UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Detección de eventos a bajo nivel dentro del sistema de supervisión automática SUPERVASION

TRABAJO FIN DE MÁSTER

Máster Universitario en Gestión de la Información

Autor: David Nieves Cordones

Tutor: Cèsar Ferri Ramírez
Carlos Monserrat Aranda
José Hernández Orallo

Curso 2016-2017

Resumen

En este trabajo se recoge el desarrollo de un prototipo para la detección de eventos de bajo nivel dentro del marco del proyecto SUPERVASION. Este proyecto propone un sistema innovador de supervisión automática por observación aplicado a la monitorización de destrezas en el aprendizaje de cirugía mínimamente invasiva. El prototipo debe informar al sistema sobre lo que sucede dentro del entorno de supervisión, procesando vídeos y devolviendo un listado con los eventos detectados. Se han usado y adaptado diversas técnicas de visión artificial y aprendizaje automático para el modelado de los eventos. Asimismo, se ha fabricado un simulador laparoscópico que reproduce las condiciones y funcionalidades de uno original donde se ha desarrollado este trabajo. La comparación experimental entre el prototipo y el etiquetado a mano de eventos demuestra que la detección es correcta, y con la velocidad suficiente para cumplir los requisitos de supervisión del sistema.

Palabras clave: Detección de eventos, supervisión automática, visión por ordenador, aprendizaje automático

Resum

En este treball s'arreplega el desenvolupament d'un prototip per a la detecció d'esdeveniments de baix nivell dins del marc del projecte SUPERVASION. Este projecte proposa un sistema innovador de supervisió automàtica per observació aplicat a la monitorització de destreses en l'aprenentatge de cirurgia mínimament invasiva. El prototip ha d'informar el sistema sobre el que succeïx dins de l'entorn de supervisió, processant vídeos i tornant un llistat amb els esdeveniments detectats. S'han utilitzat i adaptat diverses tècniques de visió artificial i aprenentatge automàtic per al modelatge dels esdeveniments. Així mateix, s'ha fabricat un simulador laparoscòpic on s'ha desenvolupat este treball, que reprodueix les condicions i funcionalitats d'un original. La comparació experimental entre el prototip i l'etiquetatge a mà d'esdeveniments demostra que la detecció és correcta, i amb la velocitat suficient per a complir els requisits de supervisió del sistema.

Paraules clau: Detecció d'esdeveniments, supervisió automàtica, visió per ordinador, aprenentatge automàtic

Abstract

This document summarises the development of a prototype for low-level event detection in the SUPERVASION project context. This project proposes an innovative system for automatic supervision by observation applied to skill acquisition monitoring in the minimally invasive surgery domain. The prototype will report about the events within the supervised environment, processing recorded videos and returning a list with events detected. We have used and adapted different computer vision and machine learning techniques for event modeling. Furthermore, we have made a laparoscopic simulator that recreates the requirements and features of a real one where we have developed this work. The experimental comparison between prototype output and hand-labelled ground truth shows that the detection is correct and fast enough for the supervision system requirements.

Key words: Event detection, automated monitoring, computer vision, machine learning

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Desde el origen de la inteligencia artificial y el aprendizaje automático, se ha intentado que las computadoras pudieran aprender y simular el comportamiento humano en las numerosas facetas que lo caracterizan, buscando, en muchas ocasiones, la mejor manera de asistirle en el día a día. En la actualidad, existen sistemas informáticos que pueden mantener conversaciones coherentes con personas y agentes inteligentes que reproducen su comportamiento en diversos contextos controlados. A pesar de ello, hay tareas para las que las personas son indispensables y que su automatización no se prevé en un futuro cercano. Un entorno físico complejo que requiera conocimientos abstractos de la situación, la necesidad de responder frente a sucesos inesperados o, simplemente, el coste de tiempo y esfuerzo en lograr su automatización, hacen indispensable el factor humano. Si bien es cierto que se ha logrado que una máquina aprenda y desarrolle una tarea en un dominio específico, invirtiendo años de investigación; todavía es inalcanzable la emulación de la capacidad humana para aprender tareas de forma habitual, partiendo únicamente de su comprensión del dominio, unas pocas instrucciones y la observación sobre cómo realizarlas.

Sin embargo, con los medios actuales, y dado un contexto controlado, es posible concebir que las computadoras pudieran reproducir este proceso. Así pues, se podría pensar en sistemas que pudieran aprender los pasos que definen una tarea determinada para supervisar el desempeño de la misma por parte de un humano, todo ello mediante el empleo de la observación. De esta manera, podría asistirle con información útil y comprensible para la persona cuando ésta realizara la tarea. Esto se conoce como *supervisión automática por observación*. Un sistema de este tipo tendría un impacto socioeconómico muy positivo en ciertas áreas de aplicación.

En puestos de trabajo críticos, donde se requiere el cumplimiento de tareas bajo un protocolo establecido y riguroso, el sistema sería capaz de detectar cuándo se ha roto dicho protocolo y avisar al trabajador de este hecho. Respecto a lo anterior, son varios los casos en los últimos años, en el que un fallo de este tipo ha acarreado costes muy elevados. Un ejemplo podría encontrarse en el incidente que tuvo lugar en 2014 en España, donde aparentemente, un fallo en el protocolo de retirada del traje protector ocasionó el contagio del virus ébola por parte de la enfermera que trataba a un paciente infectado [3] . Este tipo de fallos podrían verse reducidos mediante sistemas de supervisión automática que pudieran indicar cuándo, dónde y cómo se está fallando y que transmitieran el error de procedimiento en tiempo real y de una manera comprensible para el usuario o profesional. Otro caso podría ser el del accidente ferroviario de Angrois en 2013, el cual ha ocasionado que se busquen formas de supervisión dentro de las cabinas de los trenes, por medio de cámaras de vídeo y micrófonos, para monitorizar qué sucede en ellas [85]. Si bien es cierto que la solución que se propone en el caso anterior es muy similar al comportamiento de las "cajas negras" instaladas en las aeronaves (sistema de registro para su análisis posterior en caso de accidente), podrían explorarse soluciones que permitiera aprovechar dichos medios para detectar los errores de procedimiento antes de que originaran la catástrofe, y así poder evitarla en lo posible.

Otra posibilidad podría ser la aplicación de dichos sistemas en entornos de entrenamiento. Existen habilidades que los humanos podemos aprender de una manera mucho más eficiente cuando tenemos un experto supervisándonos mientras la practicamos. Esto está sujeto a limitaciones obvias, ya que el entrenamiento bajo este tipo de supervisión está supeditado a la disponibilidad del experto, no siendo accesible en muchas ocasiones. Si bien es cierto que hoy en día existen simuladores que reproducen muchos entornos de entrenamiento, ofreciendo un programa que evalúa el desempeño de tareas; por lo general, estos suelen ser muy costosos, limitando así su uso por parte de los aprendices a las horas de prácticas. De esta forma, un supervisor automático que pudiera monitorizar, detectar y asistir con información útil y en tiempo real, los errores de concepto o de procedimiento, en fases iniciales del aprendizaje de nuevas rutinas, ayudaría a reducir en gran medida la curva de aprendizaje; suponiendo un ahorro considerable de tiempo y dinero.

En el dominio de la cirugía mínimamente invasiva (Minimally Invasive Surgery o MIS), lo anterior cobra especial importancia. La adquisición y maestría de las habilidades manuales necesarias para poder ejercer este tipo de cirugía por parte de los aprendices, requiere un número de horas de entrenamiento muy elevado hasta llegar a dominarlas. Además, en un campo tan crítico como es el de la cirugía, un desarrollo pobre de estas competencias podría acarrear graves consecuencias para los pacientes, como puede ser un mayor número de complicaciones durante el postoperatorio o incluso la muerte [8]. A su vez, los errores asociados a la técnica, durante las intervenciones quirúrgicas, son el principal motivo de reoperaciones y readmisiones en hospitales [69]. Por tanto, los avances que se pudieran ofrecer en este sentido, que tuvieran como objetivo mejorar la forma en que los aprendices adquieren sus habilidades técnicas, repercutiría en una asistencia más efectiva y segura a los pacientes.

Por contra, existen limitaciones considerables con respecto a los medios disponibles para poder desarrollar estas competencias. Concretamente, en el campo de la cirugía laparoscópica, estas limitaciones se hallan en el acceso a entornos de tests realistas y la disponibilidad de expertos que puedan supervisar los entrenamientos. En la actualidad, existen entornos virtuales de cirugía que permiten cubrir significativamente la carencia de un entorno real de entrenamiento. Gracias a ellos, los estudiantes pueden ejercitar sus habilidades de una forma sumamente realista. Aunque dichos simuladores han intentado paliar la necesidad de la supervisión del experto, por medio de la evaluación automática de los ejercicios, su presencia, la del experto, sigue siendo más que necesaria para poder considerar el aprendizaje efectivo [58][40].

En este marco nace el ambicioso proyecto *SUPERVASION* (TIN2014-61716-EXP), cuyo objetivo es el desarrollo y aplicación de sistemas de supervisión automática por observación, trabajando actualmente sobre el dominio del entrenamiento de habilidades laparoscópicas. Esto significa la creación de un sistema que, en primera instancia, aprenda el protocolo de un ejercicio partiendo de una descripción en lenguaje natural y unas pocas observaciones de su realización. Tras esta fase, debe de ser capaz de monitorizar y asistir a aprendices cuando estos se encuentran realizándolo, empleando también la observación para ello. Como puede apreciarse, este proyecto requiere una solución multidisciplinar, sustentándose, en su faceta más tecnológica, sobre las áreas del aprendizaje automático y la visión por ordenador.

De este proyecto se desprenden numerosos retos. Cuestiones acerca de cómo el sistema conoce lo que está sucediendo en el entorno de entrenamiento o cómo es capaz de asociarlo a tareas más abstractas, y que éstas sean significativas para los seres humanos, son solo algunos de ellos. En este sentido, SUPERVASION propone afrontarlo desde el punto de vista de la programación inductiva, o *IP* [34], la cual aúna el aprendizaje automático con la adaptabilidad y potencia de la programación declarativa, junto con un proceso de *Inducción por Abducción* [60].

Uno de los pilares básicos para que este proceso se pueda llevar a cabo, reside en la detección de eventos observables. Estos eventos están asociados a acciones del mundo real (el entorno de prácticas de laparoscópia, en este caso) y que son significativas para la monitorización del ejercicio. Por tanto, serán las porciones de información más finas, o de bajo nivel, que requiere el sistema,

indicándole qué está haciendo el usuario en todo momento. Acciones como "*el instrumento A se ha abierto...*" o "*El instrumento B ha entrado en la zona del objetivo...*'; serán cruciales para el correcto seguimiento del ejercicio. En trabajos previos [60], esta detección se había realizado a mano, para así reproducir el proceso de *Inducción por Abducción*, aunque se introducían las bases para su automatización. Concretamente, esta responsabilidad se planificaba que recayera sobre un componente de software denominado *Event Detector*, encargado de ser los *ojos* del sistema de supervisión automática. Todo estos detalles se verán con mayor profundidad dentro de este trabajo (Capítulo 2.2).

Por tanto, en este trabajo se recogerá ese testigo. En el presente documento se mostrará la aproximación desarrollada, en forma de prototipo, para el componente *Event Detector*, encargado de la detección de estos eventos de bajo nivel dentro del proyecto SUPERVASION. Partiendo de vídeos con la realización de ejercicios de laparoscopia, este componente será capaz de identificar los eventos que han acontecido, los elementos implicados y el instante en el que se detectó. Para poder realizar esta tarea, se requerirá la adaptación y despliegue de una serie de técnicas de visión por ordenador [80] y de aprendizaje automático [28] que permitan el modelado de los eventos, permitiendo la automatización completa de este proceso de detección.

1.1 Objetivos

1.1.1. Objetivo Principal

Desarrollo de un prototipo para la detección de eventos a bajo nivel dentro del proyecto de supervisión automática por observación SUPERVASION. El prototipo debe ser capaz de detectar los eventos de bajo nivel que tengan lugar dentro del entorno supervisado por medio del procesado de vídeo. La salida obtenida tiene que reflejar los eventos sucedidos junto con su información temporal.

1.1.2. Objetivos Secundarios

- Adaptación de técnicas de visión por ordenador y aprendizaje automático que permitan el modelado y detección de eventos de bajo nivel asociados al instrumental quirúrgico.
- Creación de un entorno laparoscópico para la realización de ejercicios de entrenamiento de laparoscopia. Este entorno debe de reunir las condiciones y características de un simulador original que favorezca la realización de ejercicios. Además debe de adaptarse y permitir el correcto despliegue de las técnicas requeridas para la detección de eventos observables.
- Facilitar la integración del componente desarrollado con los futuros módulos de SUPERVASION.
- Puesta en marcha del prototipo sobre una simulación de ejercicio del dominio, modelando los eventos a bajo nivel que intervienen en su desarrollo.
- Evaluación de los resultados obtenidos, así como su difusión y publicación del código desarrollado en un repositorio público.

1.2 Estructura de la memoria

El presente trabajo tiene la siguiente estructura:

- En el capítulo 2 se detallan los diferentes tópicos que rodean el desarrollo de este trabajo, entre ellos el estado del arte, el contexto y las técnicas empleadas.
- En el capítulo 3 se describen los recursos y métodos usados para la adaptación de técnicas y creación del prototipo del Event Detector.
- En el capítulo 4 se muestran los resultados obtenidos tras el despliegue del prototipo creado sobre un ejercicio del dominio junto con la discusión de los mismos.
- En el capítulo 5 se exponen las conclusiones obtenidas de esta memoria y el trabajo futuro.

CAPÍTULO 2
Conceptos previos

En este capítulo se recopilan los distintos temas relacionados con el desarrollo que se recoge en este trabajo. Se repasará el estado del arte en torno a la supervisión automática actual. Posteriormente, se introducirá el proyecto SUPERVASION y su sistema de supervisión automática por observación, para dar paso después al dominio donde se aplicará dicho sistema. Por último, se detallarán las técnicas de visión por ordenador que se han implementado para la detección de eventos a bajo nivel.

2.1 Antecedentes de la supervisión automática

La relevancia de los sistemas de supervisión automáticos, en diversos sectores industriales, ha experimentado un crecimiento considerable en los últimos años. Estos suelen estar orientados a la vigilancia y detección de incidencias. Se pueden ver propuestas de monitorización para el sector del transporte, como en aviones y aeropuertos [90] [92], trenes y transporte metropolitano [19] y control de vehículos [45]. A su vez, la proliferación de sistemas de vigilancia basados en el reconocimiento de actividades humanas también ha sido destacable, avanzando en tópicos como detección de comportamientos anómalos [19], reconocimiento de actividades en sistemas multi-cámara [27] o sistemas tolerantes a la oclusión y al multi-tracking de personas como W/sup 4 [37]. En este último campo, el del reconocimiento de actividades humanas, han surgido gran cantidad de datasets enfocados al entrenamiento de sistemas de este tipo, siendo [18] una recopilación de muchos de ellos, catalogados por actividades y dominios de aplicación.

La evolución de estos sistemas ha ido a la par de la evolución de la capacidad de computación de los ordenadores, junto con el surgimiento de nuevas técnicas de aprendizaje automático y visión por ordenador, que han permitido automatizar varios de los procesos. El comienzo de la supervisión o *cybersupervisión* distaba mucho de lo que es hoy. Durante los años 70 y comienzo de los 80, la supervisión de tareas consistía en sistemas analógicos sustentados en canales cerrados de televisión, donde el papel que juega la computadora, en los mejores casos, quedaba relegado a ser mero monitor de información, no siendo una pieza activa en el proceso de supervisión. Esta labor recaía sobre un operario, el cual cumplimentaba formularios según las tareas que se iban completando, usando un equipo informático para monitorizar la escena.

Con la llegada de la tecnología digital para multimedia y la consolidación de Internet, los nuevos sistemas de supervisión pudieron ser más sofisticados. Durante los años 80 surgieron propuestas que unían soluciones del campo de la visión por ordenador y aprendizaje automático a sistema de vigilancia [30]. En ellas se buscaba asistir al operario de vigilancia, con la detección automática de ciertos comportamientos en la escena que se estaba registrando. Es en este momento donde surgen ciertos tópicos de interés de este campo, como pueden ser la creación de algoritmos cada vez más eficientes para la detección en vivo de eventos, donde el tiempo de respuesta es crucial; trabajar con entornos no controlados y con incertidumbre de sucesos en la escena; o el entrenamiento de

sistemas para que sean capaces de interpretar la escena, detectando los objetos que aparecen y cómo se relacionan entre ellos. En definitiva, la interpretación automática va tomando cada vez más peso en los sistemas de supervisión, reduciendo la labor activa del operario.

En la actualidad, debido al abaratamiento, miniaturización y conectividad de los sensores, se ha ido derivando a entornos híbridos, donde la fusión de datos o *Data fusión* juega su principal baza. Los sistemas de supervisión automáticos ya no solo trabajan con los recursos que captan las cámaras, sino que permite la combinación y ampliación de esta información con otros datos recogidos por multitud de sensores que pueden estar distribuidos por la área supervisada. Los beneficios de esta fusión de datos pueden ser varios [7]. Por un lado, se pueden tener distintos niveles de granulidad de la información, desde su perfil más bajo, como son los datos de los sensores en crudo; hasta las capas más abstractas, con las agregaciones de estos datos en eventos de alto nivel. Además, el uso de sensores permite completar la información del entorno, de igual modo, que puede mejorar la precisión de aquellos datos derivados de otros recursos. De esta fusión surgen nuevos retos, como puedan ser la definición de protocolos de interoperabilidad, el diseño eficiente de estos entornos o cómo explotar esta información para mejorar los propios sistemas de supervisión.

Cuando nos adentramos en el dominio del entrenamiento de destrezas de cirugía mínimamente invasiva, lo sistemas de supervisión actuales se centran principalmente en la evaluación final del usuario. Entre las propuestas de métodos para la obtención de estas evaluaciones automáticas podemos encontrar el antecendente del proyecto SUPERVASION [61], en el cual se proponía un sistema para valorar el desempeño de cirujanos noveles mediante el cálculo de distancias de edición. Inicialmente, se registra la ejecución del cirujano experto, generando un patrón de caracteres con las acciones efectuadas. Este mismo proceso se realiza con la ejecución del aprendiz, calculándose después la distancia de edición con el algoritmo LCS [36] entre el patrón del aprendiz y el del experto. Por otro lado, la evaluación automática de destrezas quirúrgicas es un campo bastante prolífico [72] [21]. De forma más concreta, y teniendo en cuenta el método que emplean para estimar la evaluación del entrenamiento de los aprendices, algunas de estas propuestas se podrán agrupar en: uso de pesos, Análisis de Discriminates Lineales (LDA), Modelos Oculos de Markov (HMM) y clasificadores difusos (fuzzy classifiers).

Por ejemplo, el sistema CELTS (Clinical-Based Computer Enhanced Laparoscopic Analysis) [78], es un ejemplo representativo de la evaluación basada en pesos. Esta aproximación toma en consideración factores como la profundidad a la que se introduce el instrumental, orientación, camino seguido o el tiempo en concluir el ejercicio como indicadores para la evaluación.

Una combinación de pesos y el uso de métodos basados en LDA, es lo que se propone con WKS-2RII [77]. Este trabajo se centra en la evaluación de habilidades de sutura, donde se tienen en cuenta datos como la presión ejercida sobre la piel o distancia entre los puntos de sutura y el límite de la herida para estimar el nivel del cirujano. Estos factores se combinan linealmente, aplicando después LDA para discriminar así el nivel. Los grupos posibles en los que clasifica son *experto*, *intermedio* y *aprendiz*.

Otras aproximaciones, aprovechan el potencial de los modelos de Markov en este dominio [50] [47]. Un ejemplo interesante se puede encontrar en [71]. Partiendo de una analogía entre la secuencia de gestos (que ellos denominan estados o *states*) requeridos en un entrenamiento y la secuencia de palabras en el lenguaje textual, lo traslada al dominio de los modelos ocultos de Markov. Con esta premisa, construyen dos modelos HMM: uno representando la secuencia de estados del cirujano expertos y el otro la transición de estados del cirujano aprendiz. Ambas están basadas en un proceso entrenamiento previo con cirujanos clasificados en cada grupo. Cuando están entrenados ambos modelos, un nuevo cirujano es evaluado calculando su secuencia de estados y midiendo la distancia de dicha secuencia con respecto a los modelos aprendidos, asignándole el grupo más cercano al despeño realizado.

Acerca del empleo de clasificadores difusos, los sistemas propuestos en [36] y [41], los usan para crear tres tipos de *clusters*, en función del nivel de habilidad de los cirujanos: aprendiz,

intermedio o experto. Los tres clusters se aprenderán de datos previos, procedentes de cirujanos ya evaluados con diferentes niveles de experiencia. Una vez creados los clusters, el clasificador difuso devolverá el nivel de habilidad del nuevo candidato cuando se evalúe su ejercicio.

Varios de los trabajos mencionados emplean datos extraídos de simuladores de cirugía profesionales para el entrenamiento de los modelos. Algunos ejemplos como Simbionix [1][6], MIST–VR [91] o da Vinci Surgical System (dVSS) [2] [35], incorporan interfaces para extraer métricas y datos de los ejercicios realizados. Con este propósito encontramos el dataset *JIGSAW* [31]. Amparado por la universidad Johns Hopkins (JHU - *Johns Hopkins University*) de Baltimore, consiste en un conjunto de datos sobre actividades quirúrgicas y de disposición pública (bajo petición por email). Los datos fueron extraídos empleado el sistema dVSS, registrándose información cinemática de los distintos instrumentos empleados y el vídeo del ejercicio en cuestión. Este conjunto de datos se centra en tres tipos de ejercicios: sutura, anudado y paso de aguja. Para cada uno, tiene diversas muestras o ejemplos de ejecución realizadas por cirujanos con distintas horas de experiencia. Incorpora también la anotación manual, a modo de *ground truth*, de la secuencia de gestos que tienen lugar en el vídeo, para así poder facilitar el entrenamiento de sistemas automáticos.

Fuera de la cirugía, pero todavía dentro del ámbito sanitario, también han habido interesantes incursiones en sistemas de supervisión automática y detección de eventos. Se puede mencionar el robot propuesto en [64]. Los autores presentan un robot con capacidad de monitorizar el hogar, detectando, por ejemplo, la caída de una persona. Cuando esto sucede, el asistente se aproxima ofreciéndole ayuda, llegando incluso a pedir auxilio de forma remota para los casos más graves. Esto puede tener especial interés para personas mayores que residen solas. En este contexto de asistencia en el hogar destaca el sistema COACH [59]. Este sistema supervisa y asiste a personas con un grado moderado a severo de demencia durante el proceso de lavarse las manos. Por medio de una cámara, el sistema supervisa la escena, el lavabo en este caso, empleando técnicas de visión por ordenador para realizar el seguimiento de las manos y los objetos relacionados con la tarea. Esto se combina con un modelo POMDP (Partially Observable Markov Decision Process) encargado de la toma de decisiones del sistema, determinando en qué paso del proceso se encuentra la persona, y estimando también el grado de ayuda que debe de ofrecerle, dependiendo del comportamiento que haya percibido. Para la asistencia, COACH posee un monitor que irá mostrando los distintos mensajes de ayuda, tanto de vídeo como de voz, en relación al grado de asistencia que ha estimado y el paso en el que se halla la persona.

2.2 Proyecto SUPERVASION

Proyecto financiado por el Ministerio de Economía, Industria y Competitividad de España (TIN2014-61716-EXP) durante la convocatoria de proyectos “Explora Ciencia” y “Explora Tecnología” de 2014, tiene como principal motivación la creación de sistemas de supervisión automática mediante observación, con el objetivo de alcanzar tecnologías aplicables a la adquisición de destrezas de forma autónoma y la asistencia en procedimientos humanos. Seguidamente, se describirá el marco general que cubre este proyecto junto con las fases y procesos que componen el sistema de supervisión por observación que se propone.

2.2.1. Marco general

Este proyecto nace con la idea explorar las tecnologías que puedan hacer posible la creación de sistemas que asistan a humanos durante el proceso de aprendizaje o desempeño de tareas, cuando estas requieran ciertas habilidades. De las distintas maneras que se podría abordar este propósito, dicho proyecto busca su consecución por medio de técnicas basadas en aprendizaje automático y

[1] http://simbionix.com/simulators/lap-mentor/library-of-modules/basic-skills/
[2] https://intuitivesurgical.com/products/skills_simulator/

visión por ordenador, enmarcándose de esta manera en el campo de supervisión automática por observación. La motivación principal es la creación de asistentes automáticos que, partiendo de descripciones en lenguaje natural de la tarea, y la observación de ejemplos significativos de su consecución por parte de expertos, pueda supervisar posteriormente la realización de la misma por un aprendiz, empleando también la observación. Durante el proceso de supervisión, el sistema debe de ser capaz de detectar las desviaciones del aprendiz con respecto al protocolo aprendido, indicándoselo con información útil y comprensible sobre la correcta ejecución de ese paso.

Con el objetivo de poder prototipar rápidamente las ideas anteriormente expuestas, el proyecto SUPERVASION se ha enfocado al dominio de los ejercicios de cirugía mínimamente invasiva, concretamente, el entrenamiento de habilidades laparoscópicas. Este campo ofrece numerosas ventajas, como un contexto de aplicación sumamente controlado y los elementos sencillamente modelables partiendo de los recursos de que se dispone. Por tanto, se pretende el desarrollo de un sistema de supervisión automática que, partiendo de la descripción textual de un ejercicio y de unas pocas ejecuciones realizadas por un cirujano experto, pueda asistir a los aprendices de cirugía durante su entrenamiento.

Se pretende que el sistema sea capaz de aprender a través de estos tres recursos o entradas:

- **Narrativa:** Consiste en la descripción a alto nivel de la tarea a realizar. Es proporcionada por el cirujano experto, donde se describe paso a paso la secuencia de acciones necesarias, en lenguaje natural y fácilmente comprensible. Esta descripción será traducida posteriormente a unas propiedades abstractas que el sistema pueda interpretar, pudiendo así seguir el flujo narrativo del ejercicio durante su desempeño.

- **Background knowledge:** Consiste en un conjunto de reglas que permiten dotar al sistema de conocimiento acerca del dominio en el que se está trabajando. Escritas en lenguaje lógico, estarán formadas tanto por reglas generales, independientes del dominio, como específicas al ejercicio a aprender.

- **Eventos a observables:** Son los eventos a bajo nivel y directamente observables por el sistema. Estos eventos serán capturados de los vídeos con la realización de los ejercicios por parte del cirujano experto. Por tanto, el sistema tendrá que ser capaz de detectarlos en tiempo de reproducción del vídeo y, posteriormente, poder codificar dicha información en un formato apto para emplearse en las capas posteriores de abstracción.

En la Figura 2.1 se puede ver el esquema general de los recursos necesarios para la fase de aprendizaje, y posterior supervisión, para un ejercicio de precisión de corte. Inicialmente, se comenzará con la fase de adquisición de conocimiento o aprendizaje por parte de la máquina. Se requerirá la descripción a alto nivel del ejercicio, estando proporcionada por el cirujano experto (parte superior de la figura). En ella se pormenorizarán los pasos a seguir, expresados en lenguaje natural, del ejercicio en cuestión. Esta narrativa sería ampliable con información relevante que le pueda ser de interés al aprendiz para su mejor desarrollo. A su vez, se modelarán las restricciones generales del dominio, junto con las específicas del ejercicio, ambas en lenguaje formal (*background knowledge*); teniendo siempre en cuenta las especificaciones y criterio del experto. Con estos recursos se le especificaría al supervisor automático qué objetos intervienen, las propiedades y relaciones que existen entre ellos, restricciones sobre su comportamiento, etc; todo lo anterior en un formato lógico de reglas que la máquina pueda interpretar. Finalmente, se emplearán diversas grabaciones de vídeo con ejemplos "ricos" y representativos de la correcta ejecución del ejercicio por parte del experto (parte izquierda de la figura). Esto último permitirá al sistema supervisor identificar qué eventos observables lo forman, así como la secuencia de ejecución de los mismos. Hasta este punto sería el proceso de entrenamiento, estando el sistema capacitado para la supervisión automática una vez finalizado.

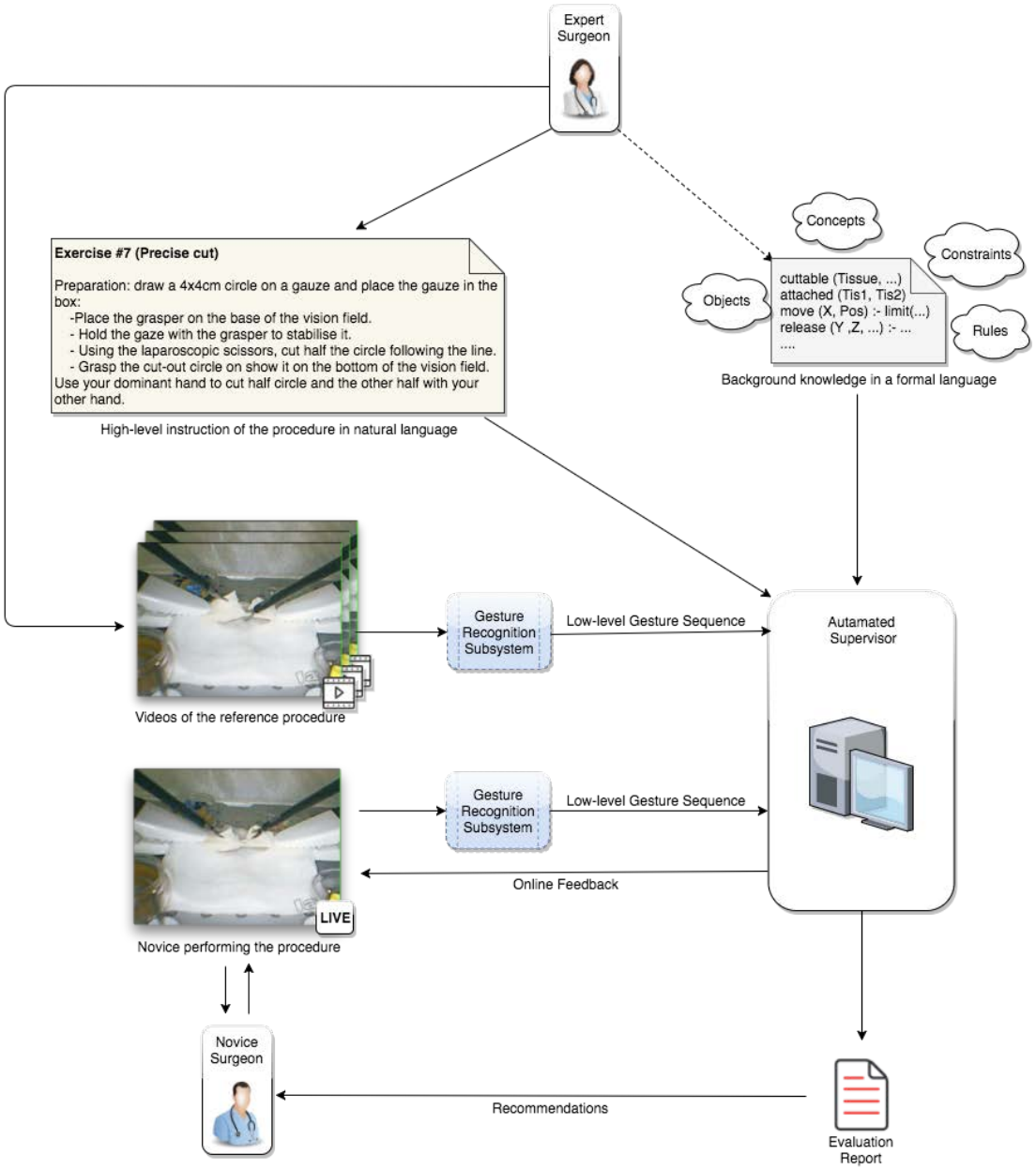


Figura 2.1: Esquema general del proceso de supervisión automática.(Adaptado de [12])

Teniendo en cuenta la heterogeneidad de los recursos mencionados, se requerirá un paradigma que permita la unión de los mismos bajo un mismo marco, haciendo posible que un sistema informático pueda asimilarlos y aprenderlos. Para este proyecto se ha seguido una aproximación basada en el paradigma de la programación inductiva [34], más concretamente, mediante el uso del lenguaje lógico Event Calculus (EC) [43]. Este lenguaje proporcionará las funcionalidades necesarias para poder realizar el proceso de abstracción, desde los eventos de bajo nivel (eventos observables), hasta las descripciones en lenguaje natural de alto nivel (narrativa).

A continuación, describiremos de forma más detallada cada una de las fases de la supervisión automática por observación propuesta anteriormente, así como el proceso de abstracción mencionado, Pudiéndose encontrar una descripción mucho más profunda en [12].

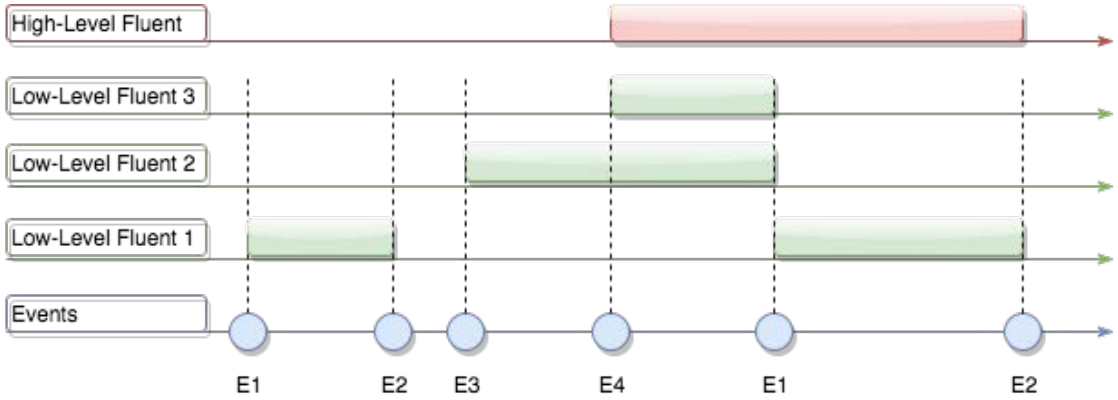


Figura 2.2: Elementos de EC definidos: Eventos (Ei), Fluents a bajo nivel (en verde) y Fluents a alto nivel (en rojo). (Adaptado de [12])

2.2.2. Fase de adquisición de conocimiento

En esta fase, el sistema debe de aprender las reglas que modelan el dominio sobre el que supervisará. Para lograr esto, se requerirá la codificación y agregación de las distintas entradas previamente comentadas, es decir: narrativa, background knowledge y eventos observables; en reglas lógicas que el sistema pueda interpretar y seguir. Es por ello que se necesite de un lenguaje lógico, EC en nuestro caso, para poder llevar a cabo este objetivo.

EC es un lenguaje destinado a la formalización de acciones y los cambios que producen las mismas. Este lenguaje consta de acciones, denominadas *eventos*, y de propiedades dependientes de una variable temporal, llamadas *fluents* [87]. Estos fluents serán ciertos durante un periodo concreto de tiempo, teniendo un inicio y un final. En relación a este proyecto, se han identificado 3 elementos de EC, siguiendo la aproximación propuesta en [4], siendo lo siguientes: *events*, *simple fluents* (*low-level fluent*) y *statically determined fluents* (*high-level fluent*).

En la Figura 2.2 se puede ver la relación que existe entre estos elementos. Los fluents de bajo nivel (*Low-level fluents*) corresponden a propiedades observables que son ciertas durante un periodo de tiempo, estando iniciados y finalizados por los eventos observables (*Events*). Estos eventos son acciones observables y puntuales en el tiempo, es decir, que suceden en un instante determinado, y serán los detonantes de que un fluent de bajo nivel comience o finalice. Por otra parte, los fluents a alto nivel (*High-level fluents*) serían extraídos directamente de las descripciones a alto nivel proporcionadas por el cirujano experto, siendo estos no observables directamente. Estos fluents se obtendrían de las narrativas de los ejercicios y se traducirían posteriormente a EC.

En la Figura 2.3 se puede ver un ejemplo simplificado de la relación que existe entre eventos y fluents, una vez aplicados al dominio. El evento *opens(forceps)* inicia el fluent *open(forceps)*, indicando que el instrumento fórceps se acaba de abrir y éste está abierto a partir de ese momento. El anterior fluent finaliza cuando sucede el evento *picks(forceps, bead)*. Esto significa que el fórceps acaba de pinzar el objeto del ejercicio (una cuenta de hama o bead), finalizando así el fluent anterior, y dando inicio al siguiente, en este caso, *carry(forceps, bead)*. La relación de inicio/fin de los fluents de bajo nivel anteriores, estaría asociado al fluent de alto nivel *clip_object(forceps, bead)*, el cual podría ser una indicación de la narrativa del tipo "*Empleando el fórceps, pinza la cuenta...*". Dicho fluent de alto nivel será cierto hasta que el fluent de bajo nivel *carry(forceps, bead)* finalice y comience el siguiente fluent. Este caso se da cuando se detecta el evento relativo a que el objeto ha sido soltado (*drops(forceps, bead)*), por lo que el fluent de bajo nivel *carry(forceps, bead)* finaliza, concluyendo a su vez el fluent de alto nivel. Lo anterior hace que se inicie un nuevo fluent de bajo nivel relacionado con el estado del fórceps, en este caso, que está abierto (*opens(forceps)*).

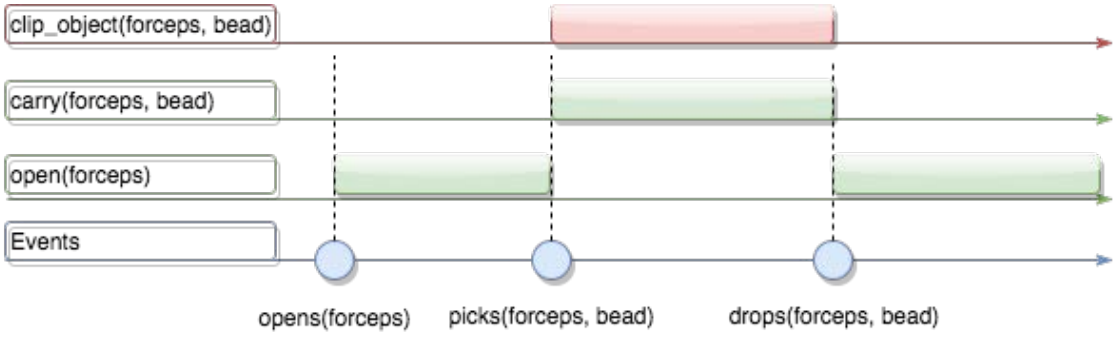


Figura 2.3: Ejemplo de la relación entre eventos y fluents dentro del dominio. (Imagen de elaboración propia)

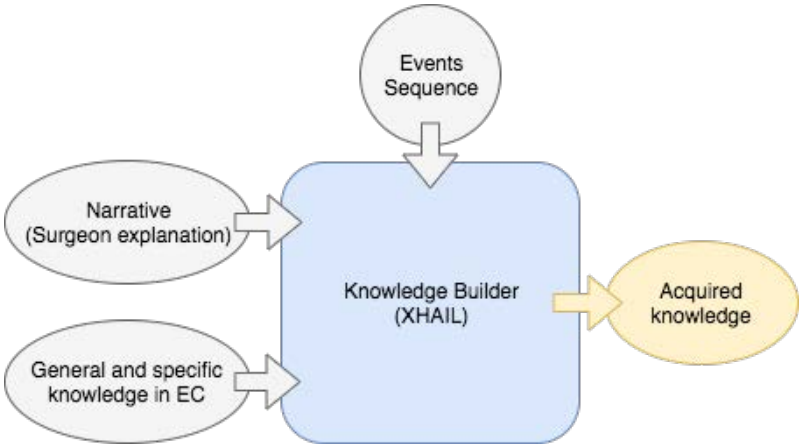


Figura 2.4: Esquema de la adquisición de conocimiento por el sistema supervisor (adaptado de [12]) .

Como se puede apreciar en el ejemplo anterior existen una serie de elementos (eventos y fluents) que están relacionados entre sí. Será necesario transportar estos elementos y sus relaciones a reglas lógicas que una computadora pueda seguir y tomar como conocimiento. La herramienta escogida para la adquisición de conocimiento será XHAIL (*eXtended Hybrid Abductive Inductive Learning*) [3] [68]. Esta herramienta se empleará durante la fase de aprendizaje, haciendo posible que el sistema supervisor aprenda nuevo conocimiento de los recursos de entrada. Para lograr esto, se aplica un proceso de aprendizaje denominado *inducción por abducción*. Éste consiste en partir de unas descripciones de hechos y poder llegar a las condiciones que permiten inducirlos. En nuestro caso, XHAIL permite crear las reglas que conecten los fluents de alto nivel con los eventos de bajo nivel. Esto significa descender por las capas de abstracción desde las propiedades abstractas, como las extraídas de las narrativas, y asociarlas a propiedades observables, cumpliendo así las reglas definidas en el *background knowledge*.

El esquema general de la fase de adquisición de conocimiento se puede ver en la Figura 2.4. En la parte izquierda se localizan las entradas relativas a las narrativas y las reglas del *background knowledge* ya descritas. En la parte superior se halla la secuencia de eventos observables de bajo nivel. Éstos serán las porciones de información de grano más fino relacionados con la ejecución de los ejercicios. Su detección y gestión serán responsabilidad de un componente de software denominado Event Builder o EB. El foco del presente trabajo se centrará en él, por lo que será descrito con mayor detalle en el apartado 2.2.4.

[3] https://github.com/stefano-bragaglia/XHAIL

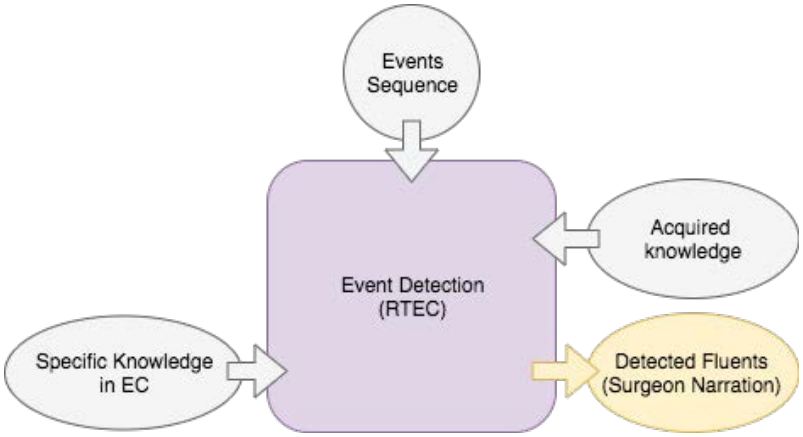


Figura 2.5: Esquema de la supervisión en tiempo real por el sistema supervisor (adaptado de [12]).

Por tanto, de un ejercicio de ejemplo, realizado por el cirujano experto, se obtendrá la secuencia de eventos de bajo nivel que componen dicho ejercicio, uniéndose a la narrativa y al *background knowledge* previo. Estos se introducirán como entradas a XHAIL, nuestro constructor de nuevo conocimiento en este caso. Tras su ejecución, la salida que se genera consiste en un conjunto de reglas lógicas que representan el nuevo conocimiento adquirido acerca del ejercicio a supervisar.

Tras la conclusión de esta fase, es decir, la de adquisición de conocimiento por parte del sistema, se dará paso a la fase de supervisión automática y asistencia.

2.2.3. Fase de supervisión online

En esta fase el sistema supervisor procederá a la monitorización en tiempo real del aprendiz durante su realización del ejercicio de entrenamiento, asistiéndole en el proceso. Por tanto, y a diferencia de la fase de adquisición de conocimiento, el sistema deberá trabajar, en esta ocasión, con una señal de vídeo en vivo con el desempeño del aprendiz. Su rol también habrá cambiado, ya que ahora tendrá que informar, aconsejar y corregir a lo largo del entrenamiento; haciendo así que su asistencia mejore el aprendizaje y éste sea efectivo.

El esquema general para esta fase se recoge en la Figura 2.5, mostrándose las entradas requeridas y la salida que se genera. Las entradas consistirán en: la secuencia de eventos en tiempo real, el conocimiento específico en EC y el conocimiento adquirido tras la fase anterior. Llegados a este punto, se hace necesario una herramienta que permita gestionar el conocimiento adquirido durante la fase de aprendizaje, a la vez que haga posible su aplicación dados nuevos casos. Teniendo en cuenta lo anterior, se ha escogido la herramienta RTEC [4] [4] (*Run-Time Event Calculus*), la cual permite la detección de fluents de alto nivel a través de una secuencia de eventos y fluents de bajo nivel. Así pues, la salida que obtendremos de esta fase consistiría en la detección de cuándo un fluent de alto nivel se inicia y finaliza, aplicando para ello el conocimiento de los ejemplos aprendidos. Como resultado se obtendrá la parte de la narrativa que estaba asociada a dicho fluent de alto nivel, pudiendo en todo momento identificar en qué parte del ejercicio se encuentra la ejecución.

Una vez descritas las dos fases en las que se compone el proceso de supervisión automática por observación, procederemos a detallar el componente de software Event Builder.

[4] https://github.com/stefano-bragaglia/XHAIL

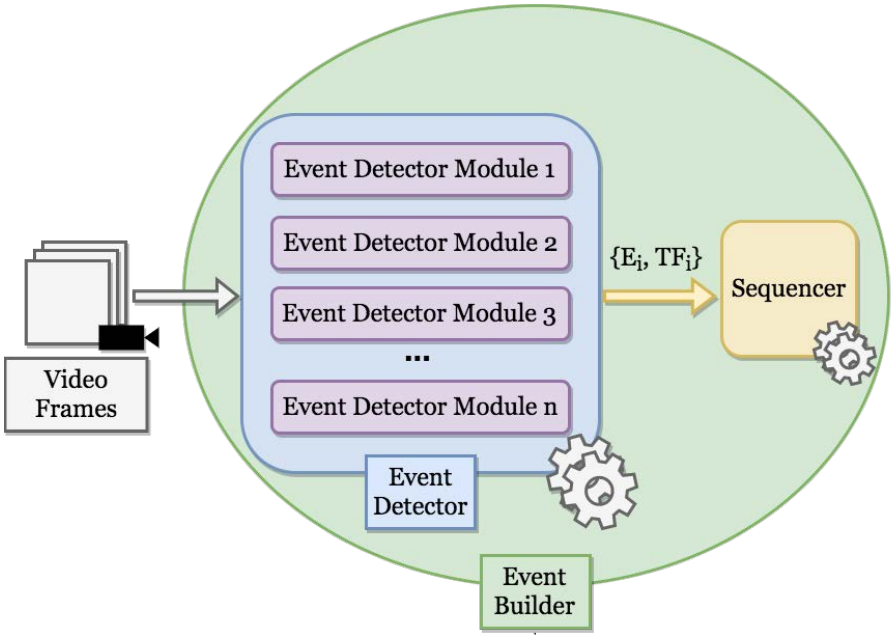


Figura 2.6: Esquema del componente Event Builder durante la fase de adquisición de conocimiento. (Imagen de elaboración propia).

2.2.4. Event Builder (EB)

Este componente de software tendrá la misión de procesar, detectar y ordenar la información de lo que sucede en el entorno supervisado, extrayendo los eventos observables de más bajo nivel relacionados con la ejecución de los ejercicios. Como salida se obtendrá una secuencia de eventos en lenguaje EC que será la entrada del sistema XHAIL o RTEC, dependiendo de la fase en la que nos encontremos. Al estar en ambas fases, será necesario que pueda trabajar tanto con grabaciones de vídeo como con una señal de vídeo en tiempo real, debiendo de ser configurable para ese propósito.

Tal y como se puede ver en la Figura 2.6, el componente EB estará formado por dos componentes más específicos para llevar a cabo su función, estos serán: el Detector de Eventos (*Event Detector* o *ED*) y el Secuenciador de Eventos (*Sequencer*). El primero de ellos consistirá en un set de módulos, cada uno especializados en la detección de uno o varios eventos de bajo nivel. Cada módulo empleará las técnicas pertinentes para poder asegurar la correcta detección de los eventos para los que está especializado. Estos eventos estarán asociados a marcas distintivas adheridas al instrumental y objetos del ejercicio, permitiendo identificarlos y poder trazar sus movimientos y estados. Para ello se emplearán técnicas de visión por ordenador y de aprendizaje automático.

El flujo de trabajo del ED consistirá en recibir como entrada los frames de vídeo, aplicar sobre ellos los módulos de detección de eventos, obteniendo así los eventos detectados (Ei), junto con el instante de frame cuando se detectó (TFi). Posteriormente, el componente Sequencer se encargará de ponerlos en orden temporal para poder traducirlos a secuencias de eventos de Event Calculus. Los distintos procesos para la detección de los eventos formará parte del núcleo central de este trabajo, siendo descrito en mayor detalle en el apartado 3.3 de este trabajo.

Una vez descrito el sistema donde se enmarca este trabajo, se procederá a introducir el dominio donde se usará la supervisión automática.

2.3 Dominio de la Cirugía Mínimamente Invasiva

Entre los diversos ámbitos en los que se podría aplicar el sistema propuesto por el proyecto SUPERVASION, se demostrará sus posibilidades dentro del dominio de la supervisión de destrezas en el aprendizaje de cirugía mínimamente invasiva. Este dominio será propicio por los siguientes motivos:

- El contexto donde se realizan los ejercicios está sumamente controlado. A priori, todos los ejercicios especifican los elementos que intervendrán para su ejecución, cerrándose el entorno laparoscópico tras introducirlos. Por tanto, no se estima la aparición de otros que no sean esos, dado a que el entorno es cerrado y opaco. Esto no solo evitará la aparición de objetos extraños, sino también problemas relacionados con la contaminación lumínica. Este último factor es crucial de cara a la correcta aplicación de las técnicas de visión por ordenador que se emplearán para la detección de eventos. Como consecuencia, se puede centrar el esfuerzo de desarrollo en la detección de eventos y la optimización, en cuanto a coste de computación se refiere, de los mismos.

- El entorno de prácticas incorpora por si mismo los recursos multimedia que se requerían para el despliegue de la supervisión automática descrita en el proyecto SUPERVASION. El propio dominio prevé el uso de cámaras, dado el carácter poco invasivo que pretende este tipo de cirugía. Por este motivo, los simuladores de entrenamiento serán una reproducción fiel de estos condicionantes. Estos box dispondrán de cámaras y luz artificial, para capturar la escena de su interior. Los aprendices tendrán, por tanto, que entrenar las habilidades inherentes a manejar el instrumental quirúrgico a través de un monitor, teniendo como referencia la imagen de vídeo únicamente.

- Dentro del dominio, existen gran cantidad de ejercicios de entrenamiento, lo que abre el abanico de posibilidades de desarrollar ejemplos en el proyecto SUPERVASION. Además, muchos de los elementos son comunes entre ellos, por lo que una vez modelados sus eventos, se pueden reutilizar para los siguientes ejercicios que se registren. Gracias a que también existen variaciones sobre los mismos, con el modelado de un único ejercicio, se puede disponer de varias versiones del mismo de cara a tener más muestras y ejemplos, minimizando el esfuerzo de desarrollo.

- Gracias a la potencia y posibilidades que ofrece la programación inductiva se reduce sustancialmente la labor de programación en los nuevos ejercicios que se introdujeran. La responsabilidad de generar la nueva lógica recaería sobre la computadora a través de la fase de adquisición de conocimiento. Además la adaptabilidad del sistema propuesto abre la puerta a poder trasladarlo a otros entornos y dominios, donde se pudiera introducir la observación artificial y descripciones a alto nivel de los procedimientos a supervisar.

- En caso de llegar a una versión comercial, podría ser una alternativa económica a los sistemas de simulación, tanto virtuales como tradicionales, que hay disponibles actualmente. Los sistemas actuales ofrecen una gran precisión en cuanto a detección y medición de gestos se refiere, siendo difícilmente igualables por otros medios, como la visión por ordenador. Pero también es cierto que estos simuladores poseen costes muy elevados, estando muy restringido su uso a aquellas instituciones que se los puedan permitir. Como consecuencia, han surgido en Internet numerosos tutoriales y artículos sobre la elaboración de estos entornos de forma económica y reproduciendo las características de los originales [70]. Teniendo en cuenta lo anterior, el sistema propuesto por SUPERVASION es mucho más económico, pudiendo ser de interés para los casos de entrenamiento básicos o introductorio a la cirugía laparoscópica, donde el coste pueda ser una barrera de iniciación. Además, al emplear los mismos recursos que ya disponen los entornos de entrenamiento, se pueden integrar sobre ellos de una forma

relativamente sencilla. Por último, estos sistemas no incorporan la parte de la supervisión automática y *online* que pretende SUPERVASION, ofreciendo una factor diferenciador y que puede repercutir positivamente en el aprendizaje de los alumnos.

Adquirir destreza en el manejo de instrumental quirúrgico requiere un gran número de horas de entrenamiento y dedicación. A este reto hay que sumarle las propias condiciones que rodean la cirugía laparoscópica, lo cual exige mucha más precisión, sensibilidad y destreza que otras cirugías. Competencias como el manejo de distintos tipos de instrumentos: pinzas, tijeras, disectoras o empujanudos; así como su uso a través de una cámara, serán requisitos básicos para la formación de cirujanos profesionales. En este sentido, uno de las principales limitaciones en las fases iniciales de entrenamiento consiste en la adquisición de experiencia relacionada con la disociación mano-ojo. El aprendiz debe de entrenar habilidades como la óptica espacial o la sensibilidad con el instrumental, para poder moverse en un entorno tridimensional, teniendo como referencia la imagen plana que captura la cámara. Esto tiene especial relevancia cuando distintos objetos pueden tapar u ocluir la visión de la cámara, dificultando el ejercicio, debiéndose tener experiencia suficiente para tener otras referencias más allá de la visión.

Las rutinas de entrenamiento están enfocadas para la adquisición de destrezas de distintos tipos. Por ejemplo, existen ejercicios destinado al entrenamiento de competencias generales, como la coordinación y adaptación a las dos dimensiones o al manejo de instrumental con ambas manos. Por otro lado, hay otras rutinas orientadas a habilidades específicas, como puedan ser el anudado o el corte. Un ejemplo de ejercicio del primer tipo, se puede encontrar en el Apéndice A de este trabajo (Figura A.1). Específicamente, este ejercicio está destinado a que el aprendiz adquiera la habilidad de coordinar mano y ojos. El ejercicio en sí consistirá en el pinzado de pequeños objetos y su traslado de un contenedor a otro, residiendo la dificultad en realizarlo con distintas manos y teniendo como referencia únicamente el monitor que emite la señal de la cámara.

La metodología que se emplea para introducir a los aprendices en la cirugía laparoscópica consiste en un curso de 50 horas de duración, dividido en teoría y prácticas. Los estudiantes, inicialmente, reciben las clases de teoría (14 horas), donde se les presenta el material laparoscópico, los tipos que existen y cómo se clasifican. También se les describe el entorno de prácticas, como puedan ser los sistemas de visualización y distintos tipos de cajas de entrenamiento. Por último, se les detallan conceptos específicos relacionados con las habilidades más complejas, como el anudado o el corte. Tras la teoría, comienza el periodo de prácticas y entrenamiento (36 horas). Para estas sesiones, los aprendices disponen de un boletín de ejercicios, donde se describe textualmente cada una de las rutinas a realizar paso a paso. Complementario a esto, a los alumnos también se les facilita un ejemplo en vídeo de cada ejercicio. Con estos recursos, los alumnos tienen que repetir las rutinas hasta que hayan desarrollado las competencias y habilidades que se requieren. Durante estas sesiones, el profesor va desplazándose por el aula de prácticas, observando y supervisando la ejecuciones de las rutinas de los alumnos, mientras les asesora y corrige cuando es necesario. Finalmente, y concluidas las horas de prácticas, los estudiantes son evaluados con respecto a unos parámetros relacionados con las capacidades que han tenido que desarrollar con estos ejercicios. Se valorarán especialmente factores como el tiempo de realización, soltura y coordinación con el instrumental o la precisión en la ejecución, entre otros.

En relación a las aulas de prácticas y los entornos de entrenamiento, en la Figura A.2 del Apéndice A se puede ver un ejemplo de una de ellas. Concretamente, el entorno de entrenamiento constará del simulador laparoscópico y una pantalla o monitor por el que se verá la señal de la cámara del interior. Existen varias formas de denominar al simulador, refiriéndose a él como box-trainer, endotrainer o pelvitrainer. Este dispositivo consiste en una caja de proporciones pequeñas y opaca (véase Figura A.3 del Apéndice A), con una cámara y foco de luz colocados en una de sus paredes interiores. En la parte superior dispone de una serie de orificios para introducir el instrumental que se requiere para la elaboración del ejercicio. Previo al comienzo de la rutina, se abre el simulador para colocar en su interior los distintos elementos que intervendrán en el

entrenamiento, con la distribución y configuración que se indica en la descripción del boletín, cerrándose el simulador posteriormente para comenzar el entrenamiento.

Podemos encontrar diversos modelos y alternativas de simuladores laparoscópicos en Internet. La gama de simuladores *eoSim MIS simulators*[5] ofrecen una equipación completa compuesta por distintos tipos de instrumentos quirúrgicos, caja de entrenamiento con cámara y un set de ejercicios con varios módulos de entrenamiento. La gama se divide en Core, Advanced y Elite, dependiendo del conjunto de módulos que incorporen, oscilando sus precios entre los 750€ y 1,300€. Además, todos ellos disponen de un software de *tracking*, denominado *SurgTrac*[55], encargado de la toma de datos del comportamiento del instrumental durante los ejercicios de laparoscopia. Entre las métricas que calcula se pueden encontrar la velocidad y suavidad de los movimientos efectuados o la distancia entre instrumentos, mostrando finalmente una puntuación para cada una de ellas. Cabe remarcar que incorpora un sistema de feedback textual, proporcionado por SurgTrac, que a la conclusión del ejercicio, muestra consejos y recomendaciones en función de la evaluación obtenida en cada métrica.

La empresa Inovus Medical, por su parte, ofrece también un simulador completo llamado *Pyrux*[6]. En su oferta de modelos encontramos que pueden incorporar el sistema completo de visualización, con monitor regulable en altura y posición, y un simulador de introducción del trocar para entrenamientos de este tipo. Ofrece, asimismo, varias configuraciones de accesorios. El precio puede oscilar entre 455€ y los 1,733€, dependiendo del conjunto de utensilios seleccionados. Dicha empresa posibilita también la adquisición del software SurgTrac y la integración de éste en su simulador. Por último, , prestan un repositorio con vídeos de ejemplos de ejercicios, así como, la posibilidad de poder registrar en vídeo nuestras prácticas y publicarlas en Youtube.

Entre los simuladores comerciales más asequibles podemos encontrar el modelo *LAPARO ASPIRE* de la empresa Laparo Medical Simulators. Al igual que los casos anteriores, también dispone de varias versiones comerciales, difiriendo en los módulos de entrenamiento que incorpora. Sus precios se hallan entre los 355€ de su modelo básico, hasta los 627€ de su gama más alta. Las prestaciones que proporciona son más simples, incluyendo instrumental y simulador con cámara y luz artificial. En relación a los módulos de entrenamiento, estos están enfocados principalmente a la ejercitación de la coordinación y destreza, proponiendo distintos tipos de retos de habilidad, y estando orientado así hacia las etapas más tempranas del aprendizaje de la cirugía laparoscópica.

2.4 Técnicas de visión por ordenador

En esta sección se describirán las técnicas de visión por ordenador que se han empleado para el modelador y detección de eventos a bajo nivel. Estas técnicas serán las siguientes: segmentación, *tracking*, visión estereoscópica por ordenador y reconocimiento de objetos. Cada una de ellas será descrita de forma resumida, junto con la motivación y aplicabilidad para el problema en cuestión.

2.4.1. Segmentación de imágenes

El proceso de segmentación consiste en descomponer una imagen en las partes u objetos que forman la escena que retrata [33]. De esta manera, es posible localizar ciertas regiones que sean relevantes bajo un criterio de segmentación, pudiendo así delimitarlas para un posterior análisis o uso. Como puede apreciarse, existirán dos tareas en este proceso: una de descomposición, buscando los elementos mínimos que componen la imagen para; y otra de agrupación, formando estructuras más complejas que doten de sentido a lo que se ve en la escena capturada [16].

[5] https://www.eosurgical.com/collections/simulators-individual?
[6] http://www.inovus.org/simulators

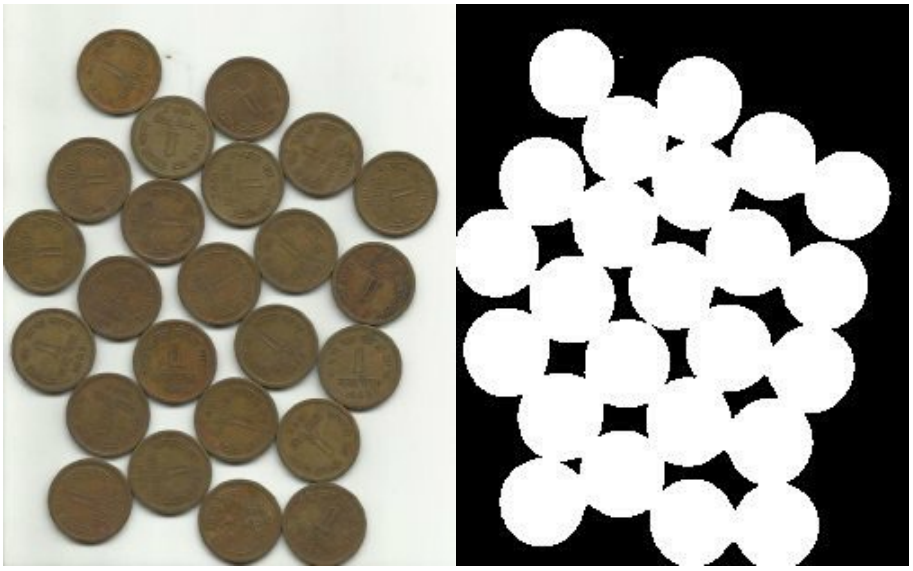


Figura 2.7: Ejemplo de aplicación de la técnica de segmentación. (Imagen extraída de [84]).

Partiendo de una imagen donde se muestran diversos objetos, estos suelen tener dos propiedades que los algoritmos de segmentación aprovechan para poder discernir su contorno y forma, y así poder separarlos del resto. La primera propiedad consiste en la similaridad de píxeles. Esto significa que los píxeles que forman un objeto en una imagen tendrán cierto aspecto uniforme que permite tratarlos como partes de una única unidad. Esta propiedad posee limitaciones considerables, ya que puede ser altamente sensible a la iluminación, ruido o reflejos en otros elementos. Sin embargo, existen técnicas como la segmentación en cascada (*Watershed Segmentation*) [33] o algoritmos de *clustering* [14], que permiten mejorar la uniformidad de los píxeles dentro de los objetos, facilitando su segmentación [49].

En este sentido, la técnica más extendida que permiten explotar la similitud entre los píxeles es la umbralización o *thresholding*. Esta técnica hace posible filtrar los píxeles de una imagen que cumplen cierta propiedad, como por ejemplo, un rango de brillo o color; destacando dichas zonas sobre el resto de la imagen. Algoritmos como Otsu [63], permiten calcular el rango óptimo de umbralización, generando una nueva imagen en blanco y negro o binarizada. Esta imagen solo mostrará las regiones que se encuentran dentro del rango, denominándose al resto de la escena fondo o *background*, que se dejará de color negro. La especificación de estos rangos se recogerá en una *máscara*, la cual se aplicará sobre la imagen a segmentar, dejando pasar solo aquellos píxeles que cumplan la máscara definida.

La segunda propiedad que se suele aprovechar en imágenes, en combinación con la anterior (similaridad), será la de discontinuidad. Dicha propiedad está basada en la detección de zonas con grandes contrastes dentro de la imagen, lo que suele estar asociado a contornos de objetos. Existen diversos algoritmos de detección de estos contornos [81] [56], siendo el algoritmo Canny [13] uno de los más empleados. De un contorno detectado se podrá calcular otros recursos de interés, como su visualización, su centro de masas o una figura geométrica que se ajuste y simplifique ese contorno.

En la Figura 2.7 se muestra un ejemplo de aplicación de la técnica de segmentación por medio del algoritmo Otsu [63], donde se busca diferenciar un conjunto de monedas del fondo de la escena. Como puede apreciarse, inicialmente se ha tenido en cuenta la similitud entre píxeles para encontrar los que pueden formar parte de un objeto común. Posteriormente, el algoritmo es capaz de detectar la discontinuidad de píxeles que existe en los bordes de las monedas, pudiendo generar una imagen binarizada donde se diferencia los objetos del fondo de la escena.

Con relación al trabajo desarrollado, se ha empleado la umbralización para detectar cuando los elementos que intervienen en el ejercicio están presente en la escena, así como de cuales se tratan. Por ejemplo, se han adherido ciertas marcas de colores al instrumental quirúrgico usado durante el desarrollo. Dichas marcas son fácilmente segmentables por medio de su umbralización, pudiendo detectar así cuándo aparecen en escena y la región de la imagen en la que están. Este proceso se podrá ver con mayor detalle en la subsección 3.3.1.

2.4.2. Tracking

En el campo de la visión por ordenador, el concepto de tracking es conocido como el seguimiento de un objeto dentro de una secuencia de imágenes mientras éste se mueve por la escena. Para que esto se pueda realizar, se requiere una combinación de técnicas que permitan el reconocimiento del objeto y su localización espacial en la imagen. Del desplazamiento de esta localización a lo largo de una sucesión de imágenes, se podrá extraer información de interés como, por ejemplo, la trayectoria del objeto rastreado [17].

Previo a que la computadora pueda rastrear un objeto, éste debe de ser identificado y reconocido por la misma. Esto requerirá técnicas como la umbralización o la detección de contornos, que tomando de referencia el color o la forma del objeto, permitan detectarlo en una imagen. Pueden verse estas técnicas con mayor profundidad en la subsección anterior.

Existen distintas maneras de representar la localización del objeto, siendo el punto la más simple y extendida. También es común el uso de una figura geométrica que acote y caracterice el objeto monitorizado. Usualmente, se suele emplear una figura geométrica simple, como una circunferencia o rectángulo, que no requiera excesivo cómputo para su generación. De está figura se suele calcular el punto central (centroide o centro de masas) y tener así un punto de referencia en el espacio. El proceso de tracking será, por tanto, la monitorización del comportamiento de este punto a lo largo de las imágenes capturadas [93].

La implementación más básica de tracking consiste en asumir que la imagen es un sistema bidimensional, o plano cartesiano, tomándose un punto dentro de ella como eje de coordenadas. De forma habitual, se suele escoger una de las esquinas, como la superior izquierda, y medir en píxeles las distancias desde ese punto al punto del objeto que se está siguiendo. De la variación de posición de este último punto con respecto al eje, se deducirá el movimiento del objeto por la escena. El sistema planteado permite extender la técnica de tracking a no solo el seguimiento de un objeto, sino a muchos otros dentro de una misma secuencia de imágenes. Lo que posibilita saber si los objetos se aproximan o se alejan, o bien qué trayectorias llevan a lo largo de la escena.

Sobre el desarrollo realizado, la técnica de tracking ha sido empleada para la detección de la posición y movimiento de los distintos elementos dentro del entorno laparoscópico. A partir de sus posiciones, y las variaciones de éstas, podremos detectar también cuando dos elementos estén aproximándose o alejándose dentro de la escena. Para poder lograr la aplicación del tracking, se ha usado una vez más las marcas del instrumental como referencia. Por medio de la umbralización, se ha logrado obtener el centro de masas de cada una de ellas. Del comportamiento de dicho centro de masas a lo largo de las secuencias de imágenes se podrá deducir su posición y movimiento. Asimismo, a través del cruce de estos datos con otros centros de masas, se podrá deducir cuándo estos se acercan o se alejan.

Como se habrá podido apreciar, el entorno donde se debe de aplicar el tracking es tridimensional, mientras que las técnicas descritas hasta el momento trabajan exclusivamente en las dos dimensiones de una imagen. Será, por tanto, necesario combinar la técnica de tracking con otros recursos que permitan completar la información faltante. Determinar la localización espacial de un objeto en un espacio de tres dimensiones, empleando la visión por ordenador, es un hecho complejo que puede ser abordado de diversas maneras. Una de ellas, consiste en la combinación del tracking junto con el uso de múltiples cámaras. A través de dos o más cámaras, se puede generar nuevos recursos

que permitan completar la información de la escena grabada, pudiendo derivar la distancia a la que se encuentra un objeto con respecto a ellas. De entre las técnicas de visión por ordenador que, empleando varias cámaras, puedan ofrecer esta información, la visión estereóscopica es una de las más extendidas y sencillas de aplicar.

Seguidamente se pasará a describir en qué consiste dicha técnica, junto con uno de los recursos más comunes que puede generar, como es el *mapa de disparidad*. Además, se describirá el proceso de calibración, necesario para poder aplicar la visión estereoscópica de forma apropiada.

2.4.3. Visión estereoscópica por ordenador

La visión estereoscópica por ordenador permite, por medio del uso de varias cámaras, conocer información sobre la profundidad de la escena que están grabando. Al igual que el mecanismo de visión de muchos seres vivos, humanos entre ellos; ésta requiere de dos imágenes de la misma escena, obtenidas en el mismo instante, pero desde planos ligeramente distintos. En los seres vivos, el cerebro superpone ambas imágenes, percibiendo las similitudes y diferencias entre ellas. De esta información es capaz de deducir la proximidad de los objetos en una escena que esté presenciando. En el campo de la visión por ordenador, estas similitudes y diferencias entre imágenes se conoce como disparidades, calculándose por medio de geometría epipolar.

En la figura 2.8 se muestra un ejemplo de escenario donde se aplica esta geometría a la visión estereoscópica por ordenador. Dicho escenario estará formado por dos cámaras y un objeto. Las cámaras vendrán representadas por sus correspondientes focos O y O', junto con el respectivo plano que forma la imagen que capturan (delante de cada uno de ellos). El elemento X indica el objeto, representado como un punto de la escena, y del que se desea conocer la profundidad a la que se encuentra. Los valores x y x' corresponderán a las proyecciones de dicho punto en los planos de las respectivas imágenes capturadas. Sobre los puntos e y e', estos representarán el punto donde se localiza cada foco en el plano de la cámara contraria. Estos puntos se conocerán como epipolos o *epipoles*, mientras que la línea que los une, junto con los focos, se denominará linea base o *baseline*. Por otro lado, el plano triangular formado por los puntos O, X y O' se denominará como epiplano o *epiplane*. Finalmente, las rectas l y l' se conocen como líneas epipolares o *epilines* y serán las que permitan la triangulación de la posición 3D del punto.

La idea detrás del escenario descrito reside en que, partiendo de una sola cámara, no es posible tener ninguna referencia de la profundidad a la que se encuentra el objeto con respecto a la cámara que lo captura. Esto se puede ver en dicha figura, tomando únicamente la cámara izquierda como referencia. Así pues, todas las proyecciones del punto X a lo largo de la recta OX, se superpondrán en un único punto en la imagen (punto x en el plano de la izquierda). Lo anterior impide discernir la información de profundidad del objeto, ya que esa información está solapada en un solo punto de la imagen. Aunque hoy en día existen propuestas para poder extraer información de profundidad de una única imagen [73], el empleo de la visión estereoscópica por medio de una segunda cámara, sigue siendo el método más usado para este propósito.

De vuelta a la figura, nótese que la segunda cámara (parte derecha de la imagen) está ligeramente desplazada con respecto a la primera, ofreciendo una perspectiva parcialmente distinta del objeto. Esto se traducirá en que existan distintas proyecciones de los diferentes puntos X, a lo largo de la recta OX, en la imagen de la derecha (puntos x'). Estos puntos deberán estar recogidos dentro de la línea epipolar l' (cámara derecha), cumpliéndose así la restricción epipolar o *epipolar contraint*. Esta restricción obliga a que las cámaras estén correctamente colocadas para poder cumplir dicha restricción, obteniendo de esta manera la máxima correspondencia de puntos X en la línea epipolar l'. Gracias a estas líneas epipolares, se podrá triangular la posición del objeto dentro del espacio tridimensional; pero para ello se requerirá trazar varias de ellas para maximizar la precisión del cálculo.

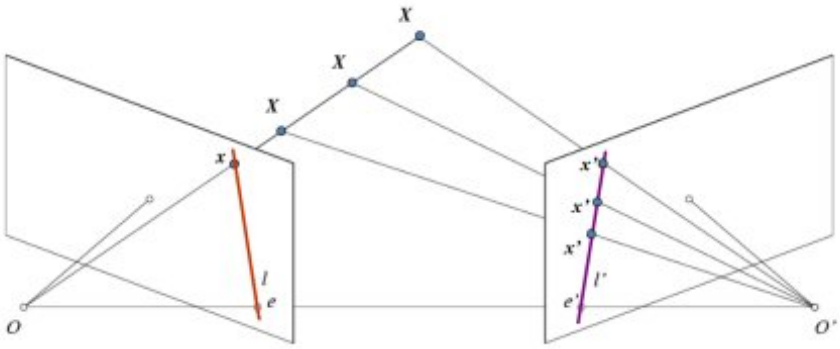


Figura 2.8: Escenario de aplicación de aplicación de la geometría epipolar a la visión estereoscópica por ordenador. (Imagen extraída de [83]).

La extracción de líneas epilolares que sean significativas y que cubran el mayor número de regiones posibles de la imagen, es un problema sustancialmente difícil de resolver. Este proceso no solo consistirá en la elección de varios puntos, sino que se necesitará encontrar su correspondiente proyección en ambas imágenes y que estas proyecciones correspondan unívocamente al punto de la escena real. Existen diversas aproximaciones de cara a poder obtener estos puntos. Tradicionalmente, se ha resuelto por medio de la estimación de la *matriz fundamental* [52]. Esta matriz recoge la correspondencia entre los puntos de ambas imágenes por medio del cálculo de la orientación de las cámaras con respecto a la escena (transformación proyectiva). Si bien es cierto que este método permite resolver el problema de geometría epipolar planteado anteriormente, por medio de la matriz fundamental, también es cierto que es sensible a errores en la correspondencia correcta entre los puntos de las imágenes. Otras aproximaciones buscan la detección de ciertas características dentro de la escena que permitan usarlas de referencia para establecer la relación entre ambas imágenes estereoscópicas. De esta manera, se buscarán características que estén presentes en ambas imágenes y que sean fácilmente detectables. Un ejemplo de ello puede ser la detección de bordes o esquinas de objetos dentro de las imágenes, usándose como referencia para el trazado de las líneas epipolares [32]. En este sentido, existen algoritmos que permiten la detección automática de estas características entre imágenes, como pueda ser el algoritmo SIFT [51]. Referencias al respecto de su uso para visión estereoscópica se puede encontrar en [83].

Tras la estimación de los puntos de referencia y líneas epipolares en ambas imágenes, se necesitará disponer de un método de triangulación de posiciones dentro la escena tridimensional. Esto se llevará a cabo por medio del cálculo de las diferencias entre los puntos localizados en las líneas epipolares. El resultado de estas diferencias generará una imagen con la información de profundidad de la escena, conociéndose a éste como mapa de disparidad.

2.4.3.1. Mapa de disparidad

El mapa de disparidad consiste en una imagen en escala de grises, donde la intensidad de cada píxel está inversamente relacionada con la distancia de las cámaras al punto de la escena que representa. Este mapa estará formado por la información capturada por las imágenes estereoscópicas, teniendo en consideración las disparidades existentes entre ambas imágenes. El escenario para la obtención de la disparidad entre ellas vendrá representado por la figura 2.9. En dicho diagrama, los valores x e x' serán las distancias desde el borde izquierdo de cada imagen hasta su respectiva proyección del punto X. El valor f será la distancia focal de las cámaras, siendo B la línea base de separación entre las mismas. Estos dos valores serán conocidos y se obtendrán durante el proceso de calibración de las cámaras, el cual se verá en el apartado 2.4.3.2 de esta subsección. Finalmente, el valor Z representará la profundidad que existe entre el punto X y las cámaras. Partiendo de este

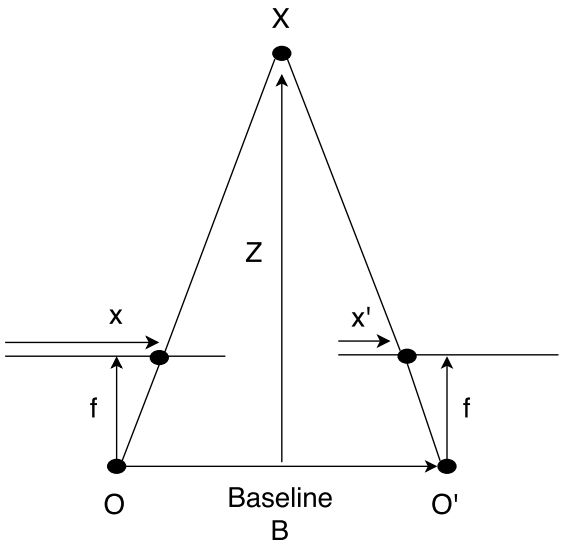


Figura 2.9: Diagrama para el cálculo del mapa de disparidad con imágenes estereoscópicas. (Imagen extraída de [83]).

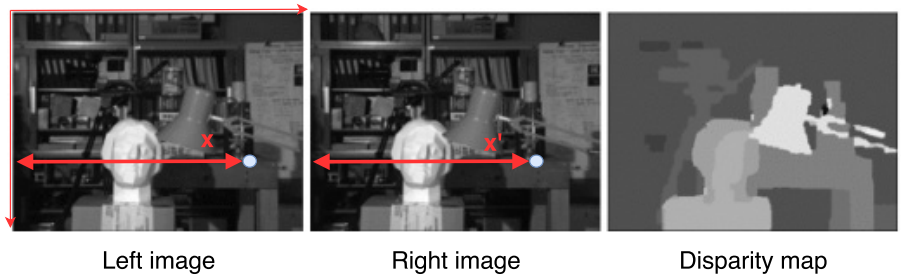


Figura 2.10: Ejemplo de mapa de disparidad. (Imagen adaptada de [57]).

escenario, la ecuación que permite calcular la disparidad, y así derivar el valor de Z, vendrá dada a continuación (ecuación 2.1):

$$disparity = x - x' = \frac{Bf}{Z} \tag{2.1}$$

De esta ecuación se deduce que la profundidad de un punto dentro de la escena tridimensional (Z), estará inversamente relacionada a la diferencia entre las distancias x y x'. Esto significa que los objetos más próximos a las cámaras poseerán una disparidad mayor, es decir, una intensidad en escala de grises más alta dentro del mapa de disparidad. Por contra, los objetos más alejados tenderán a una intensidad de gris más baja, hasta fundirse con el fondo, que en el mapa de disparidad se representa con el color negro. Un ejemplo del resultado que se obtendría tras el cálculo de disparidades entre dos imágenes estereoscópicas puede verse en la Figura 2.10. En este caso, tomando como eje de coordenadas en cada imagen la esquina superior izquierda, el cálculo de la diferencia entre distintos puntos x y x', irán formando el mapa de disparidad. En la derecha de esta imagen se puede ver cómo los objetos más próximos poseen una intensidad de gris más elevada que los objetos más distantes.

Como se ha comentado anteriormente, será necesario conocer la distancia focal y la distribución de las cámaras con respecto a la escena para poder obtener la disparidad. Estos datos estarán relacionados con la configuración interna de las cámaras, así como del escenario donde se esté aplicando la visión estereoscópica. Por tanto, gracias al proceso de calibración, no solo se podrá

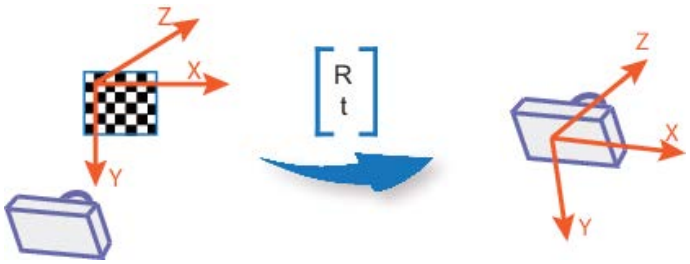


Figura 2.11: Proceso de transformación rígida por medio de los parámetros extrínsecos. Estos parámetros representan la posición de la cámara con respecto a la escena, siendo R la rotación de la cámara y t la traslación de la misma. (Imagen extraída de [24]).

extraer esta información, sino que además se podrán corregir ciertas distorsiones inherentes a los focos de las cámaras.

2.4.3.2. Calibración

El proceso de calibración de una cámara destinada a visión estereoscópica se realizará con el objetivo de calcular su matriz de cámara. Esta matriz hará posible que, dado un punto de la escena en el mundo real, se pueda conocer su correspondencia dentro de las imágenes capturadas. Para obtener esta matriz se requerirán datos sobre cómo están colocadas las cámaras en la escena y qué configuración interna poseen. Estos datos serán conocidos como parámetros y estarán divididos en extrínsecos (orientación de la cámara) e intrínsecos (ajustes internos de la cámara). A su vez, pueden existir distorsiones producidas por imperfecciones en la cámara que alteran el ajuste de estos parámetros. Es por este motivo que se requerirá calcular, a su vez, unos coeficientes de corrección que permita rectificar la distorsión en las imágenes.

Profundizando en estos parámetros, los extrínsecos representarán la disposición espacial de la cámara, es decir, la posición y orientación de la cámara con respecto a la escena. Por tanto, se buscará ajustar estos parámetros para lograr la relación entre el sistema de coordenadas del mundo real con respecto al sistema de coordenadas de la cámara. De esta forma, dado un punto dentro de la escena en el mundo real se podrá calcular su posición dentro de un sistema de coordenadas que parta de la cámara. Este proceso se denominará transformación rígida, mostrándose gráficamente en la figura 2.11. Los parámetros extrínsecos serán los encargados de especificar esta relación entre sistemas de coordenadas, recogiéndose en la matriz de rotación-translación ($R|T$ matrix). Durante la fase de calibración se calculará la rotación y traslación de la cámara con respecto al patrón empleado para calibrar, recogiéndose esta información en la matriz mencionada.

Sobre los parámetros intrínsecos, estos representarán la configuración interna de la cámara, y estarán formados principalmente por la distancia focal y el centro óptico. Permitirán realizar la proyección de un punto dentro del sistema de coordenadas de la cámara, al plano formado por la imagen capturada. Estos parámetros son propios de cada cámara y se recogerán en una matriz K definida como (Ecuación 2.2):

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

donde el centro óptico vendrá representado por c_x y c_y y la distancia focal por f_x y f_y. Finalmente, y una vez calculada esta matriz, se podrá almacenar y reutilizarse sin necesidad de repetir la calibración, ya que son propiedades inherente a la cámara que se ha empleado.

Por otra parte, en la figura 2.12 se puede ver la relación que existe entre los parámetros previamente descritos. Inicialmente, se parte de un punto de 3 dimensiones en el mundo real. Por

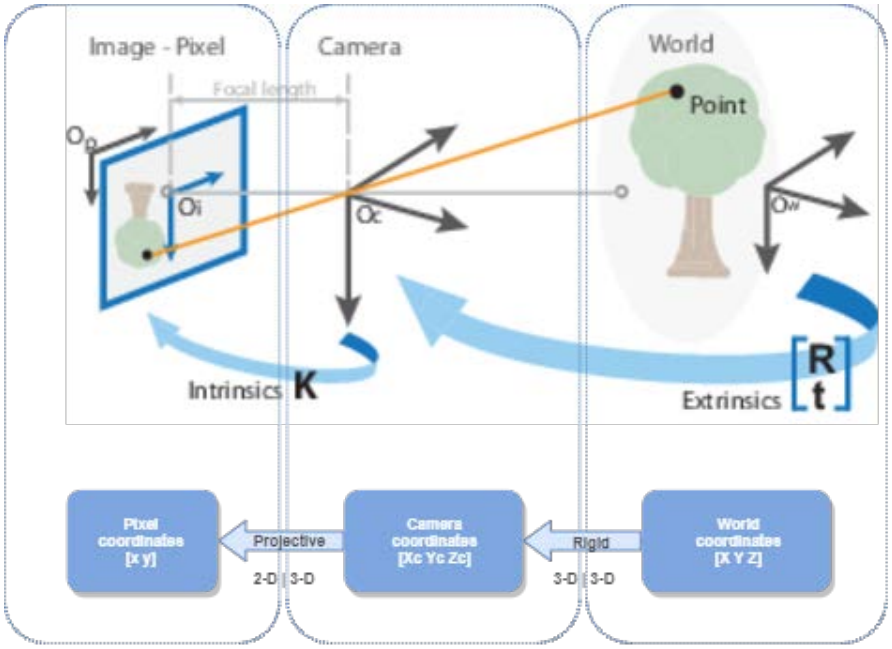


Figura 2.12: Relación entre parámetros extrínsecos e intrínsecos con el mundo real. (Imagen adaptada de [24]).

medio de la matriz $R|T$ (parámetros extrínsecos) se puede trasladar dicho punto a un sistema de coordenadas que parta de la cámara. Posteriormente, se realiza la proyección de dicho punto al plano de la imagen captada, usando la matriz K (parámetros intrínsecos). Esta relación vendrá definida formalmente por la operación que se muestra en la ecuación 2.3. En ella se puede observar que la matriz de cámara vendrá definida por el producto de los parámetros intrínsecos e extrínsecos. Posteriormente, multiplicará esta matriz por el punto en el mundo real para calcular el punto proyectado en el plano de la imagen capturada. Por último, a este punto se le multiplicará un factor s que permitirá reescalar el punto del mundo real a la resolución de la imagen [5].

$$\underbrace{s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{Image Point}} = \underbrace{\underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Parameters}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{\text{Extrinsic Parameters}}}_{\text{Camera Matrix}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\text{World Point}} \tag{2.3}$$

Como se comentó previamente, existe otro motivo por el que realizar la calibración y será el de conocer las distorsiones (aberraciones) que provoca la cámara en las imágenes que captura. Por medio de la calibración se conseguirá que afloren estas perturbaciones, pudiendo calcular unos coeficientes para rectificarlas y obtener así medidas más fiables. Habitualmente, las cámaras comerciales de gama media-baja, suelen ofrecer una imagen ligeramente distorsionada de la escena que están grabando. Esto se debe a que la lente que lleva en su interior posee imperfecciones que provocan que la imagen capturada no sea completamente acorde a la real.

Las aberraciones de tipo radial suelen ser las más comunes, las cuales provocan que las imágenes parezcan dilatadas o achatadas por los márgenes, acentuándose conforme nos aproximamos a los bordes. En la figura 2.13 pueden verse las distorsiones más usuales de este tipo, estas serán: barril (barrel), alfiletero (pincushion) y bigote (mustache). Las aberraciones radiales suelen estar relacionadas con el tamaño de la lente de captura, donde las lentes más pequeñas son las que más provocan estos tipos de distorsiones [24].

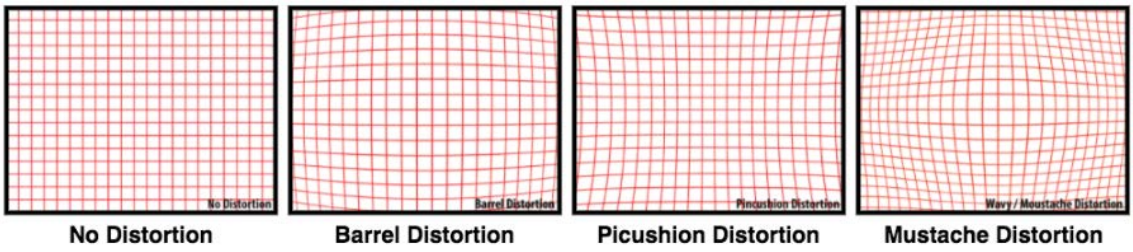


Figura 2.13: Tipos de distorsiones radiales (aberraciones) más comunes en imágenes (Imagen extraída de [54]).

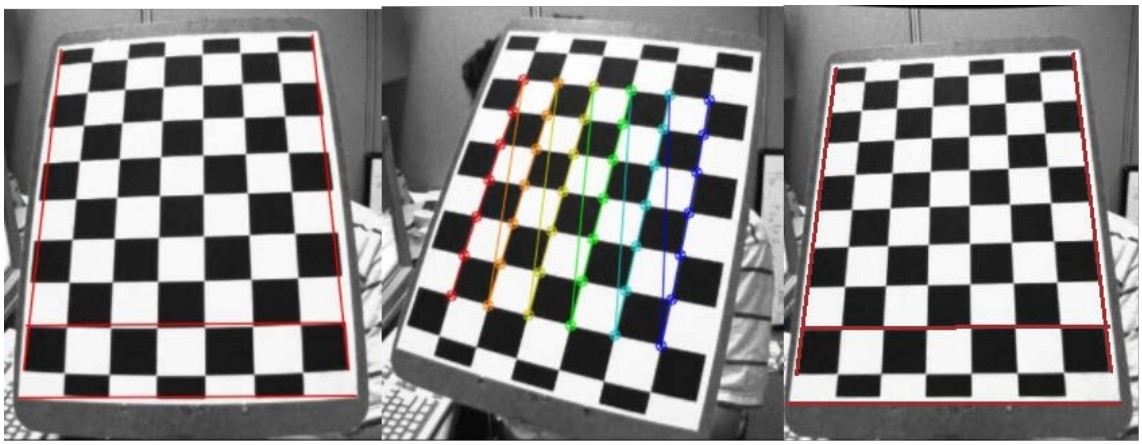


Figura 2.14: Proceso de calibración y rectificación de las cámaras con OpenCV. Inicialmente, la imagen posee una distorsión radial de tipo barril (izquierda). Tras el proceso de calibración (centro), se aplica la rectificación calculada corrigiendo la distorsión (derecha). (Imágenes extraídas de [82]).

Los coeficientes aplicados para este tipo de distorsión se pueden ver en la siguiente ecuación (Ecaución 2.4):

$$\begin{aligned} x_{corrected} &= x(1 + K_1 r^2 + K_2 r^4 + k_3 r^6) \\ y_{corrected} &= y(1 + K_1 r^2 + K_2 r^4 + k_3 r^6) \end{aligned} \tag{2.4}$$

Los coeficientes vendrán representados por $k1$, $k2$ y $k3$, siendo empleados solo los dos primeros coeficientes de forma habitual.

Por otro lado, un ejemplo de cómo la calibración puede corregir estas distorsiones se puede encontrar en la figura 2.14. En imagen de la izquierda se puede apreciar la distorsión de tipo barril que tiene el tablero de calibración. Seguidamente, en la imagen central se ha obtenido durante el proceso de calibración, concretamente, en la fase de detección de las esquinas internas del patrón de calibración. Finalmente, en la imagen de la derecha de ve cómo se ha corregido la distorsión una vez aplicada la rectificaciones calculadas.

Para llevar a cabo la calibración, se suele emplear una imagen con un patrón gráfico bien definido que permita al algoritmo de calibración el cálculo de los parámetros y la detección de distorsiones. El patrón visual más empleado para calibración suele ser el de tablero de ajedrez, aunque existen otros, como los patrones circulares [22]. Ambos se basan en patrones repetitivos, formados por figuras geométricas negras sobre fondo blanco, donde sea fácil identificarlas por los algoritmos de visión por ordenador. El objetivo principal será detectar los puntos en dicho patrón que sirvan de referencia espacial para la calibración, usualmente las esquinas internas que forman los cuadrados del patrón.

Asimismo, se requerirá la toma de un conjunto de imágenes donde se muestre dicho patrón frente a las cámaras. Se necesitarán numerosas muestras de imágenes, colocando el tablero en distintas posiciones de la escena que captan la cámaras, para que el proceso de calibración sea óptimo. En este sentido, se recomienda comenzar con desplazamientos laterales y ascendente/-descendentes del patrón de calibración, manteniendo la misma distancia de profundidad con las cámaras. Posteriormente, se irá desplazando a posiciones más alejadas, repitiendo en cada una de ellas los desplazamientos anteriormente mencionados.

Una vez obtenidas las imágenes, se procederá al reconocimiento del patrón dentro de cada una de ellas. En este paso se marcan las esquinas internas donde se intersectan los cuadrados blancos y negros. Estos puntos, junto con la relación espacial que existe entre ellos, serán usados por el algoritmo de calibración para calcular los parámetros intrínsecos y extrínsecos que definen la matriz de cámara, al igual que los coeficientes de rectificación.

Hoy en día, existen diversas librerías y soluciones informáticas que facilitan todo este proceso de calibración, agilizando tanto la fase de toma de imágenes y detección de esquinas, como la fase de cálculo de las matrices de cámara y coeficientes de rectificación. Matlab[7], por ejemplo, con su módulo de procesamiento de imágenes y visión por ordenador, ofrece una serie de herramientas que automatizan estas tareas. Por contra, si bien es cierto que OpenCV incorpora los algoritmos que se requieren durante la calibración, recae sobre el desarrollador la implementación del código que realice cada tarea del proceso de calibración. Es por este motivo por el que han proliferado en los últimos años varios desarrollos de terceros que facilitan esta labor para lenguajes como Python [26] [76].

Finalmente, tras finalizar el proceso de calibración, la forma de usar los recursos obtenidos consistirá en aplicar los coeficientes de corrección en cada imagen capturada, corrigiendo así las aberraciones. Después, se emplearán las matrices de cámara para poder derivar la profundidad de un punto de la escena, dada las imágenes rectificadas.

Como se comentó al comienzo de esta subsección, gracias a la visión estereoscópica y al mapa de disparidad, se podrá completar otras técnicas como la umbralización o la de tracking. En el caso particular de este trabajo, por medio del mapa de disparidad se ha podido agregar la información de profundidad a los datos de posición obtenidos por medio del tracking. De esta unión se calculará la posición tridimensional de los elementos dentro de la escena, cubriendo el espectro de posiciones posibles de el espacio tridimensional del entorno laparoscópico. Sobre la combinación de estas dos técnicas, es decir, tracking y visión estereoscópica, al igual que el proceso de calibración llevado a la práctica durante este proyecto, se verá con mayor detalle en la sección 3.3.2.

2.4.4. Reconocimiento de formas y objetos

En la actualidad, uno de los tópicos más activos dentro del campo de la visión por ordenador, es el de reconocimiento de formas y objetos. La principal motivación detrás de esta idea es que una computadora sea capaz de, dado uno o varios ejemplos de un determinado objeto, pueda detectarlo cuando está presente en una imagen. Para que esto pueda realizarse, la computadora tiene que ser capaz de extraer, inicialmente, las características que modelizan el objeto en cuestión (extracción de descriptores). Posteriormente, se analiza la imagen definiendo las regiones de interés donde se pueden localizar elementos que pudieran ser el objeto (generación de candidatos). Finalmente, se valoran los posibles candidatos usando las características encontradas previamente como criterio, escogiéndose el que mejor cumpla estas características (clasificación de candidatos).

Los pasos anteriores suponen una serie de retos, como la diferenciación objeto/fondo en una imagen, la variabilidad de forma que puede presentar el objeto o del entorno donde éste se está mostrando. Es por ello que existe una extensa bibliografía en cuanto algoritmos y técnicas que permitan afrontar estos retos. En lo referente a la extracción de descriptores, una de las

[7] https://es.mathworks.com/help/vision/index.html

aproximaciones más sencillas es la conocida como *Template Matching* [11]. En ella se emplea a modo de descriptor una imagen de detalle del objeto a reconocer, calculándose posteriormente la similitud o disimilitud de los posibles candidatos con dicho patrón. Una vez obtenidas, se escogerá el candidato que mayor o menor diferencia ofrezca, dado el criterio de clasificación. Otros métodos buscan descriptores que permitan generalizar el objeto, como puedan ser propiedades básicas (color, contorno, ..), evitando así la variabilidad de formas que puede presentar [86]. Por otra parte, se pueden generar descriptores, no apreciables a simple vista, pero que se pueden derivar de la relación entre los píxeles de la imagen, siendo más robustos de cara a cambios de iluminación o de brillo. Entre estos descriptores se pueden encontrar los LBP [65] o HOG [23]; o bien los producidos por la aplicación de filtros de Haar [88].

En lo que se refiere a la búsqueda de candidatos, una de las técnicas más empleadas suele ser la de *ventana deslizante* [88] [75]. Ésta consiste en la definición de un marco con un tamaño determinado (tamaño canónico) que se desliza por la imagen, evaluándose el contenido de la ventana con respecto al descriptor del objeto que se posee. Esta técnica suele estar complementada con el re-escalado a distintas dimensiones de la imagen objetivo, conociéndose este método como *pyramidal sliding window* [53]. Gracias a este re-escalado, se podrá cubrir las distintas posiciones de profundidad a la que se puede encontrar el objeto en la escena. Existen otros casos en los que, mediante un preprocesado de la imagen, se puedan delimitar las zonas donde se aplique el reconocimiento de objetos. Estas zonas se conocen como *Region of Interest* o ROI, siendo regiones acotadas donde se aplicará las técnicas de clasificación directamente, sin tener que aplicar algoritmos de búsqueda de candidatos.

Por último, sobre las técnicas de clasificación y discriminación de candidatos, de forma habitual se suelen emplear algoritmos del campo del aprendizaje automático. La elección del algoritmo más adecuado dependerá, en gran medida, de la naturaleza de los descriptores extraídos. En [23], por ejemplo, se emplean descriptores de tipo HOG, combinados con un clasificador lineal (*Support Vector Machine* o SVM), para discriminar los candidatos que son el objeto buscado del resto de candidatos. Otras soluciones, como la que se propone en [89], une la extracción de características de Haar, por medio de filtros, con el uso del algoritmo AdaBoost [29]. Con ello se obtiene un clasificador fuerte que permite discernir si el candidato es o no el objeto a reconocer.

Hasta el momento, se han descrito las técnicas tradicionales de reconocimiento de objetos, pero este campo ha experimentado un cambio considerable en los últimos años. Con el auge de las redes neuronales y las nuevas técnicas de *Deep Learning*, han nacido las denominadas *Convolutional Neural Networks* o CNNs, también conocidas como *ConvNets* [74]. Las redes convolucionales poseen gran cantidad de similitudes con las redes neuronales tradicionales, pero aportan ciertas diferencias que les permiten trabajar con imágenes de forma muy eficiente, reduciendo, por ejemplo, el número de parámetros a ajustar.

Una red convolucional estará compuesta por una secuencia de capas, existiendo siempre una capa de entrada o *input layer*, una o varias de capas ocultas o *hidden layers*, y una capa de salida o *output layer*. Cada capa oculta estará formadas por un conjunto de neuronas, donde cada una de ellas estará totalmente conectada a todas las neuronas de la capa antecesora, teniendo a su vez una función de activación asociada. La misión de la red convolucional, por tanto, consistirá en discernir a partir de los píxeles de una imagen, y capa tras capa, la clase del objeto que se presenta en ella.

Los componentes básicos de una red convolucional consistirán en los siguientes [42]:

- **Input layer:** Capa de entrada de datos de la red convolucional. Recibirá los píxeles de la imagen directamente, siendo una capa de tres componentes, es decir, ancho, alto y canales de color (R,G,B) de la imagen. Cabe destacar que cada neurona de la capa de entrada tendrá asignada, o se especializará, en una pequeña región de la imagen.

- **Convolutional layer:** Esta capa operará sobre la salida de las neuronas de la capa de entrada, las cuales están conectadas a una pequeña región de la imagen original. La operación que

realizará la capa de convolución consistirá en aplicar una serie de filtros de Haar sobre la región recibida. Una parte de ellos se destinarán a detectar altas frecuencias en escalas de gris, lo que permitirá detectar contornos. Complementariamente, otros filtros se focalizarán en detectar características de baja frecuencia, como son los canales de color. Por tanto, la dimensión de esta capa vendrá definida por el ancho, alto y número de filtros empleados.

- **RELU layer:** Capa que aplica la función de activación sobre la entrada de la capa convolucional. De forma habitual, se encontrará unida a esta última. Dicha función vendrá definida por la siguiente ecuación:

$$f(x) = x^{+} = max(0,x) \tag{2.5}$$

donde x será la entrada de la neurona. Esta función se conocerá como REtified Linear Unit o RELU, de ahí el nombre de esta capa. La aplicación de una función de activación permite reducir la linealidad en la toma de decisiones de la red convolucional.

- **Pool layer:** Esta capa suele aplicarse entre capas convolucionales, y consistirá en un redimensionamiento de la salida de la capa convolucional predecesora. Su función, por tanto, será reducir el volumen espacial de información sobre la que operarán las siguientes capas. Esto reducirá la cantidad de parámetros a ajustar a lo largo de la red convolucional, así como reducir en cierta medida el sobrejuste.

- **Full-connected layer:** Consistirá en la última capa donde se codifican las clases de salida de la red convolucional. Todas las neuronas estarán conectadas a la capa predecesora, de ahí el nombre de esta capa. Para discriminar las clases de salida, esta capa estará completada con una función que permita realizar esta separación como, por ejemplo una función Softmax o SVM.

Aunque los elementos anteriormente expuestos son los más comunes en la composición de redes convolucionales, existen muchas otras capas que permiten elaborar arquitecturas mucho más sofisticadas. Además, debido a las posibles combinaciones que se pueden hacer de las capas comentadas, no existe una arquitectura estándar de red convolucional. Y es que en los últimos años, el campo del diseño de arquitecturas para redes convolucionales ha experimentado un gran crecimiento. Tanto es así, que en la actualidad existe una gran cantidad de alternativas y propuestas de arquitecturas que, dependiendo del contexto de uso, pueden ofrecer mejor rendimiento en cuanto a tiempo o precisión en la clasificación [20] [42]. Entre ellas podemos encontrar la arquitectura clásica de LeNet [46], o las ganadoras de las distintas ediciones de la competición *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [8], como son la arquitectura AlexNet [44] o la GoogLeNet [79], entre otras. Esta competición ha sido un claro ejemplo del impacto que han tenido las *ConvNets* dentro del campo del reconocimiento de objetos, de igual modo que de su evolución y mejora.

En el presente trabajo, el reconocimiento de objetos ha servido para la detección del estado del instrumental quirúrgico. De forma más precisa, ha permitido determinar si el extremo de los distintos utensilios quirúrgicos empleados durante el proyecto estaban abiertos, cerrados o poseían algún elemento pinzado. Para lograr esto, se ha tenido que idear un método para poder extraer una imagen detalle de la pinza del instrumental, siempre que éste estuviera en escena. Una vez implementado el método, se ha usado para la recogida de muestras de posibles estados que puede presentar la pinza y que eran de interés para el dominio. Posteriormente, se ha creado la arquitectura de una red convolucional, la cual se ha entrenado con estas imágenes; creándose así un clasificador de estados del instrumental. Finalmente, se ha combinado el método implementado, junto con la red convolucional entrenada, para un detector de estados del instrumental identificar en tiempo real. Todo este proceso se verá con mayor detalle en el apartado 3.3.3.

[8] http://www.image-net.org/challenges/LSVRC/

CAPÍTULO 3
Detección de eventos usando visión artificial

En este capítulo se describirán los recursos y métodos empleados para desarrollar el prototipo que permita la detección de eventos a bajo nivel dentro del dominio de la cirugía mínimamente invasiva. Se presentará en detalle el simulador laparoscópico creado como entorno de pruebas. Se introducirán también los recursos de software usados durante el desarrollo. A continuación, se detallarán las adaptaciones que se han creado usando las técnicas de visión por ordenador expuestas en el capítulo anterior para la detección de eventos. El prototipo que las implementa se presentará en último lugar, junto con su estructura y pasos para ejecutarlo.

3.1 Creación del entorno laparoscópico

Como punto de partida, se ha necesitado un entorno físico del dominio donde poder realizar la experimentación con las técnicas descritas en el apartado 2.4 para la detección de eventos. Por este motivo, se ha reproducido lo más fiel posible un simulador laparoscópico de este tipo, emulando las características y funcionalidades de uno real. Nuestro entorno de ensayos consistirá una caja cerrada y opaca, en cuyo interior se halla unas cámaras con las que visualizar la escena, junto con iluminación artificial. En la parte superior se encuentran los orificios por donde introducir el instrumental que se requiera para el ejercicio en cuestión. Los distintos elementos relacionados con el entrenamiento a realizar (cuentas hama, pequeños contenedores de plástico o gomas elásticas) se colocan en la base de la caja, enfrente de la cámara y bajo la iluminación artificial.

La figura 3.1 muestra el prototipo de pelvitrainer que se ha fabricado durante el transcurso de este trabajo. Se ha usado como armazón una caja de plástico con las siguientes medidas: 73cm x 41cm x 32cm (Largo x Ancho x Alto). Nótese que el tamaño de este entorno es sensiblemente más grande que los modelos comerciales. Esto se debe a que ciertas técnicas, como la visión estereoscópica, requieren cierto rango de profundidad para la detección de eventos. Partiendo de este armazón, se ha forrado de vinilo blanco para evitar la influencia de luz externa. Esto, a su vez, ayuda a que destaquen más las marcas de colores empleadas para identificar el instrumental. Adicionalmente, se han efectuado unos orificios en la tapa superior de la caja para la inserción del material quirúrgico. Se han realizado en distintas posiciones para facilitar la realización de los ejercicios.

Dentro del simulador, se han adherido a la cara interna de la tapa dos lámparas LED, modelo *JESWELL USB LED Sensor Light*, para facilitar la visualización de la escena tras cerrar la caja. Respecto a su distribución, se ha buscado la mayor cobertura de luz para la zona de acción de los ejercicios, centrándose sobre la plataforma donde se realizan. Ayuda a este propósito que los focos sean de forma alargada, ofreciendo más superficie de iluminación. Además, la luz que emiten

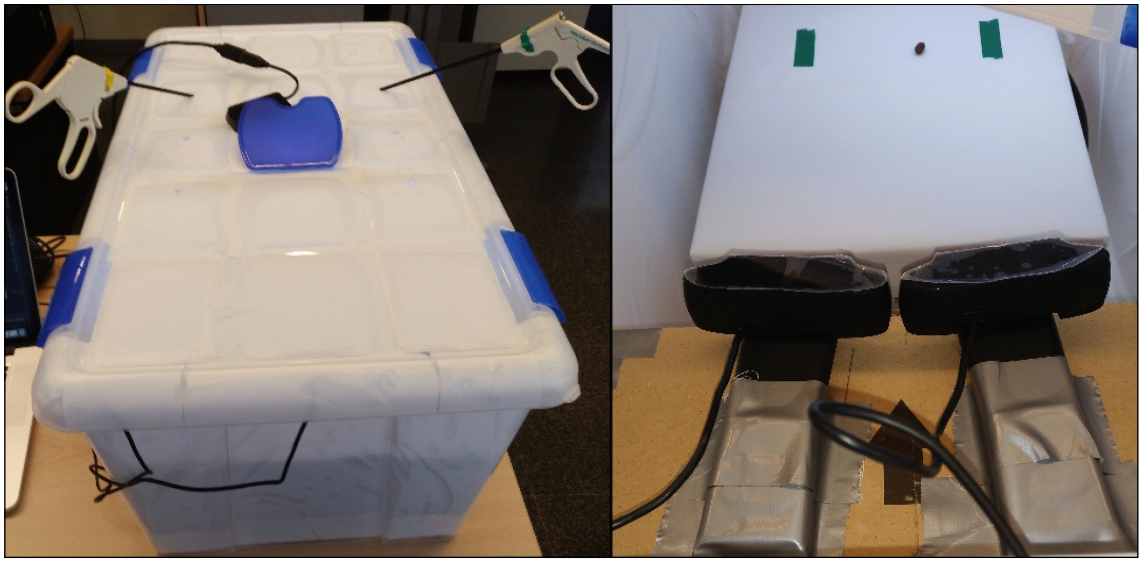


Figura 3.1: Prototipo de entorno laparoscópico desarrollado para el proyecto. (Imagen de elaboración propia).

es de tipo *blanco-cálido*, reduciéndose la alteración de colores de los objetos. A pesar de que poseen baterías de larga duración, se ha optado por conectarse al ordenador por medio de un HUB USB (Portable 4 Port SuperSpeed Mini USB 3.0 Hub - Black), que las dotará de corriente en todo momento. Esto evitará varianzas en la iluminación, manteniéndose constante a lo largo del proceso de grabación de los ejercicios.

Sobre las cámaras, se han empleado dos webcams (Logitech C922 Pro Stream Webcam) para poder ver en el interior del box de entrenamiento una vez se cierre; de igual modo que permitirán capturar la escena para el despliegue las técnicas de visión por ordenador que se necesitan. Perteneciente a la gama media-alta de webcams que ofrece Logitech, este modelo puede grabar en alta definición, capturando imágenes a una resolución de 1080p y con una frecuencia de 30 frames por segundo. También incluye el manejo de codecs de compresión de vídeo, como el H.264, que garantizan una alta calidad con menos ancho de banda, en caso de optar por un sistema distribuido para el procesamiento de los vídeos. Para el manejo de los parámetros internos de las cámaras, Logitech dispone del software *Logitech Gaming Software*[1] con el que ajustar el foco de la cámara o la configuración de colores. Este software ha sido de gran ayuda durante el proceso de calibración de las cámaras, con vistas a emplear la visión estereoscópica por ordenador; ya que permite desactivar el auto-enfoque y ajustar éste a las dimensiones reducidas del entorno. La necesidad de estereoscopía por ordenador motivará además el empleo de dos cámaras en lugar de una sola, como ofrecen los entornos laparoscópicos tradicionales. Se puede ver con más detalle lo relativo a la visión estereoscópica en el apartado 2.4.3 de este trabajo.

Por último, se ha añadido una plataforma blanca sobre la base de nuestro pelvitrainer. Sobre ella se realizarán las rutinas y ejercicios de laparoscopia, colocándose los distintos elementos que se manipularán durante la tarea. De esta manera, los instrumentos pueden alcanzar los objetos desde los orificios superiores de una forma más cómoda. Esta plataforma se ha cubierto también de blanco para evitar colisiones de color durante los procesos de detección de marcas sobre el instrumental. Asimismo, confiere una superficie plana sobre la que será más sencillo agarrar o dejar fijos elementos, evitando desplazamientos por irregularidades o pequeñas inclinaciones. Finalmente, y para una consulta más detalla de la configuración interna y medidas del entorno laparoscópico creado, se ha confeccionado un plano completo al respecto, estando disponible en el Apéndice B del presente documento (Figura B.1) .

[1] http://support.logitech.com/es_es/software/lgs

El instrumental quirúrgico empleado durante la experimentación puede verse en la Figura B.2 del Apéndice B. Estos han sido un fórceps *"AutoSuture Endo Clinch II 5 mm"* y un disector *"AutoSuture Endo Dissect 5 mm"*. Puede verse en esa imagen también, las marcas de colores que se les han añadido para poder identificarlos fácilmente. Otras marcas se han añadido sobre la plataforma de ejercicios para tener así las zonas de referencia que se requieren en ciertas rutinas de entrenamiento, como las de agarre y traslado de objetos. Por otro lado, se han usado judías de color rojo, fácilmente detectables mediante visión por ordenador, para representar esos objetos. La configuración completa de las marcas y colores empelados pueden consultarse en la Tabla B.1 del Apéndice B.

3.2 Lenguajes, tecnologías y herramientas

A continuación, se describirán brevemente los lenguajes y las tecnologías empleadas durante el desarrollo del prototipo de detección de eventos a bajo nivel.

3.2.0.1. Python[2]

Lenguaje creado por Guido van Rossum durante los años ochenta, es un lenguaje de script semi interpretado de amplia difusión dentro del mundo del desarrollo de software. Entre las muchas características que posee Python, destaca su portabilidad, sintaxis sencilla y la gran comunidad de desarrolladores que hay detrás, siendo el lenguaje más popular a mes de agosto de 2017, según el portal web *IEEE Spectrum*[15].

Para este proyecto se ha usado como lenguaje principal de desarrollo, debido a su facilidad de uso y a la gran cantidad de recursos disponibles que existen, principalmente paquetes (packages). A su vez, y debido al perfil técnico del proyecto, se ha necesitado de una plataforma de carácter científico, basada en Python, y que incluyera por defecto muchas de las librerías requeridas. La plataforma escogida ha sido *Anaconda* [3] de Quantuum Analytics. A esto se ha sumado el uso de entornos virtuales [25] (virtualenv) para encapsular las dependencias de librerías empleadas durante el desarrollo y facilitar la labor posterior de empaquetado y despliegue. Por último, como entorno de programación se ha usado el IDE PyCharm Community Edition 2017.1.5 [4].

3.2.0.2. JSON

Formato estándar para estructuración de datos, muy ligero y fácilmente procesable por una máquina. A diferencia de otros formatos como XML, que emplean etiquetado para estructurar la información; JSON emplea la notación de objetos del lenguaje JavaScript, creado diccionarios con pares atributo-valor. Este lenguaje se empleará para dotar de formato a la salida de nuestro prototipo, buscando facilitar su visualización e integración con futuros componentes de software.

3.2.0.3. OpenCV [5]

Librería de código abierto especializada en visión por ordenador, con interfaz disponible y con soporte para Python [9]. Esta librería recoge e implementa un gran número de técnicas del campo de la visión por ordenador y el aprendizaje automático, haciéndolas fácilmente aplicables e integrables en una aplicación. Actualmente se encuentra en su versión 3.2.0, no existiendo una retrocompatibilidad completa con su versión más difundida, la 2.4.

[2] https://www.python.org/
[3] https://www.continuum.io/downloads
[4] https://www.jetbrains.com/pycharm/
[5] http://opencv.org/

Un detalle a destacar es que OpenCV hace uso de la librería Numpy[62], entre otros recursos, para el manejo de imágenes como matrices . De este modo, logra que el uso de algoritmos, como los empleados para la segmentación o de detección de contornos, por ejemplo, se apliquen sobre las imágenes de una forma muy eficiente.

Para este proyecto se ha usado la versión 2.4.13.2, debido la compatibilidad que posee esta versión con las librerías de terceros que se han requerido para ciertas funcionalidades. Un ejemplo de esto se puede ver en el apartado 3.2.0.4 de este capítulo. Por otro lado, está librería ha sido de gran ayuda a la hora de modelar la detección de eventos a bajo nivel, gracias a los recursos que ofrece para aplicar técnicas de visión por ordenador. Es por ello que ha sido una pieza clave y fundamental para el desarrollo de los distintos módulos de detección, así como la gestión de las cámaras del entorno laparoscópico.

3.2.0.4. StereoVision[6]

Librería desarrollada para Python, y creada por terceros, que simplifica el proceso de calibración y rectificación de las cámaras, cuando éstas se emplean para visión estereoscópica con OpenCV. Como consecuencia del rediseño que se llevó a cabo en OpenCV, durante su evolución de la versión 2.4 a 3.0, diversos métodos que usaba la librería StereoVision dejaron de ser compatibles. Debido al peso que tiene este recurso en la detección de ciertos eventos, se optó por usar una versión inferior de OpenCV (2.4.13.2) pero que fuera compatible con esta librería.

En relación al proyecto, se ha usado el conjunto de herramientas que ofrecer StereoVision, tanto para el proceso de calibración de las cámaras, como, posteriormente, la aplicación de las rectificaciones a las mismas. Una descripción más detallada sobre lo anterior se puede encontrar en el apartado 2.4.3 de este capítulo.

3.2.0.5. Keras[7]

Keras es una interfaz, desarrollada en Python, que encapsula el uso de librerías especializadas en la creación y entrenamiento de arquitecturas de Deep Learning, como TensorFlow [1] o Theano [2]. Keras simplifica tanto el proceso de definición de la arquitectura de la red neuronal, como la fase de entrenamiento, evaluación y puesta en producción de la misma.

En nuestro caso, se ha empleado para crear una red convolucional que permita detectar el estado del instrumental que realiza el ejercicio. Por tanto, se ha usado para definir la arquitectura de dicha red, de igual manera que para entrenarla y evaluarla. Adicionalmente, Keras también permite exportar la red convolucional entrenada para poder usarla posteriormente en otros programas. Este proceso se verá con mayor detalle en la sección dedicada a la detección del estados del instrumental y el clasificador de imágenes.

3.3 Adaptación de técnicas para la detección de eventos

En esta sección se detallará cómo se han empleado las técnicas descritas en el apartado 2.4 para dar solución a la detección de eventos de bajo nivel. A continuación se expondrán los tres tópicos en los que se agrupan las adaptaciones realizadas, serán los siguientes: *detección de marcas*, *detección de posición, movimiento y aproximación* y *detección del estado del instrumental quirúrgico*.

[6] https://github.com/erget/StereoVision
[7] https://github.com/fchollet/keras

3.3.1. Detección de marcas

La detección de marcas será uno de los procesos de mayor importancia para la detección de eventos a bajo nivel, además de ser sobre el que el resto de adaptaciones de técnicas, de una manera u otra, se soportarán. La principal motivación que existe detrás del uso de las marcas es poder identificar los diferentes elementos que se encuentran presentes en escena. Estarán especialmente focalizados en la detección de los eventos relacionados con el instrumental quirúrgico, aunque pueden existir regiones dentro del entorno que también posean marcas, dependiendo de los requerimientos del ejercicio. Por ejemplo, podremos detectar cuando un instrumento entra en escena por medio de la detección de sus marcas, o cuando desaparece de ésta por la ausencia de las mismas. Pero, a su vez, permitirá extraer mucha más información con la que completar las demás técnicas destinadas a la detección de otros eventos, como calcular la localización del instrumental dentro de la imagen o generar una región de interés sobre su extremo.

El número de marcas y su color será variable dependiendo del elemento a monitorizar. Para facilitar la configuración del sistema en este sentido, se ha añadido un archivo de configuración al respecto. Este archivo consistirá en un doscumento XML, donde se especificarán los elementos que intervienen y la configuración de colores que posee cada uno. El formato escogido para la codificación de colores será HSV. En el Apéndice B de este trabajo (Código B.1) se muestra un ejemplo de este archivo de configuración con su contenido y forma. Como puede apreciarse, estará estructurado por las marcas de colores que intervendrán. Cada una identificará a un objeto, añadiéndole además el nombre y el tipo del mismo. Se completará la información con el número de marcas que se emplearán y el tipo de color que poseen, en lenguaje natural. Por último, para cada marca se definirá un límite inferior y superior en el formato especificado (HSV). Esto se usará posteriormente a la hora de la umbralización, creando así un margen en el que se ubica el color de la marca a segmentar.

Sobre la elección del formato de color HSV, éste será de gran utilidad para poder llevar a cabo la umbralización de forma eficaz, siendo un formato de amplia difusión en el campo de la visión por ordenador. A diferencia de otros formatos, como pueda ser RGB, en HSV la información de color (croma) e intensidad (luma) están separados en diferentes canales, por lo que se podrá trabajar directamente sobre el canal de color a la hora de segmentar. Esto ayudará a controlar mejor el ruido que pueda ocasionar la iluminación, entre otros factores. Para poder usarse en OpenCV, se requiere la especificación de un margen inferior y superior del color a umbralizar, de ahí que en el archivo de configuración anterior se defina un margen de color inferior (*lower_color*) y un margen superior (*upper_color*).

En la Figura 3.2 puede verse el proceso que se ha seguido para la detección de marcas. Partimos de la imagen original, capturada a través de las cámaras dentro del entorno laparoscópico, y en formato BGR. Este formato es empleado por la librería OpenCV y consiste en una variación del RGB, donde se han cambiado el orden de los canales de color. Seguidamente, transformamos la imagen al formato HSV para prepararla de cara a la segmentación. Una vez transformada, se puede proceder a crear una máscara de color para aplicar la segmentación de las marcas. Para generar la máscara se requerirá un margen inferior y superior con el que poder segmentar la imagen, filtrando los píxeles que se encuentren dentro de estos márgenes, y descartándose el resto. Tal y como se comento con anterioridad, estos márgenes vendrán definidos por el archivo de configuración *config.xml*. De la segmentación se obtendrá una imagen binarizada donde se muestran en blanco los píxeles que cumplen el filtro, dejando el resto en negro en la imagen (fondo). Debido a que este proceso tiene una gran dependencia de la imagen original, puede existir ruido en la imagen binarizada. Para reducirlo se aplicará un postprocesado de la imagen con el objetivo de eliminar los píxeles que no forman parte de las marcas. Un procedimiento habitual consiste en aplicar un proceso de erosión, donde se eliminan los elementos más pequeños provocados por el ruido; seguidamente, se aplica un proceso de dilatación para ensanchar las marcas segmentadas; y por último, se aplica un proceso de suavizado para allanar las figuras resultantes, con vistas a que sean lo más ajustados

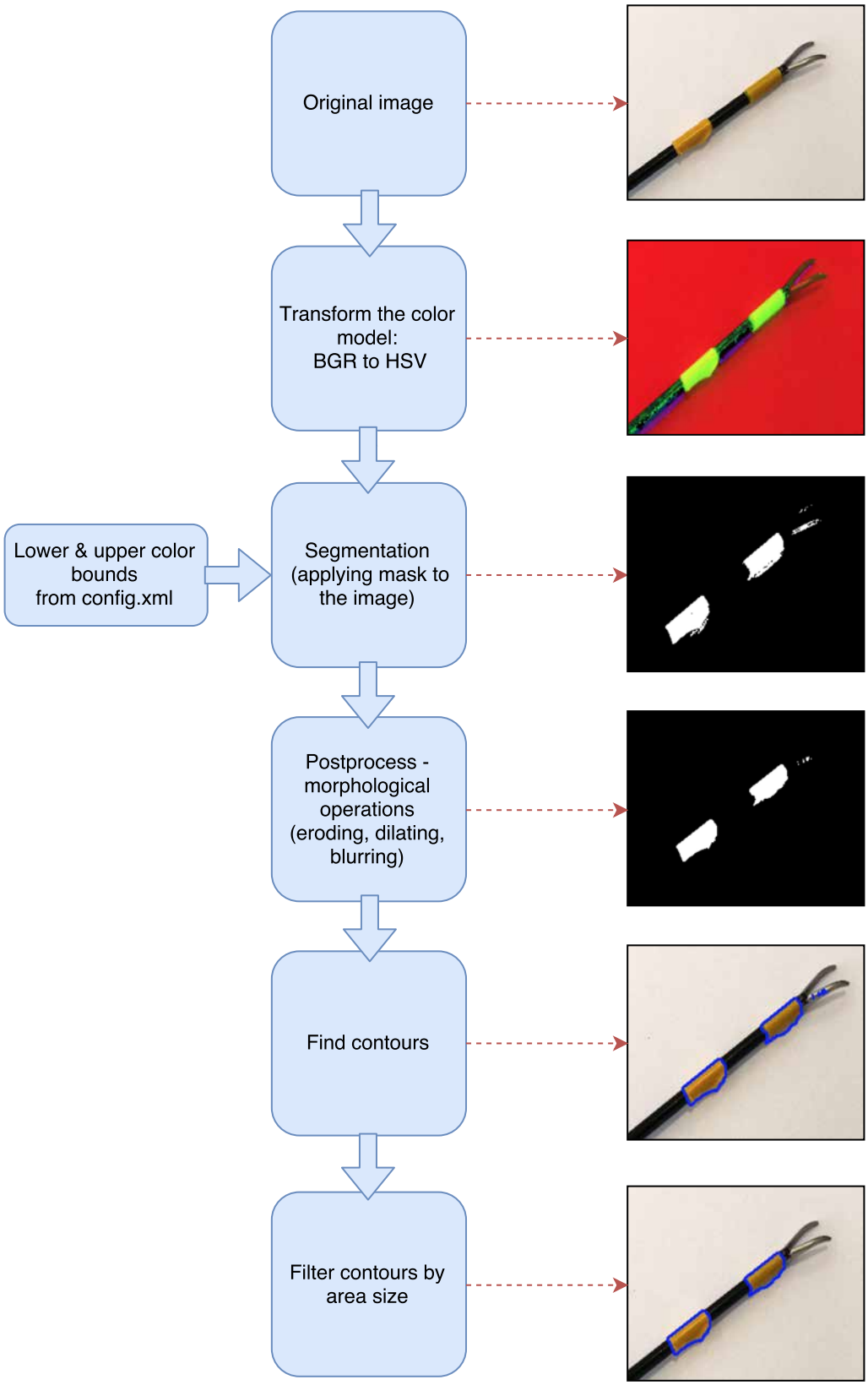


Figura 3.2: Proceso de detección de las marcas de colores. (Imagen de elaboración propia).

posibles a las marcas originales. Tras el postprocesado, se pasará a la detección de contornos. En este paso se emplearán la función de OpenCV *findContours* [8], que realizará esta labor devolviendo el listado de contornos detectados. A pesar de haber aplicado un postfiltrado previamente, todavía pueden quedar píxeles residuales que no sean de las marcas, siendo capturados por el algoritmo de detección de contornos. El hecho anterior puede provocar problemas durante la captura de eventos debido a que no forman parte de las marcas y pueden confundir al programa. Es por este motivo por el que se aplica un segundo postprocesado, donde se filtrarán los contornos por su tamaño. Para ello, se define un umbral mínimo para el área de los contornos, descartándose todos aquellos que estén por debajo de dicho umbral. Finalmente, y como resultado del proceso de detección de marcas, se obtendrá las coordenadas (en píxeles) del contorno cada una de ellas.

Sobre la integración de este proceso en el prototipo desarrollado, tendrá repercusión en dos fases de la ejecución: en una primera instancia, en el arranque del programa; y en una segunda instancia, en el bucle de reproducción de los vídeos. En la primera de ellas, se identificarán las marcas de los elementos estáticos que estén en el entorno, como contenedores de objetos o regiones que intervengan en el trascurso del ejercicio. Esto permitirá descargar de esta tarea al bucle principal, ya que son elementos que no se prevé que se desplacen o desaparezcan del entorno. Por tanto, una vez detectados y localizados, el sistema ya sabrá de su presencia. La segunda fase de detección de marcas se hará dentro de este bucle, donde se procesará cada frame buscando las marcas de los distintos elementos movibles. Un ejemplo se puede ver en el instrumental quirúrgico que intervine en el ejercicio. En cada ocasión que el instrumento este presente en la escena, se aplicará la detección de marcas para determinar sus eventos de aparición o retirada del entorno. Por otro lado, se activarán también las siguientes técnicas adaptadas para determinar la posición y el estado del extremo del instrumental.

3.3.2. Detección de posición, movimiento y aproximación

Dentro del dominio, existen diversos eventos de bajo nivel que requieren conocer la localización de los elementos dentro del entorno laparoscópico. Por ejemplo, saber si el instrumental que está realizando el ejercicio permanece quieto o ha comenzado a desplazarse de forma deliberada. O bien, si éste se está acercando a una zona o alejándose de ella. Estas acciones tendrán especial interés de cara al modelado de los ejercicios del prototipo. Si bien es cierto que no se requiere una precisión milimétrica de la posición de cada elemento, sí que es necesario conocer su localización aproximada y las variaciones de ésta. Esto se traducirá en eventos más generales como los expuestos arriba. Como punto de partida se tratará la detección de la posición de los distintos elementos dentro del entorno laparoscópico. Posteriormente, y una vez calculadas las posiciones, se podrán derivar otras acciones como el movimiento o la aproximación.

El proceso de localización ideado puede verse en la Figura 3.3. Como punto de partida se toma el conjunto de contornos que devolvía el proceso de detección de marcas. Para cada uno de ellos calculamos su centro de masas, obteniendo así un punto único que representa a la marca de color. Dado que con este punto solo tenemos una localización parcial dentro de la escena, faltará calcular la profundidad a la que se encuentra el punto dentro del escenario. Completaremos esta información con el mapa de disparidad. Gracias a la correspondencia entre la imagen original y el mapa de disparidad, podremos trasladar los contornos a este último, cayendo sobre la zona donde está el elemento que se está monitorizando. Con el contorno ya localizado en el mapa de disparidad, calcularemos la media del valor de los píxeles dentro de esta región. Al ser una imagen en escala de grises, los valores de los píxeles se encontrarán entre 0 y 255, hallándose la media dentro de este rango. Como consecuencia de la relación inversa que existe entre los valores de los píxeles y la distancia de los objetos a las cámaras, se invertirá el valor obtenido con la media anterior. Esto hará que sea más natural la interpretación de las coordenadas. Como último paso se normalizarán los

[8] http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours

valores para equiparar el peso de cada componente. Este paso se hace necesario debido al distinto rango en el que puede oscilar cada componente. Mientras que las escalas de las componentes (X,Y) del centro de masas dependerán de la resolución de la imagen, pongamos 640X480, la componente Z (profundidad) dependerá del rango de la escala de grises (255). Para que todas las componentes tengan la misma escala se aplicará la siguiente ecuación de normalización por cada una de ellas (Ecuación 3.1):

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

donde X será el valor real de la componente y los valores X_{min} y X_{max} serán el mínimo y máximo del rango. De su aplicación se obtendrá una escala única con el rango [0,1] para todas las componentes.

La detección de posiciones, según el proceso descrito, se aplicará cada vez que una marca se encuentre presente en escena. Para los objetos que posean varias marcas, como el instrumental quirúrgico, se tomará de referencia la posición de la marca más próxima al extremo de la pinza. Internamente, el sistema irá registrando la ubicación de cada marca y asignándola al objeto que la porta usando la especificación color/objeto que se le proporcionó con el archivo *config.xml*. Parlamentariamente, para los elementos que sean movibles se les asignará un lapso de tiempo para el cálculo de las nuevas posiciones. De la diferencia de posición en ese lapso de tiempo se deducirá que se ha movido. Asimismo, usa secuencia encadenada de varios cambios de posición se traducirán en que el objeto se está moviendo, pudiendo detectar así cuándo ha cambiado de reposo a movimiento y viceversa. La manera de discernir esto último será por medio de un umbral de distancia que permita diferenciar el movimiento deliberado del que no lo es. Esto significa que las variaciones de posición por debajo de ese umbral no se asociarán a acciones de movimiento.

Relativo al cálculo de distancias entre elementos, se empleará la distancia euclídea como medida de referencia (Ecuación 3.2):

$$distancia(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.2}$$

donde se usarán los datos normalizados del proceso de detección descrito arriba. Gracias a la normalización realizada, las variaciones en cualquiera de las tres coordenadas tendrán el mismo peso en la distancia, a pesar de la naturaleza distinta de una de las componentes.

Con las posiciones y distancias podremos modelar también las zonas de proximidad entre marcas. En la Figura 3.4 puede verse el sistema propuesto para determinar la aproximación entre elementos. Como ejemplo tomaremos una marca colocada en la base del simulador. Alrededor de la marca el sistema creará dos regiones simbólicas que representarán una distancia media y una distancia corta de aproximación. Al mismo tiempo, cada una tendrá definido un margen superior y un margen inferior. La condición que se introduce para que un elemento se considere dentro de una de estas regiones simbólicas es que no estuviera en esa zona y atravesara su margen inferior. Por contra, se considerará que un elemento ha salido de una zona de proximidad cuando se encuentre en ella y atraviese el margen superior definido. Un detalle a tener en cuenta es que la región de proximidad corta está contenida dentro de la región de proximidad media, por lo que al abandonar la primera, automáticamente se estará en la segunda.

Pongamos, por ejemplo, el movimiento de aproximación de un forceps a una marca en el suelo del simulador. En este escenario, el instrumento tendría que atravesar el margen inferior de la zona intermedia para que se le asignara esta distancia. Si el instrumento continúa aproximándose más a la marca, tendrá que pasar el último margen de la zona corta para que el sistema lo identifique como tal. Llegados a este punto, si el forceps comienza a alejarse de la marca, tendrá que pasar el margen superior de la zona corta para considerarlo que ha salido de ésta. De igual manera, si

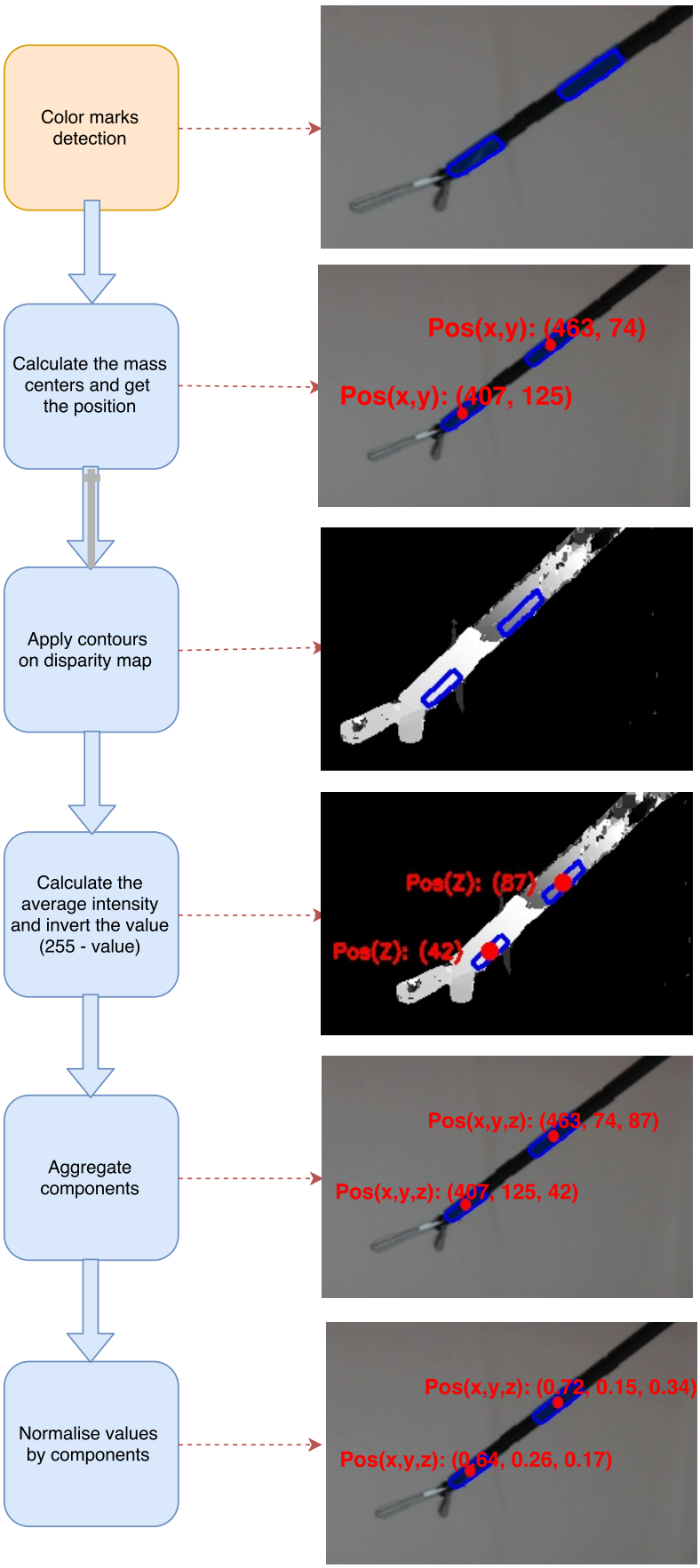


Figura 3.3: Proceso de posicionamiento 3-D de las marcas de colores. (Imagen de elaboración propia).

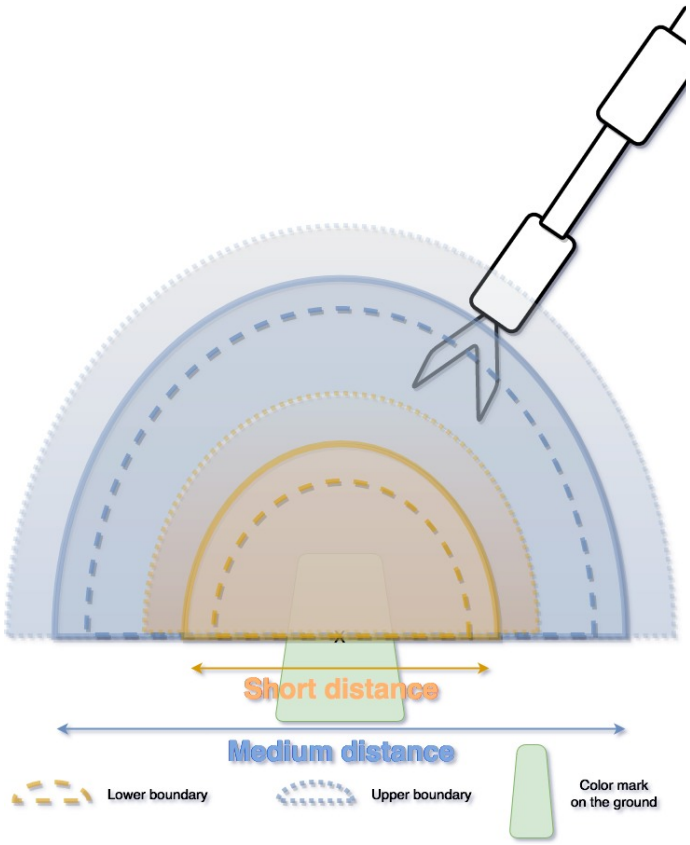


Figura 3.4: Sistema de detección de proximidad entre marcas. (Imagen de elaboración propia).

continúa alejándose y atraviesa el margen superior de la zona intermedia, se tomará como que ha abandonado dicha región y ya no estará próximo a la marca.

Los beneficios de este sistema de proximidad serán varios:

- En primer lugar, se logra la discretización de estas acciones. Al pasar de una medida continua (distancia entre dos puntos), a una medida discreta (división en regiones), simplificará la interpretación de las acciones en la salida que genera del programa. A su vez, esta adaptación permite una mejor integración con el sistema de fluents de bajo nivel que se propone el proyecto SUPERVASION. Los pasos entre zonas se traducirán en eventos de bajo nivel que permitirán modelar los ejercicios del dominio cuando estos requieran acciones de aproximación entre elementos.

- Por otra parte, permite crear un sistema más flexible a la hora de detectar los eventos aproximación. Los margenes propuestos pueden configurarse al rendimiento del proceso de detección de posiciones expuesto en este apartado. Dado el caso de que exista una elevada oscilación en las lecturas de posición entre secuencias de imágenes, pueden introducirse unos márgenes más amplios para cubrir este problema.

- Relacionado con lo anterior, el uso de dos margenes por zona consigue reducir el ruido en la salida de eventos detectados. En el caso de que existiera un único margen de proximidad por zona, cualquier movimiento cerca de este margen lanzaría eventos de aproximación y alejamiento. Con la introducción de un segundo margen por región, se proporciona una zona de confianza y se corrige este tipo de ruido.

Como se ha visto, la importancia del mapa de disparidad en el proceso de detección de posiciones es máxima. Para poder generar este recurso con la mayor fiabilidad posible se necesita

previamente una fase de calibración de las cámaras. El objetivo principal de este proceso será la rectificación de las distorsiones que poseen las cámaras. También se obtendrá como resultado las respectivas matrices de cámara de cada una de ellas. En este sentido, se han reproducido los pasos que ya se detallaron en el apartado 2.4.3.2 de este trabajo. Relativo al patrón de calibración, se ha optado por el tipo *tablero de ajedrez*, el cual se ha redimensionado para poder llevar a cabo la calibración dentro del simulador laparoscópico fabricado. En el Apéndice B de este trabajo se adjunta una captura de dicho patrón (Figura B.3a), junto con sus medidas. El patrón está formado por 10 columnas y 7 filas de casillas, siendo cada una de 1.1 cm de lado aproximadamente. Las dimensiones totales del patrón serán de 11.5 cm de ancho por 8 cm de alto. En lo relativo al número de imágenes tomadas para calibrar, éstas han sido alrededor de 60 por cada cámara, almacenándose en formato .ppm [38]. Durante esta fase se ha ido desplazando el patrón por distintas posiciones para ofrecer un conjunto de muestras más rico. Después, las imágenes generadas se han introducido en StereoVision para la detección de los puntos de calibración. En la Figura B.3b del Apéndice B puede verse una de estas muestras, donde la librería marca los puntos dentro del patrón para cada una las cámaras. Esto se repetirá para cada muestra recogida, dando paso, seguidamente, al cálculo de las matrices de cámara y los coeficientes de corrección usándose estos puntos como referencia. Por último, la librería generará una serie de archivos con los resultados de la calibración que podrán ser cargados para calibrar las cámaras cada vez que se vayan a usar.

3.3.3. Detección del estado del instrumental quirúrgico

La manipulación de objetos con el instrumental es una de las acciones más comunes dentro del entrenamiento de destrezas en la cirugía mínimamente invasiva. Cualquier ejercicio exigirá al menos la interacción con uno o varios elementos dentro del entorno laparoscópico, siendo el extremo del utensilio la zona más relevante en estas acciones. Por este motivo se hace vital conocer su estado en cada momento. Dentro del dominio existe una gran variedad de instrumentos con comportamientos muy distintos entre sí. Esto puede exigir diferentes soluciones para la correcta detección de cada uno de ellos. En el caso particular de este trabajo, nos hemos centrado en la detección de estados en instrumentos de tipo *grasper*, caracterizándose por su extremo en forma de pinza. Partiendo de este hecho, acotaremos los estados posibles a los que pueda presentar dicha pinza, es decir: *abierto*, *cerrado* o *pinzado*.

Frente a este reto, el reconocimiento de objetos y formas, dentro del campo de la visión por ordenador, es la aproximación más adecuada dado los condicionantes que posee un simulador laparoscópico. En el capítulo 2.4.4 se introdujeron algunas de las técnicas habituales para el reconocimiento de formas y objetos a través de la visión artificial. Para el problema concreto que nos enfrentamos aquí se ha optado por las bondades que ofrecen las CNN. Así pues, se ha entrenado una red convolucional para crear un clasificador de imágenes que permita identificar el estado de la pinza durante los ejercicios. Para ello será necesario poder obtener imágenes del extremo del instrumental para generar los ejemplos de entrenamiento del clasificador. Asimismo, una vez entrenado, se hará indispensable poder proporcionarle, en tiempo real, las imágenes para que las clasifique. Por todo esto, se ha ideado un sistema para generar una región de interés, más conocido como)*ROI*, sobre la pinza del instrumento cuando éste se encuentre en escena.

En la figura 3.5 se puede observar de forma simplificada el proceso creado. Usando las marcas como punto de partida, calculamos sus contornos y centros de masas. Cabe destacar que las marcas fueron colocadas estratégicamente para poder tomarlas como referencia de distancia en este proceso. De esta manera, calculamos el punto intermedio entre ambas. A continuación, proyectamos ese punto sobre el extremo del instrumento usando el centro de masas más bajo como referencia. Por último, OpenCV permite generar figuras geométricas sobre las imágenes. Con esta funcionalidad podemos crear, por ejemplo, una circunferencia sobre la pinza, usando el nuevo punto como centro y la distancia entre éste y el centro de masas más próximo como radio. Para finalizar, extraeremos el contenido de esta figura para usarlo más adelante, completando así el proceso.

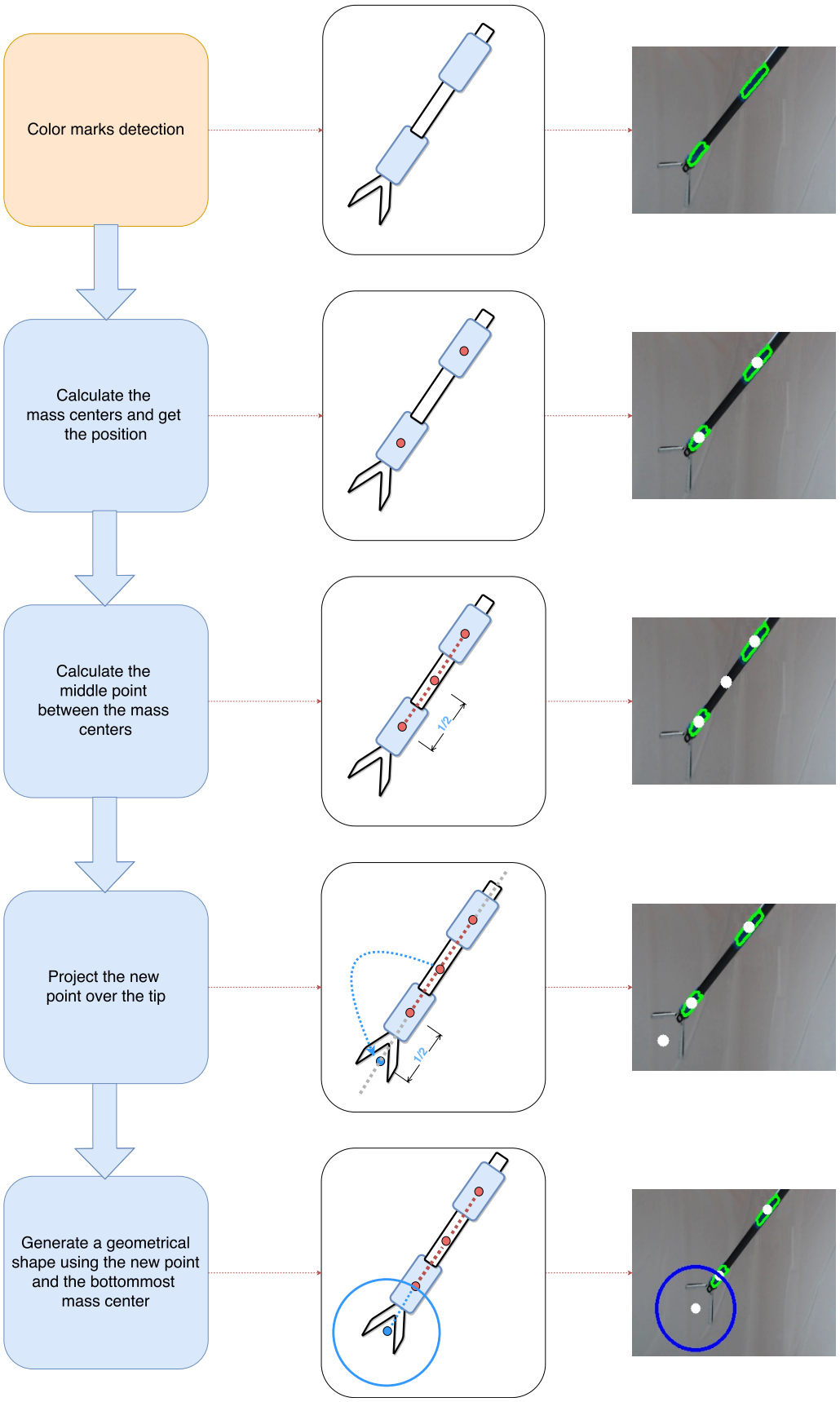


Figura 3.5: Proceso creado para la generación de una región de interés (ROI) sobre el extremo del instrumento. (Imagen de elaboración propia).

	close	object_clamped	open
close	245	0	6
object_clamped	0	251	0
open	0	0	250

Tabla 3.1: Matriz de confusión del clasificador de imágenes.

En una primera fase, el proceso descrito se usará para la generación de muestras de entrenamiento para el clasificador. Más adelante, este proceso se volverá a emplear para tener una imagen detallada de la pinza y poder usarla con el clasificador durante la ejecución de los ejercicios. Centrándonos en la primera fase, interesará entrenar el clasificador para que pueda diferenciar entre tres clases posibles:

- *Closed:* La pinza del instrumento está cerrada
- *Open:* La pinza se encuentra abierta
- *Object_clamped:* La pinza está sujetando un objeto

Para la creación de la red convolucional se ha usado la librería Keras, escogiendo un modelo secuencial y una arquitectura simple [9] para la ocasión. Esta arquitectura está diseñada para trabajar con imágenes de dimensiones pequeñas, creada originalmente para usarse sobre el dataset CIFAR-10 [10]. Además, en el Apéndice B se adjunta una reproducción gráfica de la arquitectura para su mejor visualización (Figura B.4).

Acerca del proceso de entrenamiento, se tomaron alrededor de 1200 muestras de imágenes por cada tipo de clase. El 80 % de los datos se emplearon para entrenamiento del clasificador, mientras que el 20 % restante se usó para evaluación del modelo. La Tabla 3.1 muestra los resultados obtenidos en este último paso. Como puede apreciarse el clasificador tuvo un resultado muy positivo. Adicionalmente, indicar que se usó la librería Keras como herramienta para llevar a cabo el entrenamiento de la red convolucional.

Una vez finalizada la fase de entrenamiento, se exportaron los ficheros correspondientes a la configuración de la CNN y los pesos de la red. Estos ficheros serán usados durante la fase de detección de estados del instrumental. En esta fase se volverá a usar Keras para poder utilizar el clasificador. Inicialmente se cargarán los ficheros con los ajustes de la red para, a continuación, emplear la API de Keras para lanzar consultas al clasificador. Como se mencionó con anterioridad, el proceso de creación del ROI volverá a usarse en este momento para obtener la imagen de la pinza y consultar de esta manera el estado en el que se encuentra.

En la tabla anterior se observó que la CNN entrenada cometió algunos errores de clasificación. Con el objetivo de evitar los cambios constantes de estado por errores de este tipo, se ha introducido un pequeño búfer donde se almacenan las últimas predicciones del clasificador para un instrumento en particular. La gestión de esta memoria será de tipo *cola*, calculándose la clase mayoritaria como criterio para indicar cuál es el estado en el que se encuentra la pinza. Con esta solución se evita las variaciones bruscas de estados reduciéndose el ruido en la salida del programa.

Sobre cómo se alinea la detección de estados del instrumental con los eventos a bajo nivel, nuestro principal interés se encontrará en la detección de las transiciones entre estados. Saber cuándo se ha abierto o cerrado la pinza, o si el objeto que tenía pinzado lo acaba de liberar, nos interesará de cara a monitorizar la ejecución de los pasos del ejercicio.

[9] https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py
[10] https://www.cs.toronto.edu/~kriz/cifar.html

3.4 Prototipo del componente Event Detector

Para poder automatizar el despliegue de las técnicas descritas anteriormente, se ha creado una primera aproximación al componente Event Detector dentro del marco del proyecto SUPERVASION. Esta aproximación, en forma de prototipo, implementará la adaptación de las técnicas descritas anteriormente para la detección de eventos a bajo nivel en ejercicios del dominio. Tomará como entrada vídeos con la realización del ejercicio, procesándolos frame a frame para la detección de los correspondientes eventos. Como salida se obtendrá un listado con aquellos eventos que se capturaron en formato JSON.

Paralelamente a la detección, este prototipo también permitirá comprobar el comportamiento de las técnicas en conjunto, al igual que si éstas detectan los eventos para los que se han adaptado. Para favorecer la reproducibilidad de los resultados, de igual manera que la difusión y extensibilidad del proyecto, todo el código desarrollado está público y disponible a través del repositorio creado para este propósito en GitHub [11].

El prototipo creado constará de los siguientes recursos:

- *calibration_results:* Carpeta que contiene los recursos generados por la librería StereoVision para la calibración de nuestras cámaras. Asimismo, incluye la matriz de cámara de cada una para calcular la profundidad y el mapa de disparidad. Dicha librería ofrece métodos para la carga de estos ficheros y la rectificación de las imágenes para corregir las distorsiones de las lentes.

- *models:* Esta carpeta contiene el modelo de la red convolucional con el clasificador de imágenes para la detección dos del instrumental. Los ficheros fueron generados por la librería Keras después de nuestro proceso de entrenamiento. En su interior, el fichero *multiclass_model_8.json* contiene la configuración de la red, como son las capas que la componen, mientras que el archivo *multiclass_model_8_weights.h5* posee los pesos de la red con los ajustes del entrenamiento realizado. Al igual que los datos de calibración, estos ficheros se cargarán cuando arranque el prototipo, desplegándose la red convolucional con los pesos. Tras este paso, la API de Keras permitirá usar el clasificador, pasándole la imagen que se obtiene con la aplicación del ROI sobre el extremo del instrumental.

- *videos/exercise_1:* En esta carpeta se encuentran los vídeos del ejemplo de ejercicio que se ha realizado para probar el prototipo. Es este ejercicio el que se empleará también para la evaluación de resultados que se verá posteriormente en este trabajo (apartado 4.1). Estos vídeos se pasaran como parámetros al programa y éste los procesará en tiempo real detectando los distintos eventos que suceden.

- *config.xml:* Archivo que define los elementos que intervendrán en el ejercicio y la configuración de colores HSV que tendrán las marcas de cada uno. Una descripción más detallada se puede encontrar en la sección de este capítulo 3.3.1.

- *cv_utils.py:* Librería que se ha creado para la reulizicación de las funciones de OpenCV que se han adaptado para la detección de eventos.

- *elements.py:* Módulo de Python con las clases de los elementos que intervienen en el ejercicio de ejemplo. Cada clase ofrecerá los métodos para la detección de los eventos del elemento que representa.

- *events.py:* Módulo con los eventos disponibles para el ejercicio modelado. El listado completo y la descripción de cada uno se puede encontrar en la Tabla 4.1 del capítulo 4.

[11] https://github.com/DNC87/EventDetector.git

- *main.py:* Script principal de ejecución del prototipo. Es el que recibirá los parámetros de arranque y conducirá el flujo principal del programa..
- *requirements.txt:* Listado de dependencias que requiere el prototipo para su funcionamiento. Este fichero está preparado para usarse con el gestor de paquetes *pip*[12]. Posteriormente, se describirá paso por paso cómo se puede cargar con dicho gestor.
- *output.json:* Fichero de ejemplo con la salida del prototipo, tras su ejecución con los vídeos del ejercicio. Contendrá, por tanto, la lista de eventos detectados, junto con información como el instrumento relacionado o el instante de vídeo en el que se ha sucedido. Posteriormente se describirá con más detalle su contenido en esta sección.
- *settings.py:* Archivo de configuración del prototipo donde están definidas las variables globales que se requerirán a lo largo de su ejecución.

Acerca de la instalación y puesta en funcionamiento del prototipo, éste requerirá Python 2.7 para su ejecución. Además, se recomienda la instalación de la plataforma Anaconda, al incorporar la mayoría de librerías que se requerirán. Se recomienda, asimismo, la creación y uso de un entorno virtual para evitar conflictos de versiones de paquetes que se requerirán, como por ejemplo virtualenv o el que incorpora Anaconda por defecto. Para facilitar la creación del entorno, activación y la instalación de dependencias, se ofrece el siguiente script (Código 3.1):

```
# Virtual environment creation with conda
conda create -n supervasion python=2.7 anaconda
# Virtual environment activation
# activate supervasion (for Windows)
source activate supervasion (for Linux/Mac)
# OpenCV installation with conda
conda install -c jjhelmus opencv=2.4.12
# Prototype dependencies installation
pip install -r requirements.txt
```

código 3.1: Instalación de dependencias del prototipo (Mac/Linux/Windows))

Por otro lado, para la ejecución del prototipo se requerirá el siguiente comando (Código 3.2):

```
python main.py -n videos/exercise_1/video_example_cam_1.mp4 videos/exercise_1/
    video_example_cam_2.mp4 --output_file <file_name>
```

código 3.2: Ejecución del prototipo con los vídeos de ejemplo

Una vez en funcionamiento, el prototipo irá reproduciendo el vídeo mientras despliega las técnicas sobre él, detectando las marcas y el comportamiento de las mismas, de igual manera que monitorizará las acciones del extremo del instrumental. Entretanto, irá imprimiendo en la terminal los eventos que se van detectando. Una vez finalizado el vídeo, el programa escribirá en el archivo introducido como *output_file* el listado completo de los eventos capturados.

Relativo a este fichero de salida, se ha definido una estructura basada en JSON para la información que genere y escriba el prototipo como resultado final. Tal y como se vio en la sección 2.2 de este trabajo, dentro de las fases del proceso de supervisión por observación que se proponían, el componente Event Detector debía comunicarse con otros componentes como el Event Builder. Con esto en mente, se ha optado por usar un formato de salida definido y que sea fácil de procesar por una máquina. El formato definido para este propósito puede verse a continuación (Código 3.3):

```
[{
"Instrument": "forceps",
"Frame": 121,
"Time_Frame": 15,
```

[12] https://pip.pypa.io/en/stable/quickstart/

```
"Event": "appears"
}, {
"Instrument": "forceps",
"Frame": 133,
"Time_Frame": 17,
"Event": "starts_moving"
}, {
"Instrument": "forceps",
"Frame": 162,
"Time_Frame": 17,
"Event": "crosses_in_10",
"Target": "right_pan"
},  ...]
```

código 3.3: Ejemplo de salida del prototipo en formato JSON

Sobre los campos que se muestran en la salida, el campo *Frame* y *Time-Frame* representarán el frame concreto y el instante de tiempo en el vídeo en el que se ha detectado el evento. Adicionalmente, el campo *Event* indicará el evento en cuestión que se detectó, mientras que los campos *Instrument* y *Target* contendrán la información complementaria del evento, que serán de interés en fases posteriores donde se crucen los eventos con los *fluents* de bajo nivel.

CAPÍTULO 4
Evaluación

Tras la fabricación del entorno laparoscópico, la adaptación de las técnicas y la unión de estas bajo el prototipo del componente Event Detector, en este capítulo se procederá a presentar los resultados obtenidos tras la aplicación de lo anterior sobre un ejercicio del dominio.

De entre los distintos ejercicios disponibles en el entrenamiento de destrezas, se ha escogido uno de agarre y desplazamiento de un objeto de una región de la plataforma a otra. Éste será una versión simplificada de la rutina destinada al entrenamiento de coordinación mano-ojo disponible en el Apéndice A de este trabajo. Pese a que el prototipo es configurable para la aparición de múltiples objetos e instrumentos en escena, gracias al fichero *config.xml*, se ha optado por un ejercicio básico de este tipo y, de ese modo, reducir las posibilidades de ruido, en cuanto a eventos detectados se trata. El objetivo de este experimento será, por tanto, analizar el comportamiento en conjunto de las técnicas adaptadas, comprobando si los eventos más básicos son detectados, y si estos son capturados en el orden lógico en el que han sucedido. Por ello, se ha reducido el número de elementos a un solo instrumento quirúrgico y un solo objeto objetivo.

4.1 Resultados

En Figura 4.1 se puede ver una breve secuencia de frames del ejercicio en cuestión. El escenario creado consiste en dos marcas de color verde adheridas sobre la plataforma de entrenamiento, separadas entre sí 14.5 cm, colocándose el objeto objetivo del ejercicio encima de la marca derecha. El instrumento empleado, en este caso, será el fórceps, el cual llevará adheridas dos marcas distintivas azules para aplicar así las técnicas adaptadas de detección de marcas y seguimiento. La secuencia de pasos consistirá en la introducción del fórceps por la parte derecha del simulador (siempre desde las perspectiva de las cámaras), el agarre del objeto para, seguidamente, su desplazamiento hacia la marca de la izquierda. Tras depositar el objeto en esta marca, el instrumento se retirará del simulador, concluyendo de esta manera el ejercicio.

Después de registrar el ejercicio en vídeo, produciéndose dos archivos (uno por cada cámara), se ejecutó el prototipo sobre ellos, para capturar la secuencia completa de eventos observables que se habían producido. La Tabla 4.2 recoge el resultado que se obtuvo. También está disponible en el Apéndice B (Código B.2) la salida que generó el prototipo para este caso. La tabla mencionada se completa con la detección manual de la secuencia de eventos observados directamente en los vídeos, a modo de *ground truth*. De esta manera, se podrá contrastar el resultado generado por el prototipo con el detectado por un humano. Concretamente, las filas (*Events*) de la tabla representarán los eventos de bajo nivel que se han modelado para este ejercicio, pudiéndose encontrar su descripción detallada la tabla 4.1. Las columnas (*States*) identificarán el orden de estados en los que ha trascurrido el ejercicio. Por consiguiente, las celdas mostrarán las transiciones de estados entre eventos que ha capturado el prototipo (representadas por el icono rojo) y las que realmente se han observado, reflejados como *ground truth* (representadas por el icono azul).

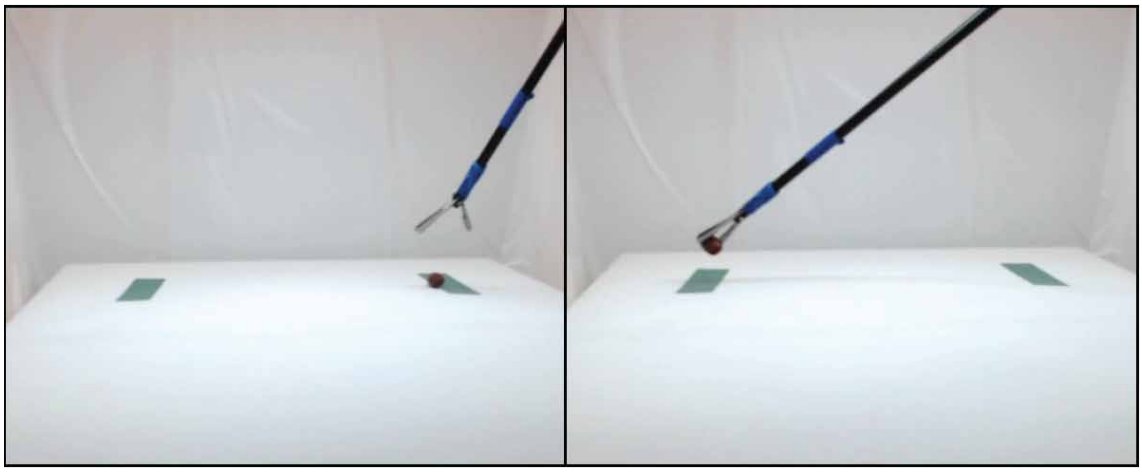


Figura 4.1: Secuencia de frames extraídos del vídeo con el ejercicio de ejemplo.

Events	Description
closes(<instrument>)	The instrument is closed
drops(<instrument>, <object>)	The object caught by the instrument is released
starts_moving(<instrument>)	The instrument starts moving
stops(<instrument>)	The instrument that was moving stops
appears(<instrument>)	The instrument enters the camera framing
disappear(<instrument>)	The instrument leaves the camera framing
opens(<instrument>)	The instrument is opened
picks(<instrument>, <object>)	The instrument clamps the object
crosses_ in_5(<instrument>, <target>)	The instrument enters into the 5 units zone from the target
crosses_ in_10(<instrument>, <target>)	The instrument enters the 10 units zone from the target
crosses_out_5(<instrument>, <target>)	The instrument leaves the 5 units zone from the target
crosses_out_10(<instrument>, <target>)	The instrument leaves the 10 units zone from the target

Tabla 4.1: Listado de eventos a bajo nivel que intervienen en el ejercicio de ejemplo.

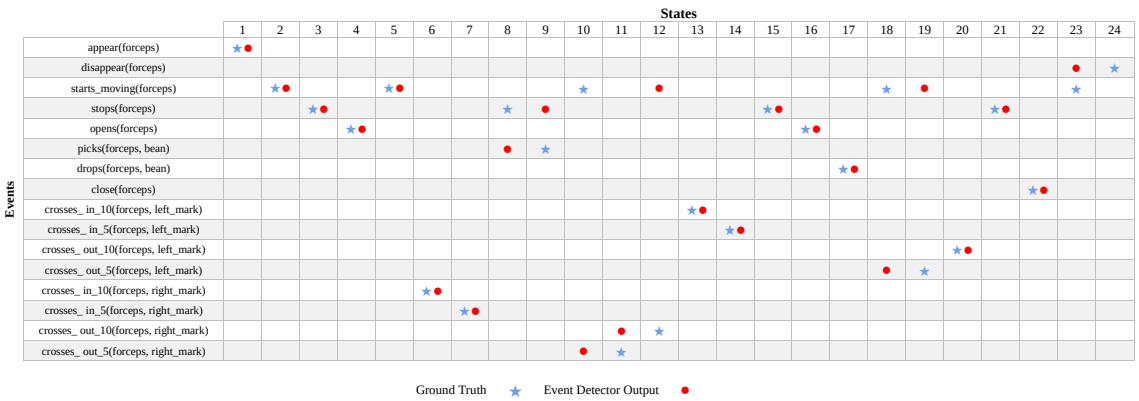


Tabla 4.2: Comparativa entre el *ground-truth* y la salida del prototipo. (Imagen de elaboración propia).

Sobre los resultados podemos apreciar que entre los estados 1 y 7, el prototipo capturó de forma apropiada los eventos que sucedieron. Además, se detectaron en el orden correcto que se ven en el vídeo, siguiendo el patrón del *ground truth*. Esto significa que capturó la aparición del fórceps en escena, su apertura (previa detención) y posterior desplazamiento, aproximándose a la marca de la derecha. Luego, aparece un desajuste en entre los estados 8 y 12, ambos incluidos. Aunque se aprecia que los eventos fueron capturados por el prototipo, el orden en el que lo hizo no fue el

adecuado. En algunos de ellos se ve un patrón de permuta en el orden (*swap pattern*), como en los estados 8–9 (eventos *stops* y *closes*) y 10–12 (evento *starts_moving*); mientras que en otros se ve un patrón de demora en el tiempo (*delay-in-time pattern*), como sucede entre el estado 10 y el 12 (eventos *crosses_out_5* y *crosses_out_10*). Esta desalineación sucede cuando el instrumento se detiene para coger el objeto y comienza a trasladarlo hacia la otra marca, alejándose de la primera. Tras este desajuste de eventos, el prototipo vuelve a seguir correctamente el *ground truth* entre los estados 13 y 17. Esta vez la aproximación a la marca se detecta de forma acertada, capturándose en el orden correcto la detención, apertura y liberación del objeto. Después, se distingue otro patrón de permuta entre los eventos 18–19 (eventos *starts_moving* y *crosses_out_5*), cuando el fórceps se aleja de la marca izquierda. A continuación, ambos patrones se vuelven alienar entre los eventos 20 y 22. Esto comprende el momento en que el instrumento termina de alejarse de la marca, deteniéndose para cerrarse. En la parte final del ejercicio se vuelve a ver el patrón de demora en el tiempo, en relación a los estados 23–24 (eventos *start_moving* y *disappear*). Esto se produce en el momento de la retirada del instrumental del simulador, capturándose correctamente la desaparición del fórceps de la escena, pero no la reanudación del movimiento tras haber estado detenido.

4.2 Discusión

A la luz del resultado obtenido, éste puede considerarse bueno. La salida del prototipo se ajusta correctamente durante varios estados a lo que se observa en los vídeos. En lo relativo a la detección de eventos, todos salvo uno (el último evento *starts_moving*) fueron detectados por el sistema. Esto hace pensar que la adaptación de las técnicas empleadas han funcionado de forma bastante eficaz. Los eventos relacionados con el estado del instrumental (abierto, cerrado u objeto pinzado), han sido detectados perfectamente. Lo mismo sucede con los eventos de entrada y salida de la escena del fórceps, que fueron capturados bien igualmente. En lo concerniente a la detección del movimiento y posición de elementos, la aproximación efectuada puede arrojar más dudas. Si bien es cierto que los eventos de este tipo se han detectado en su gran mayoría bien , el hecho de que el movimiento de salida no se haya detectado requiere un análisis pormenorizado. En el vídeo se aprecia que la retirada del instrumento es sumamente rápida, además de partir de una zona bastante próxima al final del plano que registran las cámaras. Es posible que esto haya afectado al cálculo de distancias entre la posición de partida y la que iba obteniéndose en su desplazamiento, estando por debajo del umbral de detección que disparara el evento. Estas circunstancias deben de tenerse en cuenta para futuras mejoras del sistema de detección de movimiento.

Por otro lado, y en lo concerniente al orden de detección de eventos, los resultados obtenidos han sido correctos en ciertas fases pero se han encontrado más errores. Estos fallos han venido ocasionados, principalmente, por la demora en el tiempo de detección de ciertos eventos, con el consiguiente desplazamiento de los siguientes estados. Un ejemplo lo podemos encontrar con los eventos *stops* y *starts_moving*. En muchos casos se han detectado bien pero han estado retrasados en el tiempo. El motivo principal se podría encontrar en los umbrales que se introdujeron para determinar cuando el instrumental permanece parado en una posición o ha comenzado a desplazarse. Y es que el modelado de este tipo de eventos es sumamente complejo, debido a que el sistema debe de ser capaz de distinguir si el movimiento que ha comenzado es deliberado o no. Factores como el pulso humano, por ejemplo, deben de ser contemplados en el modelado, con el objetivo de evitar que la detección de eventos de movimiento y parada sea constante, con el consecuente ruido en la salida del programa. A su vez, y como fruto de que el calculo de la disparidad entre las imágenes de la escena es un proceso complejo, el mapa que se crea (mapa de disparidad) está sujeto a constantes fluctuaciones de forma y valores de los píxeles que lo componen. Una fluctuación de este tipo puede ocasionar que el cálculo de la posición de las marcas varíe también, pudiendo confundir al detector de eventos. Aunque es verdad que estos umbrales palían las consecuencias de los factores comentados, tienen la contraprestación de que demora la detección de los eventos en el tiempo. Aunque a simple vista, en el vídeo, se aprecia que el movimiento que ha comenzado

es deliberado, hasta que no se supere el umbral que se le establecido al sistema, éste no lanzará el evento. Esto provoca que otros eventos puedan suceder durante este transcurso, detectándose antes que los que umbral. El efecto no deseado que se obtiene, es que la secuencia generada por el prototipo no es lógica en el mundo real. Un caso de este tipo se puede ver entre los estados 10 y 12 de la tabla anterior. El sistema ha detectado que el instrumento se está alejando de la marca de la derecha (eventos *crosses_out_5* y *crosses_out_10*), pero el evento relacionado con el comienzo del movimiento (evento *start_moving*) aparece posteriormente tras estos dos. En este mismo sentido, sucede de igual manera con el patrón de permuta entre los estados 18–19, ya que el fórceps no ha podido abandonar la zona de la marca sin haber comenzado a moverse. Algo similar acontece en el final del ejercicio (estados 23–24), dado que un instrumento detenido (como estaba el fórceps), solo podrá desaparecer de la escena si, y solo si, comienza a desplazarse.

Como solución al problema de la incoherencia en la secuencia generada, se han barajado varias propuestas para solventarlo. Una de ellas es la introducción de restricciones entre eventos, tal y como sucede en la realidad. Al igual que en el mundo que nos rodea, los objetos no pueden desaparecer instantáneamente o alejarse sin desplazarse; estás reglas de dependencia tendrían que poder introducirse al sistema. Lo anterior englobaría también las relaciones de eventos que en el ejemplo han funcionado correctamente. En el caso de que se detectara que un instrumento ha pinzado el objeto, cuando previamente se había cerrado, no es posible en el mundo físico. Tendría que estudiarse, por tanto, las relaciones existentes entre eventos, su formalización en reglas y la integración óptima en el flujo de funcionamiento del prototipo. También se contempla la posibilidad de que estas restricciones pudieran ser inferidas por el propio sistema, a través del análisis de las narrativas de los ejercicios. En dichas descripciones a alto nivel, en muchas ocasiones, se puede deducir las relaciones causales entre acciones y su orden de consecuencia. Destacar algunos estudios en este campo como [94] [10] [48]. Por otro lado, la introducción de sensores también podría ayudar para mejorar la monitorización de instrumental, sustituyendo el sistema de umbrales o reduciéndolos para así incrementar su precisión. Esta idea acercaría nuestra solución a la de los sistemas de medición que incorporan los simuladores laparoscópicos comerciales actualmente. Tecnologías como la infrarroja permitiría calcular distancias de profundidad de forma más precisa que la visión estereoscópica y la generación de mapas de disparidad. De igual modo, que ya existen líneas de investigación en torno a la medición de presión en instrumental quirúrgico por medio de sensores integrados en ellos [67] [66].

Paralelamente a los datos extraídos del ejercicio, y debido a los pocos elementos que han intervenido en él, pueden haber otros problemas que no hayan emergido pero que se deben de tener en consideración. Un problema recurrente en el campo de la visión por ordenador es el de la oclusión. Este fenómeno sucede cuando un objeto se oculta, parcial o totalmente, detrás de otro desde la perspectiva de las cámaras. Este problema tiene especial impacto en técnicas como el *tracking*, cuando el objeto que se está siguiendo queda tapado por otro. Cuando esto sucede, el sistema puede perder la referencia del objeto, lo que puede desembocar en errores o malos funcionamientos. En nuestro caso, la oclusión de las marcas podrían afectar al proceso de posicionamiento ideado, al igual que a la generación del ROI sobre el extremo del fórceps para la detección de su estado. Sumado a lo anterior, la ocultación de la pinza tras algún objeto, podría afectar también al clasificador de imágenes que determina el estado de la misma. En lo relativo a la oclusión de marcas en el instrumental, la oclusión total de las dos marcas parece poco probable, debido a que los elementos que se manipulan están posados en base del simulador, quedando las marcas muy por encima en el plano. Por contra, sí que se prevé la oclusión parcial de la marca más próxima al extremo, como consecuencia de la cercanía a la zona de manipulación de objetos. Acerca de las marcas colocadas en la plataforma, es más previsible que queden ocluidas por los objetos o por el extremo del instrumento cuando se posa sobre ellas. Existen actualmente diversas alternativas para tratar este tipo de problemas. Por ejemplo, el uso de *flocks* [39], podrían corregir este problema en oclusiones parciales. Este proceso consiste en la creación de una serie de características que se reparten por el objeto monitorizado, las cuales se van adaptando según las partes visibles que se

registran. Su distribución se va ajustando a lo largo del *tracking*, pero siempre tratando de mantener la relación de aspecto del objeto. Otras opciones podrían pasar por un sistema multi-cámara dentro del entorno laparoscópico. Al encontrarse las dos cámaras en la misma ubicación y perspectiva, es más fácil que el objeto ocluido esté tapado para ambas. Aunque en ciertos casos, gracias a la propia visión estereoscópica, se podría completar la información faltante de una cámara con la otra, al encontrarse ligeramente separadas; se estima que esta solución no cubra otros escenarios con más objetos que entorpezcan la visión. Se buscaría con el empleo de otras cámaras cubrir esas zonas de visión muerta y tener una observación total de todo el entorno. Por contra, el sistema aumentaría en complejidad, al tener que integrar estas nuevas entradas de vídeo; de igual manera que se debe de estudiar el mejor aprovechamiento de estos nuevos recursos en relación a la detección de eventos observables.

CAPÍTULO 5
Conclusiones

Este trabajo ha estado enmarcado dentro del ámbito de la supervisión automática, donde el aprendizaje automático y la nuevas técnicas de visión artificial han hecho aflorar innovadoras propuestas de sistemas inteligentes que puedan asistir a humanos en el día a día. Entre ellas se encuentra el proyecto SUPERVASION, y dentro de él, el presente documento ha recogido la implementación, a modo de prototipo, del componente *Event Detector*, una de las piezas clave que se requieren para su sistema de supervisión por observación. En este capítulo, por tanto, recogeremos a modo de resumen todos los tópicos abordados a lo largo del documento, para finalizar con las conclusiones extraídas por cada uno de los objetivos planteados y la exposición final de propuestas de mejora a modo de trabajo futuro.

Inicialmente, en este trabajo se ha presentado el contexto en el que se ha desenvuelto el desarrollo realizado. Primero se ha expuesto la motivación del proyecto SUPERVASION junto a su propuesta de sistema de supervisión por observación. Seguidamente, se han detallado las dos fases que lo componen, siendo una inicial de adquisición de conocimiento por parte del sistema, donde se usarán las descripciones a alto nivel y los ejemplos en vídeo, y una segunda en la que el sistema supervisará *on-line* sobre vídeo en tiempo real. También se ha profundizado en el proceso de aprendizaje propuesto (Inducción por Abducción), donde se ha descrito la relación que existe entre los eventos de bajo nivel y los distintos tipos de fluents (bajo y alto nivel). Por último, se ha descrito el módulo de software requerido para la captura y procesamiento de eventos a bajo nivel (Event Builder). En su interior se halla el programa que ha sido el objetivo de este trabajo (Event Detector). Adicionalmente, se ha detallado brevemente el dominio en el que se manejó el prototipo, siendo éste el de entrenamientos de destrezas en el ámbito de la cirugía mínimamente invasiva.

Con el contexto ya definido, se han descrito las técnicas de visión por ordenador y aprendizaje automático empleadas. Estas han consistido en las siguientes: segmentación, visión estereoscópica por computador y reconocimiento de formas y objetos. La primera de ellas consistía en la extracción de regiones de una imagen cuando éstas cumplían una determinada propiedad, como por ejemplo el color o el contorno. Por otro lado, la visión estereoscópica artificial permitía generar recursos que representaran la profundidad de la escena grabada. Llegados a este punto, nos centramos en describir el mapa de disparidad, detallando cómo mediante el cálculo de disparidades entre dos imágenes, se podía crear una imagen nueva en la cual el valor de cada píxel representara la profundidad de un punto en la escena. Tras esto desglosamos las diferentes técnicas en el área del reconocimiento de formas y objetos para la detección de elementos dentro de una imagen. En esta parte, se destacaron las ConvNets, redes neuronales especializadas en el procesamiento de imágenes, permitiendo crear así clasificadores que identifiquen formas u objetos.

Con el marco de recursos y técnicas disponibles aclarado, se han presentado las adaptaciones de estas técnicas para el modelado de eventos. En primer lugar se ha mostrado el proceso creado para la detección de las marcas de colores que identifican los elementos que intervienen durante las grabaciones. Por medio de la segmentación del color se ha podido detectar qué elementos

estaban presentes en la escena, de igual modo que se ha podido obtener la localización de sus marcas por medio de la detección de contornos. Posteriormente, se ha introducido el proceso para la localización de elementos en el espacio tridimensional del simulador. Para ello nos hemos valido de los contornos obtenidos por el proceso de detección de marcas y del mapa de disparidad generado con la visión estereoscópica artificial. Con estos recursos hemos logrado obtener la posición en tres dimensiones de las distintas marcas que estaban en escena. A continuación, se ha mostrado cómo se han usado estas posiciones para derivar eventos relacionados con el movimiento y la proximidad entre marcas dentro del entorno laparoscópico. Destacar aquí el sistema de márgenes ideado para discretizar las acciones de acercamiento o alejamiento de los utensilios de cirugía con respecto a otras marcas. Adjunto a esto, se ha expuesto también el proceso de calibración llevado a cabo para corregir las distorsiones de las lentes y generar las matrices de las cámaras. Por último, se ha presentado el sistema creado para generar una región de interés (ROI) sobre el extremo de un instrumento. De esta forma hemos obtenido una monitorización constante de la pinza. Con esta región de interés se han generado muestras de ejemplo para entrenar una red convolucional, creándose así un clasificador de imágenes con el que reconocer el estado del instrumental. A su vez, el sistema de ROI ha servido para poder usar el clasificador en tiempo real.

En la parte de Resultados se ha probado el prototipo creado durante la simulación de un ejercicio del dominio. Para comprobar el rendimiento del programa creado se ha evaluado la salida que generó frente a un etiquetado a mano de los eventos observados (*ground-truth*). A pesar de que el ejercicio realizado era sencillo, se obtuvieron una serie de resultados interesantes. La detección de eventos fue bastante positiva, capturándose todos salvo uno. Lo que lleva a pensar que las técnicas adaptadas de reconocimiento de eventos han actuado eficazmente. Si bien es cierto, que el único evento no capturado era relativo a la detección de movimiento. Por lo que se debe de tomar nota para mejorar en este sentido. Por contra, en lo concerniente a la secuencia capturada de eventos, ésta presentaba varias desalineaciones con respecto al *ground-truth*. Se apreció un retraso en la detección de ciertos eventos que alteraba la secuencia lógica de estos. El principal motivo que se argumentó como causa fueron los umbrales introducidos para reducir el ruido en la generación de eventos. Aunque umbrales cumplieron su objetivo satisfactoriamente, ya que los eventos detectados son prácticamente los mismos que los del *ground-truth*, también provocaron un retraso en la detección de algunos eventos, especialmente los de movimiento.

5.1 Conclusiones finales

Sobre los objetivos de este trabajo, se ha logrado la creación del prototipo que se pretendía para el componente de software Event Detector. El programa creado es configurable, en muchos sentidos, pudiéndose adaptar a diferentes escenarios de uso. Gracias al archivo de configuración *config.xml* se pueden introducir múltiples juegos de marcas al sistema. Esto permite el uso de numerosos instrumentos u objetos a la vez en el entorno. Además, al ser parametrizable puede usarse con cualquier vídeo de ejercicios, incluso pudiendo automatizarse el proceso de detección en lotes.

En los referente a las técnicas escogidas de visión por ordenador y aprendizaje automático, éstas se han podido adaptar para la detección de eventos, como en el caso de ejemplo del capítulo de Resultados. De esta manera, se han podido cubrir eventos de muy diversa índole (consultar Tabla 4.1). Tal y como se comento anteriormente, y a la luz de los resultados de la sección 4.1, las técnicas respondieron bien en cuanto a la detección pero fallaron en cuanto a la velocidad de respuesta. Por otro lado, parece que el prototipo pudo desplegar todas ellas bien y pudieron trabajar de forma coordinada. También destacar que al tratarse de un trabajo todavía en desarrollo con una única prueba, los datos obtenidos no dejan de se meramente orientativos, no pudiéndose sacar conclusiones contrastadas.

Por otra parte, se ha creado con éxito un entorno laparoscópico en el que se pudiera reproducir ejercicios del dominio. El simulador fabricado reúne las condiciones básicas de estos dispositivos como la iluminación artificial, el acceso de múltiples instrumentos, la opacidad de las paredes y el sistema de visionado interno. A su vez, este entorno ha permitido el despliegue de las técnicas de visión por ordenador que se requerían. Se han podido ubicar cómodamente las dos cámaras necesarias para la visión estereoscópica artificial, mientras que el color escogido para el forrado no ha creado ningún problema en técnicas como la segmentación o el clasificador de imágenes. Finalmente, ha permitido realizar una simulación de ejercicio de laparoscopia con él, pudiendo así poder probar el prototipo con una rutina completa.

Se ha facilitado también la integración del prototipo creado con otros procesos o futuros componentes de software. El formato de salida escogido (JSON) es fácilmente procesable por un ordenador; por lo tanto, el listado de eventos que se genera puede ser usado por otros programas, como por ejemplo el componente Event Sequencer dentro del módulo Event Builder.

Acerca de la divulgación del trabajo realizado, se han tomado varias iniciativas de cara la difusión del código generado y los resultados obtenidos. En primer lugar, se han elaborado dos publicaciones, ambas aceptadas en el *4th international Workshop on Sensor-based Activity Recognition and Interaction* de Rostock (Alemania):

- *Low-level Event Detection System for Minimally-Invasive Surgery Training:* Publicación que recoge de forma resumida el desarrollo del prototipo presentado en este documento.
- *Knowledge Extraction from Task Narratives:* trata sobre la generación de modelos de situación y extracción de reglas directamente de las descripciones en lenguaje natural de una serie de ejercicios de laparoscopia. Los primeros resultados obtenidos han sido positivos de cara a modelar el dominio a alto nivel. lográndose extraer numerosas relaciones entre acciones y objetos simplemente del análisis de las descripciones.

Por otro lado, la segunda publicación En último lugar, el programa desarrollado, junto con los resultados obtenidos, se han subido a un repositorio público de GitHub para su libre acceso.

5.2 Trabajo futuro

En lo relativo al trabajo futuro, a lo largo de esta memoria se han abierto varias puertas para próximas mejoras que se recogen a continuación:

- Se necesitan más casos de ejemplos con los que poder contrastar los resultados obtenidos hasta el momento. A parte de diversas muestras del ejercicio ya descrito, se requerirán otras pruebas con más factores. Dado que el prototipo está preparado para el uso de múltiples instrumentos y marcas, se puede pensar en escenarios más elaborados o complejos. Todo esto con el objetivo de aflorar las carencias y debilidades de las técnicas empleadas para así mejorarlas y el sistema más robusto y fiable.
- Se requiere estudiar la posibilidad de introducir reglas o restricciones entre eventos para dotar de coherencia a las secuencias generadas de salida. Debido a los retrasos observados en la detección de ciertos eventos, varias de las secuencias estaban retrasadas en el tiempo, dándose casos incoherentes en el mundo real. Este hecho puede poner en riesgo las siguientes etapas que forman la fase de adquisición de conocimiento del sistema supervisor. Una de las soluciones que se han planteado consiste en la creación de dependencias entre eventos, tal y como ocurre en la realidad. En este sentido, se debe estudiar la mejor forma de definir estas dependencias, ya sea por medio de una especificación directa (archivo de configuración) o bien que el sistema sea capaz de inferirlas de los recursos que ya dispone (descripciones a alto nivel).

- Acerca de los umbrales, se ha visto reflejado en los resultados que su comportamiento ha generado retrasos en la detección de ciertos eventos. Así pues, surge la necesidad de una mejor aproximación para el problema de la reducción de ruido. Se debe de estudiar otras soluciones que permitan controlar el ruido pero que no penalicen en el tiempo la detección de los eventos. Asimismo, con el objetivo de mejorar la fiabilidad en la detección de eventos, se plantea la posibilidad de introducir sensores que mejoren la precisión y velocidad de respuesta en la detección. Acelerómetros o sensores de presión pueden resultar de ayuda en este caso, sustituyendo así al sistema de umbrales actual.

- Relacionado con las técnicas de visión por ordenador empleadas, se expuso con anterioridad el caso de la oclusión como posible problema en escenarios más complejos. Por ello debe de ser contemplado por el sistema, en aras de la robustez de comportamiento. Se requerirá, por tanto, la investigación de posibles mejoras en el sistema de visión artificial que puedan cubrir este problema, como los sistemas multicámara o algoritmos resistentes a la oclusiones parciales (flocks).

Bibliografía

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and others. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

[2] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, and others. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint* (2016).

[3] Pilar Álvarez and E L País. 2014. La enfermera infectada: "El fallo pudo ser al quitarme el traje". https://politica.elpais.com/politica/2014/10/08/actualidad/1412769972_972884.html. (9 Oct. 2014). Accessed: 2017-7-31.

[4] Alexander Artikis, Marek Sergot, and Georgios Paliouras. 2015. An Event Calculus for Event Recognition. *IEEE Trans. Knowl. Data Eng.* 27, 4 (2015), 895–908.

[5] Anthony P Badali, Yahui Zhang, Peter Carr, Paul J Thomas, and Richard I Hornsey. 2005. Scale factor in digital cameras. In *Photonic Applications in Biosensing and Imaging*.

[6] Simon Bar-Meir. 2006. Simbionix simulator. *Gastrointestinal Endoscopy Clinics* 16, 3 (2006), 471–478.

[7] David Bellot, Anne Boyer, and François Charpillet. 2002. A new definition of qualified gain in a data fusion process: application to telemedicine. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, Vol. 2. IEEE, 865–872.

[8] John D Birkmeyer, Jonathan F Finks, Amanda O'reilly, Mary Oerline, Arthur M Carlin, Andre R Nunn, Justin Dimick, Mousumi Banerjee, and Nancy JO Birkmeyer. 2013. Surgical skill and complication rates after bariatric surgery. *New England Journal of Medicine* 369, 15 (2013), 1434–1442.

[9] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[10] S. R. K. Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning High-level Planning from Text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1 (ACL '12)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 126–135. http://dl.acm.org/citation.cfm?id=2390524.2390543

[11] Roberto Brunelli. 2009. *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley Publishing.

[12] C. Monserrat, J. Hernández-Orallo, J.F. Dolz, M.J. Rupérez, P. Flach. 2016. Knowledge Acquisition by Abduction for Skills Monitoring: Application to Surgical Skills. In *International Conference on Inductive Logic Programming*.

[13] John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), 679–698.

[14] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. 2002. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 8 (2002), 1026–1038.

[15] Stephen Cass. 2017. The 2017 Top Programming Languages. (Jul 2017). https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

[16] José Carlos Castillo, Antonio Fernández-Caballero, and María T López. 2017a. A Review on Intelligent Monitoring and Activity Interpretation. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial* 20, 59 (2017).

[17] José Carlos Castillo, Antonio Fernández-Caballero, and María Teresa López. 2017b. A Review on Intelligent Monitoring and Activity Interpretation. *Inteligencia Artificial* 20, 59 (2017), 53.

[18] Jose M Chaquet, Enrique J Carmona, and Antonio Fernández-Caballero. 2013. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding* 117, 6 (2013), 633–659.

[19] Zhengying Chen, Yonghong Tian, Wei Zeng, and Tiejun Huang. 2015. Detecting abnormal behaviors in surveillance videos based on fuzzy clustering and multiple Auto-Encoders. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*. IEEE, 1–6.

[20] Soumith Chintala. 2014. Convnet Benchmarks. (2014). https://github.com/soumith/convnet-benchmarks

[21] Magdalena K Chmarra, Niels H Bakker, Cornelis A Grimbergen, and Jenny Dankelman. 2006. TrEndo, a device for tracking minimally invasive surgical instruments in training setups. *Sensors and Actuators A: Physical* 126, 2 (2006), 328–334.

[22] Enrique Coronado. 2014. Patrones de calibración para OpenCV. https://mecatronicauaslp.wordpress.com/2014/03/08/patrones-de-calibracion-para-opencv/. (8 March 2014). Accessed: 2017-7-24.

[23] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 886–893.

[24] MATLAB Docs. 2017. What Is Camera Calibration? - MATLAB & Simulink - MathWorks España. (2017). https://es.mathworks.com/help/vision/ug/camera-calibration.html

[25] Conda Documentation. n. d. Managing environments — Conda documentation. (n. d.). https://conda.io/docs/using/envs.html

[26] Daniel Lee (Erget). 2014. Building and calibrating a stereo camera with OpenCV (<50€). (Jul 2014). https://erget.wordpress.com/2014/02/01/calibrating-a-stereo-camera-with-opencv/

[27] Loren Fiore, Duc Fehr, Robot Bodor, Andrew Drenner, Guruprasad Somasundaram, and Nikolaos Papanikolopoulos. 2008. Multi-camera human activity monitoring. *Journal of Intelligent and Robotic Systems* 52, 1 (2008), 5–43.

[28] Peter Flach. 2012. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

[29] Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*. Springer, 23–37.

[30] BE Furby and BD Roney. 1984. Autonomous surveillance in the visual spectral region. In *In Materials Research Labs. Extracts from Symp.: Countersurveillance 1983 p 48-68 (SEE N84-34779 24-43)*.

[31] Yixin Gao, S Swaroop Vedula, Carol E Reiley, Narges Ahmidi, Balakrishnan Varadarajan, Henry C Lin, Lingling Tao, Luca Zappella, Benjamın Béjar, David D Yuh, and others. JHU-ISI gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling.

[32] Ovidiu Ghita, John Mallon, and Paul F Whelan. 2001. Epipolar line extraction using feature matching. (2001).

[33] Rafael C González. 2010. Digital image processing. (2010).

[34] Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H Muggleton, Ute Schmid, and Benjamin Zorn. 2015. Inductive programming meets the real world. *Commun. ACM* 58, 11 (2015), 90–99.

[35] Gary S Guthart and J Kenneth Salisbury. 2000. The Intuitive/sup TM/telesurgery system: overview and application. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, Vol. 1. IEEE, 618–621.

[36] Ima Hajshirmohammadi and Shahram Payandeh. 2007. Fuzzy set theory for performance evaluation in a surgical simulator. *Presence: Teleoperators and Virtual Environments* 16, 6 (2007), 603–622.

[37] Ismail Haritaoglu, David Harwood, and Larry S. Davis. 2000. W/sup 4: real-time surveillance of people and their activities. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 809–830.

[38] Bryan Henderson. 2016. PPM Format Specification. (2016). http://netpbm.sourceforge.net/doc/ppm.html

[39] Jesse Hoey, A von Bertoldi, P Poupart, and A Mihailidis. 2006. Tracking using Flocks of Features, with Application to Assisted Handwashing.. In *BMVC*. 367–376.

[40] S Barry Issenberg, William C McGaghie, Emil R Petrusa, David Lee Gordon, and Ross J Scalese. 2005. Features and uses of high-fidelity medical simulations that lead to effective learning: a BEME systematic review. *Med. Teach.* 27, 1 (Jan. 2005), 10–28.

[41] Huang Jeff, Shahram Payandeh, Peter Doris, and Ima Hajshirmohammadi. 2005. Fuzzy Classification: Towards Evaluating Performance on a Surgical Simulator. *Medicine Meets Virtual Reality 13: The Magical Next Becomes the Medical Now* 111 (2005), 194.

[42] Andrej Karpathy. 2014. Convolutional Neural Networks (CNNs / ConvNets). (2014). http://cs231n.github.io/convolutional-networks/

[43] Robert Kowalski and Marek Sergot. 1989. A Logic-Based Calculus of Events. In *Topics in Information Systems*. 23–55.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[45] Manish Kushwaha, Songhwai Oh, Isaac Amundson, Xenofon Koutsoukos, and Akos Ledeczi. 2008. Target tracking in urban environments using audio-video signal processing in heterogeneous wireless sensor networks. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*. IEEE, 1606–1610.

[46] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[47] Julian Leong, Marios Nicolaou, Louis Atallah, George Mylonas, Ara Darzi, and Guang-Zhong Yang. 2006. HMM assessment of quality of movement trajectory in laparoscopic surgery. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006* (2006), 752–759.

[48] Xiaochen Li, Wenji Mao, Daniel Zeng, and Fei-Yue Wang. 2010. Automatic construction of domain theory for attack planning. In *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on*. IEEE, 65–70.

[49] Dingding Liu, Bilge Soran, Gregg Petrie, and Linda Shapiro. 2012. A review of computer vision segmentation algorithms. *Lecture notes* 53 (2012).

[50] Constantinos Loukas, Vasileios Lahanas, and Evangelos Georgiou. 2013. An integrated approach to endoscopic instrument tracking for augmented reality applications in surgical simulation training. *The International Journal of Medical Robotics and Computer Assisted Surgery* 9, 4 (2013), e34–e51.

[51] David G Lowe. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Vol. 2. Ieee, 1150–1157.

[52] Quan-Tuan Luong and Olivier D Faugeras. 1996. The fundamental matrix: Theory, algorithms, and stability analysis. *International journal of computer vision* 17, 1 (1996), 43–75.

[53] Antonio M. López, Maria Vanrell, and Ernest Valveny. 2015. Detección de objetos. (2015). https://www.coursera.org/learn/deteccion-objetos/home/welcome

[54] Nasim Mansurov. n. d. What is Distortion? (n. d.). https://photographylife.com/what-is-distortion

[55] John D. Mason, James Ansell, Neil Warren, and Jared Torkington. 2013. Is motion analysis a valid tool for assessing laparoscopic skill? *Surgical Endoscopy* 27, 5 (01 May 2013), 1468–1477.

[56] Jiri Matas, Charles Galambos, and Josef Kittler. 2000. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding* 78, 1 (2000), 119–137.

[57] Stefano Mattoccia. 2011. Stereo vision: algorithms and applications. (2011).

[58] William C McGaghie. 2008. Research opportunities in simulation-based medical education using deliberate practice. *Acad. Emerg. Med.* 15, 11 (Nov. 2008), 995–1001.

[59] Alex Mihailidis, Jennifer N Boger, Tammy Craig, and Jesse Hoey. 2008. The COACH prompting system to assist older adults with dementia through handwashing: An efficacy study. *BMC geriatrics* 8, 1 (2008), 28.

[60] C. Monserrat, J. Hernández-Orallo, J.F. Dolz, M.J. Rupérez, and P. Flach. 2016. Knowledge Acquisition by Abduction for Skills Monitoring: Application to Surgical Skills. *26th International Conference on Inductive Logic Programming, ILP2016* (2016).

[61] C. Monserrat, A. Lucas, J. Hernández-Orallo, and M. José Rupérez. 2014. Automatic supervision of gestures to guide novice surgeons during training. *Surgical Endoscopy* 28, 4 (2014), 1360–1370.

[62] Travis Oliphant. 2005. NumPy — NumPy. (2005). http://www.numpy.org/

[63] Nobuyuki Otsu. 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics* 9, 1 (1979), 62–66.

[64] German I Parisi and Stefan Wermter. 2016. A Neurocognitive Robot Assistant for Robust Event Detection. In *Trends in Ambient Intelligent Systems*. Springer, 1–27.

[65] Matti Pietikinen, Abdenour Hadid, Guoying Zhao, and Timo Ahonen. 2011. *Computer Vision Using Local Binary Patterns* (1st ed.). Springer Publishing Company, Incorporated.

[66] Srinivas K. Prasad, Masaya Kitagawa, Gregory S. Fischer, Jason Zand, Mark A. Talamini, Russell H. Taylor, and Allison M. Okamura. 2003. *A Modular 2-DOF Force-Sensing Instrument For Laparoscopic Surgery*. Springer Berlin Heidelberg, Berlin, Heidelberg, 279–286.

[67] János Radó, Csaba Dücső, Gábor Battistig, Gábor Szebényi, Péter Fürjes, Zbigniew Nawrat, and Kamil Rohr. 2016. 3D force sensors for laparoscopic surgery tool. In *Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP), 2016 Symposium on*. IEEE, 1–4.

[68] Oliver Ray. 2009. Nonmonotonic abductive inductive learning. *J. Appl. Log.* 7, 3 (2009), 329–340.

[69] Scott E Regenbogen, Caprice C Greenberg, David M Studdert, Stuart R Lipsitz, Michael J Zinner, and Atul A Gawande. 2007. Patterns of technical error among surgical malpractice claims: an analysis of strategies to prevent injury to surgical patients. *Annals of surgery* 246, 5 (2007), 705–711.

[70] Antonio Morandeira Rivas, Arancha Cabrera Vilanova, Fátima Sabench Pereferrer, Mercè Hernández González, and Daniel del Castillo Déjardin. 2010. Simulador de bajo coste para el entrenamiento de habilidades laparoscópicas básicas. *Cirugía Española* 87, 1 (2010), 26–32.

[71] Jacob Rosen, Jeffrey D Brown, Lily Chang, Mika N Sinanan, and Blake Hannaford. 2006. Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete Markov model. *IEEE Transactions on Biomedical engineering* 53, 3 (2006), 399–413.

[72] Richard M Satava. 2005. Identification and reduction of surgical error using simulation. *Minimally Invasive Therapy & Allied Technologies* 14, 4-5 (2005), 257–261.

[73] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. 2008. 3-d depth reconstruction from a single still image. *International journal of computer vision* 76, 1 (2008), 53–69.

[74] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

[75] Henry Schneiderman and Takeo Kanade. 2000. A statistical method for 3D object detection applied to faces and cars. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, Vol. 1. IEEE, 746–751.

[76] smidm. 2015. Camera Calibration Using OpenCV. (2015). https://github.com/smidm/video2calibration

[77] Jorge Solis, Nobuki Oshima, Hiroyuki Ishii, Noriyuki Matsuoka, Atsuo Takanishi, and Kazuyuki Hatake. 2009. Quantitative assessment of the surgical training methods with the suture/ligature training system WKS-2RII. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 4219–4224.

[78] Nicholas Stylopoulos, Stéphane Cotin, SK Maithel, M Ottensmeyer, PG Jackson, RS Bardsley, PF Neumann, DW Rattner, and SL Dawson. 2004. Computer-enhanced laparoscopic training system (CELTS): bridging the gap. *Surgical Endoscopy and Other Interventional Techniques* 18, 5 (2004), 782–789.

[79] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[80] Richard Szeliski. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.

[81] Vincent Torre and Tomaso A Poggio. 1986. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1986), 147–163.

[82] OpenCV-Python Tutorials. 2014a. Camera Calibration — OpenCV-Python Tutorials 1 documentation. (2014). http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html

[83] OpenCV-Python Tutorials. 2014b. Epipolar Geometry — OpenCV-Python Tutorials 1 documentation. (2014). http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html

[84] OpenCV-Python Tutorials. 2014c. Image Segmentation with Watershed Algorithm — OpenCV-Python Tutorials 1 documentation. (2014). http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html

[85] César Urrutia and El Mundo. 2014. Renfe gasta 3,8 millones en cámaras para vigilar a los maquinistas. http://www.elmundo.es/economia/2014/04/01/5339f845e2704eac368b458b.html. (1 April 2014). Accessed: 2017-7-31.

[86] Koen EA van de Sande, Theo Gevers, and Cees GM Snoek. 2008. Color descriptors for object category recognition. In *Conference on Colour in Graphics, Imaging, and Vision*, Vol. 2008. Society for Imaging Science and Technology, 378–381.

[87] Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. 2008. *Handbook of Knowledge Representation*. Elsevier.

[88] Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, Vol. 1. IEEE, I–I.

[89] Paul Viola and Michael J Jones. 2004. Robust real-time face detection. *International journal of computer vision* 57, 2 (2004), 137–154.

[90] Ian H White, Michael J Crisp, and Richard V Penty. 2010. A photonics based intelligent airport surveillance and tracking system. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. IEEE, 11–16.

[91] MS Wilson, A Middlebrook, C Sutton, R Stone, and RF McCloy. 1997. MIST VR: a virtual reality trainer for laparoscopic surgery assesses performance. *Annals of the Royal College of Surgeons of England* 79, 6 (1997), 403.

[92] Ziyan Wu, Yang Li, and Richard J Radke. 2015. Viewpoint invariant human re-identification in camera networks using pose priors and subject-discriminative features. *IEEE transactions on pattern analysis and machine intelligence* 37, 5 (2015), 1095–1108.

[93] Alper Yilmaz, Omar Javed, and Mubarak Shah. 2006. Object tracking. *Comput. Surveys* 38, 4 (2006), 13–es.

[94] Kristina Yordanova and Thomas Kirste. 2016. Learning Models of Human Behaviour from Textual Instructions.. In *ICAART (2)*. 415–422.

APÉNDICE A

Recursos para el entrenamiento de habilidades laparoscópicas

GENERALITAT VALENCIANA

La Fe Hospital Universitari i Politècnic

Innovación y experiencia al servicio del paciente

EJERCICIOS DE TRAINER LAPAROSCÓPICOS.

(Adaptación de ejercicios para Endo-Trainer ETHICON ENDO-SURGERY, INC – Johnson & Johnson company)

Coordinación mano-ojos:

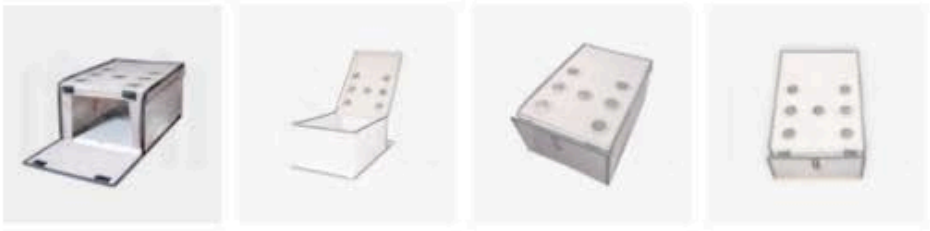
EJERCICIO #1: (8 mov)

- Coloca el envase de piezas cilíndricas abierto con las piezas en la base perforada en el campo de visión.
- Coloca la tapa del envase a 4-5 espacios del envase.
- Utilizando la mano dominante y el disector, mueve las piezas a la tapa del envase.
- Repite el ejercicio usando la mano no dominante, moviendo las piezas otra vez al envase.
- Repite el ejercicio – esta vez pasa las piezas de la mano dominante a la mano no dominante y después colócalas en el envase.

HOSPITAL UNIVERSITARI I POLITÈCNIC "LA FE". Bulevar Sur, s/nº. 46026 VALENCIA. TEL. 961244227 - 961244000

Figura A.1: Ejemplo de ejercicio de entrenamiento de laparoscopia para la coordinación mano-ojo. (Imagen extraída del curso de iniciación la cirugía laparoscópica del Hospital Universitari i Politécnic LaFe de Valencia).

Figura A.2: Aula de entrenamiento de habilidades laparoscópicas en el Hospital LaFe (Valencia). (Imagen extraída del curso de iniciación la cirugía laparoscópica del Hospital Universitari i Politécnic LaFe de Valencia).



Description

Additional Information

Reviews (0)

Additional Information

height	22 Cm
length	47 cm
weight	5 kgm
width	35 cm

Figura A.3: Dimensiones y peso de box-trainer original empleado para el entrenamiento de habilidades laparoscópicas. (Imagen proporcionada por el Hospital Universitari i Politécnic LaFe de Valencia).

APÉNDICE B

Recursos empleados en la experimentación

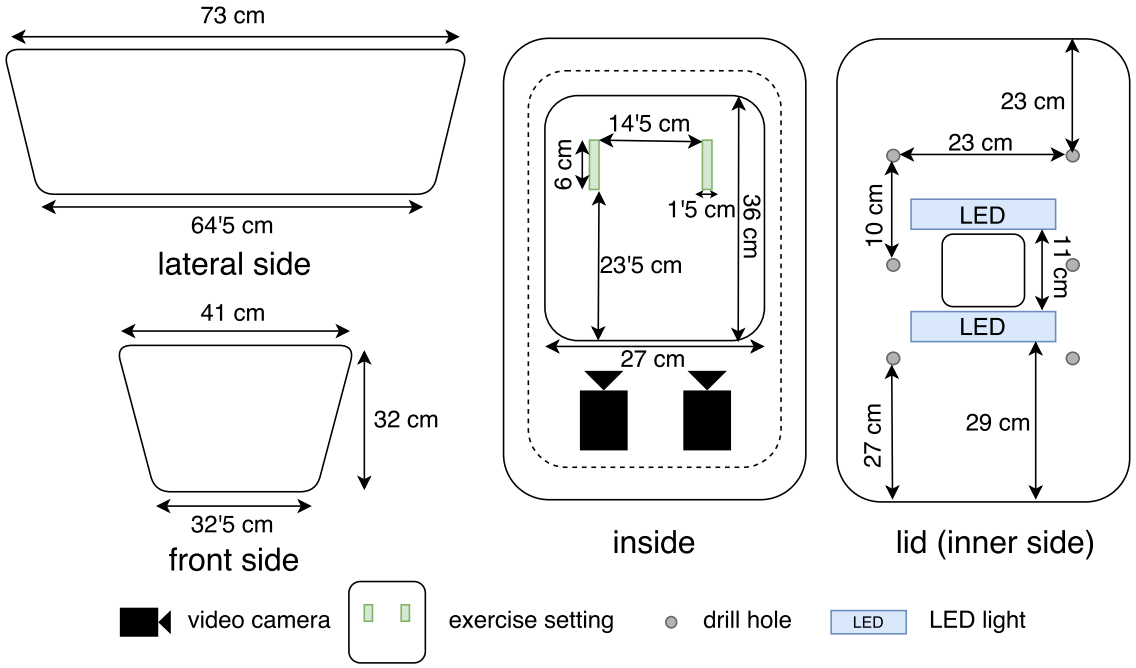


Figura B.1: Plano del simulador laparoscópico creado durante el proyecto. (Imagen de elaboración propia).

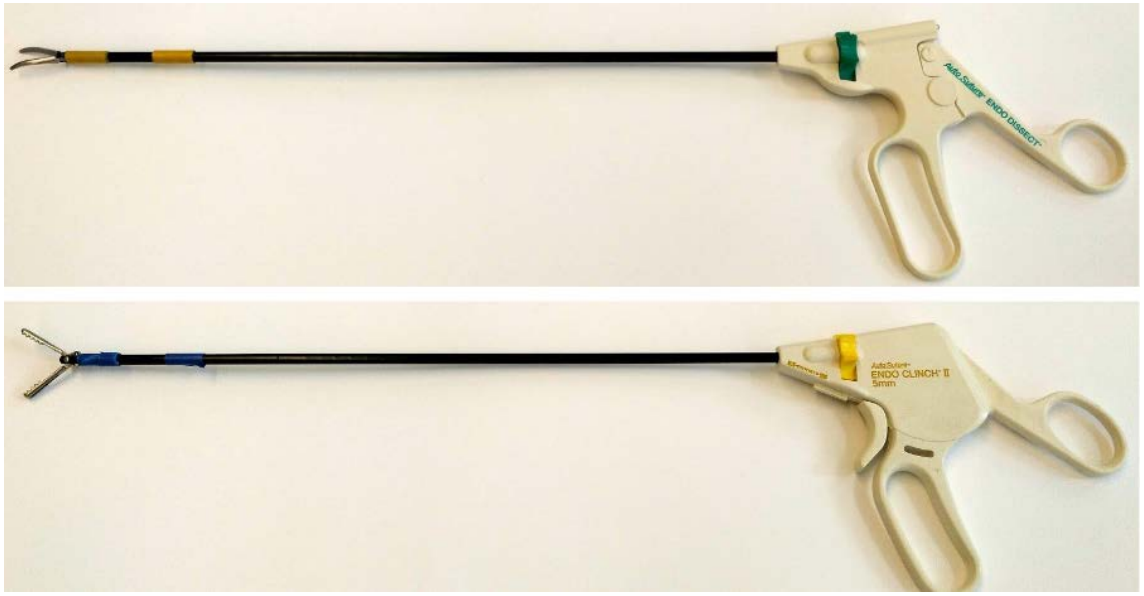


Figura B.2: Instrumental quirúrgico empleado para la experimentación. Disector *AutoSuture Endo Dissect 5 mm* (parte superior) y fórceps *AutoSuture Endo Clinch II 5 mm* (parte inferior). (Imagen de elaboración propia).

Element	Picture	Color	Min HSV	Max HSV
Forceps			(101,164,40)	(171,255,255)
Dissector			(20,35,40)	(75,255,255)
Target Mark			(30,40,42)	(94,255,255)
Object			(125,45,25)	(179,255,255)

Tabla B.1: Conjunto colores y marcas usados durante el desarrollo

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<config>
   <color_marks>
      <color_mark id="01" name="forceps" type="instrument"
         meta="blue" num_of_marks="2">
         <lower_color H="101" S="164" V="40" />
         <upper_color H="171" S="255" V="255" />
      </color_mark>
      <color_mark id="02" name="dissector" type="instrument"
         meta="yellow" num_of_marks="2">
         <lower_color H="20" S="35" V="40" />
         <upper_color H="75" S="255" V="255" />
      </color_mark>
      <color_mark id="03" name="left_pan" type="pan" meta="
         green" num_of_marks="1">
         <lower_color H="30" S="40" V="42" />
         <upper_color H="94" S="255" V="255" />
      </color_mark>
      <color_mark id="04" name="right_pan" type="pan" meta="
         green" num_of_marks="1">
         <lower_color H="30" S="40" V="42" />
         <upper_color H="94" S="255" V="255" />
      </color_mark>
      <color_mark id="05" name="object" type="object" meta="
         brown" num_of_marks="1">
         <lower_color H="125" S="45" V="25" />
         <upper_color H="179" S="255" V="255" />
      </color_mark>
   </color_marks>
</config>
```

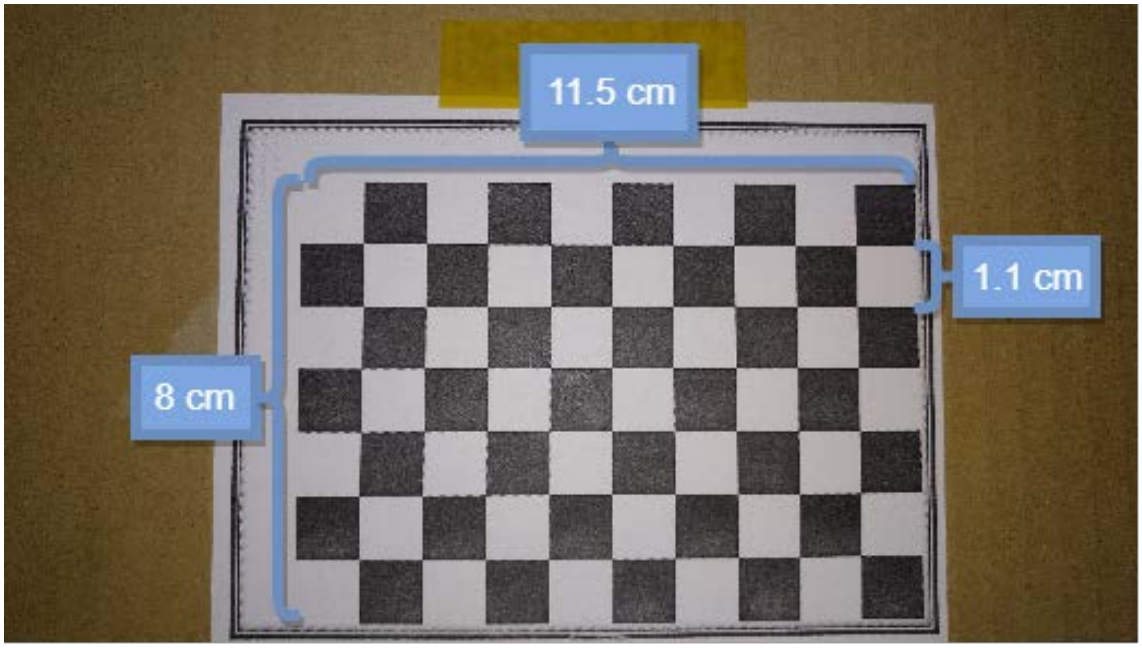
código B.1: config.xml - Archivo de configuración de marcas y colores

```
[
{
"Instrument": "forceps",
"Frame": 121,
"Time_Frame": 11,
"Event": "appears"
},
{
"Instrument": "forceps",
"Frame": 133,
"Time_Frame": 13,
"Event": "starts_moving"
},
{
"Instrument": "forceps",
"Frame": 140,
"Time_Frame": 14,
"Event": "stops"
},
{
"Instrument": "forceps",
"Frame": 141,
"Time_Frame": 14,
"Event": "opens"
},
{
"Instrument": "forceps",
"Frame": 162,
"Time_Frame": 17,
"Event": "starts_moving"
},
{
"Instrument": "forceps",
"Frame": 162,
"Time_Frame": 17,
"Event": "crosses_in_10",
"Target": "right_pan"
},
{
"Instrument": "forceps",
"Frame": 173,
"Time_Frame": 19,
"Event": "crosses_in_5",
"Target": "right_pan"
},
{
"Time_Frame": 21,
"Instrument": "forceps",
"Frame": 186,
"Object": "Bean",
"Event": "picks"
```

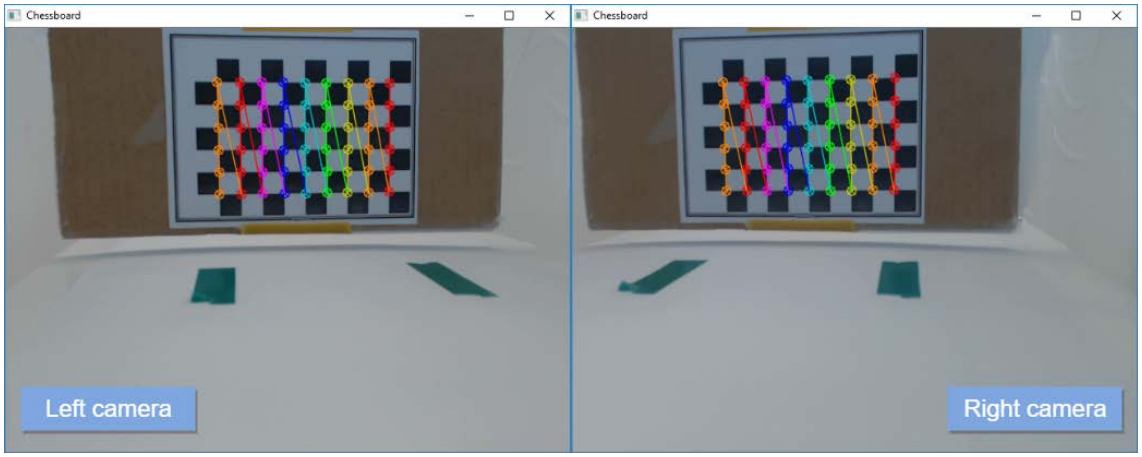
```
},
{
"Instrument": "forceps",
"Frame": 194,
"Time_Frame": 22,
"Event": "stops"
},
{
"Instrument": "forceps",
"Frame": 215,
"Time_Frame": 25,
"Event": "crosses_out_5",
"Target": "right_pan"
},
{
"Instrument": "forceps",
"Frame": 218,
"Time_Frame": 25,
"Event": "crosses_out_10",
"Target": "right_pan"
},
{
"Instrument": "forceps",
"Frame": 219,
"Time_Frame": 25,
"Event": "starts_moving"
},
{
"Instrument": "forceps",
"Frame": 245,
"Time_Frame": 29,
"Event": "crosses_in_10",
"Target": "left_pan"
},
{
"Instrument": "forceps",
"Frame": 247,
"Time_Frame": 29,
"Event": "crosses_in_5",
"Target": "left_pan"
},
{
"Instrument": "forceps",
"Frame": 264,
"Time_Frame": 32,
"Event": "stops"
},
{
"Instrument": "forceps",
"Frame": 284,
"Time_Frame": 35,
"Event": "crosses_out_5",
```

```
"Target": "left_pan"
},
{
"Instrument": "forceps",
"Frame": 284,
"Time_Frame": 35,
"Event": "opens"
},
{
"Time_Frame": 35,
"Instrument": "forceps",
"Frame": 284,
"Object": "Bean",
"Event": "drops"
},
{
"Instrument": "forceps",
"Frame": 286,
"Time_Frame": 35,
"Event": "starts_moving"
},
{
"Instrument": "forceps",
"Frame": 288,
"Time_Frame": 35,
"Event": "crosses_out_10",
"Target": "left_pan"
},
{
"Instrument": "forceps",
"Frame": 302,
"Time_Frame": 37,
"Event": "stops"
},
{
"Instrument": "forceps",
"Frame": 308,
"Time_Frame": 38,
"Event": "closes"
},
{
"Instrument": "forceps",
"Frame": 317,
"Time_Frame": 39,
"Event": "disappears"
}
]
```


código B.2: Salida del prototipo en formato JSON tras la ejecución con el ejercicio de ejemplo



(a) Patrón de calibración empleado durante el desarrollo. (Imagen de elaboración propia).



(b) Detección de los puntos de calibración en cada una de las cámaras con la librería StereoVision. (Imagen de elaboración propia).

Figura B.3: Proceso de calibración de las cámaras para aplicar la visión estereoscópica por ordenador.

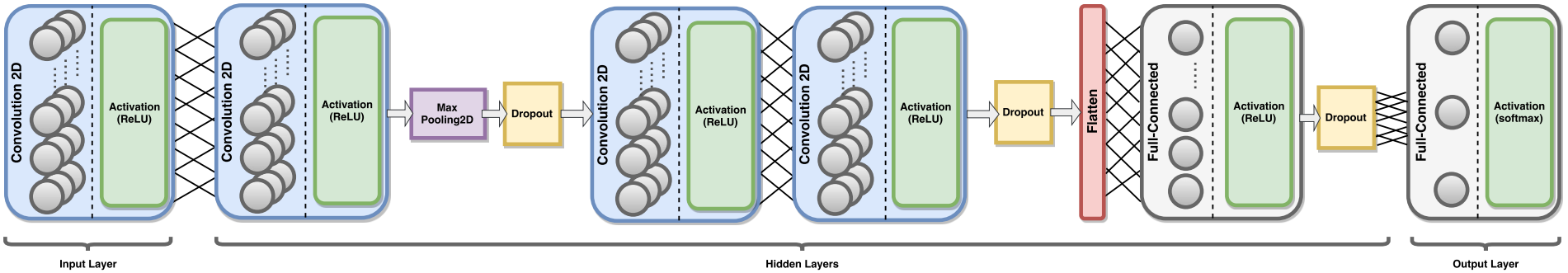


Figura B.4: Arquitectura de la red convolucional empleada. (Imagen de elaboración propia).

APÉNDICE C

Publicaciones relacionadas con el trabajo (Texto completo)

C.1 Low-level Event Detection System for Minimally-Invasive Surgery Training

David Nieves, Cèsar Ferri, José Hernández-Orallo and Carlos Monserrat

iWOAR 2017 — 4th international Workshop on Sensor-based Activity Recognition and Interaction, Rostock (Germany), 2017

Low-level Event Detection System for Minimally-Invasive Surgery Training

David Nieves
Universitat Politècnica de València
Valencia, Spain
daniecor@dsic.upv.es

Cèsar Ferri
Universitat Politècnica de València
Valencia, Spain
cferri@dsic.upv.es

José Hernández-Orallo
Universitat Politècnica de València
Valencia, Spain
jorallo@dsic.upv.es

Carlos Monserrat
Universitat Politècnica de València
Valencia, Spain
cmonserr@dsic.upv.es

ABSTRACT

We present an event detection system in a laparoscopic surgery domain, as part of a more ambitious supervision by observation project. The system, which only requires the incorporation of two cameras in a laparoscopic training box, integrates several computer vision and machine learning techniques to detect the states and movements of the elements involved in the exercise. We compare the states detected by the system with the hand-labelled ground truth, using an exercise of the domain as example. We show that the system is able to detect the events accurately.

Author Keywords

Event Detection; Automated Monitoring; Learning by Observation; Computer Vision; Machine Learning

INTRODUCTION

Computers, endowed with artificial intelligence techniques, are able to automate many processes nowadays. However, many tasks still resist automation, and are performed by humans, especially those that require complex cognition and abstract knowledge. Still, with current technology, we can help humans learn, supervise and complete these tasks more efficiently. We envisage those technologies that *assist* humans when they are learning or performing tasks. In this way, we pursue the automation of supervision by observation. This means automated assistants that, after the observation of how an expert performs a task, are able to supervise whether other humans are performing this task correctly, also by observation.

iWOAR '17, September 21–22, 2017, Rostock, Germany

ACM ISBN 978-1-4503-5223-9/17/09...$15.00
https://doi.org/10.1145/3134230.3134241

According to previous description, we propose a learning process based on a few meaningful demonstrations, together with a high-level narrative about its proper fulfilment. As a result, the assistant will act as a virtual expert in charge of advising and assessing the apprentice during task achievement. Considering the character of the learning resources described above, classical machine learning techniques may not be appropriate for extracting knowledge in this context, because the level of abstraction, the need for background knowledge and the small number of examples. However, inductive programming (IP) [8], an area between machine learning and declarative programming, can be very useful for this goal. IP has recently shown a versatility in novel applications, as the Flash Fill tool in Microsoft Excel [7].

Supervision by observation is made up of two sequential phases. The first one involves all knowledge acquisition by the system from examples and the narrative. More specifically, the knowledge acquisition is accomplished with "Induction by Abduction" [16], which merges a high-level description of the task, called narrative, with the observable low-level events captured from a few video recordings where the task is developed. The second phase is properly when the system monitors the task performed by apprentice. More precisely, the system monitors new sequences of events and generalises them to extract the high-level description, which is compared with the intended high-level description of the task. In this manner, the supervision agent will be capable to produce an informative feedback to the trainee.

The detection of these observable events is a crucial factor for the whole system. The more accurate events are recognised at the low-level, the better the mapping with the high-level description will be made. In a previous work [16], the event detection was performed manually to show the "Induction by Abduction" process. In this paper, we focus on the automation of the event detection [15], with a fully-fledged *Event Detector* that detects these low-level events. The system adapts a series

of state-of-the-art computer vision [27] and machine learning techniques [5] and methods to provide an integrated solution.

We will demonstrate the potentiality of supervision by observation with a significant application domain, Minimally-Invasive Surgery (MIS). MIS requires periods of surgeon training, guided by experts who give immediate feedback during each exercise [22]. Experts teach basic surgery skills to apprentices with the use of a number of exercises and workouts [19]. Apprentice has to dominate the required skills, all inside a controlled environment (a laparoscopic training box). The critical resources in the whole process are the cost of the boxes (physical boxes are much cheaper than virtual simulators) and the expert time (teachers have to share their limited time among a series of students, practising for many hours on many exercises, so quality feedback is expensive). With the setting proposed in the paper we make it possible to combine physical boxes (by adding simple, cheap equipment to each of them, such as two cameras) and performing a high-level supervision automatically. Therefore, this domain seems a favourable context where we can significantly advance the state of the art and achieve a cost-effective product in the market.

The rest of the paper is organised as follows. The following

section summarises some relevant related work. The third section describes the general approach for the Event Detector, along with the hardware and software specification required to create it. Furthermore, the main computer vision techniques applied during this process is overviewed. Then, in the *Results* section, we will show an example of the type of information that is generated by the Event Detector. At the same time, we will compare an output of this component with a hand-labelled ground truth, using an exercise of the domain as example. In the *Discussion* section, we will analyse the strengths and weaknesses we have. Finally, the last section shows the major outcomes from this work and the following steps.

RELATED WORK

In [17], the authors present a novel method for automatic evaluation of a surgeon's skills that is able to supervise inexpert surgeons in their training session with surgical simulators. This idea of using computers to assist the training of inexperienced surgeons, mainly by simulation, has been previously proposed by other works [26, 2, 23]. Particularly, the different approaches in literature can be categorised depending on the methods they are based on: weights, Linear Discriminant Analyses (LDA), Markov models and fuzzy classifiers. For instance, the Clinical-Based Computer Enhanced Laparoscopic Training System (CELTS) [25] is a representative example of an evaluation method based on weights. This approach considers factors such as the accumulated depth carried out by the surgical instrument, the orientation of the tool, the length of the path followed by the instrument and the time required to finish the training. The novice's skill level is then employed to normalise the obtained values before providing a final mark.

A combination of the methods based on weights and the methods based on LDA is the so-called WKS-2RII approach [24]. This work considers parameters related to the evaluation of sutures: pressure applied on the skin, time required to finish the task, distance between the suture point and the limit of the wound, distance between sutures and the final wound opening. LDA is applied on a linear combination of these factors to classify the surgeons into *expert*, *intermediate* or *novice*.

Among the systems that use Markov Models, in [21] the authors estimate a distance between a sequence of words in spoken language and the sequence of gestures used during the surgery training. Two Hidden Markov Models (HMM) are inferred, one represents the state transitions of expert surgeons during laparoscopic surgery, while the other one is in charge of modeling the state transitions of an inexpert surgeon. After both HMMs are learned, a new surgeon is evaluated by estimating the distance between its sequence of states and the sequence of states from both HMMs. The closer group is used to label the new surgeon (minimum distance between the new surgeon and the considered group). Similar approaches can be found in [14, 12].

Systems presented in [10] and [9] are based on fuzzy classifiers to obtain three clusters of surgeons according to the skill levels: *expert*, *intermediate* and *novice*. The three clusters are learned using previous training with surgeons of known skills and applying a Fuzzy C-Means classifier. This fuzzy classifier can be employed to compute the skill level of a new surgeon.

Finally, event detection approaches are also of utmost relevance to this work and some of them have shown promising results in health applications. In particular, in [18] a robot monitors a person in a household environment. The robot is trained to detect falls in real time and then to approach the patient and ask them whether some assistance is required. On the other hand, sports could also be considered as a related field where the accuracy and repeatability of movements is crucial for the player performance. However, works in this area are not centered on online event detection but from videos [13, 28].

MATERIALS AND METHODS

Here we describe the hardware and software settings, as well as the specific integration techniques and ideas that we used in our event detector.

Hardware

We recreated a training box for laparoscopic exercises, usually known as "pelvitrainer", following the structure and conditions of the chosen domain. Figure 1 shows the plan of our box trainer. We took a plastic box with the following measurements: 73 cm x 41 cm x 32 cm (Length x Width x Height). Then, we covered it with a white vinyl layer to avoid the external light influence. The box lid was drilled to allow the insertion of the surgical instruments. The holes are placed at different positions to make it easier for different exercises. Inside the box, we added a couple of LED lights (*JESWELL USB LED Sensor Light*), attached to the inner side of the box lid, to facilitate the scene visualisation. We also added two video cameras (*Logitech C922 Pro Stream Webcam*) to capture the scene. Finally, the surgical instruments used during our tests were an *AutoSuture Endo Clinch II 5 mm* forceps and a

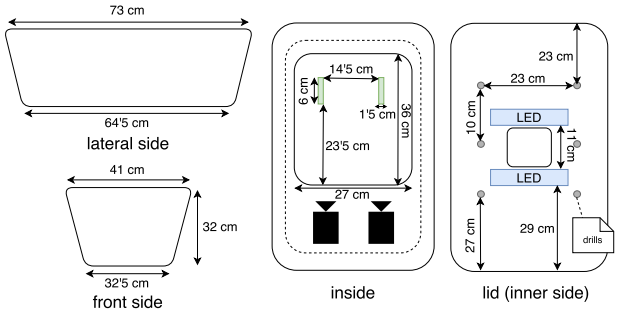


Figure 1. Laparoscopic box trainer to reproduce exercises for Minimally-Invasive Surgery (MIS).

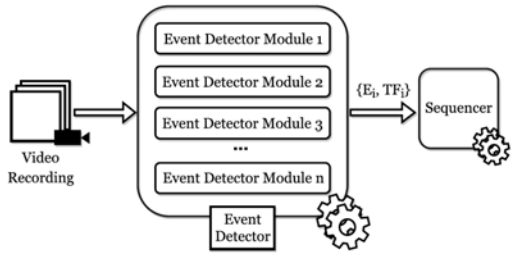


Figure 2. General software architecture.

AutoSuture Endo Dissect 5 mm dissector, to which we added some colour marks to be identified more easily.

Software Architecture

The *Event Detector* is implemented as a software component that is in charge of processing the video recordings and detecting the finest grain of information about the observable events. Specifically, the Event Detector consists of a set of modules, which work as independent threads, each one specialised in detecting one or several kinds of events, and a single *Sequencer* that integrates all of them according to their times. Figure 2 shows the general schema about how this process works. When the Event Detector component detects an event E_i, it is sent to the *Sequencer*, with the time frame information TF_i (a real value that represents the instant of time in which the event has been detected). Due to the fact that the events are detected in parallel by the different Event Detector modules, the sequence in which events are detected is not always the order in which these events happened. Finally, the Sequencer component will use the information generated by Event Detector, to put them in the right order as a function of time.

The software solution has been developed using *Python 2.7* together with *OpenCV 2.4.13.2* [1] as computer vision library. Furthermore, we required the *Logitech Gaming Software* tool to adjust some low-level parameters of our cameras. Additionally, *Keras 2.0.1* [3] was used with a *TensorFlow* [4] backend to train an image classifier necessary for some events detection. All the code is available on GitHub[1].

[1] https://github.com/supervasion/EventDetector

Event recognition

Our approach for covering the low-level events detection relies on four computer vision techniques: segmentation, tracking, stereo vision and object recognition. We have used a threshold-based image segmentation method [6], which allows us to extract the colour marks added to the instruments and objects throughout video sequences. It generates a binary image that facilitates the detection of shapes and contours. Using this method, we are able to know when an instrument appears or disappears inside the box trainer and which instrument it is. In addition, the previous method was combined with a tracking technique to get the approximate 2D position of the screen-projected objects present in the scene and how they move.

Regarding the stereo vision, one of the most common approaches is the generation of a *disparity map image* [20], where the grey-scale intensity of each pixel is inversely related to the distance from the cameras. Before using it, we calibrated the cameras for fixing the lens distortion. In our case, we used a third-party library called Stereovision[2] to perform this calibration. Combining stereo vision with thresholding, we can detect the colour marks in the disparity map and extract the depth value of them. This is obtained by averaging pixel values inside the colour mark contour (discarding zero-values). We added this information to the position obtained by tracking, being able to determine the 3D position of the different elements improving the tracking measure.

Finally, we used *Convolutional Neural Networks* [11], better known as *ConvNets* or *CNNs*, to train an image classifier that recognises the state of the instrument tip during the exercise. To facilitate this task, we use the colour marks added on the surgical instruments as a reference in order to create a *Region of Interest* in real-time over the end where the clamp is located. Figure 3 shows this process. First of all, we detect the contours of each mark and their mass center. Then, we calculate the distance between them and project it over the bottommost mark. In this manner, we can create a geometrical shape on the tip of the instrument and extract this piece of the image to use it later. Particularly, we used these images to create a sample generator, to train our classifier, as well as getting a specific image of the instrument tip and detect its state while the exercise goes by. Concerning the image classifier, we were interested in the detection of three classes related to the possible states that the instrument tip can take. They are:

- *close*: The clamp of the instrument is closed.
- *open*: The instrument is open but it has not grabbed any object.
- *object_clamped*: The instrument tip is grabbing an object.

Figure 3 shows one example of each class described above. For the creation of the classifier, we chose the Python library *Keras*[3]. We selected the multiclass Sequential Model and a *simple deep CNN architecture*[4] to work with small images.

[2] https://github.com/erget/StereoVision

[3] https://keras.io/

[4] https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py

	close	object_clamped	open
close	245	0	6
object_clamped	0	251	0
open	0	0	250

Table 1. Confusion matrix of the image classifier

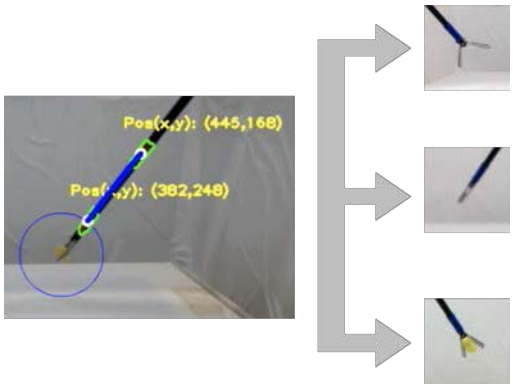


Figure 3. Region of Interest (ROI) extraction over the instrument tip. The right side shows a sample of three different states (classes) for this tip. From top to bottom: *open*, *close* and *object_clamped*.

Subsequently, we took about 1200 images of each class for both training and evaluation: 80% of the data is used for training, and the rest of unseen data (20%) is used for test. Table 1 shows the confusion matrix for the test data.

Given a classifier for each object, for each frame captured with the cameras, the prototype executes the classifiers, returning an output file with a sequence of the events detected. Table 2 shows a brief example of this type of output and the event information generated by the prototype. The frame and time-frame columns indicate the concrete frame and the instant of time when an event is detected. The next three columns (Event, Instrument, Target) contain information about the type of event and its parameters. In this phase of development, we focus on ensuring the detection of events and whether they are in the right order.

RESULTS

With the aim of testing our prototype, we recorded a video of a simulation exercise in which the trainee has grasp and moved a small object. Figure 4 shows a couple of frames taken from that video recording. Despite the fact that this prototype can be configured for exercises with multiple instruments, we chose a one-instrument exercise as a starting point to ease the evaluation of the results. Concretely, the exercise starts with a surgical instrument (forceps) appearing on the right side of the box trainer. After that, the trainee picks up a bean (using the forceps) placed at particular mark on the ground and then moves the bean to the mark on the left side. Finally, the instrument moves away from the latter mark and disappears from the box trainer.

The Event Detector module is executed in order to obtain the whole sequence of events of the video. Table 4 shows the results obtained (sequence of events) as well as a comparison between the output of our prototype and the hand-labelled

Frame	Time-Frame	Event	Instrument	Target
135	12	appears	dissector	-
144	13	starts_moving	dissector	-
158	15	crosses_in_10	dissector	right_pan
168	16	opens	dissector	-
169	17	crosses_in_5	dissector	right_pan
170	17	stops	dissector	-

Table 2. Output example from the event detector.

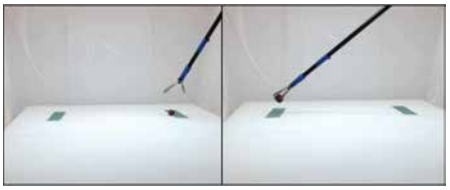


Figure 4. Frames extracted from the video recording with the exercise simulation.

ground truth. Specifically, rows represent the low-level events involved in this exercise (explained in more detail in Table 3). The columns state the order in which each event happens. Regarding the results in the cells, from states 1 to 7, our prototype perfectly matches the ground truth sequence. It captures: the forceps appearance, its change of state from closed to open and how it approaches to the right mark. All of these events are detected in the correct order. Later, there is a mismatch from states 8 to 12. Even though the captured events are correct, the sequence is not. Some of them are swapped with respect to their correct order: states 8 and 9 (*stops* and *closes* events), and states 10 and 12 (*starts_moving* events). Additionally, some others are delayed in time: states from 10 to 12 (*crosses_out_5* and *crosses_out_10* events). Then again, the prototype perfectly matches the ground truth from states 13 to 17: the forceps approaches the left mark and leaves the bean. After that, another swap appears between states 18 and 19 (*starts_moving* and *crosses_out_5*), and both sequences match again from states 20 to 22, when the forceps moves away from left mark, stopping and closing the tip. At the end of the exercise, from states 23 to 24, the prototype detects the forceps disappearance but not its movement. In the following section we discuss these results.

DISCUSSION

The results show that the prototype fits the ground truth reasonably well during the example execution. On one hand, and concerning the detection of events, all the events were captured by the prototype, with the only exception of the last *starts_moving* event. This led us to think that the classifiers are working properly. On the other hand, and focusing only on the accuracy of the sequence, we found some mistakes in the order of events. Specifically, we can identify a swap pattern, mainly generated by *stops* and *starts_moving* events. These particular events are the most difficult to model due to the fact that the system has to distinguish when a deliberate move starts from another that is not deliberate. Factors like the trainee pulse or fluctuations on the disparity map values can trigger the movement detection, generating noise in the output. Therefore, as an straightforward solution, we decided to add a

Events	Description
closes(<instrument>)	The instrument is closed
drops(<instrument>, <object>)	The object caught by the instrument is released
starts_moving(<instrument>)	The instrument starts moving
stops(<instrument>)	The instrument that was moving stops
appears(<instrument>)	The instrument enters the camera framing
disappear(<instrument>)	The instrument leaves the camera framing
opens(<instrument>)	The instrument is opened
picks(<instrument>, <object>)	The instrument clamps the object
crosses_ in_5(<instrument>, <target>)	The instrument enters into the 5 units zone from the target
crosses_ in_10(<instrument>, <target>)	The instrument enters the 10 units zone from the target
crosses_out_5(<instrument>, <target>)	The instrument leaves the 5 units zone from the target
crosses_out_10(<instrument>, <target>)	The instrument leaves the 10 units zone from the target

Table 3. Low-level events involved in the sample exercise.

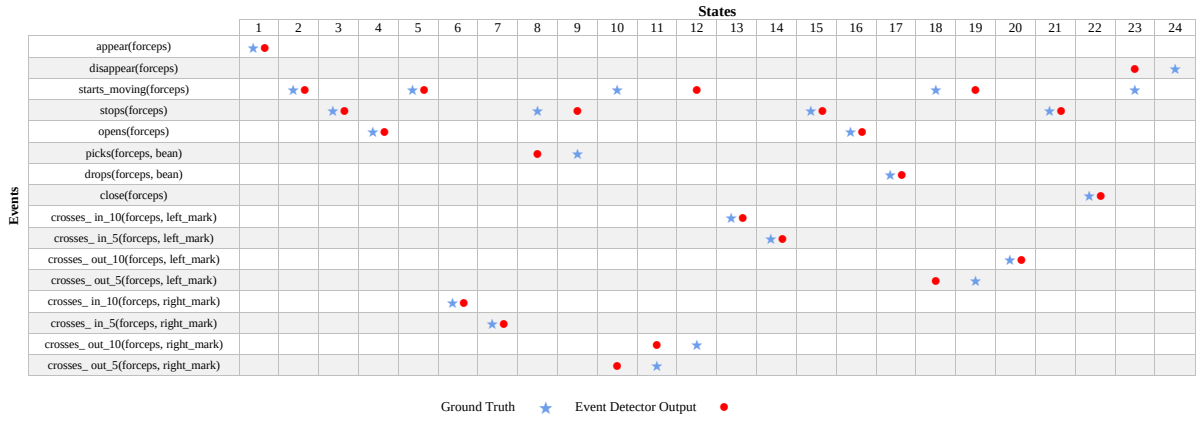


Table 4. Comparative between ground truth and the output from the Event Detector. The description of the events in rows can be found in Table 3. Columns (States) represent the order in which events happens.

threshold to these kind of events thus providing a safety margin where the different instruments are allowed to (slightly) move around without triggering the events. However, this generates the undesired side effect of delaying the detection of new events, therefore misaligning the output sequence with regard to the ground truth. One possible solution would be to introduce dependency rules and constraints between events. For instance, the forceps could only approach, move away or disappear from the box trainer if the *starts_moving* event was launched firstly. In the same manner, an object (the bean in our example) could only be picked up if the instrument tip has been open previously. This will include real world restrictions within the event detection process, therefore fixing some of the inconsistencies that appear in the output sequence. We consider that these constraints could be derived either from the analysis of further exercises, or from the narrative in natural language of the exercises.

Besides, issues such as object occlusion have to be addressed too. Since video cameras are located only in one side of the box, an instrument may hide other color marks during exercise performance. This fact can alter the event detection methods based on a thresholding technique.

CONCLUSION AND FUTURE WORK

In this paper we have described part of a more ambitious goal for supervision by observation, outlining here the Event Detection component that includes computer vision and machine learning techniques for this purpose. We have tested a prototype with an example of the domain (Minimally-Invasive Surgery), comparing the output with the ground truth. The results are sufficiently good to move from this low-level event detection and recognition to a more high-level interpretation of events such that the feedback can be of a higher quality. This requires the integration of knowledge from the description of the exercise in natural language, constraints about the world learnt from one or more exercises (and possible their descriptions) and possibly some feedback about the supervision process itself.

REFERENCES

1. Gary Bradski and Adrian Kaehler. 2013. *Learning OpenCV: Computer Vision in C++ with the OpenCV Library* (2nd ed.). O'Reilly Media, Inc.
2. Magdalena K Chmarra, Niels H Bakker, Cornelis A Grimbergen, and Jenny Dankelman. 2006. TrEndo, a device for tracking minimally invasive surgical instruments in training setups. *Sensors and Actuators A: Physical* 126, 2 (2006), 328–334.

3. François et al. Chollet. 2015. Keras. https://github.com/fchollet/keras. (2015).
4. Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.
5. Peter Flach. 2012. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, New York, NY, USA.
6. R.C. Gonzalez and R.E. Woods. 1992. *Digital Image Processing*. Addison-Wesley.
7. Sumit Gulwani, William R Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* 55, 8 (2012), 97–105.
8. Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, and Benjamin Zorn. 2015. Inductive Programming Meets the Real World. *Commun. ACM* 58, 11 (Oct. 2015), 90–99.
9. Ima Hajshirmohammadi and Shahram Payandeh. 2007. Fuzzy Set Theory for Performance Evaluation in a Surgical Simulator. *Presence: Teleoper. Virtual Environ.* 16, 6 (Dec. 2007), 603–622.
10. Huang Jeff, Shahram Payandeh, Peter Doris, and Ima Hajshirmohammadi. 2005. Fuzzy Classification: Towards Evaluating Performance on a Surgical Simulator. *Medicine Meets Virtual Reality 13: The Magical Next Becomes the Medical Now* 111 (2005), 194.
11. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
12. Julian Leong, Marios Nicolaou, Louis Atallah, George Mylonas, Ara Darzi, and Guang-Zhong Yang. 2006. HMM assessment of quality of movement trajectory in laparoscopic surgery. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006* (2006), 752–759.
13. Baoxin Li and M Ibrahim Sezan. 2001. Event detection and summarization in sports video. In *Content-Based Access of Image and Video Libraries, 2001.(CBAIVL 2001). IEEE Workshop on*. IEEE, 132–138.
14. Constantinos Loukas, Vasileios Lahanas, and Evangelos Georgiou. 2013. An integrated approach to endoscopic instrument tracking for augmented reality applications in surgical simulation training. *The International Journal of Medical Robotics and Computer Assisted Surgery* 9, 4 (2013), e34–e51.
15. Gérard Medioni, Isaac Cohen, François Brémond, Somboon Hongeng, and Ramakant Nevatia. 2001. Event detection and analysis from video streams. *IEEE Transactions on pattern analysis and machine intelligence* 23, 8 (2001), 873–889.
16. C. Monserrat, J. Hernández-Orallo, J.F. Dolz, M.J. Rupérez, and P. Flach. 2016. Knowledge Acquisition by Abduction for Skills Monitoring: Application to Surgical Skills. *26th International Conference on Inductive Logic Programming, ILP2016* (2016).
17. C. Monserrat, A. Lucas, J. Hernández-Orallo, and M. José Rupérez. 2014. Automatic supervision of gestures to guide novice surgeons during training. *Surgical Endoscopy* 28, 4 (2014), 1360–1370.
18. German I Parisi and Stefan Wermter. 2016. A Neurocognitive Robot Assistant for Robust Event Detection. In *Trends in Ambient Intelligent Systems*. Springer, 1–27.
19. Mark C Porte, George Xeroulis, Richard K Reznick, and Adam Dubrowski. 2007. Verbal feedback from an expert is more effective than self-accessed feedback about motion efficiency in learning new surgical skills. *The American Journal of Surgery* 193, 1 (2007), 105–110.
20. Ning Qian. 1997. Binocular disparity and the perception of depth. *Neuron* 18, 3 (1997), 359–368.
21. Jacob Rosen, Jeffrey D Brown, Lily Chang, Mika N Sinanan, and Blake Hannaford. 2006. Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete Markov model. *IEEE Transactions on Biomedical engineering* 53, 3 (2006), 399–413.
22. Tony Rousmaniere. 2014. Using technology to enhance clinical supervision and training. *The Wiley international handbook of clinical supervision* (2014), 204–237.
23. Richard M Satava. 2005. Identification and reduction of surgical error using simulation. *Minimally Invasive Therapy & Allied Technologies* 14, 4-5 (2005), 257–261.
24. Jorge Solis, Nobuki Oshima, Hiroyuki Ishii, Noriyuki Matsuoka, Atsuo Takanishi, and Kazuyuki Hatake. 2009. Quantitative assessment of the surgical training methods with the suture/ligature training system WKS-2RII. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 4219–4224.
25. N Stylopoulos, S Cotin, S Dawson, M Ottensmeyer, P Neumann, R Bardsley, M Russell, P Jackson, and D Rattner. 2003. CELTS: a clinically-based Computer Enhanced Laparoscopic Training System. *Studies in health technology and informatics* 94 (2003), 336.
26. Nicholas Stylopoulos, Stéphane Cotin, SK Maithel, M Ottensmeyer, PG Jackson, RS Bardsley, PF Neumann, DW Rattner, and SL Dawson. 2004. Computer-enhanced laparoscopic training system (CELTS): bridging the gap. *Surgical Endoscopy and Other Interventional Techniques* 18, 5 (2004), 782–789.
27. Richard Szeliski. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.
28. Di Zhong and Shih-Fu Chang. 2004. Real-time view recognition and event detection for sports video. *Journal of Visual Communication and Image Representation* 15, 3 (2004), 330–347.

C.2 Knowledge Extraction from Task Narratives

Kristina Yordanova, Carlos Monserrat, David Nieves, José Hernández-Orallo

iWOAR 2017 — 4th international Workshop on Sensor-based Activity Recognition and Interaction, Rostock (Germany), 2017

Knowledge Extraction from Task Narratives

Kristina Yordanova
University of Rostock
Rostock, Germany
kristina.yordanova@uni-rostock.de

Carlos Monserrat
Universitat Politècnica de València
Valencia, Spain
cmonserrat@dsic.upv.es

David Nieves
Universitat Politècnica de València
Valencia, Spain
daniecor@dsic.upv.es

José Hernández-Orallo
Universitat Politècnica de València
Valencia, Spain
jorallo@dsic.upv.es

ABSTRACT

One of the major difficulties in activity recognition stems from the lack of a model of the world where activities and events are to be recognised. When the domain is fixed and repetitive we can manually include this information using some kind of ontology or set of constraints. On many occasions, however, there are many new situations for which only some knowledge is common and many other domain-specific relations have to be inferred. Humans are able to do this from short descriptions in natural language, describing the scene or the particular task to be performed. In this paper we apply a tool that extracts situation models and rules from natural language description to a series of exercises in a surgical domain, in which we want to identify the sequence of events that are not possible, those that are possible (but incorrect according to the exercise) and those that correspond to the exercise or plan expressed by the description in natural language. The preliminary results show that a large amount of valuable knowledge can be extracted automatically, which could be used to express domain knowledge and exercies description in languages such as event calculus that could help bridge these high-level descriptions with the low-level events that are recognised from videos.

Author Keywords

Knowledge extraction; plan inference; possible worlds; surgical training; natural language processing; situation models

INTRODUCTION AND MOTIVATION

Humans are usually able to learn how to do something through a short explanation of the task and an explanatory video. The combination of both worlds (verbal and visual) are extremely powerful in reducing the ambiguities and misunderstandings that would otherwise appear by trying to do the same thing from just one of the sources.

In activity recognition, it is quite common to focus on the video or observation of a scene, from which the activities and events are recognised. This is so because in many applications the sequence of activities is completely open. For instance, in a house, when a person enters a room, many actions are possible and the goal is precisely to determine what the person is doing.

However, in other applications, there is a previously known descripton of what the person is going to do (or has to do). This is a common situation when humans have to follow pre-specified procedures, such as recipes, protocols or plans. Trying to recognise the activities ignoring the information of the procedure in natural language makes the problem more difficult than it truly is. Automated supervision or monitoring is one of these areas where we have a reference in natural language of what a user has to do (or can do).

Here we present work in progress with the goal of automating monitoring in a surgical training domain. The ultimate goal is to learn procedures from a high-level explanation given by an expert (a narrative) and a few video-recorded executions of the procedure (one in most cases). In this paper we focus on the first part, addressing the automatic interpretation of laparoscopic exercises in natural languags. For the acquisition of the procedural knowledge, we propose the use of a tool that allows translating natural textual explanations to different knowledge representation settings, showing the objects that are involved in an exercise, their constraints and other relationships. We also extend the exercise by including possible (but incorrect) actions, such that we can distinguish three kinds of actions: impossible actions (e.g., a bead cutting the instrument), possible but incorrect actions (e.g., taking the wrong bead) and possible and correct action (e.g., taking the right bead).

Previous work in the area of surgical training has been able to process the behaviour from video executions of exercises [8]. However, the high-level descriptions provided in natural

language from medical experts are not used in the process of identifying the events and monitoring the apprentice.

This information provides a valuable source of additional domain information, which could enrich the ability of the system to provide better explanations of the user behaviour and potentially assist them in correctly completing the task. To address this problem, in this paper, we analyse the processing of the high level descriptions so that it could be used to complete the process of acquiring knowledge from video records. To achieve that, we adapt a model learning approach first proposed in [15, 16]. It extracts the knowledge base needed to describe the problem domain from textual instructions and later uses this knowledge to generate rules describing the possible user behaviour.

The contribution of this paper is twofold: (1) we present an approach to extracting the domain knowledge from textual instructions in the form of situation model and then generating rules, describing the possible correct execution sequences based on this knowledge; and (2) we apply the approach to the domain of minimally invasive surgery, and analyse the results when using the precise description of the task versus the extension of the task with other possible (but incorrect) actions.

The paper is structured as follows. In Section "Related Work" we discuss the state of the art in generating models from narratives and in supervising minimally invasive surgery training. We then outline our approach to knowledge extraction and rules generation from textual narratives in Section "Extracting Knowledge from the Text". We illustrate the approach with an example from the training exercises for minimally invasive surgery (Section "An Example") and we compare the model generated from correct description and possible but incorrect descriptions (Section "Comparison to Extended Model"). We conclude the paper with a short discussion and conclusion.

RELATED WORK

There are various approaches for extracting models of human behaviour from textual descriptions: through grammatical patterns that are used to map the sentence to a machine understandable model of the sentence [1]; through machine learning techniques [11, 3]; or through reinforcement learning approaches that learn language by interacting with an external environment [1]. Models learned through model grounding have been used for plan generation [1], for learning the optimal sequence of instruction execution [2], for learning navigational directions [3], and for interpreting human instructions for robots to follow them [12].

When learning models from textual narratives, it is important to identify the relevant domain knowledge and the semantic structure of this knowledge [15]. We call this collection of knowledge a situation model. It is the basis for building rules, describing the possible correct behaviour [15]. Existing approaches that learn human behaviour from text, however, make simplifying assumptions about the problem, making them unsuitable for more general activity recognition problems. Most of them rely on a manually defined situation model, or do not use one. However, one needs a situation model to deal with model generalisation problems and as a means for expressing the semantic relations between model elements.

An important aspect of the situation model is discovering causal relations between actions. Causal relations are important, because they provide information about the possible "causally" correct execution sequences a person can make. They are the basis for defining the rules describing the "allowed" execution sequences of the actions. However, in existing approaches the rules are learned through explicit causal relations, that are grammatically expressed in the text [11]. They however, either rely on initial manual definitions to learn these relations [1], or on grammatical patterns and rich texts with complex sentence structure [5]. They do not address the problem of discovering causal relations between sentences, but assume that all causal relations are expressed within the sentence [12]. They also do not identify implicit relations. To find causal relations in instructions without a training phase, however, one has to rely on alternative methods, such as time series analysis [13].

In previous works, a new approach is able to learn planning models of human behaviour from textual instructions [15, 16]. These works have shown that the models can be applied to the domain of everyday activities and especially to cooking activities. In this paper we show that it is possible to use the proposed approach to generate rules for the domain of minimally invasive surgery.

In minimally invasive surgery, the surgeon students are required to spend long periods of training. This training is often limited by the lack of access to realistic testbeds and to immediate feedback and guidance from experts during each exercise [9]. To address this problem, some works propose the introduction of virtual surgery simulators [10]. However, the metrics used in these virtual environments only evaluate but do not supervise [7]. Although new approaches are making it possible to detect wrong gestures and immediately suggest gesture corrections during a surgical training [7], these supervision systems are very low level, quite rigid and inefficient. To our knowledge, the use of the description of the exercise in natural language is something that has not been incorporated into these systems, which is what we study in this paper.

In our previous works, the Planning Domain Definition Language (PDDL) has been used as the targeted planning language [14]. In this work, we discuss the mapping of the situation model to event calculus as it has been shown that event calculus is suitable for describing the activities of the surgeons in training [8].

EXTRACTING KNOWLEDGE FROM THE TEXT

In this section we describe the way the information is extracted from the text that explains to the surgery apprentices how an exercise has to be done.

Identifying text elements of interest

To extract the text elements that describe the expected behaviour, the actions and their relations to other entities in the environment have to be identified. This is achieved through assigning each word in the narrative the corresponding part

of speech (POS) tag. Furthermore, the dependencies between text elements are identified through dependencies parser. To identify the human actions, the verbs from the POS-tagged text are extracted. We are interested in present tense verbs, as textual descriptions are usually written in present tense, imperative form. We also extract any nouns that are direct objects to the actions. These will be the objects in the environment with which the human can interact. Furthermore, we extract any nouns that are in conjunction to the identified objects. These will have dependencies to the same actions, to which the objects with which they are in conjunction are dependent.

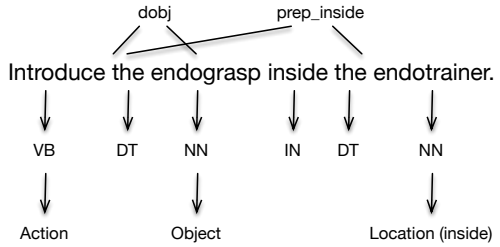


Figure 1. Elements of a sentence necessary for the model learning.

Figure 1 gives an example of a sentence and the identified elements based on the POS-tags and dependencies.

Moreover, any preposition relations such as *in*, *on*, *at*, etc. between the objects and other elements in the text are identified. These provide spatial or directional information about the action of interest. For example, in the sentence *"Introduce the endograsp inside the endotrainer."* our action is *introduce*, while the object on which the action is executed is *endograsp*. The action is executed inside the location *endotrainer* identified through the *inside* preposition. Finally, we extract "states" from the text. The state of an object is the adjectival modifier or the nominal subject of an object. As in textual instructions the object is often omitted (e.g. *"Move (the endograsp) to the left corner."*), we also investigate the relation between an action and past tense verbs or adjectives that do not belong to an adjectival modifier or to nominal subject, but that might still describe this relation. The states give us information about the state of the environment before and after an action is executed.

Extracting causal relations from textual instructions

To identify causal relations between the actions, and between states and actions, we use an approach proposed in [13]. It transforms every word of interest in the text into a time series and then applies time series analysis to identify any causal relations between the series. More precisely, each sentence is treated as a time stamp in the time series. Then, for each word of interest, the number of occurrences it appears in the sentence is counted and stored as element of the time series with the same index as the sentence index. We generate time series for all actions and for all states that change an object. To discover causal relations based on the time series, we apply the Granger causality test. It is a statistical test for determining whether one time series is useful for forecasting another. More precisely, Granger testing performs statistical significance test for one time series, "causing" the other time series with different time lags using auto-regression [4].

The causality relationship is based on two principles. The first is that the cause happens prior to the effect, while the second states that the cause has a unique information about the future values of its effect. Given two sets of time series x_t and y_t, we can test whether x_t Granger causes y_t with a maximum p time lag. To do that, we estimate the regression $y_t = a_o + a_1 y_{t-1} + ... + a_p y_{t-p} + b_1 x_{t-1} + ... + b_p x_{t-p}$. An F-test is then used to determine whether the lagged x terms are significant.

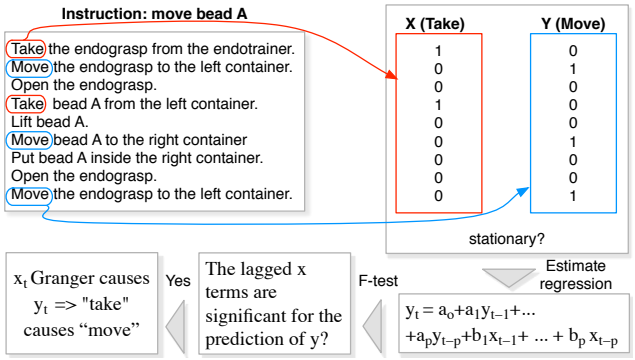


Figure 2. The procedure for discovering causal relations.

Figure 2 shows the procedure of converting text elements into time series and using them to discover causal relations.

Building the situation model

The situation model, as the name suggests, contains information about the situation in which the task is being executed [14]. It describes the objects, locations, and any elements in the environment that are taken as arguments in the actions as well as the abstraction levels of the elements. Furthermore, it contains the actions with their causal relations. The situation model also contains any relations between the actions and the elements in the environment, such as those describing which action is executed on which objects, in what location, through what means, etc. Example of a situation model can be seen in Figure 3.

To learn the abstraction levels of the elements in the environment, a semantic lexicon (e.g. WordNet [6]) is used to build the initial ontology. As the initial ontology does not contain some types that unify arguments applied to the same action, the ontology has to be extended. To do that, the prepositions with which actions are connected to indirect objects are also extracted (e.g. *in*, *on*, etc.). They are then added to the initial ontology as relations of the arguments they connect. In that manner, the locational properties of the arguments are described (e.g. *endograsp* has the property to be *inside* something). Later, the actions found in the text and the causal relations discovered with the Granger causality test are added to the situation model.

Generating Event Calculus Rules

The next step in the process is the generation of rules that describe the actions and the way they change the world. We then use these rules to reason about the possible correct exercise execution. In previous works we have used PDDL (Planning

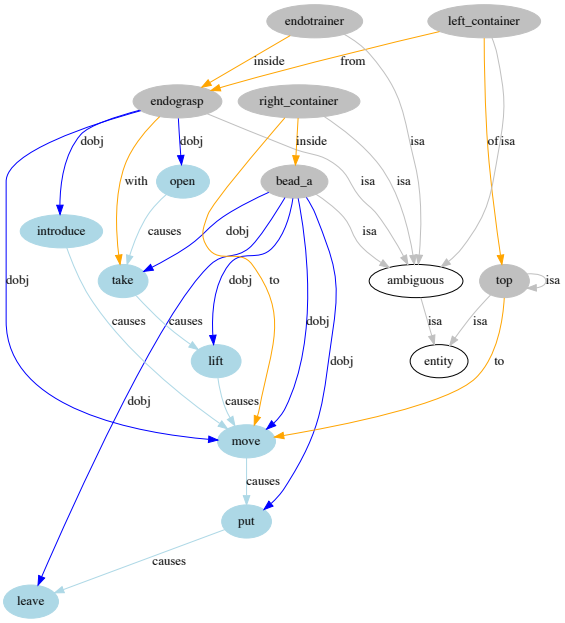


Figure 3. Situation model for moving a bead from one container to another.

Domain Definition Language) as the targeted format [14, 15, 16]. In this work, however, we show that it is possible to map the situation model to event calculus, as it has already been shown that event calculus is suitable for reasoning in the context of minimally invasive surgery exercises [8]. The basis for the rules is the situation model. For example, Figure 4

```
initiates(take(O,O1), done_take(O,O1), T+1) :- holdsAt(done_open(O), T).
terminates(lift(O), done_take(O), T).
```

Figure 4. Example of the rules for actions *take* and *left* in event calculus.

shows the rules for changing the state of the fluent *done_take* through the execution of the actions *take* and *lift*. The action *take* changes the state of the fluent *done_take* to true if the fluent *done_open* is true at this time point. Then the action *lift* terminates the fluent *done_take*.

We take the action name from the list of actions in the situation model, while the two parameters *(O,O1)* are taken from the arguments in the situation model to which the action is connected.

Furthermore, we consider the relations in each sentence in order to identify fluents affected by the actions, which cannot be identified directly in the situation model. For example, in the sentence "Introduce the endograsp inside the endotrainer." we have the relation (or fluent) "inside(endograsp, endotrainer)" which is set to true through the execution of the action "introduce". From the situation model, we can infer that the action "introduce" is executed on the object "endograsp" but not that it changes the "inside" relation. However, if we consider the narrative sentence-wise, then we see that the "inside" relation appears together with "introduce". Thus "introduce" changes the value of "inside". In a sense, this process builds micro situation models for each sentence in the narrative.

To determine which action terminates the fluents activated by a given action, we use the causal relations discovered in the text. For example, in Figure 3, the action *take* causes *lift*. For that reason, in our rules we define that the execution of lift changes the state of the fluent *done_taken* to false. On the other hand, as the action *open* causes *take*, we define that in order to initiate the action *take*, the fluent *done_open* has to be true.

AN EXAMPLE

To illustrate our approach, we take a textual instruction describing how to move a bead from one container to another. Table 1 shows the instruction.

Table 1. Example narrative.

```
1 Introduce the endograsp inside the endotrainer.
2 Move the endograsp to the top of the left_container.
3 Open the endograsp.
4 Take the bead_A with the endograsp from the left_container.
5 Lift the bead_A.
6 Move the bead_A to the right_container.
7 Put the bead_A inside the right_container.
8 Open the endograsp.
9 Leave bead_A inside the right_container.
```

From the text in Table 1, with the help of our approach we were able to extract six entities (or objects) and seven actions. The objects were abstracted into a hierarchy with three levels. After applying the Granger causality test, six causal relations were discovered. Apart from them, eight action – direct object relations and seven relations describing direct object / verb – indirect object were discovered. Figure 3 shows the situation model for the text in Table 1. There, light blue shows the actions, grey the objects, and white is the abstraction hierarchy. The blue connectors represent direct object – verb relations, and the orange connectors – direct object / verb – indirect object relations. The grey connectors represent the "is-a" relation in the abstraction hierarchy.

Based on the situation model we then generated event calculus model representing the knowledge from the situation model.

From the objects (with grey in Figure 3) and the actions (with light blue in Figure 3) in the situation model, we extracted the list of objects and actions in our event calculus model (see Table2).

Table 2. The identified objects and actions, represented in event calculus.

```
1 object(top;right_container;left_container;
2 endotrainer;endograsp;bead_a)
3
4 event(take;move;put;open;lift;leave;introduce)
```

Furthermore, we generated three types of fluents: (1) "done-action" which indicates that an action is being executed; (2) fluents describing the type of the object (e.g. "ambiguous-object"), which are extracted from the abstraction hierarchy

(with white in Figure 3) and the type of prepositions with which the indirect objects are related to direct objects or verbs; (3) fluents describing relations between two objects (e.g. "inside(object1,object2)" indicates that object1 is inside object2). Table 3 shows the identified fluents.

Table 3. The identified fluents, represented in event calculus.

```
fluent(done-take(O1,O2); done-move(O1,O2); done-put(O);
done-open(O); done-lift(O); done-leave(O);
done-introduce(O1,O2); region-object(O);
ambiguous-object(O); from-object(O);
inside-object(O); of-object(O); to-object(O);
with-object(O); inside(O1,O2); to(O1,O2))
```

We then generated the rules that describe how the fluents are changed through the execution of the actions. Table 4 shows an example of the rules for the fluents "inside" and "done-introduce". "inside(O1,O2)" is activated through the execution of the action "introduce". We extracted this information from the first sentence of the text. Then based on the situation model, we added that for this to happen, O1 has to be "ambiguous-object" and O2 has to be an "inside-object". The same was done for the "done-introduce" fluent, which when active indicates the execution of the action "introduce". To

Table 4. An example of the axioms for the fluent "inside" and "done-introduce", represented in event calculus.

```
initiates(intorduce(O1,O2),inside(O1,O2),T) :-
holdsAt(ambiguous-object(O1),T),
holdsAt(inside-object(O2),T).

initiates(intorduce(O1,O2),done-introduce(O1,O2),T) :-
holdsAt(ambiguous-object(O1),T),
holdsAt(inside-object(O2),T).

terminates(move(O1,O2),done-introduce(O1,O2),T) :-
holdsAt(ambiguous-object(O1),T),
holdsAt(to-object(O2),T).

terminates(move(O1,O2),inside(O1;O2),T) :-
holdsAt(ambiguous-object(O1),T),
holdsAt(to-object(O2),T).
```

identify the action, through which the fluents are terminated, we used the causal relations in the situation model. In this case, "introduce" causes "move", so we assume that the action "move" will make the fluents "inside" and "done-introduce" false.

After generating the rules for the execution of all actions, we then generated the initial state of the model. There we actually defined the properties of the objects through the "property-object" fluent (e.g. "ambiguous-object(O)"). These are static properties, which do not change during the execution of the actions. The reason for modelling them as fluents instead of as types is that one object can have multiple roles (e.g. "endotrainer" can be both "ambiguous-object" and "inside-object"). In that manner we restrict the objects to which a given rule applies.

COMPARISON TO EXTENDED MODEL

To compare the difference in the knowledge between narrative that describes only the correct goal oriented execution of the exercise and one that is possible in the domain but incorrect with respect to the exercise, we also generated a situation model from narrative that has 35 sentences (in comparison to the 8 in the original narrative). We call the situation model generated from the original narrative SM_o and the one generated from the extended narrative SM_e. The extended narrative contains the execution sequence described in the original text plus some additional moves and objects that are irrelevant for the completion of the exercise.

Figure 5 shows SM_e. It can be seen that the model contains much more domain knowledge than SM_o. More precisely, the model has 8 objects (in difference to 6 in SM_o), 11 actions (in difference to 7 in SM_o), and 34 semantic relations (in difference to 15 in SM_o). Interestingly, however, SM_e has less causal relations than SM_o (4 in SM_e and 6 in SM_o). This means that despite the richer domain knowledge, the smaller number of causal relations produces relaxed model that allows much more execution sequences to be valid in the given domain. In other words, the model will be able to also explain sequences that are plausible in the domain but that do not represent the exercise.

DISCUSSION AND CONCLUSION

In this work we presented an approach that uses experts' instructions in natural language to generate situation models that are later used to automatically create event calculus rules for automated supervision system for minimally invasive surgery training. These rules are later used to reason about the way in which the trainees execute the exercises and to potentially detect errors in the behaviour.

In the future, we want to explore the model generated from the extended narrative, as well as the means for reasoning about the erroneous task execution. This could be done by assigning higher weight or "correct" flag to execution sequences that conform with the description in the original model, and less weight or "incorrect" flag to the remaining execution sequences. This will still make them possible in the domain but will provide a mechanism for detecting errors in the training.

Another aspect we will investigate in the future is the mapping of the high level event calculus rules to the low level events detected by the video-based activity recognition system.

REFERENCES

1. S. R. K. Branavan, N. Kushman, T. Lei, and R. Barzilay. 2012. Learning High-level Planning from Text. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. 126–135. http://dl.acm.org/citation.cfm?id=2390524.2390543

2. S. R. K. Branavan, L. S. Zettlemoyer, and R. Barzilay. 2010. Reading Between the Lines: Learning to Map High-level Instructions to Commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1268–1277. http://dl.acm.org/citation.cfm?id=1858681.1858810

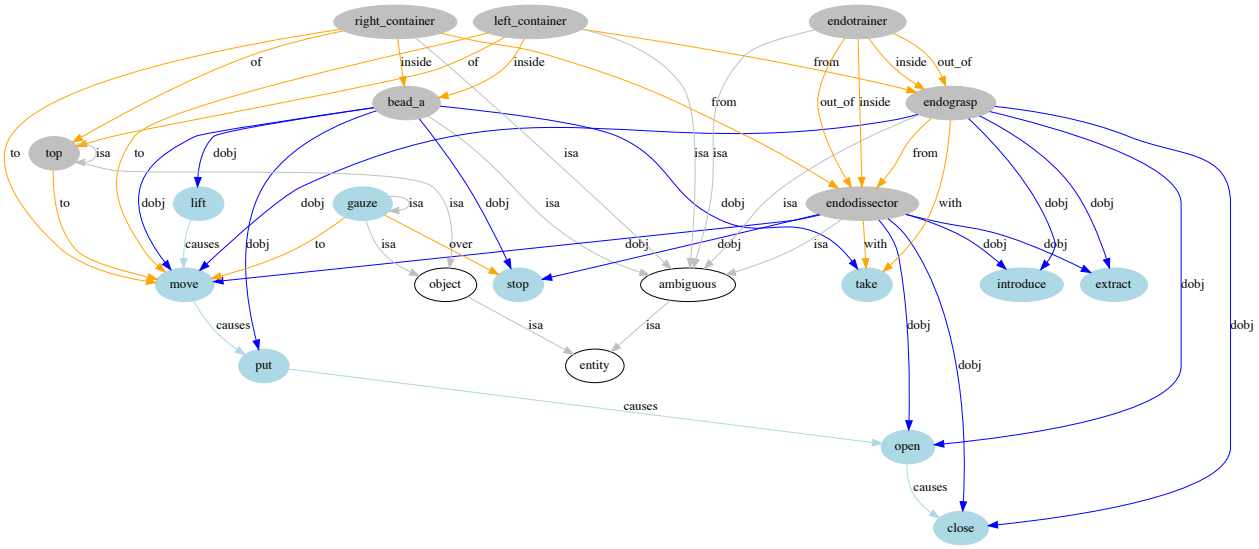


Figure 5. The extended situation model for moving a bead from one container to another.

3. David L. C. and Raymond J. M. 2011. Learning to Interpret Natural Language Navigation Instructions from Observations. In *Proc. of the AAAI Conference on Artificial Intelligence*. 859–865. http://www.cs.utexas.edu/users/ai-lab/?chen:aaai11

4. C. W. J. Granger. 1969. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica* 37, 3 (August 1969), 424–438. DOI: http://dx.doi.org/10.2307/1912791

5. Xiaochen L., Wenji M., Daniel Z., and Fei-Yue W. 2010. Automatic construction of domain theory for attack planning. In *Int. Conference on Intelligence and Security Informatics*. 65–70. DOI: http://dx.doi.org/10.1109/ISI.2010.5484775

6. G. A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. DOI: http://dx.doi.org/10.1145/219717.219748

7. C. Monserrat, A. Lucas, J. Hernández-Orallo, and M. José Rupérez. 2014. Automatic supervision of gestures to guide novice surgeons during training. *Surgical Endoscopy* 28, 4 (2014), 1360–1370. DOI: http://dx.doi.org/10.1007/s00464-013-3285-9

8. Dolz J.F. Rupérez M.J. Flach P Monserrat C., Hernández-Orallo J. 2016. Knowledge Acquiaisiont by Abduction for Skills Monitoring: Application to Surgical Skills. In *26th International Conference on Inductive Logic Programming*.

9. M. C. Porte, G. Xeroulis, R. K. Reznick, and A. Dubrowski. 2007. Verbal feedback from an expert is more effective than self-accessed feedback about motion efficiency in learning new surgical skills. *The American Journal of Surgery* 193, 1 (2007), 105 – 110. DOI: http://dx.doi.org/10.1016/j.amjsurg.2006.03.016

10. R. M. Satava. 1993. Virtual reality surgical simulator - The first steps. *Surgical Endoscopy* 7, 3 (5 1993), 203–205. DOI:http://dx.doi.org/10.1007/BF00594110

11. A. Sil and E. Yates. 2011. Extracting STRIPS representations of actions and events. In *Recent Advances in Natural Language Processing*. 1–8.

12. M. Tenorth, D. Nyga, and M. Beetz. 2010. Understanding and executing instructions for everyday manipulation tasks from the World Wide Web. In *Int. Conference on Robotics and Automation*. 1486–1491. DOI: http://dx.doi.org/10.1109/ROBOT.2010.5509955

13. K. Yordanova. 2015. Discovering Causal Relations in Textual Instructions. In *Recent Advances in Natural Language Processing*. 714–720.

14. K. Yordanova. 2017a. Automatic Generation of Situation Models for Plan Recognition Problems from Textual Instructions. In *Proceedings of Recent Advances in Natural Language Processing*. Varna, Bulgaria. submitted.

15. K. Yordanova. 2017b. TextToHBM: A Generalised Approach to Learning Models of Human Behaviour for Activity Recognition from Textual Instructions. In *Proceedings of the AAAI Workshop on Plan, Activity and Intent Recognition (PAIR)*. AAAI, San Francosco, USA.

16. K. Yordanova and T. Kirste. 2016. Learning Models of Human Behaviour from Textual Instructions. In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016)*. Rome, Italy, 415–422. DOI: http://dx.doi.org/10.5220/0005755604150422