gratuito para Netbeans)
Posee una excelente ayuda a la codificación (ofreciendo sugerencias a medida que se va escribiendo el código)	Los errores únicamente aparecen al compilar la aplicación
Ofrece también el IDE Eclipse C++, permitiendo así programar tanto en C++ como en JAVA bajo el mismo entorno de programación	No ofrece soporte para C++
No dispone de herramientas para crear elementos visuales, teniendo que recurrir a su programación directamente	Ofrece herramientas para crear elementos visuales (botones, ventanas, etc…) de forma rápida y sencilla

Eclipse fue creado inicialmente por IBM (y administrado actualmente por la Fundación Eclipse) mientras que NetBeans fue patrocinado por SUN MicroSystems. En cuanto a

actualizaciones, ambos tienen soporte regular por parte de sus patrocinadores. Es precisamente en el tema de los plugins donde *Eclipse* cuenta con una mayor comunidad de usuarios y crear plugins para Eclipse es más sencillo, lo cual lo convierte en un IDE extremadamente potente [Dau04]. Más concretamente las ventajas de Eclipse pueden detallarse en los siguientes aspectos:

- Ofrece un interfaz más amigable y rápido que Netbeans (éste requiere gran cantidad de memoria para funcionar correctamente).
- La curva de aprendizaje es menor (en poco minutos se puede realizar un nuevo proyecto).
- Ofrece un muy buen soporte de refactorización (proceso de llevar a cabo cambios en el código sin afectar el comportamiento de la aplicación), del cual Netbeans carece (solo hay un pequeño módulo gratuito para Netbeans).
- Posee una excelente ayuda a la codificación (ofreciendo sugerencias a medida que se va escribiendo el código).
- Ofrece también el IDE Eclipse C++, permitiendo así programar tanto en C++ como en JAVA bajo el mismo entorno de programación.

Como desventajas de Eclipse frente a Netbeans tenemos fundamentalmente que:

- No tiene buen soporte de webapps (.war, jsp y servlets). Los plugins existentes no son ni tan potentes ni tan sencillos como el módulo que en Netbeans viene preinstalado.

Todo esto, unido a la facilidad de uso del interfaz y el menor tiempo de aprendizaje necesario, han motivado la elección del IDE Eclipse como entorno de programación para implementar la arquitectura multi-agente de la tesis que se presenta, entorno cuyas características generales se muestran en el Anexo I

4.2 Ejecución de código nativo de C++ en Java

En el apartado anterior se ha justificado la necesidad de usar el lenguaje de programación Java, junto con el IDE Eclipse y la plataforma multi-agente JADE. Sin embargo, existe una gran cantidad de librerías de visión desarrolladas únicamente en C++ (las versiones de Java no ofrecen la misma funcionalidad) que contienen funciones necesarias para la correcta implementación de los reconocedores. Por esta razón se propone el empleo de una herramienta de desarrollo que permita ejecutar funciones pertenecientes a librerías desarrolladas en código nativo C++ desde el entorno Eclipse bajo el lenguaje Java.

En la actualidad existen gran variedad de herramientas de desarrollo software que permiten ejecutar código nativo desarrollado en un lenguaje de programación determinado (en nuestro caso funciones compiladas en una DLL de Visual C++) en otro entorno de programación distinto (Java, usando el entorno eclipse). A este tipo de herramientas se les conoce comúnmente como "w*rappers*". Es tal la variedad existente que resulta complicado elegir cuál se adapta mejor a cada situación, siendo muy diversas las opciones que ofrece

cada una en cuanto a simplicidad y posibilidades. De forma general podrían enumerarse las siguientes herramientas de desarrollo:

- Java Native Interface (JNI): Gratuito. Interfaz que permite integrar una aplicación Java con código nativo escrito en lenguajes como C o C++. Es el interfaz más a bajo nivel que se puede usar, lo cual permite implementar casi cualquier interfaz. Tiene como inconveniente el tiempo que se debe dedicar a su estudio y programación, pudiendo llevar meses el programar un interfaz sencillo. Es precisamente esta complejidad en su programación lo que ha llevado a desarrollar otras herramientas que simplifican el proceso y que generan el código JNI correspondiente sin necesidad de conocer complejas instrucciones. Así pues, y en base a la experiencia obtenida, es la última opción que se debe escoger, y sólo se debe recurrir a ella si ninguna otra herramienta nos ofrece lo que buscamos.
- JNIWrapper (TeamDev Software): De pago. Permite acceder a librerías nativas y componentes desde una aplicación en Java sin necesidad de usar JNI. Es una opción sencilla y rápida para implementar un interfaz en nuestra aplicación Java y posibilitarnos ejecutar una DLL compilada en C++. Presenta como limitación frente a otras opciones la no implementación de librerías STL (Standard Template Library), siendo un problema en el caso de que la librería en C++ use este tipo de datos. A esto se le añade la necesidad de pagar una licencia para su uso. Tan sólo se recomienda su uso en el caso de que necesitemos un interfaz muy básico, sin vectores ni estructuras propias de datos.
- NoodleGlue: Gratuito. Proyecto sin ánimo de lucro en fase beta (actualmente ha cerrado) que permitía disponer de un interfaz también muy básico, pero algo más complejo de desarrollar. Se diseñó como herramienta para ejecutar el código nativo usado en la plataforma interactiva de música NoodleHeaven.
- SWIG: Gratuito. Herramienta de desarrollo software que conecta programas escritos en C y C++ con una gran variedad de lenguajes de programación de alto nivel, como Perl, PHP, Python, Tcl, Guile, Ruby, Common Lisp, C#, Lua, Modula-3, Octave y JAVA. Está en constante evolución y presenta versiones estables completamente funcionales. Su punto fuerte está precisamente en la gran variedad de posibilidades que ofrece, permitiendo usar *arrays*, vectores, librerías STL, estructuras, *strings*, etc. Incorpora sus propias librerías para facilitar el trabajo, siendo únicamente necesario incluirlas en nuestro proyecto. Dado el gran número de ventajas que ofrece, se recomienda encarecidamente su empleo como *wrapper* de C++ a Java.

La siguiente tabla comparativa muestra a modo de resumen las ventajas y desventajas que posee cada una de las herramientas descritas.

Tabla 4.4. Tabla comparativa entre '*wrappers*'

Wrapper	*Ventajas*	*Desventajas*
Java Native Interface (JNI)	1 - Licencia gratuita 2 - Al ser un interfaz de muy bajo nivel permite implementar casi cualquier interfaz	1 - Se debe dedicar mucho tiempo a su estudio y programación, pudiendo llevar meses el programar un interfaz sencillo 2 - Debido a su complejidad en la programación se hace necesaria la existencia de herramientas que simplifiquen el proceso y generen el código JNI correspondiente
JNIWrapper	1 - Permite acceder a librerías nativas y componentes desde una aplicación Java sin necesidad de usar JNI	1 - Software comercial de pago 2 - Tan sólo implementa una serie de tipo de datos muy básicos, imposibilitando el uso de estructuras y vectores STL 3 - Solución para interfaces muy básicos
NoodleGlue	1 - Gratuito	1 - Proyecto en fase beta (actualmente ha sido abandonado) 2 - Propuesta para interfaces muy básicos
SWIG	1 - Gratuito 2 - Conecta programas escritos en C y C++ con una gran variedad de lenguajes de programación de alto nivel, como Perl, PHP, Python, Tcl, Guile, Ruby, Common Lisp, C#, Lua, Modula-3, Octave y JAVA 3 - Está en constante evolución y presenta versiones estables completamente funcionales 4 - Permite implementar cualquier tipo de dato, pudiendo definir estructuras propias de datos, vectores, librerías STL 5 - Incorpora sus propias librerías para facilitar el trabajo, siendo únicamente necesario incluirlas en el proyecto	

Así pues, de todas las posibilidades estudiadas se ha escogido la herramienta SWIG, dada su fácil implementación a la hora de reutilizar código existente en otro lenguaje de programación, además de permitir manejar estructuras de datos personalizadas sin mucha complejidad (vectores STL, por ejemplo). En el caso de las otras opciones resulta demasiado tedioso y el tipo de estructuras disponibles se encuentra muy limitado.

4.3 Estudio de la viabilidad del uso de agentes para el reconocimiento

Para determinar la viabilidad del uso de agentes aplicados al campo del reconocimiento, se realizó inicialmente una implementación del reconocedor de símbolos RecoGes presentado en el Anexo II de esta tesis, en un sistema multi-agente. El propósito principal, además de estudiar la viabilidad de estos sistemas aplicados al reconocimiento, fue la introducción al manejo de los agentes, esto es, la comunicación entre agentes, el paso de

estructuras de datos, y la organización y gestión de un sistema de múltiples agentes en diferentes niveles.

Básicamente, se realizó la distribución de las tareas del reconocedor RecoGes para paralelizar el algoritmo original, de forma que cada tarea fuera realizada por un agente, creando una arquitectura basada en agentes e implementando un sistema de comunicaciones y transferencia de datos entre los mismos, lo que ha servido para posteriores desarrollos en esta tesis.

La tendencia en la programación de muchos algoritmos de reconocimiento suele ser recurrir a una estructura secuencial, existiendo una serie de procesos que se ejecutan uno detrás de otro, tal y como muestra la Figura 4.2.

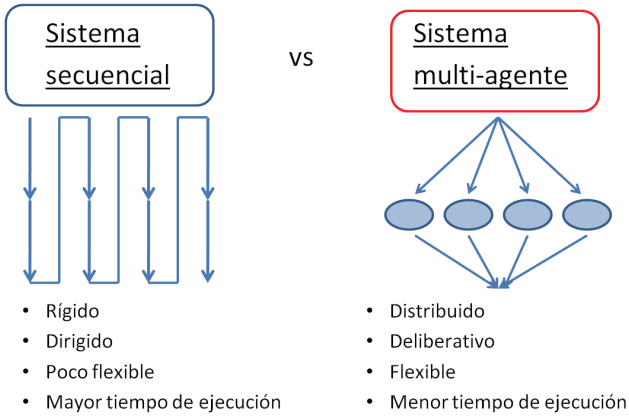


Figura 4.2. Diferencias entre un sistema secuencial y uno multi-agente

En este nuevo RecoGes, al que nos referiremos a partir de ahora como Recoges-A, se pretendía disponer de varios agentes autónomos que permitieran ejecutar los procesos de forma distribuida. Aunque comúnmente los agentes son utilizados en entornos que necesitan de negociaciones o debates para la toma de decisiones en función de lo que vaya aconteciendo [Woo02, Dam05 y PW04], como ya se ha comentado, en este trabajo previo sólo se abordó la creación de una arquitectura basada en agentes con el fin de paralelizar en todo lo posible el reconocedor, dejando como trabajo para la tesis la implementación de agentes en su estado más puro.

En este entorno multi-agente, se determinaron las tareas específicas del proceso de reconocimiento y se asignaron a agentes diferentes que pudieran actuar de manera independiente, evitando el costoso proceso secuencial y reorganizando el proceso de ejecución de una forma más lógica.

Con respecto a la fase de pre-procesado, existen una serie de tareas que deben seguir siendo llevadas a cabo en un orden secuencial, ya que una no empieza hasta que la anterior haya terminado. A tal efecto se asignó un único agente llamado PA (Pre-processing Agent), encargado de las tareas de pre-procesado. Una vez el pre-procesado había terminado,

comenzaba la extracción de características. Éstas eran extraídas en paralelo ya que eran independientes unas de otras. Así pues, se asignaron diferentes agentes para la extracción de diferentes características, de modo que existieran tantos agentes como características deseáramos extraer del gesto. A estos agentes se les llamó FA (Feature Agents), y comenzaban su ejecución autónoma cuando el agente encargado del pre-procesado PA se lo notificaba.

Cuando un FA terminaba, se lo notificaba al agente de nivel superior responsable de la clasificación del gesto llamado SICA (Sketch Interpretation/Classification Agent). Este agente esperaba a que todos los FA terminaran, y con la información generada por éstos clasificaba el gesto del mismo modo que lo hacía el RecoGes secuencial, utilizando el teorema de Bayes mediante un análisis discriminante no lineal. La Figura 4.3 muestra la arquitectura del sistema implementado basado en agentes y en la Figura 4.4 se puede ver el comportamiento del agente SICA. Para la implementación de la arquitectura se usó la plataforma JADE [BCG07, BPR00], ya detallada con anterioridad.

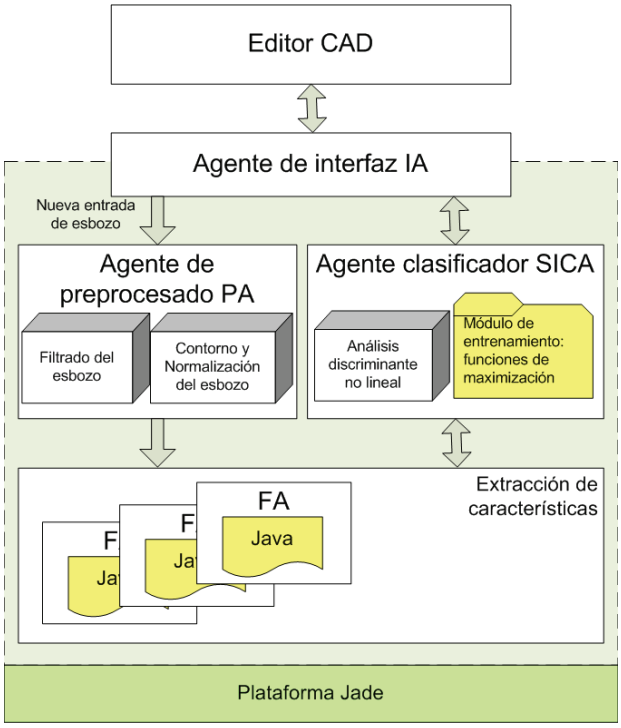


Figura 4.3. Arquitectura multi-agente inicial del reconocedor RecoGes-A

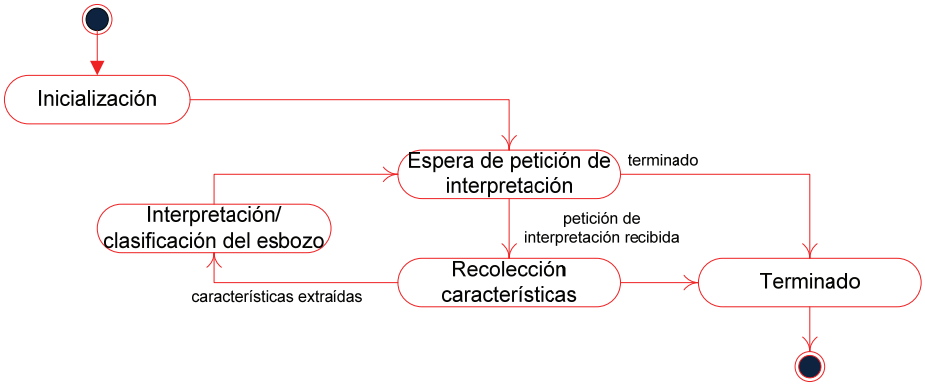


Figura 4.4. Diagrama de comportamiento del SICA

Este trabajo, el cual queda recogido en [FAC09], evolucionó posteriormente profundizándose en la creación y gestión de agentes más autónomos e inteligentes organizados en diferentes niveles jerárquicos, de modo que se han destinado agentes más simples para las tareas de extracción de *cues* básicas de los trazos y agentes más complejos para la toma de decisiones que afecta al reconocimiento final. En esta arquitectura implementada las comunicaciones son más complejas, así como el código y los comportamientos de los agentes diseñados. La estructura diseñada en este trabajo se aprecia en la Figura 4.5 [FCA09].

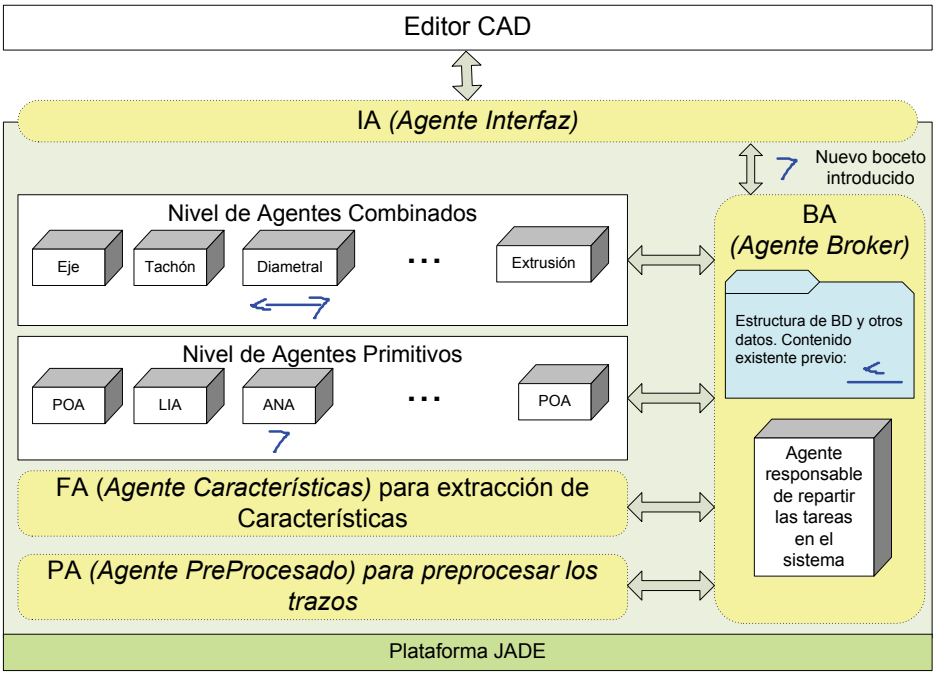


Figura 4.5. Arquitectura posterior del reconocedor RecoGes-A

Así pues, estos trabajos demuestran la viabilidad de aplicar este tipo de sistemas al campo del reconocimiento frente a los tradicionales, debido fundamentalmente a las ventajas que aportan los sistemas paralelos frente a los secuenciales y a la flexibilidad de añadir nuevos gestos al reconocedor.

4.4 Arquitectura del sistema multi-agente

El primer paso antes de implementar una arquitectura multi-agente consiste en definir el paradigma del proceso que se pretende soportar. En el caso de la presente tesis dicho paradigma está orientado a la interpretación de bocetos dibujados a mano por los diseñadores de productos durante las etapas creativas del diseño, buscando a su vez una solución escalable que permita cierta flexibilidad. Por lo tanto, el objetivo consiste en definir una aproximación de una estructura basada en agentes, aprovechando la flexibilidad de estos sistemas y la autonomía que un agente por sí solo puede ofrecer. La estructura propuesta trabaja en tres niveles:

a) Los agentes básicos, residentes en el nivel inferior, actúan para uniformar y extraer características de los trazos introducidos por el usuario (agentes pre-procesado y características).

b) Los agentes primitivos, que permanecen en un nivel intermedio, usan estos resultados para encontrar los contenidos sintácticos de los trazos.

c) Y por último, en el nivel superior, los agentes combinados obtienen los contenidos semánticos de los grupos de trazos.

La organización y distribución de estos agentes agrupados en distintos niveles se muestra en las secciones siguientes.

4.4.1 Paradigma propuesto del sistema de reconocimiento

Una estructura multi-agente ofrece, por su propia naturaleza, la posibilidad de centrar las tareas de reconocimiento de un gesto en función del conjunto de símbolos que interesa detectar. De esta manera, es posible agrupar los gestos en áreas temáticas definidas en ingeniería. A modo de ejemplo podemos bocetar gestos empleados en el dibujo artístico (línea y arco artísticos), símbolos que permitan ejecutar operaciones de modelado (extrusión, revolución, sección o eje de simetría/revolución), símbolos relacionados con el análisis estructural (uniones fijas, uniones articuladas o uniones deslizantes), símbolos relacionados con el diseño industrial (roscas, rodamientos o ruedas dentadas), símbolos asociados a restricciones del diseño (horizontalidad, verticalidad, paralelismo, perpendicularidad, igualdad, concentricidad, tangencia), o símbolos de acotación (cota radial, cota diametral, cota lineal).

La plataforma que se propone presenta una estructura integrada, por un lado, de un núcleo central que actúa de interfaz con el usuario y en el que se definen los agentes encargados de realizar las tareas básicas de reconocimiento, coordinados por un agente broker (BACC); y por otro lado, de una serie de módulos gestionados cada uno de ellos por otro agente broker (BAM), que reparte las tareas a los agentes encargados del reconocimiento de cada uno de los símbolos definidos en el módulo y que informa a un núcleo central del

resultado de sus agentes para que tome la decisión final (Figura 4.6). Cabe mencionar que en esta figura (y en sucesivas) cada icono representa a un agente del sistema.

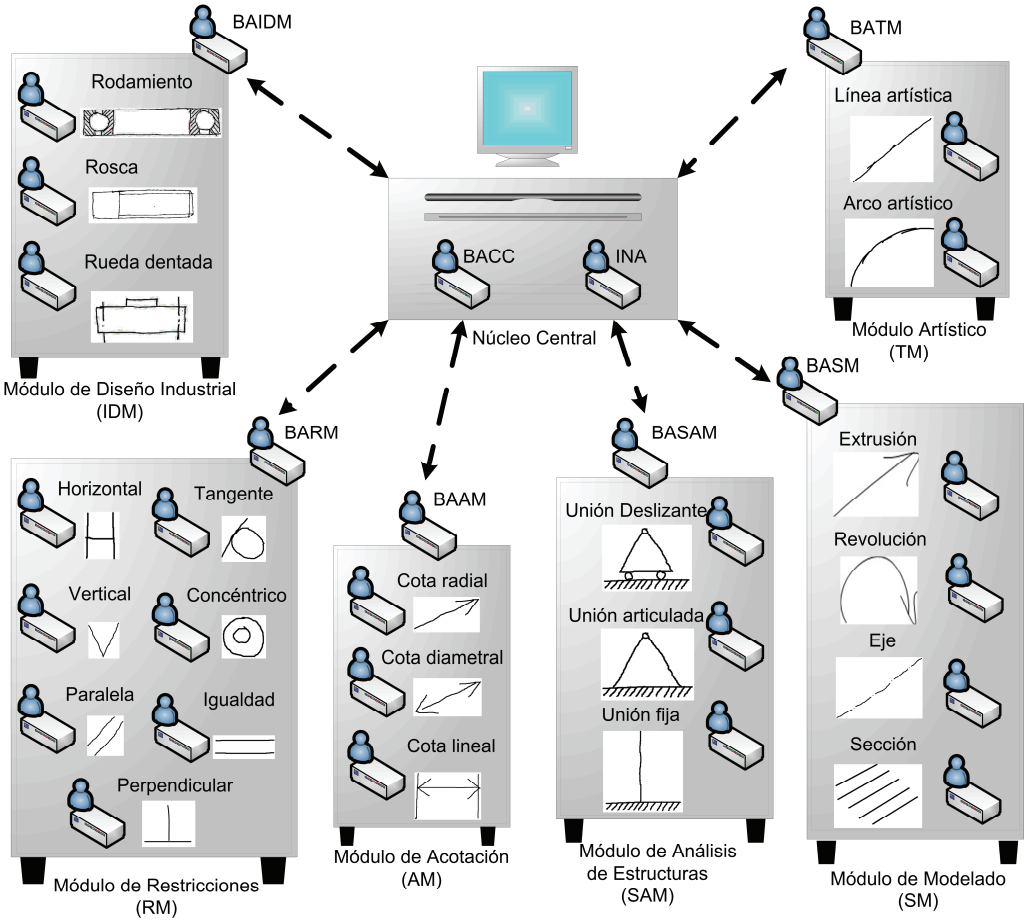


Figura 4.6. Esquema general de la plataforma de bocetado basado en agentes

4.4.2 El núcleo central

Para poder aprovecharse de la flexibilidad y autonomía de una arquitectura jerárquica multi-agente para el proceso de reconocimiento, se debe definir previamente una descomposición jerárquica de los símbolos. Así pues, los símbolos están formados por líneas, donde dichas líneas son como “fonemas” de un alfabeto. Uno o varios fonemas forman a su vez un símbolo (como si de una “palabra” se tratara), el cual pertenece a un conjunto de símbolos (equivalente a un “diccionario”).

El reconocimiento de los distintos símbolos definidos en cada módulo de la plataforma, basan su funcionamiento en el reconocimiento previo de entidades más simples.

En muchas ocasiones un trazo equivaldrá a una línea, pero son varios los casos en los que un único trazo puede contener un "símbolo primitivo" o varias líneas que definirán parte de la geometría. Por esta razón se debe tener en cuenta el "*overtracing*" y la segmentación del trazo.

El término "overtracing" se define como el uso de varios trazos para representar una única línea [KQW06]. En la Figura 4.7 se muestra un ejemplo de "overtracing" aplicado al trazado de un arco, como si de un dibujo artístico se tratara. Precisamente con esta finalidad se ha creado el módulo artístico de la arquitectura propuesta, tratando de abordar esta forma de trazado, si bien como iniciación se ha contemplado únicamente el trazado de líneas y arcos.

Figura 4.7. Ejemplo de "overtracing"

Se denomina segmentación al proceso de dividir un trazado complejo en sus primitivas geométricas correspondientes, proceso que actualmente es un problema conocido en el campo del reconocimiento de bocetos [PG09, CVP09].

A su vez, el sistema debe proporcionar al usuario cierta libertad a la hora del trazado, lo que implica que se deben buscar todas las soluciones posibles basándonos en los trazos/fonemas existentes, sin que éstos se hayan tenido que introducir de una manera concreta. Este término se denomina "interspersing" de trazos desde objetos diferentes [SD08, HD02 y HD09], mejora que ha sido implementada en el reconocedor.

Así por ejemplo, el reconocimiento de una cota diametral pasa por la premisa de haberse reconocido previamente el trazo de dos ángulos y una línea (siendo indiferente el orden en el que hayan sido introducidos), cumpliendo a su vez ciertas condiciones de proximidad y posición relativa entre ambos trazos, definiendo por tanto una estructura jerárquica (Figura 4.8). Se entiende por trazo las formas introducidas por el usuario desde que apoya el lápiz hasta que lo levanta.

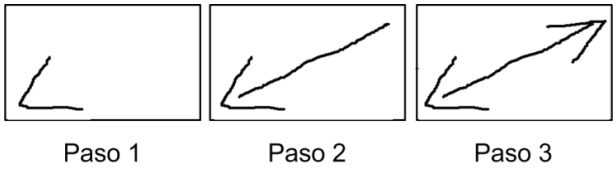


Figura 4.8. Ejemplo de secuencia de trazos que definen una cota diametral

En el núcleo central de la aplicación es donde se realiza el reconocimiento de todos los trazos introducidos por el usuario. En este trabajo se han diferenciado hasta un total de 9 trazos a los que se les ha denominado símbolos primitivos y a partir de los cuales es posible componer todos los símbolos definidos en los distintos módulos, conformando el diccionario de primitivos. Para identificar cada símbolo primitivo se ha definido un agente, y al conjunto de éstos se les ha denominado agentes primitivos (Figura 4.9).

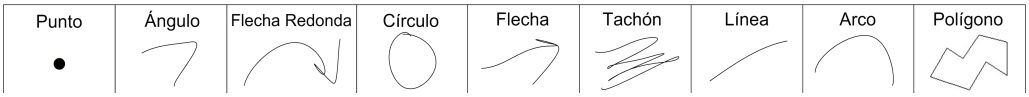


Figura 4.9. Diccionario de símbolos primitivos

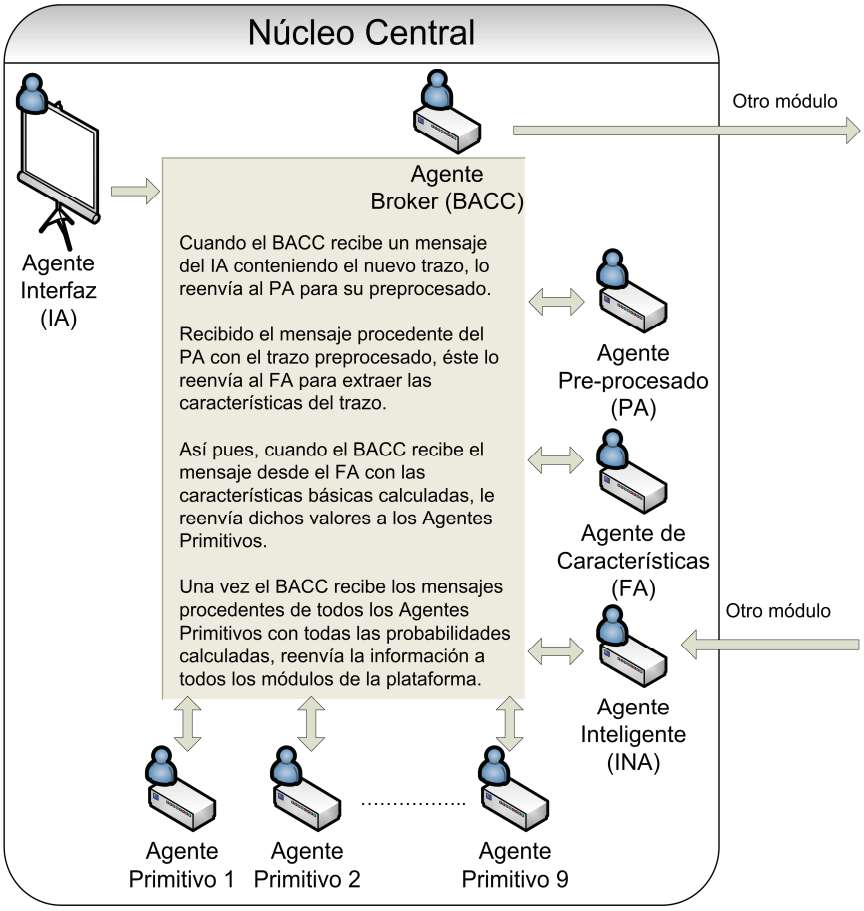


Figura 4.10. Estructura del Núcleo Central

Cuando un usuario dibuja un trazo, el agente interfaz IA lo muestra y envía los puntos digitalizados a través de una estructura de datos al agente Broker del núcleo central (BACC). El agente Broker BACC recibe la notificación y envía los puntos del trazado al agente de pre-procesado PA o *Pre-processing Agent*. De forma ideal, estos puntos deberían estar distribuidos de forma uniforme. Sin embargo, cuanto más rápido se dibuja el trazo, menos puntos se digitalizan, lo que provoca una variación en la concentración de puntos, además de que el trazado pueda ser más o menos tembloroso o titubeante. El agente PA debe filtrar y eliminar este ruido para conseguir un trazo óptimo listo para el reconocimiento.

Una vez se ha completado el pre-procesado, el agente BACC solicita la extracción de características al agente FA o *Feature Agent*. El agente de extracción de características FA posee un funcionamiento secuencial, distinguiéndose dos partes claramente diferenciadas. Por un lado se realiza la "segmentación" del trazo, extrayendo las primitivas geométricas que lo integran y aportando información de las mismas que posteriormente se empleará en las etapas de decisión. Por otro lado se obtienen una serie de características o "*cues*" del trazo introducido. Estas características han sido escogidas de forma que sean invariantes a la escala, posición y orientación, como son los momentos de Hu [Hu62], la circularidad, y otros [WK00]. Además de las mencionadas también se calcula la FFT [TMH95] de la firma de radio (también denominada firma polar) y de la firma del "*arc length versus cumulative turning angle*" (también conocida como firma de dirección). Estas características se explican con más detalle en el apartado 4.5.

Una vez el agente FA ha extraído ambas informaciones, devuelve dichas características al agente BACC, el cual reenviará la información a los respectivos agentes primitivos.

Los agentes primitivos

Cada uno de los agentes primitivos empleará únicamente la información que le resulte significativa con el fin de encontrar pistas relevantes que le permitan reconocer el símbolo primitivo del cual se encarga.

Cada agente primitivo evalúa el trazo introducido y cuantifica su información a partir de dos procesos claramente diferenciados:

- Por un lado, se lleva a cabo un reconocimiento basado en la geometría del trazo, devolviendo un resultado de coincidencia o no-coincidencia tras un análisis sintáctico. Dicho análisis sintáctico se realiza sobre los vértices y las primitivas aproximadas a partir de la segmentación obtenida por el agente FA. La Figura 4.11 muestra un ejemplo de coincidencia para un gesto de extrusión.

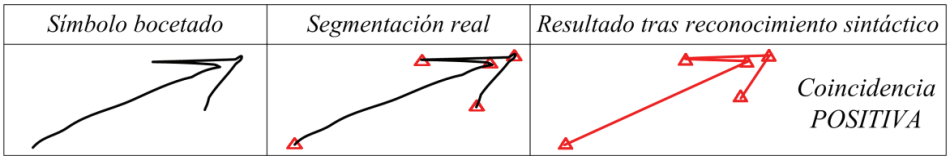


Figura 4.11. Coincidencia de los vértices y primitivas aproximadas

- Por otro lado, se calcula un valor cuantitativo como resultado de una función de maximización basada en un análisis discriminante Bayesiano no-lineal, cuyos parámetros de entrada son las características extraídas por el agente FA descritas más adelante en el apartado 4.4.5. Esta función de maximización se ha obtenido para cada clase, y maximiza su valor de salida

para una determinada clase y un vector observado de una forma bocetada cuando ésta pertenece a dicha clase (4-1).

$$P(w_i \mid x) = \frac{p(x \mid w_i)P(w_i)}{\sum_{j=1}^{m} p(x \mid w_j)P(w_j)}; i = 1,...m \qquad (4\text{-}1)$$

Donde *m* es el número de clases, *x* es el vector *n*-dimensional observado para una forma bocetada, w_i *(i=1..m)* es una de las *m* clases diferentes y $P(w_i)$ es la probabilidad "a priori" (sin conocerse previamente el vector observado) de que dicha forma pertenezca a una determinada clase, y que en este caso particular se ha considerado la misma para cada clase.

Los agentes primitivos evalúan y cuantifican la correspondencia o no de un trazo con su correspondiente símbolo primitivo pero no toman una decisión final, devolviendo así al agente BACC los resultados de la primera etapa de reconocimiento. El agente BACC se sirve de la información suministrada por los agentes primitivos para tomar la decisión sobre qué tipo de símbolo primitivo se trata e informar a los distintos agentes BAM de todos los módulos y al agente INA, iniciándose así la segunda etapa de reconocimiento.

La información facilitada por los agentes primitivos y el análisis realizado por el agente BACC será el punto de partida para intentar intuir qué símbolo está siendo esbozado por el usuario, o si por el contrario, el trazo forma parte de la geometría del boceto. Un agente inteligente (INA) será el encargado de tomar la decisión final, pero ésta dependerá, tanto de la información facilitada por el agente BACC como de la información que cada módulo de la aplicación envíe al agente INA.

4.4.3 Los módulos del sistema

Todos los módulos que formarán parte de la plataforma deben tener una estructura similar (Figura 4.12). En cada módulo se define un agente por cada símbolo con significado dentro del módulo. A estos símbolos se les denominará "gestos" y a cada agente encargado de su reconocimiento "Agente combinado". Los agentes combinados compiten entre sí para reconocer su gesto a partir de los símbolos primitivos que lo forman.

Cada módulo está gestionado por un agente broker (BAM), el cual tiene una doble función. Por un lado, envía a los agentes combinados de su módulo la información de cada trazo generada por los agentes primitivos y que recibe del agente broker del módulo central (BACC). Por otro lado, y una vez procesada esta información por los agentes combinados de su módulo, envía al agente inteligente del núcleo central (INA) los resultados del módulo que gestiona.

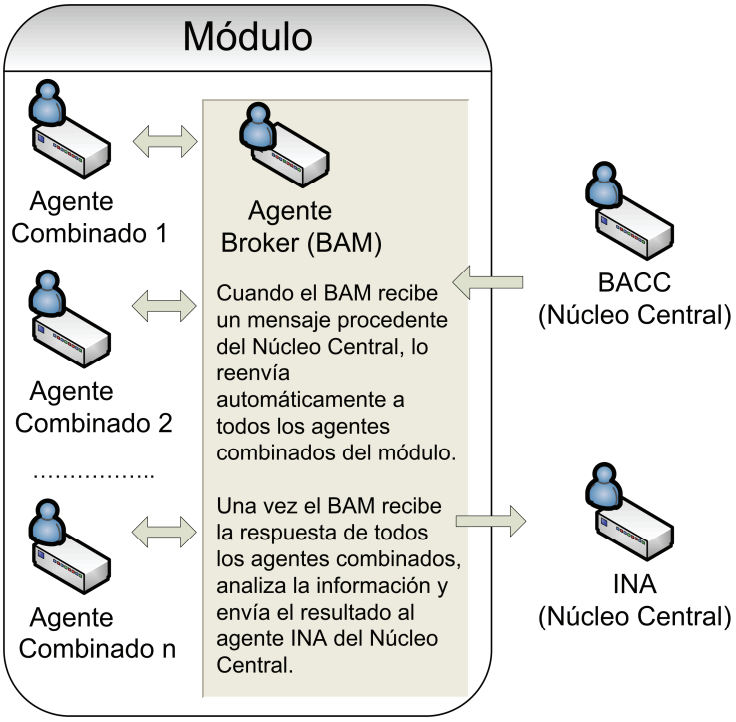


Figura 4.12. Estructura de los módulos

Los agentes combinados

A partir de la información que reciben del análisis realizado por los agentes primitivos, los agentes combinados buscan el reconocimiento de su gesto. Esta búsqueda se realiza a tres niveles:

1. Estudio contextual.

   Este análisis está basado en la posición relativa entre los distintos símbolos primitivos reconocidos y las proporciones entre los mismos. En el capítulo 5 se explica con más detalle el funcionamiento de los distintos agentes combinados, exponiendo los distintos análisis implementados para cada agente.

2. Premisas para que el gesto tenga significado.

   Es importante distinguir las distintas acciones que vienen asignadas a cada gesto combinado. Así pues, podemos clasificar en tres grupos las tareas que suponen el reconocimiento de un gesto:

   - Agentes de modelado (extrusión y revolución): son aquellos cuya ejecución precisa de la selección previa de una superficie. El reconocimiento del gesto por parte de estos agentes combinados se realiza únicamente si existe tal

selección. El agente BACC será el encargado de informar a cada modulo si existe o no la selección previa de dicha superficie.

- Agentes con referencias (paralelismo, perpendicularidad, igualdad, tangencia, concentricidad): son agentes que deben establecer una relación entre dos entidades geométricas, por lo que el reconocimiento del gesto pasa por la premisa de que debe haber sido introducido dos veces consecutivas.
- Agentes independientes (verticalidad, horizontalidad, tachado, cota radial, cota lineal, cota diametral, eje de revolución/simetría, sección, línea y arco artísticos): son agentes que suponen un reconocimiento inmediato, independientemente de cualquier otro gesto introducido previamente y no precisan de selección alguna.

3. Interpretación semántica de los trazos.

Es preciso tener presente que un trazo puede representar por sí solo un gesto o formar parte de un gesto más complejo, y que un mismo gesto puede estar formado por un mismo conjunto de trazos pero introducidos en distintos orden. A modo de ejemplo el gesto de cota diametral podría ser reconocido a partir una de las combinaciones de símbolos primitivos que se muestra en la Figura 4.13, al igual que ocurre con el gesto de revolución (Figura 4.14).

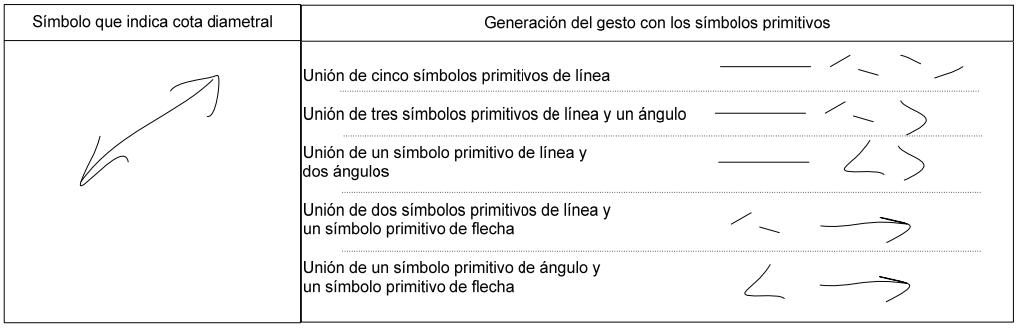


Figura 4.13. Posibilidades de bocetar el comando gestual de cota diametral

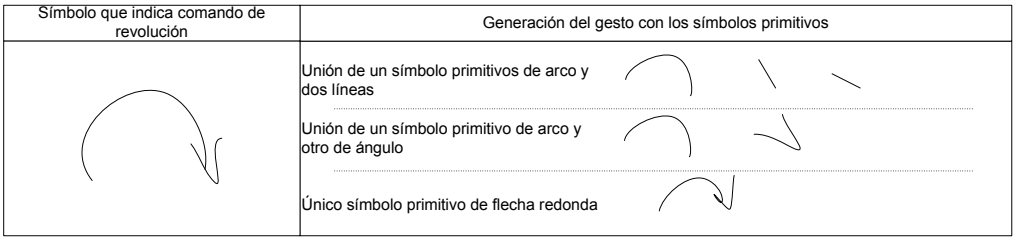


Figura 4.14. Posibilidades de bocetar el comando gestual de revolución

Además, considerando el diccionario de gestos definido para el módulo de acotación que se muestra en la Figura 4.15, el gesto de cota diametral podría formar parte de un gesto más complejo reconocido por el agente combinado cota lineal.

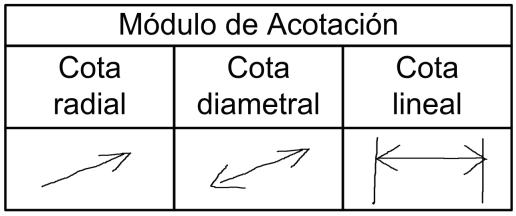


Figura 4.15. Diccionario de gestos del módulo acotación

A partir de la información que reciben del análisis realizado por los agentes primitivos, los agentes combinados buscan el reconocimiento de su gesto realizando un estudio contextual que haga al gesto independiente del orden en el que son introducidos sus símbolos primitivos. De esta forma, los símbolos primitivos enviados al agente combinado son añadidos a una lista. A continuación, el agente combinado analiza si el conjunto de símbolos primitivos en la lista constituyen o forman parte del gesto que busca.

De forma paralela, y una vez realizados los distintos análisis mencionados, los agentes combinados actualizan su estado, pudiendo tomar tres valores:

- Rechazado: si el símbolo primitivo no forma parte del gesto.
- En proceso: si símbolo primitivo por sí solo, o junto con los símbolos primitivos enviados anteriormente son susceptibles de formar parte del gesto.
- Aceptado: si el conjunto de símbolos primitivos existentes en la lista identifican el gesto.

Una vez actualizado su estado, el agente combinado informa al agente broker de su módulo.

Los agentes broker

Los agentes BAM se encargan de llevar a cabo dos funciones. La primera de ellas es la de ejercer de mero puente para enviar a los agentes combinados la información suministrada por los agentes primitivos y recibida a través del BACC, con la que poder comenzar la etapa de reconocimiento correspondiente.

La segunda función, más compleja, es la de gestionar los resultados que arrojan cada uno de los agentes combinados e informar al agente inteligente de las conclusiones de su módulo. La gestión se realiza a partir de la comparación de los estados de los agentes combinados.

Cuando un agente broker recibe un nuevo símbolo primitivo, éste es enviado a los agentes combinados cuyo estado se encuentre en el estado “en proceso”. Una vez recibidos

los informes de actualización de estado de todos los agentes combinados del módulo, el agente broker emite un informe al agente inteligente que puede contener tres estados:

- Rechazado: si todos los agentes combinados indican el estado “rechazado”.
- En proceso: si al menos uno de los agentes combinados indica el estado “en proceso”.
- Aceptado: si uno de los agentes combinados indica el estado “aceptado” y todos los demás el estado “rechazado”.

En la Figura 4.16 se muestra un ejemplo en el que se observan los estados por los que pasan los agentes combinados y el agente BAM del módulo de acotación durante el reconocimiento de una cota lineal.

Inicialmente todos los agentes se encuentran en el estado “en proceso”. Tras los dos primeros trazos, el estado del agente combinado de cota radial pasa al estado “aceptado”, dado que la combinación de un ángulo y una línea en las posiciones relativas esbozadas definen su gesto, mientras que los estados del resto de agentes combinados no se modifican dado que la combinación de los trazos pueden formar parte de su gesto. El BAM permanece en estado “en proceso” dado que al menos un agente combinado permanece con el estado “en proceso”.

El bocetado del tercer trazo provoca que el agente combinado cota radial pase al estado “rechazado”, dado que el nuevo trazo introducido no forma parte de su gesto, mientras que el agente combinado cota diametral pasa al estado “aceptado” al definir todos los trazos introducidos su gesto. El agente combinado cota lineal sigue en el estado “en proceso” esperando los nuevos trazos que definan su gesto, y por lo tanto el agente BAM no modifica su estado.

En el paso cuarto sólo el agente combinado cota lineal permanece en estado “en proceso” a la espera de un nuevo trazo mientras que el resto de agentes combinados están en estado “rechazado”, dado que el conjunto de trazos no forman sus gestos. El agente BAM permanece invariante.

Cuando finalizamos el gesto con el quinto trazo, el agente combinado cota lineal pasa al estado “aceptado”, dado que todos los trazos definen su gesto, y el agente BAM pasa al estado “aceptado”, dado que en su módulo sólo existe un agente combinado con el estado “aceptado” y el resto se encuentran en “rechazado”.

	Estado inicial	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5
Trazo						
Estado del Agente Cota Radial	En proceso	En proceso	Aceptado	Rechazado	Rechazado	Rechazado
Estado del Agente Cota Diametral	En proceso	En proceso	En proceso	Aceptado	Rechazado	Rechazado
Estado del Agente Cota Lineal	En proceso	En proceso	En proceso	En proceso	En proceso	Aceptado
Respuesta del BAM	En proceso	En proceso	En proceso	En proceso	En proceso	Aceptado

Figura 4.16. Ejemplo de estados e informes del módulo de acotación.

4.4.4 El agente inteligente INA

El agente inteligente del núcleo central es el encargado de gestionar los informes procedentes de los agentes BAM de los distintos módulos instalados en la plataforma y tomar la decisión final sobre el reconocimiento del gesto introducido, estableciéndose así una especie de 'negociación' entre el agente INA y los agentes Brokers de los distintos módulos.

La gestión realizada por el agente inteligente está basada en la información dada por los agentes primitivos y los estados de los agentes BAM de los distintos módulos. En su estado inicial, los agentes BAM se inicializan al estado "en proceso".

Para el correcto funcionamiento del algoritmo se precisa definir una lista, en la que se van introduciendo todos los trazos realizados por el usuario. Cada vez que se introduce un nuevo trazo, y tras ser éste pre-procesado, el agente BACC envía sus características a los agentes broker de los módulos que se encuentren en el estado "en proceso". Cada agente BAM establece comunicación con sus respectivos agentes combinados y actualizan sus estados, notificándolos posteriormente al agente INA.

Una vez el agente INA ha recibido los mensajes de actualización de todos los agentes BAM, toma su decisión, basándose en el siguiente guión (ver Figura 4.17):

1. Si el trazo no es reconocido por ningún agente primitivo:
    a. Si algún agente BAM "en proceso" contiene un agente combinado cuyo estado es "aceptado", se reconoce su gesto y el nuevo trazo se considera geometría.
    b. En caso contrario la lista y el nuevo trazo se consideran geometría.

    Finalizada la comprobación se borra la lista.
2. Si el trazo es reconocido por algún agente primitivo:

a. Si el trazo es reconocido por el agente primitivo de tachado:
    i. Si algún agente BAM tiene un agente combinado cuyo estado es "aceptado" se reconoce su gesto.
    ii. En caso contrario, la lista se considera geometría.
    En cualquiera de los casos anteriores y a posteriori, se reconoce el símbolo primitivo de tachado y se borra la lista.
b. En caso contrario el trazo se añade a la lista y se espera la actualización de los agentes BAM:
    i. Si el estado de todos los agentes BAM es actualizado a "rechazado":
        1. Si previo a añadir el último trazo algún agente BAM tenía un agente combinado cuyo estado era "aceptado" se reconoce su gesto, se borra la lista y vuelve a introducirse el último trazo para iniciar el proceso de reconocimiento, ordenando previamente a todos los agentes BAM y combinados a actualizar su estado a "en proceso" (inicialización).
        2. En caso contrario, la lista se considera geometría y se borra su contenido.
    ii. Si existe algún agente BAM cuyo estado se encuentre "en proceso", se espera al siguiente trazo.
    iii. Si todos los agentes BAM tienen su estado como "rechazado" y sólo uno mantiene su estado como "aceptado", se asume que la lista es reconocida como gesto. El agente combinado del módulo en cuestión cuyo estado se mantenga en "aceptado" indica de qué tipo de gesto se trata. Finalizada la comprobación se borra la lista.

Cada vez que se borra la lista, el agente INA envía un mensaje de inicialización al agente broker BACC, para que éste a su vez les comunique a los agentes BAM de cada módulo que deben pasar al estado "en proceso". Éstos a su vez deberán informar a los agentes combinados de su módulo que también deben pasar al estado "en proceso".

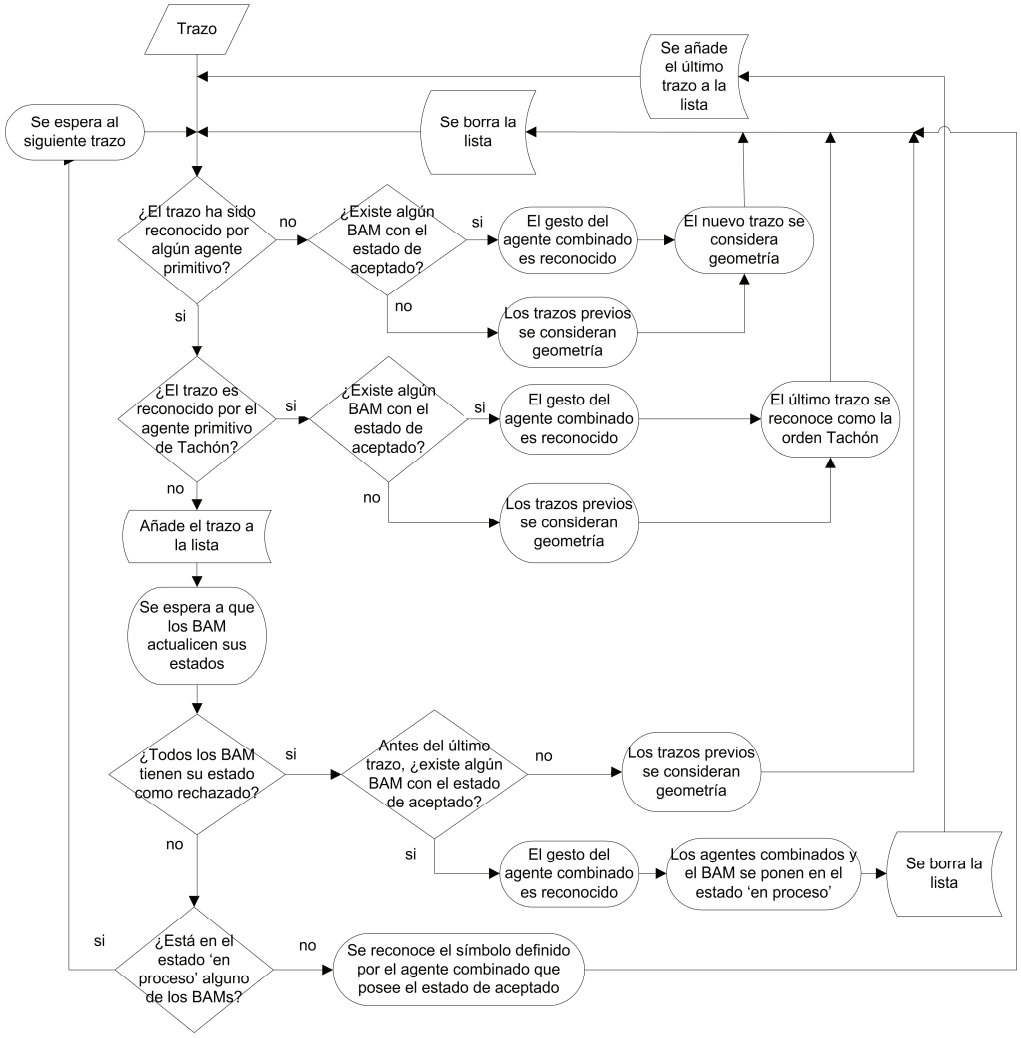


Figura 4.17. Diagrama de flujo de la toma de decisión del agente INA

Para ilustrar mejor el funcionamiento de la plataforma, en la Figura 4.18 se analiza de nuevo el reconocimiento de una cota lineal, mostrando los estados de todos los agentes combinados y agentes broker de los módulos de restricciones, acotación, modelado y artístico:

1. Estado inicial: Todos los agentes se inicializan al estado “en proceso”.
2. Paso 1: El primer trazo provoca que los agentes combinados concentricidad, igualdad, revolución y arco artístico pasen al estado “rechazado”, dado que una línea no puede formar parte de sus respectivos gestos.

Todos los agentes BAM permanecen con el estado “en proceso”, dado que al menos uno de sus agentes combinados se encuentra en ese mismo estado. Mientras tanto, el agente INA espera la entrada de un nuevo trazo, dado que al menos un agente BAM permanece con el estado “en proceso”.

3. Paso 2: El nuevo símbolo primitivo línea detectado, y a su vez paralela a la anterior, hace que los símbolos primitivos vertical, tangente, perpendicular, cota radial, cota diametral, extrusión, eje y línea artística pasen al estado “rechazado”. Sólo los agentes BAM de los módulos de restricciones, de acotación y de modelado permanecen en estado “en proceso”, por tener agentes combinados en este mismo estado.

   El agente INA espera la entrada de un nuevo trazo, ya que al menos un agente BAM permanece en el estado “en proceso”.

4. Paso 3: El nuevo trazo introducido elimina de la competición por la búsqueda de su gesto a los agentes combinados paralela y sección, que pasan al estado “rechazado”, mientras que el agente horizontal pasa al estado “aceptado”, ya que el conjunto de símbolos primitivos reconocidos hasta el momento conforman su gesto. El agente BAM del módulo de restricciones pasa al estado “aceptado”, dado que su agente combinado horizontal ha reconocido el gesto. El agente BAM del módulo de modelado pasa al estado “rechazado” al haber pasado el agente combinado de sección también a dicho estado.

   El agente INA espera la entrada de un nuevo trazo, pues al menos un agente BAM permanece con el estado “en proceso”.

5. Paso 4: El trazo reconocido como ángulo hace que el agente combinado horizontal pase al estado “rechazado”, así como el agente BAM de su módulo. Mientras el agente combinado cota lineal y el agente BAM de su módulo permanecen en el estado “en proceso”.

   El agente INA espera la entrada de un nuevo trazo, ya que al menos un agente BAM permanece con el estado “en proceso”.

6. Paso 5: El nuevo trazo reconocido como ángulo finaliza el gesto del agente combinado cota lineal, pasando su estado a “aceptado”, así como el estado de su agente BAM, ya que sólo existe en el módulo un agente combinado con el estado “aceptado”.

   El agente INA reconoce el gesto cota lineal al existir un único agente BAM en estado “aceptado”. Luego manda borrar la lista e inicializa todos los agentes (BAM y combinados) al estado “en proceso”.

7. Paso 6: Un nuevo trazo reconocido como símbolo primitivo de flecha inicializa el proceso de reconocimiento, pasando todos los agentes del módulo de restricciones y

del módulo artístico (BAM y combinados) al estado “rechazado”, manteniendo los agentes cota diametral y cota lineal del módulo de acotación con el estado “en proceso” y cambiando al estado “aceptado” los agentes cota radial, extrusión y BAM del módulo de modelado. Los agentes combinados revolución, eje y sección pasan al estado “rechazado".

El agente INA espera la entrada de un nuevo trazo, dado que al menos un agente BAM permanece con el estado “en proceso”.

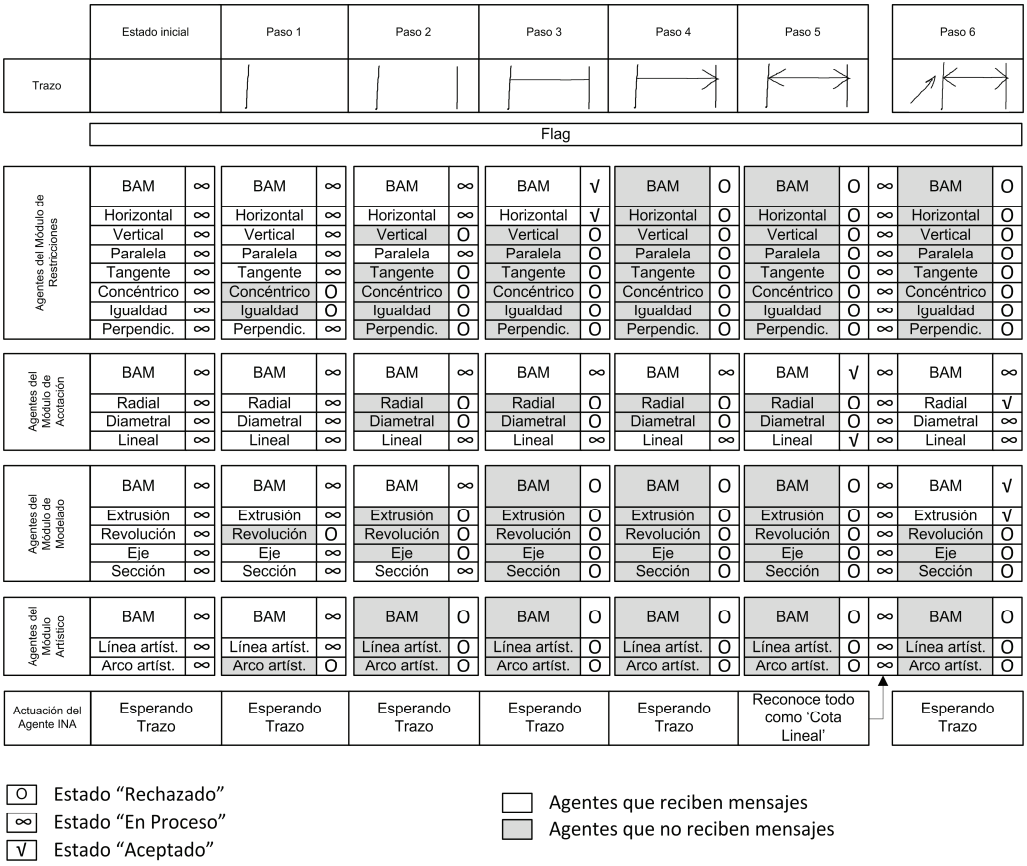


Figura 4.18. Estado de los agentes durante el reconocimiento de una cota lineal (Módulos activos: restricciones, acotación, modelado y artístico).

4.4.5 Ejemplo con *Interspersing*

La implementación del *interspersing*, esto es, la interrupción del gesto para realizar otra acción y posterior completado del gesto interrumpido, se ha llevado a cabo mediante un parámetro de valor entero positivo (≥ 0) que indica los trazos de espera para la aceptación de un gesto. Para ilustrar el funcionamiento se ha propuesto el ejemplo de la Figura 4.19, en la

que la variable *interspersing* se ha fijado con el valor 1. Observando la figura se aprecia que al introducir un arco de geometría en el paso tercero se realiza una pausa, contemplando la posibilidad de que se esté dando el *interspersing* y reduciendo el valor de la variable en una unidad. En consecuencia todos los agentes combinados posibles permanecen en el estado "en proceso", aunque el arco introducido no pertenezca al gesto en curso, esperándose una nueva entrada. La introducción en el paso quinto de un círculo conlleva la confirmación del gesto de cota diametral, pues al tener la variable de *interspersing* el valor 0 no permite de nuevo esta posibilidad. Una vez reconocido el gesto, dicha variable vuelve a fijarse al valor inicial.

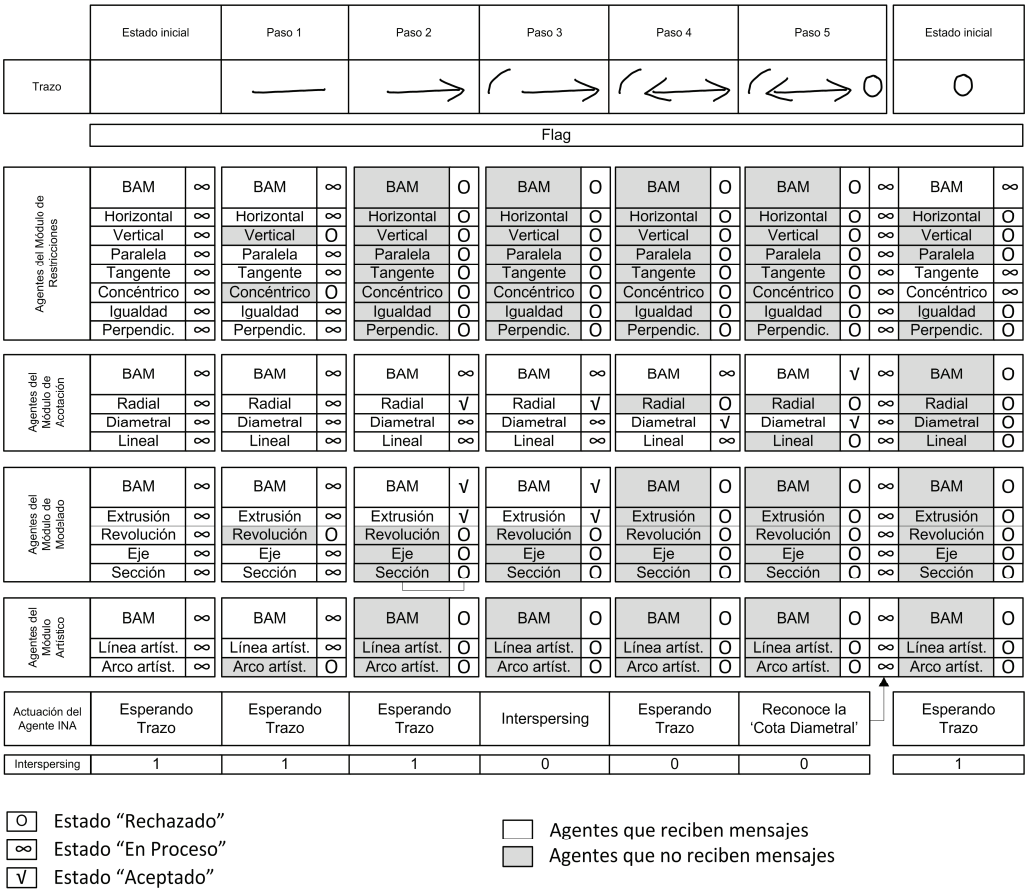


Figura 4.19. Ejemplo de reconocimiento de una cota diametral con *interspersing* (siendo *interspersing=1)*.

No se recomienda un valor superior a la unidad para la variable *interspersing*. En el caso de que su valor fuese superior a la unidad, los agentes broker de los módulos (BAMs) esperarían tantas entradas como valor indicase dicha variable. Si los valores de *interspersing* son superiores, se corre el riesgo de que el usuario no vea sus acciones realizadas

inmediatamente, con lo que el efecto de realimentación que debe realizar el interfaz no se produce, generando retrasos indeseados y confusión.

4.5 *Cues* básicas

Existen muchas características que se pueden extraer de un gesto/símbolo y que posteriormente servirán de ayuda a la hora de clasificar los trazos introducidos. En el presente trabajo se ha escogido para su estudio y empleo aquellos descriptores que son invariantes al tamaño y la rotación, ya que los esbozos pueden aparecer en cualquier posición, tamaño y orientación en el espacio. A estas características las llamaremos Cues básicas, y son pistas o indicios de la forma que puede tener el trazo. Atendiendo a estos requerimientos de invarianza, los descriptores utilizados son los momentos de Hu, la circularidad, el perímetro del "Convex-Hull" y la transformada rápida de Fourier.

Por otro lado, se lleva también a cabo la segmentación del trazo con el fin de extraer la información de las primitivas geométricas que lo integran.

A continuación se describe la extracción y cálculo de las características mencionadas, así como el proceso llevado a cabo para la segmentación del trazo (extracción de las primitivas que lo componen).

4.5.1 Los momentos de Hu

Hu deduce un conjunto de momentos a partir de invariantes algebraicos. En particular, Hu define siete valores calculados normalizando los momentos centrales de tercer orden, que son invariantes al tamaño, posición y orientación. Los momentos utilizados aquí son los seis primeros, cuyas ecuaciones se detallan en [Hu62]. En términos de los momentos centrales, los siete momentos de Hu se escriben como:

$$
\begin{aligned}
h_1 &= \eta_{20} + \eta_{02} \\
h_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^{\ 2} \\
h_3 &= (\eta_{30} - \eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
h_4 &= (\eta_{30} - \eta_{12})^2 + (\eta_{21} - \eta_{03})^2 \\
h_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
h_6 &= (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
h_7 &= (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]
\end{aligned}
\qquad (4\text{-}2)
$$

Donde $\eta_{i,j}$ son los momentos centrales normalizados de 2do y 3er orden. Todos ellos son invariantes a la escala y rotación, excepto el séptimo que cambia el signo con la simetría. Para nuestro reconocedor, se han utilizado únicamente los momentos de Hu h_1-h_6, puesto que el término h_7 no aporta información relevante.

4.5.2 La circularidad

En el caso de la circularidad *C*, se ha utilizado la ecuación general (4-3), donde *P* es el perímetro del contorno cerrado del trazo y *A* el área contenida dentro de dicho contorno [WK00]. Son varias las comprobaciones que demuestran que en el caso de que el trazo no sea cerrado, se deben unir los extremos del mismo y calcular así su contorno, haciendo de esta manera que dicha característica de circularidad sea invariante al tamaño.

$$C = \frac{P^2}{A} \geq 4\pi \qquad (4\text{-}3)$$

4.5.3 El perímetro del Convex-Hull

Se define el Convex-hull como un polígono convexo que contiene la forma introducida. Concretamente, el algoritmo para su obtención calcula los ejes principales de inercia (color amarillo) y su perímetro (color verde), para luego estirarlo como si de una goma se tratara y calcular así el polígono convexo (color rojo) [BOK00]. Ver figura Figura 4.20.

El Convex-hull es una característica que puede ser muy útil a la hora de diferenciar unos trazos de otros. Por ejemplo, el trazo de tachón que hará las veces de borrado en operaciones de edición de la geometría, se distingue muy fácilmente a partir del ratio entre su perímetro y el perímetro del polígono convexo que lo contiene [Alb06]. A continuación se explica el algoritmo seguido para su obtención.

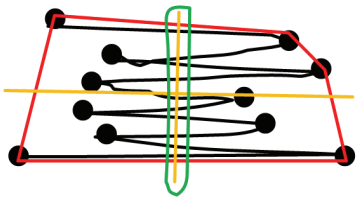


Figura 4.20. Perímetro del Convex-Hull para el gesto de tachón

Para exponer la creación de este contorno, es importante concretar previamente varios aspectos acerca de los valores angulares:

- Se considerará como sentido positivo el sentido anti-horario.
- El rango de los ángulos se considera comprendido entre –180º y 180º, por lo que cuando se realicen restas de ángulos se deberá comprobar que el resultado continua en el rango de trabajo, y se deberá reajustar en caso contrario

Inicialmente, se tiene un conjunto con los puntos del trazo introducido. Dicho conjunto, constituirá la “nube de puntos” donde se encontrarán los vértices del menor polígono convexo que los incluye. Para definir el polígono, es necesario determinar cuáles son sus vértices y el orden en que se tomarán, para lo que se sigue un algoritmo de naturaleza recursiva, en el que se empieza con un polígono formado por dos vértices (es un polígono ficticio, puesto que no

tiene área y simplemente es un segmento de línea) al que se le irán añadiendo vértices de forma que el polígono va "engordando" hasta incluir a todos los puntos de la nube. En la Figura 4.21a aparece una nube de puntos (muy poco densa en comparación con un caso real) que emplearemos como ejemplo para calcular paso a paso el mínimo polígono convexo que los incluye.

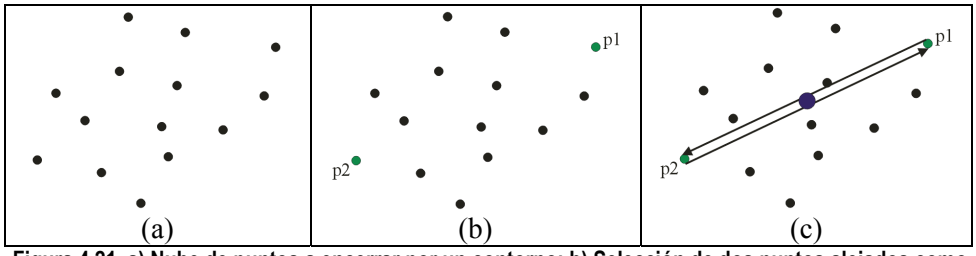


Figura 4.21. a) Nube de puntos a encerrar por un contorno; b) Selección de dos puntos alejados como puntos iniciales; y c) Polígono ficticio de dos lados y centro

Como se ha dicho anteriormente, partiremos de un "polígono" de dos lados, por lo que será necesario obtener los dos vértices iniciales, cuyo único requisito será que estén muy alejados. El método elegido aquí ha sido, en primer lugar calcular el centroide de la nube de puntos como la media de las coordenadas de todos ellos, seguidamente tomar el punto más alejado del centroide (al que llamaremos punto 1, o de forma simplificada *p1*), y a continuación tomar el punto más alejado al punto 1 (al que llamaremos punto 2, o *p2*). En la Figura 4.21b, se pueden ver los dos vértices iniciales *p1* y *p2* para el ejemplo que se está mostrando.

Los puntos *p1* y *p2* definirán el "polígono" inicial formado por sólo 2 lados: *p1p2* y *p2p1* y, además, el punto situado a medio camino entre *p1* y *p2*, ejercerá la función de "centro" del polígono. Este centro, se tomará como origen para calcular los ángulos de las coordenadas polares de cada uno de los puntos de la nube. Al estar situado en la recta *p1p2*, la diferencia de ángulo entre *p1* y *p2* será de 180º. Si restamos el ángulo de *p1* al ángulo de cada punto, el ángulo de *p1* será 0º y el de *p2* será 180º, o también -180º. En la Figura 4.21c aparece el punto central y los dos lados del polígono representados mediante los dos vectores *p1p2* y *p2p1*, que se han separado ligeramente para una mejor apreciación.

Así pues, el lado *p1p2* barre todos los ángulos entre 0º y 180º, mientras que el lado *p2p1* barre los ángulos entre -180º y 0º, lo cual es equivalente a decir que los puntos con ángulo entre 0º y 180º podrán ser vértices entre *p1* y *p2* del polígono buscado, mientras que los que tienen ángulo entre -180º y 0º podrán ser vértices de dicho polígono entre *p2* y *p1*.

A partir de aquí, se inicia un proceso recursivo que se repite para cada uno de los lados que tenga el polígono en cada instante. Este proceso se ejemplifica en la Figura 4.22a para el lado *p5p1* de un polígono en un estado ya avanzado de su formación:

- En primer lugar se comprueba qué puntos son los candidatos a formar nuevos lados del polígono insertándose entre el punto inicial y el punto final del lado a analizar. La característica de estos puntos es que el ángulo, desde

el centro del polígono, está en el rango entre los puntos inicial y final del lado, es decir, aquellos que están dentro del sector amarillo en la Figura 4.22a.

- Seguidamente, se calcula el ángulo del lado en proceso como el ángulo del vector que va desde el punto inicial al final del lado (vector *p5p1* en la Figura 4.22a), y también el ángulo entre el punto inicial del lado (*p5*) y cada uno de los puntos del sector que barre el lado (vectores de color rojo y verde en la Figura 4.22a).
- Se eliminan definitivamente de la nube de puntos los que quedan a la izquierda del lado (vectores de color rojo en la Figura 4.22a), es decir, aquellos para los que el ángulo desde el punto inicial del lado (*p5*) hasta ellos menos el ángulo del lado (*p5p1*) sea positivo.
- Serán candidatos a vértice nuevo entre los vértices inicial (*p5*) y final (*p1*) del lado los que queden a la derecha del lado (vectores de color verde en la Figura 4.22a), es decir, aquellos para los que el ángulo desde el punto inicial del lado (*p5*) hasta ellos menos el ángulo del lado (*p5p1*) sea negativo.

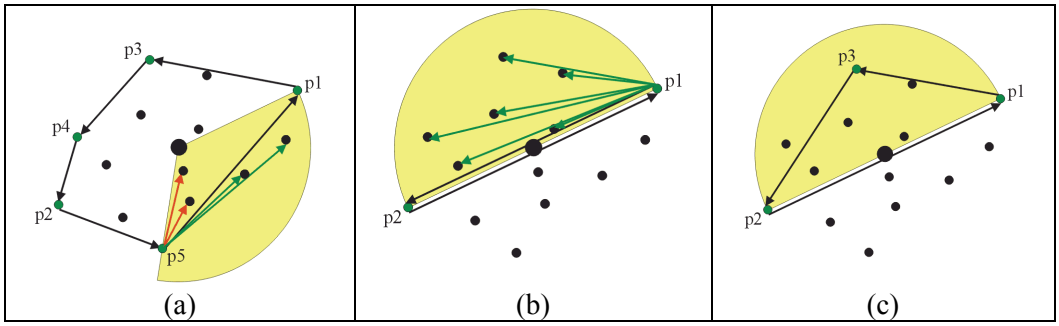


Figura 4.22. a) Puntos a eliminar (flechas de color rojo) y puntos candidatos a "engordar" el polígono (flechas de color verde) determinados por el lado *p1p5*; b) y c) Inserción del punto *p3* entre *p1* y *p2*, con lo que se sustituye el lado *p1p2* por 2 lados: *p1p3* y *p3p1*

De todos los candidatos a vértice nuevo, se elige el punto más alejado (al que llamaremos *p'*) del lado considerado (calculando la distancia de cada punto a la recta) y sustituiremos el lado en cuestión (de forma genérica *papb*), por dos nuevos lados: *pap'* y *p'pb* en el orden indicado. El proceso finalizará cuando ya no se añadan más vértices (habrán quedado todos eliminados o formarán parte del polígono). En las siguientes figuras se muestra el desarrollo completo para el ejemplo actual. En la Figura 4.22c aparece el resultado del proceso para el lado *p1p2* (Figura 4.22b): no se elimina ningún punto, ya que todos quedan a la derecha del lado *p1p2*, y se inserta el vértice *p3*, con lo que el lado *p1p2* es sustituido por los lados *p1p3* y *p3p2*.

En la Figura 4.23a se aplica el proceso al lado *p1p3*, que queda fijo, puesto que no se incluye ningún vértice, y se eliminan dos puntos. En la Figura 4.23b, se aplica al lado *p3p2*, que se divide en los lados *p3p4* y *p4p2* y quedan eliminados dos puntos (Figura 4.23c).

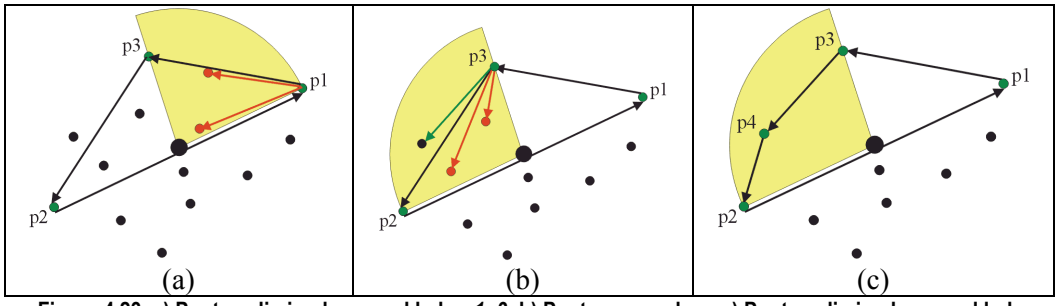


Figura 4.23. a) Puntos eliminados por el lado *p1p3*; b) Punto agregado; y c) Puntos eliminados por el lado *p3p2*

En la Figura 4.24a se empieza con la otra parte del polígono inicial, añadiendo un punto entre *p2* y *p1* (Figura 4.24b). En la Figura 4.24c se comprueba que el lado *p2p5* ya no se puede dividir pero se elimina un punto, mientras que en la Figura 4.24d mediante el lado *p5p1* se eliminan dos nuevos puntos, y se inserta *p6* (Figura 4.24e). Finalmente, en la Figura 4.24f, entre *p5* y *p6* se elimina el último punto que quedaba libre.

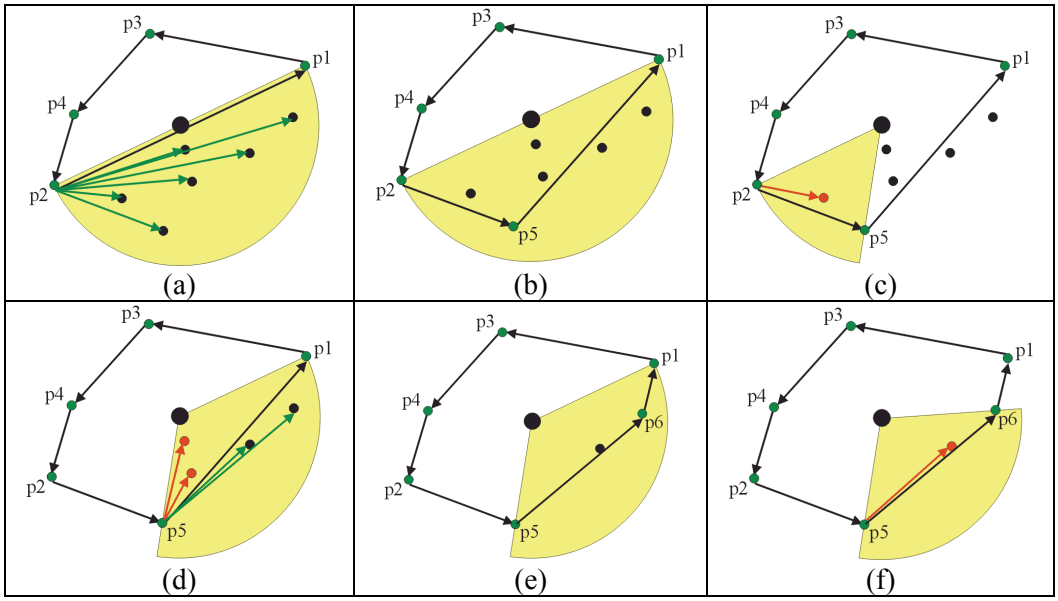


Figura 4.24. En este orden: a) y b) Punto agregado entre *p2* y *p1*; c) Punto eliminado entre *p2* y *p5*; d) Puntos eliminados entre *p5* y *p1*; e) Punto añadido *p6*; y f) Punto eliminado entre *p5* y *p6*

Si se realiza el proceso recursivo siguiendo el orden de poner primero en la lista de vértices *p1*, se debe efectuar la recursión sobre el lado *p1p2*, poner en lista de vértices *p2* y efectuar la recursión sobre el lado *p2p1*, y dentro de cada recursión al encontrar un nuevo punto, se debe dividir el lado, efectuar la recursión en el primer sub-lado, poner en la lista el vértice intermedio y realizar la recursión en el segundo sub-lado. Al finalizar quedarán en la

lista los vértices del polígono en orden correcto, que en el ejemplo anterior es: *p1*, *p3*, *p4*, *p2*, *p5* y *p6*.

En la Figura 4.25 aparece el polígono final, con todo el resto de puntos incluidos en su interior.

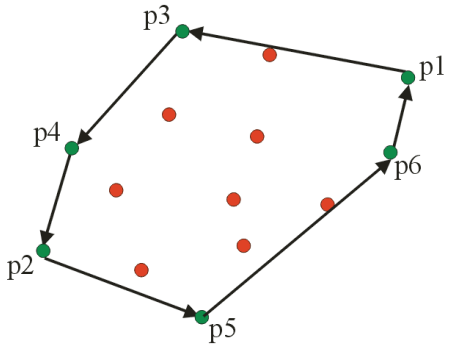


Figura 4.25. Puntos del polígono convexo y puntos incluidos

El algoritmo se escribe de la siguiente manera:

```
p_centroide=CalcularCentroide();
p_inicial  =ObtenerPuntoMasAlejado(p_centroide);
p_final    =ObtenerPuntoMasAlejado(p_inicial);
p_centro   =CalcularPuntoMedio(p_inicial,p_final);

Repetir {
        //Para todos los lados del polígono
cto_puntos =CalcularPuntosCandidatosPoligono();
angulo_lado=CalcularAngulo(p_inicial,p_final);

Repetir {
                //Para todos los puntos candidatos encontrados en
        cto_puntos
        angulo=CalcularAngulo(p_inicial,p_candidato);
        Si (angulo-angulo_lado > 0) {
                        cto_puntos=EliminarPuntoCandidato();
                }
        Sino {
                        cto_puntos=GuardarPuntoCandidato();
                }
        } //fin bucle


distancia_maxima=0;
        vertice_poligono=null;
        Repetir {
                //Para todos los puntos candidatos restantes en
        cto_puntos
        distancia=CalcularDistancia(p_candidato,lado_poligono);
        Si (distancia > distancia_maxima) {
                vertice_poligono=p_candidato;
                        distancia_maxima=distancia;
```

```
                }
            } //fin bucle

            Si (vertice_poligono <> null) {
                    cto_puntos=EliminarPuntoCandidato(vertice_poligono);
            AñadeLadoPoligono(vertice_poligono);
            VerticesEncontrados=true;
            }
            Sino
            VerticesEncontrados=false;

    } Mientras (VerticesEncontrados);
```

4.5.4 Los descriptores de Fourier

La transformada discreta de Fourier (DFT – *Discrete Fourier Transform*) se ha utilizado ampliamente para análisis de espectros de señal en el dominio de la frecuencia. La transformada rápida de Fourier (FFT - *Fast Fourier Transform*) es un algoritmo [TMH95] para el cálculo de la transformada de Fourier de un conjunto de datos discreto, el cual debe estar normalizado, pues el algoritmo FFT requiere como entrada 2^n elementos. Dado un conjunto de puntos, la FFT expresa esos datos en términos de sus componentes de frecuencia, que son particulares para cada gesto. El resultado de la FFT da una serie de números que representa sus armónicos. En nuestro caso, la transformada rápida de Fourier (FFT) se ha calculado a partir de tres firmas del trazo. Estas firmas son:

1) La distancia de los puntos del gesto al centroide (conocida como *firma de radio* o firma polar).
2) La dirección de los puntos del gesto (también denominada *arc length versus cumulative turning angle* o firma de dirección).
3) La curvatura en cada uno de los puntos del gesto (denominada firma de curvatura).

Para la obtención de la primera firma, primero se calcula el centroide como la media de las coordenadas *X,Y* de todos los puntos del trazo. Luego se calculan las distancias Euclídeas desde cada punto del trazo al centroide y se almacenan en un vector. A esta firma se le llama firma de radio (*radius signature*), también conocida como “firma polar”.

Para la obtención de la segunda firma, se utiliza la definición que Yu propone en [Yu03] (ver ecuación 1-1), de modo que cuando un gesto mantiene su dirección, sus puntos están alineados y no hay incremento de ángulo (ver ejemplo de la Figura 4.26). En caso contrario el incremento es distinto de cero. Esta firma se denomina *arc length versus cumulative turning angle signature*, que aquí y para abreviar llamaremos “firma de dirección”.

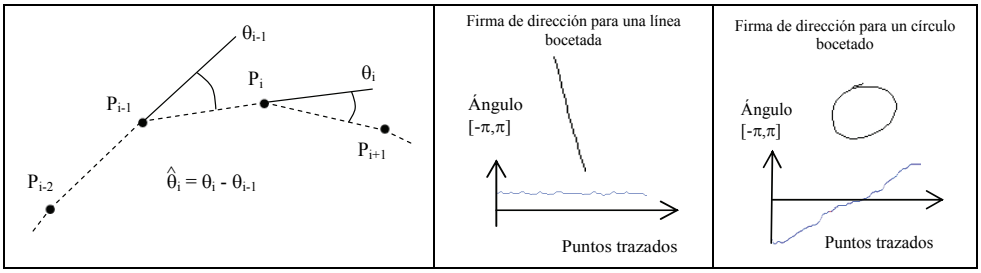


Figura 4.26. Ejemplo de puntos bocetados y su dirección (izquierda), firma de dirección de una línea (centro), y firma de dirección de un círculo (derecha)

En la Figura 4.27 se muestra, según la secuencia seguida de bocetado, un ejemplo de ambas firmas para un símbolo de cota diametral. A continuación se muestran las firmas unidas de todos los trazos del símbolo (Figura 4.28 izquierda), y ambas firmas completas por separado (Figura 4.28 derecha). Los dos símbolos ◙ corresponden a los puntos de flecha que están más separados del centroide. El símbolo ◘ corresponde al punto del trazo más próximo al centroide. En la Figura 4.29 se muestran ejemplos de firmas de otros símbolos esbozados.

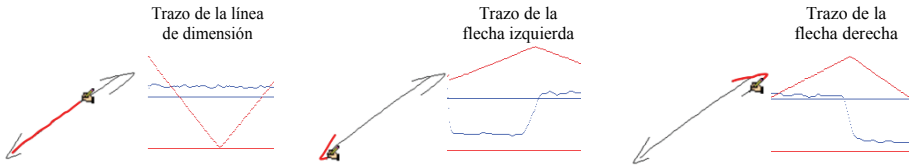


Figura 4.27. Cota diametral de la que se ha calculado la firma polar (en rojo) y la firma de dirección (en azul) de los tres trazos que la integran

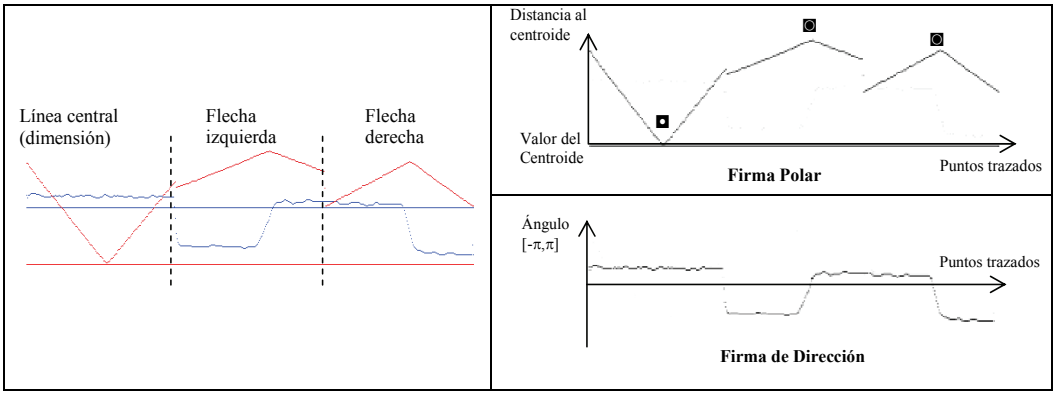


Figura 4.28. Firmas encadenadas (izquierda), descomposición de ambas firmas: firma polar arriba-der y firma de dirección abajo-der

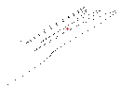
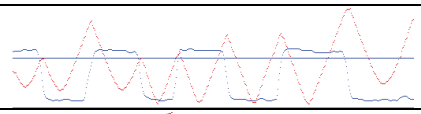
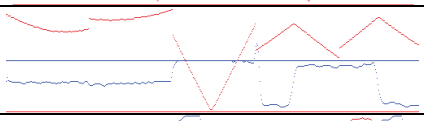
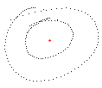
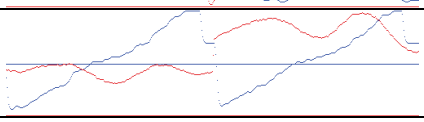
Esbozo	Esbozo pre-procesado	Firma polar (rojo) / Firma de dirección (azul)	Símbolo
			Tachón
			Dimensión
			Concéntrico

Figura 4.29. Ejemplos de símbolos combinados esbozados, y las firmas polar y de dirección de los strokes que los componen

Para la obtención de la tercera firma, empleamos la definición de curvatura también propuesta por Yu [Yu03] ya introducida en el punto de introducción (ver ecuación 1-2). En esta tesis, la FFT se calcula para las tres firmas. El número de armónicos usado para la clasificación es de 10, ya que Tao et al. [TMH95] demostraron que la información más relevante necesaria para reconstruir la forma original de los objetos estaba contenida en los primeros 10 armónicos. Como resultado se obtiene un total de 30 armónicos, que serán calculados por el agente FA junto con el resto de características extraídas.

4.6 La segmentación del trazo

En este contexto, se entiende como segmentación del trazo la separación de las primitivas que lo componen, para lo que se hace necesaria la detección de vértices que nos darán los puntos donde confluyen dichas primitivas. Son muchos y variados los métodos de detección de vértices implementados en la literatura, como ya se ha visto en el capítulo relativo al estado del arte de la tesis. Sin embargo, presentan problemas a la hora de identificar vértices en los cambios entre zonas rectas y curvas, y en los puntos de inflexión de curvas. De hecho, la mayoría de los métodos más recientemente desarrollados solo son capaces de detectar vértices esquina, dando resultados realmente bajos de detección de vértices tangentes, lo que impide la segmentación eficaz del trazo y, por tanto, la posterior interpretación del mismo. En la Figura 4.30 se muestran ejemplos de vértices esquina y vértices tangentes.

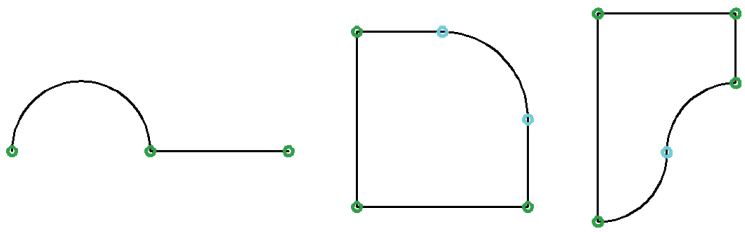


Figura 4.30. Formas que contienen vértices esquina (en color verde) y vértices tangentes (en azul cian)

Para esta tesis, se ha desarrollado un nuevo método de detección de vértices que mejora los resultados existentes respecto a los vértices esquina, y detecta con éxito los vértices tangentes. El método desarrollado consiste en identificar los vértices esquina, los tramos rectos y los tramos curvos de un trazo a partir de la función del radio, calculándose ésta como la inversa de la función de curvatura del trazo. Ésta se calcula a su vez a partir de la función de dirección de dicho trazo, calculándose tanto la dirección de la curvatura de forma aproximada a partir de diferencias. Así, primero se sitúan los vértices esquina en los lugares donde hay picos con menores valores absolutos de la función de radio. Después, dependiendo de la función del radio, los tramos entre los vértices detectados se aproximan a primitivas de línea o arco. Para diferenciar curvas y rectas se obtiene la función de radio de forma analítica, a partir de la aproximación del trazo mediante curvas cúbicas paramétricas entre pares de vértices esquina. De esta manera se consigue evitar el ruido inherente de la función de radio calculada de forma aproximada. Finalmente, los vértices tangentes se localizan entre primitivas consecutivas donde previamente no se han detectado vértices esquina. La descripción del método se realiza con todo detalle en el Anexo III.

Para probar la superioridad del método desarrollado, se han realizado diferentes tests que lo comparan con los métodos actuales que obtienen mejores resultados, y se observa una mejora significativa en la detección de vértices esquina, y más aún, un avance considerable en la detección de los vértices tangentes. Dos de los trabajos actuales con los que se compara son el ShortStraw [WEH08] y el IStraw [XL09], este segundo desarrollado para mejorar el primero. Concretamente, el IStraw soluciona ciertas limitaciones relacionadas con la detección de vértices en trazos que incluyen curvas. Aunque estos dos métodos presentan muy buenos resultados en la detección de vértices esquina, incluyendo una alta precisión en la segmentación todo-o-nada del trazo (*all-or-nothing accuracy*), presentan muchos inconvenientes acerca de la detección de vértices tangentes. Este inconveniente lo intentan solucionar Pu y Gur [PG09], los cuales aplican un método que también aproxima el trazo, pero en su caso utilizando una aproximación mediante funciones basadas en el radio ("radial basis functions" o RBFs), en lugar de curvas cúbicas paramétricas. El método que proponen también detecta vértices tangentes, aunque no es capaz de distinguir entre vértices esquina y vértices tangentes, siendo además muy elevado el número de falsos positivos (alrededor del 25%). A continuación se exponen brevemente los métodos ShortStraw e IStraw, ya que su descripción detallada y sencillez han permitido implementarlos para comparar resultados con el desarrollado exclusivamente para esta tesis y al que se ha llamado TCVD (*Tangent Corner Vertices Detection*). Seguidamente se menciona el método propuesto por Pu y Gur, para posteriormente, justificar y exponer el método TCVD, que se describe de una forma más completa en el Anexo III. Para finalizar se muestran algunas sugerencias de mejora para el TCVD.

4.6.1 ShortStraw

En 2008 Wolin et al. [WEH08] introdujeron su ShortStraw como "un detector de esquinas simple y efectivo para poli-líneas", esto es, para trazos sin curvas. Este método utiliza primero una aproximación *bottom-up* para la obtención inicial del conjunto de esquinas, y posteriormente realiza una aproximación *top-down* para añadir esquinas no detectadas y

eliminar falsas esquinas. Los pasos seguidos por el ShortStraw son primero un re-muestreo de puntos para espaciarlos uniformemente. Posteriormente obtiene los "*straws*", siendo calculado el *straw* para cada punto re-muestreado p_i como:

$$straw_i = \left| p_{i-W}, p_{i+W} \right| \quad (4\text{-}4)$$

Donde *W* es una ventana de tamaño constante igual a 3, y la expresión $|p_{i-W},p_{i+W}|$ es la distancia euclídea entre los puntos p_{i-W} y p_{i+W}. Una vez estos valores se han calculado, se obtiene el conjunto inicial de esquinas, consistente en todos los mínimos locales por debajo de un umbral basado en la mediana de los valores de straws obtenidos.

Tras la obtención de este conjunto de vértices, la aproximación *top-down* refina el conjunto a través de un test de linealidad: dos esquinas (puntos del trazo) en los índices *a* y *b* cumplen el criterio de linealidad si la distancia euclídea entre ellas y la distancia a través del *path length* (la suma de todas las distancias euclídeas entre los puntos re-muestreados que hay entre ellas) son relativamente iguales.

Esta comprobación se aplica primero para encontrar esquinas no detectadas, de modo que si esta comprobación entre dos esquinas consecutivas no se cumple, se añade una nueva esquina en el punto del trazo situado entre *a* y *b* que tenga el mínimo valor de *straw*. Este proceso se repite hasta que no se añaden más esquinas al conjunto inicial. Posteriormente, esta comprobación se aplica para eliminar falsos positivos, esto es, para cada esquina, si es posible la aproximación a línea entre sus dos esquinas adyacentes, la esquina central se elimina. Este proceso se repite hasta que no se eliminan más esquinas del conjunto.

4.6.2 IStraw

En 2009 Xiong y La Viola presentaron su IStraw [XL09] como una revisión del ShortStraw incluyendo algunas mejoras sobre el mismo. Estas mejoras consisten en la posibilidad de segmentar trazos que contengan curvas, el empleo de información sobre velocidad (dado que al dibujar un trazo, se reduce la velocidad en las esquinas), y otras mejoras como umbrales dinámicos.

La principal mejora que aporta IStraw consiste en una comprobación para eliminar los falsos positivos que aparecen en las curvas. Esta comprobación (Figura 4.31) se basa en la diferencia de ángulos entre una esquina falsa (C_i) posicionada en una curva, y dos pares de puntos re-muestreados (*A*-*B* y *D*-*E*). Los índices de *A*, *B*, *D* y *E* (calculados empíricamente) son *i-shift, i+shift, i-(shift/3)* e *i+(shift/3)*, donde $shift=min(15,C_i-C_{i-1},C_{i+1}-C_i)$, siendo C_{i-1} y C_{i+1} las esquinas previa y siguiente a C_i respectivamente. C_i se considera una esquina real si $(\beta-\alpha)<t_a$, donde el umbral $t_a=10+800/(\alpha+35º)$ depende del ángulo α (ya que $\beta-\alpha$ aumenta si α disminuye) y se calcula empíricamente.

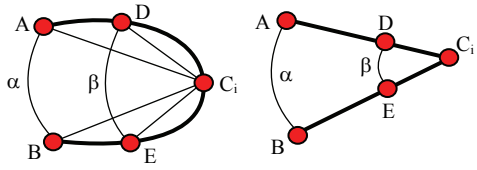


Figura 4.31. Diferencia entre una esquina falsa detectada en un tramo curvo (izquierda) y una esquina real entre dos segmentos rectos (derecha)

4.6.3 Pu y Gur

El método de Pu y Gur [PG09] también data de 2009 y, al igual que el método TCVD propuesto, emplea una aproximación matemática al trazado y obtiene vértices tangentes, no existiendo aparte más coincidencias. Estas son algunas de las características de Pu y Gur que difieren del TCVD:

- La aproximación no se realiza mediante curvas cúbicas paramétricas (como en el TCVD), sino mediante funciones basadas en el radio ("*radial basis functions*"). Además, se realiza al principio, no después de encontrar los vértices esquina.
- Los vértices se sitúan en los puntos que se van necesitando para que la aproximación se ajuste con precisión al trazo. Por este motivo, no se distingue entre vértices esquina y vértices tangentes. Evidentemente tampoco se encuentran rectas y curvas, que permitirían identificar a los vértices tangentes.
- Al igual que ShortStraw e IStraw realiza un uso intensivo de diversos refinamientos posteriores para "pulir" el conjunto de vértices.

4.6.4 El método de segmentación desarrollado: TCVD (*Tangent Corner Vertices Detection*)

4.6.4.1 La segmentación de los bocetos basada en el radio de curvatura

Como ya se ha mencionado en el capítulo del estado del arte, una forma común de detección de esquinas en bocetos consiste en buscar picos de máxima curvatura en valor absoluto (calculándose la curvatura como el cambio de dirección en cada punto). Sin embargo, el método desarrollado para la detección de vértices de esquina y tangentes (TCVD) usa el radio (inversa de la curvatura), lo que permite obtener las esquinas (picos de mínimo radio en valor absoluto) así como los arcos (secuencias de puntos para los que el radio en valor absoluto está por debajo de un determinado umbral). El tratar con arcos de esta manera, nos permite fijar umbrales más intuitivos para arcos y vértices esquina (los cuales pueden considerarse como arcos cortos de radio pequeño como se muestra en la Figura 4.32).

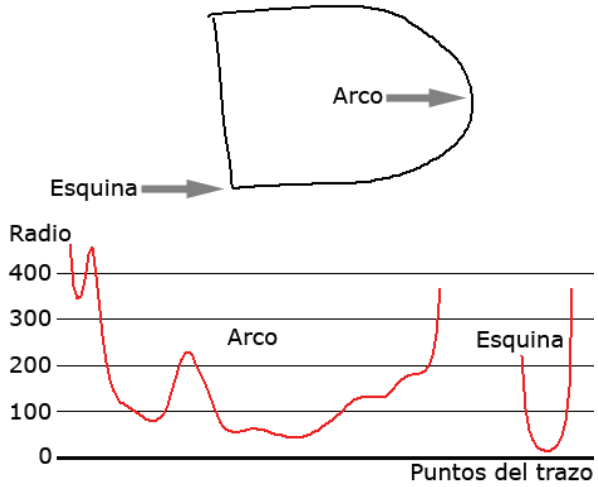


Figura 4.32. Valores de radio para arcos y esquinas en rojo (los valores altos de radio no se han representado)

Sin embargo, tanto la curvatura como el radio son muy sensibles al ruido. En la Figura 4.33 se muestra de arriba a abajo: a) un trazo con un arco y un vértice esquina con coordenadas obtenidas a partir de las ecuaciones de recta y arco; b) la función del radio obtenida a partir de los puntos del trazo en coordenadas con valores enteros (aunque se ha aplicado un ligero suavizado para eliminar parte del ruido); y c) la función del radio con el mismo suavizado obtenida a partir de las coordenadas reales de los puntos del trazo. Como puede observarse, el vértice esquina está bien definido mediante un pico de radio mínimo en ambos casos b) y c). Sin embargo, la función del radio para el tramo del trazo correspondiente al arco contiene mucho ruido debido al *aliasing,* o efecto escalera que se produce por la cuadrícula del ráster b).

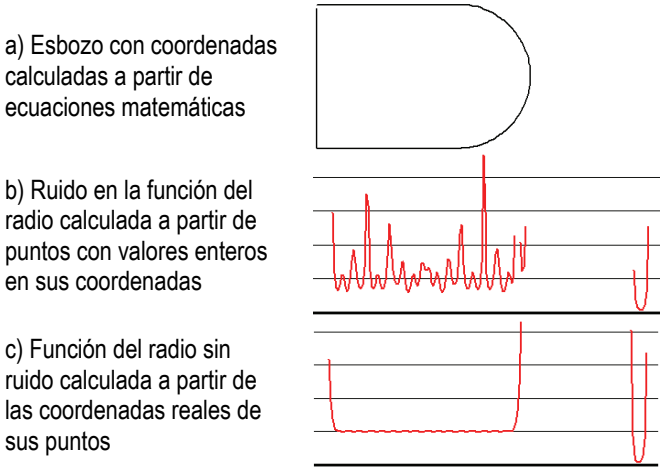


Figura 4.33. Efecto del ráster sobre la función de radio en el esbozo

Una forma de eliminar este ruido evitando la aplicación de filtros de suavizado a los puntos del trazo, pasa por obtener la aproximación de dicho tramo de arco a una curva paramétrica, de modo que la función del radio del arco se calcule a partir de las expresiones matemáticas de la curva paramétrica aproximada, eliminando así el ruido y las discontinuidades en los radios de arco (Figura 4.34).

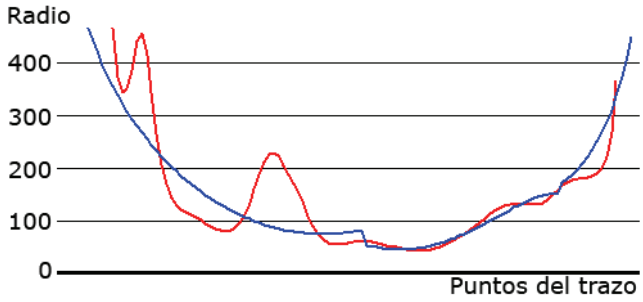


Figura 4.34. Diferencia entre el radio discreto obtenido a partir de diferencias (rojo) con un gran suavizado, y el radio analítico obtenido de las expresiones matemáticas de las curvas paramétricas (azul) para el arco de la Figura 4.32

Por otra parte, no es aconsejable realizar la aproximación mediante curvas cúbicas paramétricas antes de obtener los vértices esquina, ya que el mayor error de la aproximación siempre aparece en dicho tipo de vértices (dado que estamos realizando transiciones suaves en los vértices esquina), distorsionando los valores de la función de radio en los mismos.

4.6.4.2 Cálculo de la función discreta de radio

Para calcular la función discreta del radio primeramente se debe normalizar el tamaño de longitud del trazo, de modo que si la diagonal del *bounding box* del trazo es mayor que *MAX_DIAGONAL_BBOX=450*, éste se escala a dicho valor. Esto se realiza porque se ha demostrado que el método TCVD trabaja bien con valores de radios de arco de hasta 300. Luego, se lleva a cabo un re-muestreo del trazo [WWL07] con el inter-espaciado entre puntos *INTERSPACING_DISTANCE=3*, quedando los *n* puntos del trazo espaciados uniformemente.

Estos puntos son suavizados utilizando un filtro Gausiano para reducir los efectos del ruido que se producen en el cálculo de la dirección, curvatura y radio. La ventana utilizada para el filtro ha sido de tamaño *FILTER_WINDOW=8*. Los puntos suavizados se han obtenido por convolución discreta de los puntos re-muestreados con el filtro Gausiano.

La tangente en el punto se ha calculado a partir de las diferencias de coordenadas entre los puntos extremos de una ventana de tamaño *DIRECTION_WINDOW=2* centrada en la misma. La dirección del trazo en el punto es el ángulo calculado a partir de la tangente.

La curvatura en un punto se calcula a partir de las diferencias de los ángulos de dirección entre los puntos extremos de una ventana de tamaño *CURVATURE_WINDOW=1* centrada en ella.

Finalmente, el radio es la inversa de la curvatura.

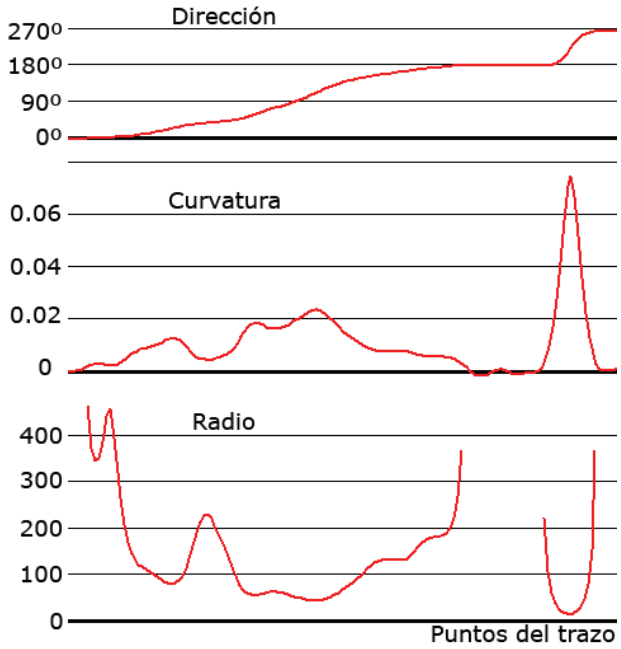


Figura 4.35. De arriba abajo: función de dirección del trazo (corregida para evitar discontinuidades), funciones de curvatura y radio respectivamente, calculadas a partir de la forma de la Figura 4.32

4.6.4.3 Cálculo de los vértices esquina a partir del radio

Las esquinas se localizan en los picos con mínimo valor absoluto de radio. Así, el umbral se ha fijado en *MAX_CORNER_RADIUS=60*, de modo que se buscan mínimos por debajo de dicho umbral. Estos puntos se consideran esquinas si cumple cualquiera de las siguientes dos condiciones (ver Figura 4.36).

- El mínimo está por debajo del umbral *CORNER_RADIUS=16*.
- El pico que forma el mínimo es muy acusado, es decir, el aumento del radio es grande en el entorno del mínimo local. Para comprobarlo se obtiene (sólo para radios del mismo signo) el radio medio en los *NEIGHBORING_WINDOW=8* puntos anteriores y el radio medio en los *NEIGHBORING_WINDOW=8* puntos posteriores al radio mínimo. Los cocientes de las divisiones de ambos radios medios y el radio mínimo deben ser mayores que *MIN_RADIUS_RADIO=1.8* para que sea un esquina.

Además, los puntos inicial y final del trazo se consideran puntos o vértices de esquina.

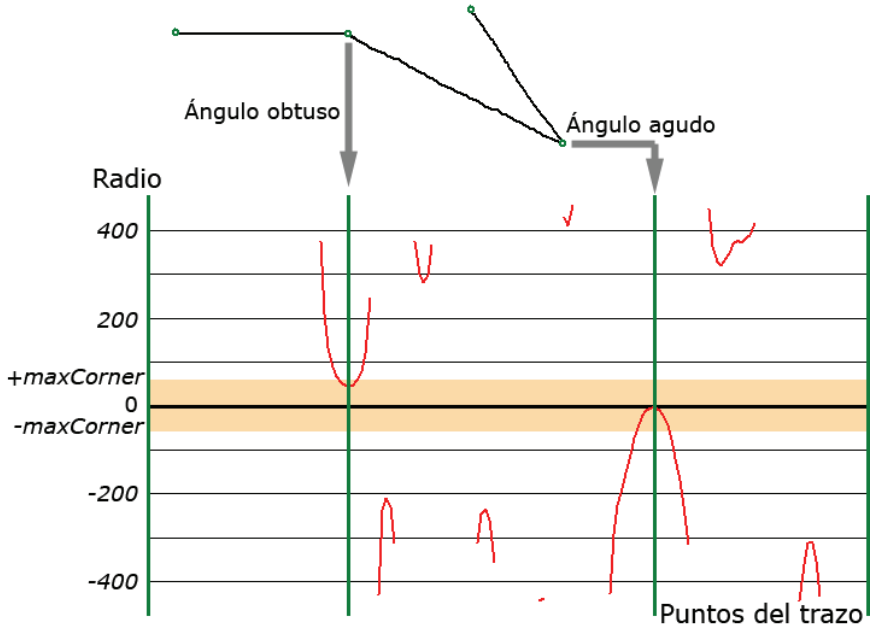


Figura 4.36. Vértices esquina en color verde: punto inicial del trazo, ángulo obtuso, ángulo agudo y punto final del trazo. Valores de radio positivos para giros horarios y negativos para giros antihorarios

4.6.4.4 Aproximación a curvas cúbicas paramétricas

La aproximación de los puntos remuestreados mediante curvas cúbicas paramétricas tiene el objetivo principal de calcular con mayor precisión la función de radio para obtener las curvas del trazo de forma más fiable.

Para evitar errores en vértices esquina (como se ha indicado con anterioridad), se aproximan a curvas paramétricas tramos de puntos remuestreados entre pares de vértices esquina. Estas curvas tienen las restricciones de pasar por los vértices esquina inicial y final de cada tramo, y aproximar los puntos situados entre ambos. En el caso de la curva aproximada supere en algún punto el parámetro de distancia máxima *MAX_DISTANCE=5* en alguno de los puntos remuestreados, el tramo de puntos se divide por la mitad y dos restricciones más para ambas curvas se añaden: deben pasar por el punto central, y debe haber continuidad de primer orden (igual dirección de tangente) en dicho punto para que exista una transición suave.

Como este proceso se puede realizar varias veces, se obtienen cuatro formas distintas de aproximar un tramo de puntos mediante una curva cúbica paramétrica, según las restricciones de tangencia que deban cumplirse: en el punto inicial, en el punto final, en ambos puntos, o en ninguno de ellos; existiendo en todos los casos siempre la restricción de pasar por los puntos inicial y final del tramo.

Cuando un tramo de puntos se divide, la tangente en el punto central (primera derivada) se calcula a partir de la dirección discreta del trazo en dicho punto calculada previamente. Para dejar los suficientes grados de libertad para que la curvas se adapte a los puntos, el vector tangente tendrá la dirección indicada, pero el módulo se dejará libre.

Empleando las expresiones matemáticas de las curvas cúbicas paramétricas y sus derivadas, así como las expresiones de las restricciones, cada una de las cuatro formas distintas de aproximar un tramo de puntos genera uno o dos sistemas de ecuaciones sobredimensionados y que, según el número de restricciones empleadas, pueden tener desde 8 hasta 2 grados de libertad. Estos sistemas sobredimensionados se resuelven por mínimos cuadrados, en concreto, empleando el método de Householder [BF85 y Gol91] que es mucho más estable que el método convencional.

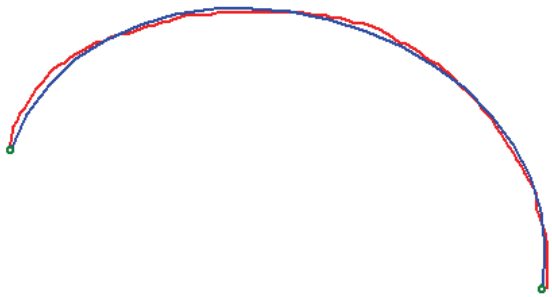


Figura 4.37. Puntos resampleados de un trazo consistente en un único arco (rojo) y curva cúbica paramétrica de aproximación (azul)

4.6.4.5 Cálculo de la función analítica de radio

La tangente en cada punto del trazo se calcula a partir de la derivada de la correspondiente curva paramétrica. La dirección del trazo en el punto es el ángulo calculado a partir de la tangente.

La curvatura en un punto se calcula como la variación de la dirección. La curvatura es el cociente de la derivada de la tangente dividida entre la longitud de la curva cúbica paramétrica correspondiente.

Finalmente, el radio es la inversa de la curvatura.

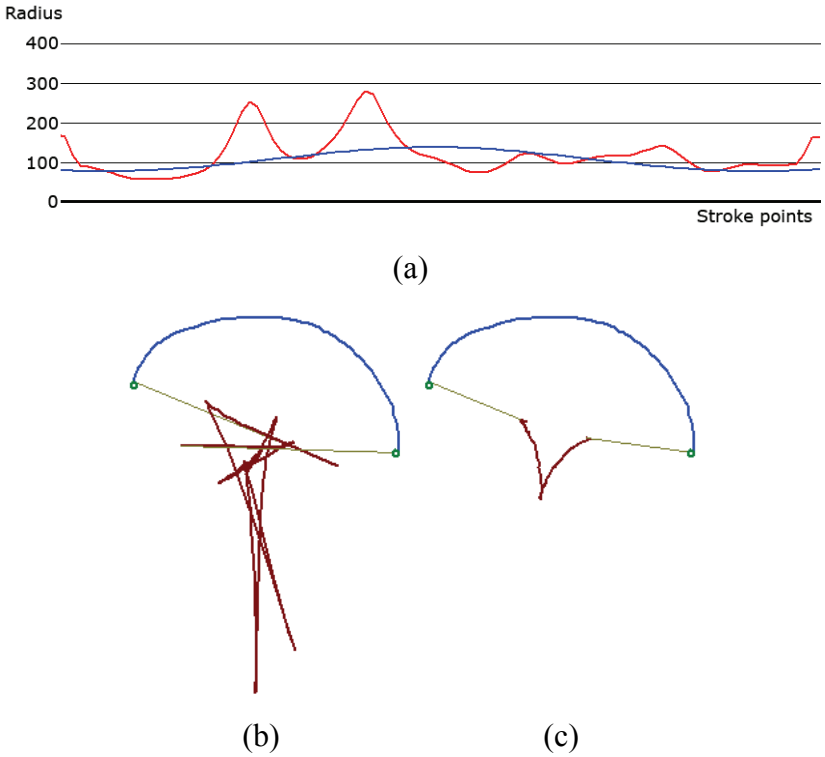


Figura 4.38. Radios y centros de rotación del mismo ejemplo de la figura anterior: a) funciones discreta (rojo) y analítica (azul) del radio; b) centros de rotación (marrón) para cada punto del trazo calculados por medio de la dirección y el radio discretos; c) centros calculados con la dirección y el radio analíticos

4.6.4.6 Delimitando arcos a partir de la función del radio

Un tramo de puntos del trazo puede ser un curva si los valores de sus radios tienen el mismo signo y están por debajo del umbral *MAX_CURVE RADIUS=320*. A efectos de cálculo, las curvas se consideran arcos, y las características calculadas son las siguientes:

- Radio: es la mediana del radio de todos los puntos del tramo del trazo considerado.
- Longitud: número de puntos del tramo considerado multiplicado por la distancia de interespaciado.
- Ángulo: es igual a Longitud / Radio.
- Distancia de cuerda a arco (ver figura): Radio · [1 – cos (Ángulo / 2)]

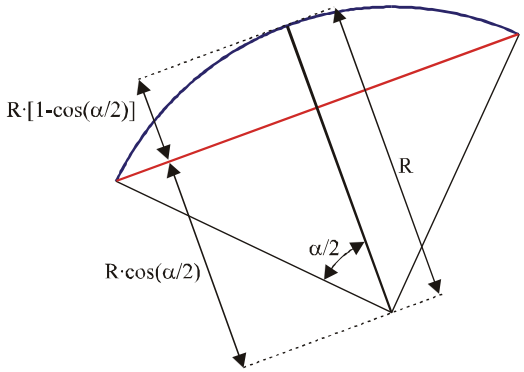


Figura 4.39. Distancia de cuerda (rojo) a arco (azul)

Los tramos candidatos a curvas, serán curvas si su distancia de cuerda a arco es mayor que el parámetro *MIN_DIST_CA*=7.2. Los tramos que no son curvas, son considerados como rectas.

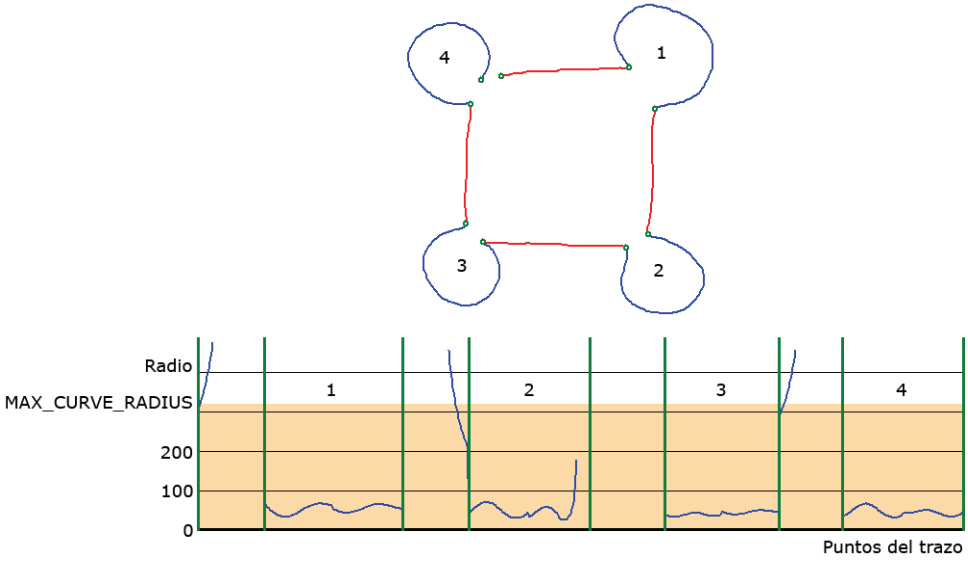


Figura 4.40. Forma esbozada con líneas rectas y un arco separados mediante vértices esquina (rectas en color rojo y arcos en azul sobre la forma esbozada), y sus valores de radio (en la gráfica con el valor del radio de algunos puntos del arco sobre el umbral establecido)

4.6.4.7 Calculo de los vértices tangentes

Tras la obtención de los vértices esquina, arcos y rectas, se procede a localizar los vértices tangentes en las transiciones (sin vértices esquina) entre recta y arco, o entre dos arcos.

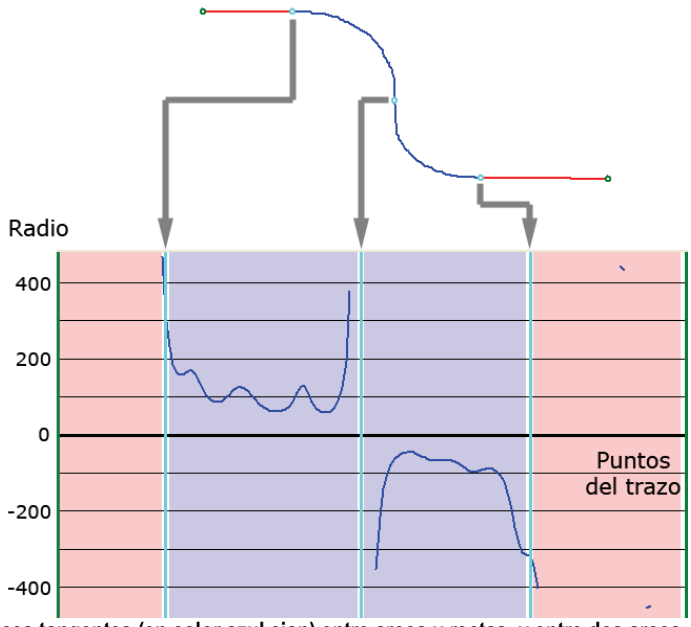


Figura 4.41. Vértices tangentes (en color azul cian) entre arcos y rectas, y entre dos arcos. El color de fondo de la función de radio es azul para los arcos y rojo para las rectas

Los vértices tangentes (ver Figura 4.41) aparecen siempre en los pasos de rectas a curvas (y viceversa) y en los puntos de inflexión (cambio del signo en la función del radio de curvatura). Pero antes se realiza una pequeña consideración: las rectas con una longitud menor que un umbral *MIN_LINE_LENGTH=45* se convierten a curvas si están al lado de una curva sin que exista un vértice esquina entre la recta y la curva. En el caso de que la pequeña recta se encuentre entre dos arcos, se añade a los arcos (mitad a cada uno) quedando el vértice tangente en medio.

4.6.5 Mejoras a realizar al método TCVD

El principal inconveniente del TCVD tiene que ver con el tamaño de las rectas y curvas que forman el trazo. Individualmente, cada recta o curva debe tener un tamaño suficiente, pero por otra parte, las curvas de radio grande pueden ser confundidas con rectas, y para evitar este inconveniente el trazo es escalado para que no supere un tamaño máximo, por lo que si en un trazo grande hay rectas o curvas pequeñas, todavía se van a reducir más. En resumen, los trazos con primitivas elementales (rectas y curvas) pequeñas, o muy complicados (con muchas primitivas) son difíciles de tratar, sin embargo, no es natural dibujar de un solo trazo formas complicadas.

El inconveniente anterior, nos puede llevar a pensar que el TCVD se podría ampliar para tratar “multistrokes”, pero esta tarea corresponde a los agentes, TCVD sólo se encarga de segmentar trazos básicos.

Una necesidad que sí es básica es realizar las ampliaciones necesarias para tratar formas cerradas, detectando cuando el punto inicial y el punto final del trazo están muy cercanos, y considerar que existe continuidad entre el final y el principio.

Por último, existe una recomendación básica para que TCVD consiga los mejores resultados: al dibujar el trazo hay que pensar en lo que se está haciendo. Aunque parezca evidente, normalmente no lo hacemos, y somos capaces de reconocer lo que hemos dibujado gracias a los conocimientos que hemos adquirido con el tiempo. Debemos tener en cuenta los siguientes aspectos:

- Parar en los vértices esquina, para cambiar la dirección y evitar confusiones con curvas de radio pequeño.
- No parar en los vértices tangente, para evitar cambiar la dirección de forma brusca.
- Dibujar rectas donde haya rectas y curvas donde haya curvas, ya que si al lado de una curva dibujamos una recta ligeramente curvada, lo más fácil es que TCVD lo agrupe todo en una misma curva que se va abriendo.

Capítulo 5:
LOS AGENTES DEL SISTEMA

5. LOS AGENTES DEL SISTEMA

En este capítulo se describen los agentes contenidos en cada uno de los niveles del sistema multi-agente diseñado ya descrito en el capítulo anterior. En primer lugar se presentan los agentes del nivel inferior, denominados Agentes Básicos. Luego se continúa con los agentes del nivel intermedio, también llamados Agentes Primitivos, y finalmente se presentan los agentes del nivel superior, a los que se ha denominado Agentes Combinados.

5.1 Los Agentes Básicos

Los Agentes Básicos están contenidos en el nivel inferior del sistema multi-agente, y son los encargados de realizar las tareas más básicas asociadas al soporte del interfaz y al proceso de reconocimiento. Dentro del núcleo central, se han establecido cinco agentes básicos que realizan las tareas de interfaz de usuario, pre-procesado de la entrada digitalizada, extracción y cálculo de características, gestión y distribución de las tareas entre los distintos agentes, y toma de decisiones sobre el reconocimiento del gesto introducido. Adicionalmente, cada módulo de la plataforma dispondrá también de un agente básico encargado de gestionar las comunicaciones entre los agentes combinados de cada módulo y el núcleo central.

Para nombrar a estos agentes, se utilizan dos letras, la primera de ellas está relacionada con la acción que realiza, y la segunda corresponde a la "A" de Agente. Así pues, para el agente encargado del interfaz, se utilizan las siglas IA que corresponden a *Interface Agent*. Para nombrar al agente encargado del pre-procesado se utilizan las siglas PA de *Pre-processing Agent*. Para nombrar al agente que extrae y calcula las características o *cues* básicas se utilizan las siglas FA de *Feature Agent*. En el caso del agente encargado de tomar la decisión final sobre el reconocimiento del gesto introducido se han utilizado las siglas INA de *INtelligent Agent*, para no confundirlo con el agente interfaz. De igual manera, dado que tanto el núcleo central como los módulos de la plataforma poseen un agente broker encargado de gestionar el núcleo o módulo, se ha optado por llamar BACC (Broker Agent Central Core) al agente broker del núcleo central, y BAXM a los distintos brokers de los módulos, siendo X la inicial de cada módulo. En el caso de la presente tesis, dado que se han implementado 4 módulos, los nombres asignados han sido: BARM para el agente broker del módulo de restricciones; BAAM para el agente broker del módulo de acotación; BASM para el agente broker del módulo de modelado; y BATM para el agente broker del módulo artístico. A continuación se explica cada uno de los Agentes Básicos.

5.1.1 Agente IA

El agente IA (*Interface Agent)* es el encargado del interfaz de la aplicación. Su tarea consiste principalmente en esperar y redibujar la entrada del usuario en pantalla y enviar los puntos capturados al agente broker del núcleo central BACC para que éste lo distribuya a quién corresponda. El agente IA queda siempre a la espera de más entradas por parte del usuario y de resultados por parte del reconocedor, que le son notificados a través del agente inteligente INA. Fundamentalmente sus tareas son de gestión de la información gráfica.

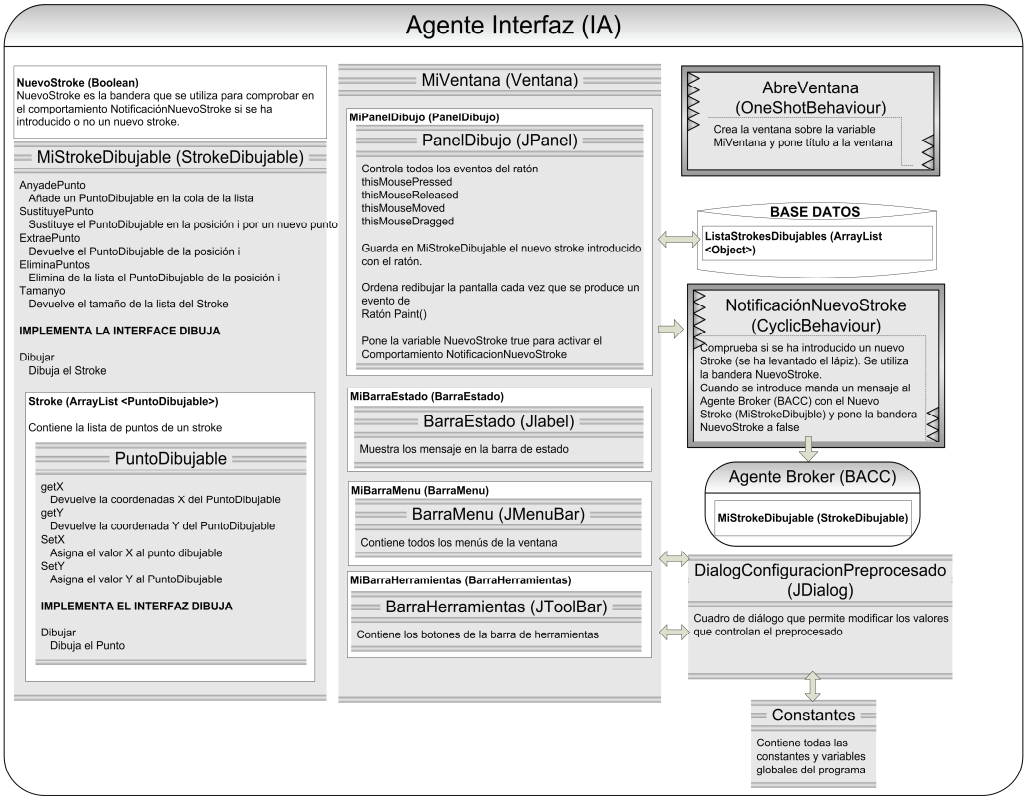


Figura 5.1. Diagrama de funcionamiento del IA

5.1.2 Agente PA

El agente PA (*Pre-processing Agent)* es el encargado de "limpiar" la entrada del usuario. Su tarea consiste en pre-procesar los puntos capturados por el interfaz y proporcionados por el agente BACC, y enviarle posteriormente los puntos pre-procesados para que continúe con el proceso de reconocimiento.

El proceso que se sigue es el siguiente: cuando un usuario dibuja un trazo, el agente interfaz IA lo muestra y envía los puntos digitalizados a una estructura de datos y al agente broker BACC. Este último agente recibe la notificación y envía los puntos del trazado al agente de pre-procesado PA. De forma ideal, estos puntos deberían estar distribuidos de manera uniforme. Sin embargo, cuanto más rápido se dibuja el trazo, menos puntos se digitalizan, lo que provoca una variación en la concentración de puntos (Figura 5.2.a). Para resolver este problema y otros derivados del manejo del dispositivo, el agente PA detecta y elimina los puntos aislados, estimando la distancia entre cada punto y sus vecinos. Después, este agente pasa una máscara de suavizado por los puntos de cada trazo, disimulando así el efecto causado por posibles temblores al dibujar (Figura 5.2b). Y finalmente calcula la ecuación lineal entre cada par de puntos digitalizados consecutivamente para rellenar los huecos existentes y conseguir una distribución uniforme lista para el reconocimiento (Figura 5.2c).

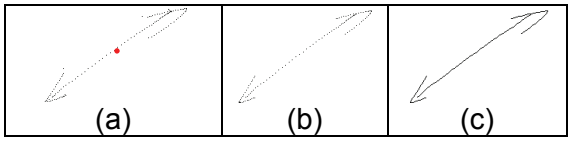


Figura 5.2. a) Esbozo de cota diametral; b) corrección del esbozo y c) esbozo uniforme

El suavizado del trazo se lleva a cabo sumando las coordenadas de los dos puntos anteriores y los dos posteriores al actual que se está filtrando, más la suma de dos veces las coordenadas del punto en cuestión. El resultado de esta suma se divide por seis, que es el número de puntos del trazo involucrados en la operación. La coordenada *X* del punto filtrado toma como valor el resultado de esta operación, siendo *n* los puntos del trazo, e *i* el punto a filtrar (5-1). El proceso es el mismo para la coordenada *Y* del punto filtrado.

$$X_i = \frac{\sum_{n=i-2}^{i+2} X_n + 2X_i}{6} \qquad (5\text{-}1)$$

En la Figura 5.3 se muestra el diagrama de funcionamiento del PA.

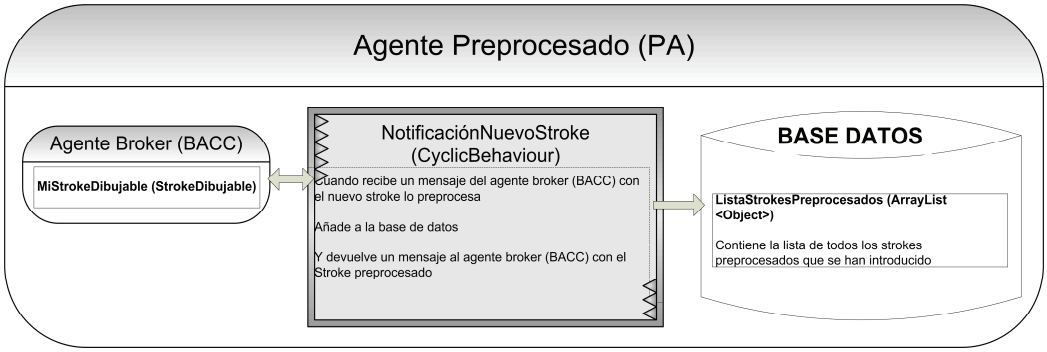


Figura 5.3. Diagrama de funcionamiento del PA

5.1.3 Agente FA

El agente FA (*Feature Agent)* es el encargado de realizar la segmentación del trazo y de extraer y calcular todas las características o *features* del trazo, también llamadas *cues* básicas. Esta información debe ser relevante e invariante a la posición, escala y rotación. Toda la información relativa al cálculo de dichas *cues* está detallada en el capítulo anterior de Materiales y métodos. En la Figura 5.4 se muestra el diagrama de funcionamiento del FA.

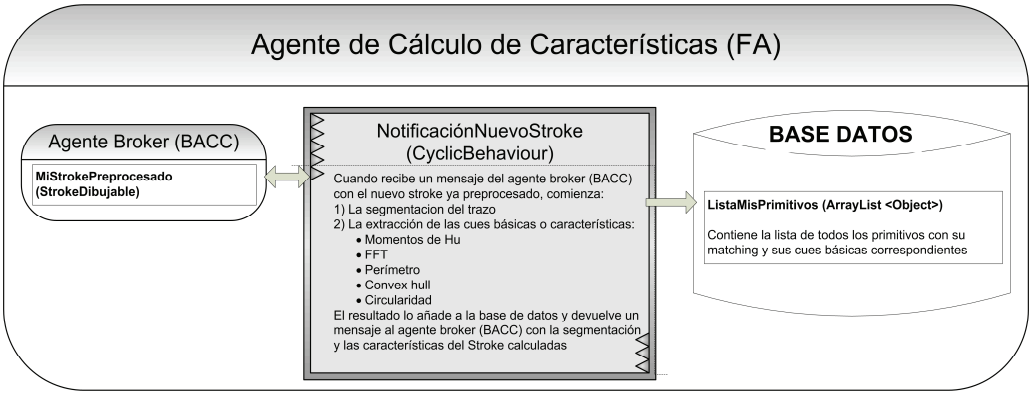


Figura 5.4. Diagrama de funcionamiento del FA

Para la extracción de las características se han utilizado librerías estándar de tratamiento de imagen. La mayoría de estas librerías están programadas en C++. De hecho, las librerías que se han utilizado en esta tesis son las IPL 2.5 de Intel y las OpenCV de código abierto en su versión estable 1.0. Así pues, aunque la plataforma donde se ha implementado la arquitectura multi-agente está programada únicamente mediante el lenguaje Java, parte del código del agente FA está programado en C++, contenido en unas librerías dinámicas o *"dlls" (dynamic link libraries)* que hacen referencia a dichas funciones. Todo el interfaz utilizado para permitir la llamada a rutinas programadas en C++ desde Java se ha implementado a través del *wrapper* SWIG, explicado también en el capítulo anterior de Materiales y métodos.

A continuación se muestra la implementación del interfaz para la llamada desde el agente FA programado en Java a las librerías de visión programadas en C++ para el cálculo de las características:

```
public class FA extends Agent //FA: Feature Agent
{
          //Declaración de datos

          static {
                    System.loadLibrary("Recog");
          }

          public void setup(){
                              …
          }
          public class ComportamientoCaracteristicasNuevoStroke extends CyclicBehaviour
          {
              …
              // Llamada a las funciones de la DLL, enviándole los vectores de C creados
              MisCaracteristicas_C = Recog.ExtraeCaracteristicas(PuntosOriginales_C,PuntosPreprocesados_C,
                                                                        MisParametros,true);
              Contorno_C = Recog.ExtraeContornoStrokes(PuntosPreprocesados_C, MisParametros,true);
              …
          }
}
```

Donde la librería Recog.java contiene el siguiente código, posibilitando la llamada a las funciones de C++ de forma transparente al usuario:

```
/* ----------------------------------------------------------------------------
 * This file was automatically generated by SWIG (http://www.swig.org).
 * Version 1.3.36
```

```
 *
 * Do not make changes to this file unless you know what you are doing--modify
 * the SWIG interface file instead.
 * ----------------------------------------------------------------------------- */
public class Recog implements RecogConstants {
  public static VectorPuntos_C PreprocesadoPuntos(VectorPuntos_C PuntosOriginales, Parametros_C
Parametros) {
    return new VectorPuntos_C(RecogJNI.PreprocesadoPuntos(VectorPuntos_C.getCPtr(PuntosOriginales),
      PuntosOriginales, Parametros_C.getCPtr(Parametros), Parametros), true);
  }

  public static VectorPuntos_C ExtraeContornoStrokes(VectorPuntos_C PuntosPreProc, Parametros_C
Parametros,
    boolean debugmode) {
    return new VectorPuntos_C(RecogJNI.ExtraeContornoStrokes(VectorPuntos_C.getCPtr(PuntosPreProc),
PuntosPreProc,
      Parametros_C.getCPtr(Parametros), Parametros, debugmode), true);
  }

  public static Caracteristicas_C ExtraeCaracteristicas(VectorPuntos_C PuntosCapturados, VectorPuntos_C
    PuntosPreProc, Parametros_C Parametros, boolean debugmode) {
    return new Caracteristicas_C(RecogJNI.ExtraeCaracteristicas(VectorPuntos_C.getCPtr(PuntosCapturados),
      PuntosCapturados, VectorPuntos_C.getCPtr(PuntosPreProc), PuntosPreProc,
Parametros_C.getCPtr(Parametros),
      Parametros, debugmode), true);
  }
} //end of class Recog.java
```

Precisamente es el entorno SWIG el que posibilita la generación automática del código necesario para el interfaz entre la librería nativa escrita en C++ y el entorno de programación en Java. El archivo RecogJNI.java ofrece una muestra de dicho código:

```
/* ----------------------------------------------------------------------------
 * This file was automatically generated by SWIG (http://www.swig.org).
 * Version 1.3.36
 *
 * Do not make changes to this file unless you know what you are doing--modify
 * the SWIG interface file instead.
 * ----------------------------------------------------------------------------- */

class RecogJNI {
  public final static native long new_VectorPuntos_C__SWIG_0();
  public final static native long new_VectorPuntos_C__SWIG_1(long jarg1);
  public final static native long VectorPuntos_C_size(long jarg1, VectorPuntos_C jarg1_);
  public final static native long VectorPuntos_C_capacity(long jarg1, VectorPuntos_C jarg1_);
  public final static native void VectorPuntos_C_reserve(long jarg1, VectorPuntos_C jarg1_, long jarg2);
  public final static native boolean VectorPuntos_C_isEmpty(long jarg1, VectorPuntos_C jarg1_);
  public final static native void VectorPuntos_C_clear(long jarg1, VectorPuntos_C jarg1_);
  public final static native void VectorPuntos_C_add(long jarg1, VectorPuntos_C jarg1_, long jarg2,
PUNTO_C
    jarg2_);
  public final static native long VectorPuntos_C_get(long jarg1, VectorPuntos_C jarg1_, int jarg2);
  public final static native void VectorPuntos_C_set(long jarg1, VectorPuntos_C jarg1_, int jarg2, long
jarg3,
    PUNTO_C jarg3_);
  public final static native void delete_VectorPuntos_C(long jarg1);

          …

  public final static native void Caracteristicas_C_Trazos_set(long jarg1, Caracteristicas_C jarg1_, long
jarg2,
    VectorEntidad_C jarg2_);
  public final static native long Caracteristicas_C_Trazos_get(long jarg1, Caracteristicas_C jarg1_);
  public final static native void Caracteristicas_C_Perimetro_set(long jarg1, Caracteristicas_C jarg1_,
double
    jarg2);
  public final static native double Caracteristicas_C_Perimetro_get(long jarg1, Caracteristicas_C jarg1_);
  public final static native void Caracteristicas_C_PerimetroConvexHull_set(long jarg1, Caracteristicas_C
jarg1_,
    double jarg2);
  public final static native double Caracteristicas_C_PerimetroConvexHull_get(long jarg1,
Caracteristicas_C
    jarg1_);
```

```
  public final static native void Caracteristicas_C_PerimetroContornoE_set(long jarg1, Caracteristicas_C
jarg1_,
    double jarg2);
  public final static native double Caracteristicas_C_PerimetroContornoE_get(long jarg1, Caracteristicas_C
    jarg1_);
  public final static native void Caracteristicas_C_CircularidadE_set(long jarg1, Caracteristicas_C
jarg1_, double
    jarg2);
  public final static native double Caracteristicas_C_CircularidadE_get(long jarg1, Caracteristicas_C
jarg1_);
  public final static native void Caracteristicas_C_HumomentsE_set(long jarg1, Caracteristicas_C jarg1_,
long
    jarg2, VectorDouble_C jarg2_);
  public final static native long Caracteristicas_C_HumomentsE_get(long jarg1, Caracteristicas_C jarg1_);
  public final static native void Caracteristicas_C_Fourier_set(long jarg1, Caracteristicas_C jarg1_, long
jarg2,
    VectorDouble_C jarg2_);
  public final static native long Caracteristicas_C_Fourier_get(long jarg1, Caracteristicas_C jarg1_);
  public final static native void Caracteristicas_C_Centroide_set(long jarg1, Caracteristicas_C jarg1_,
long
    jarg2, PUNTO_C jarg2_);
  public final static native long Caracteristicas_C_Centroide_get(long jarg1, Caracteristicas_C jarg1_);
  public final static native long new_Caracteristicas_C();
  public final static native void delete_Caracteristicas_C(long jarg1);
} //Fin de RecogJNI.java
```

5.1.4 El agente BACC

El agente BACC (*Broker Agent Central Core)* es el encargado tanto de distribuir las tareas entre todos los agentes del sistema, como de controlar el flujo de datos en el mismo. Es un agente que concentra toda la información y la canaliza a donde corresponda, esto es, recoge notificaciones de todos los agentes del sistema, ya pertenezcan éstos al nivel básico, al primitivo o al superior, y redirecciona el flujo de datos al agente correspondiente en función del origen del mensaje o notificación. En definitiva, el BACC se encarga de la organización y gestión del sistema y de sus datos, permitiendo que los agentes estén coordinados en todo momento y de que no se produzcan pérdidas de sincronismo entre los mismos. A su vez, toma la decisión en base a la información enviada por los agentes primitivos de si el trazo introducido se corresponde con algún primitivo o se trata más bien de geometría. Esta decisión, así como una explicación más detallada del funcionamiento del sistema se puede encontrar en el apartado 4.4 del capítulo anterior. En la Figura 5.5 se muestra de forma resumida el diagrama de funcionamiento del BA.

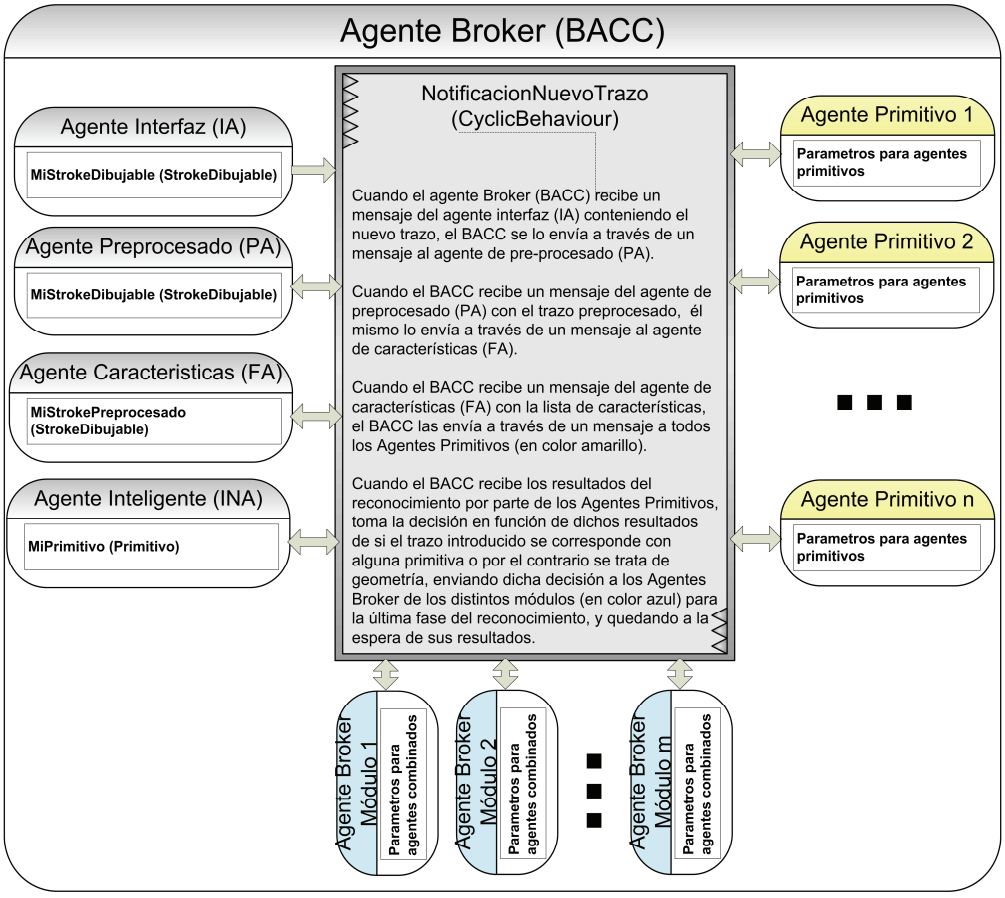


Figura 5.5. Diagrama de funcionamiento del BACC

5.2 Los Agentes Primitivos

Los Agentes Primitivos se han establecido en nueve, y son los encargados de reconocer los trazos simples pertenecientes al alfabeto establecido, ya definido en el capítulo 4 de Materiales y métodos. Estos trazos simples son equivalentes a los fonemas que compondrán todos los gestos o símbolos del interfaz caligráfico implementado en esta tesis. Se ha implementado un agente por cada símbolo primitivo y el nombre de estos agentes se ha escogido atendiendo a las siguientes reglas: los dos primeros caracteres para el tipo de fonema a reconocer (en inglés), y el tercer carácter corresponde a la letra A de agente. De esta manera, el agente primitivo encargado de encontrar un punto se llamará POA (PO de "*point*" y A de "*agent*"). En la Figura 5.6 se muestran todos los Agentes Primitivos implementados.

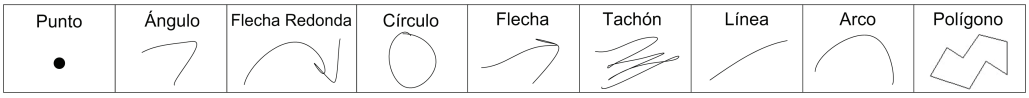


Figura 5.6. Agentes primitivos implementados

El comportamiento de los agentes primitivos se lleva a cabo de la manera descrita a continuación. El agente broker del núcleo central (BACC) notifica a todos los agentes primitivos de forma simultánea que deben comenzar el reconocimiento, enviándoles para ello los datos extraídos por el agente de características FA. Cada uno de los agentes primitivos empleará únicamente la información que le resulte significativa con el fin de encontrar pistas relevantes que le permitan reconocer el símbolo primitivo del cual se encarga. Cada agente primitivo evalúa el trazo introducido y cuantifica su información a partir de dos procesos claramente diferenciados: 1) reconocimiento basado en la geometría del trazo, basándose para ello en la segmentación realizada previamente, y 2) cálculo de la función de maximización de algunas características extraídas por el agente FA. Una vez los agentes primitivos han terminado el proceso de reconocimiento, éstos devuelven el resultado de ambos procesos al agente BACC, el cual tomará la decisión de si el trazo introducido se corresponde con algún símbolo primitivo o por el contrario se trata de geometría.

En los apartados posteriores se detalla el funcionamiento de cada uno de los agentes primitivos, concretamente del proceso de reconocimiento basado en la geometría del trazo, empleando para ello la segmentación realizada previamente. Dicha segmentación permite dividir cada trazo en varios tramos (o *stretches*), usando para ello los vértices detectados.

5.2.1 El agente POA. Punto

El agente POA (POint Agent) realiza la comprobación de que el trazo introducido sea un punto. Es el agente más sencillo de todos, ya que una vez recibido el trazo y sus características, se comprueba que únicamente exista un *stretch* (o tramo) y que éste sea del tipo punto, en cuyo caso dará un reconocimiento positivo.

5.2.2 El agente CIA. Círculo

El agente CIA (CIrcle Agent) es el encargado de determinar si el trazo introducido corresponde a un círculo. Este agente recibe el trazo y las características extraídas del mismo, y comprueba si contiene un único *stretch* del tipo círculo, en cuyo caso dará un reconocimiento positivo.

5.2.3 El agente LIA. Línea

El agente LIA (LIne Agent) realiza la comprobación de si el trazo introducido es una línea recta. Una vez recibido el trazo y las características extraídas del mismo, comprueba si únicamente contiene un *stretch* del tipo línea, en cuyo caso dará un reconocimiento positivo.

5.2.4 El agente ARA. Arco

El agente ARA (ARc Agent) realiza la comprobación de que el trazo introducido sea un arco. Tras recibir el trazo y las características extraídas del mismo, este agente comprueba si existe un único *stretch* contenido en el trazo, y si éste es un arco, en cuyo caso dará un reconocimiento positivo.

Como se puede apreciar, los reconocimientos positivos de los símbolos primitivos de punto, círculo, línea y arco son incompatibles con reconocimientos positivos de otros símbolos

primitivos. Esto se debe sencillamente a que puede existir únicamente una entidad contenida en el trazo, por lo que no hay posibilidad de reconocer otros fonemas, ya que el resto está compuesto por más de una entidad.

5.2.5 El agente ANA. Ángulo

El agente ANA (ANgle Agent) realiza la comprobación de que el trazo introducido sea un ángulo, esto es, similar a un signo de "menor o mayor que". Cuando este agente recibe el trazo y las características extraídas del mismo, lleva a cabo las siguientes comprobaciones:

1) Que existan únicamente dos *stretches* y que sean del tipo línea.
2) Que la diferencia entre las longitudes de ambas líneas sea inferior a la longitud de la línea menor.
3) Que el ángulo comprendido entre ambas líneas esté dentro de un determinado rango.

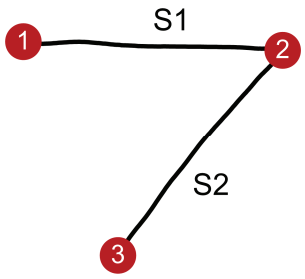


Figura 5.7. Símbolo primitivo de ángulo

5.2.6 El agente AWA. Flecha

El agente AWA (ArroW Agent) realiza la comprobación de que el trazo introducido sea una flecha. Una vez el agente recibe el trazo y las características extraídas del mismo, realiza las siguientes comprobaciones:

1) Que existan cuatro *stretches*.
2) Que todos los *stretches* sean del tipo línea.
3) Que el ángulo comprendido entre las líneas 3-4 y 4-5 esté dentro de un determinado rango.
4) Que el vértice 2 se encuentre a una distancia del vértice 4 inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
5) Que el ángulo de la primera línea 1-2 esté comprendido entre el rango abarcado por las líneas 3-4 y 5-4.
6) Que la longitud de la línea 1-2 sea al menos el doble de la longitud de la mayor de las líneas 3-4 y 4-5.

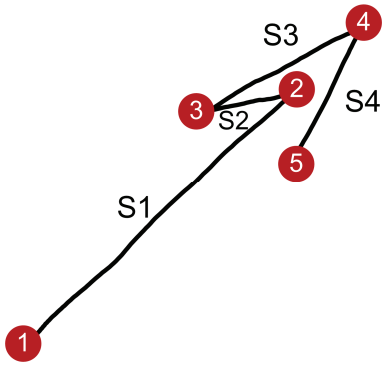


Figura 5.8. Símbolo primitivo de flecha

5.2.7 El agente ROA. Flecha redonda

El agente ROA (ROund arrow Agent) detecta si el trazo introducido se corresponde con una flecha redonda a modo de gesto de revolución. A partir del trazo y las características extraídas se realizan las siguientes comprobaciones:

1) Que existan cuatro *stretches*.
2) Que el primer *stretch* sea del tipo arco y los tres restantes del tipo línea.
3) Que el ángulo comprendido entre las líneas 3-4 y 4-5 esté dentro de un determinado rango.
4) Que el vértice 2 se halle a una distancia del vértice 4 inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
5) Que el ángulo formado por la línea ficticia 2-4 esté comprendido entre el rango abarcado por las líneas 3-4 y 5-4.

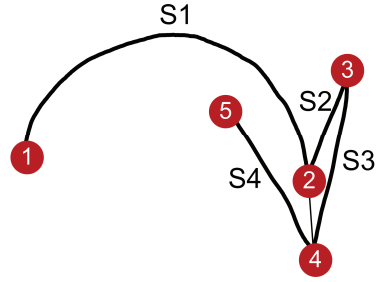


Figura 5.9. Símbolo primitivo de flecha redonda

5.2.8 El agente CRA. Tachón

El agente CRA (CRoss-out Agent) realiza la comprobación de que el trazo introducido sea un tachón, a modo de gesto de borrado. Este agente recibe el trazo y las características extraídas del mismo, y comprueba lo siguiente:

1) Que hayan al menos dos *stretches* (sin importar del tipo que sean).

2) Que el ratio entre el perímetro del convex-hull (contorno azul de la Figura 5.10) y el perímetro del trazo (línea de color negro) esté por debajo de un determinado valor.

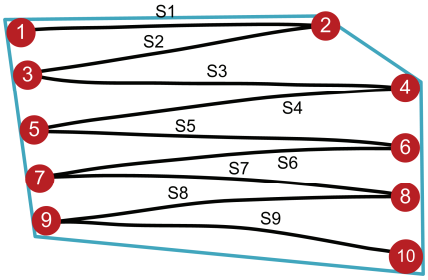


Figura 5.10. Símbolo primitivo de tachón

La característica basada en el ratio entre el perímetro del convex-hull (ver apartado 4.5.3) y el perímetro del trazo ayuda enormemente a la clasificación del gesto de tachón, ya que el perímetro del tachón contiene un elevado número de puntos en relación con el perímetro del convex-hull, a diferencia de lo que ocurre con otros símbolos primitivos. Así pues, este parámetro inequívocamente caracteriza este fonema.

5.2.9 El agente GPA. Polígono general

El agente GPA (General Polygon Agent) realiza la comprobación de que el trazo introducido sea un polígono general cerrado, formado únicamente por líneas rectas.

En este caso las comprobaciones que realiza el agente son:

1) Existen al menos tres *stretches,* y todos son del tipo línea.

2) Los puntos inicial y final del trazo deben cumplir una tolerancia de proximidad, asegurando así que el polígono sea cerrado.

3) No deben existir cruces (contemplando la posibilidad de que al cerrar el polígono se produzca una intersección cercana al cierre).

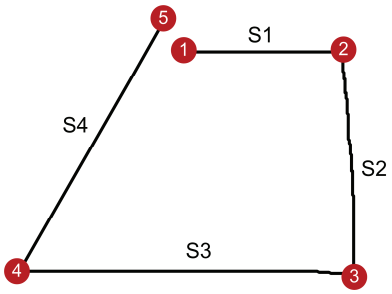


Figura 5.11. Símbolo primitivo de polígono

Estos resultados positivos que obtienen los agentes primitivos son enviados al agente broker del núcleo central (BACC), junto con el valor de la función de maximización de las características extraídas por el agente FA (ver apartado 4.5). Este agente BACC tomará la decisión final basándose en el siguiente criterio:

1) Si existe un agente primitivo cuyo *matching* es positivo (coincidencia), y además su valor de la función de maximización es el mayor entre los valores ofrecidos por todos los agentes primitivos, el agente BACC escoge el gesto reconocido por este primitivo como gesto detectado.
2) En el caso de que no se produzca el caso primero, si existe un agente primitivo cuyo *matching* es positivo, y su valor de la función de maximización es el segundo mayor entre los valores ofrecidos por todos los agentes primitivos, el agente BACC también escogerá al gesto reconocido por dicho primitivo como gesto detectado.
3) Si no se da ninguno de los casos anteriores, el trazo es clasificado como geometría.

Si lo que se detecta es el gesto de tachón o una geometría, el agente BACC envía la información directamente al agente inteligente INA, el cual realizará el borrado si se trata del gesto tachón, o la inserción del trazo como geometría en el segundo caso. Si por el contrario se detecta un primitivo, el agente BACC envía la información del primitivo detectado a los agentes broker de los distintos módulos de la plataforma. Estos agentes broker BARM, BASM, BAAM y BATM enviarán a su vez la información del primitivo detectado a cada uno de sus agentes combinados existentes en cada módulo, los cuales llevarán a cabo la segunda etapa del reconocimiento, de ahí que se diga que estos agentes combinados se encuentran en el nivel superior (o tercer nivel) de la plataforma.

5.3 Los Agentes Combinados

En el tercer nivel se lleva a cabo el reconocimiento de los símbolos del alfabeto implementado en cada uno de los módulos. Al igual que en el nivel intermedio, existe al menos un agente combinado por cada símbolo que se desea reconocer. En la Figura 5.12 se muestran los distintos agentes combinados implementados, así como los módulos donde se encuentran incluidos.

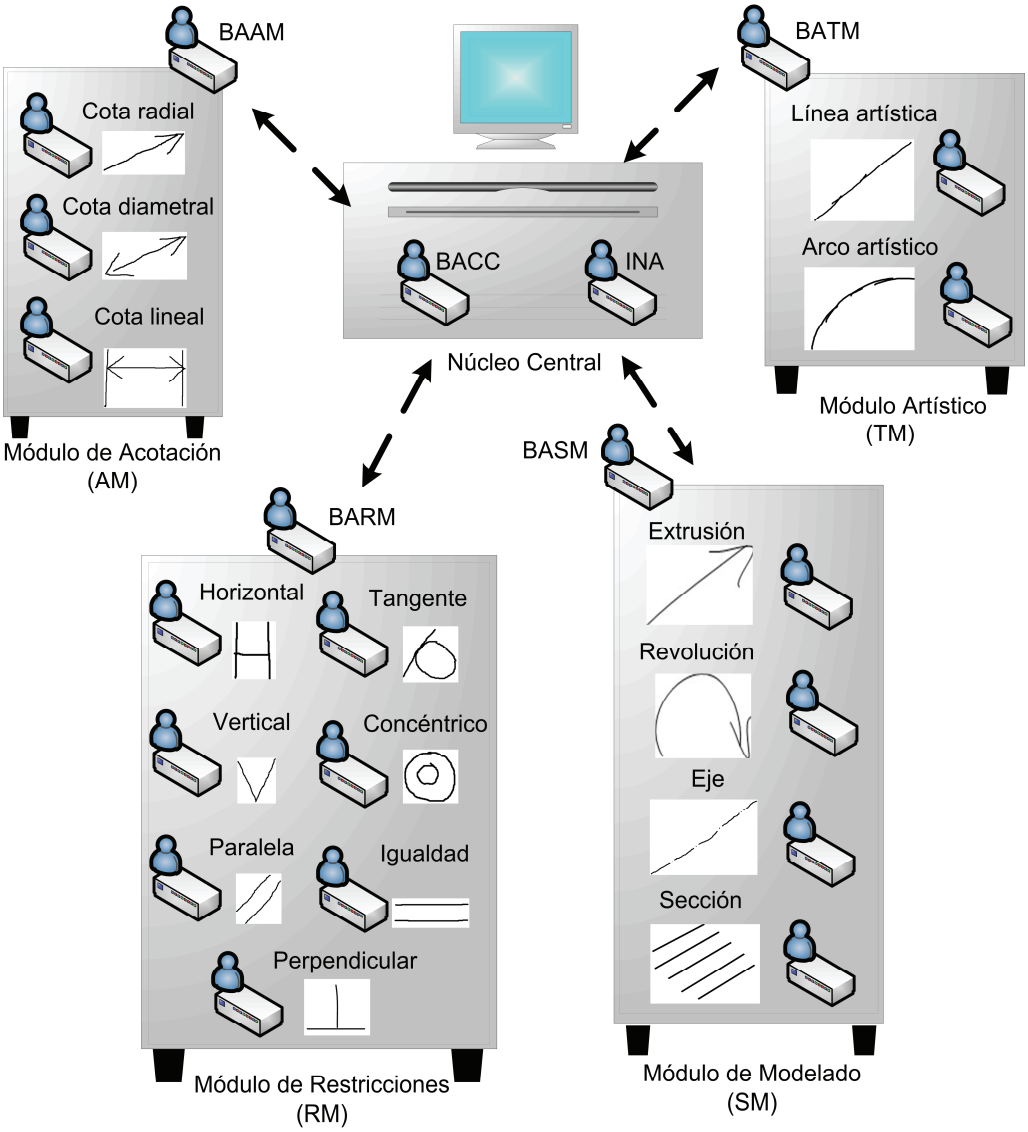


Figura 5.12. Agentes combinados implementados y sus módulos

Una vez el agente BACC envía la información del primitivo detectado a los agentes broker de los distintos módulos, éstos reenvían dicha información a los agentes combinados de su módulo. Los agentes combinados disponen de una lista o *storehouse* donde irán agregando los primitivos que se vayan reconociendo como si de fonemas se tratara hasta que puedan conformar un símbolo completo, es decir, hasta que alcancen un significado semántico pleno.

Así pues el sistema comprueba las siguientes situaciones:

a) Si el fonema es un símbolo formado por un único trazo y que tiene significado pleno por sí mismo, esto es, no pertenece a ningún símbolo más complejo, se asigna y el proceso de reconocimiento actual finaliza.

b) Si es factible de pertenecer a otros símbolos formados por varios trazos, entonces se añade a la lista y el sistema debe esperar a que se introduzca el siguiente trazo:

1) Si el siguiente trazo es factible de formar conjuntamente un símbolo combinado, entonces se añade y el símbolo actual finaliza, actualizando el estado del agente combinado a "aceptado".

2) Si el siguiente trazo es factible de pertenecer, pero no existe suficiente significado semántico como para formar de forma conjunta un símbolo combinado, entonces se añade y se actualiza al estado de "en proceso".

3) Si el siguiente trazo no es factible de pertenecer, el agente combinado correspondiente se pone en el estado "rechazado".

Nótese que cada trazo puede pertenecer a más de un símbolo formado por múltiples trazos, y que el mismo símbolo se puede dibujar de distintas maneras. Así pues, todos los posibles candidatos se buscan a la vez en paralelo, hasta que alguno de ellos alcanza un significado semántico pleno (estado de "aceptado"). En el apartado 4.4 se explica con mayor detalle este proceso.

Para nombrar a los agentes combinados, se han utilizado cuatro letras, las tres primeras están relacionadas con el símbolo combinado que detectan, y la cuarta corresponde a la "A" de Agente. Así pues, para el agente combinado encargado de detectar el símbolo de horizontalidad, se utilizan las siglas HORA (*HORizontal Agent).*

Estos agentes combinados precisan a su vez de unos márgenes de tolerancia que permitan asegurar si un determinado símbolo primitivo cumple las condiciones necesarias para poder ser considerado parte del símbolo combinado. Así pues se han considerado tolerancias de posición, de dimensión, de inclinación, de perpendicularidad y de simetría. Dentro de las tolerancias de inclinación también se han tenido en cuenta las tolerancias de horizontalidad y de verticalidad.

A continuación se definen los distintos agentes combinados, así como las reglas que guían el reconocimiento en cada uno de ellos.

5.3.1 El agente HORA. Horizontalidad

El agente HORA (*HORizontal Agent*) comprueba si los trazos introducidos conforman el símbolo de horizontalidad. Se trata de un agente cuyo reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. Para el agente de horizontalidad se ha considerado únicamente el caso en el que el símbolo introducido conste de tres trazos (ver Figura 5.13).

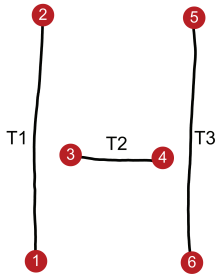


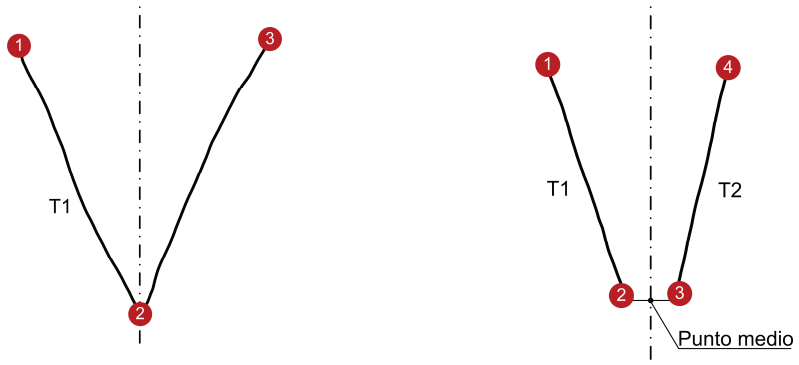
Figura 5.13. Caso considerado para el símbolo de horizontalidad

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones por primitivo:
  - Dos trazos (T1 y T3) deben cumplir la tolerancia de verticalidad.
  - Un trazo (T2) debe cumplir la tolerancia de horizontalidad.
- Condiciones entre primitivos:
  - La longitud del trazo que cumple la tolerancia de horizontalidad (T2) debe ser menor o igual a las longitudes de los trazos que verifican la tolerancia de verticalidad (T1 y T3).
  - El vértice 3 debe cumplir una tolerancia de posición respecto el punto medio de la línea 1-2.
  - El vértice 4 debe cumplir una tolerancia de posición respecto del punto medio de la línea 5-6.

5.3.2 El agente VERA. Verticalidad

El agente VERA (*VERtical Agent*) verifica si los trazos introducidos conforman el símbolo de verticalidad. Se trata también de un agente cuyo reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. En este agente se han considerado dos casos: 1) se introduce un único símbolo primitivo de ángulo debidamente orientado, 2) se introducen dos símbolos primitivos de línea que forman un determinado ángulo (ver Figura 5.14).



a) A partir de un primitivo ángulo b) A partir de dos primitivos línea

Figura 5.14. Casos considerados para el símbolo de verticalidad

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

Caso a) A partir de un único símbolo primitivo de ángulo

- Condiciones por primitivo:
  - Los vértices 1 y 3 deben cumplir la tolerancia de simetría respecto de un eje vertical trazado por el vértice 2.
  - El ángulo definido por la línea 2-3 debe estar comprendido en un determinado rango.

Caso b) A partir de dos símbolos primitivos de línea

- Condiciones por primitivo:
  - El ángulo definido por la línea 3-4 debe estar comprendido en un determinado rango.
  - El ángulo definido por la línea 2-1 debe estar comprendido en un determinado rango.
- Condiciones entre primitivos:
  - Los vértices 1 y 4 deben cumplir la tolerancia de simetría respecto de un eje vertical trazado por el punto medio de la recta que une los vértices 1 y 4.
  - El vértice 2 debe cumplir una tolerancia de posición respecto el vértice 3.

5.3.3 El agente PARA. Paralelismo

El agente PARA (*PARallel Agent*) realiza la comprobación de si los trazos introducidos conforman el símbolo de paralelismo. Se trata de un agente que establece una relación entre dos entidades geométricas, por lo que su reconocimiento pasa por la premisa de que debe haber sido introducido dos veces de forma consecutiva. El único caso a considerar por este

agente será aquel en el que el símbolo se introduce por medio de dos primitivos tipo línea (Figura 5.15).

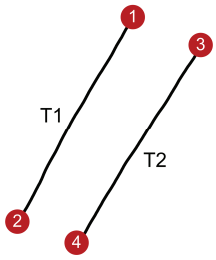


Figura 5.15. Caso considerado para el símbolo de paralelismo

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones por primitivo:
  - Las líneas 2-1 y 4-3 deben cumplir una tolerancia de inclinación.
- Condiciones entre primitivos:
  - Las líneas 1-2 y 3-4 deben cumplir una tolerancia de dimensión, manteniendo longitudes similares.
  - Los vértices 2 y 4 deben cumplir una tolerancia de posición.

Por último, y dado que se trata de un agente con referencias, para que el reconocimiento del símbolo finalice debe repetirse a continuación una segunda vez, cumpliendo nuevamente todas las condiciones expuestas.

5.3.4 El agente PERA. Perpendicularidad

El agente PERA (*PERpendicular Agent*) realiza la comprobación de si los trazos introducidos conforman el símbolo de perpendicularidad. Se trata de un agente que establece una relación entre dos entidades geométricas, por lo que su reconocimiento pasa por la premisa de que debe haber sido introducido dos veces de forma consecutiva. Se ha considerado únicamente el caso en el que el símbolo sea introducido a partir de dos primitivos tipo línea (Figura 5.16).

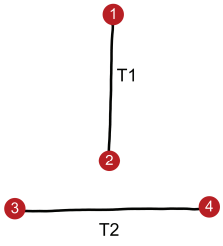


Figura 5.16. Caso considerado para el símbolo de perpendicularidad

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones por primitivo:
  - La línea 1-2 debe cumplir la tolerancia de verticalidad.
  - La línea 3-4 debe cumplir la tolerancia de horizontalidad.
- Condiciones entre primitivos:
  - Las líneas 1-2 y 3-4 deben cumplir una tolerancia de dimensión, manteniendo longitudes similares.
  - El vértice 2 debe cumplir una tolerancia de posición respecto al punto medio de la línea 3-4.

Por último, y al igual que el agente combinado anterior, por tratarse de un agente con referencias, para que el reconocimiento del símbolo finalice debe repetirse seguidamente una segunda vez cumpliendo nuevamente todas las condiciones expuestas.

5.3.5 El agente TANA. Tangencia

El agente TANA (*TANgency Agent*) verifica si los trazos introducidos conforman el símbolo de tangencia. Se trata nuevamente de un agente que establece una relación entre dos entidades geométricas, por lo que su reconocimiento pasa por la premisa de que debe haber sido introducido dos veces de forma consecutiva. El único caso a considerar por este agente será aquel en el que el símbolo sea introducido a partir de un primitivo círculo y un primitivo tipo línea (Figura 5.17).

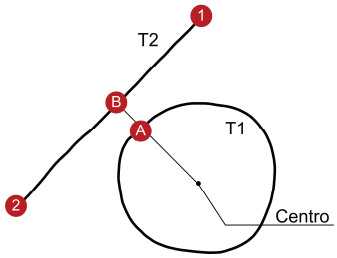


Figura 5.17. Caso considerado para el símbolo de tangencia

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones por primitivo:
  - La línea debe cumplir una tolerancia de inclinación.
- Condiciones entre primitivos:
  - Trazada la perpendicular desde el centro del círculo a la línea 1-2, y hallados los puntos de incidencia de dicha recta con ambos primitivos

(puntos A y B), el punto B deberá cumplir una tolerancia de posición respecto del punto A.

De nuevo, por tratarse de un agente con referencias, para que el reconocimiento del símbolo finalice debe repetirse seguidamente una segunda vez, cumpliendo nuevamente todas las condiciones expuestas.

5.3.6 El agente CONA. Concentricidad

El agente CONA (*CONcentric Agent*) comprueba si los trazos introducidos conforman el símbolo de concentricidad. Se trata otra vez de un agente que establece una relación entre dos entidades geométricas, por lo que su reconocimiento pasa por la premisa de que debe haber sido introducido dos veces de forma consecutiva. El único caso a considerar por este agente será aquel en el que el símbolo sea introducido mediante de dos primitivos tipo círculo (Figura 5.18).

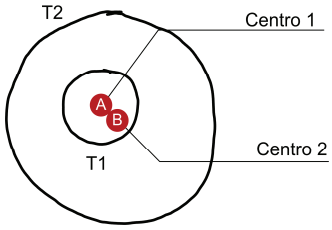


Figura 5.18. Caso considerado para el símbolo de concentricidad

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones entre primitivos:
  - Los centros de los círculos deberán cumplir una tolerancia de posición.
  - El radio del círculo de mayor diámetro deberá ser proporcional al radio del círculo de diámetro menor.

Por último, y dado que se trata nuevamente de un agente con referencias, para que el reconocimiento del símbolo finalice debe repetirse seguidamente una segunda vez, cumpliendo nuevamente todas las condiciones expuestas.

5.3.7 El agente EQUA. Igualdad

El agente EQUA (*EQUal dimension Agent*) verifica si los trazos introducidos conforman el símbolo de igualdad. Se trata de un agente que establece una relación entre dos entidades geométricas, por lo que su reconocimiento pasa por la premisa de que debe haber sido introducido dos veces de forma consecutiva. El único caso a considerar por este agente será aquel en el que el símbolo sea introducido mediante dos primitivos de línea (Figura 5.19).

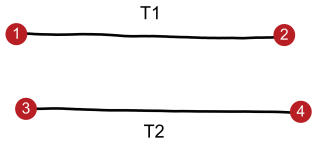


Figura 5.19. Caso considerado para el símbolo de igualdad

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones por primitivo:
  - Las dos líneas deben cumplir la tolerancia de horizontalidad.
- Condiciones entre primitivos:
  - Las líneas 1-2 y 3-4 deben cumplir una tolerancia de dimensión, manteniendo longitudes similares.
  - Los vértices 1 y 3 deben cumplir una tolerancia de posición.

Por último, y dado que se trata de un agente con referencias, para que el reconocimiento del símbolo finalice debe repetirse seguidamente una segunda vez cumpliendo nuevamente todas las condiciones expuestas.

5.3.8 El agente EXTA. Extrusión

El agente EXTA (*EXTrusion Agent*) realiza la comprobación de si los trazos introducidos conforman el símbolo de extrusión. Se trata de un agente de modelado cuya ejecución precisa de la selección previa de una superficie. En este agente se han considerado tres casos en función de los primitivos introducidos: 1) un único primitivo de extrusión, 2) un primitivo línea y un primitivo ángulo, y 3) tres primitivos tipo línea.

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

Caso a) A partir de un primitivo de extrusión

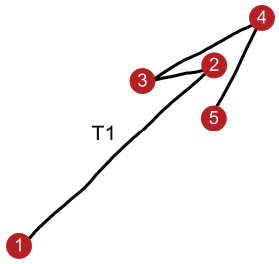


Figura 5.20. Extrusión a partir de un primitivo de extrusión

Se trata de un caso en el que el agente combinado reconoce directamente el primitivo como símbolo.

Caso b) A partir de un primitivo línea y un primitivo ángulo

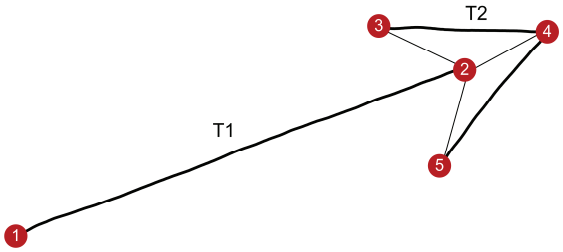


Figura 5.21. Extrusión a partir de un primitivo línea y un primitivo ángulo

- Condiciones entre primitivos:
  - La distancia entre los vértices 2-4 debe ser inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
  - El ángulo de inclinación de la línea 1-2 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 5-4.
  - La longitud de la línea 1-2 debe ser al menos el doble de la longitud de la mayor de las líneas 3-4 y 4-5.

Caso c) A partir de tres primitivos línea

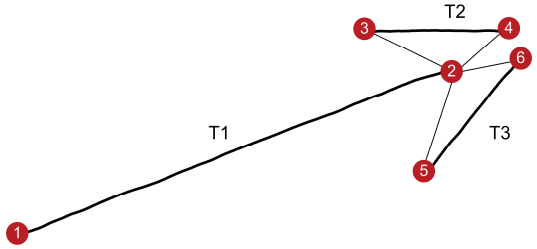


Figura 5.22. Extrusión a partir de tres primitivos línea

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la longitud de la línea 3-4.
  - La distancia entre los vértices 2 y 6 debe ser inferior a la longitud de la línea 5-6.
  - El ángulo de inclinación de la línea 1-2 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 5-6.

- La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4 y 5-6.
- El vértice 4 debe cumplir una tolerancia de posición respecto el vértice 6.

5.3.9 El agente REVA. Revolución

El agente REVA (*REVolution Agent*) realiza la comprobación de si los trazos introducidos conforman el símbolo de revolución. Se trata de un agente de modelado cuya ejecución precisa de la selección previa de una superficie. En este agente se han considerado tres casos en función de los primitivos introducidos: 1) un único primitivo de revolución, 2) un primitivo arco y un primitivo ángulo, y 3) un primitivo arco y dos primitivos línea.

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

Caso a) A partir de un primitivo de revolución

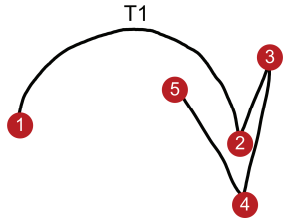


Figura 5.23. Revolución a partir de un primitivo de revolución

Se trata de un caso en el que el agente combinado reconoce directamente el primitivo como símbolo.

Caso b) A partir de un primitivo arco y un primitivo ángulo

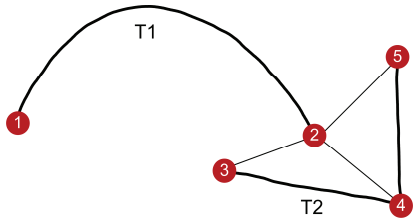


Figura 5.24. Revolución a partir de un primitivo arco y un primitivo ángulo

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
  - El ángulo formado por la línea que une los vértices 2 y 4 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 5-4.

- La longitud del arco que une los vértices 1 y 2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4 y 4-5.

Caso c) A partir de un primitivo arco y dos primitivos línea

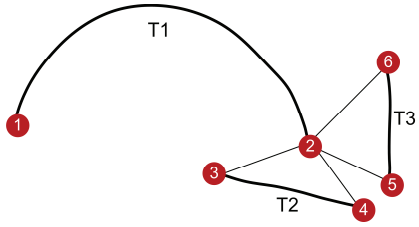


Figura 5.25. Revolución a partir de un primitivo arco y dos primitivos línea

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la longitud de la línea 3-4.
  - La distancia entre los vértices 2 y 5 debe ser inferior a la longitud de la línea 5-6.
  - La bisectriz del ángulo formado por los vértices 4-2-5 debe estar comprendida entre los ángulos formados por las líneas 3-4 y 6-5.
  - La longitud del arco 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4 y 5-6.
  - El vértice 4 debe cumplir una tolerancia de posición respecto el vértice 5.

5.3.10 El agente AXIA. Eje de simetría/revolución

El agente AXIA (*AXIs Agent*) comprueba si los trazos introducidos conforman el símbolo de un eje. Se trata de un agente cuyo reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. Este agente reconocerá el símbolo cuando se introduzcan de forma alternada una secuencia de primitivos línea y punto, debiendo existir al menos dos primitivos de tipo línea y un primitivo punto.

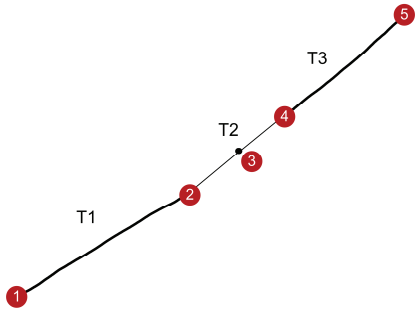


Figura 5.26. Eje de simetría/revolución a partir de una secuencia de líneas y puntos

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones entre primitivos:
  - Al introducir un primitivo punto (3), la línea ficticia que une dicho punto con el extremo final (2) del primitivo línea introducido previamente (1-2) debe cumplir una tolerancia de inclinación respecto de ésta.
  - Al introducir un nuevo primitivo línea, la línea ficticia que une el primitivo punto introducido previamente (3) con el extremo inicial (4) del nuevo primitivo línea debe cumplir una tolerancia de inclinación respecto la nueva línea.

5.3.11 El agente RADA. Cota Radial

El agente RADA (*RADial dimension Agent*) realiza la comprobación de si los trazos introducidos conforman el símbolo de cota radial. Se trata de un agente cuyo reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. Para el agente cota radial se ha implementado el mismo algoritmo que para el agente EXTA. Sin embargo, lo que diferencia a ambos agentes es que el agente RADA sólo se ejecutará si no existe selección previa de una superficie.

5.3.12 El agente DIAA. Cota Diametral

El agente DIAA (*DIAmetral dimension Agent*) verifica si los trazos introducidos conforman el símbolo de cota diametral. Se trata de un agente cuyo reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. En este agente se han considerado cinco casos en función de los primitivos introducidos: 1) un primitivo de extrusión y un primitivo ángulo, 2) un primitivo de extrusión y dos primitivos línea, 3) un primitivo línea y dos primitivos ángulo, 4) un primitivo ángulo y tres primitivos línea, y 5) cinco primitivos línea.

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

Caso a) A partir de un primitivo de extrusión y un primitivo ángulo

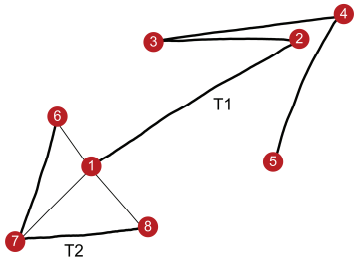


Figura 5.27. Cota diametral a partir de un primitivo de extrusión y un primitivo ángulo

- Condiciones entre primitivos:
    - La distancia entre los vértices 1 y 7 debe ser inferior a la menor de las longitudes de las líneas 6-7 y 7-8.
    - El ángulo formado por la línea que une los vértices 7 y 1 debe estar comprendido entre los ángulos formados por las líneas 7-8 y 7-6.
    - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 6-7 y 7-8.

Caso b) A partir de un primitivo de extrusión y dos primitivos línea

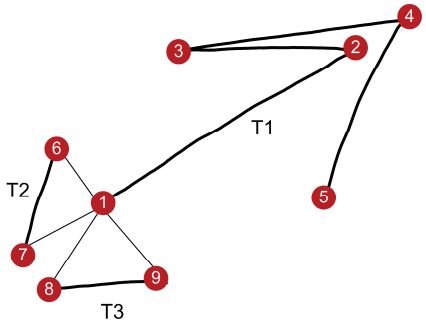


Figura 5.28. Cota diametral a partir de un primitivo de extrusión y dos primitivos línea

- Condiciones entre primitivos:
    - La distancia entre los vértices 1 y 7 debe ser inferior a la longitud de la línea 6-7.
    - La distancia entre los vértices 1 y 8 debe ser inferior a la longitud de la línea 8-9.
    - La bisectriz del ángulo formado por los vértices 7-1-8 debe estar comprendida entre los ángulos formados por las líneas 6-7 y 9-8.
    - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 6-7 y 8-9.
    - El vértice 7 debe cumplir una tolerancia de posición respecto el vértice 8.

Caso c) A partir de un primitivo línea y dos primitivos ángulo

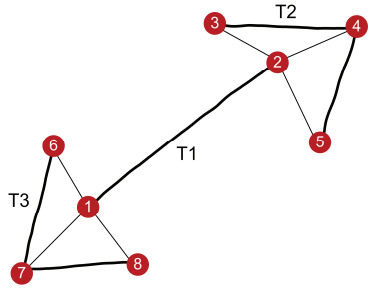


Figura 5.29. Cota diametral a partir de un primitivo línea y dos primitivos ángulo

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
  - La distancia entre los vértices 1 y 7 debe ser inferior a la menor de las longitudes de las líneas 6-7 y 7-8.
  - El ángulo formado por la línea que une los vértices 2 y 4 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 5-4.
  - El ángulo formado por la línea que une los vértices 1 y 7 debe estar comprendido entre los ángulos formados por las líneas 8-7 y 6-7.
  - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4, 4-5, 6-7 y 7-8.

Caso d) A partir de un primitivo ángulo y tres primitivos línea

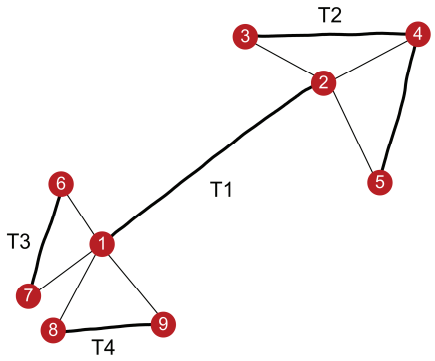


Figura 5.30. Cota diametral a partir de tres primitivos línea y un primitivo ángulo

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
  - La distancia entre los vértices 1 y 7 debe ser inferior a la longitud de la línea 6-7.
  - La distancia entre los vértices 1 y 8 debe ser inferior a la longitud de la línea 8-9.
  - El ángulo formado por la línea que une los vértices 2 y 4 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 5-4.
  - La bisectriz del ángulo formado por los vértices 7-1-8 debe estar comprendido entre los ángulos formados por las líneas 9-8 y 6-7.
  - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4, 4-5, 6-7 y 8-9.
  - El vértice 7 debe cumplir la tolerancia de posición respecto del vértice 8.

Caso e) A partir de cinco primitivos línea

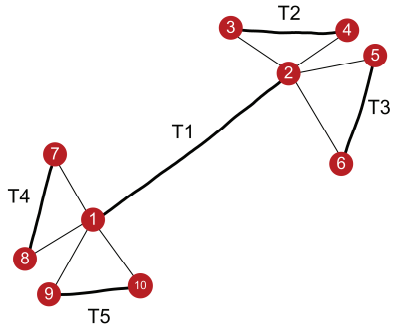


Figura 5.31. Cota diametral a partir de cinco primitivos línea

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la longitud de la línea 3-4.
  - La distancia entre los vértices 2 y 5 debe ser inferior a la longitud de la línea 5-6.
  - La distancia entre los vértices 1 y 8 debe ser inferior a la longitud de la línea 7-8.
  - La distancia entre los vértices 1 y 9 debe ser inferior a la longitud de la línea 9-10.
  - La bisectriz del ángulo formado por los vértices 4-2-5 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 6-5.
  - La bisectriz del ángulo formado por los vértices 8-1-9 debe estar comprendido entre los ángulos formados por las líneas 10-9 y 7-8.
  - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4, 5-6, 7-8 y 9-10.
  - El vértice 4 debe cumplir una tolerancia de posición respecto el vértice 5.
  - El vértice 8 debe cumplir una tolerancia de posición respecto el vértice 9.

5.3.13 El agente DIMA. Cota Lineal

El agente DIMA (*lineal DIMension Agent*) comprueba si los trazos introducidos conforman el símbolo de cota lineal. Se trata de un agente cuyo reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. En este agente se han considerado cinco casos en función de los primitivos introducidos: 1) un primitivo de extrusión, un primitivo ángulo y dos primitivos línea, 2) un primitivo de extrusión y cuatro primitivos línea, 3) tres primitivos línea y dos primitivos ángulo, 4) un primitivo ángulo y cinco primitivos línea, y 5) siete primitivos línea.

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

Caso a) A partir de un primitivo de extrusión, un primitivo ángulo y dos primitivos línea

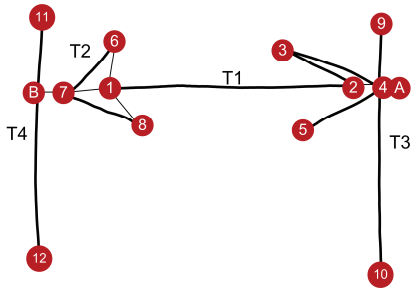


Figura 5.32. Cota lineal a partir un primitivo de extrusión, un primitivo ángulo y dos primitivos línea

- Condiciones entre primitivos:
  - La línea 1-2 debe cumplir la tolerancia de perpendicularidad respecto a las líneas 9-10 y 11-12.
  - La distancia entre los vértices 1 y 7 debe ser inferior a la menor de las longitudes de las líneas 6-7 y 7-8.
  - El ángulo formado por la línea que une los vértices 1 y 7 debe estar comprendido entre los ángulos formados por las líneas 8-7 y 6-7.
  - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 6-7 y 7-8.
  - Trazada la recta perpendicular desde el vértice 4 a la línea 9-10 y determinado el punto A de intersección entre ambas líneas, el vértice 4 debe cumplir una tolerancia de posición respecto del vértice A.
  - Trazada la recta perpendicular desde el vértice 7 a la línea 11-12 y determinado el punto B de intersección entre ambas líneas, el vértice 7 debe cumplir una tolerancia de posición respecto del vértice B.

Caso b) A partir de un primitivo de extrusión y cuatro primitivos línea

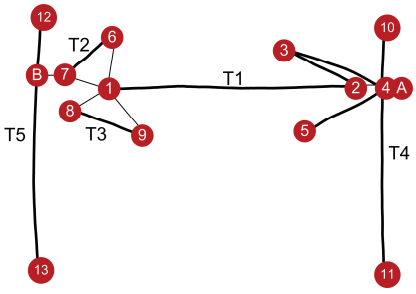


Figura 5.33. Cota lineal a partir un primitivo de extrusión y cuatro primitivos línea

- Condiciones entre primitivos:
  - La línea 1-2 debe cumplir la tolerancia de perpendicularidad respecto a las líneas 10-11 y 12-13.
  - La distancia entre los vértices 1 y 7 debe ser inferior a la longitud de la línea 6-7.
  - La distancia entre los vértices 1 y 8 debe ser inferior a la longitud de la línea 8-9.
  - La bisectriz del ángulo formado por los vértices 7-1-8 debe estar comprendido entre los ángulos formados por las líneas 6-7 y 9-8.
  - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 6-7 y 8-9.
  - Trazada la recta perpendicular desde el vértice 4 a la línea 10-11 y determinado el punto A de intersección entre ambas líneas, el vértice 4 debe cumplir una tolerancia de posición respecto el vértice A.
  - Trazada la recta perpendicular desde el vértice 7 a la línea 12-13 y determinado el punto B de intersección entre ambas líneas, el vértice 7 debe cumplir una tolerancia de posición respecto el vértice B.
  - El vértice 7 debe cumplir una tolerancia de posición respecto el vértice 8.

Caso c) A partir de tres primitivos línea y dos primitivos ángulo

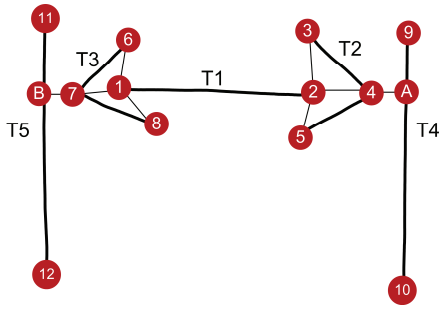


Figura 5.34. Cota lineal a partir tres primitivos línea y dos primitivos ángulo

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
  - La distancia entre los vértices 1 y 7 debe ser inferior a la menor de las longitudes de las líneas 6-7 y 7-8.
  - El ángulo formado por la línea que une los vértices 2 y 4 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 5-4.
  - El ángulo formado por la línea que une los vértices 1 y 7 debe estar comprendido entre los ángulos formados por las líneas 8-7 y 6-7.

- La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4, 4-5, 6-7 y 7-8.
- Trazada la recta perpendicular desde el vértice 4 a la línea 9-10 y determinado el punto A de intersección entre ambas líneas, el vértice 4 debe cumplir una tolerancia de posición respecto el vértice A.
- Trazada la recta perpendicular desde el vértice 7 a la línea 11-12 y de terminado el punto B de intersección entre ambas líneas, el vértice 7 debe cumplir una tolerancia de posición respecto el vértice B.

Caso d) A partir de un primitivo ángulo y cinco primitivos línea

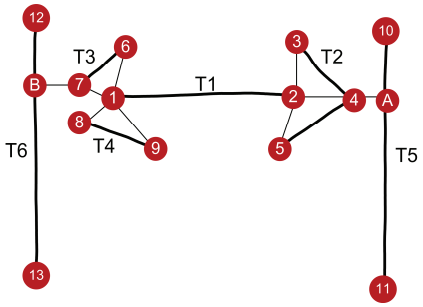


Figura 5.35. Cota lineal a partir un primitivo ángulo y cinco primitivos línea

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la menor de las longitudes de las líneas 3-4 y 4-5.
  - La distancia entre los vértices 1 y 7 debe ser inferior a la longitud de la línea 6-7.
  - La distancia entre los vértices 1-8 debe ser inferior a la longitud de la línea 8-9.
  - El ángulo formado por la línea que une los vértices 2 y 4 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 5-4.
  - La bisectriz del ángulo formado por los vértices 7-1-8 debe estar comprendido entre los ángulos formados por las líneas 9-8 y 6-7.
  - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4, 4-5, 6-7 y 8-9.
  - El vértice 7 debe cumplir una tolerancia de posición respecto el vértice 8.
  - Trazada la recta perpendicular desde el vértice 4 a la línea 10-11 y determinado el punto A de intersección entre ambas líneas, el vértice 4 debe cumplir una tolerancia de posición respecto el vértice A.
  - Trazada la recta perpendicular desde el vértice 7 a la línea 12-13 y determinado el punto B de intersección entre ambas líneas, el vértice 7 debe cumplir una tolerancia de posición respecto el vértice B.

Caso e) A partir de siete primitivos línea

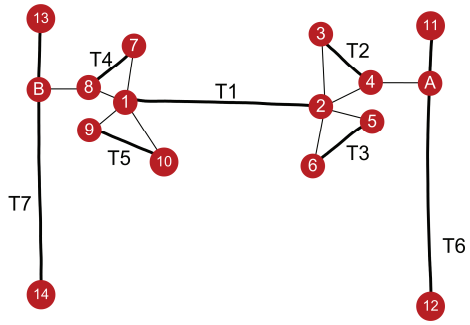


Figura 5.36. Cota lineal a partir de siete primitivos línea

- Condiciones entre primitivos:
  - La distancia entre los vértices 2 y 4 debe ser inferior a la longitud de la línea 3-4.
  - La distancia entre los vértices 2 y 5 debe ser inferior a la longitud de la línea 5-6.
  - La distancia entre los vértices 1 y 8 debe ser inferior a la longitud de la línea 7-8.
  - La distancia entre los vértices 1 y 9 debe ser inferior a la longitud de la línea 9-10.
  - La bisectriz del ángulo formado por los vértices 4-2-5 debe estar comprendido entre los ángulos formados por las líneas 3-4 y 6-5.
  - La bisectriz del ángulo formado por los vértices 8-1-9 debe estar comprendido entre los ángulos formados por las líneas 10-9 y 7-8.
  - La longitud de la línea 1-2 debe ser al menos el doble de la mayor de las longitudes de las líneas 3-4, 5-6, 7-8 y 9-10.
  - El vértice 4 debe cumplir una tolerancia de posición respecto el vértice 5.
  - El vértice 8 debe cumplir una tolerancia de posición respecto el vértice 9.
  - Trazada la recta perpendicular desde el vértice 4 a la línea 11-12 y determinado el punto A de intersección entre ambas líneas, el vértice 4 debe cumplir una tolerancia de posición respecto el vértice A.
  - Trazada la recta perpendicular desde el vértice 8 a la línea 13-14 y determinado el punto B de intersección entre ambas líneas, el vértice 8 debe cumplir una tolerancia de posición respecto el vértice B.

5.3.14 El agente LINA. Línea Artística

El agente LINA (*artistic LINe Agent*) realiza la comprobación de si los trazos introducidos conforman el símbolo de línea artística. Se trata de un agente cuyo

reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. El único caso a considerar por este agente será aquel en el que se introduzca una serie consecutiva de primitivos línea, permitiendo solape entre ellos (Figura 5.37).

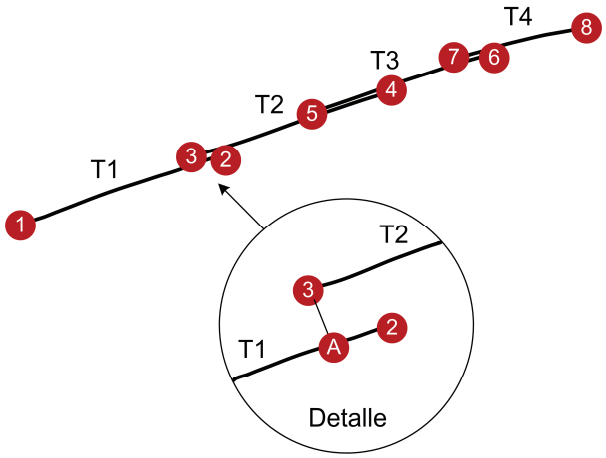


Figura 5.37. Línea artística a partir de primitivos línea

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones entre primitivos:
  - Cada primitivo línea debe cumplir una tolerancia de inclinación respecto la línea anterior.
  - Trazada la perpendicular desde el extremo inicial (3) del nuevo primitivo línea (3-4) al primitivo línea anterior (1-2), y calculado el punto de intersección (A) entre ambas líneas, el extremo inicial (3) de la nueva línea deberá cumplir una tolerancia de posición respecto el punto de intersección (A).

5.3.15 El agente ARCA. Arco Artístico

El agente ARCA (*artistic ARC Agent*) verifica de si los trazos introducidos conforman el símbolo de arco artístico. Se trata de un agente cuyo reconocimiento resulta inmediato, independientemente de cualquier otro gesto introducido previamente, y no precisa de selección alguna. El único caso a considerar por este agente será aquel en el que se introduzca una serie consecutiva de primitivos arco, permitiendo solape entre ellos (Figura 5.38).

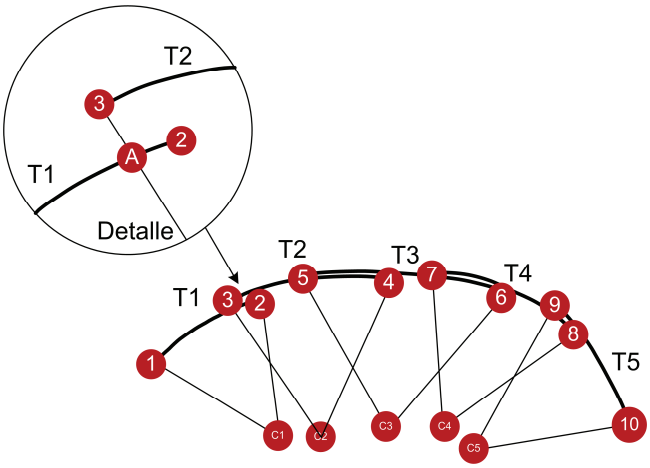


Figura 5.38. Arco artístico a partir de varios primitivos arco

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones entre primitivos:
  - La posición de los centros de cada primitivo arco debe cumplir unas determinadas condiciones (que estén en el mismo lado cóncavo de los arcos, que se ajusten a una regresión lineal o curva, etc.).
  - Trazado el radio desde el extremo inicial (3) del nuevo primitivo arco (3-4), y calculado su punto de intersección (A) con el arco anterior (1-2), el extremo inicial (3) del nuevo arco deberá cumplir una tolerancia de posición respecto el punto de intersección (A).

5.3.16 El agente SECA. Rayado.

El agente SECA (*SECtion Agent*) realiza la comprobación de si los trazos introducidos conforman el símbolo de sección/rayado. Se trata de un agente de modelado cuya ejecución precisa de la selección previa de un contorno. El único caso a considerar por este agente será aquel en el que se introduzcan más de dos primitivos del tipo línea (Figura 5.39).

Para que este agente lleve a cabo un reconocimiento positivo se deben cumplir las siguientes condiciones:

- Condiciones entre primitivos:
  - Los primitivos línea deben encontrarse en el interior del contorno seleccionado previamente.
  - Cada nuevo primitivo línea debe cumplir una tolerancia de orientación y espaciado respecto a la línea introducida anteriormente.

- Los extremos de cada línea introducida (2, 4, 6, etc…) deben cumplir una tolerancia de posición respecto de los extremos de la línea anterior introducida.

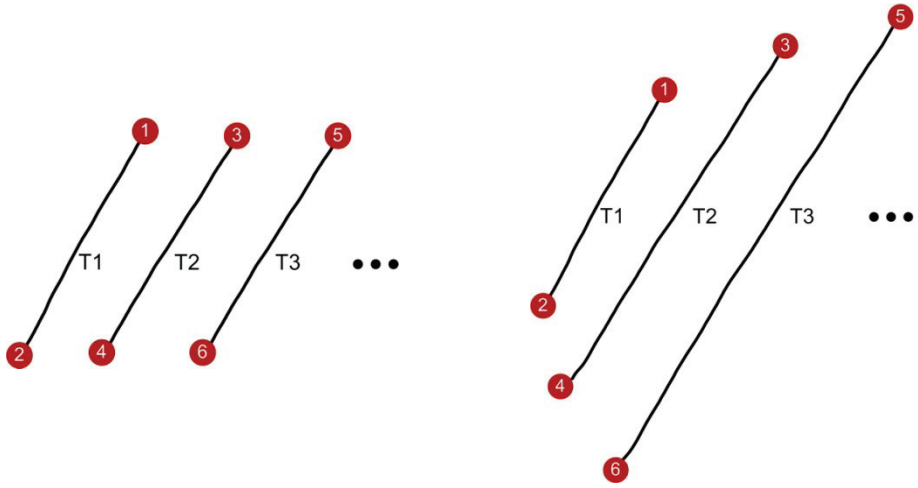


Figura 5.39. Caso considerado para el símbolo de rayado

Capítulo 6:
OPTIMIZACIÓN DE LOS PARÁMETROS DEL PROCESO DE SEGMENTACIÓN

6. OPTIMIZACIÓN DE LOS PARÁMETROS DEL PROCESO DE SEGMENTACIÓN

Comúnmente, los parámetros iniciales de muchas aplicaciones se presentan con unos valores por defecto, los cuales suelen estar a disposición del usuario que irá ajustándolos a medida que lo necesite, para mejorar el comportamiento de dichas aplicaciones u obtener unos determinados resultados. El ajuste de estos parámetros suele ser manual, y el número de combinaciones de los valores de todos ellos es tan elevada, que nunca se consiguen los valores óptimos para la obtención de los mejores resultados.

En muchos problemas relacionados con los sistemas de visión o el análisis de imagen, el rendimiento del proceso de segmentación es altamente dependiente del algoritmo implementado y de los parámetros utilizados. El ajuste de dichos parámetros para el buen funcionamiento de la aplicación es complejo y costoso, conociéndose este ajuste como "entrenamiento del sistema". Es por ello que los valores de los parámetros utilizados en la fase de segmentación desarrollada para este reconocedor, se han determinado a través de técnicas de optimización, y más concretamente a través de la técnica Simulated Annealing o SA.

Siendo que el entrenamiento del sistema es un proceso obligado en este tipo de sistemas, se ha considerado obligado el aplicar técnicas de optimización para mejorar al máximo el rendimiento de dicho entrenamiento. El por qué se ha utilizado le técnica del SA y no otra es debido a que esta técnica está muy extendida y contrastada, a la vez que es sencilla y fácilmente adaptable al problema que nos ocupa. Cualquier otra comparativa entre técnicas de optimización para evaluar la bondad de cada una de ellas relativo a este problema queda fuera del ámbito de esta tesis. Así pues, en este capítulo se incluye una revisión de trabajos relacionados con procesos de segmentación o asociados a él y que utilizan alguna técnica de optimización; se describe la técnica del Simulated Annealing utilizada para el ajuste de los parámetros de la segmentación; y se muestran los resultados obtenidos tras la aplicación del algoritmo de mejora.

6.1 Revisión de trabajos relacionados

La optimización se ha usado en muchas y variadas aplicaciones para conseguir mejores resultados finales, y aunque en el campo de la segmentación no está todavía muy extendido su uso, sí que se pueden encontrar en la literatura trabajos interesantes que mejoran diseños de redes, de sistemas de iluminación para aplicaciones de visión por computador, y otro tipo de aplicaciones relacionadas con lo que se pretende en esta tesis. Así por ejemplo, Li y Ni [LN09] sustituyen el método tradicional *constrained fitting* en ingeniería inversa y proponen un método más flexible en el que una curva a modo de plantilla se va deformando para ir ajustándose a los puntos de la sección usando técnicas de optimización.

Lavoué y Wolf [LW08] también aplican técnicas de optimización al análisis de mallas y *clustering* de mallas 3D, desarrollando un algoritmo basado en modelos de gráficos probabilísticos y obteniendo soluciones globales óptimas utilizando únicamente iteraciones locales, lo que consiguen gracias a la propiedad de campo aleatorio (*ramdom field*) de Markov.

En [TW04], Taylor y Wolf optimizan diez parámetros usados para detección de texto en un algoritmo de indexado semántico que detecta texto en imágenes y secuencias de vídeo, demostrando que los parámetros elegidos son más eficaces que los previamente recomendados.

Brazil y Thomas [BT07] abordan la optimización de redes para tratar problemas de diseño en la infraestructura de minas; la idea es diseñar un sistema conectado de descensos, rampas, conducciones y pozos para minimizar los costes de desarrollo y transporte de las minas.

El dibujo o diseño mediante bocetos representa una amplia variabilidad de formas, y el razonamiento cualitativo tradicional es incapaz de modelar tal grado de ambigüedad en las formas a través de sus parámetros. Mukerjee et al. [MA97] en su trabajo *Qualitative sketch optimization*, controlan la variación de estos parámetros definiendo una discretización apropiada sobre el espacio de dichos parametros que deriva en una clase de formas similares a partir de las cuales puede obtenerse una forma optimizada. Para ello usan algoritmos genéticos y dan resultados para contornos de automóviles, partes mecánicas, perfiles de construcciones, etc.

También, Davis et al. [DCL08] en su trabajo *K-sketch: a 'kinetic' sketch pad for novice animators*, desarrollaron una aplicación 2D de animación de *sketching* utlizando una técnica novedosa de optimización produciendo un interfaz rápido, simple y potente. El resultado fue un sistema caligráfico donde los participantes trabajaron tres veces más rápido, necesitaron la mitad del tiempo de aprendizaje que para sistemas similares de animación que no utilizan *sketching*, y necesitaron menos especialización que con aplicaciones similares (concretamente se compararon con PowerPoint).

En [MS07], los autores trabajan sobre la segmentación en color de etiquetas visuales para identificación de objetos. Ya que pequeñas variaciones en la iluminación producen mucha incertidumbre cuando se aplican umbrales de color que además afectan al resultado final del sistema, los autores proponen una solución que pasa por el empleo de un algoritmo genético para generar umbrales fiables para la identificación del color, demostrando más precisión y fiabilidad que con el método de optimización basado en la distancia máxima.

Un marco novedoso de actuación para la segmentación precisa de nódulos de tiroides en imágenes por ultrasonidos se presenta en [ISK07], donde se incorpora el uso de regiones de fondo variable para reducir el efecto de la intensidad en las imágenes de ultrasonidos. Este sistema contempla el ajuste de los parámetros a través de un mecanismo basado en algoritmos genéticos para encontrar las mejores soluciones sin necesidad de perfiles expertos. Los experimentos llevados a cabo fueron efectivos y eficientes a la hora de delinear los contornos de los nódulos.

En [Gre86], los autores afirman que la tarea de optimización de sistemas complejos presenta al menos dos niveles de problemas. El primer nivel correspondería a la elección del algoritmo de optimización apropiado para un determinado sistema; y el segundo nivel en el que se precisa el ajuste de los parámetros para mejorar el resultado. En este trabajo los autores

utilizan algoritmos genéticos tanto para la elección del método de optimización como para el ajuste de los parámetros del sistema que aplican a la resolución de un conjunto de problemas de optimización numérica. Los resultados probaron que los algoritmos genéticos fueron efectivos en ambos niveles.

Son muchos los problemas encontrados en aplicaciones de visión artificial, donde el rendimiento del proceso de segmentación es fuertemente dependiente del algoritmo empleado y de su parametrización, tareas que suelen ser complejas y altamente costosas. En [MT07] los autores presentan una propuesta para realizar tareas orientadas a la segmentación y que está basada en técnicas de aprendizaje. Su contribución principal es la propuesta de una metodología para configurar la segmentación en sistemas de visión de una manera rápida y sencilla. Los experimentos demostraron que su esquema de optimización es fiable para diferentes algoritmos de segmentación del estado del arte y los resultados obtenidos que la precisión final obtenida es mayor que en el caso de no existir optimización.

Gelasca et al. proponen en [GSE03] un marco de trabajo para el ajuste intuitivo de parámetros en algoritmos de segmentación de imágenes y video. Este marco permite a los investigadores y diseñadores encontrar rápida y automáticamente los mejores valores de los parámetros en los algoritmos de segmentación que han desarrollado, permitiendo además facilitar el entendimiento de cómo funcionan dichos parámetros y su repercusión sobre el resultado final de la segmentación. Las pruebas específicas se llevaron a cabo utilizando un algoritmo de segmentación de video invariante a las sombras y para la optimización se utilizó el método de *down-hill* simple.

Crevier [Cre08] intenta resolver el problema de la cuantificación de la precisión cuando se comparan los resultados obtenidos del sistema de visión con las referencias de la segmentación. Para ello utiliza una mezcla entre método determinístico y método de Monte-Carlo con los que optimiza los parámetros de los algoritmos a comparar.

Ahmed y Eid [AE07] por su parte presentan una técnica de filtrado basada en el contenido para mejorar documentos escaneados. Una fase de clasificación de la imagen clasifica cada pixel en texto, fondo o regiones de la imagen. En la fase de segmentación resaltan el texto y detalles similares mientras suavizan el fondo y el contenido de la imagen. Para seleccionar los parámetros óptimos en la segmentación, formulan una función de coste que minimice el número de píxeles erróneamente clasificados comparando el resultado con imágenes de referencia mediante algoritmos genéticos. También Yu y Fan [YF08] usan un algoritmo genético para determinar el valor óptimo del parámetro principal de la técnica de segmentación *Generalized fuzzy entropy thresholding,* entre otros parámetros. Los resultados experimentales mostraron que dicho método mejoraba los resultados previos obtenidos.

Como se deduce de los trabajos expuestos brevemente, los métodos de optimización mejoran definitivamente los resultados de los algoritmos de segmentación y clasificación utilizados en aplicaciones de visión, lo que viene a justificar el trabajo llevado a cabo en este capítulo de la tesis.

6.2 Antecedentes históricos

Desde 1983, el campo de aplicación de esta técnica se ha extendido a infinidad de problemas de optimización de tipo combinatorio con excelentes resultados. La técnica del "recocido simulado" (simulated annealing) es una técnica de búsqueda aleatoria dirigida, que surge en 1983 con la publicación del artículo de Kirkpatrick et al. [KGV83]. Las técnicas de búsqueda aleatoria dirigida (Figura 6.1) están basadas en métodos enumerativos aunque utilizan información adicional para guiar la búsqueda. Éstas son técnicas de propósito general y se pueden aplicar a un amplio número de problemas.

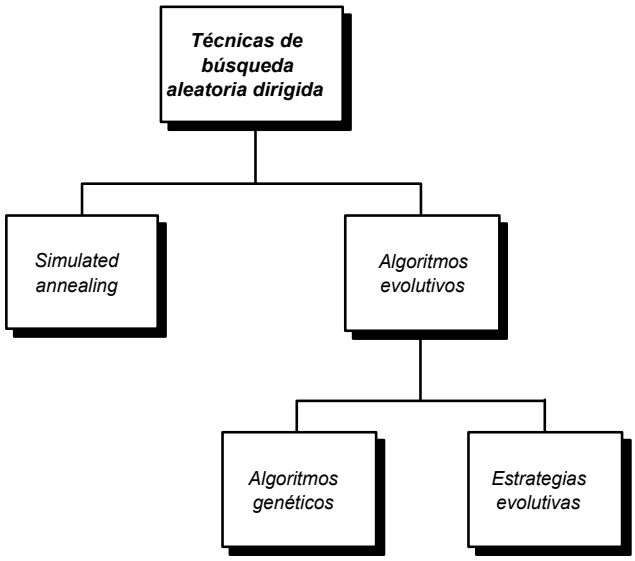


Figura 6.1. Clasificación de técnicas de búsqueda aleatoria dirigida

Estas técnicas se dividen fundamentalmente en dos familias principales: "Simulated Annealing" o SA y "Algoritmos Evolutivos". Ambos tipos de técnicas son procesos evolutivos, ya que las técnicas SA utilizan un proceso de evolución termodinámica para buscar el estado de mínima energía, y los algoritmos evolutivos utilizan el principio de selección natural darwiniano.

El término "Simulated Annealing", que se podría traducir al castellano por "Recocido Simulado" hace referencia a un tratamiento térmico como es el "recocido" que, según la acepción recogida en el Diccionario de la Lengua Española de la Real Academia Española significa "caldear los metales para que adquieran de nuevo la ductilidad o el temple que suelen perder al trabajarlos". En el desarrollo del presente capítulo se justifica la razón de utilizar este símil cuando realmente se está tratando de resolver un problema de optimización.

Además, la técnica de "Simulated Annealing", que de aquí en adelante se denominará genéricamente SA, difiere de las técnicas tradicionales de mejora iterativa en su capacidad para escapar de mínimos locales gracias al empleo del criterio Metropolis [MRR53].

6.3 El algoritmo de segmentación TCVD (*Tangent Corner Vertices Detection*)

El algoritmo de segmentación (detección de vértices tangentes y esquina) del que se pretende optimizar sus parámetros para su mejor funcionamiento posible está detallado en el capítulo 4. Materiales y métodos de esta tesis. Como ya se ha ilustrado en dicho capítulo, su diagrama de flujo es el siguiente.

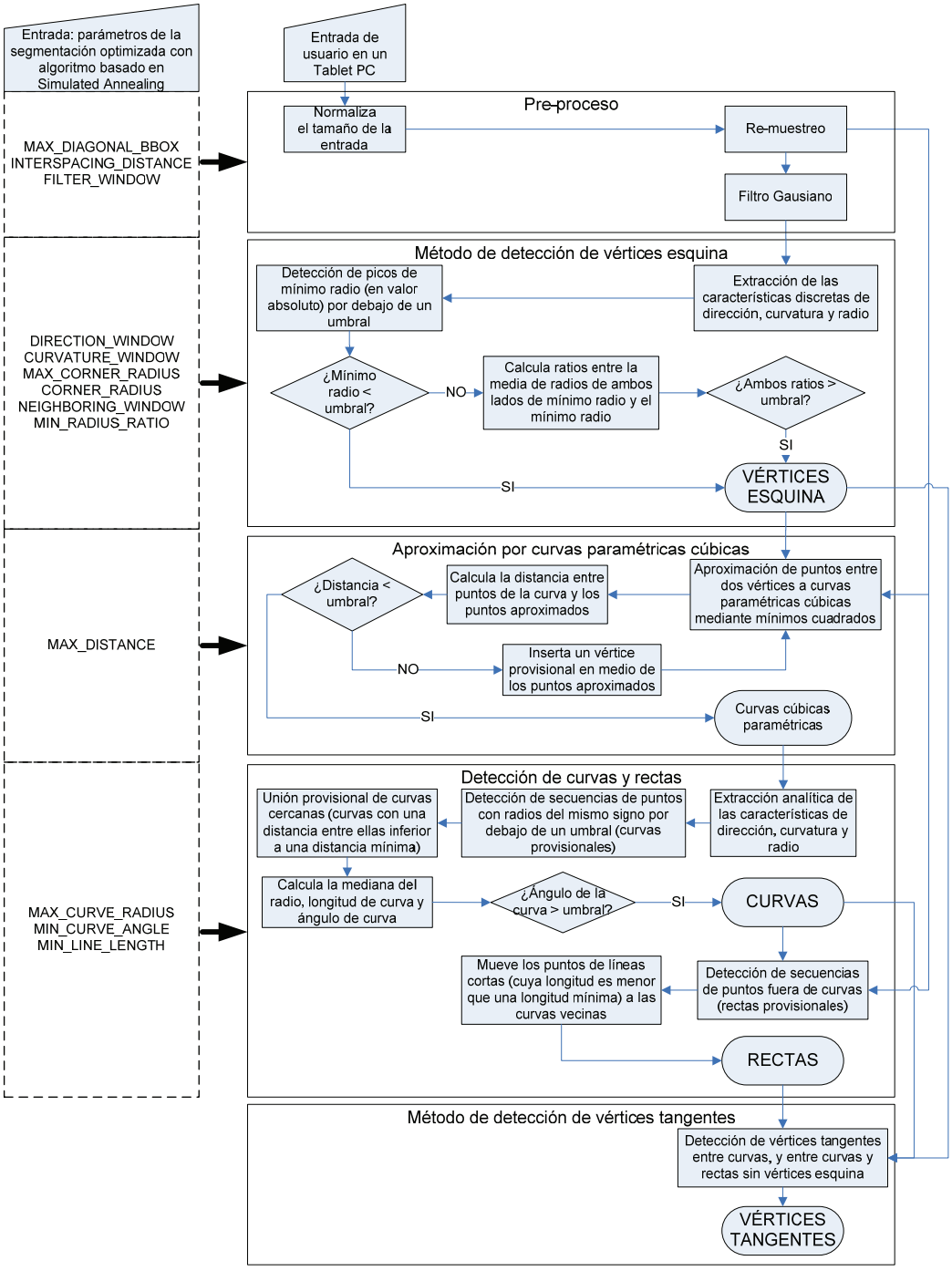


Figura 6.2. Diagrama de flujo del algoritmo de segmentación TCVD

A continuación, en la Tabla 6.1 se muestran los parámetros que controlan el proceso de segmentación, y que en nuestro caso son 13. Para cada parámetro o variable (columna de la izquierda) se establece empíricamente un rango de valores (columna de la derecha) deducido de la experiencia, es por esto que se habla de rango inicial de valores.

Tabla 6.1. Parámetros de la segmentación y su rango inicial de valores

Parámetros	Descripción	Rango
MAX_DIAGONAL_BBOX	Máxima longitud de la diagonal de la caja de inclusión del trazo	500
INTERSPACING DISTANCE	Distancia entre los puntos del trazo	[1,6]
FILTER_WINDOW	Tamaño de la ventana para realizar el filtrado Gaussiano	[1,8]
DIRECTION_WINDOW	Tamaño de la ventana para obtener la dirección de la tangente en cada punto del trazo	[1,8]
CURVATURE_WINDOW	Tamaño de la ventana para obtener la curvatura en cada punto del trazo	[1,3]
MAX_CORNER_RADIUS	Máximo valor para el radio en los candidatos a vértice esquina (corner vertex)	[10,100]
CORNER_RADIUS	Valor de radio por debajo del cual el candidato es un vértice esquina seguro	[5,50]
NEIGHBORING_WINDOW	Tamaño de la ventana para comparar el radio mínimo en un candidato a vértice esquina con el radio de sus vecinos	[2,20]
MIN_RADIUS_RATIO	Mínimo valor del cociente del radio en el entorno de un candidato a vértice y el radio mínimo en dicho vértice, para que el candidato sea vértice esquina	[1,10]
MAX_DISTANCE	Máxima distancia posible entre los puntos del trazo y las curvas paramétricas de la aproximación	5
MAX_ARC_RADIUS	Máximo valor para el radio en los candidatos a arco	[100,1000]
MIN_ARC_ANGLE	Mínimo valor de ángulo para que el candidato sea un arco	[0º,60º]
MIN_LINE_LENGTH	Mínima longitud de una recta para que no la absorba en un arco vecino	[25,100]

6.4 El ajuste o *tuning* de los parámetros de la segmentación

El rango inicial de cada parámetro, establecido empíricamente en la Tabla 6.1, no es en principio un rango óptimo para el proceso de segmentación, ya que se ha fijado a partir de observaciones previas y deducido finalmente de la experiencia, lo que puede provocar que la solución óptima del problema no se encuentre dentro del rango establecido inicialmente y, por tanto, no se obtengan los mejores resultados en la segmentación.

Habrá que realizar pues un proceso previo para ajustar el rango válido de cada parámetro, y asegurarse de que su valor óptimo se encuentra dentro de dicho rango. Para el proceso de corrección del rango se establecen cuatro pasos principales:

1) En primer lugar, se establece un rango inicial para cada parámetro deducido de la experiencia, tal como se muestra en la Tabla 6.1.
2) Luego se ejecuta el proceso de SA dentro de los rangos iniciales de los parámetros y se obtienen los valores óptimos de cada parámetro dentro de dichos rangos.

3) Posteriormente, para el parámetro p_i del que se desea corregir el rango, se ejecuta el algoritmo de segmentación variando dicho parámetro en todo su rango inicial y fijando el resto de parámetros al valor óptimo encontrado en el paso anterior b).
4) Por último, el valor corregido del rango del parámetro p_i se establece primero centrando su valor óptimo, encontrado en el paso c), en el nuevo rango, y posteriormente se fijan los valores que delimitan dichos rangos.

Además, el proceso de optimización (que afecta al paso 4) se ha llevado a cabo repetidas veces con valores diferentes de los parámetros intrínsecos al SA para observar su influencia en el proceso (como el resultado final, tiempo de convergencia, etc.) y obtener diferentes optimizaciones. Fruto de estos resultados se han elegido unos parámetros de preferencia del SA, como se muestra más adelante en los apartados 0 y 6.4.3, donde además se muestra todo este proceso.

6.4.1 Cálculo de la función de coste

Para evaluar la calidad final de la segmentación que se obtiene tras el proceso de optimización, es importante definir una función de coste consistente en la minimización de un valor. La función de coste (6-1) devuelve un valor real (coste) a partir de un conjunto n de parámetros que controlan la segmentación $P = \{p_1, p_2, \ldots p_n\}$.

$$c : P \rightarrow R \qquad (6\text{-}1)$$

El valor del coste para un conjunto de parámetros dado será el ratio de formas mal segmentadas usando dicho conjunto de parámetros. Se ha creado una librería de figuras bocetadas junto con su segmentación correcta para implementar la función de coste. La Figura 6.3 muestra en cada fila la forma modelo, la segmentación correcta para la forma bocetada (△) y la segmentación por el método TCVD (□) respectivamente, donde las primitivas aproximadas aparecen en color rojo, fijándose la segmentación correcta en acuerdo a la forma modelo donde los vértices tangentes aparecen en azul cian y las esquinas en color verde.

El criterio aplicado para definir una forma bien segmentada se ha basado en el criterio todo o nada (*all-or-nothing accuracy*), esto es, una forma estará bien segmentada si se cumplen lo siguiente:

- El número de vértices debe coincidir con la segmentación ideal o correcta.
- El tipo de las primitivas aproximadas entre los vértices encontrados deben coincidir con las primitivas existentes en la segmentación correcta.

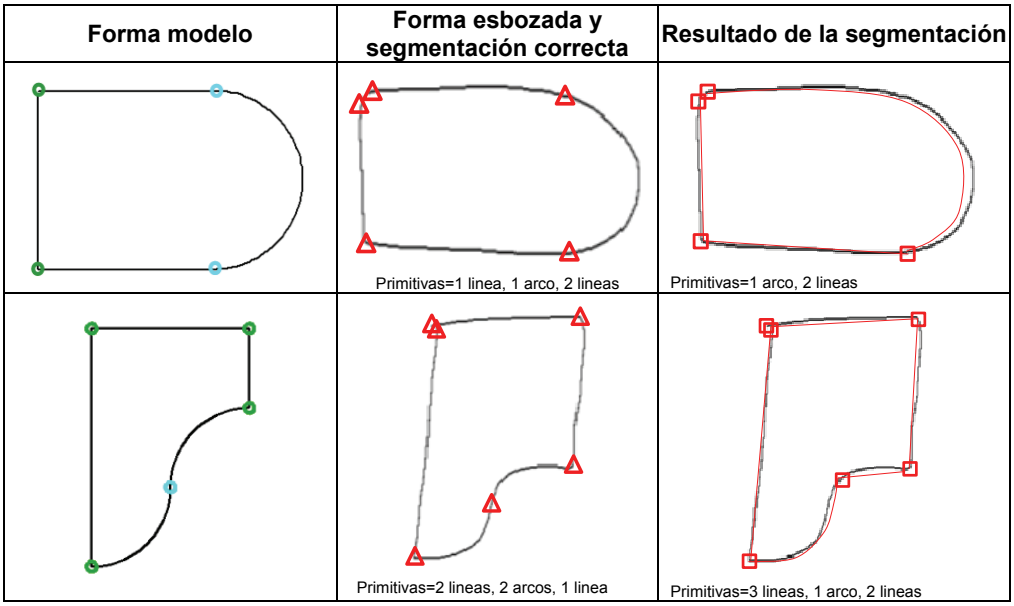


Figura 6.3. Ejemplos de errores en la segmentación

6.4.2 El algoritmo de Simulated Annealing

Como se ha comentado, el rango óptimo de valores de los parámetros de la segmentación se ha determinado utilizando el algoritmo de simulated annealing [KGV83]. El algoritmo de SA se muestra en la Figura 6.4, pudiéndose encontrar en [LA87] y [OG89] todo el desarrollo matemático incluyendo los aspectos de convergencia del mismo.

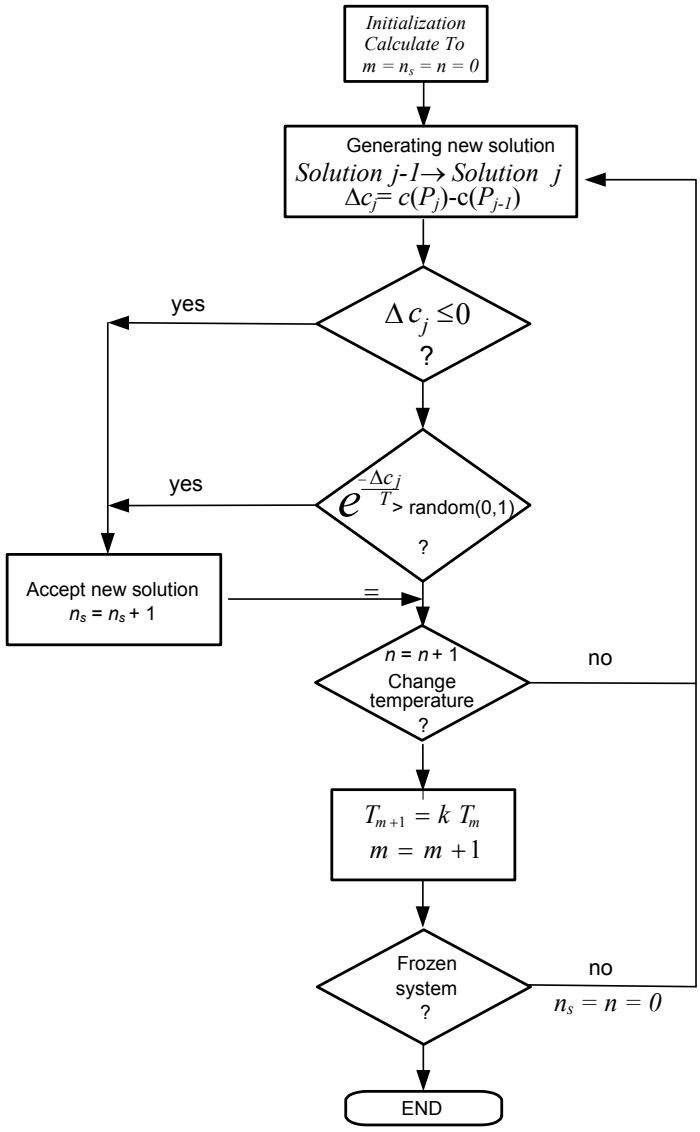


Figura 6.4. Algoritmo SA

Los parámetros involucrados en el SA pueden ser clasificados como genéricos (propios del SA) o específicos (del problema de optimización). Los **parámetros genéricos** controlan el funcionamiento del algoritmo y son los siguientes:

- *Temperatura inicial*: valor inicial de la temperatura al comienzo del proceso SA. Se calcula mediante la expresión:

$$T_0 = \frac{\overline{\Delta C^{(+)}}}{\ln\left(\chi_0^{-1}\right)} \qquad (6\text{-}2)$$

Donde $\overline{\Delta C^{(+)}}$ es el valor medio del incremento de coste asociado a aquellos desplazamientos con un incremento en la función de coste. En nuestro caso particular, los test realizados mostraron que para 100 desplazamientos, $\overline{\Delta C^{(+)}} \approx 1.5$. χ_0 es el coeficiente inicial de aceptación (χ es el ratio de desplazamientos aceptados con respecto del número total de intentos de desplazamiento). Usando un valor típico deducido de la literatura para este coeficiente (χ_0=*0.75*), el valor para la temperatura inicial resulta $T_0 \approx 5$.

- *Esquema de enfriamiento*: indica como la temperatura varía entre dos escalones diferentes. La forma más común de este esquema consiste en un enfriamiento potencial, esto es, más rápido a más altas temperaturas:

$$T_{m+1} = k \cdot T_m \qquad (6\text{-}3)$$

Donde *m* es el escalón de temperatura actual y *k* es el coeficiente de enfriamiento. Un enfriamiento rápido (valores pequeños de *k*) causa una caída rápida en el coste, pero si es demasiado rápido, el SA puede terminar sin alcanzar buenos resultados. Además, un enfriamiento lento hace que la función de coste experimente subidas y bajadas (se permiten desplazamientos de coste elevado) pero se incremente el número de iteraciones (tiempo requerido) y la probabilidad de encontrar la solución óptima. Un valor de 0.5 para *k* ha demostrado ser un buen compromiso entre el coste y la velocidad de convergencia.

- *Longitud de la cadena de Markov (criterio de cambio de la temperatura)*: decide cuando la temperatura debe bajar al siguiente escalón utilizando la condición lógica siguiente:

$$\left(n_s > n_s^{\max}\right) OR \left(n > n^{\max}\right) \qquad (6\text{-}4)$$

Donde:
  - n_s es el número de desplazamientos con éxito en el actual escalón de temperatura.
  - n_s^{max} es el máximo número de desplazamientos con éxito en un escalón de temperatura. Normalmente, este número se determina con respecto al tamaño del espacio de soluciones. En este caso, el calor utilizado es el número de parámetros de la segmentación a optimizar, que es n_s^{max}=*13*.
  - *n* es el número de desplazamientos intentados en el actual escalón de temperatura.
  - n^{max} es el máximo número de desplazamientos intentados en un escalón de temperatura, y normalmente es proporcional al valor n_s^{max}. En el contexto actual, $n^{max} = 2 \cdot n_s^{max}$, y por tanto n^{max} =*26*.

- *Criterio de congelación*: es el criterio de parada que termina el proceso de SA:

$$\left(\chi < \chi^{\min}\right) OR \left(m > m^{\max}\right) \qquad (6\text{-}5)$$

El criterio de congelación puede alcanzarse cuando la temperatura ha experimentado un número determinado de descensos (m^{max}), o cuando el coeficiente de aceptación es muy bajo ($\chi = n_s / n$), donde:

- χ^{min} es el mínimo coeficiente de aceptación, donde se ha seleccionado el valor 1% para este caso particular.
- m^{max} es el máximo número permitido de escalones de temperatura, que en este caso se ha fijado al valor 50.

Los **parámetros específicos** del SA en el contexto de nuestro problema son los siguientes:

- *El espacio de soluciones P*, que en este caso corresponde con los 10 parámetros utilizados en la segmentación (ver Tabla 6.1):

$$P = \{p_1, p_2, \ldots p_{13}\} \qquad (6\text{-}6)$$

- *Un mecanismo de desplazamiento para elegir una solución próxima a la actual.* Para cada parámetro p_i a optimizar en la iteración j, su nuevo valor se obtendrá a partir de su valor anterior (correspondiente a la iteración $j-1$) como sigue:

$$p_{ij} = p_{i(j-1)} + \delta_i \cdot \xi_{ij} \qquad (6\text{-}7)$$

Donde $p_i \in [p_i^{min}, p_i^{max}]$, $\delta_i = \alpha \cdot (p_i^{max} - p_i^{min})$, y α es el máximo ratio de desplazamiento ($\alpha = 0.1$ ha proporcionado un buen rendimiento para el problema a resolver). ξ_{ij} es un valor aleatorio contenido en el rango $[0,1]$, y que se ha obtenido para cada parámetro en cada iteración.

- *La función de coste* $c : P \rightarrow R$ que proporciona un valor que debe ser minimizado. Para un conjunto de entrenamiento n_t de figuras bocetadas, esta función se ha definido como:

$$c = n_m / n_t \qquad (6\text{-}8)$$

Donde n_m es el número de errores correspondientes a aquellas formas mal segmentadas aplicando el criterio definido en el apartado 6.4.1.

La mejor solución obtenida durante el proceso de optimización se almacena para evitar que la solución final no se corresponda con la solución óptima.

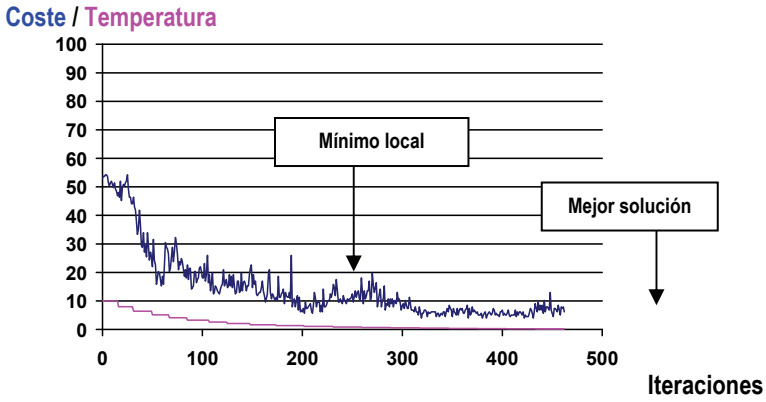


Figura 6.5. Evolución del coste y la temperatura en el proceso SA

Como ya se ha visto, el algoritmo de Simulated Annealing realiza una serie de iteraciones, partiendo de un estado inicial de los parámetros, y llevando a cabo desplazamientos acotados de los mismos para buscar el estado óptimo en el que el coste se minimiza. Este algoritmo se puede escribir de la siguiente manera:

```
InicializarParametros();
CalcularCoste();
InicializarTemperatura();
Repetir {
Repetir {
        EvaluarDesplazamiento();
        CalcularIncrementoCoste();
        Si (IncrementoCoste <= 0) {
                        RealizarDesplazamiento();
                }
        Sino {
                        -IncrementoCoste
                Si (e  Temperatura     > Aleatorio(0,1)) {
                                RealizarDesplazamiento();
                        }
                        Sino {
                                RechazarDesplazamiento();
                        }
                } //fin si
        } //fin si
Hasta (CondicionEquilibrio);
EnfriarTemperatura();
} Hasta (CondicionCongelacion);
```

6.4.3 Ajuste del rango de los parámetros específicos

Como se ha comentado anteriormente, para establecer el rango definitivo de un parámetro p_i se han fijado el resto de parámetros a su valor óptimo (obtenido de la aplicación previa del simulated annealing) y se han obtenido los valores de la función de coste para todos los valores de dicho parámetro dentro del rango inicial. El rango definitivo se ha fijado centrando el valor óptimo obtenido y los límites del rango se han expandido o comprimido para impedir que la función de coste no supere en ningún caso el 50%.

La Figura 6.6(a) muestra la gráfica de las variaciones en todo el rango inicial del valor del parámetro *MIN_ARC_ANGLE*, en la que se observa que el menor coste obtenido para todos los valores del rango fijado inicialmente puede no ser el mínimo coste posible: aparentemente se podrían obtener menores valores de la función de coste si se ampliara el rango por la parte derecha. Tras el proceso llevado a cabo, los rangos se establecen de modo que el valor del parámetro que corresponde al menor valor del coste quede centrado en dicho rango. En la Figura 6.6(b) se aprecia la corrección del rango útil del parámetro para contener la mejor solución posible. El proceso de corrección del rango se repite para todos los parámetros del proceso de segmentación, haciendo variar únicamente el parámetro a corregir y dejando el resto a valores fijos.

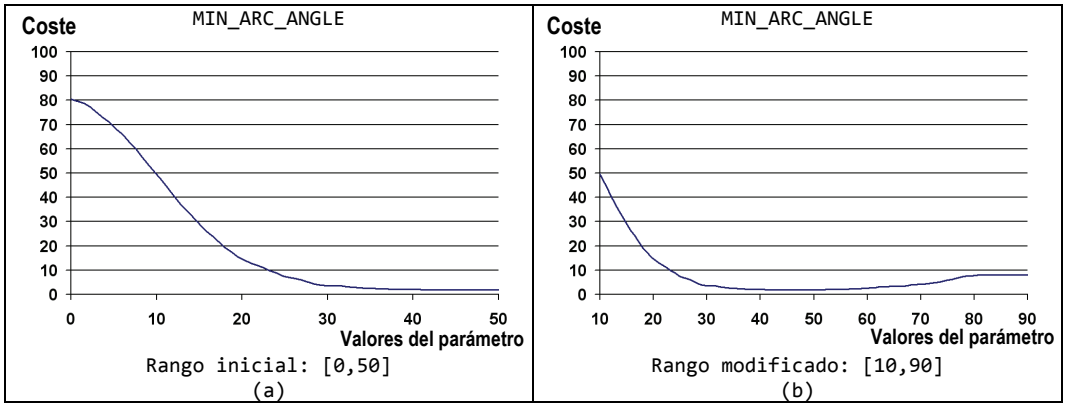


Figura 6.6. Coste fijando el resto de parámetros a sus valores óptimos, y variando únicamente el parámetro especificado: para el rango inicial (izquierda); y para el rango corregido (derecha)

Una vez todos los rangos han sido ajustados, otro proceso de simulated annealing proporciona el valor final de cada parámetro para realizar la segmentación.

6.5 Diseño de experimentos. Resultados

Para el diseño experimental se ha seguido el siguiente proceso:

1. Para cada parámetro del problema a optimizar:
    a. Asignar sus valores iniciales estimados:
        i. Valor inicial.
        ii. Rango inicial de posibles valores en el que el parámetro varía.
    b. Segmentación del conjunto de test de figuras bocetadas, variando el valor del parámetro a optimizar dentro de su rango inicial y dejando fijos los valores del resto de parámetros. Lo que nos permite comprobar que la mejor solución está contemplada dentro del rango del parámetro. En caso contrario, el rango se corrige para contemplar dichos valores óptimos.

2. Para los parámetros del SA:
    a. Se fijan los valores iniciales de sus parámetros como sigue:

α	T_0	k	ns^{max}	n^{max}	χ^{min}	m^{max}
0.2	5	0.8	13	26	0.01	50

    b. Se llevan a cabo varias pruebas variando los parámetros a valores mayores y menores de los inicialmente establecidos en el punto anterior y se comprueban los siguientes resultados:
        i. Resultado: evolución del coste cuando se obtiene el mejor resultado.
        ii. Coste temporal: número de iteraciones y tiempo empleado.
    c. Ajuste de los valores de los parámetros del SA (los nuevos valores establecidos del ajuste realizado aparecen abajo en rojo):

α	T_0	k	ns^{max}	n^{max}	χ^{min}	m^{max}
0.1	5	**0.5**	13	26	0.01	50

El proceso de segmentación se ha aplicado a una colección de 680 muestras de 17 figuras diferentes bocetadas, introducidas por distintos usuarios y guardadas en una BD. Para la evaluación del algoritmo de optimización SA se ha procedido de la siguiente manera:

- El SA toma como entrada la colección de figuras bocetadas y almacenadas en la Base de Datos.
- Cada una de dichas figuras tiene asociada una segmentación de referencia con la que se compararán los resultados obtenidos de la segmentación en cada iteración del SA.
- Los parámetros de la segmentación varían en cada iteración para intentar obtener su valor óptimo.
- Cuando se completa una iteración del SA, los resultados obtenidos de esa segmentación se comparan con los datos de referencia y se calcula la función de coste. Dicha función de coste se corresponderá con el porcentaje de figuras geométricas erróneamente segmentadas, de modo que la minimización del coste pasa por segmentar correctamente el mayor número posible de figuras.

Los valores óptimos obtenidos para cada parámetro de la segmentación se muestran en la Tabla 6.2.

Tabla 6.2. Parámetros de la segmentación y su valor óptimo para el mejor resultado del SA

Parámetros para la segmentación	Valor
MAX_DIAGONAL_BBOX	500
INTERSPACING DISTANCE	2.9442
FILTER_WINDOW	8
DIRECTION_WINDOW	2
CURVATURE_WINDOW	1
MAX_CORNER_RADIUS	60
CORNER_RADIUS	22
NEIGHBORING_WINDOW	17
MIN_RADIUS_RATIO	3.1616
MAX_DISTANCE	5
MAX_ARC_RADIUS	314.1374
MIN_ARC_ANGLE	47º
MIN_LINE_LENGTH	37

La Figura 6.7 y la Figura 6.8 muestran algunos ejemplos de figuras bocetadas utilizadas en los test realizados para el entrenamiento del sistema de segmentación (optimización de parámetros).

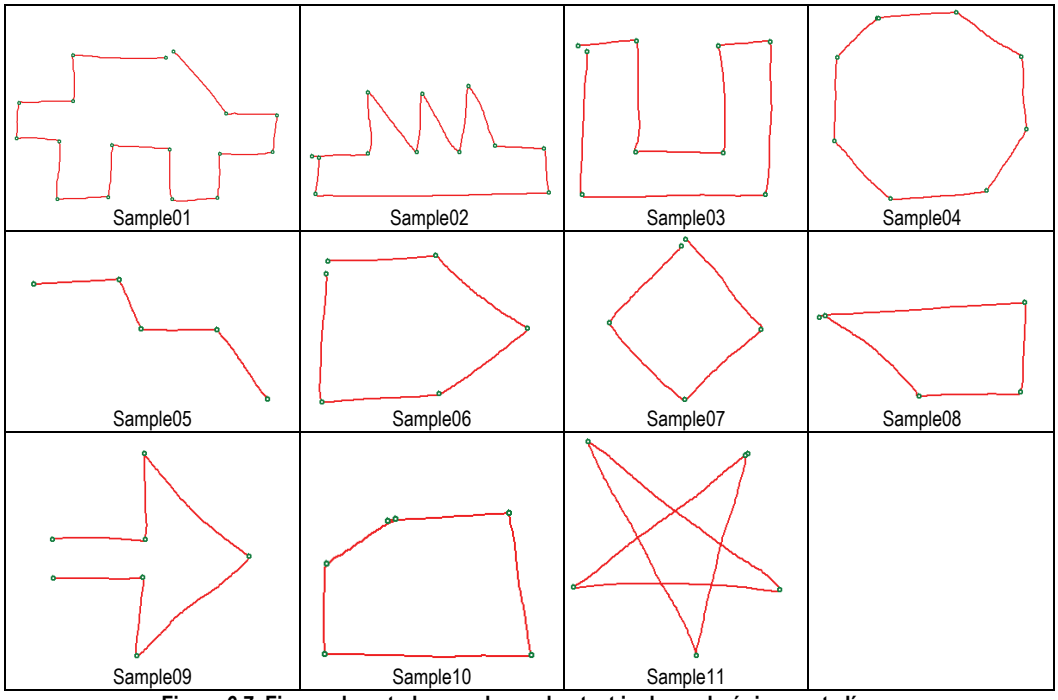


Figura 6.7. Figuras bocetadas usadas en los test incluyendo únicamente líneas

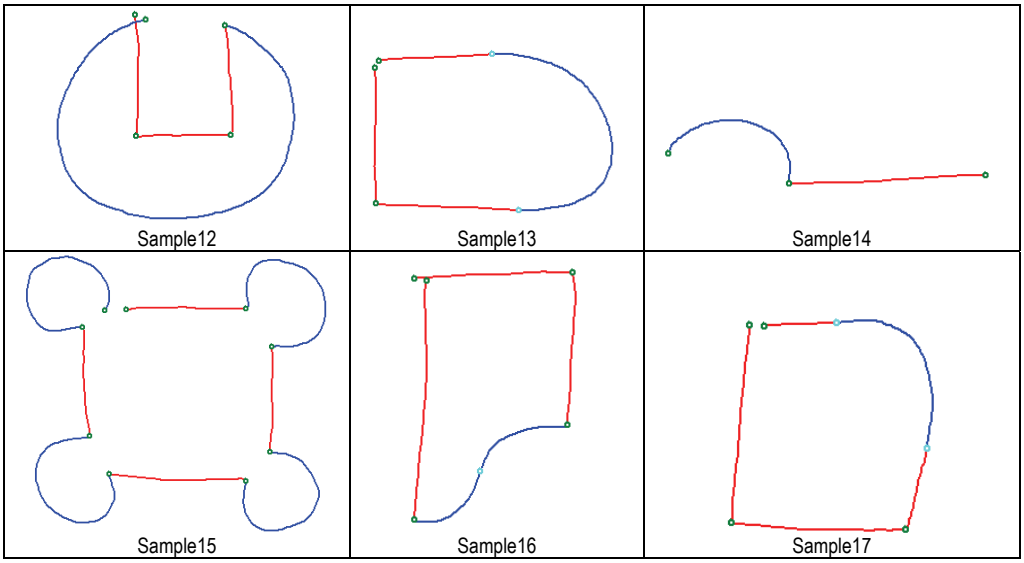


Figura 6.8. Figuras bocetadas usadas en los test incluyendo líneas (color rojo) y arcos (color azul)

Para considerar una figura segmentada correctamente, el número de vértices y las primitivas aproximadas entre ellos deben coincidir con la segmentación de referencia. En caso contrario, la función de coste se incrementa. Para calcular la función de coste, se ha tenido en cuenta el criterio de todo o nada (*all-or-nothing accuracy*). La Figura 6.9 muestra algunos ejemplos de figuras bocetadas segmentadas.

Los mejores resultados obtenidos durante los test corresponden a un valor de la función de coste de 1.9%, es decir, 13 errores frente a un total de 680 muestras (un porcentaje de éxito del 98.1% en la segmentación). Cabe destacar que sobre el rango inicial de valores de los parámetros fijados empíricamente, el porcentaje de aciertos en la segmentación aumentó del 76.6% al 98.1%.

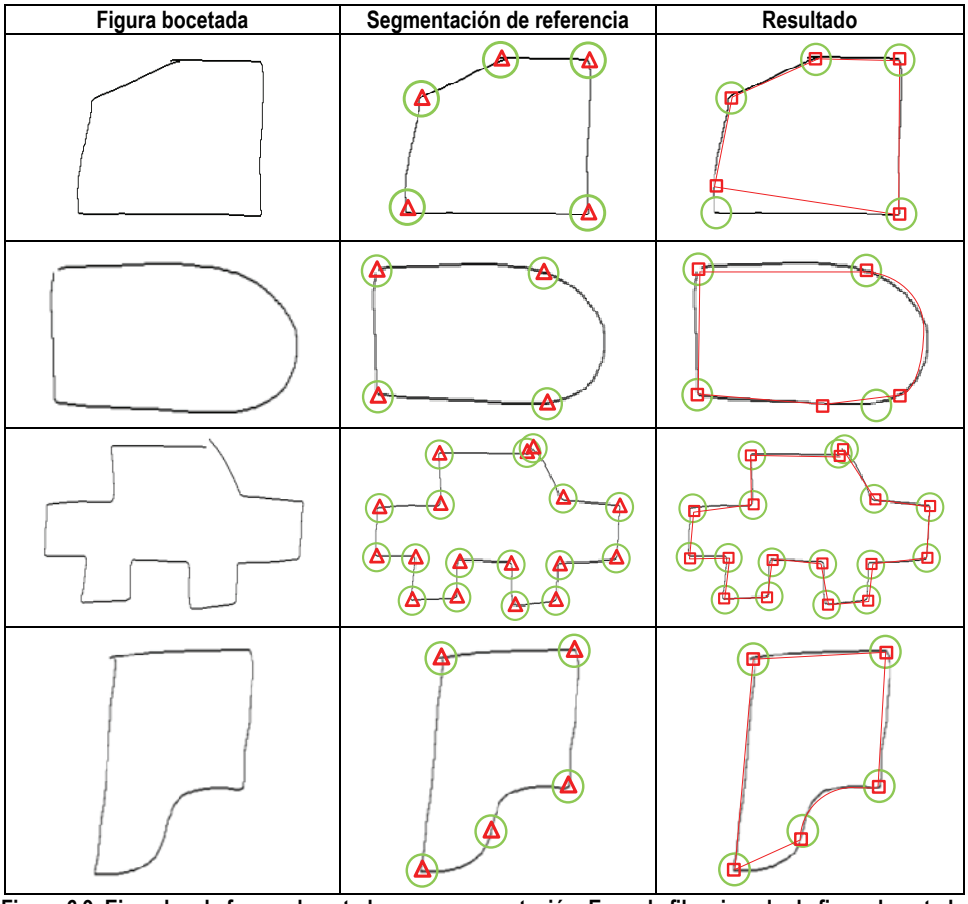


Figura 6.9. Ejemplos de formas bocetadas y su segmentación. En cada fila: ejemplo de figura bocetada; segmentación de referencia (△) con posible posición del vértice (○); resultado de la segmentación (□) con aproximación de primitivas (líneas rojas)

6.6 Conclusiones

De los resultados obtenidos, podemos concluir que las técnicas de optimización mejoran significativamente los resultados asociados a procesos de segmentación, demostrando que, en general, el proceso de reconocimiento se puede beneficiar considerablemente si se aplican procesos de optimización para el ajuste de los parámetros. Además, la aplicación de la optimización se muestra como una forma efectiva de automatizar los procesos de entrenamiento necesarios en este tipo de sistemas.

Otras conclusiones que se derivan de los test realizados con relación al ajuste de los parámetros genéricos del algoritmo de SA son las siguientes:

1. Los parámetros que se presentan como más importantes son aquellos que controlan el esquema de enfriamiento:

   a. Mecanismo de desplazamiento: es decir, la máxima proporción de desplazamiento sobre el rango que se puede realizar a un parámetro del problema debe ser un valor lo suficientemente alto como para poder salir de los mínimos locales, pero a su vez no demasiado elevado como para que no dé saltos a cualquier posición.
   b. Esquema de enfriamiento: si se desea que aumente la probabilidad de encontrar una solución óptima habrá que realizar un enfriamiento más lento, o tener una condición de equilibrio menos restrictiva para que realice más pruebas, aumentando, en consecuencia, el coste temporal.

2. Tanto para la condición de equilibrio como para la condición de enfriamiento, son muy útiles los valores clásicos indicados en la literatura:
   a. Condición de equilibrio: con valores menores en ambos parámetros se realiza un enfriamiento más rápido, obteniendo resultados antes, pero existiendo la posibilidad de dejarnos alguna solución favorable. Con valores mayores se aumenta el número de pruebas en cada temperatura y es más fácil obtener un buen resultado, pero a costa de emplear mucho más tiempo. Los valores aconsejados en las referencias clásicas de SA han demostrado ser una buena elección.
   b. Condición de congelación: con un coeficiente de aceptación (χ^{min}) mayor, se finaliza antes, y se pueden dejar buenas soluciones; con un coeficiente menor se pueden realizar muchas iteraciones sin encontrar nada que mejore lo existente. Las pruebas realizadas muestran que la variación de este parámetro no modifica en gran manera el momento de concluir el proceso. Nuevamente, los valores aconsejados en las referencias clásicas de SA reflejan ser una buena elección.

3. Como resumen, podemos concluir que:
   a. El SA es muy estable, es decir, se comporta bien (puede encontrar buenas soluciones) aunque se varíen sus parámetros dentro de un rango bastante amplio.
   b. El parámetro que se debe ajustar con mayor precisión es el máximo porcentaje de desplazamiento α, que regula el mecanismo de desplazamiento dentro del espacio de parámetros del problema a optimizar. Este parámetro debe ser lo suficiente grande para poder salir de un mínimo local, pero no demasiado para evitar saltos sin sentido (Figura 6.10).
   c. Todos los cambios que provocan que se realicen más desplazamientos suelen acabar encontrando mejores soluciones, a costa de aumentar el coste temporal. Existe un carácter aleatorio, por lo que en dos ejecuciones distintas del proceso, pueden variar los resultados. Por este motivo, también se considera que puede ser más efectivo (mejor resultado en menor tiempo) realizar el proceso varias veces con un esquema de enfriamiento rápido, que una sola vez con un enfriamiento lento.

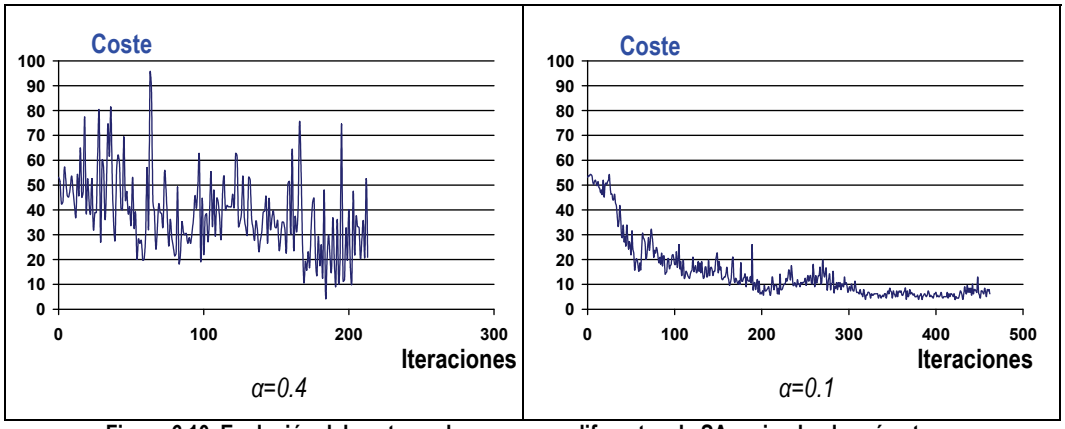


Figura 6.10. Evolución del coste en dos procesos diferentes de SA variando el parámetro *α*

Así pues, los parámetros genéricos del SA en orden de importancia son el máximo porcentaje de desplazamiento α y aquellos que controlan el esquema de enfriamiento: la temperatura inicial T_0 y el coeficiente de enfriamiento k. La probabilidad de encontrar una solución óptima aumenta cuando se realizan más test con un enfriamiento lento (mayor T_0, o valor de k cercano a 1), pero incrementándose el tiempo de convergencia del algoritmo. También pueden encontrarse buenas soluciones con un conjunto de procesos de SA con un esquema rápido de enfriamiento o con un único proceso SA con un esquema de enfriamiento lento.

Los test realizados muestran que valores típicos para los criterios de equilibrio y congelación (dependientes del número de parámetros a optimizar) son apropiados, y que cambios en el criterio de equilibrio tienen efectos similares que cambios en el esquema de enfriamiento, sin embargo, cambios en el criterio de congelación no afectan significativamente al tiempo de convergencia del SA o a su resultado.

Capítulo 7:

RESULTADOS Y DISCUSIÓN

7. RESULTADOS Y DISCUSIÓN

En este capítulo se muestran los resultados obtenidos de los desarrollos llevados a cabo en el campo de la segmentación y del interfaz caligráfico basado en agentes diseñado.

7.1 Resultados del método de segmentación TCVD (Tangent Corner Vertices Detection)

Para la evaluación del método de segmentación TCVD desarrollado en esta tesis y descrito en el capítulo 4, se ha tomado un conjunto de datos de 17 modelos de formas diferentes, consistente en 6 formas que incluyen curvas con características similares a las que aparecen en dibujos de ingeniería (Figura 7.1) y 11 formas incluyendo únicamente líneas rectas (definidas en [WEH08]. Ver Figura 7.2). Como se observa, en tres de ellas aparecen vértices tangentes (en color azul cian). Todas las formas son abiertas, considerando siempre como vértices esquina los puntos inicial y final del trazo.

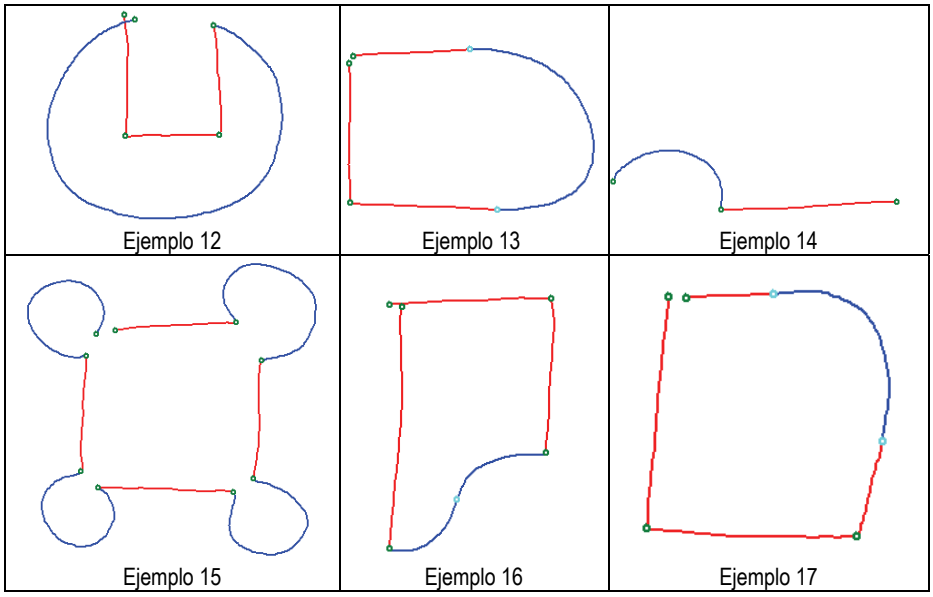


Figura 7.1. Trazos que contienen curvas: con vértices esquina (en color verde) y vértices tangentes (en color azul cian)

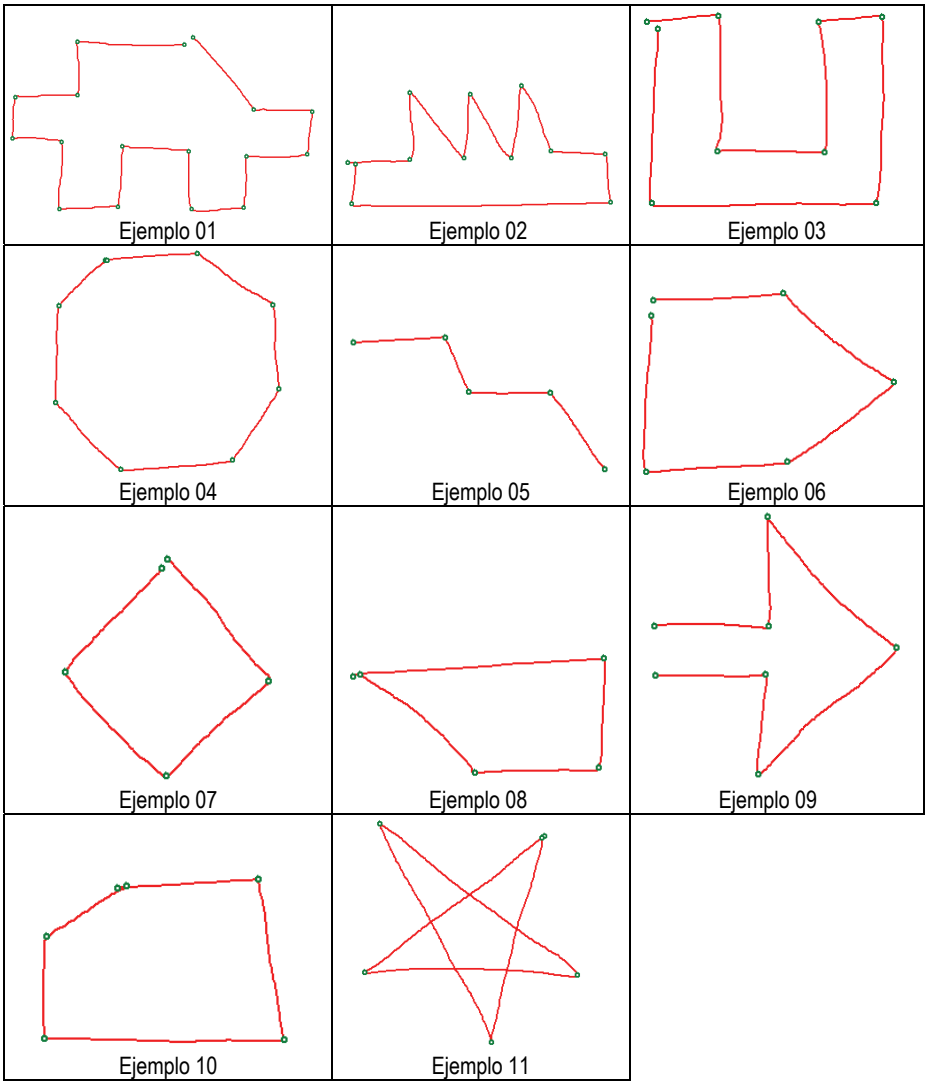


Figura 7.2. Modelos de formas que incluyen únicamente líneas rectas

Las formas se han recogido de 8 usuarios diferentes, y cada uno de ellos ha dibujado 6 veces dicha forma, dando un total de 816 muestras. Los usuarios no recibieron instrucción alguna sobre la precisión del dibujo, de modo que cada uno de ellos imitó la forma proporcionada de manera diferente.

De éstas, se han seleccionado un total de 136 (8 por modelo de forma) que han sido utilizadas para fijar los parámetros del TCVD utilizando el algoritmo Simulated Annealing descrito anteriormente en el capítulo 6, y las 680 restantes se han utilizado para el conjunto de prueba.

Para poder comprobar resultados se han implementado los métodos ShortStraw e IStraw descritos en el capítulo 4. Durante su implementación se observó que el método IStraw presentaba ciertos inconvenientes sobre las formas del conjunto de prueba, por lo que se tuvo que realizar alguna corrección para que alcanzase los mejores resultados, siendo la corrección más importante la de no utilizar la información de velocidad debido a que el número de errores aumenta cuando el conjunto de esquinas inicial está compuesto por *straws* junto con la información de velocidad.

Los resultados obtenidos de los test de prueba pueden verse en la Tabla 7.1 y en la Tabla 7.2 para las 440 formas sin curvas, y en la Tabla 7.3, Tabla 7.4 y Tabla 7.5 para las 680 formas del conjunto de prueba (con y sin curvas). Los resultados están expresados en las mismas medidas en que lo hacen los métodos comparados ShortStraw e IStraw: "*Correct Corners Accuracy*" (o acierto de esquinas correctas) y "*All-or-Nothing Accuracy*" (o acierto en toda la forma o nada). En el primer caso, el acierto se calcula dividiendo el número correcto de vértices detectados por el número total de vértices. En el segundo caso, el acierto es igual al número de formas correctamente segmentadas (sin falsos positivos o negativos) dividido por el número total de formas. La medida "*All-or-nothing Accuracy*" es la más significativa ya que toma en cuenta tanto los falsos negativos como los falsos positivos. Para poder comprobar aisladamente la importancia de los falsos positivos se ha añadido la medida "*False Positive Rate*" que es igual al número de falsos positivos dividido por el número total de vértices. En este caso, los valores más bajos corresponden a los mejores resultados.

Tabla 7.1. Precisión en % del resultado para las 440 formas sin curvas o vértices tangentes

	ShortStraw	IStraw	TCVD
Falsos Positivos	31	3	0
Falsos Negativos	23	62	3
Esquinas Correctas	3457	3418	3477
Esquinas Totales	3480	3480	3480
Correct Corners Accuracy	99.3%	98.2%	99.9%
False Positive Rate	0.9%	0.1%	0.0%

Tabla 7.2. Precisión en % de todo o nada para las 440 formas sin curvas o vértices tangentes

	ShortStraw	IStraw	TCVD
All-or-Nothing Accuracy	89.8%	91.4%	99.3%

Tabla 7.3. Precisión en % de los resultados para las 680 formas incluyendo curvas (solo vértices esquina)

	ShortStraw	IStraw	TCVD
Falsos Positivos	1525	400	12
Falsos Negativos	24	125	6
Esquinas Correctas	4616	4515	4634
Esquinas Totales	4640	4640	4640
Correct Corners Accuracy	99.5%	97.3%	99.9%
False Positive Rate	32.9%	8.6%	0.3%

Tabla 7.4. Precisión en % de los resultados para las 680 formas incluyendo curvas (solo vértices tangentes)

	ShortStraw	IStraw	TCVD
Falsos Positivos	---	---	4
Falsos Negativos	170	180	16
Vértices Tangente Correctos	30	20	184
Vértices Tangente Totales	200	200	200
Correct Tangent Vertices Accuracy	15.0%	10.0%	92.0%
False Positive Rate	---	---	2%

Tabla 7.5. Precisión en % de todo o nada para las 680 formas incluyendo esquinas y vértices tangentes

	ShortStraw	IStraw	TCVD
All-or-Nothing Accuracy	58.1%	63.5%	96.6%

A raíz de estos resultados, vemos que para la detección de vértices esquina en formas compuestas únicamente por poli-líneas, el método TCVD obtiene una mayor precisión de todo o nada (99%) sobre ambos métodos ShortStraw e IStraw, que ya de por sí obtienen buenos resultados (en torno al 90%). ShortStraw no es un método diseñado para tratar con formas curvas, y obviamente la precisión baja mucho para estos casos ya que aparecen muchos falsos positivos. El método IStraw sí que está preparado para tratar con formas que contienen curvas, y de hecho elimina muchos falsos positivos detectados en el ShortStraw. Pero también elimina algunas esquinas correctas, lo que hace que el resultado final no mejore mucho sobre el ShortStraw, mientras que el TCVD da valores muy altos (en torno al 96%), en gran medida debido al bajo número de falsos positivos.

El TCVD encuentra la mayoría de vértices tangentes (92%) incluyendo unos pocos vértices tangentes falsos (falsos positivos). Ni el ShortStraw ni el IStraw buscan vértices tangentes, aunque sí encuentran algunos vértices esquina falsos (falsos positivos) cerca de la posición de vértices tangentes, ya que la tolerancia de posición debe ser alta. Esto lo afirman Company et al. en su trabajo [CVP09], donde incluso las personas encuentran dificultad para establecer la posición de los vértices tangentes en bocetos a mano.

La mayoría de fallos en la segmentación encontrados en TCVD son debidos a que los usuarios dibujaron libremente sin recibir ninguna instrucción de actuación sobre la imitación de los modelos de forma proporcionados y precisión de los mismos. Así pues, algunas formas fueron esbozadas con una calidad pobre, resultando fácil confundir vértices esquina con arcos de radio pequeño (Figura 7.3 fila superior), y rectas con arcos de radio grande (Figura 7.3 fila inferior). Para la columna de “Segmentación” las líneas rectas aproximadas se representan en color rojo y los arcos en color azul.

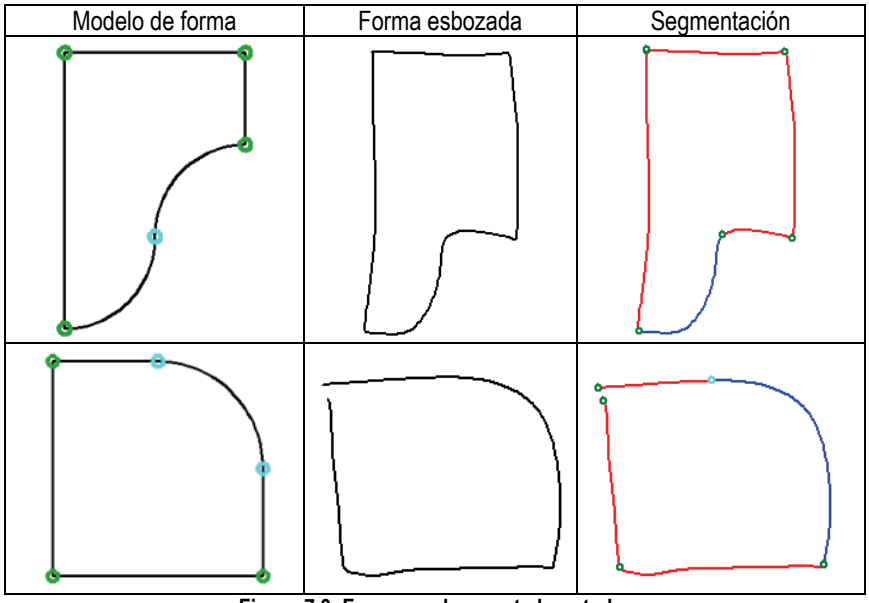


Figura 7.3. Formas pobremente bocetadas

El método TCVD presenta dos ventajas más sobre los métodos comparados ShortStraw e IStraw en la detección de esquinas. En primer lugar, mientras que ShortStraw e IStraw realizan un post-proceso intensivo sobre el conjunto de vértices detectados (para evitar fundamentalmente los falsos positivos y añadir falsos negativos al conjunto), TCVD se centra en la obtención de un buen conjunto inicial de vértices esquina a partir de la función del radio de curvatura sin aplicar ningún refinamiento posterior o post-proceso. En segundo lugar, TCVD mantiene una precisión superior de vértices esquina ya que la distancia de interespaciado utilizada en el remuestreo de los puntos del trazo es menor, y los puntos filtrados únicamente se usan para el cálculo de la función del radio de curvatura.

Respecto al método de Pu y Gur [PG09], no se ha realizado una implementación debido a su mayor complejidad, y por tanto la comparación a realizar es más relativa: por una parte utilizan figuras distintas, algunas de ellas con mayor complejidad y con proporción distinta de vértices esquina y tangentes; pero por otra también prueban con figuras realizadas mediante software CAD que están perfectamente dibujadas, y con un conjunto muy pequeño de figuras dibujadas a mano (30). Los resultados de esta comparación "relativa" pueden verse en la Tabla 7.6. Los resultados están expresados en las medidas utilizadas anteriormente: "*Correct Vertices Accuracy*" (o acierto de vértices correctos, incluyendo tanto vértices esquina como tangentes), y "*False Positive Rate*" (o porcentaje de vértices incorrectos sobre el total existente). Sin embargo, Pu y Gur no proporcionan la medida "*All-or-nothing Accuracy*" que, como se ha indicado anteriormente, es la más significativa.

Tabla 7.6. Precisión en % de vértices (esquina + tangente) correctos e incorrectos

	Pu y Gur (CAD)	Pu y Gur (Hand-drawn)	TCVD
Correct Vertices Accuracy	99.2%	97.8%	99.5%
False Positive Rate	24.5%	24.5%	0.3%

Según los valores mostrados en la tabla anterior, los porcentajes de vértices correctos encontrados son muy similares, pero TCVD presenta unos resultados claramente mejores en cuanto a falsos positivos. Este elevado número de falsos positivos, se dejará sentir negativamente en la medida "*All-or-nothing Accuracy*". Además, Pu y Gur no distinguen entre vértices esquina y tangentes (lo cual resulta imprescindible), motivado porque tampoco realizan distinción entre rectas y curvas. Resulta curioso que incluso empleando las figuras realizadas mediante software CAD el número de falsos positivos sea tan elevado como con las figuras realizadas a mano, algo que puede dar a entender que este método es poco robusto (al contrario de lo que indican sus autores). Por último, remarcar que Pu y Gur también realizan un uso intensivo de refinamientos posteriores.

7.2 Resultados del interfaz caligráfico basado en agentes

La evaluación del paradigma propuesto se ha realizado en dos fases. En la primera fase se ha evaluado el reconocimiento llevado a cabo por los Agentes Primitivos, y en la segunda fase se ha evaluado el resultado del reconocimiento final en el que intervienen los Agentes Combinados y el Agente Inteligente INA.

En los test realizados para la primera fase de evaluación han intervenido 10 usuarios de aplicaciones CAD, introduciendo cada uno de ellos varias ocurrencias de todos los símbolos primitivos del alfabeto. En total se han evaluado unos 2200 símbolos primitivos con diferentes orientaciones y tamaños. Además, el sistema se ha entrenado off-line usando parte de estos símbolos, lo que ha permitido el cálculo de unas funciones de clasificación utilizadas posteriormente en el proceso on-line y que han servido como *cues* para el reconocimiento realizado por los Agentes Primitivos.

Los resultados obtenidos de este reconocimiento se muestran en la Tabla 7.7, donde los datos que aparecen enmarcados en negrita en la diagonal de la matriz de confusión se corresponden con el porcentaje de los datos de acierto, apareciendo en el resto de celdas las clasificaciones fallidas para cada símbolo esbozado y su correspondiente porcentaje de confusión.

Tabla 7.7. Resultados del reconocimiento efectuado por los Agentes Primitivos (valores en %)

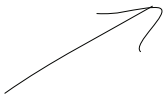
Símbolos reconocidos / Símbolos esbozados	Ángulo	Arco	Círculo	Línea	Flecha	Flecha redonda	Punto	Tachón	Polígono
Ángulo	98.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4
Arco	0.0	94.9	0.0	2.5	0.0	0.0	2.5	0.0	0.0
Círculo	0.0	7.0	93.0	0.0	0.0	0.0	0.0	0.0	0.0
Línea	0.0	0.0	0.0	99.3	0.0	0.0	0.7	0.0	0.0
Flecha	0.0	0.0	0.0	0.0	94.2	5.8	0.0	0.0	0.0
Flecha redonda	0.0	0.0	0.0	0.0	5.7	94.3	0.0	0.0	2.8
Punto	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
Tachón	0.0	0.0	0.0	1.8	0.0	0.0	0.0	95.4	2.8
Polígono	0.0	0.0	2.1	0.0	0.0	0.0	0.0	0.0	97.9

A partir de estos resultados, el promedio final del reconocimiento llevado a cabo por los Agentes Primitivos ha resultado en un porcentaje de éxito del 96.41%.

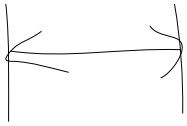
Los errores principales en la clasificación se han producido cuando el símbolo se ha bocetado de forma muy imprecisa, y también en casos particulares como, por ejemplo, un círculo que se confunde con un arco cuando éste no está claramente cerrado; o el caso de un arco que posee un radio tan grande que se aproxima a una línea; o cuando una flecha o flecha-redonda se confunden entre sí dependiendo de si el tramo largo principal del trazo es más o menos curvo, independientemente de la intención que tuviera el usuario.

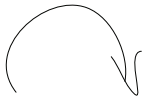
Los test realizados para la segunda fase de evaluación, o evaluación final del interfaz, fueron llevados a cabo con 9 usuarios de aplicaciones CAD. Cada usuario introdujo varias ocurrencias de cada una de las diferentes clases de símbolos combinados. Esto símbolos fueron almacenados hasta un total de 2500 con diferentes orientaciones y tamaños. Una vez introducidos, el interfaz los clasificó según los resultados que muestra la matriz de confusión de la Tabla 7.8.

Para la evaluación del interfaz, se permitió a los usuarios casi cualquier entrada. Esto se refleja en las siguientes figuras, donde se puede apreciar que los símbolos combinados aceptados no dependen de la secuencia de los trazos y de la forma de los mismos (por ejemplo una punta de flecha se puede hacer también utilizando 2 trazos rectos).

Símbolo que indica cota radial / comando de extrusión	Generación del gesto con los símbolos primitivos
	Unión de tres símbolos primitivos de línea Unión de un símbolo primitivo de línea y otro de ángulo Único símbolo primitivo de flecha

Símbolo que indica cota diametral	Generación del gesto con los símbolos primitivos
	Unión de cinco símbolos primitivos de línea Unión de tres símbolos primitivos de línea y un ángulo Unión de un símbolo primitivo de línea y dos ángulos Unión de dos símbolos primitivos de línea y un símbolo primitivo de flecha Unión de un símbolo primitivo de ángulo y un símbolo primitivo de flecha

Símbolo que indica cota lineal	Generación del gesto con los símbolos primitivos
	Unión de siete símbolos primitivos de línea Unión de cinco símbolos primitivos de línea y un ángulo Unión de tres símbolos primitivo de línea y dos ángulos Unión de cuatro símbolos primitivos de línea y un símbolo primitivo de flecha Unión de dos primitivas de línea, un ángulo y un símbolo primitivo de flecha

Símbolo que indica comando de revolución	Generación del gesto con los símbolos primitivos
	Unión de un símbolo primitivos de arco y dos líneas Unión de un símbolo primitivo de arco y otro de ángulo Único símbolo primitivo de flecha redonda

Símbolo que indica eje	Generación del gesto con los símbolos primitivos
	Unión de varios símbolos primitivos de línea y punto alternados y en número variable

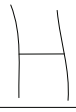
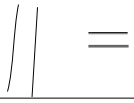
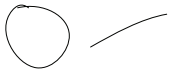
Símbolo que indica horizontalidad	Generación del gesto con los símbolos primitivos
	Unión de tres símbolos primitivos de línea

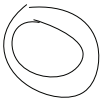
Figura 7.4. Formas posibles de bocetar algunos símbolos combinados

Símbolo que indica verticalidad	Generación del gesto con los símbolos primitivos
	Unión de dos símbolos primitivos de línea  Único símbolo primitivo de ángulo 

Símbolo que indica paralelismo / igualdad	Generación del gesto con los símbolos primitivos
	Unión de dos símbolos primitivos de línea 

Símbolo que indica perpendicular	Generación del gesto con los símbolos primitivos
	Unión de dos símbolos primitivos de línea 

Símbolo que indica tangencia	Generación del gesto con los símbolos primitivos
	Unión de un símbolo primitivo de línea y un círculo 

Símbolo que indica concentricidad	Generación del gesto con los símbolos primitivos
	Unión de dos símbolos primitivos de círculo 

Símbolo que indica borrado	Generación del gesto con los símbolos primitivos
	Un símbolo primitivo de tachón 

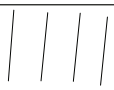
Símbolo que indica sección	Generación del gesto con los símbolos primitivos
	Unión de más de dos primitivos de línea 

Figura 7.5. Formas posibles de bocetar algunos símbolos combinados

Tabla 7.8. Resultados del reconocimiento efectuado por el interfaz (valores en %)

Gestos esbozados \ Gestos Reconoc.	Módulo de Restricciones Geométricas								Módulo de Modelado				Módulo de Acotación			Módulo de Dibujo Artístico		Núcleo Central	
	Concéntrico	Tangente	Vertical	Horizontal	Paralela	Perpendicular	Igualdad	Sección	Eje	Revolución	Extrusión / Cota Radial	Cota Lineal	Cota Diametral	Línea artística	Arco artístico	Geometría	Tachón
Concéntrico	96.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.30	0.00
Tangente	0.00	99.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
Vertical	0.00	0.00	98.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
Horizontal	0.00	0.00	0.00	96.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.30	0.00
Paralela	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Perpendicular	0.00	0.00	0.00	0.00	0.00	97.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	0.00
Igualdad	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sección	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Eje	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	93.00	0.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00
Revolución	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	98.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
Extrusión / Cota Radial	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	99.00	0.00	0.00	0.00	0.00	1.00	0.00
Cota Lineal	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	90.00	0.00	0.00	0.00	10.00	0.00
Cota Diametral	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	97.00	0.00	0.00	3.00	0.00
Línea Artística	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	95.00	0.00	5.00	0.00
Arco Artístico	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	90.00	10.00	0.00
Geometría	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
Tachón	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	96.00

Los resultados de la clasificación se muestran en la Tabla 7.8, donde los ratios de éxito en la clasificación para cada clase se sitúan en las celdas de la diagonal de la tabla (resaltadas en negrita), y el porcentaje de errores en el resto de celdas. El porcentaje medio de éxito alcanzado con el reconocedor propuesto es del 96.8%.

Como se puede apreciar en la tabla anterior, el empleo del paradigma propuesto posibilita una clasificación correcta de los símbolos bocetados, salvo en aquellos casos en los que el símbolo combinado no es reconocido como palabra del diccionario, en cuyo caso la forma esbozada se clasifica como geometría, lo que a veces no coincide con la intención del usuario, que debe borrarla y esbozar mejor lo que pretendía en un principio. En cualquier caso, y gracias a las *cues* extraídas por el Agente de Cálculo de Características y por los Agentes Primitivos, y gracias a la información del contexto que realizan los agentes combinados, el tratamiento del indeseado "no gesto" (cuando el gesto se acepta siempre como perteneciente al diccionario) imposible de discriminar en muchos reconocedores, se ha resuelto.

Capítulo 8:
CONCLUSIONES

8. CONCLUSIONES

Las conclusiones más relevantes que se extraen de esta tesis son las siguientes:

- Se ha diseñado un paradigma de reconocimiento de bocetos fiable y robusto que puede dar soporte a las primeras fases del diseño, y como consecuencia del mismo se ha desarrollado un interfaz natural (que permite la generación espontánea de ideas en papel), adaptativo (que permite al usuario definir geometría imprecisa e incompleta) y transparente (que el usuario puede dibujar una línea a trazos o un rayado de un área sin anunciar dicha intención a la aplicación por medio de algún menú).
- En la implementación del interfaz natural se han incorporado varias técnicas de bocetado:
  - *Interspersing*: interrupción de la actividad que realiza el usuario para realizar una acción concreta y proseguir luego en el punto en que se quedó. Esta técnica se ha implementado en una primera fase, estando limitado su valor a un único trazo, esto es, permite interrumpir el gesto que se esté realizando para introducir un trazo ajeno al gesto y continuar enseguida con el que se estaba realizando. Dado que esta técnica ofrece grandes posibilidades se propone como trabajo futuro a esta tesis su implementación sin ningún tipo de limitación.
  - *Overtracing*: creación de geometría mediante trazos superpuestos imitando el bocetado "artístico" en papel. Para la implementación de esta técnica se ha recurrido a la creación de un módulo artístico que permita dibujar líneas y arcos mediante varios trazos superpuestos.
  - Cambio de modo automático: no es necesario atender a menús para cambiar del modo "geometría" al modo "comando" o al modo "introducción de restricciones u otros símbolos del boceto", reconociéndose automáticamente la intención del usuario.
  - No-Gesto: en el caso de que los trazos introducidos no puedan conformar ningún símbolo del diccionario, estos se considerarán geometría o 'no gesto', evitando de esta manera los falsos positivos en el reconocimiento.
- Se ha comprobado que los sistemas basados en agentes son válidos para aquellas aplicaciones que necesitan de reglas de decisión guiadas por el conocimiento, y para este tipo de aplicaciones en particular, donde es importante la información disponible del contexto para tomar decisiones que guíen el reconocimiento. Esto ha sido posible gracias al empleo de técnicas de razonamiento similares a las utilizadas por los sistemas expertos que permiten cambiar las decisiones en función de diferentes posibilidades, permitiendo que el

usuario pueda dibujar con total libertad sin importar lo que dibuje, la acción que desee realizar, el número de trazos o la secuencia de introducción de los mismos.

- Se ha desarrollado e implementado un nuevo método de segmentación del boceto en el que se detectan los vértices esquina y tangentes, con un porcentaje de acierto cercano al 100%. Este método mejora todos los algoritmos de segmentación existentes hasta la fecha:
  - Mejora los resultados de detección de vértices esquina, obteniendo un 99.9% de aciertos frente al 99.3% del mejor de los existentes.
  - Aparte del método de segmentación propuesto, únicamente existe otro trabajo que detecte los vértices tangentes, con un porcentaje de falsos positivos en la detección de vértices tangentes del 24.5%. El método de segmentación implementado en esta tesis tan sólo posee un 0.3% de falsos positivos en vértices tangentes.
- Además, se ha recurrido al empleo de un marco para la optimización que permite el entrenamiento automatizado del sistema, que en la mayoría de aplicaciones que necesitan de este entrenamiento previo se lleva a cabo manualmente, consumiendo mucho tiempo y ofreciendo pobres resultados.

  Este marco de optimización ha permitido el entrenamiento, la configuración y el ajuste inicial de los parámetros del sistema de forma automatizada para que éste alcance la mejor solución posible.
- También cabe destacar que el interfaz diseñado es fácilmente extensible, dotándose de un carácter modular, que permitirá posteriormente añadir nuevos gestos/símbolos al diccionario con el mínimo intrusismo sobre el sistema diseñado.

En definitiva, esta tesis desarrolla los algoritmos, métodos y herramientas necesarios para proporcionar un entorno de diseño paramétrico basado en bocetos con la posibilidad de introducir la geometría y controlar la parametrización de forma directa, manuscrita y sin interferencia de menús, proporcionando más potencialidad que las herramientas tradicionales de lápiz y papel gracias a que, además de proveer su misma funcionalidad, aporta otras ventajas obvias como la reutilización de los bocetos para fases posteriores a la del diseño conceptual.

En un futuro inmediato se pretende integrar el interfaz caligráfico diseñado en una aplicación de modelado basada en bocetos 2D paramétricos.

BIBLIOGRAFÍA

9. BIBLIOGRAFÍA

[ABG02] Ablameyko, A., Bereishik, V., Gorelik, A., Medvedev, S.: “Reconstruction of 3D object models from vectorised engineering drawings”. Pattern and Analysis & Applications 5:2, 2002, pp. 2-14.

[ACD06] Azar, S., Couvreury, L., Delfosse, V., Jaspartz, B., Boulanger, C.: “An agent-based multimodal interface for sketch interpretation”. In Proceedings of International Workshop on Multimedia Signal Processing (MMSP-06), British Columbia, Canada, 2006.

[AD05] Allvarado, C. and Davis, R.: “Sketchread: A multi-domain sketch recognition engine”. Proceedings of the 17th annual ACM symposium on User interface software and technology, vol. 29, 2005, pp. 518–532.

[AE05] Alhajj, R., Elnagar, A.: “Multiagents to separating handwritten connected digits”. IEEE Transactions On Systems Man and Cybernetics Part A-Systems and Humans, 35(5), 2005, pp. 593-602.

[AE07] Ahmed, M.N., Eid, A.,: “Parameter Optimization for Content-based Image Enhancement”. In Proc. of the 23rd International Conference on Digital Printing Technologies and Digital Fabrication 2007 (NIP23), Alaska, pp. 183-188;

[AGB04] Alexe, A, Gaildrat, V, Barthe, L.: “Interactive modelling from sketches using spherical implicit functions”. Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, Afrigraph 2004, New York: ACM Press, 2004, p. 25-34.

[AJ02] Achten, H.H., Jessurun, A.J.: “An agent framework for recognition of graphic units in drawings”. Proceedings of 20th International Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe’02), Warsaw, 2002, pp. 246–253.

[Alb06] Albert Gil, Francisco: “Análisis de motivos decorativos de tejidos y revestimientos cerámicos en el entorno de la visión artificial. Aplicación a la reconstrucción de motivos históricos y al diseño”. Tesis doctoral, Valencia, 1/12/2006.

[ANC07] Aleixos, N., Naya, F., Contero, M., Jorge, J., Varley, P. and Company, P.: "Interpreting annotated engineering drawings". Invited Workshop on Pen Computing. Brown University, 2007.

[AVK93] Apte, A., Vo, V., Kimura, T.D.: “Recognising Multistroke Geometric Shapes: An Experimental Evaluation”. Proceedings of the ACM (UIST’93), Atlanta, Georgia, (1993) 121-128.

[BAM03] Blasco, J., Aleixos, N., Moltó, E.: “Machine vision system for automatic quality grading of fruit”. Biosystems Engineering 85 (4) (2003), 415-423.

[Bar01] Barberá, H.M.:"A Distributed Architecture for Intelligent Control in Autonomous Mobile Robots". Masters thesis, Dept. of Communications and Information Engineering, University of Murcia, 2001

[Bar04] Barr, R.E.: “The Current Status of Graphical Communication in Engineering Education”. 34th ASEE/IEEE Frontiers in Education Conference. October 20–23, 2004, Savannah, GA. pp S1D8-13.

[BBS08] Bae, S.H., Balakrishnan, R. & Singh, K.: "ILoveSketch: As-Natural-As-Possible Sketching System for Creating 3D Curve Models". ACM Symposium on User Interface Software and Technology 2008, Monterey (USA)

[BCG07] Bellifemine, F., Caire, G., Greenwood, D.: “Developing multi-agent systems with JADE”. John Wiley & Sons (2007).

[BF85] Burden, R. L., Faires, J. D.: “Análisis numérico”. Grupo Editorial Iberoamérica, 1985.

[BOK00] Berg, M., Overmars, M., Kreveld, M., Schwarzkopf, O.: “Computational geometry: algorithms and applications”. Springer, 2000.

[BPR00] Bellifemine, F., Poggi, A., Rimassa, G.: ”Developing multi-agent systems with JADE”. In Seventh International Workshop Agent Theories Architectures and Languages (ATAL2000), Lecture Notes in Computer Science, vol. 1986, Springer, Berlin, 2000, pp. 89–103.

[BRQ05] Bryll, R., Rose, R.T., Quek, F.: “Agent-based gesture tracking”. IEEE Transactions On Systems Man And Cybernetics Part A-Systems And Humans, 35(6), 2005, pp. 795-810.

[BT07] Brazil, M., Thomas, D.A.: “Network optimization for the design of underground mines”. Wiley Periodicals, Inc. NETWORKS, 2007, Vol. 49(1), pp. 40-50

[BZ98] Bloomenthal, K.; Zeleznik, R.C. et al.: “SKETCH-N-MAKE: Automated Machining of CAD Sketches”. Proceedings of ASME DETC'98, pp. 1-11, 1998.

[CAN08] Company, P., Aleixos, N., Naya, F., Varley, P.A.C., Contero, M. and Fernandez-Pacheco, D.G.: "A new Sketch-Based Computer Aided Engineering Pre-Processor". Proceedings of the Sixth International Conference on Engineering Computational Technology, 2008, Paper-149

[CBK05] Chen, G.Y., Bui, T.D., Krzyzak, A.: “Rotation invariant pattern recognition using ridgelets, wavelet cycle-spinning and Fourier features”. Pattern Recognition 38 (2005) 2314-2322.

[CCC04] Company, P., Contero, M., Conesa, J., Piquer, A. : “An optimisation-based reconstruction engine for 3D modelling by sketching”. Computers & Graphics. Vol 28, No 6. pp. 955-979.

[CCN06] Company, P., Contero, M., Naya, F. and Aleixos, N.: "A Study of Usability of Sketching Tools Aimed at Supporting Prescriptive Sketches". Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2006, pp. 139-146

[CCP04] Company, P., Contero, M., Piquer, A., Aleixos, N., Conesa, J. and Naya, F.: "Educational Software for Teaching Drawing-Based Conceptual Design Skills". Computer Applicacions in Engineering Education, vol 12 (4), 2004, pp. 257-268

[CCV09] Company, P., Contero, M, Varley, P.A.C., Aleixos, N. and Naya, F.: “Computer-Aided Sketching as a Tool to Promote Innovation in the New Product Development Process”. Computers in Industry. 60 (8), 2009, pp. 592–603

[CDM08] Casella, G., Deufemia, V., Mascardi, V., Costagliola, G., Martelli, M.: “An agent-based framework for sketched symbol interpretation”. Journal of Visual Languages and Computing, 19 (2008), pp. 225–257.

[Che05] Cheng, Y.: “Mean Shift, Mode Seeking, and Clustering". IEEE Transactions on Pattern Analysis and Machine Intelligence (2005), Vol. 17(8), pp. 790-799.

[CNC05] Contero, M., Naya, F., Company, P. Saorin, J.L., Conesa, J.: “Improving visualization skills in engineering education”. IEEE Computer Graphics and Applications. Vol. 25, No 5. pp. 24-31.

[CNJ03] Contero, M., Naya, F., Jorge, J., Conesa, J.: “CIGRO: A Minimal Instruction Set Calligraphic Interface for Sketch-Based Modeling”. Lecture Notes in Computer Science. Volume 2669, pp 549-558.

[Con90] Connolly, D.T.: “An improved annealing scheme for the QAP”. European Journal of Operational Research, 1990, vol. 46, pp. 93-100.

[CPC04] Company P., Piquer A. and Contero M.: “On the evolution of geometrical reconstruction as a core technology to sketch-based modelling”. Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM04), 2004, pp. 97-106.

[CPC05] Company, P., Piquer, A., Contero, M., Naya, F.: “A Survey on Geometrical Reconstruction as a Core Technology to Sketch-Based Modeling”. Computers & Graphics. Vol. 29 (6), 2005, pp. 892-904.

[Cre08] Crevier, D.: “Image segmentation algorithm development using ground truth image data sets”. Computer Vision and Image Understanding, 2008, Vol. 112 (2), pp. 143-159

[CS06] Chetty, G., Sharma, D.: “Distributed face recognition: A multiagent approach”. In Proc. of Knowledge-Based Intelligent Information And Engineering Systems, Pt 3, vol. 4253, 2006, pp. 1168-1175

[CV09] Company, P. and Varley, P.A.C.: "Operating modes in actual versus virtual paper-and-pencil design scenarios". Intelligent User Interfaces (IUI) Workshop on Sketch Recognition, 2009.

[CVP09] Company, P., Varley, P.A.C., Piquer, A., Vergara, M., Sanchez-Rubio, J.: “Benchmarks for Computer-based Segmentation of Sketches”. Pre-Proceedings of the Eighth IAPR International Workshop on Graphics Recognition GREC 2009, pp. 103-114.

[Dam05] Consorcio Dammad (2005): ”Diseño y aplicación de modelos multiagente para la ayuda a la decisión”. Universidad Rey Juan Carlos, Servicio de Publicaciones, Madrid.

[Dau04] Daum, Berthold : “Eclipse 3 para desarrolladores Java”. Anaya Multimedia (2004).

[DBM77] Dudani, S.A.; Breeding, K.J.; Mcghee, R.B.: “Aircraft identification by moment invariants”. IEEE Trans. Comput. (1977) C-26: 39–46

[DCL08] Davis, R.C., Colwell, B. and Landay, J.A: “K-Sketch: A Kinetic Sketch Pad for Novice Animators”. In Proc. of the 26th Annual CHI Conference on Human Factors in Computing Systems (CHI 2008), Italy. pp 413-422

[DG04] Dimri, J., Gurumoorthy, B.: “Handling sectional views in volumen-based approach to automatically construct 3D solids from 2D views”. Computer Aided Design vol. 37, pp. 485-495.

[Dow94] Dowsland, K.A.: “Variants of simulated annealing for practical problem solving”. Adaptative Computing and Information Processing, 1994, London Vol. 1, Unicom

[Egl90] Eglese, R.W.: “Simulated annealing: A tool for operational research”. European Journal of Operational Research, 1990, vol. 46, pp. 271-281

[FAC09] Fernández-Pacheco, D. G., Aleixos, N., Conesa, J., Contero, M.: “Natural interface for sketch recognition". Advances in Intelligent and Soft Computing, Springer, 2009, vol 55, pp 510-519

[Far93] Farin, G.: “Curves and surfaces for CAGD. A practical guide”. Academic Press Ltd., San Diego, 1993

[FCA09] Fernández-Pacheco, D. G., Conesa, J., Aleixos, N.:" Reconocimiento de Bocetos mediante Agentes ". Congreso Internacional Conjunto XXI Ingegraf – XVII ADM, 2009

[FCN08a] Fernández-Pacheco, D. G., Contero, M., Naya, F., Aleixos, N.: "Introducing 2D Sketches in a Cad Environment". 20º Congreso Internacional de Ingeniería Gráfica INGEGRAF 08, 2008

[FCN08b] Fernández-Pacheco, D. G., Contero, M., Naya, F., Aleixos, N.: "A Calligraphic Interface in a Sketch-Based Modelling Environment". 20º Congreso Internacional de Ingeniería Gráfica INGEGRAF 08, 2008

[Fer92] Ferguson, E.S.: “Engineering and the Mind's Eye”. MIT Press.

[FFJ09] Fonseca, M.J., Ferreira, A., Jorge, J.A.: "Sketch-based retrieval of complex drawings using hierarchical topology and geometry". Computer-Aided Design, vol 41 (12), 2009, pp. 1067-1081.

[FIPA02] FIPA ORG: “FIPA ACL Message Structure Specification”. Document no. SC00061G, 2002.

[FJ00] Fonseca, M.J., Jorge, J.: “Using Fuzzy Logic to Recognise Geometric Shapes Interactively”. Proceedings of 9th IEEE Conference on Fuzzy Systems, Vol. 1 (2000), pp. 291-296.

[FJM09] Flasinski, M., Jurek, J., Myslinski, S.: ”Multi-agent System for Recognition of Hand Postures”. In Proc of Computational Science - ICCS 2009, Springer, Lecture Notes in Computer Science, vol. 5545, 2009, pp 815-824

[GKS05] Gennari, L., Kara, L.B., Stahovich, T.F., Shimada, K.: “Combining geometry and domain knowledge to interpret hand-drawn diagrams”. Computers & Graphics, 29(4), 2005, pp. 547 -562

[Gol91] Goldberg, J. L.: "Matrix Theory with Applications". McGrawHill, 1991

[Gre86] Grefenstette, J.J.: “Optimization of Control Parameters for Genetic Algorithms”. IEEE Transactions on Systems, Man and Cybernetics, 1986, 16 (1), pp. 122-128

[Gro94] Gross, M.D.: “Recognising and Interpreting Diagrams in Design”. Proceedings of ACM (AVI’94), Bari, Italy, (1994) 88-94.

[GSE03] Gelasca, E.D., Salvador, E., Ebrahimi, T.: “Intuitive strategy for parameter setting in video segmentation”. In Proc. of the Visual communications and image processing Conference, Italy, 2003, vol. 5150 (3), pp. 998-1008

[GW02] Gonzalez, R.C., Woods, R.E.: “Digital Image Processing. 2nd Edition”. Prentice Hall, UpperSaddle River, NJ, 2002.

[HCE97] Hutton, G, Cripps, M, Elliman, D, Higgins, C.: “A strategy for on-line interpretation of sketched engineering drawings”. Proceedings of ICDAR ’97, pp. 771-775.

[HD02] Hammond, T., Davis, R.: “Tahuti: A geometrical sketch recognition system for UML class diagrams”. AAAI Spring Symposium on Sketch Understanding (March 25-27 2002), 59–68.

[HD05] Hammond, T., Davis, R.: “LADDER, A Sketching Language for User Interface Developers”. Computers & Graphics, 29(4), 2005, pp. 518-532

[HD09] Hammond, T., Davis, R.: "Recognizing interspersed sketches quickly". In Proceedings of Graphics Interface 2009, Vol. 324, pp. 157-166.

[HE04] Harding, P.R.G., Ellis, T.J.: “Recognising Hand Gesture Using Fourier Descriptors”. Proceedings of the 17th International Conference on Pattern Recognition, 2004, Vol. 3, 286-289.

[HEP08] Hammond, T., Eoff, B., Paulson, B., Wolin, A., Dahmen, K., Johnston, J., Rajan, P.: “Free-Sketch Recognition: Putting the CHI in Sketching“.CHI '08 extended abstracts on Human factors in computing systems (2008). ACM Press, pp. 3027-3032.

[HLL02] Hong, J., Landay, J., Long, A.C., Mankoff, J.: “Sketch Recognisers from the End-User’s, the Designer’s, and the Programmer’s Perspective”. American Association for Artificial Intelligence, 2003

[HN04] Hse, H., Newton, A.R.: “Sketched symbol recognition using Zernike moments”. In Proc. of the 17th International Conference On Pattern Recognition, vol. 1, 2004, pp. 367-370.

[Hof00] Hoffmann, D.: “Visual Intelligence. How we create what we see”. Norton Publishing. 2000.

[Hu62] Hu, M.: “Visual pattern recognition by moment invariants”. IRE Trans. Inf. Theor. (1962) IT-8: 179–187

[IM06] Iyer, G.R., Mills, J.J.: “Design intent in 2d CAD: definition and survey”. Computer–Aided Design & Applications, vol. 3, nºs. 1-4, pp. 259-267.

[IMT99] Igarashi, T, Matsuoka, S, Tanaka, H. Teddy: “A sketching interface for 3D freeform design”. Proceedings of ACM SIGGRAPH ’99, Los Angeles, California 8-13 August 1999. p. 409-416.

[ISK07] Iakovidis, D. K., Savelonas, M. A., Karkanis, S. A., Maroulis, D. E.: “A genetically optimized level set approach to segmentation of thyroid ultrasound images”. Applied Intelligence, 2007, 27(3), pp. 193-203

[JCR02] Julian, V., Carrascosa, C., Rebollo, M., Soler, J., Botti, V.: “SIMBA: An approach for real-time multi-agent systems”. In Proc. of Topics in Artificial Intelligence, Springer, Lecture Notes in Artificial Intelligence, vol. 2504, 2002, pp. 282-293

[JK84] Jansen, H, Krause, F.L.: “Interpretation of freehand drawings for mechanical design processes”. Computers & Graphics, vol. 8, no. 4, pp. 351-369

[JKS95] Jain, R., Kasturi, R., Schunck, B. G.: “Machine Vision”. MacGraw-Hill, Inc. 1995

[JLA05] Juchmes, R., Leclercq, P., Azar, S.: “A freehand-sketch environment for architectural design supported by a multi-agent system”. Computers & Graphics 29 (6) (2005) 905–915.

[JS02] Jarrett, R. and Su, P.: "Building Tablet PC applications". Microsoft Press, 1st edition (25 Sep 2002), ISBN: 978-0735617230.

[KCF92] Kouvelis, P., Chiang, W., Fitzsimmons, J.: “Simulated annealing for machine layout problems in the presence of zoning constraints”. European Journal of Operational Research, 1992, vol. 57, pp. 203-223

[KGV83] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.: “Optimization by simulated annealing”. Science, 1983, vol. 220 (4598), pp. 671-680

[KK06] Kim, D. and Kim, M.-J.: “A curvature estimation for pen input segmentation in sketch-based modelling”. In Computer-Aided Design, vol. 38, 2006, pp. 238–248.

[KL07] Kunttu, I., Lepisto, L.: ”Shape-based retrieval of industrial surface defects using angular radius Fourier descriptor”. IET Image Processing 1(2), 2007, pp. 231–236.

[KQW06] Ku, D.C., Qin, S.F., Wright, D.K.: “Interpretation of Overtracing Freehand Sketching for Geometric Shapes”. WSCG’2006. Full papers proceedings, G41, pp. 263-270.

[KS02] Kan, C., Srinath, M.D.: “Invariant character recognition with Zerkine and orthogonal Fourier-Mellin moments”. Pattern Recognition 35 (2002) 143-154.

[KS05] Kara, L.B., Stahovich, T.F.: “An image-based, trainable symbol recognizer for hand-drawn sketches". Computers & Graphics-UK (2005), Vol.29(4),pp. 501-517.

[KSP95] Kauppinen, H., Seppanen, T. and Pietikainen, M.: “An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D Shape Classification”. IEEE Transactions on pattern analysis and machine intelligence 17(2), 1995, pp. 201-207.

[KSS86] Kalvin, A., Schonberg, E., Schwartz, J.T. and Sharir, M.: "Two-dimensional, model-based, boundary matching using footprints". International Journal of Robotics Research, vol. 5 (4), 1986, pp. 38-55.

[LA87] van Laarhoven, P.J.M., Aarts, E.H.L.: “Simulated Annealing: Theory and Applications”. Kluwer Academic Publishers, 1987

[LCS09] Lopes, R., Cardoso, T., Silva, N. and Fonseca, M.J.: "Sketch-Based Interaction and Calligraphic Tags to Create Comics Online". Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM09), 2009, pp. 13-20.

[LF92] Leclerc, Y., Fischler, M.: “An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames”. International Journal of Computer Vision. Vol. 9, No. 2, pp. 113-136. 1992.

[LJZ07] Laviola, JR. J. J. and Zeleznik, R. C.: “A practical approach for writer-dependent symbol recognition using a writer independent symbol recognizer”. IEEE Trans. Pattern Anal. Mach. Intell. 29 (11), 2007, pp. 1917–1926.

[LKS07] Lee, W. , Kara, L.B., Stahovich, T.F.: “An efficient graph-based recognizer for hand-drawn symbols”. Computers & Graphics, 31(4), 2007, pp. 554-567.

[LN09] Li, Y., Ni, J.: “Constraints Based Nonrigid Registration for 2D Blade Profile Reconstruction in Reverse Engineering”. Journal of computing and information science in engineering, 2009, vol. 9 (3), Art. 031005 (9 pages)

[LP98] Liao, S.X., Pawlak, M.: “On the accuracy of Zernike moments for image analysis”. IEEE Trans. Pattern Anal. Mach. Intell. (1998) 20: 1358–1364

[LS04] Licsar, A. and Sziranyi, T.: “Hand Gesture Recognition in Camera-Projector System”. Computer Vision in Human-Computer Interaction. Lecture Notes in Computer Science 3058, 2004, pp. 83-93.

[LS96] Lipson, H., Shpitalni, M.: “Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing”. Computer-Aided Design. Vol. 28, No. 8, pp. 651-663. 1996.

[LSC95] Lee, J.-S., Sun, Y.-N., Chen, C.-H.: “Multiscale corner detection by using wavelet transform”. Image Processing, IEEE Transactions on 4, 1995, pp. 100–104.

[LW08] Lavoué, G., Wolf, C.: “Markov Random Fields for Improving 3D Mesh Analysis and Segmentation”. In Proc. of the Eurographics Workshop on 3D Object Retrieval 2008 (3DOR 08), Greece, pp. 25-32

[LZ04] Laviola, J. and Zeleznik, R.: “Mathpad2: A system for the creation and exploration of mathematical sketches”. Proceedings of SIGGRAPH 2004. ACM Transactions on Graphics 23 (3), 2004, pp. 432–440.

[LZA05] Li, J., Zhang, X., Ao, X., Dai, G.: “Sketch recognition with continuous feedback based on incremental intention extraction". Proceedings of the 10th international conference on Intelligent user interfaces 2005. ACM Press, pp. 145-150.

[MA03] Mackenzie, G., Alechina, N.: “Classifying sketches of animals using an agent-based system”. In Proceedings of 10th International Conference on Computer Analysis of Images and Patterns (CAIP’03), Lecture Notes in Computer Science, vol. 2756, Springer, Berlin, 2003, pp. 521–529.

[MA05] Mokhtarian, F., Abbasi, S.: “Robust automatic selection of optimal views in multi-view free-form object recognition”. Pattern Recognition 38 (2005) 1021-1031.

[MA97] Mukerjee, A., Agrawal, R.B., Tiwari N. and Hasan N.: “Qualitative sketch optimization”. Artificial intelligence for engineering design, analysis and manufacturing, 1997, vol. 11 (4), pp. 311-323

[Mar91] Marill, T.: “Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects”. International Journal of Computer Vision. Vol. 6, No. 2, pp. 147-161. 1991.

[MC95] Maniezzo, V., Colorni, A.: “Algodesk: an experimental comparison of eight evolutionary heuristics applied to the quadratic assignment problem”. European Journal of Operational Research, 1995, no. 81, pp. 188-204

[MI07] Mori, Y. and Igarashi, T.: "Plushie: An Interactive Design System for Plush Toys". ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007), vol 26 (3), 2007, Article 45.

[MRR53] Metropolis, N., Rosenbluth, A.W. and Rosenbluth, M.N. and Teller, A.H.: “Equation of state claculations by fast computing machines”. The Journal of Chemical Physics, 1953, vol. 21 (6), pp. 1087-1091

[MS07] Mbogho, A. J., Scarlatos, L. L.: “Genetic parameter tuning for reliable segmentation of colored visual tags”. In Proc. of the 9th Annual Conference on Genetic and Evolutionary Computation, London, England, GECCO 2007. ACM, New York, pp. 1525-1525.

[MSR09] Murugappan, S., Sellamani, S. and Ramani, K.: "Towards beautification of freehand sketches using suggestions". Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM09), 2009, pp. 69-76.

[MT07] Martin, V., Thonnat, M.: “A Cognitive Vision Approach for Image Segmentation Thresholding Images of Historical Documents with Back-to-Front Interference”. In Proc. of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), vol. 1, pp.480-487.

[NCA07a] Naya, F., Contero, M., Aleixos, N. and Company, P.: "Sketch-based Interface for 3D Parametric Modelling". XI International Congresss on Project Engineering, 2007, pp. 733-744.

[NCA07b] Naya, F., Contero, M., Aleixos, N. and Company, P.: "ParSketch: A Sketch-Based Interface for a 2D Parametric Geometry Editor". Human-Computer Interaction. Interaction Platforms and Techniques 2007. Lecture Notes in Computer Science, Vol 4551, 2007, pp. 115-124.

[NCA07c] Naya, F., Contero, M., Aleixos, N. and Jorge, J.: "Parametric Freehand Sketches". Computational Science and Its Applications (ICCSA 2004). Lecture Notes in Computer Science, Vol 3044, 2004, pp. 613-621.

[NCC07] Naya F., Contero M., Company P. and Aleixos N.: “Computer-Aided Sketching in Engineering Schools”. International Technology, Education and Development Conference (INTED07). 2007, pp. Virtual 352-Virtual 353.

[NIS07] Nealen, A., Igarashi, T., Sorkine, O. and Alexa, M.: "FiberMesh: designing freeform surfaces with 3D curves". ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007), vol 26 (3), 2007, Article 41

[OG89] Otten, R.H.J.M., van Ginneken, L.P.P.P.: “The annealing algorithm”. Kluwer Academic Publishers, 1989

[OSS09] Olsen, L., Samavati, F.F., Sousa, M.C., Jorge, J.A.: "Sketch-based modeling: A survey". Computers & Graphics, Vol 33 (1), 2009, pp. 85-103.

[Ott98] Ottosson, S.: “Qualified Product Concept Design Needs a Proper Combination of Pencil-aided Design and Model-aided Design Before Product Data Management”. Journal of Engineering Design, Vol. 9, No. 2.

[PA02] Plimmer, B., Apperley, M.: “Computer-Aided Sketching to Capture Preliminary Design”. Third Australasian conference on User interfaces. Vol. 7. pp. 9-12.

[PAG09] Palanca, J., Aranda, G., García-Fornes, A.: “Building Service-Based Applications for the iPhone Using RDF: A Tourism Application”. Advances in Intelligent and Soft Computing, Springer, 2009, vol 55, pp 421-429

[Pal99] Palmer, S. E.: “Vision Science. Photons to Phenomenology”. The MIT Press. 1999.

[PG09] Pu, J., Gur, D.: “Automated Freehand Sketch Segmentation Using Radial Basis Functions”. Computer-Aided Design. doi:10.1016/j.cad.2009.05.005.

[PH08] Paulson, B., Hammond, T.: “PaleoSketch: Accurate Primitive Sketch Recognition and Beautification”. In Proceedings of Intelligent User Interfaces 2008. pp. 1-10.

[PJB00] Pereira, J.; Jorge, J.; Branco, V.; Nunes, F.: “Towards calligraphic interfaces: sketching 3D scenes with gestures and context icons”. WSCG’2000. Conference Proceedings, Skala V. Ed., 2000.

[PJB04] Pereira, J. P., Jorge, J. A., Branco, V. A., Silva, N. F., Cardoso, T. D. and Ferreira, F. N.: “Cascading recognizers for ambiguous calligraphic interaction”. Proc. of the Eurographics Workshop on Sketch-Based Modelling (SBM'04), 2004, pp. 63–72.

[PLD04] Ponn, J., Lindemann, U., Diehl, H., Müller, F.: “Sketching in Early Conceptual Phases of Product Design: Guidelines and Tools”. Sketch-Based Interfaces and Modeling. Eurographics Symposium Proceedings. SBM'04 pp. 27-32.

[PM09] Pérez-Delgado, M.L. & Matos-Franco, J.C.: “Artificial Intelligence for Picking Up Recycling Bins: A Practical Application”. Advances in Intelligent and Soft Computing, Springer, 2009, vol 55, pp 392-400

[PMC04] Piquer, A., Martin, R.R., Company, P.: “Skewed Mirror Symmetry for Depth Estimation in 3D Line-Drawings”. Lecture Notes in Computer Science. GREC 2003 Post-proceedings. Vol. 3088. 2004, pp 138-149

[PP05] Park, C.H., Park, H.: “Fingerprint classification using fast Fourier transform and non-linear discriminant analysis”. Pattern Recognition 38 (2005) 495-503.

[PPS08] Posadas, J. L., Poza, J. L., Simo, J. E., Benet, G., Blanes, F.: "Agent-based distributed architecture for mobile robot control". Engineering Applications of Artificial Intellingence (2008), Vol.21(6), pp. 805-823

[PW04] Padgham, L., Winikoff, M.: “Developing Intelligent Agent Systems. A practical guide”. John Wiley & Sons (2004).

[QWJ01] Qin, S. F., Wright, D. K., Jordanov, I. N.: “On-line segmentation of freehand sketches by knowledge-based nonlinear thresholding operations”. Pattern Recognition 34, 2001, pp. 1885–1893.

[RBG04] Rodin, V., Benzinou, A., Guillaud, A., Ballet, P., Harrouet, F., Tisseau, J., Le Bihan, J.: “An immune oriented multi-agent system for biological image processing”. Pattern Recognition, 37(4), 2004, pp. 631-645.

[RC92] Rattarangsi, A., Chin, R.: “Scale-based detection of corners of planar curves”. Pattern Analysis and Machine Intelligence, IEEE Transactions on 14, 1992, pp. 430–449.

[Ros05] Rose, A.T.: “Graphical Communication Using Hand-Drawn Sketches in Civil Engineering”. Journal of Professional Issues in Engineering Education and Practice. Volume 131, Issue 4, pp. 238-247.

[RRS00] Rogers, T.J., Ross, R., Subrahmanian, V.S.: “IMPACT: A system for building agent applications”. Journal Of Intelligent Information Systems , 14(2-3), 2000, pp. 95-113.

[Rub91] Rubine, D.H.: “Specifying Gestures by Example”. Computer Graphics, Vol. 25, No. 4, Proceedings of the SIGGRAPH’91, (1991) 329-337.

[Sae00] Saeed, K.: “Three-agent system for cursive-scripts recognition”. In Proc. of the Fifth Joint Conference on Information Sciences, vol. 1&2, 2000, pp. 244-247.

[SC04] Shesh, A, Chen, B.: “SMARTPAPER: An interactive and user friendly sketching system”. Computer Graphics Forum 2004; 23(3): 301-310.

[SD06] Sezgin, T., Davis, R.: “Scale-space based feature point detection for digital ink”. Proc. of SIGGRAPH ’06: ACM SIGRRAPH 2006 Courses, 2006, p. 29.

[SD08] Sezgin, T.M., Davis, R.: “Sketch recognition in interspersed drawings using time-based graphical models”. Computers and Graphics, 32 (5), pp. 500-510.

[Sez01] Sezgin, T. M.: “Feature Point Detection and Curve Approximation for Early Processing of Free-Hand Sketches”. Master’s thesis, Massachusetts Institute of Technology, 2001

[SLS07] Sharf, A., Lewiner, T., Shklarski, G., Toledo, S. and Cohen-Or, D.: "Interactive topology-aware surface reconstruction". Proceedings of ACM SIGGRAPH 2007. ACM Transactions on Graphics (TOG), vol 26 (3), 2007, Article 43.

[SSD01] Sezgin, T.M., Stahovich, T., Davis, R.: “Sketch based interfaces: early processing for sketch understanding”. In Workshop on Perceptive User Interfaces, 2001.

[SSD07] Sezgin, T.M., Stahovich, T., Davis, R.: “Sketch based interfaces: early processing for sketch understanding”. International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2007, Session 3, Article 37.

[Sta04] Stahovich, T.: “Segmentation of pen strokes using pen speed”. In Proc. of AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural, 2004.

[TC89] Teh, C., Chin, R.: “On the detection of dominant points on digital curves”. IEEE Trans. Pattern Anal. Mach. Intell 17, 1989, pp. 859–872.

[TMH95] Tao, Y., Morrow, C.T., Heinemann, P.H., Sommer, H.J.: “Fourier-Based Separation Technique for Shape Grading of Potatoes Using Machine Vision”. Transactions of the ASAE, Vol. 38(3), pp. 949-957

[Tru98] Trucco, V.: “Introductory techniques for 3-D computer-vision”. Prentice Hall, 1998

[Tve02] Tversky, B.: “What do Sketches say about Thinking?”. AAAI Spring Symposium Series - Sketch Understanding. pp. 148-152.

[TW04] Taylor, G.W., Wolf, C.: “Reinforcement Learning for Parameter Control of Text Detection in Images and Video Sequences”. In Proc. of the IEEE International Conference on Information & Communication Technologies , 2004, pp. 517- 518

[WEH08] Wolin, A., Eoff, B., Hammond, T.: “Shortstraw: A simple and effective corner finder for polylines”. In EUROGRAPHICS 5th Annual Workshop on Sketch-Based Interfaces and Modeling, 2008, pp. 33–40.

[WJ95] Wooldridge, M., Jennings, N.R.: “Intelligent agents: theory and practice”. The Knowledge Engineering Review, 10 (2) (1995) 115–152.

[WK00] Wojnar, L., Kurzydłowski, K.J.: “Practical Guide to Image Analysis”. ASM International, 2000, p 157-160, ISBN 0-87170-688-1.

[Wol90] Wolfson, H.J.: "On curve matching". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.12 (5), 1990, pp. 483-489.

[Woo02] Wooldridge, M.J.: “An Introduction to multiagent systems”. John Wiley & Sons (2002).

[WS99] Wong, W.H.; Siu, W.C.: “Improved digital filter structure for fast moment computation”. IEE Proc. Vision, Image Signal Process (1999) 46: pp. 73–79

[WTB07] Wu, T-P., Tang, C-K., Brown, M.S. and Shum, H-Y.: "ShapePalettes: interactive normal transfer via sketching". International Conference on Computer Graphics and Interactive Techniques. ACM SIGGRAPH 2007, Article 44.

[WWL07] Wobbrock, J.O., Wilson, A.D. and Li, Y.: “Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes”. Proceedings of the 20th ACM symposium on User interface software and technology (UIST’07), ACM, 2007, pp. 159-168.

[XL09] Xiong, Y. and LaViola, Jr. J.J.: “Revisiting ShortStraw: Improving Corner Finding in Sketch-Based Interfaces”. In EUROGRAPHICS 6th Annual Workshop on Sketch-Based Interfaces and Modeling, 2009, pp. 101-108.

[XWJ02] Xiangyu, J., Wenyin, L., Jianyong, S., Sun, Z.: “On-Line Graphics Recognition”. Pacific Conference on Computer Graphics and Applications 2002: pp. 256-264.

[YC03] Yu, B., Cai, S.: “A domain-independent system for sketch recognition”. In Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (2003), ACM Press, pp. 141-146.

[YF08] Yu, HY., Fan, JL.: “Parameter Optimization Based on Quantum Genetic Algorithm for Generalized Fuzzy Entropy Thresholding Segmentation Method”. In Proc. of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery 2008 (FSKD 08), Vol 1 (18-20), pp. 530-534

[YTJ08] Yuan, S., Tsui, L.Y. and Jie, S.: “Regularity selection for effective 3D object reconstruction from a single line drawing”. Pattern Recognition Letters 29 (10), 2008, pp. 1486-1495.

[Yu03] Yu, B.: “Recognition of Freehand Sketches Using Mean Shift”. Proc. of the 8th International Conference on Intelligent User interfaces (IUI '03), ACM Press, pp. 204-210.

[ZHH96] Zeleznik, RC, Herndon, KP, Hughes, JF.: “SKETCH: an interface for sketching 3D scenes”. Proceedings of SIGGRAPH ’96, 1996. p. 163-170.

[ZL02] Zang, D., Lu, G.: “A Comparative Study of Fourier Descriptors for Shape Representation and Retrieval”. The 5th Asian Conference on Computer Vision, Melbourne, Australia, ACCV’02, 2002, pp. 1-6.

[ZL07] Zou, H.L. and Lee, Y.T.: “Constraint-based beautification and dimensioning of 3D polyhedral models reconstructed from 2D sketches”. Computer-Aided Design, 39 (11), 2007, pp. 1025-1036.

[ZSK99] Zion, B., Shklyar, A., Karplus, I.: “Sorting fish by computer vision”. Comput. Electron. Agric. (1999) 23: 175–187.

Enlaces de interés:

[AutCad82] Autocad. Url: http://www.autodesk.es

[BUGG] Brown University Graphics Group. Url: http://graphics.cs.brown.edu

[Catia77] Catia. Url: http://www.3ds.com/products/catia

[CCSL] Cornell Computational Synthesis Lab. Url: http://www.mae.cornell.edu/ccsl/

[CDL] Computational Design Lab. Url: http://code.arc.cmu.edu

[CGGTUI] Computer Graphics Group of the Technical University of Ilmenau. Url: http://www.tu-ilmenau.de/cg/

[CVCUAB] Centro de Visión por Computador de la Universidad Autónoma de Barcelona. Url: http://www.cvc.uab.es/

[FBSkeMe09] FaceBook Sketch Me. Url: http://www.facebook.com/applications/Sketch_Me/4260387428

[GUIR] Group for User Interface Research. Url: http://guir.berkeley.edu

[ILovSke08] ILoveSketch. Url: http://www.dgp.toronto.edu/~shbae/ilovesketch.htm

[IMMI] Intelligent MultiModal Interfaces Group. Url: http://immi.inesc.pt

[Iphon07] Iphone. Url: http://www.apple.com/es/iphone/

[JavSke97] Java Sketch. Url: http://graphics.cs.brown.edu/research/sketch/java-sketch/

[MCRPC] Microsoft Center for Research on Pen-Centric Computing. Url: http://pen.cs.brown.edu/

[MITSke07] Magic Paper – An application for sketching from MIT. Url: http://icampus.mit.edu/MagicPaper/

[ProEng88] Pro/Engineer. Url: http://www.ptc.com/products/proengineer/

[SBIMW] Sketch-Based Interfaces and Modelling Workshop. Url: http://www.eg.org/sbm

[SkeBook07] Autodesk SketchBook Pro for iPhone. Url: http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=6848332

[SkeMast05] Sketch Master. Url: http://www.redfieldplugins.com/filterSketchMaster.htm

[Sket2iP08] Sketches 2 for iPhone & Ipod-Touch. Url: http://www.sketchesapp.com/

[SketUp00] Sketch Up!. Url: http://sketchup.google.com/

[SldWks95] SolidWorks. Url: http://www.solidworks.es/

[SusApp07] Simple Sketch Applicattion on GAppEngine. Url: http://susan.appspot.com/

[TablPc01] Tablet PC. Url: http://es.wikipedia.org/wiki/Tablet_PC

[UIRG] User Interface Research Group – Igarashi Lab. Url: http://www-ui.is.s.u-tokyo.ac.jp

Anexo I : HERRAMIENTAS SOFTWARE EMPLEADAS

I. Herramientas software empleadas

I.1 El estándar FIPA

El creciente desarrollo de grupos de investigación en torno a los sistemas multi-agente fomentó la aparición de nuevas formas para el desarrollo de esta tecnología, lo cual conllevó a que cada uno de estos grupos presentara soluciones diferentes e independientes.

Algunos problemas que se presentaron fue la carencia de una definición estándar de sistema multi-agente e incompatibilidad, lo que conllevo a una incapacidad para satisfacer los fuertes requisitos de las empresas e industrias actuales. Esto promovió la creación de diversos organismos, los cuales se preocuparían de desarrollar una definición estándar para la construcción de sistemas multi-agente. Algunos de estos organismos ya mencionados son:

- OMG (Object Management Group), quienes desarrollaron MASIF (Mobile Agent System Interoperability Facilities).
- KSE (Knowledge Sharing Effort), quienes desarrollaron KQML (Knowledge Querying and Manipulation Language) y KIF (Knowledge Interchange Format).
- Agent Society, cuya labor principal es el intercambio y recopilación de información sobre agentes.
- FIPA (Foundation for Intelligent Physical Agents), cuya labor principal es el desarrollo de especificaciones de una arquitectura para sistemas multi-agente, infraestructura y aplicaciones.

De todos estos estándares, FIPA es el que más relevancia ha tomado en estos últimos años. En su versión inicial (FIPA 97) tres documentos integraban esta parte del estándar:

- Agent Management
- Agent Communication Language
- Agent/Software Integration

Posteriormente se han ido actualizando e incluyendo nuevos documentos hasta alcanzar la versión de FIPA-2000 con la que actualmente se trabaja [FIPA02].

El “Agent Management” proporciona la normativa del entorno donde los agentes FIPA se crean y operan, y establece el modelo lógico de referencia para la gestión de agentes (creación, registro, localización, comunicación, migración y terminación de los agentes). Este modelo presenta un conjunto de capacidades lógicas y no implica ninguna configuración física, sino que deja los detalles de implementación a elección del equipo de desarrollo.

El modelo de referencia para la gestión de agentes está formado por los siguientes componentes lógicos, como muestra la Figura I.1.

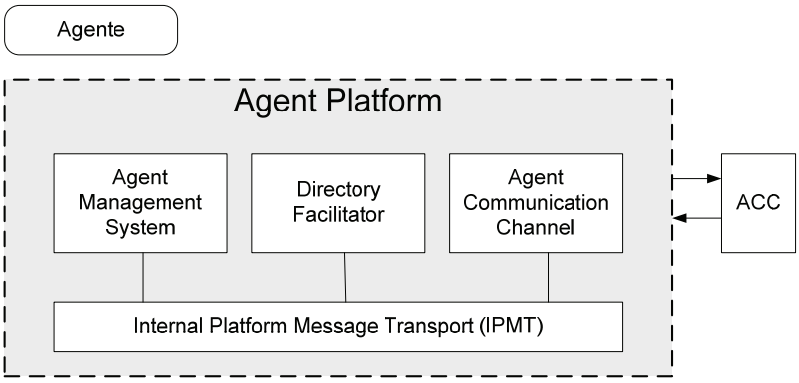


Figura I.1. Modelo de referencia para la gestión de agentes

- **Agente:** Es el componente básico y principal del modelo. Combina una o más capacidades de servicio dentro de un entorno de ejecución integrado y unificado que proporciona servicios de comunicación, acceso a software externo y acceso a los usuarios. Un agente tiene que tener uno o más dueños. Además, debe disponer de una identidad propia proporcionada por un identificador global y único GUID (Globally Unique IDentifier) denominado nombre del agente. Un agente puede registrarse con un número de direcciones en las cuales puede ser contactado.
- **Agent Platform (AP):** Proporciona la infraestructura física y lógica necesaria en la cual los agentes pueden ejecutarse. Una plataforma de agentes está constituida por el hardware (puede haber varios computadores), el sistema operativo, el software de comunicaciones y el software de agentes.
- **Directory Facilitator (DF):** Componente que siempre tiene que aparecer en cualquier plataforma de agentes FIPA. Es un agente que proporciona un servicio de "páginas amarillas" a los demás agentes. Un agente puede utilizar el DF para registrar sus servicios o para encontrar los servicios ofrecidos por otros agentes (ver Figura I.2).
- **Agent Management System (AMS):** Componente que siempre tiene que aparecer en cualquier plataforma de agentes FIPA, uno por plataforma. Es un agente de gestión que controla el estado y el acceso a la plataforma. También proporciona un servicio de "páginas blancas" que permite la localización de agentes a partir de sus nombres.
- **Agent Communication Channel (ACC):** Todos los agentes tienen acceso al menos a un ACC. El ACC es el canal de comunicación por defecto entre agentes de diferentes plataformas. Tiene que soportar el protocolo de comunicación para interoperabilidad IIOP (Internet Inter-ORB Protocol).

- **Internal Platform Message Transport (IPMT):** Método de intercambio de mensajes dentro de la misma plataforma el cual depende de la implementación.

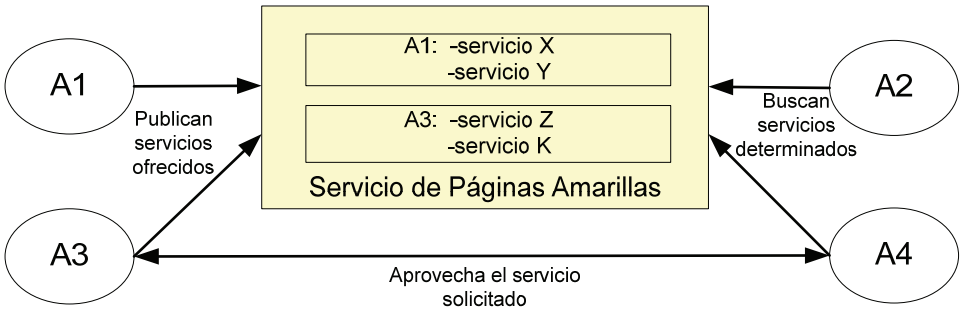


Figura I.2. Servicio de *Páginas amarillas*

El Facilitador de Directorio (DF) proporciona un servicio de páginas amarillas al resto de los agentes, los cuales lo usan para registrar sus servicios o para obtener los servicios de los demás agentes. Cualquier plataforma de agentes FIPA debe poseer al menos un DF. Una peculiaridad es que los DF's pueden registrarse entre ellos, al igual que los componentes AMS y ACC pueden registrarse con un DF.

Las acciones de gestión que soporta son las siguientes:

- **Register:** Petición de registro de los servicios de un agente.
- **Search:** Petición de búsqueda de agentes que proporcionen determinados servicios.
- **Deregister:** Petición de borrado de los servicios registrados de un agente.
- **Modify:** Petición de modificación de los servicios registrados de un agente

En una plataforma de agentes FIPA únicamente puede existir un *Agent Management System* (AMS), siendo éste el responsable de gestionar las operaciones de su plataforma tales como la creación de agentes, borrado de agentes, decidir si un agente puede registrarse en la plataforma y supervisar la migración de los agentes a/desde la plataforma. El AMS es la máxima autoridad en toda la plataforma, pudiendo solicitar a un agente que finalice su ejecución (*quit*) e incluso obligarle a finalizar.

Otra característica del AMS es la gestión de una lista con todos los agentes que se encuentran en la plataforma. En dicha lista se encuentran las identificaciones (GUID) de los agentes asociadas a sus direcciones en la plataforma. Esto implica que para que un agente pueda residir en la plataforma debe registrarse previamente con el AMS.

Las acciones de gestión que debe soportar cualquier AMS son:

- **Register-agent:** Petición de registro de un agente.
- **Deregister-agent:** Petición de borrar del registro a un agente.
- **Modify-agent:** Petición de modificación del registro de un agente.
- **Query-platform-profile:** Petición del perfil de la plataforma.
- **Authenticate:** Petición de autentificar la identidad de un agente creado en la plataforma.
- **Search-agent:** Petición de búsqueda de la dirección de un agente.

Existe una acción de gestión para movilidad que también tiene que soportar cualquier AMS:

- **Move:** Petición de transferencia/migración de un agente.

Además de las acciones de gestión intercambiadas entre el AMS y los agentes en la plataforma, el AMS puede proporcionar a la plataforma la realización de las siguientes operaciones:

- **Suspend agent:** Suspender o parar la ejecución de una agente.
- **Terminate agent:** Finalizar la ejecución de un agente.
- **Create agent:** Crear un agente.
- **Resume agent execution:** Continuar con la ejecución de un agente suspendido.
- **Invoke agent:** Empezar la ejecución de un agente recién creado.
- **Execute agent:** Ejecutar un agente después de moverse.
- **Resource management:** Gestionar los recursos.

Existe una acción de gestión que tienen que soportar los agentes utilizados por el AMS:

- **Quit:** Petición de terminación de un agente, la cual finaliza la ejecución del agente que recibe la solicitud.

Por otro lado, el ACC proporciona el encaminamiento de mensajes entre agentes de diferentes plataformas FIPA. Todos los agentes FIPA tienen acceso al menos a un ACC, de forma que únicamente aquellos mensajes que estén dirigidos a agentes podrán ser enviados a un ACC.

El encaminamiento de mensajes entre diferentes plataformas requiere un protocolo de interoperabilidad por defecto (especificándose el protocolo de transporte, codificación y direccionamiento). Este protocolo es el IIOP, que soporta la acción de gestión:

- **Forward:** Petición de envió de un mensaje desde un agente a otro.

Esta acción no debe de interpretarse como el mecanismo por defecto para enviar mensajes entre los agentes de una misma plataforma, empleándose en este caso el método interno (IPMT) que se haya implementado en la propia plataforma.

Dentro del Agent Plataform (AP) el ciclo de vida de los agentes FIPA está caracterizado por:

- Definir los posibles estados y transiciones de un agente independientemente del tipo de implementación de la plataforma.
- Cada agente es físicamente gestionado en una plataforma. Si el agente es estacionario, su ciclo de vida siempre será en la misma plataforma.
- Cada agente descrito con el modelo de ciclo de vida es asumido como una instancia con nombre propio y único que se ejecuta de forma independientemente.
- Cada agente solo puede encontrase en un estado en una única plataforma al mismo tiempo.
- El ciclo de vida de los agentes se representa mediante estados y transiciones, los cuales se muestran en la Figura I.3.

Únicamente los agentes móviles pueden entrar en el estado "*transit*". Esto asegura que los agentes estacionarios ejecuten todas sus instrucciones en el nodo donde fueron invocados.

Las acciones para pasar los agentes de un estado a otro son:

- **Create:** Creación de un nuevo agente.
- **Invoke:** Invocación o activación de un nuevo agente una vez inicializado.
- **Destroy:** Se fuerza la terminación de un agente. Esta acción sólo puede ser iniciada por el AMS y no puede ser ignorada.
- **Quit:** Se solicita la terminación de un agente. El agente puede ignorar la acción.
- **Suspend:** Pone a un agente en el estado de suspendido. Esta acción puede ser iniciada por el propio agente o por el AMS.
- **Resume:** Saca a un agente del estado suspendido. Esta acción sólo puede ser iniciada por el AMS.

- **Wait:** Pone a un agente en el estado de espera. Esta acción sólo puede ser iniciada por el propio agente.
- **Wake up:** Saca a un agente del estado de espera. Esta acción sólo puede ser iniciada por el AMS.

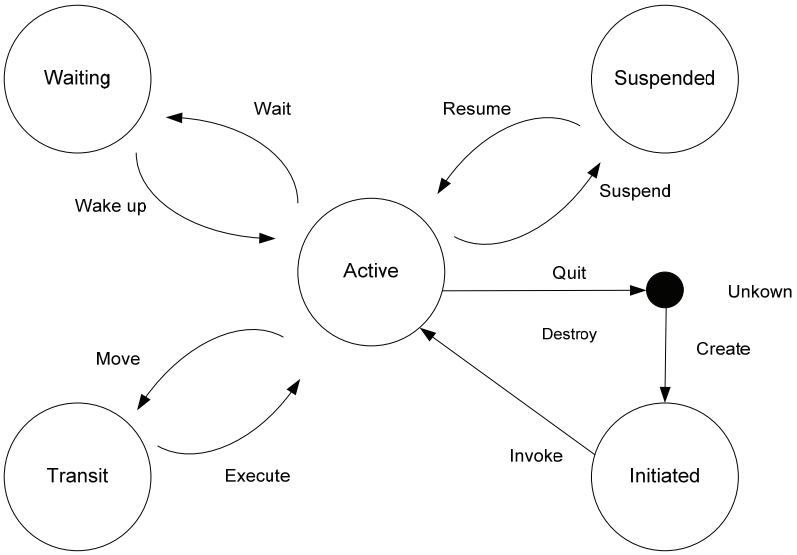


Figura I.3. Ciclo de vida de los agentes FIPA

Las siguientes acciones sólo son usadas por los agentes móviles:

- **Move:** Pone a un agente en el estado de transición. Esta acción sólo puede ser iniciada por el propio agente.
- **Execute:** Saca al agente del estado.

Paralelamente a los estados, en la Figura I.4 se muestra el ciclo de ejecución de un agente:

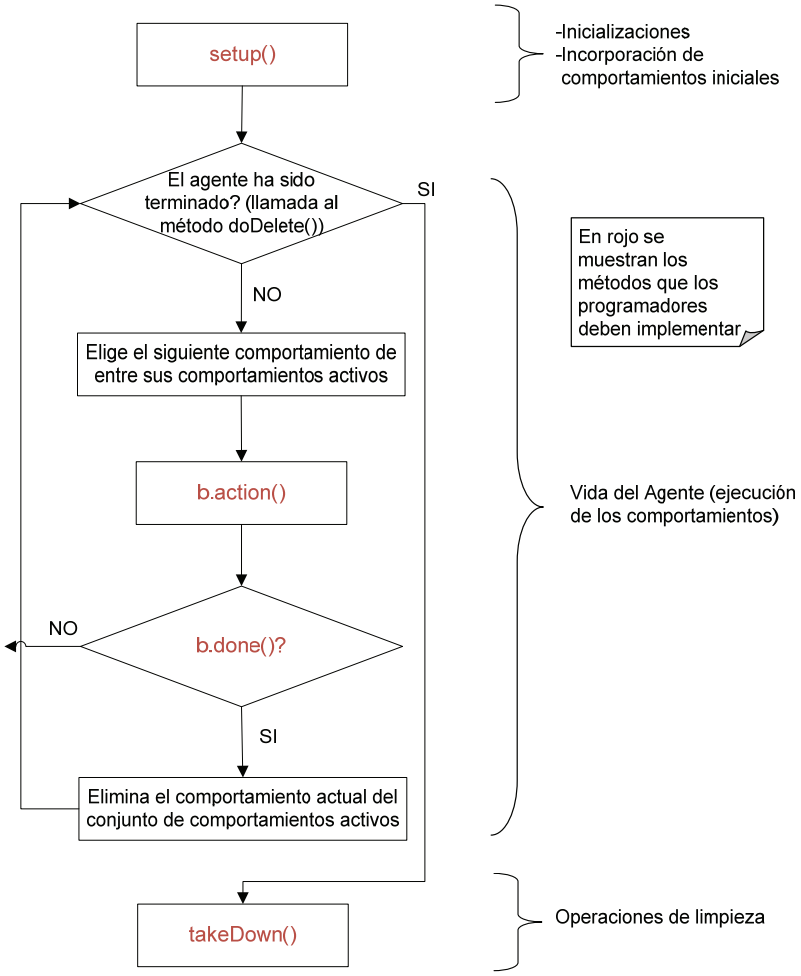


Figura I.4. Ciclo de ejecución de un Agente

La plataforma en la cual se creó un agente se denomina la *Home Agent Platform* (HAP) de dicho agente. La responsabilidad del HAP es garantizar la identidad del agente en sus relaciones con otros agentes y plataformas. El estándar FIPA requiere que cada agente tenga un HAP que responda por él ante el resto de la comunidad de agentes. Por tal motivo es necesario que a partir del análisis del GUID de un agente pueda obtenerse la dirección IIOP-URL del HAP, así como que el HAP pueda autentificar la identidad del agente en la plataforma. Para ello, el AMS del HAP soporta la acción "**authenticate**".

El AMS del HAP de un agente es pues el responsable de guardar la dirección de dicho agente. Por otro lado, el agente tiene la responsabilidad de asegurar que la dirección guardada por el AMS sea válida, por lo que un agente siempre deberá registrarse en su HAP. Esto

implica que, evidentemente, el nombre de un agente (GUID) sea el mismo durante toda su existencia.

Para que un agente pueda registrarse en una plataforma se debe cumplir alguna de las siguientes condiciones:

- El agente se creó en la plataforma.
- El agente migró a la plataforma (ambas plataformas, HAP y destino, tienen que soportar movilidad de agentes).
- El agente se registra dinámicamente en una plataforma diferente a su HAP como agente local (ambas plataformas tienen que soportar registro dinámico).

Al registrarse un agente se facilita al AMS la siguiente información:

- El identificador global y único del agente (GUID).
- La dirección local del agente.

Este registro en el AMS se lleva a cabo mediante la acción "**register-agent**".

Un agente tiene dos opciones cuando desea contactar (enviar un mensaje) con otro agente de una plataforma diferente:

- Puede solicitar que el ACC de su plataforma envíe el mensaje al ACC y agente de la otra plataforma.
- Puede contactar directamente con el ACC de la otra plataforma y enviarle el mensaje.

Para llevar a cabo esto, es necesario el GUID del agente destino para poder contactar con él. El mensaje se enviará al HAP del agente destino y a partir de aquí se hará llegar a dicho agente. Alternativamente, si se desea enviar el mensaje directamente al agente o a una plataforma (distinta del HAP) donde el agente se registró dinámicamente, entonces se deberá indicar además del GUID la dirección destino del agente. En la Figura I.5 se muestra una distribución genérica de contenedores de agentes y diferentes plataformas.

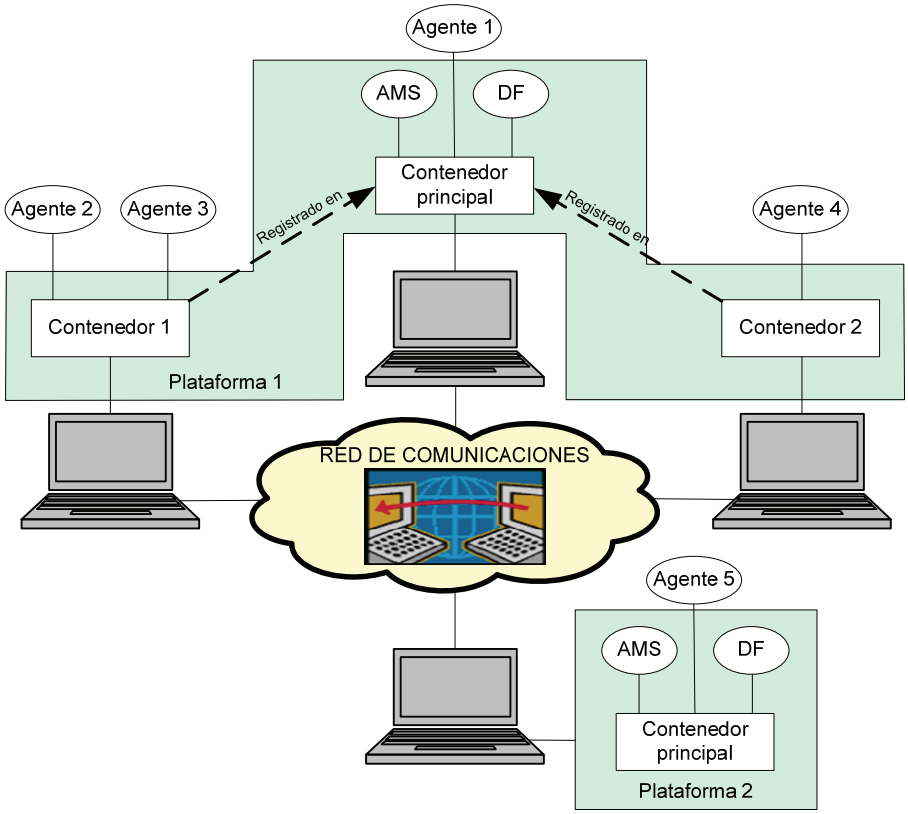


Figura I.5. Distribución genérica de Contenedores y Plataformas

I.2 El entorno de programación ECLIPSE

Eclipse es un entorno de programación que posibilita la programación en JAVA. Fue creado inicialmente por IBM (y administrado actualmente por la Fundación Eclipse). En cuanto a actualizaciones, tiene soporte regular por parte de sus patrocinadores. En cuanto al tema de los plugins, cuenta con una gran comunidad de usuarios y crear plugins para Eclipse resulta muy sencillo, lo cual lo convierte en un IDE extremadamente potente.

Como características generales, la plataforma Eclipse está diseñada para afrontar las siguientes necesidades:

- Soportar la construcción de gran variedad de herramientas de desarrollo.
- Soportar las herramientas proporcionadas por diferentes fabricantes de software independientes (ISV's)
- Soportar herramientas que permitan manipular diferentes contenidos (i.e. HTML, Java, C, JSP, EJB, XML, y GIF).
- Facilitar una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta al proveedor.
- Proporcionar entornos de desarrollo gráfico (GUI), o sólo de texto.
- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows® y Linux™.

El principal objetivo de la plataforma Eclipse es proporcionar mecanismos y reglas que puedan ser seguidas por los fabricantes para integrar de manera transparente sus herramientas. Mediante API's, interfaces, clases y métodos, se exponen estos mecanismos. La plataforma también nos posibilita la construcción de nuevas herramientas que extienden su funcionalidad.

La Figura I.6 muestra los componentes API's, de la plataforma Eclipse.

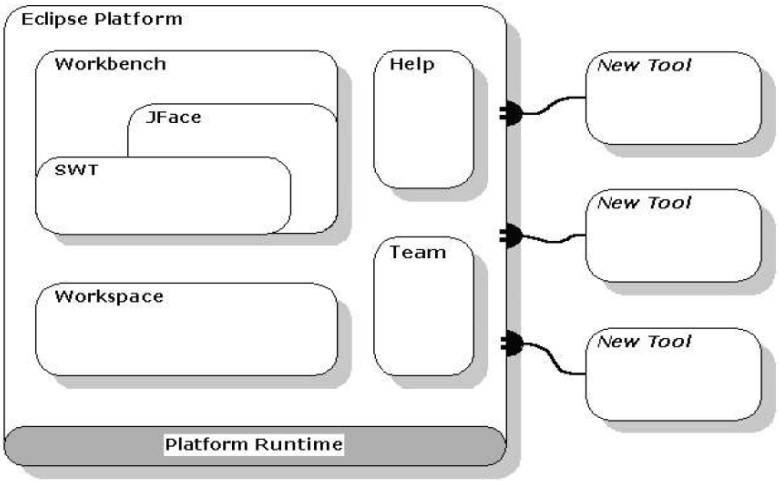


Figura I.6. Componentes API's de la plataforma Eclipse

I.2.1 Plataforma de Ejecución y arquitectura de Plug-in

Un plug-in es la unidad mínima de funcionalidad de Eclipse que puede ser distribuida de manera separada. Herramientas pequeñas se escriben como un único plug-in, mientras que en las complejas la funcionalidad está contenida en varios plug-in. Excepto un pequeño núcleo de la plataforma Eclipse, el resto de la funcionalidad de la plataforma Eclipse está implementada como plug-in.

Un plug-in está escrito en Java y formado por un fichero JAR (Java ARchive) de código Java, ficheros de lectura y otros recursos como imágenes, catálogos de mensajes, librerías de código nativo, etc. Algunos plug-in no contienen código, como por ejemplo el que nos proporciona ayuda en forma de páginas HTML.

Todos los recursos que componen el plug-in se encuentran en un directorio del sistema de archivos del sistema operativo, o en una URL de un servidor. Existe un mecanismo que permite que un plug-in pueda ser sintetizado a partir de distintos fragmentos, cada uno en su propio directorio o en su propia URL. Este mecanismo es utilizado para distribuir diferentes paquetes de idiomas para un plug-in que soporte diversos idiomas (internacionalización).

Cada plug-in tiene un fichero denominado manifest (manifiesto) en el cual se declaran sus interconexiones con otros plug-in. La interconexión sigue un modelo muy simple: un plug-in declara un número de los denominados puntos de extensión, y un número de extensiones para uno o más puntos de extensión de otros plug-in. Los puntos de extensión de un plug-in pueden ser extendidos por otros plug-in. Por ejemplo, el plug-in del banco de trabajo (workbench) declara un punto de extensión para las preferencias de usuario. Cualquier otro plug-in puede contribuir con sus propias extensiones a las preferencias de usuario, mediante la definición de extensiones a este punto de extensión.

Un punto de extensión puede tener un interfaz API. Otros plug-in contribuyen dando implementaciones a este API. Cualquier plug-in es libre de definir nuevos puntos de extensión así como de proporcionar nuevos API's para que otros plug-in puedan utilizarlos.

Al iniciar la plataforma de ejecución se descubren de manera dinámica el conjunto de plug-in disponibles, se leen sus archivos de manifiesto, y se construye en memoria un registro de plug-in. La plataforma enlaza cada extensión por el nombre con sus declaraciones de puntos de extensión. Cualquier problema, como extensiones sin sus correspondientes puntos de extensión, se detecta y se registra. El registro de plug-in que se ha generado queda disponible a través del API de la plataforma, no pudiendo ser añadido ningún nuevo plug-in después del inicio.

Los archivos de manifiesto de los plug-in contienen XML. Un punto de extensión puede declarar tipos de elementos XML adicionales para ser utilizados en las extensiones. Esto permite el paso de datos entre los plug-in. Además, la información del archivo de manifiesto está disponible desde el registro de plug-in sin haber activado los plug-in o haber cargado algo de su código. Esto es fundamental para soportar un gran número de plug-in, a pesar de que sólo un número muy pequeño de ellos sean utilizados por el usuario. Hasta que el código del plug-in no es cargado, el efecto que tiene sobre el entorno de ejecución es nulo. La utilización de archivos XML permite la utilización de herramientas de desarrollo para la construcción de los mismos. El entorno de desarrollo de Plug-in (PDE) incluido en la plataforma Eclipse es precisamente una de estas herramientas.

Un plug-in es activado cuando su código realmente necesita ser ejecutado. Una vez activado, el plug-in utiliza el registro de plug-in para descubrir y acceder a las extensiones que contribuyen a sus puntos de extensión. Por ejemplo, el plug-in que declara el punto de acceso de preferencias de usuario puede descubrir todas las preferencias de usuario contribuidoras y mostrarlas para construir un cuadro de diálogo con todas ellas. Esto puede ser realizado sin necesidad de cargar el código de esos plug-in, simplemente consultando el registro de plug-in.

El plug-in contribuidor será activado sólo cuando el usuario seleccione la preferencia de la lista. La activación de este modo no sucede de manera automática. Existen API's para la activación explícita de los plug-in. Una vez activado, el plug-in permanece activo hasta que la plataforma finaliza. Cada plug-in tiene su propio subdirectorio donde poder almacenar datos específicos del mismo, lo que permite mantener datos de sesión entre ejecuciones.

La plataforma se ejecuta en una única invocación de la máquina virtual de Java (JVM). A cada plug-in se le asigna su propio cargador de clases, el cual únicamente se usa para cargar las clases del plug-in así como los recursos del mismo.

I.2.2 Espacios de trabajo (Workspaces)

Las diferentes herramientas que son instaladas en la plataforma Eclipse actúan sobre ficheros regulares que se almacenan en lo que se denominan espacios de trabajo o workspace, que es específico para el usuario. El espacio de trabajo de un usuario contiene varios directorios a un nivel más alto (que se denominan proyectos). Cada proyecto contiene

los archivos que son creados y manipulados por el usuario. Todos los archivos en el espacio de trabajo son directamente accesibles por programas estándar y herramientas del sistema operativo.

Para minimizar el riesgo de perder archivos, existe un mecanismo de historial mantenido a nivel de cada uno de los proyectos. El usuario puede controlar cómo el historial se mantiene a través de las preferencias: tamaño, número de días que se mantiene, etc.

I.2.3 Workbench y Toolkits UI

La plataforma Eclipse tiene un interfaz gráfico (GUI), construido en base a un banco de trabajo o workbench que proporciona toda la estructura y presenta un interfaz de usuario (UI). Existe también un API de este workbench y su implementación se lleva a cabo a través de lo que se denominan toolkits, que pueden ser de dos tipos:

- SWT: conjunto de utilidades y librerías gráficas integradas con el sistema nativo de ventanas pero con un API independiente del sistema operativo.
- JFace: interfaz gráfico implementado sobre SWT que simplifica las tareas de programación.

I.2.4 Programación en Equipo

La plataforma Eclipse permite que un proyecto en el espacio de trabajo de un usuario, pueda ser puesto en un repositorio al cual pueden acceder otros desarrolladores. La plataforma tiene puntos de extensión y un API de proveedores que permite que nuevos tipos de repositorios puedan ser instalados.

En resumen, la plataforma Eclipse proporciona un núcleo de componentes de construcción básica y API's como el workspace y el workbench, varios puntos de extensión a través de los cuales podremos integrar nuevas funcionalidades.

I.3 El entorno SWIG

Se conoce como lenguaje interpretado a un lenguaje de programación que ha sido diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados. También se les conoce como lenguajes de *script*. Muchos autores rechazan la clasificación de lenguajes de programación entre interpretados y compilados, considerando que el modo de ejecución (por medio de intérprete o de compilador) del programa escrito en el lenguaje es independiente del lenguaje mismo.

SWIG es una herramienta de desarrollo software que simplifica la tarea de interpretar programas desarrollados en otros lenguajes a aplicaciones escritas en C y C++. Es decir, SWIG es un compilador que tomando declaraciones hechas en C, crea los *wrappers* necesarios para poder acceder a las declaraciones de otros lenguajes, entre ellos Perl, Python, Tcl, Ruby, Guile y Java. SWIG, normalmente, no requiere que se modifique el código existente y a menudo se puede emplear para construir un interfaz válido en tan sólo unos minutos. Entre las posibles aplicaciones de SWIG se incluye:

- Generación de interfaces interpretados para programas escritos en C.
- Desarrollo de prototipos y aplicaciones de forma rápida.
- *Debug* (depuración de código) interactivo.
- Descomposición de *software* en componentes del lenguaje interpretado.
- Desarrollo de un interfaz de usuario gráfico.
- Testeo de librerías y programas escritos en C (usando intérpretes).
- Generación de módulos en C de alto rendimiento para lenguajes interpretados.
- Hacer más entretenida la programación en C (o llevadera dependiendo del punto de vista).

SWIG fue diseñado originalmente para hacer extremadamente sencillo que científicos e ingenieros pudieran generar una herramienta software sin ser especialistas en desarrollo de software. Por esta razón el empleo de SWIG tiende a ser algo informal (por ejemplo, SWIG no requiere que los usuarios creen especificaciones formales del interfaz como podría ocurrir en un compilador dedicado IDL).

A estas alturas podría surgir la pregunta de por qué puede interesar integrar C/C++ con otros lenguajes de programación. Para responder esa cuestión lo mejor es indicar algunas de las ventajas que aporta la programación en C/C++:

- Excelente soporte para desarrollar librerías de programación.
- Alto rendimiento (cálculos numéricos, procesado de datos, gráficos, etc.).
- Programación e integración de sistemas.
- Una gran comunidad de usuarios y mucho software disponible.

En contraposición a las siguientes desventajas:

- Escribir un interfaz de usuario puede resultar bastante tedioso (por ejemplo, empleando MFC, XII, GTK o cualquier otra librería).
- El testeo suele consumir bastante tiempo (el ciclo compilación/debug).
- Modularizar el código puede resultar complicado.
- Resulta difícil asegurar que no se producirán errores de ejecución, como el típico desbordamiento de memoria (también conocido como *"buffer overflow"*).

Para localizar estas limitaciones muchos programadores han llegado a la conclusión de que es mejor usar lenguajes de programación diferentes para tareas diferentes. Por ejemplo, escribir un interfaz de usuario gráfico puede resultar mucho más sencillo en un lenguaje interpretado como Python o Tcl (considérense las razones por las que millones de programadores han usado lenguajes como Visual Basic como prueba). Un intérprete interactivo podría incluso servir como herramienta de debug y testeo. Otros lenguajes como Java podrían simplificar enormemente la tarea de escribir software de computación distribuida, etc. La clave está en que distintos lenguajes de programación ofrecen diferentes fortalezas y debilidades. Es más, no existe un lenguaje de programación perfecto. Por esta misma razón, el uso de lenguajes combinados permite utilizar las mejores características de cada lenguaje y simplificar encarecidamente el desarrollo de la aplicación software.

Desde el punto de vista de C/C++, muchos desarrolladores emplean SWIG porque desean deshacerse del típico modo de programación que resulta en programas del tipo:

- Una colección de funciones y variables que permiten implementar algo útil.
- Un programa Main() que lo comienza todo.
- Una colección de elementos mal dispuestos que forman una especie de interfaz de usuario (pero que nadie se atreve a tocar).

En lugar de seguir por ese camino, incorporar C/C++ en lenguajes de alto nivel a menudo permite un diseño más modular, menos código, más flexibilidad, y un incremento en la productividad.

SWIG intenta hacer el problema de integrar C/C++ lo menos costoso posible, lo que permite al programador centrarse en lo fundamental del programa y usar el interfaz de alto nivel, sin preocuparse por la tarea de hacer que ambos lenguajes se comuniquen. Al mismo tiempo, SWIG reconoce que todas las aplicaciones son diferentes. Además, proporciona una extensa variedad de características modificables.

Dentro de las características del lenguaje C/C++ que soporta SWIG las más destacadas son:

- Pre-procesado completo C99.
- Todos los tipos de datos ANSI C y C++.
- Funciones, variables y constantes.
- Clases.

- Herencia simple y múltiple.
- Sobrecarga de funciones y métodos.
- Sobrecarga de operadores.
- Espacios de nombres (o *Namespaces)*.
- Argumentos de longitud variable.
- Punteros Smart C++.

I.3.1 Interpretando C++

A causa de su complejidad y del hecho de que puede ser difícil integrar C++ con otros lenguajes, SWIG sólo ofrece soporte para unas características de C++. Afortunadamente esta lista es actualmente muy extensa.

En parte, el problema con el *wrapping* de C++ es que no existe una forma automática de trasladar muchas de las características avanzadas de C++ a otros lenguajes. Un ejemplo sencillo sería el intentar interpretar la herencia múltiple de C++ a un lenguaje sin ese tipo de soporte. De igual manera, el uso de operadores y funciones sobrecargados puede suponer un problema cuando dicha capacidad no existe en el lenguaje destino.

Un tema más delicado de C++ es el que tiene que ver con el modo de pensar de los programadores a la hora de programar las librerías. En el mundo de SWIG, el programador está realmente tratando de crear componentes software (a nivel binario) para usarlos en otros lenguajes. Para que esto funcione, un "componente" tiene que contener instrucciones ejecutables reales y debe haber algún tipo de mecanismo de vinculación binaria para acceder a su funcionalidad. En contraste, C++ ha confiado cada vez más en la programación genérica y en plantillas para la mayoría de su operatividad. Aunque las plantillas son una característica importante, son irrelevantes al concepto de componentes binarios y librerías. Por ejemplo, un vector STL no define ningún tipo de objeto binario para el que SWIG pueda crear un intérprete. Para complicar un poco más el tema, estas librerías a menudo emplean mucha "magia detrás de las escenas" en la que semánticas de, aparentemente, operaciones básicas (como por ejemplo el referenciado de punteros, llamadas a procedimiento, etc.) pueden ser cambiadas a veces de forma dramática y no tan obvia.

I.3.2 SWIG y Java

En el mundo de los programadores a menudo se necesita sacar provecho de las facilidades de programación que ofrece Java, estando forzados por otro lado a usar algún tipo de código nativo (normalmente en C/C++). La extensión Java que implementa SWIG posibilita de manera muy sencilla indagar en código escrito en C/C++ para su acceso desde Java, puesto que SWIG escribe el código Java Native Interface (JNI) para el programador. SWIG interpreta el código en C/C++ usando clases proxy de Java, lo cual puede resultar de gran utilidad si se desea tener acceso a un montón de código de C/C++ desde Java. SWIG posibilita que un programa en Java pueda llamar fácilmente a código en C/C++ desde Java. SWIG actualmente soporta polimorfismo entre lenguajes y el código puede ser generado para llamar desde C++ a Java cuando se interpretan métodos virtuales de C++.

SWIG trabaja con JDKs desde JDK 1.1 hasta el actual JDK 1.6 (Java 2 SDK 1.6), pudiendo trabajar con cualquier versión posterior. Si fuera posible, se recomienda usar la última versión del JDK de Sun. El módulo de Java en SWIG funciona usando la máquina virtual de java (Java Virtual Machine, JVM) tanto en Solaris, como en Linux, como en varias versiones de Microsoft Windows, incluyendo Cygwin. El código generado también se sabe que funciona en vxWorks usando WindRiver's PJava 3.1. El mejor modo de comprobar que la combinación sistema operativo mas JDK funciona correctamente es intentar ejecutar los ejemplos que vienen con SWIG, teniendo en cuenta que el módulo de Java requiere que el sistema soporte librerías dinámicas y carga dinámica, pues es el método de carga estándar para código JNI.

Anexo II :
DESARROLLOS PREVIOS DE INICIACIÓN A LA TESIS

II. Trabajo previo

Previamente a la ejecución de esta tesis, se ha participado en el desarrollo de dos reconocedores basados en técnicas tradicionales de clasificación utilizando características invariantes extraídas a los bocetos introducidos mediante el análisis de imagen. Estos reconocedores tratan de ajustar el boceto introducido con una geometría determinada e interpretan los símbolos y comandos gestuales establecidos, permitiendo parametrizar la geometría y editar el modelo. A continuación se describen brevemente ambos reconocedores.

II.1 Reconocedor geométrico: RecoGeo

Este reconocedor se encarga del reconocimiento geométrico, esto es, se encarga de convertir bocetos a mano alzada en secciones 2D. La estructura del algoritmo inicialmente desarrollado para este reconocedor se presenta en la Figura II.5. Este algoritmo analiza un *stroke* cuando ocurre un evento del tipo "*pen up*", y lo ajusta a sus primitivas más básicas a modo de polilínea. En la Figura II.4 se puede ver un ejemplo de los resultados de este algoritmo, el cual queda recogido en Fernández-Pacheco et al. [FCN08a].

Este reconocedor fue una primera implementación de un detector de vértices usando la técnica *Mean Shift* [Che05], empleada por otros autores con éxito [Yu03, YC03]. Este reconocedor está basado en la "segmentación" del trazo, esto es, extrae el número de primitivas geométricas que lo componen y clasifica de qué tipo son.

Para la obtención de la segmentación se ha recurrido al empleo de las firmas de dirección y curvatura definidas por Yu [Yu03]. La dirección da el ángulo entre dos puntos consecutivos del gesto en el rango $[-\pi,\pi]$. En la ecuación (II-1), n se define como el número de puntos, k es el número de vecinos en torno al punto n (en nuestro caso se asume $k=1$), y θ es el ángulo entre dos puntos consecutivos en el rango $[-\pi,\pi]$.

$$d_n = \frac{\sum_{i=n-k}^{n+k} \theta(i, i+1)}{2k+1} \qquad \text{(II-1)}$$

Para el cálculo de la curvatura (ver II-2), al igual que en el caso de la dirección, n se define como el número de puntos del trazo, k como el tamaño de los vecinos del punto n del trazo, P es la distancia del actual punto del trazo a sus vecinos (en nuestro caso se asume $k=1$ como el tamaño de la vecindad de cada punto del trazo), y ϕ es la diferencia entre los ángulos de dos pares de puntos consecutivos (siendo en este caso tres los puntos del trazo involucrados).

$$c_n = \frac{\sum_{i=n-k}^{n+k-1} \phi(d_{i+1} - d_i)}{P(n-k, n+k)} \qquad \text{(II-2)}$$

Aunque los datos de velocidad a la hora de introducir un trazo pueden aportar información útil en ocasiones, tal y como se refleja en [CSK02 y Sez01], queda demostrado que al final únicamente introducen ruido en la detección de vértices, como corrige el propio Sezgin en [SSD07]. De un modo breve, el algoritmo de extracción de primitivas o entidades consta de siete pasos principales:

1) Se calculan las firmas de dirección y curvatura.
2) Se suavizan dichas firmas usando la técnica *Mean Shift.*
3) Se intenta aproximar el trazo a alguna de las entidades de punto, círculo o elipse.
4) Si resulta positiva finaliza el reconocedor. En caso negativo se buscan los vértices mediante el algoritmo de detección de vértices que se describe más adelante, dividiendo de esta manera el trazo inicial en varios tramos que se analizarán de forma independiente.
5) Se vuelven a calcular las firmas suavizadas de dirección y curvatura de cada tramo, lo que permite detectar vértices que de otra forma no serían tomados en cuenta.
6) Se intenta aproximar cada tramo a alguna de las entidades de línea o arco, en ese orden, y en los casos en que la aproximación no se consiga se vuelven a buscar vértices y a dividir en nuevos tramos según los vértices hallados.
7) Este procedimiento se ejecuta de forma reiterada hasta que todos los tramos han sido aproximados a una primitiva geométrica.

A continuación se explica en detalle la técnica *Mean Shift* usada en la detección de vértices, así como los métodos de aproximación a entidades que se emplean en cada ciclo de reconocimiento.

La técnica *Mean Shift*

Entre los aspectos principales del procedimiento *Mean Shift*, cabe destacar que permite el filtrado de ruido y la agrupación de datos, ya que eleva de forma iterativa cada punto capturado al promedio del vecindario sin destruir la estructura global del espacio de características. A esto hay que añadir que no requiere de presunciones previas, por lo que resulta idóneo como preprocesado para cualquier etapa de reconocimiento.

La técnica *Mean Shift* se calcula sobre las firmas de dirección (ver ecuación II-1) y curvatura (ver ecuación II-2), y viene definida por Yu [Yu03] como:

$$z_{j+1} = \frac{\sum_{i=1}^{n} x_i k'\left(\left\|\frac{z_j - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} k'\left(\left\|\frac{z_j - x_i}{h}\right\|^2\right)}, j = 1,2,... \qquad \text{(II-3)}$$

Esta técnica se aplica al espacio bidimensional integrando dirección y curvatura, donde se considera que, siendo (*x,y*) el conjunto de *n* puntos del trazo, x_i, *i=1,2…n* se define como el vector de entrada del algoritmo *Mean Shift*, y z_i, *i=1,2…n* como el vector de salida que contiene los valores del espacio integrado d_i y c_i, esto es, que contiene las nuevas firmas de dirección y curvatura tras el proceso de suavizado.

El parámetro *h* es la "anchura de banda", esto es, el parámetro de suavizado, el cual se ha ajustado para h_d y h_c, dependiendo de que se trate de la dirección o de la curvatura. Este parámetro se calcula de la siguiente manera:

$$h_d = \sqrt{\frac{\sum_{i=1}^{n-1} \phi^2 (d_{i+1} - d_i)}{n-1}} \quad h_c = \sqrt{\frac{\sum_{i=1}^{n-1} (c_{i+1} - c_i)^2}{n-1}} \qquad \text{(II-4)}$$

Atendiendo al espacio integrado de dirección y curvatura, se obtiene el vector *mean shift* como:

$$y_{j+1} = \begin{pmatrix} d_{j+1} \\ c_{j+1} \end{pmatrix} = \frac{\sum_{i=1}^{n}\left[x_i . \exp\left(-\frac{(d_j - d_i)^2}{2h_d^2}\right) . \exp\left(-\frac{(c_j - c_i)^2}{2h_c^2}\right)\right]}{\sum_{i=1}^{n}\left[\exp\left(-\frac{(d_j - d_i)^2}{2h_d^2}\right) . \exp\left(-\frac{(c_j - c_i)^2}{2h_c^2}\right)\right]} \qquad \text{(II-5)}$$

El algoritmo basado en la técnica *Mean Shift* queda pues como sigue:

- Inicializar el primer vector resultado $y_{r,1}$ con los máximos valores de dirección y curvatura x_r, *r=1,2…n (esto es,* d_i *y* c_i*).*
- Calcular $y_{r,s+1}$ iterativamente mediante la expresión (II-5) para cada punto del trazo hasta que se cumpla la condición de convergencia. El criterio para dicha convergencia es de $\left|d_{r,s+1} - d_{r,s}\right| \le h_d/100$ para la dirección y de $\left|c_{r,s+1} - c_{r,s}\right| \le h_c/100$ para la curvatura.
- Asignar cada vector resultado convergente a z_r, *r=1,2…n*

El nuevo vector de curvatura suavizado servirá como ayuda para encontrar los vértices en los puntos del trazo cuando se busquen máximos locales, esto es, donde la curvatura del trazo cambia considerablemente (picos en la firma de curvatura).

El algoritmo de detección de vértices

De forma general, los vértices de un trazo siempre están localizados en aquellos puntos donde la curvatura cambia considerablemente, como por ejemplo el punto que conecta una línea con un arco. Como el procedimiento Mean Shift ha eliminado la mayoría del ruido en los datos de curvatura, se puede aplicar esta regla y encontrar así los vértices de forma fidedigna.

Así pues los pasos a seguir para la detección de vértices son los siguientes:

- Se filtran los puntos del trazo y se escogen aquellos que poseen una curvatura en valor absoluto mayor que el valor medio.
- Se busca en dichos puntos seleccionados y se marca como vértice aquél que cumpla los dos criterios siguientes de forma simultánea:
  - Su valor de curvatura es un mínimo o máximo local.
  - Si $|c_i - c_{i-1}| + |c_i - c_{i+1}| \geq 2h_c$. Donde *i* es el punto del trazo actual y h_c es el parámetro de suavizado para la curvatura.
- Finaliza la búsqueda y realiza una función de mezcla de los vértices, fusionando dos vértices consecutivos si la distancia entre ellos es menor que un determinado umbral, tal y como muestra la Figura I.1.

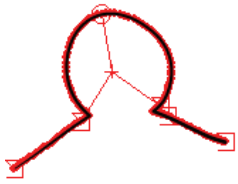


Figura II.1. Vértices encontrados (cuadrados rojos)

En la Figura II.2a se muestra un ejemplo de esbozo del que se han calculado sus firmas de dirección (Figura II.2b superior) y curvatura (Figura II.2b inferior). En la Figura II.2c se muestran estas firmas tras la aplicación de la técnica *Mean Shift* de suavizado, y es en esta última curvatura suavizada sobre la que se buscan los vértices que separan las primitivas que componen la forma o el trazo. Luego, entre cada par de vértices encontrados, se analiza el intervalo correspondiente en la firma de dirección original (Figura II.2b superior) y se aproxima el trozo del trazo a una recta si el tramo es horizontal (no hay cambio de dirección) y a un arco, círculo o elipse si el tramo es oblicuo (cuando hay cambios en la dirección). En la Figura II.2d se observa el resultado del reconocedor, distinguiéndose las entidades extraídas del trazo.

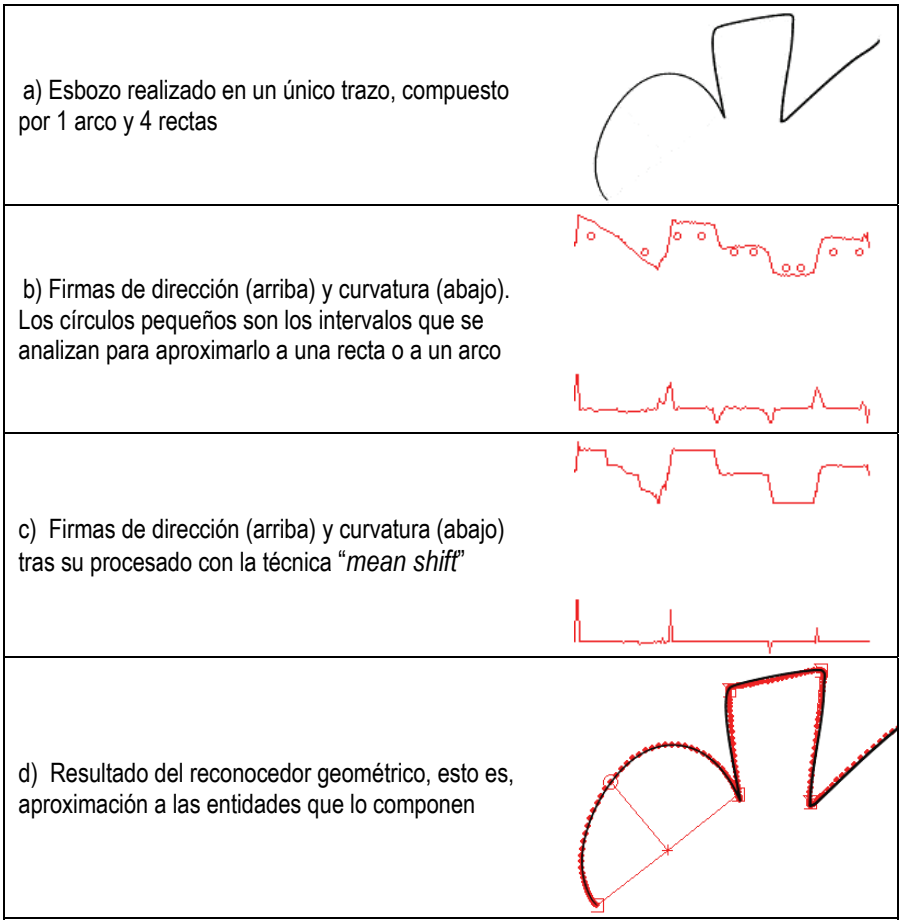


Figura II.2. a) Esbozo realizado; b) Firmas de dirección y curvatura; c) Firmas anteriores suavizadas mediante *mean shift*; d) Resultado de primitivas encontradas

El algoritmo de extracción de primitivas

Dentro de cada ciclo de ejecución se intenta aproximar cada fragmento del trazo entre dos vértices a una primitiva geométrica, siguiendo un orden establecido que asegura su correcto funcionamiento. Así pues la secuencia que se lleva a cabo es la siguiente:

1. Aproximación a PUNTO.

   Esta aproximación únicamente se intenta con el trazo inicial.

   Para aproximar un trazo a una primitiva de punto, se comprueba que sólo existen dos vértices encontrados en todo el trazo, que el número de puntos no supera un mínimo y que la distancia entre los vértices que lo delimitan es menor que un valor determinado.

En el caso de que no cumpla estas condiciones se descarta la posibilidad de que sea un punto y se intenta la siguiente aproximación.

2. Aproximación a CÍRCULO o ELIPSE.

Al igual que en el caso del punto, este tipo de aproximación tan sólo se realiza sobre el trazo inicial, pues resulta evidente que una vez dividido el trazo principal en varios tramos, no podrán encontrarse círculos o elipses en medio del trazo.

Para comprobar si el trazo puede ser aproximado a una primitiva de círculo o elipse, se debe analizar el tramo correspondiente en la gráfica de dirección original, y si se puede aproximar a una línea cuya pendiente sea $2\pi/n$, siendo n el número de puntos del tramo, entonces la aproximación será válida. En este caso, el centro del círculo/elipse vendrá dado por el centro de la caja que encierra al trazo y el radio se obtendrá como la distancia media de cada punto del trazo al centro calculado. Además se calculan otros parámetros como la orientación α (útil para elipses) que será la orientación del diámetro mayor; el diámetro mayor, el diámetro menor, la elongación (diámetro mayor/diámetro menor) y el punto *I* del tramo contenido en el diámetro mayor (útil para elipses). Será precisamente el parámetro de la elongación el que decidirá si la primitiva geométrica a aproximar deberá ser un círculo o una elipse. La Figura II.3b ilustra cómo se obtienen los parámetros de las primitivas de círculo o elipse.

En el caso de no cumplirse las condiciones mencionadas, se continuará con la siguiente aproximación.

3. Aproximación a LÍNEA.

Un tramo (parte del trazo entre dos vértices consecutivos) podrá ser aproximado a una primitiva de línea si, analizado el tramo correspondiente en la gráfica de dirección original, su pendiente resulta horizontal, esto es, próxima al valor cero. En este caso el tramo se aproximará a una línea siendo sus puntos inicial y final los dos vértices que la delimitan. En otro caso, se continúa el proceso.

4. Aproximación a ARCO.

Como último intento de aproximación antes de proceder a calcular vértices y dividir el tramo actual en nuevos tramos basándonos en los vértices hallados, se trata de aproximar el tramo en cuestión a la primitiva geométrica de arco. Para ello se debe analizar el tramo correspondiente en la gráfica de dirección original, y si se puede aproximar a una línea con una determinada pendiente distinta a $2\pi/n$, siendo n el número de puntos del tramo, entonces la aproximación será válida. Los puntos que definen el arco se obtienen de la siguiente manera: se calcula una línea perpendicular a la línea que une los dos vértices y pasa por el punto medio entre estos (Figura II.3a). Luego se busca entre los puntos del tramo aquel que cumple la ecuación de la línea perpendicular, fijando así como los tres puntos que definen el arco los dos vértices y el

punto intermedio encontrado en el tramo. A partir de estos datos se obtiene el radio *r* y el centro *C*.

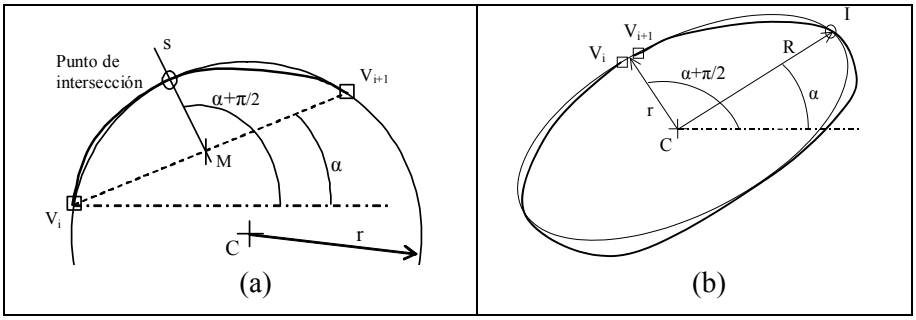


Figura II.3. a) Parámetros del arco de circunferencia calculados a partir de un tramo del esbozo; b) Parámetros del círculo/elipse

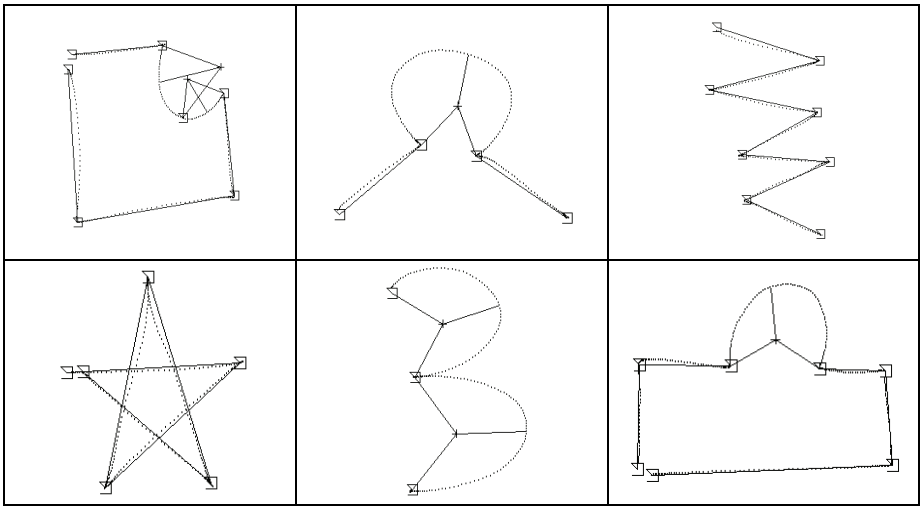


Figura II.4. Ejemplos de resultados del reconocedor RecoGeo

El algoritmo de extracción de primitivas queda de la siguiente manera:

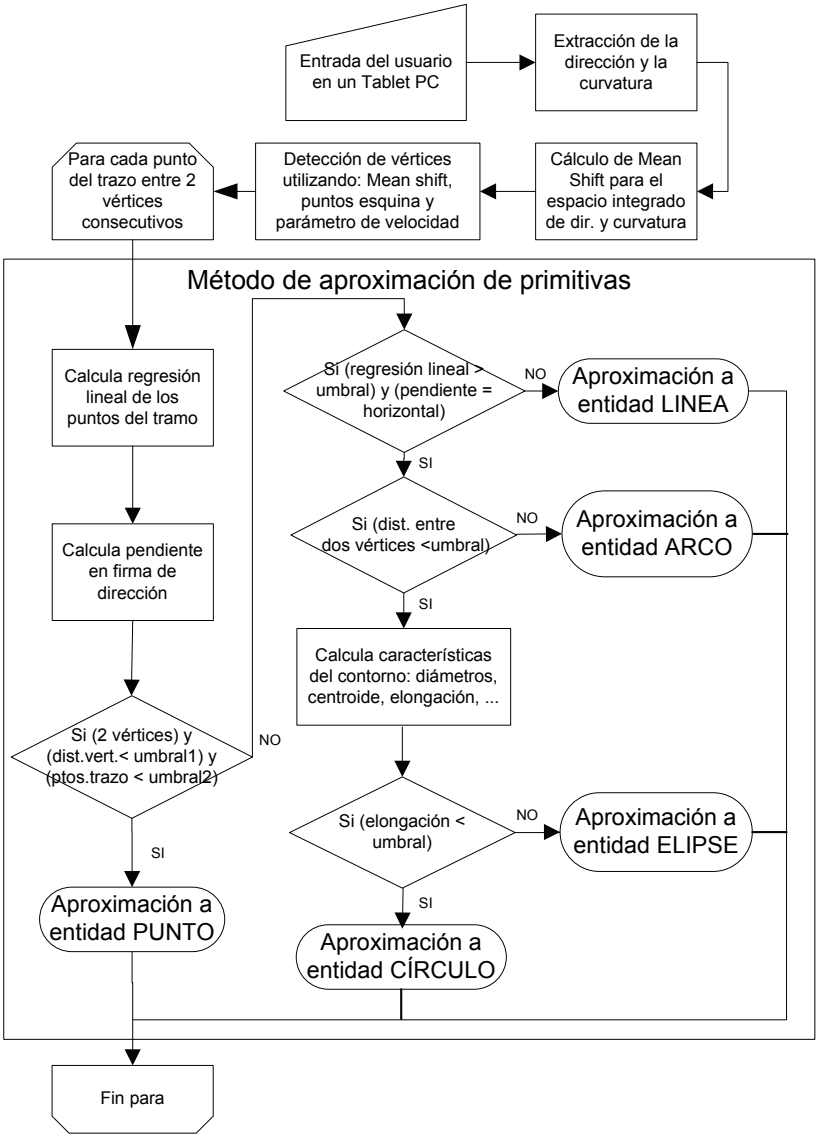


Figura II.5. Diagrama de flujo del reconocedor geométrico RecoGeo

II.2 Reconocedor gestual: RecoGes

Este reconocedor interpreta los símbolos que intervienen en la parametrización de las secciones (restricciones geométricas y dimensionales) y otros comandos gestuales necesarios para la creación de geometría 3D (gesto de borrado, extrusión, revolución etc.). El alfabeto de gestos implementado se muestra en la Tabla II.1.

Tabla II.1. Alfabeto de gestos

Restricciones	Restricciones	Comandos
 Concentricidad	 Vertical	 Extrusión
 Cota	 Horizontal	 Revolución
 Cota diametral	 Paralela	 Borrado
 Tangencia	 Perpendicular	

La estructura del reconocedor gestual se muestra en la Figura II.6, donde se aprecia que el algoritmo consta de dos partes diferentes: un proceso off-line para la recogida del conjunto de entrenamiento que contendrá muestras del alfabeto para la creación del modelo de clasificación (trazos rojos) y un proceso on-line donde los usuarios esbozan gestos del alfabeto y se realiza su interpretación. Atendiendo al proceso off-line, en él se calculan las funciones de maximización utilizando la técnica Bayesiana de análisis discriminante no lineal. Atendiendo al proceso on-line, el algoritmo principal consta de cuatro pasos. En el primer paso se utilizan técnicas de procesamiento de imagen para suavizar y eliminar el ruido de los bocetos. En el segundo paso se realiza la extracción del contorno. En el tercer paso se calculan los descriptores a partir del contorno (la transformada rápida de Fourier FFT, los momentos de Hu y otras características invariantes a la escala y la orientación). Y como paso final, se utilizan las funciones de maximización calculadas en el proceso off-line para clasificar el dibujo atendiendo a los descriptores calculados.

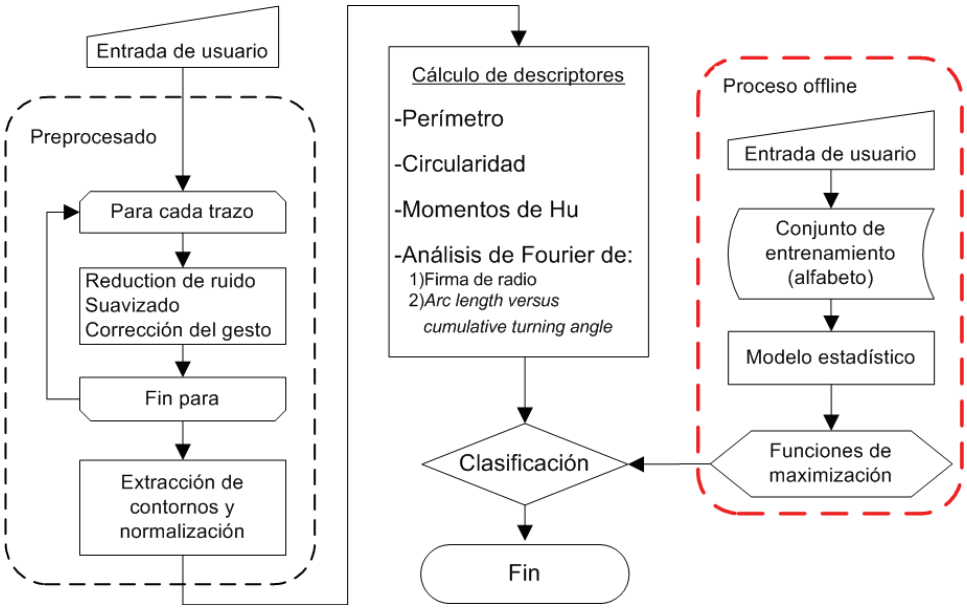
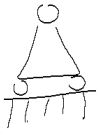
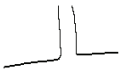


Figura II.6. Estructura del reconocedor RecoGes

El reconocedor se ha probado en la clasificación de 2590 bocetos recolectados por medio de un Tablet PC a partir de 9 usuarios diferentes, mostrando sus resultados una tasa promedio de éxito del 94,87% para dichos bocetos, aunque este porcentaje baja considerablemente cuando un nuevo usuario introduce el gesto o cuando se aumenta el catálogo de gestos/símbolos a reconocer, teniendo el problema añadido de que clasifica siempre el boceto introducido aunque éste no pertenezca al catálogo de símbolos aceptados. Este trabajo se recoge en Fernández-Pacheco et al. [FCN08b].

Una variante de este reconocedor se ha desarrollado para la interpretación de símbolos bocetados de pre-procesado CAE para análisis de estructuras, particularmente estructuras tipo barra en 2D, trabajo que se puede consultar en Company et al. [CAN08]. El alfabeto de símbolos implementados se muestra en la Tabla II.2.

Tabla II.2. Alfabeto de símbolos para análisis de estructuras

 Nodo	 Nodo Móvil	 Carga Continua
 Nodo Fijo	 Momento	 Perfil en U
 Nodo Fijo 2	 Carga	 Perfil en L

Anexo III:
EL MÉTODO TCVD

III. El método TCVD

III.1 Diagrama de bloques

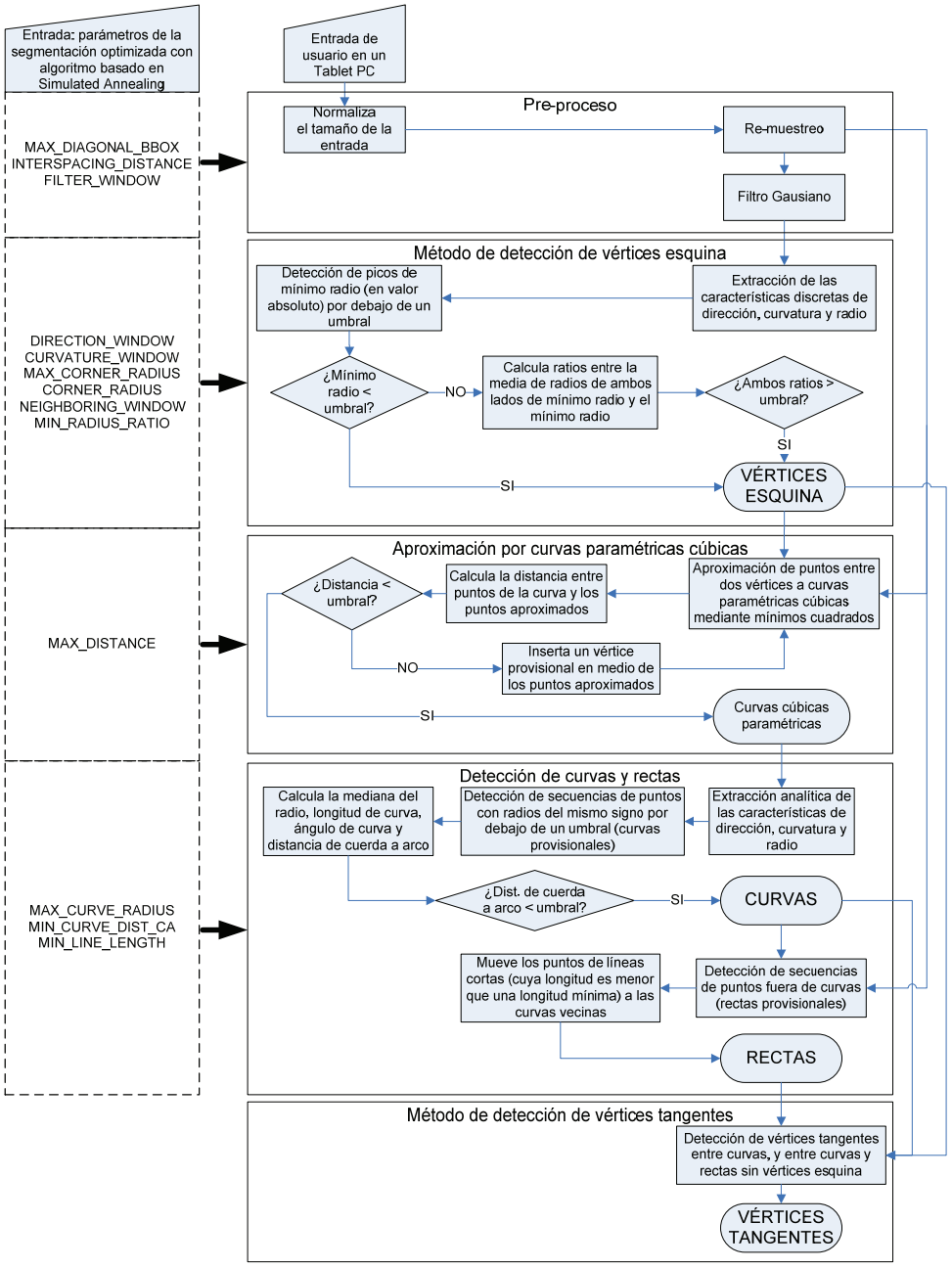


Figura III.1. Diagrama de funcionamiento del algoritmo TCVD

III.2 Entrada y salida

La entrada está constituida por los *n* puntos originales de un trazo $po_i = (xo_i, yo_i),\ i \in [0, n-1]$, mientras que la salida está formada por un vector $eo_i = eo_0, eo_1, ..., eo_{n-1}$ que contiene el tipo de entidad (vértice esquina, vértice tangente, recta o curva) que le corresponde a cada punto original del trazo.

III.3 Normalización del tamaño de la entrada

La normalización tiene como objetivo evitar la presencia de trazos de gran tamaño, con curvas de radios demasiado grandes que pueden ser confundidas con rectas porque superan el umbral empleado para los radios de curvas, estos trazos son escalados para reducir su tamaño.

El caso contrario (trazos de pequeño tamaño con curvas de radios muy pequeños) puede causar la confusión de curvas con vértices esquina, sin embargo, no admite solución, puesto que al aumentar su tamaño se separan linealmente los puntos originales pero no se añaden nuevos puntos, por lo que el trazo es como un polígono que va aumentando el tamaño de sus lados pero manteniéndose las esquinas entre ellos. Por tanto, es una restricción del método que los trazos no deben ser demasiado pequeños.

El método empleado para normalizar la entrada consiste en calcular la caja de inclusión (*bounding box*) de los puntos del trazo y obtener el tamaño de la diagonal (*DIAGONAL*) y si ésta supera el valor del parámetro *MAX_DIAGONAL_BBOX*, se obtiene un factor de escala (*DIAGONAL* / *MAX_DIAGONAL_BBOX*) que multiplica las coordenadas de los puntos de la entrada. En caso de que la diagonal no supere el valor del parámetro, el factor de escala es 1. El resultado es el conjunto de puntos normalizados $pn_i = (xn_i, yn_i),\ i \in [0, n-1]$.

III.4 Remuestreo

Para remuestrear el trazo, se emplea el método descrito en [WEH08], que proviene de [WWL07], aunque el valor de la distancia entre puntos re-muestreados (*INTERSPACING_DISTANCE*) es fijo, en lugar de depender del tamaño del trazo.

El resultado es el conjunto de puntos re-muestreados $pr = (xr_i, yr_i),\ i \in [0, m-1]$, siendo generalmente el número de puntos re-muestreados (*m*) distinto del número de puntos originales (*n*). El método se describe a continuación:

El primer punto re-muestreado (pr_1) es el primer punto normalizado (pn_1)

Inicializar la distancia acumulada (DACUM) a 0

Recorrer los puntos normalizados desde el segundo al último

> *Calcular la distancia (DIST) del punto normalizado actual (pn_i) al anterior (pn_{i-1})*

Sumarla a la distancia acumulada

Si la distancia acumulada es mayor o igual que el parámetro INTERSPACING_DISTANCE:

Obtener DIF=(DACUM - INTERSPACING DISTANCE)

Añadir un punto remuestreado entre pn_i y pn_{i-1}:

$pr_j \leftarrow [pn_{i-1} \times (DIF / DIST)] + [pn_i \times (1 - DIF / DIST)]$

Inicializar la distancia acumulada a DIF

Avanzar un punto normalizado extra ($i \leftarrow i+1$)

III.5 Filtrado gaussiano

Aplicar un filtrado lineal [JSK95] a una secuencia de puntos consiste en aplicar una convolución, por separado, a las coordenadas *x* e *y* de los puntos (consideradas como dos señales unidimensionales) con un vector constante (f_h) llamado máscara o *kernel*. El filtrado gaussiano es un tipo de filtrado lineal muy utilizado debido a sus buenos resultados, en el que el kernel sigue la forma de una función gaussiana. El resultado es el conjunto de puntos filtrados $pf_j = (xf_j, yf_j),\ j \in [0, m-1]$, donde cada punto filtrado contiene un promedio de los puntos re-muestreados en su vecindad:

$$xf_j = \frac{\sum_{h=-w}^{w} f_h \cdot xr_{j-h}}{\sum_{h=-w}^{w} f_h} \qquad yf_j = \frac{\sum_{h=-w}^{w} f_h \cdot yr_{j-h}}{\sum_{h=-w}^{w} f_h}$$

En las expresiones anteriores, el tamaño de la máscara es (*2w+1*) siendo *w* el valor del parámetro *FILTER_WINDOW*; y el kernel se obtiene de una función gaussiana (sin el factor constante) de media cero y desviación típica $\sigma = w$, para que el ancho de la ventana determine la intensidad del filtrado [Tru98]:

$$f_h = e^{\left(-\frac{\pi \cdot h^2}{2 \cdot w^2}\right)}$$

III.6 Extracción discreta de las características de dirección, curvatura y radio

III.6.1 Dirección

La dirección del trazo (α_j) en cada punto filtrado es el ángulo de la tangente al trazo que se obtiene, de forma aproximada, mediante la derivada discreta en cada punto del trazo (variación de la posición en la vecindad de cada punto).

$$\left.\begin{aligned} xf_j' &= \lim_{h\to 0}\frac{xf_{j+h}-xf_{j-h}}{2\cdot h} = \frac{xf_{j+w}-xf_{j-w}}{2\cdot w\cdot s} \\ yf_j' &= \lim_{h\to 0}\frac{yf_{j+h}-yf_{j-h}}{2\cdot h} = \frac{yf_{j+w}-yf_{j-w}}{2\cdot w\cdot s} \end{aligned}\right\},\ \alpha_j = \arctan\left(\frac{yf_j'}{xf_j'}\right) = \arctan\left(\frac{yf_{j+w}-yf_{j-w}}{xf_{j+w}-xf_{j-w}}\right)$$

En la expresión anterior, *h* es un valor genérico de distancia medida sobre la función del trazo que en la práctica es *w·s*, donde *s* es la distancia entre puntos re-muestreados (*INTERSPACING_DISTANCE*) y *w* el ancho de la ventana para la obtención de tangentes (parámetro *DIRECTION_WINDOW*), es decir, la diferencia de los índices de los puntos anterior y posterior utilizados para el cálculo de la tangente.

Es necesario tener en cuenta que la función de los valores de ángulos está en el rango [-π, π] y puede presentar saltos entre valores consecutivos debido al carácter cíclico de los ángulos, por lo que se realiza una corrección de cada valor en función del anterior:

Mientras [(α_j - α_{j-1})<(-π)] hacer ($\alpha_j \leftarrow \alpha_j + 2\cdot\pi$)

Mientras [(α_j - α_{j-1})>(π)] hacer ($\alpha_j \leftarrow \alpha_j - 2\cdot\pi$)

III.6.2 Curvatura

La curvatura del trazo (c_j) es la derivada de la tangente, que se obtiene, de forma aproximada, mediante la derivada discreta (variación de la dirección en la vecindad de cada punto del trazo).

$$c_j = \alpha_j' = \frac{\alpha_{j+w}-\alpha_{j-w}}{2\cdot w\cdot s}$$

Donde *s* es la distancia entre puntos re-muestreados (*INTERSPACING_DISTANCE*) y *w* el ancho de la ventana para la obtención de curvaturas (parámetro *CURVATURE_WINDOW*), es decir, la diferencia de los índices de los puntos anterior y posterior utilizados para el cálculo de la curvatura.

III.6.3 Radio

El radio en cada punto del trazo (r_j) es la inversa de la curvatura en cada punto del trazo. Para evitar divisiones por cero, cuando el valor absoluto de la curvatura es inferior a un valor mínimo, el radio se fija a un valor muy alto, con el mismo signo que la curvatura.

$$r_j = \frac{1}{c_j}$$

III.7 Detección de vértices esquina

Los vértices esquina se sitúan en mínimos locales (en valor absoluto) de la función de radio, es decir, en puntos con la máxima curvatura, o máxima variación de la dirección del

trazo. Dichos mínimos locales deberán estar (en valor absoluto) por debajo del parámetro *MAX_CORNER_RADIUS*.

Los mínimos locales candidatos a vértices esquina, lo serán si cumplen alguna de estas condiciones:

- El mínimo está por debajo del parámetro *CORNER_RADIUS*.
- El pico que forma el mínimo es muy acusado, es decir, el aumento del radio es grande en el entorno del mínimo local. Para comprobarlo se obtiene (sólo para radios del mismo signo) el radio medio en los *NEIGHBORING_WINDOW* puntos anteriores y el radio medio en los *NEIGHBORING_WINDOW* puntos posteriores al radio mínimo. Los cocientes de las divisiones de ambos radios medios y el radio mínimo deben ser mayores que *MIN_RADIUS_RATIO* para que sea una esquina.

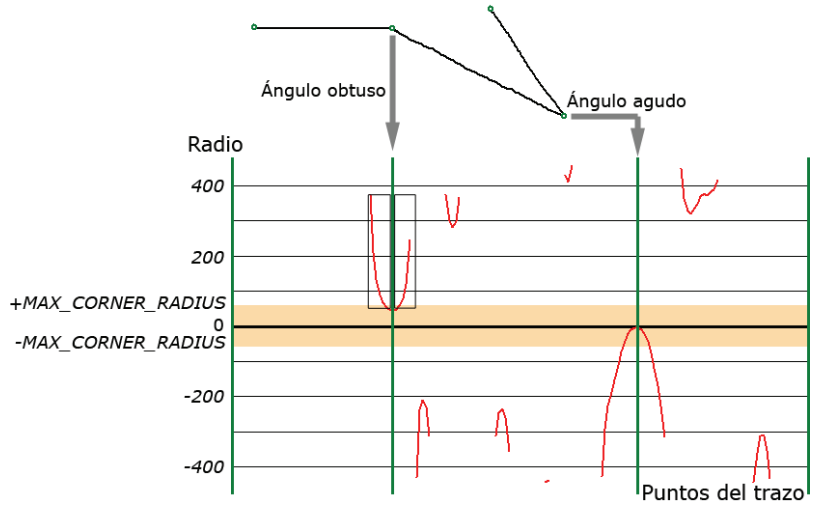


Figura III.2. Trazo con dos vértices esquina, en ángulo obtuso (enmarcando con dos rectángulos los entornos izquierdo y derecho del mínimo local) y en ángulo agudo (sólo se han representado los radios –color rojo– con valor absoluto menor que 500)

Además de los vértices esquina obtenidos a partir de los radios mínimos locales, siempre se consideran vértices esquina los puntos inicial y final del trazo.

III.8 Aproximación por curvas cúbicas paramétricas

La aproximación de los puntos re-muestreados mediante curvas cúbicas paramétricas tiene por objeto un cálculo más exacto (en forma analítica) de los valores de radios para obtener con mayor precisión posteriormente las curvas del trazo.

Las pruebas realizadas muestran que el ruido causado por el *aliasing* debido a la cuadrícula del ráster afecta más a las curvas que a los vértices esquina, y que la obtención analítica de los radios a partir de las curvas cúbicas paramétricas es más precisa que la obtención discreta realizando un filtrado gaussiano de mayor intensidad.

Por otra parte, no es aconsejable realizar la aproximación mediante curvas cúbicas paramétricas antes de obtener los vértices esquina, ya que el mayor error de la aproximación siempre aparece en dicho tipo de vértices (porque estamos realizando transiciones suaves en los vértices esquina) distorsionando los valores de la función de radio en los mismos.

Para evitar los errores de aproximación en los vértices esquina, las curvas cúbicas paramétricas aproximan secuencias de puntos re-muestreados comprendidas entre pares de vértices esquina, de forma que, en un vértice esquina siempre se cambia de una curva cúbica paramétrica a otra. Las curvas tienen la restricción de pasar obligatoriamente por los puntos inicial y final de la secuencia a aproximar.

En caso de que la curva obtenida supere el parámetro de distancia máxima (*MAX_DISTANCE*) a algún punto re-muestreado, la secuencia de puntos a aproximar se dividirá en dos por el punto central, y para conservar la continuidad de la curva se añadirá la restricción de que las dos curvas nuevas, además de pasar por el punto central, tengan la misma dirección en dicho punto, es decir, la misma tangente. Al aplicar este proceso varias veces, se obtienen cuatro formas distintas de aproximar una secuencia de puntos mediante una curva cúbica paramétrica (ver tabla III.1), en función de la cantidad de restricciones que deba cumplir la curva.

Tabla III.1. Restricciones que deben cumplir las curvas

La curva pasa por		**La curva tiene la tangente en**	
Punto inicial	Punto final	Punto inicial	Punto final
X	X		
X	X	X	
X	X		X
X	X	X	X

La tangente empleada cuando se divide una secuencia de puntos re-muestreados por su punto medio en otras dos, procede de la dirección del trazo en dicho punto calculada mediante las derivadas discretas:

$$\left.\begin{aligned} xr_j' &= k \cdot \cos(\alpha_j) \\ yr_j' &= k \cdot \sin(\alpha_j) \end{aligned}\right\}$$

Siendo *k* el módulo del vector tangente, que se deja libre para que el sistema de mínimos cuadrados obtenga el valor que mejor aproxime los puntos re-muestreados utilizando esa dirección de tangente.

III.8.1 Aproximación de puntos mediante curvas cúbicas paramétricas mediante mínimos cuadrados

Una curva cúbica paramétrica está constituida por dos polinomios de tercer grado, uno para las coordenadas *x* y otro para las coordenadas *y*. Los dos polinomios tienen 4 coeficientes cada uno (en total 8 grados de libertad) y dependen de un parámetro *t*, cuyo valor es 0 al principio de la curva y 1 al final de la misma. Las expresiones de una curva cúbica paramétrica y su derivada son las siguientes:

$$\left.\begin{array}{ll} x(t) = a_x + b_x \cdot t + c_x \cdot t^2 + d_x \cdot t^3 & x'(t) = b_x + 2 \cdot c_x \cdot t + 3 \cdot d_x \cdot t^2 \\ y(t) = a_y + b_y \cdot t + c_y \cdot t^2 + d_y \cdot t^3 & y'(t) = b_y + 2 \cdot c_y \cdot t + 3 \cdot d_y \cdot t^2 \end{array}\right\}, \; t \in [0,1]$$

La curva debe aproximarse lo máximo posible a cada uno de los puntos re-muestreados, es decir, se pretende que para cada punto $pr_j=(xr_j,yr_j)$, exista un t_j tal que:

$$\left.\begin{array}{l} a_x + b_x \cdot t_j + c_x \cdot t_j^2 + d_x \cdot t_j^3 = xr_j \\ a_y + b_y \cdot t_j + c_y \cdot t_j^2 + d_y \cdot t_j^3 = yr_j \end{array}\right\}$$

Repitiendo las expresiones anteriores para las coordenadas *x* e *y* de cada punto re-muestreado, se construyen dos sistemas lineales de ecuaciones, uno para las *x* y otro para las *y*. Cada uno de los sistemas tiene 4 incógnitas (los coeficientes *a*, *b*, *c* y *d* correspondientes) [Far93]. Como lo normal es que haya más de 4 puntos para aproximar, los sistemas están sobredimensionados, y se deben resolver por métodos de mínimos cuadrados para obtener la solución que mejor aproxime los *m* puntos re-muestreados.

$$\left.\begin{array}{c} a_x + b_x \cdot t_0 + c_x \cdot t_0^2 + d_x \cdot t_0^3 = xr_0 \\ a_x + b_x \cdot t_1 + c_x \cdot t_1^2 + d_x \cdot t_1^3 = xr_1 \\ \cdots \\ a_x + b_x \cdot t_{m-1} + c_x \cdot t_{m-1}^2 + d_x \cdot t_{m-1}^3 = xr_{m-1} \end{array}\right\}$$

Que en forma matricial queda de la siguiente manera:

$$\begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_1 & t_1^2 & t_1^3 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & t_{m-1} & t_{m-1}^2 & t_{m-1}^3 \end{pmatrix} \begin{pmatrix} a_x \\ b_x \\ c_x \\ d_x \end{pmatrix} = \begin{pmatrix} xr_0 \\ xr_1 \\ \cdots \\ xr_{m-1} \end{pmatrix}$$

Los valores del parámetro t_j para cada punto re-muestreado pr_j son desconocidos y se calculan de manera aproximada suponiendo que son proporcionales a las distancias entre los puntos re-muestreados:

Asignar a al primer punto re-muestreado (pr_0) una distancia $d_0 = 0$

Asignar a cada punto del resto (pr_j) la distancia euclídea al punto anterior *(pr_{j-1}): d_j =|| pr_j - pr_{j-1} ||*

Obtener las distancias acumuladas (da_j) de cada pr_j a pr_0: $da_j = d_0 + d_1 + ... + d_j$

Obtener los valores de los parámetros proporcionalmente a las distancias $t_j = (da_j / a_m)$

Las restricciones indicadas previamente restan grados de libertad a las curvas paramétricas, es decir, reducen el número de parámetros y, por tanto, el número de incógnitas de los sistemas de ecuaciones. Las restricciones son:

- La curva pasa por el punto inicial (se restan 2 grados de libertad pero podemos eliminar del sistema de mínimos cuadrados la ecuación de aproximación al punto inicial):

$$t = t_0 = 0 \rightarrow \begin{cases} a_x + b_x \cdot 0 + c_x \cdot 0^2 + d_x \cdot 0^3 = xr_0 & \rightarrow \quad a_x = xr_0 \\ a_y + b_y \cdot 0 + c_y \cdot 0^2 + d_y \cdot 0^3 = yr_0 & \rightarrow \quad a_y = yr_0 \end{cases}$$

- La curva pasa por el punto final (se restan 2 grados de libertad pero podemos eliminar del sistema de mínimos cuadrados la ecuación de aproximación al punto final):

$$t = t_{m-1} = 1 \rightarrow \begin{cases} a_x + b_x \cdot 1 + c_x \cdot 1^2 + d_x \cdot 1^3 = xr_{m-1} & \rightarrow \quad a_x + b_x + c_x + d_x = xr_{m-1} \\ a_y + b_y \cdot 1 + c_y \cdot 1^2 + d_y \cdot 1^3 = yr_{m-1} & \rightarrow \quad a_y + b_y + c_y + d_y = yr_{m-1} \end{cases}$$

- La curva tiene la tangente indicada en el punto inicial *(xr_0',yr_0')*. Esta restricción afecta a la dirección de la tangente, no al módulo, por lo que sólo se reduce un grado de libertad :

$$t = t_0 = 0 \rightarrow \begin{cases} b_x + 2 \cdot c_x \cdot 0 + 3 \cdot d_x \cdot 0^2 = k_0 \cdot xr_0' & \rightarrow \quad \mathrm{b}_x = k_0 \cdot xr_0' \\ b_y + 2 \cdot c_y \cdot 0 + 3 \cdot d_y \cdot 0^2 = k_0 \cdot yr_0' & \rightarrow \quad \mathrm{b}_y = k_0 \cdot yr_0' \end{cases}$$

- La curva tiene la tangente indicada en el punto final *(xr_{m-1}',yr_{m-1}')*. Esta restricción afecta a la dirección de la tangente, no al módulo, por lo que sólo se reduce un grado de libertad :

$$t = t_{m-1} = 1 \rightarrow \begin{cases} b_x + 2 \cdot c_x \cdot 1 + 3 \cdot d_x \cdot 1^2 = k_{m-1} \cdot xr_{m-1}' & \rightarrow \quad \mathrm{b}_x + 2 \cdot c_x + 3 \cdot d_x = k_{m-1} \cdot xr_{m-1}' \\ b_y + 2 \cdot c_y \cdot 1 + 3 \cdot d_y \cdot 1^2 = k_{m-1} \cdot yr_{m-1}' & \rightarrow \quad \mathrm{b}_y + 2 \cdot c_y + 3 \cdot d_y = k_{m-1} \cdot yr_{m-1}' \end{cases}$$

En función de cuantas restricciones se apliquen, los sistemas de ecuaciones lineales serán diferentes y tendrán más o menos grados de libertad: desde 4 (sólo con las restricciones de paso por puntos inicial y final) a 2 (con las restricciones de paso y las direcciones de tangentes por los puntos inicial y final).

III.8.2 Curvas sin tangente inicial ni final

Sustituyendo las restricciones en la expresión de las curvas cúbicas paramétricas se llega a los dos sistemas de ecuaciones siguientes:

$$\begin{pmatrix} (t_1^2 - t_1) & (t_1^3 - t_1) \\ (t_2^2 - t_2) & (t_2^3 - t_2) \\ \dots & \dots \\ (t_{m-2}^2 - t_{n-2}) & (t_{m-2}^3 - t_{m-2}) \end{pmatrix} \begin{pmatrix} c_x \\ d_x \end{pmatrix} = \begin{pmatrix} xr_1 - xr_0 + xr_0 \cdot t_1 - xr_{m-1} \cdot t_1 \\ xr_2 - xr_0 + xr_0 \cdot t_2 - xr_{m-1} \cdot t_2 \\ \dots \\ xr_{m-2} - xr_0 + xr_0 \cdot t_{m-2} - xr_{m-1} \cdot t_{m-2} \end{pmatrix}$$

$$\begin{pmatrix} (t_1^2 - t_1) & (t_1^3 - t_1) \\ (t_2^2 - t_2) & (t_2^3 - t_2) \\ \dots & \dots \\ (t_{m-2}^2 - t_{m-2}) & (t_{m-2}^3 - t_{m-2}) \end{pmatrix} \begin{pmatrix} c_y \\ d_y \end{pmatrix} = \begin{pmatrix} yr_1 - yr_0 + yr_0 \cdot t_1 - yr_{m-1} \cdot t_1 \\ yr_2 - yr_0 + yr_0 \cdot t_2 - yr_{m-1} \cdot t_2 \\ \dots \\ yr_{m-2} - yr_0 + yr_0 \cdot t_{m-2} - yr_{m-1} \cdot t_{m-2} \end{pmatrix}$$

Siendo:

$$\begin{cases} a_x = xr_0 \\ a_y = yr_0 \end{cases} \quad \begin{cases} b_x = xr_{m-1} - a_x - c_x - d_x \\ b_y = yr_{m-1} - a_y - c_y - d_y \end{cases}$$

III.8.3 Curvas con tangente inicial

Sustituyendo las restricciones en la expresión de las curvas cúbicas paramétricas se llega al sistema de ecuaciones siguiente. En este caso es un único sistema ya que existe una incógnita (k_0) que está tanto en el sistema de la *x* como en el de la *y*, por lo que ambos se combinan:

$$\begin{pmatrix} xr_0' \cdot (t_1 - t_1^2) & (t_1^3 - t_1^2) & 0 \\ yr_0' \cdot (t_1 - t_1^2) & 0 & (t_1^3 - t_1^2) \\ xr_0' \cdot (t_2 - t_2^2) & (t_2^3 - t_2^2) & 0 \\ yr_0' \cdot (t_2 - t_2^2) & 0 & (t_2^3 - t_2^2) \\ \dots & \dots & \dots \\ xr_0' \cdot (t_{m-2} - t_{m-2}^2) & (t_{m-2}^3 - t_{m-2}^2) & 0 \\ yr_0' \cdot (t_{m-2} - t_{m-2}^2) & 0 & (t_{m-2}^3 - t_{m-2}^2) \end{pmatrix} \begin{pmatrix} k_0 \\ d_x \\ d_y \end{pmatrix} = \begin{pmatrix} x_1 - x_0 - (x_{m-1} - x_0) \cdot t_1^2 \\ y_1 - y_0 - (y_{m-1} - y_0) \cdot t_1^2 \\ x_2 - x_0 - (x_{m-1} - x_0) \cdot t_2^2 \\ y_2 - y_0 - (y_{m-1} - y_0) \cdot t_2^2 \\ \dots \\ x_{m-2} - x_0 - (x_{m-1} - x_0) \cdot t_{m-2}^2 \\ y_{m-2} - y_0 - (y_{m-1} - y_0) \cdot t_{m-2}^2 \end{pmatrix}$$

Siendo:

$$\begin{cases} a_x = xr_0 \\ a_y = yr_0 \end{cases} \quad \begin{cases} b_x = k_0 \cdot xr_0' \\ b_y = k_0 \cdot yr_0' \end{cases} \quad \begin{cases} c_x = xr_{m-1} - a_x - b_x - d_x \\ c_y = yr_{m-1} - a_y - b_y - d_y \end{cases}$$

III.8.4 Curvas con tangente final

Sustituyendo las restricciones en la expresión de las curvas cúbicas paramétricas se llega al sistema de ecuaciones siguiente. Al igual que para las curvas con tangente inicial, también es un único sistema ya que existe una incógnita (k_{m-1}) que está tanto en el sistema de la *x* como en el de la *y*:

$$\begin{pmatrix} x_{m-1}'\cdot(t_1^2-t_1) & (t_1-2\cdot t_1^2+t_1^3) & 0 \\ y_{m-1}'\cdot(t_1^2-t_1) & 0 & (t_1-2\cdot t_1^2+t_1^3) \\ x_{m-1}'\cdot(t_2^2-t_2) & (t_2-2\cdot t_2^2+t_2^3) & 0 \\ y_{m-1}'\cdot(t_2^2-t_2) & 0 & (t_2-2\cdot t_2^2+t_2^3) \\ \cdots & \cdots & \cdots \\ x_{m-1}'\cdot(t_{m-2}^2-t_{m-2}) & (t_{m-2}-2\cdot t_{m-2}^2+t_{m-2}^3) & 0 \\ y_{m-1}'\cdot(t_{m-2}^2-t_{m-2}) & 0 & (t_{m-2}-2\cdot t_{m-2}^2+t_{m-2}^3) \end{pmatrix} \begin{pmatrix} k_{m-1} \\ d_x \\ d_y \end{pmatrix} = \begin{pmatrix} x_1-x_0-2\cdot(x_{m-1}-x_0)\cdot t_1+(x_{m-1}-x_0)\cdot t_1^2 \\ y_1-y_0-2\cdot(y_{m-1}-y_0)\cdot t_1+(y_{m-1}-y_0)\cdot t_1^2 \\ x_2-x_0-2\cdot(x_{m-1}-x_0)\cdot t_2+(x_{m-1}-x_0)\cdot t_2^2 \\ y_2-y_0-2\cdot(y_{m-1}-y_0)\cdot t_2+(y_{m-1}-y_0)\cdot t_2^2 \\ \cdots \\ x_{m-2}-x_0-2\cdot(x_{m-1}-x_0)\cdot t_{m-2}+(x_{m-1}-x_0)\cdot t_{m-2}^2 \\ y_{m-2}-y_0-2\cdot(y_{m-1}-y_0)\cdot t_{m-2}+(y_{m-1}-y_0)\cdot t_{m-2}^2 \end{pmatrix}$$

Siendo:

$$\left.\begin{matrix} a_x = xr_0 \\ a_y = yr_0 \end{matrix}\right\} \quad \left.\begin{matrix} b_x = -k_{m-1}\cdot xr_{m-1}'+2\cdot xr_{m-1}-2\cdot a_x+d_x \\ b_y = -k_{m-1}\cdot yr_{m-1}'+2\cdot yr_{m-1}-2\cdot a_y+d_y \end{matrix}\right\} \quad \begin{matrix} c_x = -xr_{m-1}+a_x+k_{m-1}\cdot xr_{m-1}'-2\cdot d_x \\ c_y = -yr_{m-1}+a_y+k_{m-1}\cdot yr_{m-1}'-2\cdot d_y \end{matrix}$$

III.8.5 Curvas con tangente inicial y final

Sustituyendo las restricciones en la expresión de las curvas cúbicas paramétricas se llega al sistema de ecuaciones siguiente. Al igual que para los casos anteriores, también es un único sistema ya que existen, en este caso, dos incógnitas (k_0 y k_{m-1}) que están tanto en el sistema de la *x* como en el de la *y*:

$$\begin{pmatrix} (xr_0'\cdot t_1-2\cdot xr_0'\cdot t_1^2+xr_0'\cdot t_1^3) & (xr_{m-1}'\cdot t_1^3-xr_{m-1}'\cdot t_1^2) \\ (yr_0'\cdot t_1-2\cdot yr_0'\cdot t_1^2+yr_0'\cdot t_1^3) & (yr_{m-1}'\cdot t_1^3-yr_{m-1}'\cdot t_1^2) \\ (xr_0'\cdot t_2-2\cdot xr_0'\cdot t_2^2+xr_0'\cdot t_2^3) & (xr_{m-1}'\cdot t_2^3-xr_{m-1}'\cdot t_2^2) \\ (yr_0'\cdot t_2-2\cdot yr_0'\cdot t_2^2+yr_0'\cdot t_2^3) & (yr_{m-1}'\cdot t_2^3-yr_{m-1}'\cdot t_2^2) \\ \cdots & \cdots \\ (xr_0'\cdot t_{m-2}-2\cdot xr_0'\cdot t_{m-2}^2+xr_0'\cdot t_{m-2}^3) & (xr_{m-1}'\cdot t_{m-2}^3-xr_{m-1}'\cdot t_{m-2}^2) \\ (yr_0'\cdot t_{m-2}-2\cdot yr_0'\cdot t_{m-2}^2+yr_0'\cdot t_{m-2}^3) & (yr_{m-1}'\cdot t_{m-2}^3-y_{m-1}'\cdot t_{m-2}^2) \end{pmatrix} \begin{pmatrix} k_0 \\ k_{m-1} \end{pmatrix} = \begin{pmatrix} xr_1-xr_0-3\cdot(xr_{m-1}-xr_0)\cdot t_1^2+2\cdot(xr_{m-1}-xr_0)\cdot t_1^3 \\ yr_1-yr_0-3\cdot(yr_{m-1}-yr_0)\cdot t_1^2+2\cdot(yr_{m-1}-yr_0)\cdot t_1^3 \\ xr_2-xr_0-3\cdot(xr_{m-1}-xr_0)\cdot t_2^2+2\cdot(xr_{m-1}-xr_0)\cdot t_2^3 \\ yr_2-yr_0-3\cdot(yr_{m-1}-yr_0)\cdot t_2^2+2\cdot(yr_{m-1}-yr_0)\cdot t_2^3 \\ \cdots \\ xr_{m-2}-xr_0-3\cdot(xr_{m-1}-xr_0)\cdot t_{m-2}^2+2\cdot(xr_{m-1}-xr_0)\cdot t_{m-2}^3 \\ yr_{m-2}-yr_0-3\cdot(yr_{m-1}-yr_0)\cdot t_{m-2}^2+2\cdot(yr_{m-1}-yr_0)\cdot t_{m-2}^3 \end{pmatrix}$$

Siendo:

$$\left.\begin{matrix} a_x = xr_0 \\ a_y = yr_0 \end{matrix}\right\} \quad \left.\begin{matrix} b_x = k_0\cdot xr_0' \\ b_y = k_0\cdot yr_0' \end{matrix}\right\} \quad \left.\begin{matrix} c_x = 3\cdot(xr_{m-1}-a_x)-k_{m-1}\cdot x_{m-1}'-2\cdot b_x \\ c_y = 3\cdot(yr_{m-1}-b_x)-k_{m-1}\cdot y_{m-1}'-2\cdot b_y \end{matrix}\right\} \quad \begin{matrix} d_x = k_{m-1}\cdot xr_{m-1}'+b_x-2\cdot(xr_{m-1}-a \\ d_y = k_{m-1}\cdot yr_{m-1}'+b_y-2\cdot(yr_{m-1}-a \end{matrix}$$

III.8.6 Resolución de los sistemas de ecuaciones

Para resolver los sistemas de ecuaciones lineales se puede emplear cualquier método de mínimos cuadrados, pero en nuestro caso se ha preferido utilizar el método de Householder, más estable numéricamente que el método convencional [BF85] [Gol91].

Para verificar la precisión de la aproximación realizada, se comprobará que la distancia máxima entre los puntos re-muestreados (pr_j) y los puntos aproximados (pa_j) mediante la curva cúbica con los parámetros t_j correspondientes es inferior al parámetro *MAX_DISTANCE*:

$$pr_j = (xr_j, yr_j), \quad pa_j = (a_x + b_x \cdot t_j + c_x \cdot t_j^2 + d_x \cdot t_j^3, a_y + b_y \cdot t_j + c_y \cdot t_j^2 + d_y \cdot t_j^3)$$

$$d_j = \| pr_j - pa_j \|$$

III.9 Extracción analítica de las características de dirección, curvatura y radio

III.9.1 Dirección

La dirección del trazo ($\alpha(t)$) es el ángulo de la tangente al trazo que se obtiene, mediante la derivada en cada punto del trazo aproximado por la curva cúbica paramétrica con el parámetro t_j correspondiente.

$$\alpha(t) = \arctan\left(\frac{y'(t)}{x'(t)}\right), \quad \left.\begin{array}{l} x'(t) = b_x + 2 \cdot c_x \cdot t + 3 \cdot d_x \cdot t^2 \\ y'(t) = b_y + 2 \cdot c_y \cdot t + 3 \cdot d_y \cdot t^2 \end{array}\right\}$$

Al igual que en el caso discreto, la función de los valores de ángulos está en el rango $[-\pi, \pi]$ y puede presentar saltos entre valores consecutivos debido al carácter cíclico de los ángulos, por lo que se realiza una corrección de cada valor en función del anterior:

Mientras $[(\alpha(t_j) - \alpha(t_{j-1})) < (-\pi)]$ *hacer* $(\alpha(t_j) \leftarrow \alpha(t_j) + 2 \cdot \pi)$

Mientras $[(\alpha(t_j) - \alpha(t_{j-1})) > (\pi)]$ *hacer* $(\alpha(t_j) \leftarrow \alpha(t_j) - 2 \cdot \pi)$

III.9.2 Curvatura

La curvatura del trazo ($c(t)$) es la variación de la dirección en cada punto del trazo, cuanto más varía la dirección en menos recorrido en el trazo, mayor es la curvatura. La curvatura es el cociente de la derivada de la tangente dividida entre la longitud del trazo. $\alpha'(t)$ es la derivada del ángulo de la tangente al trazo que se obtiene, mediante la derivada en cada punto del trazo aproximado por la curva cúbica paramétrica con el parámetro t_j correspondiente.

$$c(t) = \frac{\alpha'(t)}{(k-1)\cdot s} = \frac{\left[\arctan\left(\frac{y'(t)}{x'(t)}\right)\right]'}{(k-1)\cdot s} = \frac{\frac{\left[\frac{y'(t)}{x'(t)}\right]'}{1+\left[\frac{y'(t)}{x'(t)}\right]^2}}{(k-1)\cdot s} = \frac{\frac{\frac{y''(t)\cdot x'(t) - y'(t)\cdot x''(t)}{[x'(t)]^2}}{1+\left[\frac{y'(t)}{x'(t)}\right]^2}}{(k-1)\cdot s}$$

Donde *(k-1)·s* es la longitud de la curva cúbica paramétrica, ya que *s* es la distancia entre puntos re-muestreados (*INTERSPACING_DISTANCE*) y *k* el número de puntos de trazo que aproxima la curva cúbica paramétrica. Siendo las primeras y segundas derivadas de la curva paramétrica:

$$\left.\begin{aligned} x'(t) &= b_x + 2\cdot c_x \cdot t + 3\cdot d_x \cdot t^2 \\ y'(t) &= b_y + 2\cdot c_y \cdot t + 3\cdot d_y \cdot t^2 \end{aligned}\right\} \quad \left.\begin{aligned} x''(t) &= 2\cdot c_x + 6\cdot d_x \cdot t \\ y''(t) &= 2\cdot c_y + 6\cdot d_y \cdot t \end{aligned}\right\}$$

III.9.3 Radio

El radio del trazo (*r(t)*) es la inversa de la curvatura. Para evitar divisiones por cero, cuando el valor absoluto de la curvatura es inferior a un valor mínimo el radio se fija a un valor muy alto, con el mismo signo que la curvatura.

$$r(t) = \frac{1}{c(t)}$$

III.10 Detección de curvas y rectas

Las curvas se sitúan en secuencias (figura III.4) de puntos del trazo cuyos radios tienen el mismo signo y su valor absoluto está por debajo del parámetro *MAX_CURVE_RADIUS*. A efectos de cálculos, las curvas se consideran como arcos de circunferencia. De cada curva se calculan las siguientes características:

- Tramo: secuencia de puntos consecutivos del trazo cuyos radios (obtenidos analíticamente de las curvas cúbicas paramétricas) son inferiores a *MAX_CURVE_RADIUS*.
- Radio: la mediana de los radios de los puntos del tramo.
- Longitud: número de puntos del tramo multiplicado por *INTERSPACING_DISTANCE*.
- Ángulo: Longitud / Radio.
- Distancia de la cuerda al arco (figura III.3): Radio · [1 – cos (Ángulo / 2)]

Los tramos candidatos a curvas, lo serán si su distancia de la cuerda al arco es mayor que el parámetro *MIN_DIST_CA*.

Los tramos que no son curvas ni vértices esquina, son considerados rectas.

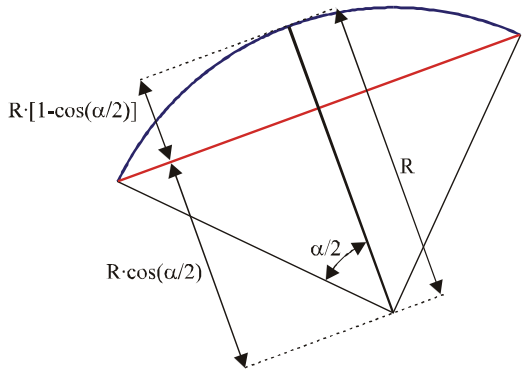


Figura III.3. Distancia de la cuerda al arco

III.11 Eliminación de rectas cortas y detección de vértices tangentes

Los tramos de rectas de longitud inferior a *MIN_LINE_LENGTH* son pasados a curvas si tienen a un lado una curva y no hay entre el tramo recta y el tramo curva un vértice esquina. En caso de que un tramo recta corto tenga curvas a los dos lados, el tramo recto se reparte a medias entre ambas curvas.

Los vértices tangentes se encuentran en:

- Cambios de tramos de recta a curva, y de curva a recta, en los que no hay un vértice esquina entre ambos tramos.
- Cambios de tramos de una curva a otra curva en los que no hay un vértice esquina entre ambos tramos (en este caso, el vértice tangente está situado en un punto de inflexión, es decir, un punto donde cambia el signo de la curvatura y del radio).

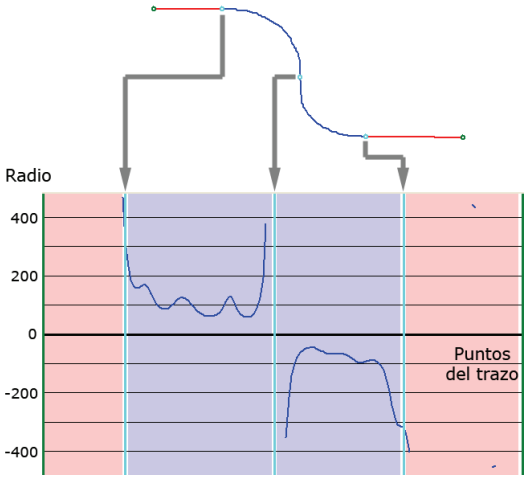


Figura III.4. Vértices tangentes (cian) entre curvas (azul) y entre líneas rectas (rojo), y entre dos curvas. El fondo de la función de radio es azul para curvas y rojo para rectas

III.12 Pasar el resultado de los puntos re-muestreados a los puntos originales

Para pasar el resultado de los *m* puntos re-muestreados a los *n* puntos originales, primero se busca el punto original más cercano a cada uno de los vértices esquina y vértices tangente de los puntos re-muestreados y se les asigna el mismo tipo, y posteriormente, se rellenan los tramos de puntos intermedios con el mismo tipo de tramo (recta o curva) que en los puntos re-muestreados.