UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

GrayAR
Técnicas de visión por computador y realidad aumentada aplicadas a la gestión documental

Manuel Hervás Ortega

Febrero, 2015

GRAYAR

TÉCNICAS DE VISIÓN POR COMPUTADOR Y REALIDAD AUMENTADA APLICADAS A LA GESTIÓN DOCUMENTAL

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE
TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

GrayAR
Técnicas de visión por computador y realidad aumentada aplicadas a la gestión documental

Autor: Manuel Hervás Ortega

Director: Dr. Carlos González Morcillo

Febrero, 2015

Manuel Hervás Ortega

Ciudad Real – Spain

E-mail: hervas.manuel@gmail.es
Teléfono: 676 834 017
Web site: `http://argos-project.com`

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

Las Interfaces Naturales de Usuario surgen como consecuencia de la evolución en la forma que tenemos para interaccionar con un computador. Su objetivo es lograr una tecnología lo más natural e intuitiva posible, de modo que el propio uso del ordenador pase desapercibido.

El potencial de la Realidad Aumentada para construir interfaces naturales es evidente. Su uso va más allá de juegos y entretenimiento; las grandes compañías informáticas están presentando novedosos sistemas que explotan sus características. Aunque actualmente la mayoría de aplicaciones utilizan pantallas o gafas como dispositivos de representación, la utilización de proyectores permite desarrollar innovadoras *«ventanas de información»* sobre cualquier superficie o dispositivo.

La aplicación de técnicas de Realidad Aumentada en el ámbito de los Interfaces Naturales permite modelar entornos interactivos que proporcionen una sensación inmersiva al usuario sobre elementos existentes del entorno físico.

El presente Trabajo Fin de Grado (con acrónimo GrayAR) surge como parte del *Proyecto ARgos*, cuyo objetivo es el uso de técnicas de Realidad Aumentada como ayuda a la gestión documental. El proyecto plantea la construcción de un dispositivo embebido que dé soporte a la detección de documentos y su posterior tratamiento empleando técnicas de realidad aumentada mediante un entorno natural e interactivo.

GrayAR se encarga de la detección e identificación de documentos, así como del cálculo de su posicionamiento dentro del espacio 3D. El sistema emplea el paradigma de *«pantalla táctil»*, utilizando el propio documento impreso como superficie interactiva. En GrayAR se ha elaborado una herramienta genérica que permite tanto el calibrado de cámaras y proyectores en el espacio 3D, como la representación de información aumentada directamente sobre el espacio físico. Además se ha construido un prototipo hardware sobre el que se han definido diferentes escenarios de explotación por la Asociación ASPRONA (Asociación para la Atención a Personas con Discapacidad Intelectual y sus Familias de la Provincia de Albacete).

Abstract

Natural User Interfaces arise from changes in the way that we have to interact with a computer. Its aim is to achieve the most natural and intuitive technology possible, so that the use of the computer itself goes unnoticed.

The potential of Augmented Reality to build natural interfaces is evident. Its use goes beyond gaming and entertainment; main computer companies are introducing new systems that exploit its features. Although most applications currently used screens or glass display devices, the use of projectors allows developing innovative «information windows» on any surface or device.

Application of Augmented Reality techniques in the field of Natural Interfaces allows modeling interactive environments that provide an immersive sensation to the user on existing elements of the physical environment.

This end-of-degree project (named GrayAR) comes as part of the *ARgos Project* whose goal is the use of Augmented Reality techniques as an aid to document management. The project involves the construction of an embedded device that supports document image recognition and its management using augmented reality techniques through a natural and interactive environment.

GrayAR is responsible for the detection and identification of documents and calculating their position within the 3D space. The system uses the paradigm of «touch screen», using the printed document as an interactive surface. In GrayAR has developed a generic tool that allows calibration of cameras and projectors in 3D space, such as increased representation of information directly on the physical space. It has also built a prototype hardware on which they have defined different operational scenarios for ASPRONA Association (Association for the Care of Persons with Intellectual Disabilities and their Families of the Province of Albacete).

Agradecimientos

En primer lugar, quiero dar las gracias a mi familia por el apoyo incondicional que he recibido a lo largo de todos estos años. En especial a mis padres, que me han dado todo lo que soy ahora.

Rosa, gracias por ser mi amiga, mi compañera de clase, mi novia, mi esposa. Gracias por tantos años juntos, en lo bueno y en lo malo.

Paula, gracias por esas sonrisas cuando llego a casa.

Santiago, gracias por los buenos ratos pasados en el Grupo AIR.

A la gente del Grupo ARCO, muchas gracias por vuestra compañía y amistad.

Juan Carlos, gracias por tu interés y apoyo en el proyecto ARgos.

Carlos, muchas gracias por tu amistad, y por darme la oportunidad de dar un giro a mi vida aquel día en el andén de la estación.

Manuel

A Rosa

Índice general

Índice de cuadros

Índice de figuras

Índice de listados

Listado de acrónimos

CLI	Command Line Interface
GUI	Graphical User Interface
HTML	HyperText Markup Language
SDK	Software Development Kit
RGB	Red - Green - Blue
GPS	Global Position System
HDM	Head Mount Display
ARM	Advanced RISC Machine
SBC	Single Board Computer
RAM	Random Access Memory
USB	Universal Serial Bus
OCR	Optical Character Recognition
IR	Infrared
XML	eXtensible Markup Language
KISS	Keep It Simple, Stupid!
ARCO	Arquitectura y Redes de COmputadores

Capítulo 1

Introducción

DESDE la introducción del ordenador personal en la década de los años 80, los esfuerzos por naturalizar la experiencia de la interacción ha estado limitada por el factor de adaptación de una persona a los distintos entornos informáticos. A día de hoy, todavía conservamos el mismo esquema de utilización de un ordenador de sobremesa de principios de los 80, con computadores que tienen prácticamente la misma configuración de monitor, teclado y ratón. Realmente no hay nada de natural en este tipo de experiencia de usuario, y en todo caso es lo más alejado de ser una función humana. Para apoyar aún más lo lejos que nos hemos desviado de esta naturaleza, el estudio de la ergonomía surgió como una manera de minimizar en nuestros cuerpos el riesgo de una lesión, ya que tratamos de adaptarlos a este entorno no natural.

Afortunadamente vivimos en una época en la que estamos rompiendo con los paradigmas tradicionales establecidos. La introducción de nuevas formas de relacionarnos con los ordenadores esta haciendo que vean la luz dispositivos portátiles, superficies multitáctiles, proyecciones interactivas,... donde estamos dejando las limitaciones de los entornos virtuales y metáforas obsoletas. Ya es hora de que volvamos a nuestro mundo natural.

1.1 Antecedentes históricos

Los computadores hasta los años 70, no eran mucho más que grandes calculadoras utilizadas para automatizar cálculos complejos en instituciones militares y para investigación de alto nivel en unas pocas universidades. Las primeras interfaces consistían en introducir una serie de tarjetas perforadas que contenían el programa codificado y más tarde revisar en un impresora el resultado de la ejecución.

Posteriormente se desarrollaron las **interfaces de línea de comandos (CLI)**, así el usuario ya no tuvo que perforar tarjetas ni tenía que esperar. Los programadores y usuarios disponían de acceso a un terminal, y lo usaban para interactuar directamente con el ordenador en algo parecido a lo que podemos denominar *«tiempo real»*. El usuario podía introducir una orden, y obtener una salida textual de vuelta más o menos inmediata.

Fue a principios de los años 80, cuando la informática dio un gran salto mediante la **interfaz gráfica de usuario (GUI)**, que era una forma más natural y convincente para interactuar

con un ordenador. Esto hizo que la informática fuese más accesible para la gente, dándole la capacidad de comunicarse con el ordenador a base de metáforas de escritorio, ventanas y punteros de ratón.

Pero en los 30 años desde el lanzamiento del primer ordenador en ofrecer una interfaz gráfica de usuario en una máquina de bajo coste, la experiencia física de la interacción no ha sufrido muchos cambios en lo que se refiere a ordenadores personales. En este tiempo se inventó la Web, progresó desde su primera iteración hasta el actual HTML 5, desarrollamos multitud de revolucionarias plataformas web para conectar y relacionarnos por todo el mundo, pero **mantenemos nuestra rígida adhesión al legado del monitor, ratón y teclado** a principios de 2010 como lo fue en la década de los 80.

1.2 Interfaces naturales de usuario

No es hasta finales de los años 2000, cuando podemos decir que empieza la ***«era post-PC»***. La generalización de los dispositivos multi-touch y la informática móvil marca uno de los mayores hitos en las interfaces persona-computador. Los dispositivos móviles se diseñan para ser ligeros, portátiles y encajar con nuestro estilo de vida personal. De repente, estamos en medio de una de las más importantes revoluciones tecnológicas, cuando nuestros dispositivos informáticos comienzan a adaptarse a nuestras funciones naturales humanas.

El primer paso en este ámbito ha sido la amplia adopción de la tecnología móvil inteligente en las funciones cotidianas, como por ejemplo, el control de una agenda con los planes del día o saber a dónde ir, independientemente del sentido de orientación que tengamos. El conocimiento aparece en el momento que sacamos el smartphone en busca de las respuestas.

Apenas estamos comenzando a descubrir las posibilidades de un mundo en el que los dispositivos se adaptan a nuestras conductas y la tecnología apoya y amplifica nuestras funciones naturales. Estaremos en el camino de lo que podemos llamar ***«computación humana»***, cuando el hardware desaparezca y nos pase desapercibido que realmente estamos interaccionado con un ordenador. Esto ocurre cuando la tecnología se integra discretamente en objetos cotidianos o en funciones naturales. La tecnología será la que se adapte a nosotros, en lugar de ser nosotros quienes se adapten a la computación y la máquina aprenda a reconocer e interpretar los patrones humanos para producir una salida basada en un contexto familiar.

Un ejemplo de una tecnología no relacionada con la computación que ha madurado con el tiempo, es la optometría. Basta con pensar en las lentes de contacto. Se coloca una capa fina directamente en nuestra córnea, alterando así los rayos de luz que convergen perfectamente en nuestra retina. Al instante, sin apenas esfuerzo, tenemos una visión perfecta. Una vez colocadas tendemos a olvidar que las llevamos puestas y esto se debe a su perfecta integración en nuestro estilo de vida cotidiana.

Para reflexionar sobre el estado actual de la informática en relación a la optometría: imaginemos que dependiésemos de un teclado y un ratón para modificar nuestra vista cada vez que necesitásemos enfocar.

Interfaz natural de usuario es un término genérico para una variedad de tecnologías que permiten a los usuarios interactuar con los ordenadores en términos humanos. Algunas de estas tecnologías son las basadas en visión por computador y que son capaces desde interpretar expresiones naturales como gestos hasta proporcionar información contextual que se proyecta dentro del campo de visión del usuario, como pretenden las Google Glass.

Las interfaces de usuario basadas en visión por computador llevan varios años desarrollándose y comercializándose con éxito dentro de la industria del videojuego. Mediante un dispositivo externo permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, reconociendo gestos, comandos de voz, objetos e imágenes.

El EyeToy es un cámara que fue lanzada en octubre de 2003 para la PlayStation 2. Este dispositivo utiliza visión por computador y reconocimiento de gestos para procesar las imágenes adquiridas por la cámara y permite a los jugadores interactuar con los juegos.

La estética de los juegos que soportaban este sistema consistían en la introducción del jugador en la pantalla mediante realidad mixta y mediante la realización de determinados gestos, sea el personaje en el que se basa la historia del juego.

Este tipo de interación supuso una revolución en la forma de interactuar con los videojuegos, vendiendo más de 10 millones de unidades por todo el mundo, y ha sido precursor de otros dispositivos como Kinect.

Kinect es un sensor que se vendió previamente como un periférico opcional, por 150 euros, para la videoconsola Xbox 360 y posteriormente para Xbox One y Windows, debido al gran

Figura 1.1: EyeToy para PlayStation 2.

Figura 1.2: EyeToy Play 3 con los jugadores inmersos en el juego.

Figura 1.3: Utilización de Kinect para la visualización de radiografías durante una cirugía.

éxito obtenido. Según datos de Microsoft, a principios de 2013 se habían comercializado más de 24 millones de Kinect.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad y un vector de micrófonos que le permite distinguir extremidades y los movimientos realizados con ellas, expresiones faciales, gestos con las manos y los dedos, incluso el pulso del usuario, y utilizarlos para controlar e interactuar con la consola. Gracias a los micrófonos es capaz de reconocer la dirección del origen del sonido y proporcionar cancelación de ruido.

Estas interesantes características motivaron a la comunidad de usuarios para obtener una solución de código abierto y utilizar el dispositivo en cualquier ordenador y no únicamente en la consola de videojuegos. Varios días después de su lanzamiento, se publicó el primer controlador para ordenador obtenido mediante ingeniería inversa. Esto dio paso a la creación de varias APIs libres que han permitido el uso del Kinect de forma no oficial más allá de lo que en un principio se hubiese esperado, impulsando el interés en explorar nuevas modalidades de interacción. Finalmente Microsoft liberó un SDK para desarrollo de aplicaciones con el dispositivo en Windows.

1.3 Realidad Aumentada

La realidad aumentada se podría considerar como la aplicación de distintas técnicas de visión por computador, mediante la cual la percepción del mundo real se complementa con información adicional generada por ordenador en tiempo real. Esta información adicional

Figura 1.4: Previsualizacion 3D mediante realidad aumentada en arquitectura

puede ser desde etiquetas virtuales, representaciones de modelos tridimensionales, o incluso cambios de iluminación.

La realidad aumentada se puede aplicar a prácticamente todos los campos. En el sector industrial, para la reparación y mantenimiento de máquinas e instalaciones complejas, visualización de datos o simulación. En aplicaciones médicas, mostrarían la situación de órganos no visibles durante una cirugía. También se han realizado aplicaciones para diseño de interiores, presentaciones de productos, educación, publicidad, turismo, arte y ocio.

Hoy en día, la mayoría de las aplicaciones de realidad aumentada se centran en la movilidad. De este modo, las pantallas de dispositivos portátiles y los smartphones se han convertido en la opción dominantes para la visualización. Sin embargo, el aumento de las prestaciones y la reducción de los costes hacen que los proyectores se hayan popularizado y establecido como herramientas habituales para la visualización. La capacidad de generar imágenes mucho mayores que el propio dispositivo prácticamente en cualquier lugar es una característica interesante para muchas aplicaciones que no pueden ser mostradas en pantallas convencionales.

Existen actualmente muchas líneas de investigación sobre este tema, en las que se pretende aplicar este potencial mediante la utilización de los proyectores de una manera poco convencional y desarrollar nuevas e innovadoras pantallas de información que van más allá de presentaciones en las típicas pantalla planas.

Los **enfoques basados en proyectores** combinan las ventajas de la realidad virtual y la realidad aumentada proporcionando sensaciones inmersivas que se pueden realizar sobre entornos cotidianos, sin la necesidad de pantallas de proyección especiales y configuraciones de pantalla dedicadas. Para muchas aplicaciones, esto requiere la perdida de la movilidad,

Figura 1.5: *Projection mapping* realizado sobre la catedral de Santiago en 2011

pero no necesariamente de la portabilidad. Otras aplicaciones, sin embargo, no requieren movilidad y más bien se benefician de las propiedades aumentadas que proporciona la proyección. Los ejemplos van desde entretenimiento educativo en los museos, con proyecciones sobre paredes o sobre las propias obras de arte, hasta proyecciones en fachadas de edificios históricos para conseguir efectos de movimiento ó 3D, dando lugar a un espectáculo artístico conocido como ***projection mapping***.

1.4 Motivación

El presente proyecto se enmarca dentro de la Cátedra Indra-UCLM, en el proyecto ***«ARgos: Sistema de Ayuda a la Gestión Documental basado en Visión por Computador y Realidad Aumentada»*** que tiene como objetivo la construcción de un sistema de ayuda a la gestión de documentos, basado principalmente en visión por computador y síntesis visual y auditiva en el espacio físico, empleando técnicas de realidad aumentada.

Todos los países de la Unión Europea aceptan las recomendaciones generales de la Organización Mundial de la Salud así como las directrices y programas de las Naciones Unidas relativas a las personas con necesidades especiales y que proponen expresamente *«su participación plena en la vida social, con oportunidades iguales a las del resto»*. El proyecto esta pensado para facilitar la integración laboral, sean cuales sean las necesidades especiales de las personas que tenga que gestionar documentación impresa.

1.5 Impacto socio-económico

Bajo el concepto de «discapacidad», se incluyen limitaciones muy diversas que afectan con mayor o menor gravedad las facultades que son habituales para desenvolverse en la vida cotidiana y que no tienen por qué impedir una inserción social y laboral normalizada. A día de hoy, las personas con discapacidad pueden desempeñar las funciones laborales, normalmente, gracias a la ayuda de aparatos o de procesos específicos de adaptación del puesto de trabajo. E incluso, una discapacidad total, no significa que el sujeto no pueda suplir o compensar su limitación mediante el uso de otras facultades.

Según un estudio [dT13] realizado por la Organización Internacional del Trabajo (OIT) se identifican varios tipos de beneficios socio-económicos en la inclusión de personas con discapacidad al mundo laboral.

La persona con discapacidad que trabaja, se convierte en un aporte para la economía del hogar, ya sea porque aporta ingresos o porque autofinancia sus necesidades. Esto disminuye la precariedad de recursos a los que se suelen enfrentar. Al mismo tiempo, **incrementa su autonomía respecto de terceros**, como su propia familia o instituciones de apoyo. Como consecuencia, **la familia gana tiempo y ahorra recursos** que antes destinaba al cuidado o manutención de esa persona.

En el estudio se señala que el sólo hecho de **incluir a las personas con discapacidad en la empresa, genera un efecto motivador en otros trabajadores**, causa sentimientos de orgullo respecto de la empresa y hace sentir que ésta es un mejor lugar para trabajar. **Las empresas inclusivas aumentan su capital simbólico y reputación**, aumentando el nivel de aprobación que reciben desde el interior de la propia empresa y desde el entorno. Finalmente, algunas empresas (con más experiencia inclusiva) señalan que **la participación de personas con discapacidad en su equipo se traduce en beneficios relacionados con la productividad**, ya que estos trabajadores serían especialmente hábiles para la ejecución de ciertas tareas y especialmente comprometidos con la empresa tras acceder a un puesto de trabajo.

Los beneficios sociales identificados radican en el plano de la cultura, pues se indica que **la inclusión cambia la base emocional de la exclusión** (desinformación, miedo, prejuicio, mito). Aumenta la valoración social de la diversidad y cambia positivamente el modo colectivo de convivir e influye en la disminución del conflicto intercultural entre personas con y sin necesidades especiales.

1.6 Estructura del documento

Este documento se ha estructurado según las indicaciones de la normativa de trabajos de fin de grado de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha, y contará con los siguientes capítulos:

Capítulo 2: Objetivos

Se realiza una exposición concreta de problema a resolver, describiendo el entorno de trabajo, la situación y qué se pretende obtener a modo de requisitos del sistema.

Capítulo 4: Antecedentes

Se resumen los antecedentes teóricos y conceptuales y estudios ya existentes, describiendo hechos clave, estado actual y la evolución de la problemática de la visión por computador en el ámbito científico y disciplinar relacionada con el desarrollo de GrayAR.

Capítulo 5: Método de trabajo

Analiza el diseño metodológico escogido, con el fin de responder cómo se ha llevado a cabo el desarrollo de GrayAR, los métodos que se han seguido y la forma de conseguir los objetivos planeados mediante las sucesivas iteraciones. También se describen los recursos empleados, tanto hardware como software.

Capítulo 6: Arquitectura

Describe el diseño e implementación del sistema, detallando los problemas surgidos y las soluciones aportadas. Se estructura acorde a los módulos y submódulos que componen el sistema, dando primero una visión general y funcional del módulo, y pasando posteriormente a los detalles técnicos y de implementación.

Capítulo 7: Resultados

Detalla los resultados obtenidos mediante el análisis de rendimiento del sistema, costes económicos del proyecto y estadísticas del repositorio. El objetivo de este estudio es obtener parámetros de referencia que permitan evaluar nuestro sistema respecto a otros productos que presenten funcionalidades iguales o similares.

Capítulo 8: Conclusiones y propuestas

Muestra un análisis del trabajo realizado y los objetivos conseguidos. Incluye una descripción de posibles líneas de trabajo futuro sobre el proyecto y una valoración personal del trabajo realizado.

Capítulo 2

Objetivos

Hoy en día, la mayor parte de la producción de información en forma de documentos se realiza por medio de herramientas informáticas. Actualmente, en la administración pública se está extendiendo el uso de la firma electrónica, que garantiza la autenticidad del documento digital, pero todavía hay un gran número de personas que no lo utilizan. Es un caso muy habitual que muchos formularios, aún encontrándose en formato digital, se presenten los documentos sobre un soporte físico.

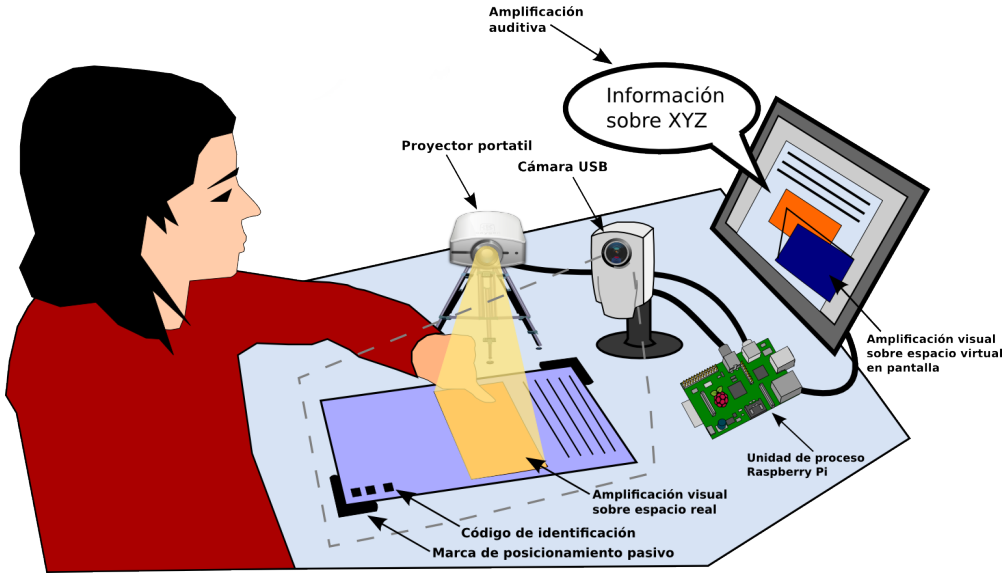


Figura 2.1: Esquema de componentes funcionales de ARgos

Según lo anterior, sería deseable disponer de una herramienta que permitiera el tratamiento directo sobre estos documentos «físicos». *ARgos* es, un proyecto de la Cátedra INDRA-UCLM que pretende la construcción de un sistema de ayuda a la gestión de documentos impresos mediante el uso de técnicas de visión por computador, síntesis visual y auditiva y técnicas de realidad aumentada. En la figura 2.1 se recogen los componentes funcionales del sistema.

GrayAR, el Trabajo Fin de Grado propuesto, abordará el desarrollo de los sistemas de captura, tracking y registro, e identificación de documentos dentro del *Proyecto ARgos*, mientras

que la parte de representación, delegación de tareas y scripting será realizada por Santiago Sánchez Sobrino en su TFG *«BelfegAR: Plataforma para el despliegue gráfico 3D y delegación de tareas en gestión documental con realidad aumentada»*.

2.1 Objetivo general

El objetivo general es construir sistema que utilice una cámara de bajo coste como entrada al módulo de visión por computador y un cañón de proyección portátil para mostrar información visual, directamente alineada sobre el documento del mundo físico. Responderá a las peticiones que el usuario realice sobre el espacio físico, ampliando información relacionada que sea relevante a la acción que quiera realizar. El documento podrá moverse dentro de una región del escritorio y la amplificación deberá quedar perfectamente alineada en el espacio físico. Como soporte hardware, se utilizará un computador en placa Raspberry Pi con arquitectura ARM.

En base a este objetivo general se plantean una serie de objetivos específicos que se detallan en la siguiente sección.

2.2 Objetivos específicos

- **Captura y preprocesado de imágenes.** Deberemos proveer al sistema de un módulo para obtener las imágenes y aplicarle el procesado previo necesario, como puede ser el escalado, umbralización, detección de bordes o detección de características [Ort12] [BET08]. Otra tarea a realizar es calcular la distorsión debida a la proyección en perspectiva mediante los parámetros extrínsecos e intrínsecos de la cámara.
- **Sistema de identificación de documentos.** GrayAR contará con un sistema de identificación rápida empleando algoritmos de recuperación de imágenes y comparará el documento que está siendo analizado con una base de datos de documentos conocidos por el sistema.
- **Implementación de técnicas de tracking y registro.** Para el correcto alineado de la información mostrada, el módulo de tracking y registro contará con funciones de cálculo de *pose* (rotación y translación del objeto en el espacio 3D) en tiempo real y algoritmos para la estimación y descripción del movimiento como *Optical Flow* [LK81].
- **Utilización de paradigmas de interacción natural con el usuario (NUI).** El usuario podrá interactuar directamente en el espacio físico sin utilizar sistemas de mando o dispositivos de entrada tradicionales como sería un ratón, teclado, etc. siendo sustituidos por funciones más naturales como el uso de movimientos gestuales con las manos.
- **Facilitar la gestión documental a personas con necesidades especiales.** Contará con diferentes modos de amplificación de la información del mundo real. Por un lado,

la información visual se amplificará empleando el cañón de proyección que mostrará información relevante al contexto directamente sobre el espacio del papel, así como otras fuentes de información visual adicionales.

- **Se debe basar en componentes de bajo coste.** Para facilitar la implantación real en el entorno de trabajo, deberá funcionar con componentes de bajo coste, incorporando mecanismos de corrección de distorsión y registro 3D totalmente software.
- **Dispositivo multiplataforma (hardware y software).** El desarrollo de GrayAR se realizará siguiendo estándares, tecnologías y bibliotecas libres multiplataforma, con el objetivo de que pueda ser utilizado en el mayor número de plataformas posibles tanto hardware (x86, x86-64 y ARM) como software (GNU/Linux, Windows y Mac).

Capítulo 3

Objectives

NOWADAYS, the largest portion of the information production of documents is carried out using IT tools. Currently, the use of the electronic signature, what guarantees the authenticity of the digital document, is being spread in the public administration, but there is still a large number of people who don't use this procedure. It is very common that in many forms, even being in digital format, documents are presented in a physical medium.

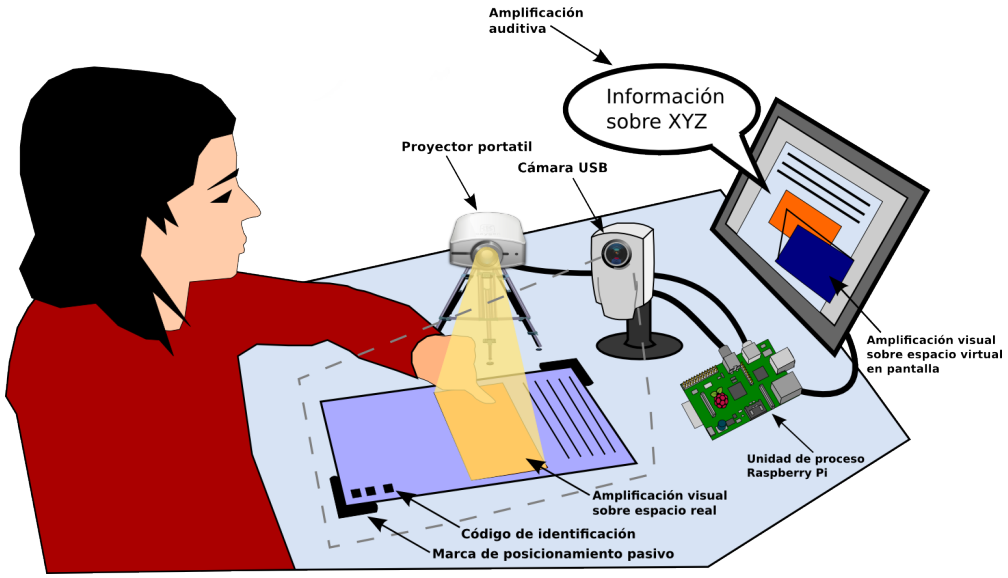


Figura 3.1: Functional components of *ARgos*

According to the above, it would be desirable to have a tool that could be able to manage these *«physical»* documents directly. *ARgos* is an INDRA-UCLM Chairs Project that expects the construction of a support system for the management of printed documents using a computer vision system, visual and hearing synthesis and augmented reality techniques. The functional components of the system are shown in the figure 3.1.

GrayAR, the proposed end-of-degree project, will discuss the development of the recording, tracking and registration systems and the identification of documents that belong to the *ARgos Project*, whereas the representation part, the delegation of tasks and scripting will be developed by Santiago Sánchez Sobrino in his final year work *«BelfegAR: Platform for*

the 3D graphic display and delegation of tasks in documental management with augmented reality». The specific objectives to be developed are summarised below.

3.1 General objective

The general objetive is to build system that uses a low-cost camera as an input in the vision-by-computer module and a portable projection screen to show the visual information, directly aligned with the document of the physical world. It will respond to the requests done by the user in the physical space, expanding information related to the desired action. The document will be able to move in a specific region of the desktop and the amplification must be perfectly aligned in the physical area. A Raspberry Pi board with ARM architecture will be used.

From the main objective described in the previous section, the specific objectives to be developed are summarised below.

3.2 Specific objectives

- **Image capture and processing.** We must provide the system with a module to obtain the images and apply a previous process, as it can be the scaling process, thresholding process, edge detection or characteristics detection [Ort12] [BET08]. Another task to be done is the calculation of the distortion due to the perspective projection using the extrinsic and intrinsic parameters of the camera.
- **Document image recognition system.** GrayAR will have a rapid identification system which will use image-retrieval algorithms and it will compare the document which is being analysed with a document database, known for the system.
- **Implementation of tracking and registration techniques.** The tracking and registration module will have real-time pose calculation features (rotation and translation of the object in the 3D space) and algorithms to estimate and describe the movement, such as *Optical Flow* [LK81], all this to get the proper alignment of the displayed information.
- **Use of Natural User Interface (NUI) paradigms.** The user will directly interact with the physical space without using control systems or traditional input devices such as mouse, keyboard, etc. These devices will be replaced by natural functions such as simple gestures of the user's hands.
- **Facilitation of document management for people with special needs.** GrayAR will have different ways to amplify the real world information. On the one hand, the visual information will be augmented using a projection screen that will show relevant information directly on the paper space, as well as other additional sources of visual information.

- **Low-cost components based.** GrayAR must operate with low-cost components to facilitate the deployment at scale, incorporating totally-software correction and 3D registration mechanisms.
- **Multi-platform device (hardware and software).** The GrayAR development will be made following multi-platform free standards, technologies and libraries, with the aim of using it in the largest number of possible platforms, hardware (x86, x86-64 y ARM) as well as software (GNU/Linux, Windows y Mac).

Capítulo 4

Antecedentes

En este capítulo se introducirán los campos y las tecnologías relacionadas con este proyecto, realizando una revisión de las mismas. Se realizará un estudio del estado del arte de los sistemas existentes.

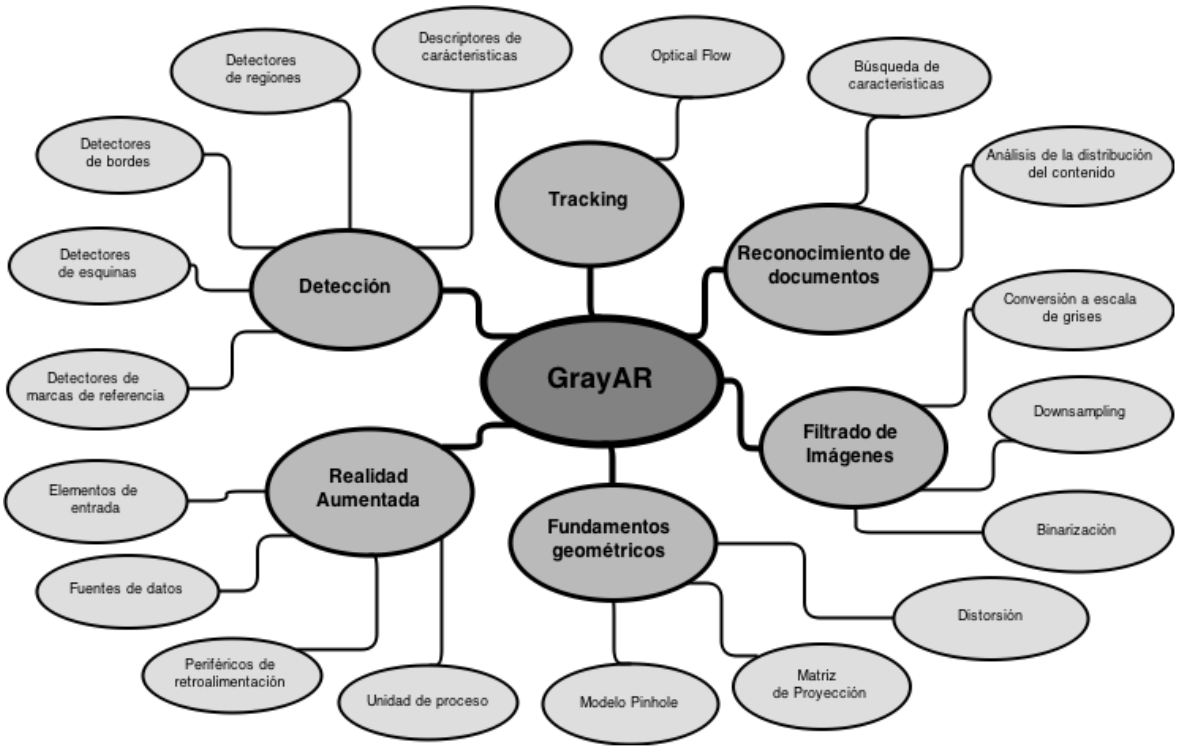


Figura 4.1: Antecedentes de GrayAR

4.1 Geometría de la formación de imágenes

La formación de las imágenes consiste en una representación bidimensional del mundo 3D, perdiéndose la información de profundidad.

La óptica geométrica clásica se basa en modelos de lentes para modelar el proceso de formación de las imágenes. Sin embargo, podemos simplificar este proceso suponiendo que todos los rayos que llegan a la cámara atraviesan un único punto y se proyectan en un plano. Este modelo se conoce como modelo ***pinhole***.

Debido a que las lentes no tienen un comportamiento ideal, habrá que añadir al modelo unos parámetros de distorsión que permitan corregirlo y aproximarlo al comportamiento real de la cámara.

4.1.1 Estructura del modelo *pinhole*

El modelo ***pinhole*** [HZ03] permite modelar el proceso de formación de las imágenes mediante una proyección central, en la cual, de cada punto del espacio tridimensional sale un rayo de luz que pasa por un punto fijo del espacio y intersecta en un plano dando lugar a la imagen.

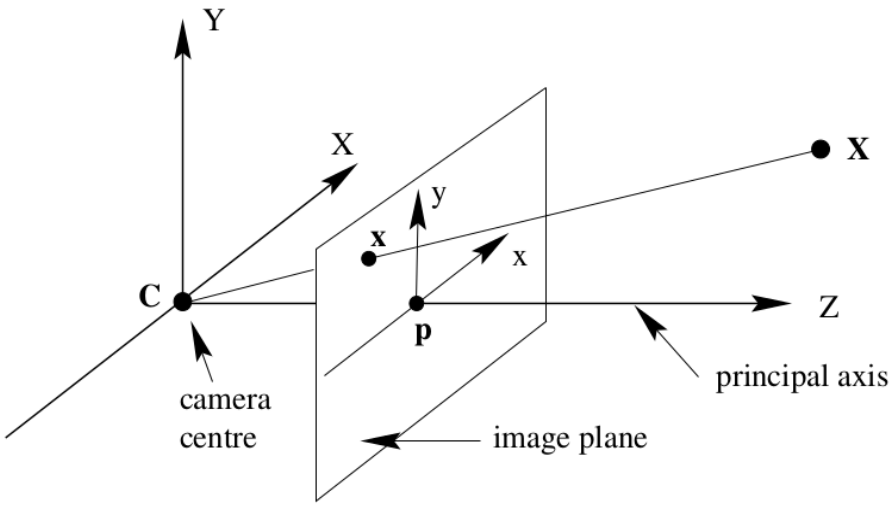


Figura 4.2: Modelo de cámara *pinhole*. La imagen de un punto 3D se forma mediante proyección de perspectiva. Un punto X es proyectado a un punto x en el plano imagen

Los elementos que forman este modelo se definen de la siguiente forma:

- El **centro óptico** es el punto fijo del espacio por donde pasan todos los rayos de luz. Se corresponde con el centro de la cámara y es donde se fija el sistema de referencia de la cámara.
- El **plano imagen** o plano focal. De cada punto del espacio parte un rayo de luz que pasa por el centro de proyección e intersecta con este plano formando la imagen. Como se puede ver en la figura, el plano focal se ha situado delante del centro óptico. Si éste estuviese detrás, las imágenes estarían invertidas.
- **Distancia focal.** Se define como la distancia entre el centro de proyección y el plano imagen.
- **Eje principal** Es la línea que pasa por el centro de proyección y es perpendicular al plano imagen.
- **Punto principal.** Es el punto de intersección del eje principal con el plano imagen. Coincide con el centro de la imagen.

- **Plano principal.** Es el plano paralelo al plano imagen y que contiene al centro de proyección. Además, este plano está formado por todos los puntos cuyas proyecciones se corresponden con puntos en el infinito en la imagen.

4.1.2 Matriz de proyección

El modelo *pinhole* [HZ03] fija un sistema de coordenadas proyectivas en el centro óptico. El eje Z de este sistema coincide con el eje principal de la cámara. A partir de ahora nos referiremos a este sistema de coordenadas como sistema de referencia de la cámara. Además, el plano imagen se fija en el plano $Z = f$.

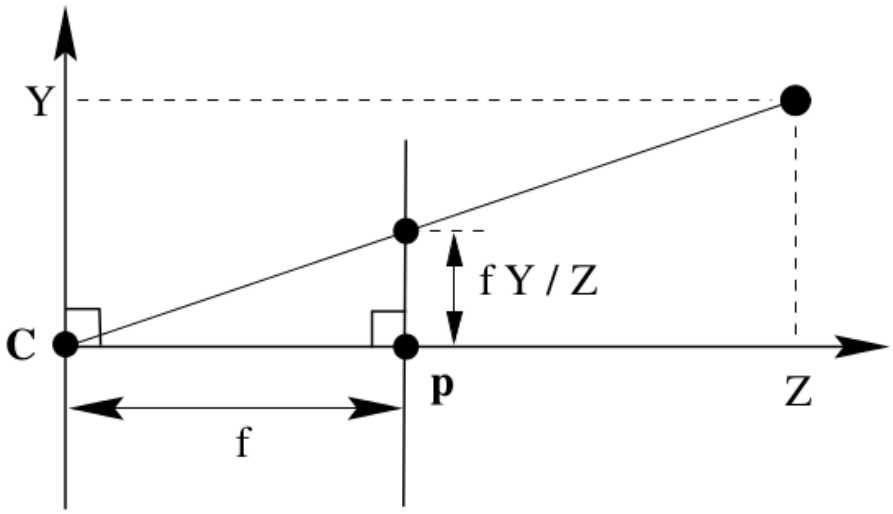


Figura 4.3: Modelo de cámara *pinhole* visto desde el eje X

Utilizaremos $[X, Y, Z]$, para representar vectores fila. Por tanto, su traspuesta, $[X, Y, Z]^T$, será un vector columna.

Por convenio con la notación anterior, las coordenadas de un punto cualquiera se van a representar por un vector columna, definiendo los siguiente elementos a considerar:

- $[X_c, Y_c, Z_c]^T$ Coordenadas de un punto del espacio tridimensional respecto al sistema de referencia de la cámara.
- $[X_w, Y_w, Z_w]^T$ Coordenadas del mismo punto del espacio tridimensional respecto a un sistema de referencia asociado al modelo del objeto. Este nuevo sistema de referencia es distinto al de la cámara.
- $[u, v]^T$ Coordenadas del punto proyectado en el plano imagen.
- f Distancia focal

Mediante relación de semejanza de triángulos, de la figura 4.3:

$$\frac{u}{f} = \frac{X}{Z} \qquad \frac{v}{f} = \frac{Y}{Z} \tag{4.1}$$

podemos determinar la correspondencia entre un punto cualquiera del espacio y su proyección en el plano imagen:

$$[X_c, Y_c, Z_c]^T \Longrightarrow [u, v]^T = [f\frac{X_c}{Z}, f\frac{Y_c}{Z}]^T \tag{4.2}$$

Las **coordenadas homogéneas** son un instrumento empleado para describir un punto en el espacio proyectivo. Consiste en ampliar el plano euclídeo (en el caso bidimensional) al plano proyectivo, es decir, incluirle los puntos impropios o del infinito. De forma que, un punto de dimensiones $[x, y, z]$, se representa por el cuaternión: $[x/w, y/w, z/w, w]$ con $w = 1$.

Este sistema de coordenadas tiene la particularidad de que permite pasar fácilmente coordenadas de un número de dimensiones a otro. Para ello, almacena las coordenadas con una dimensión adicional, de tal forma que para un espacio de 3D, utilizaríamos 4 coordenadas. El valor de la coordenada adicional indica entre otras cosas, si el punto se encuentra en el infinito, $w = 0$, o es un punto cualquiera, $w \neq 0$. En este sistema, si dos coordenadas son proporcionales, se refieren al mismo punto.

Utilizándolas, podemos expresar un punto del espacio tridimensional respecto al sistema de referencia de la cámara de forma matricial:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} fX_c \\ fY_c \\ Z_c \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.3}$$

y la expresión de relación de correspondencia, queda de la siguiente forma:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.4}$$

Y de forma abreviada:

$$q = PQ \tag{4.5}$$

donde:

- $P = diag(f, f, 1)[I|0]$
- $diag(f, f, 1)$ es una matriz diagonal.
- $[I|0]$ representa una matriz identidad de 3x3 concatenada con un vector columna nulo de dimensión 3.

- q es el vector columna, de dimensión 3, que representan las coordenadas homogéneas del punto de la imagen.
- Q es el vector columna, de dimensión 4, que representa las coordenadas homogéneas del punto del espacio respecto al sistema de referencia de la cámara.
- A la matriz P de se la denomina **matriz homogénea de proyección de la cámara**.

4.1.3 Parámetros intrínsecos de la cámara

La matriz de proyección P de la ecuación 4.5 permite transformar las coordenadas de un punto 3D del mundo real en píxeles de una imagen. Se construye a partir de una matriz K y un vector de valores nulos:

$$P = [K|0] \tag{4.6}$$

donde la K es la **matriz de la cámara**, la cuál está formada por una serie de parámetros, denominados **parámetros intrínsecos**:

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

Los **parámetros intrínsecos** son aquellos que describen las características ópticas y geométricas de una cámara y son constantes. Entre estos parámetros se encuentran la distancia focal, el centro óptico y el punto principal. En la figura 4.4 se pueden visualizar estos tres parámetros.

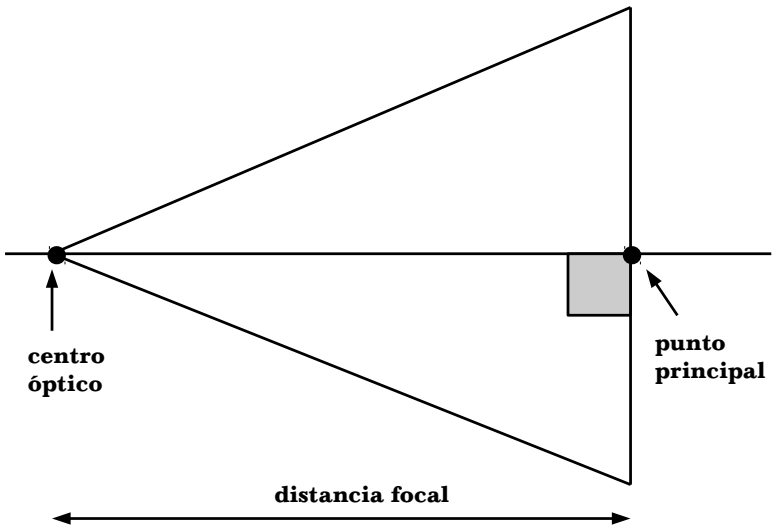


Figura 4.4: Parámetros intrínsecos de una cámara.

- El **centro óptico** o centro de proyección, es el punto de la cámara desde el cual parten todos los rayos que son proyectados en el plano de imagen de la cámara.

- El **punto principal**, denotado como $\boldsymbol{c}$, es la proyección ortogonal del centro óptico sobre el plano de cámara. c_x, c_y indican el desplazamiento del centro de coordenadas del plano imagen, respecto al punto principal. Será nulo sólo si el eje óptico coincide con el centro del sensor de la cámara, pero el eje óptico no siempre atraviesa el centro de la imagen generada.
- El **factor** γ **(*skew factor*)** determina el grado de perpendicularidad de las paredes de los píxeles del sensor. Es inversamente proporcional a la tangente del ángulo que forman los ejes X e Y, por lo que γ tendrá un valor nulo si los píxeles son rectangulares. Esto suele ser así en casi todos los sensores utilizados hoy en día.
- La **distancia focal**, es la distancia existente entre el centro óptico y el punto principal. f_x y f_y son dos distancias focales en píxeles. Son proporcionales a la longitud focal f considerada en las ecuaciones (4.1) y (4.2), según:

$$f_x = fS_x \qquad\qquad f_y = fS_y \tag{4.8}$$

  donde f es la longitud focal física de la lente, en unidades de longitud (milímetros, micras, etc.). S_x y S_y son el número de píxeles por unidad de longitud del sensor, a largo del eje X y del eje Y respectivamente. Como es obvio, si el sensor tiene el mismo número de píxeles por unidad de longitud en todas sus dimensiones, las dos focales f_x y f_y tendrán el mismo valor.

4.1.4 Distorsión

El uso de lentes facilita la entrada de luz, un enfoque adecuado y una mayor versatilidad, pero también introduce deformaciones en las imágenes que se forman en el sensor. Para el cálculo de coeficientes de distorsión, se tienen en cuenta factores radiales y tangenciales.

La **distorsión radial**, consiste en el desplazamiento de los píxeles de de la imagen, de tal modo que las líneas situadas en los extremos del encuadre aparentarán salir hacia el exterior o el interior.

Para la corrección de la distorsión radial de las coordenadas de un píxel de la imagen, se utiliza la siguiente fórmula:

$$\begin{aligned} x_{corrected} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{corrected} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{4.9}$$

La **distorsión tangencial** se produce porque la lente no se encuentra perfectamente paralela al plano de imagen. Se puede corregir a través de las siguientes fórmulas:

$$\begin{aligned} x_{corrected} &= x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{corrected} &= y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{aligned} \tag{4.10}$$

De tal forma que tenemos cinco parámetros de distorsión que son representados como una matriz fila con 5 columnas:

$$Distortion_{coefficients} = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3] \tag{4.11}$$

4.1.5 Parámetros extrínsecos de la cámara

En la sección 4.1.2 se asumió el hecho de que el centro óptico era el origen de coordenadas del mundo. Esto era así a efectos del cálculo de parámetros intrínsecos, con lo cual el modelo era válido siempre que el sistema no fuera movido de su posición inicial.

Por otro lado, en la mayor parte de las aplicaciones prácticas es necesario que la cámara se mueva o se gire, para captar adecuadamente la escena. Por ello, para poder modelar el sistema con independencia de que su posición haya sido alterada o de que un objeto se pueda referenciar respecto al origen de coordenadas de la cámara, es necesario modificar la ecuación 4.5 introduciendo un matriz de transformación $[R|t]$ que contiene los **parámetros extrínsecos** de la cámara.

$$q = K[R|t]Q \tag{4.12}$$

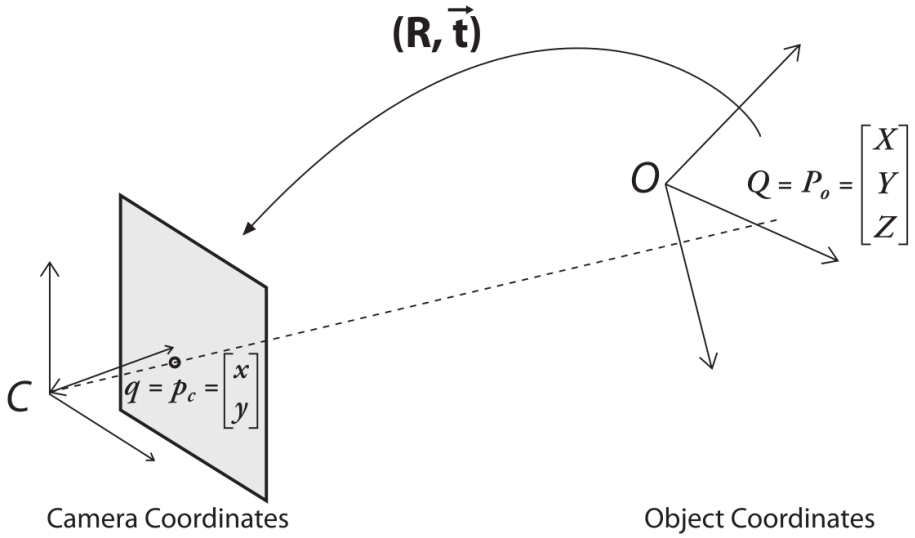


Figura 4.5: Conversión del sistemas de coordenadas del objeto al sistema de la cámara. El punto p_c está relacionado con el punto P_0 mediante la aplicación de una la matriz de rotación y el vector de traslación. (Bradski y Kaehler)

$$[R|t] = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & T_1 \\ R_{2,1} & R_{2,2} & R_{2,3} & T_2 \\ R_{3,1} & R_{3,2} & R_{3,3} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

Donde:

- R es una **matriz de rotación**, que representa un giro de la cámara (o de un objeto respecto de ella). Tendrá una forma distinta dependiendo de respecto a que eje (X, Y, Z) se haga la rotación:

$$R_x(\Psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Psi & \sin\Psi \\ 0 & -\sin\Psi & \cos\Psi \end{bmatrix} \tag{4.14}$$

$$R_y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} \tag{4.15}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

Si el giro se realiza respecto al eje Z, tal y como se muestra en la figura 4.6, las nuevas coordenadas quedarán de la siguiente forma:

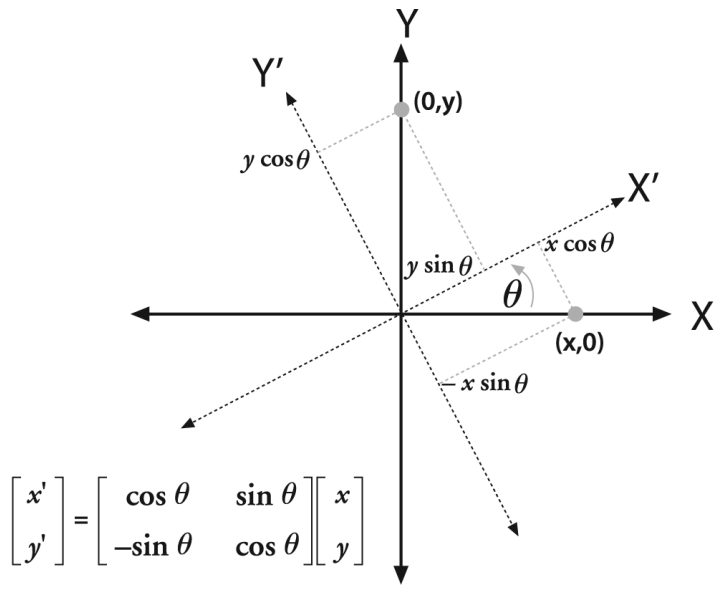


Figura 4.6: Rotación de puntos alrededor del eje Z (Bradski y Kaehler)

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \Rightarrow \begin{array}{l} X' = X\cos\theta + Y\sin\theta \\ Y' = -X\sin\theta + Y\cos\theta \\ Z' = Z \end{array} \tag{4.17}$$

- t es un **vector de translación** que representa el cambio de un sistema de coordenadas a otro cuyo origen se desplaza a otra ubicación; en otras palabras, el vector de traslación es el desplazamiento desde el origen del sistema de coordenadas inicial hasta el origen del sistema de coordenadas final. En nuestro caso, como podemos ver en la figura 4.5, el sistema inicial pertenecería al del objeto, y el final, al sistema de coordenadas de la cámara.

$$t = origen_{objeto} - origen_{camara} \tag{4.18}$$

El cálculo aproximado de la matriz de transformación $[R|t]$ a partir de la información visual capturada se denomina ***pose estimation***. La solución para la estimación de la *pose* ha sido propuesta, entre otros métodos, mediante transformada lineal directa (DLT) ó algoritmos Pnp (perspective-n-point).

4.2 Realidad Aumentada

La Realidad Aumentada es el **conjunto de técnicas que permiten integrar en tiempo real contenido digital a la percepción del mundo real**. El término fue acuñado en 1990 por Tom Caudell durante el desarrollo de un sistema de Boeing para ayudar a los trabajadores en el ensamblaje de aviones mediante la ayuda de pantallas que mostraban información del montaje.

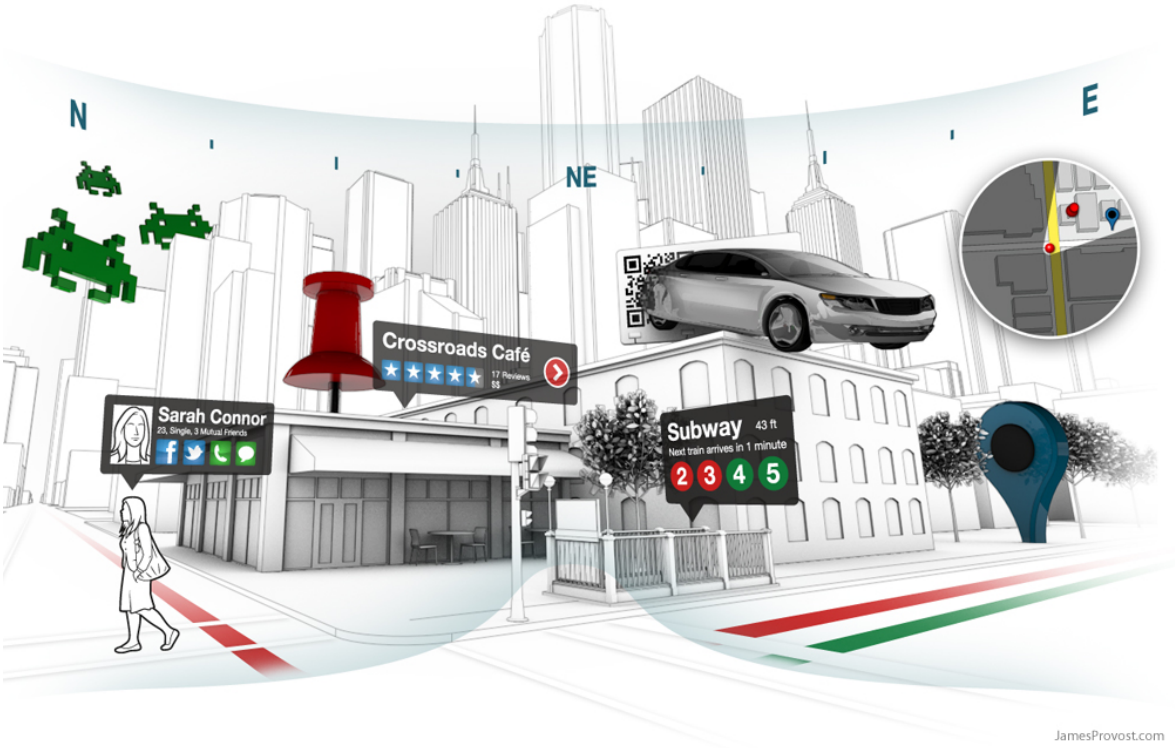


Figura 4.7: Representación conceptual de la realidad aumentada (James Provost)

Figura 4.8: Taxonomía de la *Realidad Mixta* según Milgram y Kishino

Uno de los principales problemas a resolver es denominado **registro**. Los objetos del mundo real y virtual deben estar correctamente alineados o la sensación de integración se verá seriamente afectada. Sin un registro preciso, la realidad aumentada no podría ser soportada por muchas aplicaciones, por ejemplo, en medicina para el guiado de una aguja en la realización de una biopsia.

El paradigma general por el cual se mezcla el mundo real y el virtual se denomina ***Realidad Mixta***. Milgram y Kishino [KM94] describieron la jerarquía (Figura 4.8) en la que se clasifica este tipo de paradigma. A la izquierda de la imagen, se muestra el caso extremo de *integración nula*, en el que la interacción se realiza únicamente a través de los objetos reales al no existir ningún tipo de elementos virtuales.

El siguiente paradigma es el de ***Realidad Aumentada***. En este paradigma se parte del *mundo real* como base, y a partir de él se añaden los objetos virtuales. Este paradigma contempla multitud de ventajas y aplicaciones. Henderson y Feiner llevaron a cabo un estudio evaluando las ventajas de la Realidad Aumentada frente a métodos convencionales en tareas específicas como la realización de procesos de mantenimiento [HF09].

Acercándose hacia los *entornos virtuales*, se encuentra el paradigma de la ***Virtualidad Aumentada*** [NY09]. Estos sistemas parten de un entorno completamente virtual, que puede ser compartido por varios usuarios, y se interactúa con él a través del mundo real. Algunos proyectos que han utilizado este paradigma se enfocaban como *sistemas colaborativos*, en los que se crean un punto de encuentro virtual (salas de reuniones). Regenbrecht utiliza este paradigma con fines colaborativos [RLK+04] y para implementar una videoconferencia remota [ROW+03].

Finalmente, se encuentran los sistemas de ***Realidad Virtual*** [Ste92], en los que todo es generado por ordenador, y el objetivo es que el usuario se abstraiga lo máximo posible del mundo real, para conseguir una inmersión total en el mundo virtual. Estos sistemas están

actualmente en auge, y varios son los fabricantes que disponen de gafas tipo HDM como es el caso de Oculus VR, recientemente comprada por Facebook [1], o Vuzix[2].

Para distinguir que podemos llamar realidad aumentada y que no, Ronald T. Azuma [Azu97], define una serie de características que deben cumplir las aplicaciones:

- **Combinar el mundo real con el virtual.** El resultado final debe mostrar la información sintética sobre las imágenes percibidas del mundo real.
- **Debe ser interactivo en tiempo real.** La integración debe ser realizada *en el momento*, por lo que el cálculo necesario debe realizarse n el menor tiempo posible.
- **La alineación de los elementos virtuales debe realizarse en 3D.** Los objetos sintéticos deben de estar en el espacio tridimensional.

A la hora de construir un sistema de realidad aumentada, podemos observar que depende de cuatro componentes físicos:

- **Elementos de entrada y sensores de orientación.** Proporciona al sistema la información visual, la entrada del usuario y, potencialmente, ayudar a la orientación.
- **Fuente de datos.** Proporciona información aplicable al medio ambiente para que el sistema aumente la visión del usuario.
- **Periféricos de retroalimentación de los usuarios.** Principalmente en la forma de producción visual, pero también pueden incluir audio y otras interfaces de usuario.
- **Unidad de proceso.** Combina los datos de los sensores de entrada para determinar la orientación y aumentar la visión del usuario con información de la fuente de datos. Envía el resultado a los periféricos de retroalimentación del usuario.

En los siguientes apartados se detallan cada uno de estos elementos, y como se podrá observar, el amplio abanico de aplicaciones en las que situar la realidad aumentada dependiendo de las combinaciones de estos componentes.

4.2.1 Elementos de entrada y sensores de orientación

Un sistema de realidad aumentada debe ser capaz de conocer su orientación y posición dentro del entorno circundante con el fin de ofrecer una experiencia «aumentada» satisfactoria. La combinación de la orientación y la posición de un observador que se conoce como *pose*. La orientación y posición se pueden determinar mediante técnicas de visión por computador, pero hay métodos alternativos para estimar la *pose* utilizando sensores diseñados específicamente para medir alguno o todos los parámetros de los seis grados de libertad que componen la *pose*.

[1] http://www.businessinsider.com/facebook-to-buy-oculus-rift-for-2-billion-2014-3
[2] www.vuzix.com

Localización

La manera más popular para determinar la posición es mediante la utilización del Sistema de Posicionamiento Global (GPS), desarrollado por el ejército de Estados Unidos y totalmente operativo desde 1994. Un receptor GPS es lo suficientemente preciso para determinar la ubicación en las aplicaciones de realidad aumentada, además, debido a su bajo coste, los sensores de GPS se llevan incluido en los teléfonos móviles desde 2005.

Orientación

En el espacio 3D, la orientación de un cuerpo rígido puede ser determinado de forma única por tres ángulos o grados de libertad (3DOF): *pitch, roll y yaw*. La figura 4.9 se puede observar una representación de estos tres ángulos como rotaciones alrededor de sus ejes correspondientes.

Para facilitar la tarea de orientación los dispositivos móviles, sobre todo smartphones y tablets, están equipados una serie de sensores como:

- **Sensores magneto-resistivos**, o magnetómetros, que son utilizados como una «brújula electrónica» y se utilizan para determinar el *yaw* y la dirección. Las nuevas generaciones de brújulas electrónicas utilizan tres ejes para asegurar una lectura correcta sin importar la inclinación del sensor.
- **Acelerómetros** para medir la aceleración aceleración experimentada por un objeto con respecto a un marco de referencia. Los acelerómetros no pueden determinar el *yaw*, pero pueden ser utilizados para determinar el *pitch* y el *roll*.
- **Giroscopios electrónicos** que son utilizados para determinar los tres grados de libertad de orientación a la vez.

Un problema común en los sistemas de seguimiento visual es que los movimientos bruscos de la cámara pueden ocasionar que la imagen salga borrosa y verse afectado el seguimiento

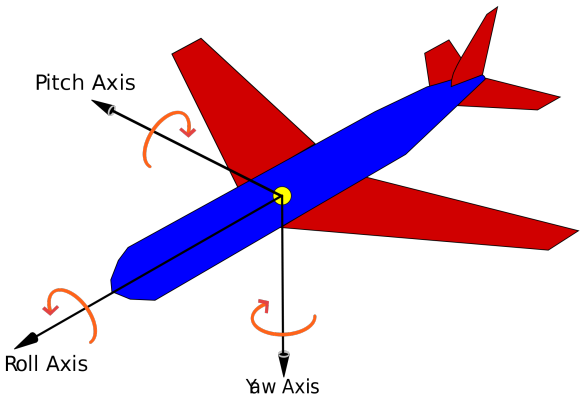


Figura 4.9: Ángulos de orientación: *yaw, pitch y roll*

visual. Un dispositivo híbrido equipado con un giroscopio y un acelerómetro será capaz de determinar rápidamente y con precisión tres de los seis grados de libertad de la *pose* y ayudar al sistema de seguimiento visual a recuperarse cuando se experimentan imágenes borrosas.

Fuentes de Vídeo

La realidad aumentada superpone información sobre lo que el usuario está viendo. Para los sistemas dinámicos en los que la información virtual debe ser mezclada con la imagen real, se requiere un dispositivo de captura, como una cámara digital. Para los propósitos de las aplicaciones de realidad aumentada es preferible una cámara de vídeo. La mayoría de las aplicaciones utilizan una única cámara y se conocen como sistemas monoculares. También existen sistemas estereoscópicos, que utilizan dos cámaras para determinar la profundidad.

El desarrollo tecnológico en el ámbito de la captura de imagen digital que han mantenido los principales fabricantes en los últimos 10 años, conocido como la *«batalla del megapíxel»*, ha supuesto un gran avance para disponer de sensores con una gran calidad de imagen y un coste reducido. Actualmente es normal encontrar cámaras de alta resolución en prácticamente todos los dispositivos móviles, permitiendo grabar vídeo en una resolución FullHD ó 4K y obtener imágenes fijas de hasta 7136x5360px.

Entrada táctil

En sistemas de realidad aumentada portátiles tales como teléfonos inteligentes o tablets, la pantalla funciona al mismo tiempo como una entrada para manipular objetos directamente en la pantalla y un dispositivo de salida.

4.2.2 Fuente de datos

Un sistema de realidad aumentada requiere una fuente de datos e información con la que aumentar la realidad de un usuario. La fuente de datos puede ser desde una base de datos directamente disponible en el sistema, o puede ser información agregada a partir de fuentes disponibles a través de una conexión de red. Actualmente existe un gran desarrollo para el uso de Internet como fuente de datos para aplicaciones de realidad aumentada. Aplicaciones como Wikitude[3] y Layar[4] son ejemplos de las sistemas que utilizan los datos disponibles en Internet como fuente de datos.

[3] http://www.wikitude.com
[4] http://www.layar.com

4.2.3 Periféricos de retroalimentación de los usuarios

Dispositivos de Visualización

Existen muchos métodos y dispositivos para mostrar la información o imágenes generadas para su combinación y alineación con la realidad:

- **Monitores y pantallas tradicionales** Sobre todo impulsado por la popularidad de frameworks y bibliotecas como ARToolkit [KB99], existen muchas implementaciones de realidad aumentada para utilizar con un ordenador de sobremesa o portátil. No requieren de tecnología especial, únicamente una webcam que el usuario puede controlar. La aplicación se suele presentar en una pagina web y permite a los usuarios superponer información o imágenes sobre lo capturado por la cámara. Para la tarea de *tracking* y registro se utilizan marcadores que el usuario previamente ha impreso.
- **HMDs (Head Mount Display)** Los HDM o Head Mount Display son dispositivos que permiten al usuario ver el el mundo real con objetos virtuales superpuesto mediante técnicas ópticas y de vídeo. Los HDM se pueden clasificar en dos categorías:
  - **Ópticos (OST - Optical See-Throug)** Permiten al usuario ver el mundo real a través de sus ojos y la imagen virtual se muestra mediante elementos ópticos holográficos, espejos semiplateados o alguna otra técnica similar. Un ejemplo de este tipo de dispositivo son las Google Glass.
  - **Vídeo (VST - Vídeo See-Throug)** En los dispositivos de esta categoría, el usuario percibe tanto el mundo real como la imagen virtual mediante una o varias pantallas. Como ventajas de los VST está una mayor consistencia entre el mundo real y el virtual.
- **Visualización basada en proyección** Un sistema de visualización basado en proyección es una buena opción para sistemas fijos, además proporciona una intrusión mínima y la capacidad de interacción con varios usuarios. Se han propuesto una gran variedad de técnicas de visualización mediante proyectores sobre objetos y otras superficies (no necesariamente pantallas). Ehnes [EHH04] amplió el trabajo anterior de Pinhanez [Pin01] sobre la proyección de vídeo para mostrar imágenes virtuales sobre objetos reales directamente utilizando técnicas de *tracking* de vídeo para seguir el movimiento y manteniendo la imagen sobre objeto mientras este se mueve.
- **Dispositivos portátiles (smartphones, tablets)** Hoy en día el dispositivo portátil por excelencia es el smartphone. Es mínimamente invasivo, muy portable y ampliamente extendido. Según el informe de comScore [5] el 66 % de los españoles tienen un teléfono inteligente, por lo que son una gran alternativa para la visualización de aplicaciones de realidad aumentada.

[5] http://www.comscore.com/esl/Insights/Presentations-and-Whitepapers/2013/2013-Spain-Digital-Future-in-Focus

Audio

El audio se utiliza principalmente de dos formas diferentes en los sistemas de realidad aumentada: como una modalidad de la interfaz de usuario multimodal o para ofrecer una experiencia aural aumentada.

Por ejemplo, el usuario pueda ser capaz de dar comandos de voz y el sistema devuelva retroalimentación con señales de audio (pitidos). Este tipo de audio no direccional es trivial desde el punto de vista tecnológico aportando una nueva dimensión a las aplicaciones móviles. El audio aumentado puede servir a personas con discapacidad visual para ayudar a comprender el entorno. Mediante unos altavoces o un par de auriculares, es bastante simple proporcionar al usuario la información en forma de audio. La síntesis de voz también se puede utilizar para leer información, por ejemplo, leer palabras en voz alta desde un sistema de reconocimiento de texto.

4.2.4 Unidad de proceso

El teléfono móvil ha evolucionado hacia una plataforma informática móvil (smartphone), en la que desarrollar las actividades habituales de un ordenador personal. El principal inconveniente de estos era la pantalla reducida que disponen lo que ha derivado en el desarrollo de un nuevo dispositivo denominado tablet.

Las tablets disponen de prácticamente las mismas características que los smartphones (incluso el mismo sistema operativo) pero las pantallas son mayores, sin necesidad de teclado físico ni ratón, con las que se interactúa principalmente mediante la pantalla táctil con los dedos o un stylus.

Otra tendencia tecnológica es el desarrollo de SBC, *(Single Board Computer)* que básicamente es un ordenador completo montado sobre un circuito. Su diseño se basa en un microprocesador de bajo rendimiento (normalmente de arquitectura ARM), RAM y diversos dispositivos de E/S como puertos USB, lectores de tarjetas, conectores Ethernet, y salida de audio y vídeo con un precio final muy reducido. Este tipo de dispositivos se crearon para utilizarlos con fines educativos, pero entre la comunidad de usuarios, debido a su gran versatilidad se utilizan para múltiples aplicaciones tales como centros multimedia o distintos tipos de servidores dado su bajo consumo.

4.3 Técnicas de Visión por Computador

En esta sección comentaremos las técnicas que pueden utilizarse para el tratamiento previo de imágenes que se utilizarán para la detección y *tracking* en sistemas de realidad aumentada. El orden en que se introducen suele ser el aplicado habitualmente.

La adquisición de imágenes es el primer paso en un sistema de realidad aumentada y es dependiente de la plataforma del sistema. Por norma general, se considera que las imágenes se adquieren con 24 bits, un canal por cada una de sus componentes de color en 8 bits:

rojo, verde y azul. También vamos a asumir que las imágenes se adquieren a 30 fotogramas por segundo. Esta suposición esta basada en la transmisión típica con la que trabajan la mayoría de webcams. Prácticamente todas las técnicas presentadas procesan y trabajan las imágenes en blanco y negro. Aún realizando la captura en color, normalmente se realizan varios pasos previos para reducir y simplificar la cantidad de información que se tiene que manejar, dejando solo la información esencial.

Por ejemplo, un frame típico obtenido a partir de una webcam o la cámara de un teléfono móvil puede tener una anchura de 640 píxeles y 480 píxeles de altura. Suponiendo que una imagen en color con tres canales (rojo, verde y azul) se obtiene con ocho bits por canal, la cantidad de datos por frame serian 640 x 480 x 3 x 8 = 7.372.800 bits de información. Esto es para una sola imagen. La mayoría de cámaras tienen una tasa de transferencia entre 20 y 30 frames por segundo, lo que significa que para cada frame se debe procesar toda esa información por completo cada 33 milisegundos antes de que llegue el siguiente fotograma. Sería muy costoso (e innecesario), por lo que se utilizan una serie de técnicas como las explicadas a continuación para reducir la cantidad de información requerida en las etapas posteriores.

4.3.1 Conversión a escala de grises

La reducción del número de canales de la imagen (rojo, verde y azul) a un solo canal de ocho bits representados como gris, reduce la cantidad de datos a procesar en un factor de tres. Una imagen en escala de grises representa la intensidad de la imagen.

No existe un único método para convertir una imagen en color a escala de grises, el proceso más común consiste en recorrer todos los píxeles de la imagen, y para cada píxel de forma individual multiplicar sus componente RGB (rojo, verde y azul), cuyos valores oscilan entre 0 y 255, por unos coeficientes y sumarlos. De esta forma obtenemos la misma luminosidad del píxel tanto en color como en escala de grises.

Los componentes RGB originales del píxel se sustituyen por el valor resultante de esa suma. Los tres componentes con el mismo valor, para lograr así el efecto de escala de grises, ya que cuando los tres componentes tienen un mismo valor lo que se obtiene es un tono gris, excepto en los extremos, donde (0, 0, 0) es negro y (255, 255, 255) es blanco.

Los coeficientes que se utilizan son 0.299 para el rojo (R), 0.587 para el verde (G), y 0.114 para el azul (B). Al sumar los tres coeficientes el total vale 1 (0.299 + 0.587 + 0.114), lo que garantiza que el valor resultante de la suma de los productos por los componentes originales del píxel será un valor comprendido también entre 0 y 255.

4.3.2 Thresholding (umbralización o binarización)

La **umbralización** consiste en definir un valor umbral y compararlo con cada uno de los píxeles de la imagen. A los píxeles que estén por debajo del umbral se les asigna un 1, y a los

que estén por encima un 0. De esta forma se obtiene una imagen «binaria», en el sentido de que sólo tiene dos valores (blanco y negro) y reduciendo considerablemente la complejidad de la información a analizar.

Aplicar un valor umbral a una imagen presenta dos dificultades. La primera es decidir que valor concreto tomar, y la segunda es conseguir que dicho valor sirva para cualquier imagen posible que pueda llegar desde la cámara. Considerando que los píxeles se encuentran dentro del rango de 0 a 255, en entornos de poca iluminación los píxeles tienden a tomar valores bajos, oscuros, cercanos al 0, y con mucha iluminación tomarán valores altos, claros, cercanos al 255. También es frecuente que la iluminación varíe dentro de la propia imagen, con algunas zonas muy oscuras y otras muy claras.

La **umbralización adaptativa** obtiene mejores resultados que el algoritmo anterior, ya que no aplica el umbral de una manera global sobre todos los píxeles de la imagen, sino que tiene en cuenta las variaciones en la iluminación de una manera local para cada píxel de forma individual.

4.3.3 Filtros Gaussianos

La aplicación de un filtro sobre una imagen normalmente consiste en recorrer todos los píxeles de la imagen, y para cada píxel de forma individual aplicar una «operación» que genere un nuevo valor para el píxel. En este caso, para conseguir difuminar la imagen de forma coherente, se utiliza como base para calcular el nuevo valor para cada píxel, los píxeles más cercanos que tiene a su alrededor. Es decir, si tenemos un píxel negro rodeado de píxeles blancos, lo que se quiere obtener es un nuevo píxel gris que difumine la imagen, eliminando ese *ruido* que representa el píxel negro aislado.

El proceso consiste es definir un tamaño de ventana o kernel, por ejemplo de 3x3 píxeles, a cada celda de este kernel asignarle un coeficiente numérico, y mover el kernel por encima de cada píxel de la imagen original de forma individual, multiplicando los coeficientes del kernel por los valores de los píxeles que cubre cada una de las celdas.

Unos mejores coeficientes serían aquellos que dieran más importancia al píxel original, y fueran decrementado la importancia a medida que se fueran alejando de él. Y eso es precisamente lo que persigue el filtro gaussiano, lo que consigue usando unos coeficientes que se calculan con la siguiente fórmula:

$$\frac{1}{2 \cdot \pi \cdot \sigma^2} \cdot e^{\frac{-(x^2+y^2)}{2 \cdot \sigma^2}} \tag{4.19}$$

Donde x e y son la distancia al píxel original, y σ (sigma) la desviación típica de la distribución gaussiana.

4.3.4 Corrección de perspectiva y proyección

La captura de documentos mediante cámaras normalmente se realiza sin estar la cámara paralela al plano en el que se encuentra, con lo que obtendrá una imagen con una distorsión de la perspectiva. Si los detectores son tolerantes a la perspectiva no es necesaria una rectificación. Sin embargo, la mayoría de procesadores de OCR son muy sensibles a la variaciones del texto bajo perspectiva y fallan.

Si la deformación de la perspectiva producida es ligera, la rectificación se puede realizar por aproximación [HYC02]. En general, supongamos que el documento está en un plano; entonces la transformación proyectiva del plano del documento en el plano de imagen puede ser modelado por una matriz de 3 x 3 en el que ocho coeficientes son desconocidos y uno es el factor de normalización. La eliminación de la perspectiva se puede realizar una vez que se encuentran los ocho incógnitas. Con cuatro pares de puntos correspondientes son suficientes para recuperar los ocho grados de libertad. Bajo el supuesto de que los bloques de texto son rectángulos en un mundo 3D, Jung [JKK$^+$02] utiliza un enfoque directo para establecer cuatro pares de correspondencias y proceder a la rectificación. Sin embargo, este método es muy propenso a errores y por lo tanto sólo puede utilizarse cuando la deformación de la perspectiva es pequeña.

En otros casos, el texto aparecerá sobre superficies curvas, como en los casos de que estemos capturando las páginas de un libro abierto. Estos casos típicos permiten que se realicen suposiciones, como la de utilizar un modelo cilíndrico de para calcular la deformación [KHP93]. Del mismo modo, si suponemos que el texto aparece como líneas rectas paralelas en la página, podemos utilizar la línea de texto para calcular la deformación que se esta produciendo en la página y obtener la imagen como si estuviese sobre un plano.

4.3.5 Downsampling

Otra técnica que se utiliza es la de *downsampling* para reducir el número total de píxeles de la imagen. Lo ideal sería que la relación de aspecto de la imagen original se conserve siempre, para ello el número de píxeles se reduce a la mitad en cada iteración. Cada *downsampling* que se realice sobre una imagen lleva como consecuencia la pérdida de datos que reducirá la precisión del *tracking* y podría influir en la robustez. A la hora de diseñar el sistema se debe decidir si es aceptable una pequeña pérdida de precisión o emplear un enfoque multiresolución, en el que se realiza el *tracking* completo por primera vez en una imagen reducida y los resultados se refinan con versiones de mayor resolución de la imagen [KM09].

La teoría sobre los diferentes niveles de resolución está presentado por Adelson [AAB84] y se describe como un conjunto de copias (con filtros paso bajo o paso banda de una imagen), donde cada una de ellas es una representación de la imagen en una escala diferente. Son esencialmente representaciones a escala múltiple de una imagen a partir de la imagen original

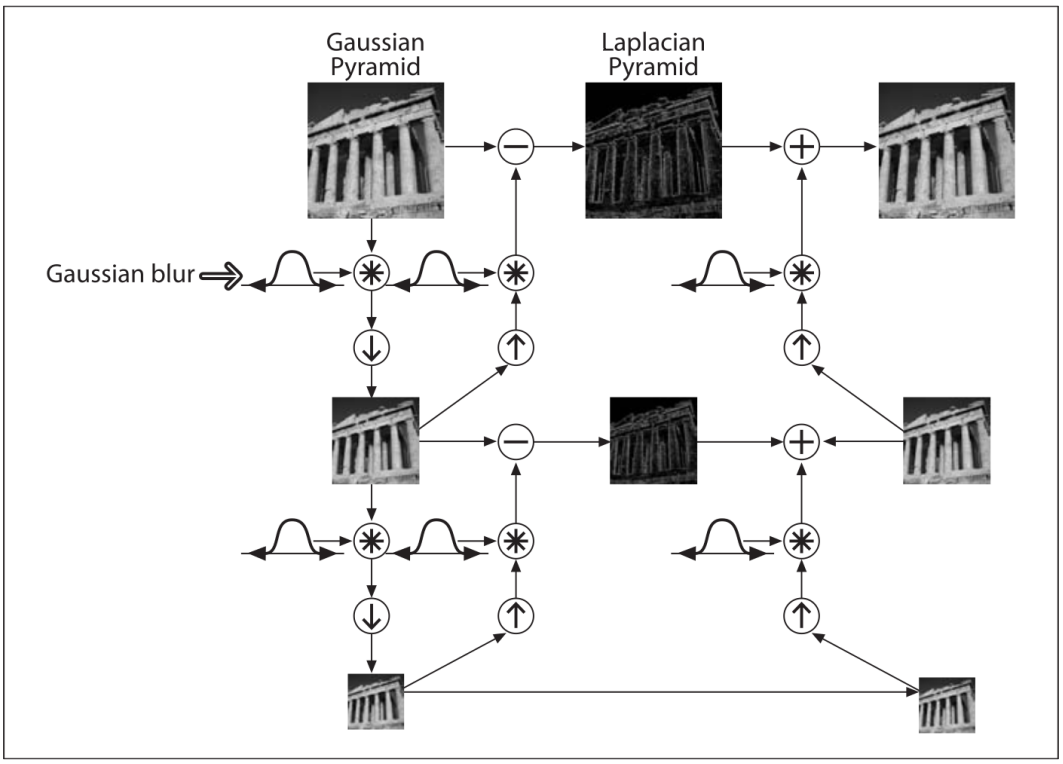


Figura 4.10: Cálculo de las pirámides de Gauss y sus inversas (Laplace). (Bradski y Kaehler)

y con una escala de un factor de dos en cada paso hacia abajo. Este proceso puede ser visualizado como una pirámide con la imagen original en la parte inferior.

La figura 4.10 muestra los pasos involucrados en el cálculo de una pirámide de imágenes y la pirámide laplaciana, que contiene toda la información perdida en cada paso. Si la pirámide de la imagen y la pirámide laplaciana están disponibles, la imagen original puede ser perfectamente reconstruida. En cada nivel de la pirámide, la imagen contiene cuatro veces menos píxeles que la capa inferior (Figura 4.11) lo que afectará directamente al tiempo total de procesamiento.

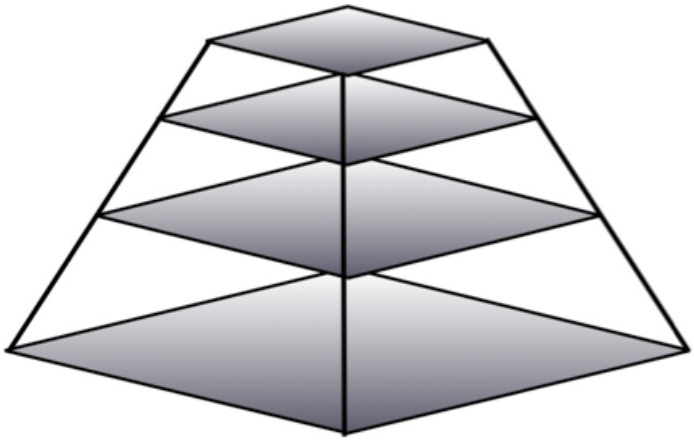


Figura 4.11: Pirámide multiresolución. (journal.code4lib.org)

4.4 Detección

En el contexto de la realidad aumentada, la detección es el proceso de localizar un objeto en una imagen capturada y calcular la posición y orientación de la cámara *(camera pose)* respecto a ese objeto.

Los enfoques que se han dado en las distintas técnicas de detección se pueden clasificar en dos tipos: La detección mediante marcas de referencia o mediante puntos de interés naturales.

4.4.1 Detección basada en marcas de referencia *(fiducial markers)*

Una marca de referencia es un objeto que se coloca en el campo de visión de la imagen a procesar y proporciona un punto de referencia y unas dimensiones conocidas de antemano, facilitando el proceso de detección y el calculo de la *pose*.

Mediante este método Kato presentó en 1999 un paper que utilizaba marcas cuadradas con un borde negro para calcular la *pose* de la cámara en tiempo real[KB99]. El resultado fue la biblioteca ARToolKit que ha popularizado la realidad aumentada.

Otros papers en la misma línea, como el de Stricker [SKR99], en el que describe un método para encontrar las coordenadas 3D de las 4 esquinas de un marcador cuadrado, mientras que Park [PJN99] describe un algoritmo para el cálculo de la *pose* de la cámara a partir de características conocidas.

Hasta el año 2002, las técnicas mediante marcas se habían estudiado ampliamente. Zhang [ZFN02] realizo un estudio recopilando y comparando varios de los principales enfoques que existían. A partir de esta fecha no se han presentado nuevos sistemas basados en marcadores.

4.4.2 Detección basada en puntos de interés naturales

El aspecto más importante de la detección de puntos de interés es hacerlo lo más coherente posible. Los mismos puntos de interés que se detectan en un frame, debe detectarse de nuevo en el siguiente cuadro, dado que todavía siguen presentes en la imagen.

Si tenemos un rectángulo sin bordes en movimiento delante de un fondo del mismo color, sería imposible de rastrear porque el sistema no podría distinguir dos puntos distintos en la imagen. Si tenemos un punto negro sobre un fondo blanco, y esté empieza a moverse, es muy fácil de seguir, ya que el sistema sólo tendría que encontrar ese punto en los siguientes frames. Este punto puede ser visto como una discontinuidad o un punto en el que se produce un cambio busco en la intensidad de la imagen.

Por tanto, el requisito deseable para considerar a un punto de interés es que debe ser una discontinuidad. Si tuviéramos muchos más de estos puntos también sería imposible diferenciar entre ellos y tendríamos el mismo problema que con el rectángulo del mismo color del

fondo. Sólo podremos distinguirlos si la región que rodee al punto de interés es diferente, al menos en cierto grado, de la región local que rodea a todos los demás. Por tanto, el segundo requisito de un punto de interés es que él y la región que lo rodea debe ser único.

Prácticamente en todos los trabajos sobre detección mediante puntos de interés se pueden establecer las siguientes fases:

- **Extracción:** El proceso de extracción consiste en la búsqueda de zonas en la imagen con diferente apariencia que las que están a su alrededor, denominadas características *(features)*. Normalmente las características suelen ser bordes, esquinas o zonas más brillantes u oscuras en función del algoritmo utilizado en particular. A esta fase también se la denomina detección.

  Existen muchos algoritmos de detección, que obtienen distintas tipos de características, por ejemplo, el detector de esquinas de Harris[HS88] o el algoritmo FAST [RD05] que devuelve los píxeles con valores máximos y mínimos en función de sus vecinos mediante técnicas de *Machine Learning*.

- **Descripción:** Se calcula el vector que describe la característica de un punto significativo para la posterior comparación entre otros puntos de interés. Los enfoques basados en el uso de descripción locales han sido ampliamente investigados y se dividen en dos tipos: el histograma de gradientes y la prueba binaria.

  El histograma de gradientes se calcula a partir de la cuantificación de los gradientes dentro de una área local. En SIFT [Low04], una zona se divide en subregiones y se calcula el histograma de gradiente en cada una de ellas. Este enfoque es utilizado y mejorado en los trabajos de Ambai [AY11], Bay (SURF) [BET08], y Wagner [WRM+08]

  Una prueba binaria es una comparación de la intensidad de dos píxeles y produce un resultado binario que representa qué píxel es más brillante. Se realizan cientos de pruebas para calcular un vector de características, ya que solo una de ellas no es lo suficientemente discriminatoria. El tema de investigación principal de este enfoque es el de muestreo eficiente entre dos píxeles y es utilizado en los métodos BRIEF [CLSF10], BRISK [LCS11], y ORB [RRKB11].

- **Búsqueda de Coincidencias:** Los vectores de características se almacenan en una base de datos. Cuando se busca una coincidencia, se accede a la base datos con los datos del vector de consulta y se devuelve el vector almacenado más similar.

  Si un vector de características es de grandes dimensiones como en SIFT [Low04], la búsqueda completa no se podría realizar para tareas en tiempo real. En lugar de ello, se utilizan técnicas de árboles de búsqueda basado en la aproximación al vecino más cercano [AMN+98] [ML09]. El costo de la búsqueda de este enfoque depende del número de características almacenados en la base de datos. Otro tipo de búsqueda consiste en

mapear el vector de características a un índice de tipo entero [DIIM04] y almacenarlo en una tabla hash. Este enfoque es teóricamente rápido con $O(1)$, independientemente del tamaño de la base de datos, pero es sensible a errores. Si un vector de características se describe como cadena binaria, se realizará una búsqueda completa en toda la tabla [CLSF10]. Para intentar corregir esto, se han planteado la utilización de árboles aleatorios [LF06] y estructuras no jerárquicas [OCLF10].

Hay muchas clases de detectores de características, pero los más utilizados son los detectores de esquinas (*corners*), de bordes (*edges*) y de regiones (*blobs*).

Detector de esquinas de Harris

El ejemplo más obvio de un punto de interés es una esquina, o la intersección de dos bordes. A lo largo de los años se han desarrollado una serie de algoritmos de detección de esquinas. La mayoría de los algoritmos de detección de puntos de interés calculan una función C de respuesta, ya sea para todos los píxeles, o sólo algunos píxeles seleccionados.

Probablemente el más famoso detector de esquinas se conoce como ***detector de esquinas de Harris*** desarrollado por Harris [HS88], que fue creado para la interpretación 3D de secuencias de imágenes. Al igual que con el detector de bordes de Canny, el cálculo de la segunda derivada parcial de una imagen en una dirección específica indicará regiones con cambios bruscos de intensidad de la imagen (discontinuidades) en esa dirección. Dada una imagen en escala de grises en dos dimensiones, I, de modo que la intensidad de la imagen en un píxel específico se da por $I(x, y)$, Harris determina C_{Harris} para un punto específico de la siguiente manera:

En primer lugar, tomamos una región de la imagen encima del área (u, v) y cambiándolo por (x, y). La forma de la región puede ser circular o rectangular. La suma ponderada de las diferencias de los cuadrados (SSD) entre estas dos regiones ,denotada por S y viene dada por:

$$S(x, y) = \sum_u \sum_v w(u, v)\ \left(I(u + x, v + y) - I(u, v)\right)^2 \tag{4.20}$$

$I(u+x, v+y)$ puede aproximarse por una serie de Taylor. Donde I_x y I_y son los gradientes vertical y horizontal de la imagen (derivadas parciales de I), tenemos:

$$I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y \tag{4.21}$$

Esto produce la aproximación:

$$S(x, y) \approx \sum_u \sum_v w(u, v)\ \left(I_x(u, v)x + I_y(u, v)y\right)^2, \tag{4.22}$$

que se puede escribir en la forma de matriz:

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} \mathbf{H}_{harris} \begin{pmatrix} x \\ y \end{pmatrix}, \tag{4.23}$$

donde $\mathbf{H}_{harris}$ es la *structure tensor*,

$$\mathbf{H}_{harris} = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{4.24}$$

$\mathbf{H}_{harris}$ es conocida como la *Matriz de Harris*. Un punto de interés se caracteriza por una variación de S en todas la direcciones del vector (x, y). Si definimos λ_1 y λ_2 como los autovalores de la matriz descrita arriba, podemos definir entonces la función de autocorrelación C_{harris} como:

$$C_{harris} = \lambda_1 \lambda_2 - \kappa \left(\lambda_1 + \lambda_2\right)^2 = \det(\mathbf{H}_{harris}) - \kappa \ \mathrm{trace}^2(\mathbf{H}_{harris}) \tag{4.25}$$

En función de los autovalores de $\mathbf{H}_{harris}$, λ_1 y λ_2, podremos distinguir los siguientes casos:

- La función tendrá un máximo si ambos autovalores son altos, lo que quiere decir que un desplazamiento en cualquier dirección va a producir un incremento importante, indicando, por tanto, que se trata de una esquina.

- La función sera casi nula si ambos autovalores son bajos, es decir que un desplazamiento en cualquier dirección va a producir cambios muy pequeños, por tanto, la región que engloba la subventana es de intensidad constante (pertenece al mismo objeto).

- Si un autovalor es alto y el otro bajo, entonces, la función tendrá forma de cresta. Por tanto, sólo los desplazamientos en una dirección van a producir pequeños cambios en C(x,y) y cambios significativos en la dirección perpendicular. Esto indicará la presencia de un borde.

El valor de κ es un parámetro experimental y es determinado empíricamente del rango de 0.04 a 0.15.

Shi-Tomasi

Shi y Tomasi [ST94] mejoraron el detector de Harris y encontraron que se obtenía un buen resultado en la detección de esquinas, siempre y cuando el autovalor más pequeño era mayor que un cierto umbral. $C_{ShiTomasi} = min(\lambda_1, \lambda_2)$. La función $C_{ShiTomasi}$ tiene el inconve-

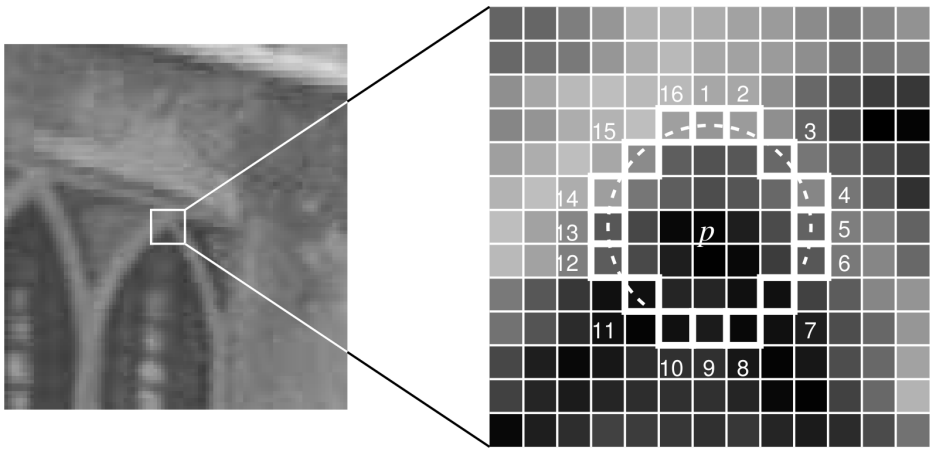


Figura 4.12: FAST: Círculo de análisis alrededor del píxel p. (Rosten y Drummond)

niente de necesitar el cálculo explícito de los autovalores, que costoso computacionalmente. Los autovalores de una matriz de 2x2 se calculan de la siguiente forma:

$$\lambda_{12} = \frac{1}{2}(trace(\mathbf{H}_{harris}) \pm \sqrt{\text{trace}^2(\mathbf{H}_{harris}) - 4\det(\mathbf{H}_{harris})}) \tag{4.26}$$

El cálculo de la raíz cuadrada puede afectar negativamente el tiempo de cálculo. Los resultados muestran que Shi-Tomasi tiene mejor criterio para determinar a un punto de interés, devolviendo más puntos que Harris para un mismo umbral. Los resultados también muestran que Shi-Tomasi es más lento que el algoritmo de Harris para el mismo umbral, posiblemente debido al cálculo de los autovalores.

FAST (Features from Accelerated Segment Test)

FAST es algoritmo de detección de esquinas presentado por Rosten y Drummond [RD05]. En contraste con los métodos de detección de puntos de interés, como Harris, FAST tiene un enfoque mucho más cualitativo, y mucho menos costoso, para determinar si un píxel representa una esquina. Para cada píxel, p, en una imagen en escala de grises, FAST examina 16 píxeles en un círculo de radio 3 alrededor de p, como se muestra en la figura 4.12. La intensidad (valor de gris) de un píxel está dado por $I(x, y)$. FAST simplemente afirma que p puede ser clasificado como un punto de interés si las intensidades de al menos 12 puntos contiguos en el círculo de prueba son más claros o más oscuros que $I(x_p, y_p)$ (la intensidad de p) para un cierto umbral, t. Un punto de interés puede ser categorizado como "positivo" (brillante) si por lo menos 12 puntos circulares contiguos tienen intensidades mayores que $I(x_p, y_p) + t$, o "negativo" (oscuro) si sus intensidades son más pequeños que $I(x_p, y_p) - t$. Esta partición puede ser útil para que los puntos positivos no sean comparado con puntos negativos en etapas posteriores del *tracking*.

Dado que los 12 puntos deben ser contiguos, la comprobación puede ser acelerada examinando inicialmente soló los píxeles 1, 9, 5 y 13. Un punto de interés sólo puede existir si tres de estos puntos de prueba son todos más brillante o más oscuro que la intensidad de p dado un valor para el umbral. Si más de uno de estos puntos de prueba está dentro del rango del umbral, p puede ser rechazado de inmediato. Si esta comprobación inicial es satisfactoria, se comprueban el resto de píxeles que quedan en el círculo. Rosten encontró que debido a esta optimización, el algoritmo examina en promedio sólo 3,8 píxeles del círculo para probar si un punto dado es un punto de interés.

De la descripción del algoritmo FAST se puede observar dos potenciales ventajas sobre otros detectores de puntos de interés:

- **Rapidez.** Todo lo necesario para procesar una imagen completa es un promedio de 3,8 sumas enteras y comparaciones por píxel. En comparación con otros detectores que requieren el cálculo de derivadas parciales, raíces cuadradas y circunvoluciones, FAST es mucho menos complejo computacionalmente.

- **Umbralización simple.** El valor del umbral, t, es un valor entero que oscila entre 0 y 255 para una imagen en escala de grises de 8 bits. La elección de t influirá directamente en el número, y, posiblemente en la calidad de los puntos de interés detectados. Para aplicaciones en las que es necesario tener un número constante de puntos de interés que deban ser detectados, t puede ser determinado dinámicamente.

Se debe tener en cuenta que estas características conllevan una serie de desventajas en el uso de FAST:

1. FAST funciona mal en imágenes borrosas. La mayoría de los detectores de puntos de interés basados esquinas también tienen este inconveniente.

2. FAST no es muy robusto en imágenes que presenten ruido. Se obtiene una alta velocidad de proceso a costa de evaluar una menor cantidad de píxeles por punto, por lo que no tiene la capacidad para promediar el ruido como en otros detectores que analizan las regiones ponderadas por filtros gaussianos.

3. Pueden ser detectados múltiples puntos de interés juntos.

El tercer inconvenientes puede ser problema, especialmente si FAST se utiliza para detectar los puntos de interés con los que posteriormente se va realizar un seguimiento (*tracking*). FAST puede detectar múltiples «puntos de interés» alrededor de una esquina en una imagen si está es mayor de uno o dos píxeles.

En general, es preferible realizar un seguimiento de puntos que se encuentren separados al menos a una cierta distancia entre sí, de modo que las regiones utilizadas para la evaluación no se superpongan.

FAST prevé este requisito, ofreciendo un paso opcional después de un punto de interés se encuentra el candidato conocido como supresión de no máximos *(non-maximal suppression)*. La supresión de lo no máximos funciona calculando una función de puntuación, V , para cada punto de interés candidato sumando la diferencia de intensidad entre $I(x_p, y_p)$ y cada uno de los píxeles contiguos en el círculo de la prueba que son más o menos brillante que $I(x_p, y_p)$ dado el umbral t (aquellos píxeles que pasaron la prueba de 12 puntos). Con la opción de supresión de los no máximos (non-maximal suppression) habilitada, sólo el candidato con la puntuación más alta en una región local es aceptado como un punto de interés. Rosten [RD05] da la siguiente definición de una función de puntuación V:

$$V = max \left(\sum_{u \in S_{bright}} |I(x_u, y_u) - I(x_p, y_p)| - t, \sum_{u \in S_{dark}} |I(x_p, y_p) - I(x_u, y_u)| - t \right) \tag{4.27}$$

donde S_{bright} y S_{dark} son los conjuntos de píxeles contiguos alrededor de p que son más brillantes u oscuro que p dado el umbral t. Se definen como:

$$S_{bright} = u|I(x_u, y_u) \geq I(x_p, y_p) + t \tag{4.28}$$

$$S_{dark} = u|I(x_u, y_u) \leq I(x_p, y_p) - t \tag{4.29}$$

La eliminación de los no máximos puede incurrir en una pequeña penalización en el rendimiento en la fase de detección, pero puede conducir a grandes ahorros en los requerimientos computacionales en la función de definición y en la etapa de *matching*.

Detección de bordes

Muchos de los algoritmos de *tracking* basado en modelos buscan coincidencia entre los bordes de un objeto del mundo real con los bordes de un modelo conocido para determinar la *pose*.

La base de estos métodos es el cálculo de una característica geométrica a partir de los bordes. En el caso del trabajo de Hagbi [HBESB11], propone un método para reconocer formas planas y estimar la *pose* de la cámara a partir del contorno de las formas. Para cada concavidad del contorno, se extraen de las líneas bitangentes, puntos clave invariantes de la proyección. La *pose* inicial se estima a partir de estos puntos y son refinados mediante la minimización del error de reproyección.

Existen gran variedad de algoritmos de detección de bordes, pero uno de los más populares y de mayor éxito es un método perfeccionado por Canny [Can86]. La figura 4.3 se muestra el resultado de la aplicación del detector de bordes de Canny. Como se puede ver en esta

figura, la forma de un modelo se puede estimar mejor si podemos detectar sus los contornos (tanto exteriores como interiores). El detector devuelve un conjunto de curvas conectadas. El *tracking* de contornos relacionados con la estructura de un objeto de la imagen en lugar realizar un seguimiento de los píxeles individuales reduce significativamente la cantidad de datos a procesar, que es exactamente el propósito de la detección de características.

El **algoritmo de Canny** realiza la siguiente secuencia de pasos para realizar la detección de bordes:

1. La imagen se difumina ligeramente utilizando un filtro de Gauss (desenfoque) para reducir el ruido que pueden estar presente debido al sensor de imagen (calidad del sensor, aumento de sensibilidad).

2. Se calcula el gradiente de intensidad de la imagen mediante las primeras derivada parciales respecto a x e y.

3. Un cambio brusco en la intensidad de la imagen podría representar un borde. La búsqueda de estas áreas se realiza buscando puntos en los que las derivadas direccionales son máximos locales.

4. Teniendo en cuenta los puntos anteriores como candidatos para bordes, el área alrededor de cada punto se inspecciona para determinar si en realidad se encuentra en un borde y la dirección del gradiente. Esta etapa se denomina supresión de los no máximos (*non-maximal suppression*) y devuelve el conjunto de puntos del borde.

5. Los bordes se trazan utilizando umbral de histéresis. La umbralización consiste en imponer un valor umbral, si los píxeles superan ese umbral serán considerados como bordes. Pero aparece un problema si imponemos un umbral muy alto perdemos parte de los bordes, por el contrario si usamos un umbral bajo aparecería ruido, por ello utilizamos la umbralización con histéresis en la que usamos dos umbrales T_h y T_l, siendo T_h mayor que T_l. Después de este paso,se devuelve una imagen binaria donde los píxeles pertenecientes a los bordes están representados por unos y el resto por ceros.

La imagen binaria devuelta puede procesarse adicionalmente para encontrar curvas y polígonos, pero esto ya no es parte de la etapa de detector de bordes de Canny.

Detección de Regiones

Podemos definir una región *(blob)* como una zona de la imagen que tiene propiedades distintas, como el brillo o el color, comparada con otras zonas que la rodean.

La investigación en este área se centra en algoritmos que podemos agrupar en dos clases principales de detectores de regiones: métodos diferenciales y métodos basados en extremos locales.

Donoser [DKB11] presenta un enfoque para extraer contornos que utiliza **MSER** (Maximally Stable Extremal Regions) para definir regiones de interes *(blob)*.

El **detector de regiones extremas MSER** (Maximally Stable Extremal Regions) selecciona aquellas regiones cuyos píxeles son más brillantes o más oscuros que todos los píxeles de su alrededor [MCUP02]. De este modo, las regiones quedan definidas por una propiedad extrema de la función de intensidad en la región y su límite exterior. El algoritmo ordena todos los píxeles de la imagen según su intensidad con un coste computacional de $O(n)$ si el rango de los valores de la imagen S es pequeño, por ejemplo los valores de la imagen en escala de grises $\{0, \ldots, 255\}$, ya que la ordenación se implementa mediante un algoritmo BIN-SORT. Después de la ordenación, los píxeles se colocan en la imagen (con orden ascendente o descendente) y se van creando regiones de componentes conectados que van creciendo y fusionándose (utilizando un algoritmo de tipo *union-find*, con una complejidad temporal de $O(n\ \log(\log n))$) hasta que todos los píxeles han sido seleccionados. Como resultado final se obtienen diferentes funciones de densidad que describen estructuras de datos que representan a cada una de las áreas de los componentes conectados que contiene la imagen.

Las regiones extremas se calculan buscando aquellos componentes conectados que permanecen constantes durante un número determinado de iteraciones. Para ello, se seleccionan como umbrales de las regiones extremas para las diferentes iteraciones los niveles de intensidad que son mínimos locales del rango de cambio de la función del área. Finalmente, el algoritmo transforma en elipses las regiones extremas, que inicialmente pueden tener cualquier forma.

4.4.3 Descriptores de Características

La búsqueda de correspondencias de punto es una tarea importante para llevar a acabo una triangulación y el cálculo de la *pose* de la cámara con éxito. Con el fin de encontrar características correspondientes más fotogramas de vídeo que debe ser posible detectar características como se describe en la sección anterior , pero también es importante para identificar las características y relacionarlas entre fotogramas. Llamamos a esta descripción de la función y se trata de extraer información de la característica.

Al igual que con la detección de características , una amplia variedad de algoritmos de descripción de funciones se han presentado en los últimos años . Una buena descripción del carácter debe exhibir estas tres características :

1. **Reproducible** Los descriptores de características deben ser fiables, encontrando los mismos puntos de interés bajo diferentes condiciones de visualización. Debe tener una alta precisión y una tasa baja de falsos positivos. También debe ser invariante a cambios en la rotación, traslación y escala.

2. **Robusto** El descriptor debe ser capaz de identificar el mismo punto entre los distintos frames, incluso si hay cambios en la iluminación, se presente ruido en la imagen o existan pequeños cambios en el punto de vista .

3. **Rápido** Debe ser capaz de extraer información de los puntos característicos y compararla con una gran base de datos en el menor tiempo posible, preferiblemente en tiempo real.

Lowe [Low04] presenta un método para la extracción de características de la imagen llamado SIFT *(Scale Invariant Feature Transform)*. Este proceso es invariante a cambios en la escala de imagen, traslación y rotación, así también, al menos parcialmente invariante, a cambios en la iluminación y transformaciones proyectivas 3D. Este enfoque transforma una imagen en una gran colección de vectores de características locales llamados *«SIFT Keys»* que se utilizan para su identificación.

SURF

El descriptor SURF, *Speeded-Up Robust Features*, fue desarrollado por Herbert Bay[BET08] como un detector de puntos de interés y descriptores robusto. El descriptor SURF guarda cierta similitud con la filosofía del descriptor SIFT [Low04], si bien presenta notables diferencias que quedarán patentes con la siguiente exposición sobre su desarrollo. Los autores afirman sin embargo que este detector y descriptor presentan principalmente 2 mejoras resumidas en los siguientes conceptos:

- Velocidad de cálculo considerablemente superior sin ocasionar perdida del rendimiento.

- Mayor robustez ante posibles transformaciones de la imagen.

Estas mejoras se consiguen mediante la reducción de la dimensionalidad y complejidad en el cálculo de los vectores de características de los puntos de interés obtenidos, mientras continúan siendo suficientemente característicos e igualmente repetitivos.

Las diferencias más originales respecto del descriptor SIFT:

- La normalización o longitud de los vectores de características de los puntos de interés es considerablemente menor, concretamente se trata de vectores con una dimensionalidad de 64, lo que supone una reducción de la mitad de la longitud del descriptor SIFT.

- El descriptor SURF utiliza siempre la misma imagen, la original.

- Utiliza el determinante de la matriz Hessiana para calcular tanto la posición como la escala de los puntos de interés

Detección de puntos de interés La primera de las etapas del descriptor SURF es análoga a la del descriptor SIFT en cuanto a la detección de puntos de interés se refiere, si bien el procedimiento para su obtención se basa en diferencias sustanciales que se detallan a continuación.

El descriptor SURF hace uso de la matriz Hessiana, más concretamente, del valor del determinante de la matriz, para la localización y la escala de los puntos. El motivo para la utilización de la matriz Hessiana es respaldado por su rendimiento en cuanto a la velocidad de cálculo y a la precisión. Lo realmente novedoso del detector incluido en el descriptor SURF respecto de otros detectores es que no utiliza diferentes medidas para el cálculo de la posición y la escala de los puntos de interés individualmente, sino que utiliza el valor del determinante de la matriz Hessiana en ambos casos. Por lo tanto dado un punto $p = (x, y)$ de la imagen I , la matriz Hessiana $H(p, \sigma)$ del punto p perteneciente a la escala σ se define como:

$$H(p, \sigma) = \begin{bmatrix} L_{xx}(p, \sigma) & L_{xy}(p, \sigma) \\ L_{xy}(p, \sigma) & L_{yy}(p, \sigma) \end{bmatrix} \tag{4.30}$$

donde $L_{xx}(p, \sigma)$ representa la convolución de la derivada parcial de segundo orden de la Gaussiana con la imagen I en el punto p. De manera análoga ocurre con los términos $L_{xy}(p, \sigma), L_{yy}(p, \sigma)$ de la matriz.

A pesar de que los filtros gaussianos son óptimos para el análisis del espacio-escala, se ha implementado una alternativa a los filtros gaussianos en el detector SURF debido a una serie de limitaciones de estos filtros (como la necesidad de ser discretizados, la falta de prevención total del indeseado efecto aliasing, etc.): los filtros tipo caja (de sus siglas en inglés box-filters). Estos nuevos filtros aproximan las derivadas parciales de segundo orden de las gaussianas y pueden ser evaluados de manera muy rápida usando imágenes integrales, independientemente del tamaño de éstas. Las imágenes integrales, son calculadas mediante la siguiente fórmula:

$$I_{\sum}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{4.31}$$

donde (x, y) representan la posición del punto en la imagen y $Ii(x, y)$representa la intensidad de la imagen en el punto. Una vez la imagen integral ha sido creada, se puede calcular la suma de las intensidades de una región mediante una simple operación:

$$\sum I = I_{iD} + I_{iA} + I_{iB} + I_{iC} \tag{4.32}$$

De esta forma, el tiempo necesario para el calculo de las operaciones de convolución es independiente del tamaño de la imagen.

El espacio escala del descriptor SIFT descrito anteriormente, se crea a partir de imágenes suavizadas repetidamente mediante la aplicación de un filtro gaussiano y que posteriormente se submuestrean para alcanzar un nivel más alto dentro de la pirámide de dicho espacio escala. Sin embargo en el caso del detector SURF, debido a la utilización de filtros de tipo caja e imágenes integrales, no es necesario aplicar el mismo filtro iterativamente a la salida de una capa filtrada previamente, sino que se pueden aplicar dichos filtros de cualquier tamaño a la misma velocidad directamente sobre la imagen original. De este modo resulta que el espacio escala es analizado mediante la elevación del tamaño del filtro, en vez de reducir el tamaño de la imagen como es el caso del detector SIFT

Las aproximaciones de las derivadas parciales se denotan como D_{xx}, D_{xy}, yD_{yy}. En cuanto al determinante de la matriz Hessiana, éste queda definido de la siguiente manera:

$$det(H_{aprox}) = D_{xx}D_{yy} - (0,9D_{xy})^2 \tag{4.33}$$

donde el valor de 0,9 está relacionado con la aproximación del filtro gaussiano. La imagen de salida obtenida tras la convolución de la imagen original con un filtro de dimensiones 9x9, que corresponde a la derivada parcial de segundo orden de una gaussiana con $\sigma = 1,2$, es considerada como la escala inicial o también como la máxima resolución espacial ($s = 1,2$, correspondiente a una gaussiana con $\sigma = 1,2$). Las capas sucesivas se obtienen mediante la aplicación gradual de filtros de mayores dimensiones, evitando así los efectos de aliasing en la imagen.

El espacio escala para el descriptor SURF, al igual que en el caso del descriptor SIFT, está divido en octavas. Sin embargo, en el descriptor SURF, las octavas están compuestas por un número fijo de imágenes como resultado de la convolución de la misma imagen original con una serie de filtros cada más grandes. El incremento o paso de los filtros dentro de una misma octava es el doble respecto del paso de la octava anterior, al mismo tiempo que el primero de los filtros de cada octava es el segundo de la octava predecesora. De esta manera obtenemos las siguientes series de octavas con sus respectivos filtros.

Finalmente para calcular la localización de todos los puntos de interés en todas las escalas, se procede mediante la eliminación de los puntos que no cumplan la condición de máximo en un vecindario de 3x3x3. De esta manera, el máximo determinante de la matriz Hessiana es interpolado en la escala y posición de la imagen. En este punto se da por concluida la etapa de detección de los puntos de interés.

Asignación de la orientación La siguiente etapa en la creación del descriptor corresponde a la asignación de la orientación de cada uno de los puntos de interés obtenidos en la etapa

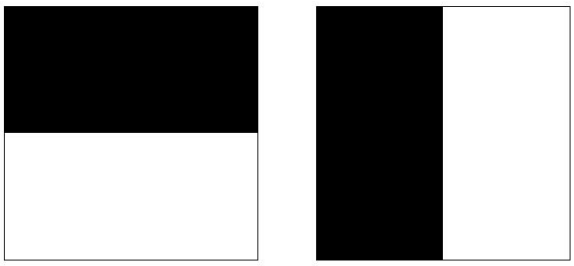


Figura 4.13: Filtros de Haar empleados en el descriptor SURF. (Bay)

anterior. Es en esta etapa donde se otorga al descriptor de cada punto la invarianza ante la rotación mediante la orientación del mismo.

El primer paso para otorgar la mencionada orientación consiste en el cálculo de la respuesta de Haar en ambas direcciones x e y mediante las funciones representadas en la figura 4.13:

El área de interés para el cálculo es el área circular centrada en el punto de interés y de radio $6s$, siendo s la escala en la que el punto de interés ha sido detectado. De la misma manera, la etapa de muestreo depende de la escala y se toma como valor s. Respecto de las funciones onduladas de Haar, se toma el valor $4s$, por tanto dependiente también de la escala, como referencia, donde a mayor valor de escala mayor es la dimensión de las *wavelets*.

Tras haber realizado todos estos cálculos, se utilizan imágenes integrales nuevamente para proceder al filtrado mediante las máscaras de Haar y obtener así las respuestas en ambas direcciones. Son necesarias únicamente 6 operaciones para obtener la respuesta en la dirección x e y. Una vez que las respuestas onduladas han sido calculadas, son ponderadas por una gaussiana de valor $\sigma = 2,5s$ centrada en el punto de interés. Las respuestas son representadas como vectores en el espacio colocando la respuesta horizontal y vertical en el eje de abscisas y ordenadas respectivamente. Finalmente, se obtiene una orientación dominante por cada sector mediante la suma de todas las respuestas dentro de una ventana de orientación móvil cubriendo un ángulo de $\pi/3$ siguiendo las especificaciones recomendadas por el autor. La orientación final del punto de interés será finalmente aquella cuyo vector sea el más grande dentro de los 6 sectores en los que han sido dividida el área circular alrededor del punto de interés.

Creación del descriptor Es en esta última etapa del proceso donde se concreta la creación del descriptor SURF. Se construye como primer paso una región cuadrada de tamaño $20s$ alrededor del punto de interés y orientada en relación a la orientación calculada en la etapa anterior. Esta región es a su vez dividida en 4x4 subregiones dentro de cada una de las cuales se calculan las respuestas de Haar de puntos con una separación de muestreo de 5x5 en ambas direcciones. Por simplicidad, se consideran d_x y d_y las respuestas de Haar en las direcciones horizontal y vertical respectivamente relativas a la orientación del punto de interés. En la

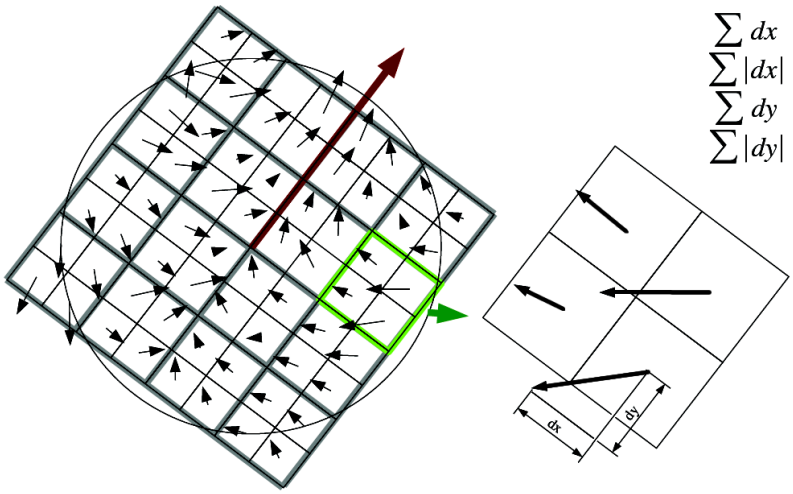


Figura 4.14: Respuestas de Haar en las sub-regiones alrededor del punto de interés. (Bay)

Figura 4.14 están representadas tanto las respuestas de Haar en cada una de las subregiones como las componentes d_x y d_y uno de los vectores. Para dotar a las respuestas d_x y d_y de una mayor robustez ante deformaciones geométricas y errores de posición, éstas son ponderadas por una gaussiana de valor $\sigma = 3,3s$ centrada en el punto de interés. En cada una de las sub-regiones se suman las respuestas d_x y d_y obteniendo así un valor de d_x y d_y representativo por cada una de las subregiones. Al mismo tiempo se realiza la suma de los valores absolutos de las respuestas $|d_x|$ y $|d_y|$ en cada una de las subregiones, obteniendo de esta manera, información de la polaridad sobre los cambios de intensidad. En resumen, cada una de las subregiones queda representada por un vector v de componentes:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{4.34}$$

y por lo tanto, englobando las 4x4 subregiones, resulta un descriptor SURF con una longitud de 64 valores para cada uno de los puntos de interés identificados.

Matching entre puntos clave Esta sección, al igual que en el caso del descriptor SIFT, representa la correspondencia de los puntos clave identificados entre dos imágenes. La estrategia utilizada para establecer las correspondencias entre los puntos clave de ambas imágenes es la de «el vecino más próximo» descrita anteriormente. En el caso del descriptor SURF, el umbral relativo es fijado con un valor de 0,7.

4.5 *Tracking*

En contraste con la detección, que estima la *pose* de la cámara en una imagen, el *tracking* es el seguimiento del objeto (y la estimación de la *pose* de la cámara respecto a ese objeto) en una secuencia de fotogramas.

Figura 4.15: Estimación de la orientación sobre puntos de interés. (Bay)

El procedimiento a seguir es la de identificar «puntos clave» visualmente significativos en un frame que podamos encontrar de forma fiable de nuevo en el siguiente. El procedimiento de *tracking* se basa en encontrar una cantidad suficiente de estas correspondencias de puntos entre los frames.

El *tracking*, es un problema complejo debido a la pérdida de información causada por la proyección del mundo 3D en una imagen 2D, la calidad de la imágenes obtenidas, fondos difíciles de segmentar, oclusiones totales o parciales, cambios en la iluminación y la exigencia de para trabajar en tiempo real.

Para construir un buen sistema de *tracking*, es deseable que cumpla con los siguientes requisitos:

- **Robusto.** Incluso en condiciones complicadas, como fondos difíciles de segmentar, cambios de iluminación, oclusiones o movimientos complejos, un algoritmo de *tracking* debe ser capaz de seguir al objeto de interés.
- **Adaptable.** Adicionalmente a los cambios de entornos que se puedan producir, el objeto en sí también puede sufrir cambios. Esto requiere que el algoritmo tenga algún mecanismo de adaptación para el seguimiento del objeto según la apariencia que tenga en cada momento.
- **Cómputo en tiempo real.** Para obtener una sensación fluida y que el ojo humano no perciba retrasos en la imagen, al menos debemos trabajar y procesar 15 imágenes por segundo. Por tanto, es necesario que el algoritmo sea rápido y esté optimizado.

Desde el punto de la información obtenida para el cálculo de la *pose* de la cámara, las técnicas de *tracking* se pueden se dividir en dos tipos:

***Tracking* por detección o búsqueda (by matching).** El cálculo de la posición y orientación de la cámara *(camera pose)* se realiza en cada frame por correspondencia entre la imagen de entrada y otra de referencia mediante una técnica de detección. La información de anteriores *«poses»* de la cámara no son tenidas en cuenta para la estimación de la nueva *pose*. El calculo de la *pose* mediante arboles aleatorios [LF06] y estructuras no jerárquicas [OCLF10] son ejemplos de este tipo de enfoque.

***Tracking* mediante *tracking*.** Para el calculo, la *pose* previa es utilizada como *pose* inicial para el cálculo de la posición y orientación actual. Una vez que se detectado un objeto, se hace un seguimiento de los puntos claves del objetos en el siguiente fotograma [WRM+08], o minimizan la diferencia entre dos imágenes consecutivas [PLW09] y analizan el cambio no lineal de iluminación producida [DM10]. La mayoría de estos enfoques se basan en la minimización del desplazamiento de la cámara entre dos imágenes sucesivas.

Otra clasificación de los métodos de *tracking* disponibles puede dividirse en dos clases: basado en detección de características y basados en detección de modelos.

4.5.1 *Tracking* por detección de características *(feature-based)*

En el mundo de la visión artificial una característica (feature) es una zona de la imagen que un algoritmo de *tracking* puede detectar y seguir a lo largo de múltiples frames. Normalmente las características suelen ser bordes, esquinas, zonas más brillantes u oscuras en función del algoritmo de *tracking* en particular.

En lugar de utilizar marcadores de referencia, la estimación de la *pose* de la cámara se puede realizar mediante la extracción de características naturales, como puntos, líneas, bordes o texturas. Esta línea de investigación también ha sido ampliamente estudiado. Park [PYN99] presenta un método en el que utiliza las características naturales como una extensión en el *tracking* de características artificiales. Después de realizar el cálculo de la estimación de la *pose* de la cámara mediante características visuales conocidos, el sistema sistema adquiere dinámicamente características naturales adicionales y los utiliza para la actualización continua de la estimación de la nueva *pose*. De esta manera proporciona un seguimiento robusto, incluso cuando las marcas originales originales ya no están a la vista.

Optical Flow

El *Optical Flow* o flujo óptico juega un papel importante en la estimación y descripción del movimiento, por lo cual es comúnmente utilizado en tareas de detección, segmentación y seguimiento de objetos móviles en una escena a partir de un conjunto de imágenes.

El flujo óptico puede ser definido como el movimiento aparente de los patrones de intensidad en una imagen. La palabra aparente indica que el movimiento espacial de los objetos (campo de movimiento) puede coincidir o no con el flujo estimado. No obstante, en situaciones en las cuales el movimiento de los objetos implica un movimiento de sus patrones de intensidad en el plano imagen, el flujo óptico puede ser directamente relacionado con el movimiento de los objetos en la escena. La mayoría de las técnicas existentes para la estimación del flujo óptico se puede clasificar en 4 categorías: las basadas en gradientes espacio-temporales, las basadas en comparación de regiones, las basadas en fase y las basadas en energía.

En todas las estrategias de estimación de flujo óptico se parte de la hipótesis de que los niveles de gris permanecen constantes ante movimientos espaciales en un tiempo dado. Dicha hipótesis da lugar a la ecuación general de flujo óptico, donde $I(x, y, t)$ corresponde a la intensidad en niveles de gris del píxel (x, y) de la imagen I en el tiempo t.

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{4.35}$$

Expandiendo la ecuación anterior en series de Taylor sobre el punto (x, y, t)

$$I(x, y, t) = I(x, y, t) + dx\frac{\partial I}{\partial x} + dy\frac{\partial I}{\partial y} + dt\frac{\partial I}{\partial t} + \epsilon \tag{4.36}$$

donde ϵ contiene la información de las derivadas de orden superior. Si se asume ϵ despreciable, la ecuación de flujo óptico puede reescribirse como

$$I_x u + I_y v + I_t = 0 \tag{4.37}$$

donde (u, v), con $u = dx/dt$ y $v = dy/dt$, corresponde al vector de flujo óptico y, I_x y I_y son las derivadas parciales horizontal y vertical de la imagen, respectivamente. Para cada píxel (x, y) de la imagen, en el tiempo t, puede plantearse la ecuación anterior, sin embargo no existe una única solución para esta ecuación. Diferentes restricciones pueden emplearse para estimar el flujo óptico en la imagen. Horn y Shunck en [HS81] restringen el flujo óptico en la imagen a variar suavemente, por lo que la ecuación es minimizada junto a un término de regularización que penaliza los cambios abruptos del flujo. Lucas y Kanade (LK) [LK81] proponen un método alternativo que se describe a continuación el cual puede ser implementado de forma más eficiente que el propuesto en [HS81]. Las técnicas propuestas en [HS81] y [LK81] se basan en gradientes espacio-temporales pues minimizan la ecuación anterior.

Lucas-Kanade

Este método asume que el flujo óptico es constante sobre una región. Sea R una región de la imagen y (u, v) su vector de flujo óptico asociado, entonces se cumple para cada píxel de la región, es decir

$$I_x(p_i)u + I_y(p_i)v = -I_t(p_i) \quad \forall p_i \in R \tag{4.38}$$

Organizando el conjunto de ecuaciones en forma matricial

$$A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \quad d = \begin{bmatrix} u \\ v \end{bmatrix} \quad b = - \begin{bmatrix} I_t(p_i) \\ I_t(p_2) \\ \vdots \\ I_t(p_n) \end{bmatrix} \tag{4.39}$$

donde la matriz A contiene las derivadas espaciales de la imagen, el vector d corresponde al vector de flujo óptico (u, v) y el vector b contiene las derivadas temporales de la imagen.

Pre-multiplicando (3) por la transpuesta de A se tiene

$$A^T Ad = A^T b \tag{4.40}$$

donde el vector de flujo óptico es encontrado como

$$d = (A^T A)^{-1} A^T b \tag{4.41}$$

El cálculo del flujo óptico implica la inversión de la matriz

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{bmatrix} \tag{4.42}$$

por la cual la solución existe si la matriz $A^T A$ es invertible y bien condicionada. Shi y Tomassi definen en [13] las propiedades que debe cumplir una región para que el flujo óptico se estimado apropiadamente utilizando la técnica de LK. Sean λ_1 y λ_2 los valores propios de la matriz $A^T A$ para cierta región R de la imagen, entonces se debe cumplir que:

- $min(\lambda_1, \lambda_2) > \lambda_{min} \in R^+$, lo cual garantiza que $A^T A$ es invertible y la región no es ruidosa.
- $\lambda_1/\lambda_2 < \tau$, lo que garantiza que $A^T A$ está bien condicionada y no se presenta bordes en una sola dirección

Bajo estas 2 condiciones el flujo óptico puede ser apropiadamente estimado. En la práctica existen ciertos factores que pueden inducir errores en la estimación. Entre ellos se encuen-

Figura 4.16: Lucas-Kanade: Seguimiento de puntos. (David Stavens)

tran la variación temporal de los niveles de gris sobre la región, desplazamientos grandes de la región entre las imágenes consecutivas e incoherencia de movimiento. El primero es independiente de la técnica de LK pero los otros 2 factores pueden ser controlados seleccionando un tamaño apropiado para la región R. Una región pequeña en comparación al tamaño del objeto garantiza una consistencia en el movimiento de las intensidades de gris, sin embargo si el objeto se desplaza rápidamente, éste puede salir de la región lo que produce un error en la estimación del flujo óptico. Por tal motivo existe un compromiso en el tamaño de la región R, el cual puede ser manejado con una implementación piramidal que estime secuencialmente el flujo en diferentes escalas.

En [Bou00] se presenta una implementación piramidal de la técnica de LK en la cual el flujo óptico es calculado recursivamente sobre versiones de diferentes escalas de las imágenes. En principio el flujo es estimado sobre imágenes en una escala baja para permitir grandes desplazamientos, posteriormente la escala se reduce para realizar una estimación más precisa y evitar inconsistencias de movimiento. El tamaño de la región se mantiene fijo sobre todas las escalas.

4.5.2 *Tracking* por detección modelos *(model-based tracking)*

La tendencia más reciente en las técnicas de *tracking* es el basados en modelos. Estas técnicas utilizan explícitamente un modelo de las características de los objetos rastreados, como por ejemplo, un modelo CAD ó un patrón del objeto basado en sus características dis-

tinguibles. El primer trabajo basado en modelos fue obra de Comport [CMC03] que en 2003 que utilizó esta aproximación utilizando las características geométricas de líneas, círculos, cilindros y esferas del modelo para el cálculo de la *pose* de la cámara.

4.6 Reconocimiento y análisis de documentos

En general, podemos considerar que cualquier escena o imagen que tenga un contenido textual como si fuera un documento. Esto incluiría tanto un libro, la matricula de un vehículo o un cartel en una pared. La mayoría de trabajos mediante cámaras están basados en la extracción de texto en imágenes fijas o secuencias de vídeo en las que los autores denominan imágenes naturales, en lugar de imágenes donde el texto está estructurado como los documentos. Ambos enfoques tienen sus desafíos y distintos modos de acometerlos, pero el objetivo final de todos es la de dotar a las cámaras la capacidad de lectura.

En el caso de documentos estructurados, que es el del dominio de este trabajo, las imágenes suelen ser documentos impresos como artículos, cartas, formularios o páginas de libros, donde gran parte de la imagen se asume que es texto, pero también puede contener figuras, diagramas e incluso algunos autores han tratado con anotaciones escritas a mano [CCSC13]

4.6.1 Identificación y recuperación de documentos

Aunque hoy en día la mayor parte de la producción de información en forma de documentos se realiza por medio de herramientas informáticas (procesadores de texto, correo electrónico, etc.), puede ocurrir, y de hecho será un caso muy habitual, que la información no se restrinja a documentos actuales, ya automatizados, sino que se encuentre impresa. Incluso puede ocurrir que nos interese disponer sólo de la información antigua (archivos y manuscritos).

En estos casos para conseguir una gestión eficaz y ágil, es necesario digitalizar previamente estos documentos para incorporarlos al sistema que tenga implementado la organización.

Las primeras aplicaciones se basaban en el paradigma de reconocimiento de caracteres (OCR), donde se utilizaban estas técnicas para realizar un análisis del contenido informativo de los documentos y utilizarlo para su clasificación y almacenamiento.

La recuperación de objetos (también nombrada por otros autores reconocimiento o identificación) se incorpora recientemente en la detección de tal manera que un objeto es capturado en una imagen, recuperado de una base de datos y su *pose* inicial se calcula simultáneamente [PS10]

El desarrollo de la investigación realizada en este ámbito se inició con los métodos que utilizaban marcas especiales en el documento, como códigos de barras [GEHL03] o glifos

para vincular contenido electrónico con las imágenes capturadas [Hec01]. Los inconvenientes de estos enfoques es que es necesario modificar el formato y la apariencia del documento para introducir las marcas, que en algunos casos pueden distraer al usuario del contenido del documento. Por otro lado, un documento válido para el sistema al que no se le hayan incluido previamente estas marcas, no será detectado y vinculado con la información a recuperar.

La utilización del teléfono móvil y otros dispositivos portátiles para la identificación de documentos ha hecho que publiquen diversos artículos con algoritmos y métodos que parten de las propias limitaciones de estos dispositivos como es la baja capacidad de computo, la calidad de las imágenes capturadas, en muchos casos borrosas, y la captura parcial del documento.

Las técnicas de identificación y recuperación de documentos se pueden dividir en dos categorías [PLC03]: basados en la búsqueda de coincidencias de características locales y las que, además de lo anterior, utilizan la distribución del contenido dentro de la página.

Basados en la búsqueda de coincidencias de características locales (feature matching)

Algunos autores proponen técnicas de detección basados en la extracción de características invariante similares a SIFT [Low04] o SURF [BET08] en trabajos como [YSSlT07] [PAK10] [BL07] aplicadas a imágenes naturales y en alta resolución. Sin embargo, como indican Uchiyama y Saito [UM11], estos métodos no funcionan correctamente para la identificación de documentos, ya que no presentan zonas de textura y se producen patrones binarios repetitivos (el texto de un documento cumple esta disposición).

Mediante una variación del proceso inspirado en la metodología de Lowe [Low04] y Bay [BET08], Augereau [AiJD13] obtiene buenos resultados sobre documentos semi-estructurados (billetes de tren, tickets,...) realizando una selección de los puntos extraídos y una adaptación del algoritmo RANSAC para la validación de supuestos aciertos en la comparación.

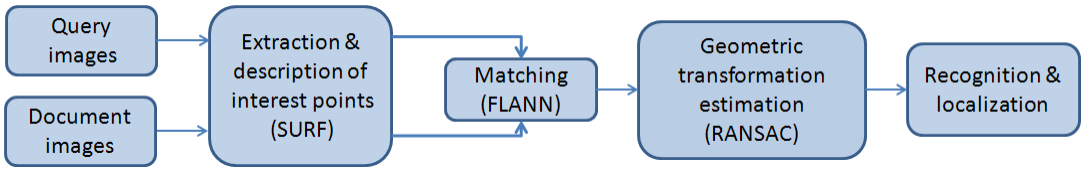


Figura 4.17: Proceso habitual para reconocimiento de imágenes mediante descriptores SURF (Augereau)

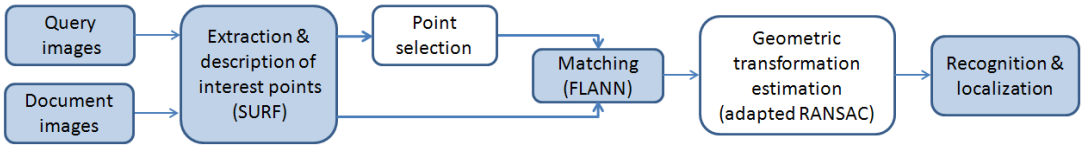


Figura 4.18: Proceso adaptado para reconocimiento de documentos mediante SURF (Augereau)

El algoritmo tiene una alta tasa de recuperación y precisión, es robusto a las deformaciones que pueda tener la imagen (perspectiva) y no necesita ningún paso previo de segmentación, pero no funciona ante grandes secciones de texto y la identificación es para obtener documentos similares (no idénticos) a la imagen utilizada como consulta.

En PaperUI, Liu y Liao [LL12] implementan hasta siete enfoques distintos para identificar un documento: códigos de barras, micro-patrones ópticos, codificación oculta, huella digital, reconocimiento óptico de caracteres, detector de características locales como SIFT y FIT (una variación de SIFT), incluso tecnología RFID. Sin embargo PaperUI no es capaz de manejar bases de datos de gran tamaño porque el almacenaje de los vectores de características de SIFT requiere gran cantidad de memoria.

Basados en combinación coincidencias de características locales y la distribución del contenido dentro de la página

Para superar las limitaciones, que tienen los enfoques anteriores en el uso específico de imágenes de documentos, otros autores proponen metodologías diseñadas para la recuperación de documentos, haciendo uso explícito de las características inherentes de la estructura del documento y el texto que contiene.

En su articulo, Liu y Doermann [LD08], presentan un método de recuperación basado en pares y tríos de tokens. En este método, se captura una página mediante un teléfono móvil y se envía a un servidor para recuperar el documento correspondiente. La aplicación ha sido desarrollada para trabajar con bases de datos de gran tamaño, sin embargo, se necesita mucho tiempo de procesamiento, alrededor de 4 segundos por consulta. Este tiempo de respuesta supone un punto bastante negativo para considerarlo introducir en aplicaciones en tiempo real.

HotPaper de Erol [EAH08], es otro método que utiliza las características locales extraídas del texto del documento y su distribución denominado *Brick Wall Coding Features (BWC)*. BWC define una característica local mediante la delimitación de palabras. Esta codificación es invariante a cambios de escala y robusta ante ligeras distorsiones de la perspectiva. El tiempo de procesamiento es rápido, alrededor de 300 ms por consulta y puede reconocer documentos a partir de imágenes con tan solo 4-5 líneas de texto y tamaños de imagen de 176 x 144. Como inconvenientes presenta problemas de escalabilidad, ya que el tamaño de la base de datos es muy pequeña, menos de 5000 páginas y la tasa de precisión es sólo alrededor del 60 %.

Utilizando el mismo enfoque, pero utilizando otra codificación para generar los descriptores a partir de los *bounding boxes*, Moraleda [Mor12] mejora la precisión hasta casi el 90 % y la escalabilidad para poder trabajar hasta con 500.000 documentos almacenados.

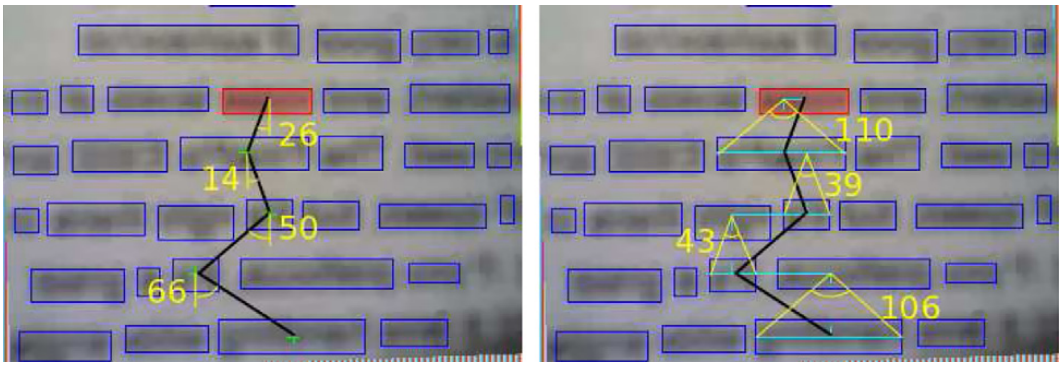


Figura 4.19: BWC: *Bounding Boxes* en un fragmento de texto (Moraleda)

Nakai y Kise proponen un método llamado *Locally Likely Arrangement Hashing (LLAH)*, que utiliza el centro de una palabra como un punto característico y calcula descriptores locales basados en estos puntos[NKI09]. LLAH tiene una alta escalabilidad y el esquema de indexación y recuperación empleado es extremadamente rápido. Los autores han confirmado que LLAH tiene una tasa de acierto del 99 % y con un tiempo de procesamiento de 50 ms en una base de datos de 20 millones de páginas.

En el algoritmo LLAH original, el movimiento de la cámara está restringido, ya que los cambios en el punto de vista causan variaciones en los descriptores locales. Uchiyama [US09] salva esta limitación estudiando el comportamiento de los descriptores al variar el punto de vista y actualizándolos continuamente. Como resultado, el método puede aplicarse a varias posiciones e inclinaciones de la cámara, permitiendo un movimiento de la cámara mucho más flexible. Iwata [IKN+09] extiende también el algoritmo LLAH para su utilización con imágenes parciales del documento en las que aparezcan tan sólo 4 o 5 líneas de texto. Por otra parte, a través del proceso de recuperación, LLAH puede estimar la *pose* de la imagen de búsqueda en el documento electrónico, que es muy útil para mostrar información relevante sobre el documento.

Esta técnica también ha sido aplica para el desarrollo de marcadores de puntos aleatorios [US11]

4.6.2 Locally Likely Arrangement Hashing (LLAH)

LLAH es un método ampliamente utilizado para la recuperación de imágenes de documentos. El algoritmo se ha tomado como base para otras aproximaciones, revisado y mejorado, tanto por sus autores, como por otros investigadores. En este apartado estudiaremos el algoritmo original que presentaron los autores [NKI06]

La Figura 4.20 muestra un esquema general del proceso. En la etapa de extracción, la imagen se transforma en un conjunto de puntos de características. A continuación, los puntos se pasan a la etapa de almacenamiento o de recuperación (en función de la tarea realizar). Estos pasos comparten la etapa de cálculo de características. En la etapa de almacenamiento, cada

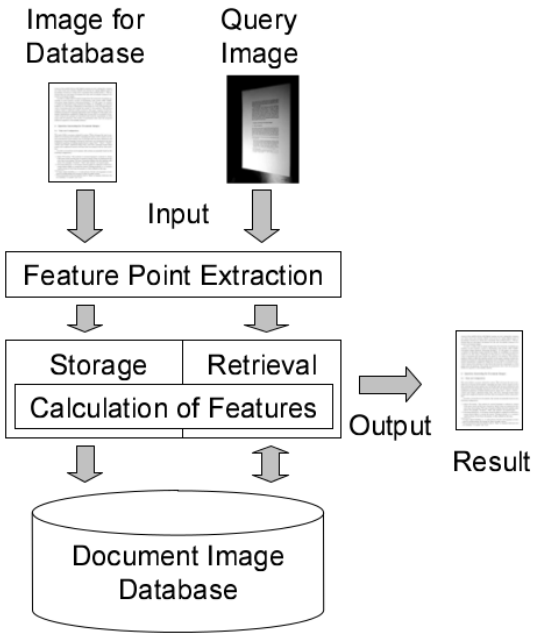


Figura 4.20: LLAH: Visión General del proceso (Nakai y Kise)

punto se almacena de forma independiente en la base de datos de imágenes de documentos usando su característica. La imagen es indexada utilizando cada uno de los puntos de características. En el paso de recuperación, se accede al documento mediante las búsqueda de los puntos encontrados.

Extracción de puntos característicos

Un requisito importante en la extracción de características es que los puntos se deben obtener de forma idéntica, bajo la misma distorsión de perspectiva, ruido y resolución. Para satisfacer este requisito, emplean como puntos característicos los centroides de las regiones que ocupan las palabras del documento.

En primer lugar, se realiza una umbralización adaptativa a la imagen de entrada (Fig.4.21(a)) y así obtener una imagen binaria (Fig.4.21(b)). A continuación, se desenfoca usando un filtro gaussiano. Seguidamente, la imagen desenfocada es umbralizada de nuevo (Fig.4.21(c)). Las regiones resultantes se supone que son espacios ocupados por palabras, y por último, (Fig.4.21(d)) se extraen los centroides de las regiones para utilizarlos como puntos característicos.

Cálculo de Descriptores locales

Los descriptores de LLAH tiene las siguientes características:

- Se define un descriptor para cada punto característico. Con el fin de obtener robustez y disponibilidad en situaciones de oclusión, un descriptor tiene una localización.

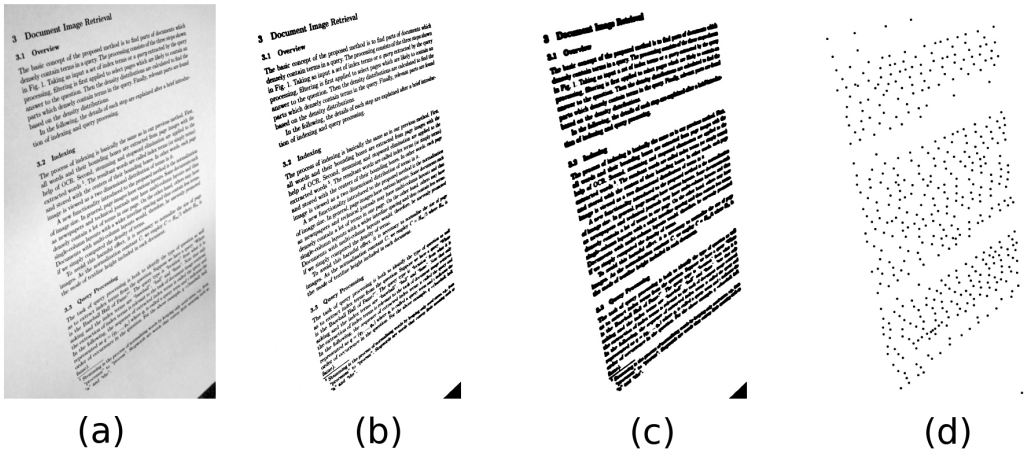


Figura 4.21: LLAH: Extracción de características (Nakai y Kise)

- Se calcula utilizando invariantes geométricos. Una invariante afín se define mediante cuatro puntos coplanares ABCD de la siguiente manera:

$$\frac{P(A,C,D)}{P(A,B,C)}$$

  donde P(A,B,C) es el área de un triángulo con vértices A, B y C.

- Un descriptor consta de más de una invariante geométrico. Para aumentar el poder de discriminación de un descriptor, se utilizan múltiples invariantes afines calculados a partir de varios puntos característicos. Cómo un invariante afín se calcula a partir de cuatro puntos, podemos calcular más de una invariante partir de más de cuatro puntos característicos. En concreto, un descriptor es $(r_{(0)}, ..., r_{{}_mC_4-1})$ calculado a partir de los m puntos contiguos donde $r_{(i)}$ es una invariante afín. Se utilizan todas las posibles combinaciones de cuatro puntos m.

- Se calculan más de un descriptor para cada punto característico. Con el fin de hacer frente a los errores de la extracción de puntos característicos,se calculan múltiples descriptores a partir $n(> m)$ puntos más cercanos. En concreto, se calculan ${}_nC_m$ descriptores, todas las combinaciones posibles de m puntos contiguos sobre n puntos totales.

Almacenamiento y Recuperación

En LLAH las imágenes se almacenan y recuperan mediante una tabla hash. En primer lugar se calculan y almacenan los descriptores de la imagen de forma preliminar. Cuando se da una imagen para recuperar, se calculan los descriptores de la consulta y se buscan y contabilizan los documentos posibles en los que coincidan el mismo descriptor. Finalmente, el documento que obtiene el mayor número de votos (descriptores encontrados) es devuelto como resultado del proceso de recuperación

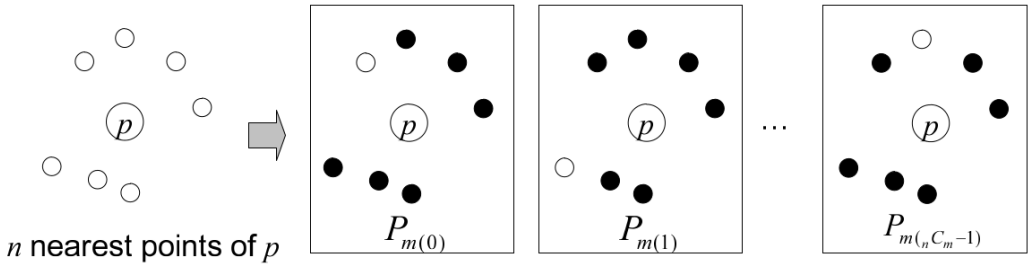


Figura 4.22: LLAH: Todas la posibles combinaciones de m(=6) puntos de los n(=7) puntos contiguos a p (Nakai y Kise)

4.6.3 Limitaciones en la identificación de documentos

El análisis de documentos mediante cámaras tiene una serie de ventajas sobre aquellos que están basados en la adquisición mediante escáner. Las cámaras son pequeñas y fáciles de transportar. También se pueden utilizar en cualquier entorno y sobre documentos que por su formato sean difíciles de manipular en un escáner como periódicos, libros, o manuscritos antiguos. Incluso para capturar texto que no se encuentra en papel, como carteles en fachadas, o texto que se encuentre el objetos que se muevan por la escena.

En la mayoría de casos, los escáneres obtienen mejor calidad en la captura de calidad que las realizadas mediante cámaras, pero los sistemas basados en cámaras son mucho más flexibles y portables.

Problemática asociada

Casi todos los algoritmos de reconocimiento de documentos obtienen grandes resultados partiendo de imágenes limpias, en alta resolución y con contrastes claramente definidos entre el texto y el fondo. Sin embargo, mediante la captura con cámaras debido a su naturaleza, a la forma en que se realiza la captura y el entorno en que nos encontremos se presentan una serie de dificultades que deben ser tenidas en cuenta.

- **Baja resolución.** Las imágenes obtenidas con las cámaras suelen estar en baja resolución, bien por las limitaciones del sensor, o por que la capacidad de computo del dispositivo que la contiene es limitada. Mientras que con un escáner es normal trabajas con una resolución de entre 150 a 600 dpi, el mismo texto en una captura con una cámara rodaría los 50 dpi.
- **Iluminación no uniforme.** La cámara, al contrario que el escáner no tiene control de la iluminación de la escena. En la captura mediante cámaras es normal encontrarse con iluminación no uniforme, varias fuentes de luz con temperaturas de color diferentes, sombras o reflejos que degradan la calidad de la imagen.
- **Distorsión por perspectiva.** Al capturar el texto sin estar la cámara paralela al plano en el que se encuentra el documento, se está produciendo una distorsión por perspecti-

va. Esto provoca que el texto presente distintos tamaños a lo largo de la imagen o que se produzca una deformación que impida el correcto reconocimiento de los caracteres.

- **Distorsión de la lente.** Las cámara incorporadas a los teléfonos móviles suelen tener una distancia focal menor para obtener un mayor ángulo de visión. La consecuencia de esto es que la lente exagera la perspectiva de los objetos, provocando mayor distorsión en las líneas cuanto más cerca se encuentre la lente del objeto.
- **Fondos complejos.** El caso ideal para la extracción de texto es que el fondo sea totalmente uniforme y con contraste diferenciado. Una mala iluminación provocará alteraciones de tono y contraste entre texto y fondo, que dificultará la segmentación el texto.
- **Zoom y autoenfoque.** Las cámaras actuales están equipadas con sistemas de zoom y autoenfoque. Una captura en la que existan distintos planos de profundidad o una mala iluminación provocará que el sistema de autoenfoque tenga dificultades para estabilizarse y durante ese tiempo las imágenes sean borrosas o fuera de foco.
- **Objetos móviles.** Por la propia naturaleza de los dispositivos móviles se entiende que o bien el dispositivo o el objeto a fotografiar está en movimiento (o incluso ambos). Si la velocidad de obturación de la cámara no es lo suficientemente rápida, la imagen obtenida estará movida.
- **Ruido del sensor.** Para compensar entornos con poca luz, las cámaras aumentan la sensibilidad amplificando la señal generada por las celdas del sensor. Como estos elementos tienen una emisión de señal de base más o menos fija, al capturar una señal lumínica débil y amplificarla, estamos amplificando también una buena porción de la emisión de datos aleatoria, con lo que se mezclará una cantidad de señal aleatoria sin contenido a la señal correspondiente a la imagen. Cuanto mayor sea la amplificación, más ruido se va a generar y peor calidad de imagen vamos a obtener.
- **Compresión de imagen.** Normalmente la imagen obtenida por el sensor se almacena comprimida mediante algoritmos con perdida de información como JPEG. La utilización de ratios altos de compresión provoca que se generen artefactos y distorsiones apreciables que restan nitidez a la imagen.
- **Algoritmos ligeros.** El objetivo final es integrar los algoritmos de análisis en los dispositivos móviles. Se deben implementar algoritmos computacionalmente eficientes ya que en la mayoría de los casos los recursos disponibles como memoria y la capacidad de computo son limitadas.

Capítulo 5

Método de trabajo

En este capítulo se describe la metodología de desarrollo aplicada, sus ventajas y motivo de elección. También se presenta la evolución del proyecto en base a la metodología empleada, los hitos conseguidos en cada fase, su complejidad y el tiempo empleado en cada una de ellas, detallando las iteraciones realizadas hasta conseguir la versión final del sistema.

Para finalizar, se listan y describen todas las herramientas utilizadas en el desarrollo, ya sean hardware o software.

5.1 Metodología del desarrollo

Para la construcción de un proyecto de cierta envergadura, como es el caso de un Trabajo Fin de Grado, la aplicación de un marco de trabajo para estructurar, planificar y controlar el proceso, es esencial para desarrollar software de calidad.

Las características del proyecto, con requisitos con posibilidad de cambios y adaptaciones a lo largo de todo el proceso, el reducido «equipo de desarrollo» o la necesidad de obtener versiones incrementales que sean testeadas y validadas por el director de proyecto, hacen que la elección se decante hacia **metodologías ágiles** de desarrollo de software.

Las **metodologías ágiles** [UG09], utilizan prácticas adaptativas (no basadas en predicciones), iterativas, centradas en personas (clientes y desarrolladores), orientadas a entregas incrementales, con mucha comunicación y necesitan que el cliente esté muy involucrado en el proyecto para recibir su *feedback*. El *feedback* continuo es indispensable para evitar que el cliente, con el software acabado, diga *«es lo que pedí, pero no es lo que necesitaba»*, algo habitual cuando se utilizan métodos clásicos.

En resumen, las principales características a las que deben dar forma las metodologías ágiles son:

- **Incremental:** Versiones pequeñas de software, con ciclos rápidos.
- **Cooperativa:** Desarrolladores y cliente siempre en contacto constante.
- **Directa:** El método es fácil de aprender, modificar y está bien documentado.
- **Adaptativa:** Son capaces de tolerar los cambios propuestos por el cliente.

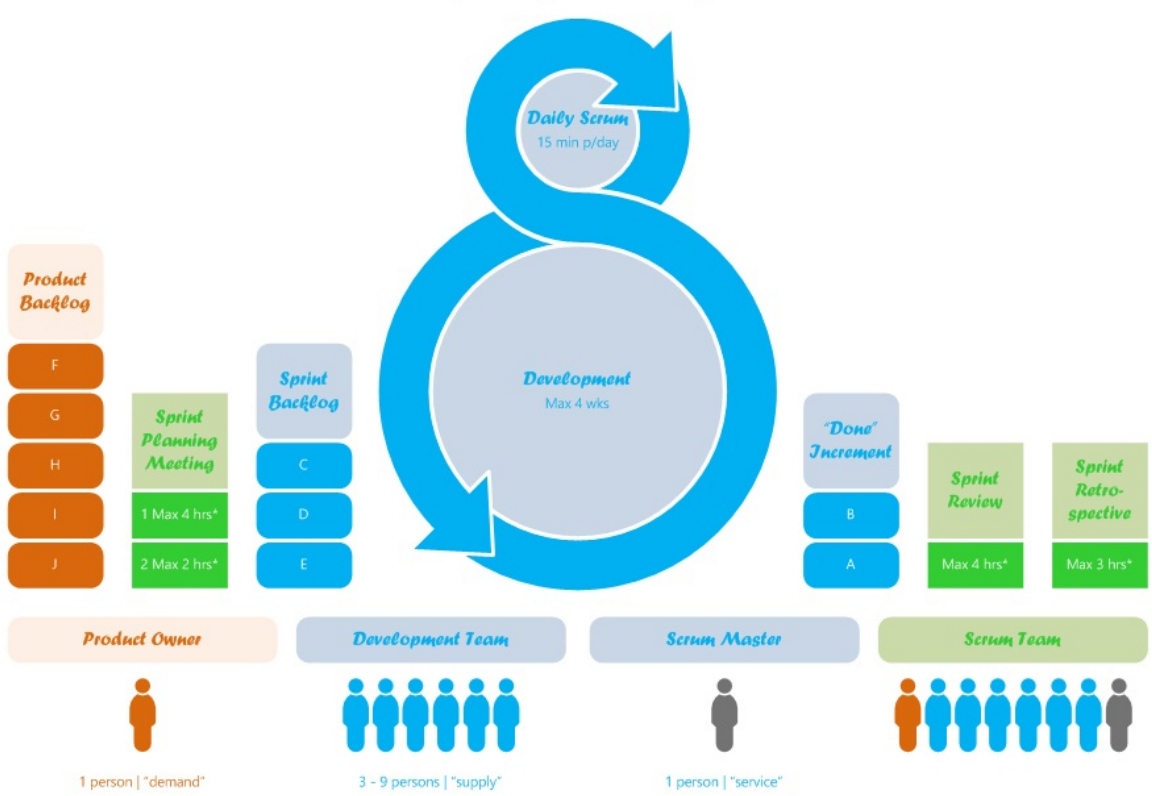


Figura 5.1: Esquema de trabajo con *Scrum*. (Mark Hoogveld)

5.2 *Scrum*

Scrum [UG09] está desarrollado para gestionar el proceso de desarrollo de sistemas aplicando ideas de flexibilidad, adaptabilidad y productividad. Sin llegar a describir ninguna técnica de desarrollo de software específica, el objetivo es definir cómo deben funcionar los miembros del equipo para que el sistema sea flexible y se adapte a condiciones altamente cambiantes.

5.2.1 Fases de *Scrum*

El proceso de *Scrum* consta de tres fases según Schwaber y Beedle [SB01]: *Pre-Game*, Desarrollo o *Game* y *Post-game*. La fase *Pre-Game* incluye dos subfases:

- ***Pre-game***
  - El **planning** incluye la definición del sistema a desarrollar y asegurar la financiación. Se crea una lista ó pila de producto, *Product Backlog List* que contiene todos los requisitos conocidos. Estos requisitos pueden ser añadidos por el cliente, los programadores o incluso otras personas relacionadas con el proyecto. Se priorizan los requisitos y se estima el esfuerzo necesario para su desarrollo. El *planning* también incluye la definición del equipo del proyecto, herramientas, valoración de riesgos y necesidades de formación.

- En la fase de **arquitectura**, se crea el diseño del sistema a alto nivel basándose en los requisitos actuales del *Backlog*. En el caso de una mejora a un sistema ya existente, se identifican los cambios necesarios, así como los problemas que puedan surgir.

- ***Game:*** Esta fase se trata como una caja negra donde se espera que ocurra lo imprevisible. Las diferentes variables técnicas y de entorno que pueden cambiar (calendario, calidad, requisitos, recursos, tecnologías, herramientas, e incluso métodos de desarrollo) se observan y controlan durante los *sprints*. En lugar de considerar estos puntos sólo al principio del proyecto, *Scrum* los controla constantemente para adaptarse a los cambios.

  Para la fase de desarrollo, *Scrum* funciona mediante lo que denomina *sprints*. Los *sprints* son ciclos iterativos donde se desarrollan o mejoran las funcionalidades para producir los nuevos incrementos. Cada *sprint* incluye las fases habituales de desarrollo del software: requisitos, análisis, diseño, desarrollo y entrega. Los *sprint* suelen tener una duración entre una semana a un mes.

- ***Post-game:*** Contiene el cierre de la versión. Se entra en esta fase cuando se completan todos los requisitos. La *release* ya está lista para lanzarse. Es en esta fase donde se integra, prueba y documenta.

5.2.2 Roles y responsabilidades

Los papeles desempeñados en *Scrum* tienen tareas y propósitos diferentes durante el proceso y sus prácticas.

- ***Scrum Master.*** Es responsable de asegurar que el proyecto se realiza según las prácticas, valores y reglas de *Scrum* y que progresa como estaba previsto. Actúa recíprocamente tanto con el equipo del proyecto como con el cliente.

- ***Product Owner.*** El propietario del producto *(Product Owner)* es oficialmente responsable del proyecto. Gestiona, controla, y administra el *Product Backlog List*. Toma las últimas decisiones de las tareas, participa estimando el esfuerzo de desarrollo para los puntos del *Backlog* y los concreta en funcionalidades a desarrollar.

- **Equipo de *Scrum.*** El equipo de *Scrum* tiene autoridad para decidir las acciones pertinentes para organizarse y lograr lo propuesto en cada *sprint*. El equipo de *Scrum* está involucrado en la estimación del esfuerzo requerido para cada parte e identificar problemas a tratar.

- **Cliente.** El cliente participa en las tareas relacionadas con los puntos del *Backlog* para diseñar o mejorar el sistema.

5.2.3 Artefactos

Documentos

- ***Product Backlog.*** El *Product Backlog* define todo lo necesario en el producto final, basándose en los conocimientos de ese momento. Por tanto, define el trabajo que se tiene que realizar en el proyecto. Incluye una lista ordenada por prioridades y actualizada de requisitos técnicos para que se realice en el sistema o mejore.

  Los elementos del *Product Backlog*, pueden incluir características, funciones, parches para *bugs*, defectos, peticiones de mejoras o actualizaciones.

  También se incluyen temas que requieren solución para poder hacer otros puntos de la lista. A la lista de *Backlog* puede contribuir el cliente, el equipo del proyecto y otras personas relacionadas con el proyecto.

- **Sprint Backlog.** Es el punto de partida de cada *sprint*. Es una selección de historias del *Product Backlog List* que se llevarán a cabo en el próximo *sprint*. El equipo de *Scrum* junto con el *Scrum Master* y el *Product Owner* seleccionan los puntos basándose en la prioridad y los objetivos. A diferencia del *Product Backlog*, el *Sprint Backlog* no se modifica hasta que el *sprint* termina.

  Cuando todos los puntos del *Sprint Backlog* se han completado, se prepara una nueva iteración del sistema. El registro que se utiliza el seguimiento, incluye valores que representan las horas de trabajo pendiente, y en función de esos valores se elabora un gráfico denominado *burndown*.

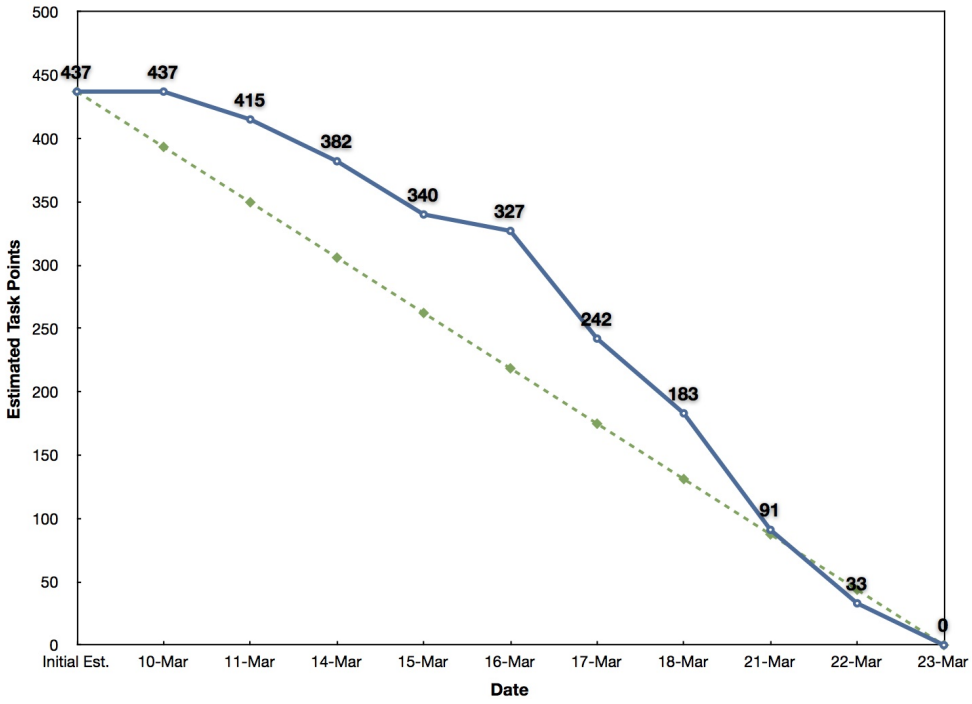


Figura 5.2: Gráfico *burndown* típico de un *sprint*

Reuniones

- **Reunión diaria de *Scrum*.** Se organizan reuniones de *Scrum* diarias para seguir el progreso del equipo, qué se ha hecho desde la última reunión y qué se hará para la siguiente. También se exponen problemas y otros asuntos que puedan aparecer, se busca y soluciona cualquier deficiencia o imprevisto del proceso. La duración de estas reuniones es de unos 15 minutos. El *Scrum Master* se encarga de dirigirlas y normalmente se suelen realizar de pie para evitar que puedan alargarse. También se valora la puntualidad de los asistentes; si alguien llega tarde, se le cobra una multa simbólica.

- **Reunión para plantear el *Sprint*.** El *sprint planning meeting* está organizado por el *Scrum Master*, y es donde se eligen los objetivos y las funcionalidades del próximo *sprint*. A continuación, el *Scrum Master* y el equipo concretan la manera de conseguir estos objetivos, lo que se denomina, *(product increment)*, en el siguiente *sprint*.

- ***Sprint Review Meeting*.** En el último día del *sprint*, el equipo y el *Scrum Master* presentan los resultados del *sprint* a la dirección, clientes, usuarios y *Product Owner* en una reunión informal. Los participantes evalúan la evolución y deciden sobre las siguientes actividades.

5.2.4 *Sprint*

Un *sprint* consiste en un ciclo iterativo donde se realiza un incremento del sistema. Está dirigido para adaptarse a las condiciones cambiantes del proyecto como requisitos, tiempo, recursos, conocimiento, tecnología, etc.

El equipo de *Scrum* se auto-organiza para producir el nuevo incremento ejecutable en aproximadamente un mes natural, que suele ser la duración habitual del *sprint*. Las herramientas activas del equipo son las reuniones para planear el *sprint*, el *sprint backlog* y las reuniones diarias de *Scrum*.

5.3 *eXtreme Programming*

Extreme Programming [UG09] o programación extrema, surgió como respuesta a la lentitud de los modelos tradicionales de desarrollo. Los orígenes de esta metodología surgieron en 1996, cuando Kent Beck comenzó a trabajar en un proyecto para reemplazar el programa de nóminas para Chrysler. Aunque las tácticas por separado que utiliza *XP* no son novedosas, la manera de unirlas sí.

En la primera edición de *XP* (1999), Beck definió 4 valores, 15 principios básicos, y 12 prácticas. Posteriormente el proceso fue revisado y se publicó en 2004 [BA04], en él se detallan 5 valores, 14 principios, 13 prácticas primarias y 11 prácticas secundarias.

5.3.1 Valores

- La mayoría de los problemas y errores provienen de la falta de comunicación. Debe haber ***comunicación*** entre los miembros del equipo y entre el equipo y los clientes. La comunicación más eficaz es la comunicación directa, interpersonal. También los artefactos deben ser fácilmente entendibles y estar actualizados.
- «Haz lo más sencillo que podría funcionar». Programar de forma sencilla, que no simplista, requiere experiencia, ideas y trabajo duro. La ***simplicidad*** favorece la comunicación, reduce la cantidad de código y mejora la calidad. La idea subyacente es que las nuevas funciones se podrán agregar cuando se necesiten si el sistema es simple.
- Siempre debería poder compararse lo que está programado con lo que se quiere programar, respecto a las funciones que se necesiten. El ***feedback*** lo proporciona el contacto con el cliente y la disponibilidad de pruebas automatizadas que se desarrollan con el propio proyecto. Cuanto más simple es un sistema, más fácil es conseguir *feedback* sobre él.
- ***Valentía***. Todos los métodos y procesos son herramientas para combatir y reducir nuestros miedos. Cuanto más miedo tengamos a un proyecto de software, mayores y más pesados serán los métodos que necesitaremos. La comunicación, la simplicidad y el *feedback* permiten adaptarse a los cambios grandes en los requisitos. También hay que tener valor para desechar código obsoleto.
- Los cuatro valores anteriores implican un quinto: el ***respeto*** entre los miembros y por su trabajo.

Los cinco valores no dan consejos específicos sobre cómo gestionar un proyecto, o cómo escribir código. Para este propósito, se utilizan las prácticas que se detallan a continuación.

5.3.2 Prácticas fundamentales

Análisis de requisitos y Planning

- Se describen todas las funciones del sistema usando **historias**, descripciones breves de funciones que el cliente podrá ver.
- El desarrollo del software se realiza **semanalmente**. Hay una reunión al principio de cada semana donde el cliente elige, según prioridades y teniendo en consideración el tiempo necesario por los programadores, las historias a programar durante la semana.
- **Evitar hacer promesas** que no puedan cumplirse.

Equipo y Factores Humanos

- Los equipos de desarrollo deben trabajar en un **espacio sin divisiones** para facilitar la comunicación.

- El equipo debe componerse de **miembros con todas las habilidades necesarias** para el proyecto, sentido de compañerismo y de ayuda mutua.
- **Programación por parejas.** El código siempre está escrito por dos programadores en una única máquina.

Diseño

- **Diseño Incremental.** *XP* se opone a un gran diseño completo inicial. El equipo escribe código lo antes posible para obtener *feedback* y mejorar el sistema continuamente. La pregunta es cuándo diseñar. *XP* sugiere hacerlo incrementalmente durante la programación.
- **Primero las pruebas.** Antes de actualizar y añadir código, es necesario escribir las pruebas para verificarlo.

Programar y lanzar versiones

- **La compilación y las pruebas automáticas deben poder finalizar en diez minutos** para ejecutarlo a menudo y obtener *feedback*.
- **Integración continua.** Los programadores deben integrar los cambios cada dos horas para evitar problemas mayores al integrar grandes partes.
- **El código y las pruebas son los únicos artefactos que se deben guardar.** Los otros documentos pueden generarse a partir del código y las pruebas.
- Cualquier miembro del equipo debe tener **acceso a todas los elementos de sistema** cuando quiera.
- **Sólo hay una versión oficial de sistema.** Se puede desarrollar una rama temporal, pero sólo usarse durante unas horas.
- **Despliegue diario.** Al finalizar la jornada se debe poner nuevo software en producción. Es arriesgado y costoso tener diferentes versiones en producción y desarrollo.

5.4 Aplicación de la metodología de desarrollo

Para la resolución de este proyecto se ha optado por una aproximación en la que se complementan las mejores prácticas y técnicas recomendadas en ***Scrum***, con sus medidas organizativas como método de gestión y ***extreme programming (XP)***, con patrones de diseño y refactorización, como metodología de desarrollo.

Se adoptarán las siguientes pautas y prácticas de ***Scrum*** a la hora de gestionar el proceso de desarrollo:

- **Equipo autodirigido** y auto-organizado.
- Una vez elegida una tarea, **no se agrega trabajo extra**.

- **Iteraciones de 30 días**; se admite que sean más frecuentes.
- **Demostración a participantes externos** al final de cada iteración.
- Al principio de cada iteración, **planificación adaptativa** guiada por el director.

El equipo de desarrollo lo han constituido el autor de este TFG y Santiago Sánchez Sobrino, con experiencia práctica y conocimientos teóricos en metodologías ágiles. Debido al tamaño del equipo y condiciones del mismo, las reuniones diarias pierden su utilidad. La figura del cliente y el director corresponde a Carlos González Morcillo, director de ambos TFG's y creador del *Proyecto ARgos*.

Al comienzo de cada iteración el equipo se reunirá y creará la lista de tareas de la iteración (*Sprint Backlog*) que consta de un subconjunto de las *historias de usuario* de la lista de objetivos (*Product Backlog*). Estas historias de usuario son seleccionadas atendiendo a las prioridades o necesidades propuestas por el Director. En esta reunión, se descompondrá cada historia en tareas, estimando el tiempo necesario para llevarlas acabo.

Las prácticas propuestas de ***XP*** que se van a utilizar son:

- **Sentarse juntos.** Todo el desarrollo se llevará a cabo en un espacio que permita un trabajo cercano, cooperativo y que facilite la comunicación directa.
- **Iteraciones cortas.** Al trabajar con pequeñas iteraciones, se obtiene el *feedback* del cliente con mucha frecuencia. Con esto se pretende que el producto final cubra ampliamente sus expectativas y necesidades.
- **Integración continua.** No se utilizarán herramientas que automaticen este proceso, no obstante, debido al tamaño reducido del equipo y a la frecuencia de las integraciones (al menos una al día), esta tarea no resultará demasiado compleja.
- **Diseño incremental.** A pesar de definir buena parte de la arquitectura en las primeras iteraciones, el diseño del sistema evolucionará iteración tras iteración, sometiéndose a sucesivas refactorizaciones para mejorar su calidad.
- **Código compartido.** Todos los miembros del equipo podrán acceder a cualquier parte del código. Sin olvidar que el presente desarrollo ágil se encuentra en el contexto de la elaboración de PFCs y que los alcances deben estar acotados para cada uno de los alumnos. Al finalizar cada iteración se destinará tiempo a completar y refinar la documentación obtenida del proceso.
- **Reutilización del código** Uno de los principales objetivos que se persiguen con las metodologías ágiles es entregar proyectos en tiempo y bajo presupuesto, minimizando el *Time To Market*, por lo que la reutilización del código constituye un aspecto muy importante, no se ha de perder tiempo *«reinventando la rueda»* en cada proyecto. Se deben obtener diseños con una alta modularidad y lo más desacoplados posibles, reutilizando como cajas negras, los elementos software que necesitemos.

5.5 Evolución del proyecto

Esta sección describe los cambios ocurridos en el proyecto según su planificación prevista. En cada una de las iteraciones, se exponen los objetos que se pretenden obtener y la toma de decisiones ocurrida en el proceso.

5.5.1 Análisis preliminar de requisitos

El objetivo de este proyecto es construir un sistema de ayuda a la gestión de documental que permita el tratamiento directo sobre documentos físicos impresos mediante el uso de técnicas de visión por computador, síntesis visual y auditiva y técnicas de realidad aumentada.

Los requisitos para el sistema serán proporcionados por Carlos González Morcillo, creador e investigador principal del proyecto; la cátedra Indra-UCLM y la fundación Adecco, como financiadores del proyecto y la Asociación ASPRONA, que con su experiencia en la atención a personas con discapacidad, propondrán escenarios y funcionalidades que sean de utilidad para este colectivo.

Características de los usuarios

Aunque el usuario final del sistema, será cualquier persona que necesite soporte en la gestión documental de documentos impresos, el fin de esté proyecto es construir un sistema que permita la integración laboral a personas con discapacidad.

El tipo de usuarios a los que estará dirigido son, en primer lugar, personas que pueden presentar un amplio espectro de discapacidades. El sistema debe proporcionar soporte a usuarios con discapacidades sensoriales (visuales y auditivas) e intelectuales.

Restricciones

Debido a los objetivos del sistema, se deben tener en cuenta las siguiente restricciones:

- **Funcional en dispositivos móviles.** El prototipo final se construirá sobre un dispositivos con arquitectura ARM con limitaciones de tanto en capacidad de computo como memoria. El sistema deberá estar optimizado para este tipo de dispositivos, obteniendo una respuesta fluida y en tiempo real.
- **Se debe basar en componentes de bajo coste.** Para facilitar la implantación real en el entorno de trabajo, deberá funcionar con componentes de bajo coste, incorporando mecanismos de corrección de distorsión y registro 3D totalmente software.

Interfaces Hardware

- La implementación del sistema se realizará sobre una Raspberry Pi Modelo B de 512MB de RAM y arquitectura ARM.

- La visualización será a través de un pico-proyector mediante conexión HDMI.
- El acceso al sistema y conexión a internet se establece por medio de cable ethernet durante el desarrollo y pruebas, siendo la conexión WiFi el tipo de conectividad final.

Requisitos funcionales

Definidos en la línea base del proyecto

- RF-001: Adquisición de imágenes mediante cámara USB.
- RF-002: Adquisición de imágenes mediante raspiCam.
- RF-003: Sistema de calibrado de cámaras y proyectores.
- RF-004: Configuración del sistema mediante argumentos por terminal.
- RF-005: Implementación de una interfaz natural de usuario.
- RF-006: Control Gestual.
- RF-007: Identificación rápida de documentos.
- RF-008: Sistema de cálculo de homografías.

Definidos por ASPRONA

- RFa-001: Realización de Videoconferencias.
- RFa-002: Mensajes en Lectura Fácil.
- RFa-003: Soporte para completar partes de trabajo.
- RFa-004: Soporte para completar formularios de protocolos de calidad.
- RFa-005: Guiado para clasificación y archivado de facturas.
- RFa-006: Entorno configurable por el usuario.

Requisitos no funcionales

- **Rendimiento:** El sistema debe funcionar de manera fluida y en tiempo real en dispositivos móviles basados en arquitectura ARM. Sólo existirá un único usuario del sistema simultáneamente y los documentos a tratar también se tratarán de uno en uno.
- **Seguridad:** A parte de la información que se proporcione al usuario directamente, se mantendrá un *log* donde se registrará toda la actividad realizada en el sistema. Debido al carácter experimental del proyecto, no se tendrán en cuenta la aplicación, por lo menos en la primera fases de desarrollo, de técnicas criptográficas para ficheros, bases de datos o comunicaciones.
- **Fiabilidad:** Todo tipo de incidente producido en el sistema debe ser controlado y tratado.

- **Disponibilidad:** La disponibilidad del sistema debe ser de al menos un 99 % del tiempo que esté en ejecución. En caso de caída del sistema se deben proporcionar mecanismos automáticos para que la maquina y el sistema se reinicien y sean nuevamente operativos sin la necesidad de intervención directa del usuario,
- **Mantenibilidad:** El desarrollo en un dispositivo tan reciente implica que se liberen con relativa frecuencia bibliotecas o controladores, en los que se corrigen *bugs* y/o mejoran su rendimiento. Seria recomendable hacer una revisión de los módulos actualizados antes de la instalación del sistema para valorar si es relevante el beneficio que aportan y no comprometer el la estabilidad del sistema.
- **Portabilidad:** El desarrollo se realizará siguiendo estándares, tecnologías y bibliotecas libres multiplataforma, con el objetivo de que pueda ser utilizado en el mayor número de plataformas posibles tanto software como hardware.

Otros requisitos

La distribución del proyecto se realizará mediante alguna licencia libre como GPLv3, para ello se utilizarán bibliotecas compatibles con dicha licencia.

5.5.2 Revisión sistemática de la bibliografía

Como punto de partida para la elaboración del estado del arte se realizo una revisión sistemática de las diferentes técnicas para la identificación de documentos y su recuperación de una base de datos. Como resultado de esta revisión se ha podido conocer las técnicas más empleadas, así como sus ventajas e inconvenientes.

El claro ganador es LLAH (Locally Likely Arrangement Hashing) con casi un 40 % de utilización ya sea con su implementación inicial o implementaciones optimizadas creadas para salvar las limitaciones del algoritmo original, lo que lo hace aún más potente y versátil.

Aunque los métodos basados en detección de descriptores de características invariantes como SIFT, SURF, aparecen como muy utilizados, realmente no funcionan correctamente en identificación de documentos ya que no presentan zonas de textura y se producen repeticiones de patrones binarios (el propio texto cumple este patrón). La mayoría son una variación del proceso inspirado en la metodología de Lowe [SIFT], y en la que se obtienen buenos resultados sobre documentos semi-estructurados realizando una selección de puntos extraídos y una adaptación del algoritmo RANSAC para la validación de supuestos aciertos en la comparación.

El algoritmo tiene una tasa elevada de recuperación y precisión, es robusto a las deformaciones que pueda tener la imagen (perspectiva) y no necesita ningún paso previo de segmentación pero, como inconvenientes, no funciona ante grandes secciones de texto y la identificación que realiza es para obtener documentos similares (un ticket, un billete de tren,....) a la imagen utilizada como consulta.

Uno de los puntos del análisis de resultados, indica que al utilizar hardware con distinto rendimiento, es difícil tener mediciones normalizadas para todos los métodos. Como trabajo futuro se puede realizar la implementación de varios de los métodos más utilizados, y ofrecer un estudio comparativo completo al ejecutarse sobre una misma plataforma.

5.5.3 Diseño general

El diseño de GrayAR, se ha abordado con la idea principal de construir un *framework* de realidad aumentada completamente modular y extensible. Para ello, se realizó una clasificación de funcionalidades organizada en módulos y submódulos, siendo cada uno de ellos lo más independiente posible del resto y de las bibliotecas en las que se basaban. Entre las *«herramientas de diseño»* aplicadas para conseguir estos objetivos, se encuentran los patrones de diseño software, que nos proporcionan catálogos de elementos reusables, y la solución estándar y óptima a problemas ya conocidos y solucionados anteriormente.

Esto puede ocasionar un aumento de la complejidad, pero definiendo una arquitectura orientada a componentes, permite la ampliación de la funcionalidad de forma sencilla, la posibilidad de realizar optimizaciones concretas o incluso, la reescritura completa de un módulo, sin que ello afecte en el resto de la plataforma.

5.5.4 Iteraciones

Iteración 0

La iteración 0, se puede considerar la fase anterior al inicio del proyecto. El tiempo empleado en esta iteración comprendió entre el 18 de noviembre de 2013 y el 12 de diciembre de 2013.

Con la idea general del proyecto definida, se adquieren un par de Raspberry Pi y se realiza una búsqueda de pico-proyectores candidatos para utilizar. El proyector debe tener un precio ajustado, ser lo más reducido posible, y además, tener una luminosidad suficiente para que las proyecciones sean visibles en estancias iluminadas.

Una vez recibida la Raspberry Pi, se procede a la configuración del entorno de trabajo. Este proceso incluye la instalación del sistema operativo Raspbian, la biblioteca OpenCV, compiladores, editores y la configuración del servicio de SSH para poder controlar la Raspberry desde el puesto de trabajo.

En esta iteración, se inicia la búsqueda y recopilación de *papers* para la elaboración del estado del arte, el aprendizaje de la biblioteca OpenCV para adquirir una base de conocimientos en el área de la visión por computador, y un estudio de fundamentos matemáticos en álgebra lineal y geometría proyectiva.

Finalmente, el 10 de diciembre de 2013, se mantuvo una reunión con los miembros de la asociación ASPRONA para explicarles los objetivos del proyecto y obtener de ellos, un escenario de casos de uso basados en la experiencia que tienen en la inserción laboral de personas

con discapacidad. Estos casos de uso, se transformó en una especificación de requisitos para nuestro sistema.

Iteración 1

La realización de la construcción y pruebas de esta iteración, se llevó a acabo entre el 12 de diciembre de 2013 y el 8 de enero de 2014.

Las limitadas capacidades de la Raspberry Pi y la experiencia que otros programadores han tenido a la hora de trabajar en sistemas de visión por computador en este dispositivo, nos pone en alerta de que es posible, llegado un momento, la alta carga de computo que tienen los procesos de visión por computador afecten al rendimiento y perdamos la sensación de tiempo real. Las consecuencias de esto, es que sea necesario que los procesos más costosos deban ser ejecutados en otro dispositivo o computador con mejores prestaciones.

Partiendo de las premisas anteriores, no queda otra opción de realizar una arquitectura completamente modular y con todos los subsistemas desacoplados. Una clase *core* será la encargada de inicializar todos los subsistemas, establecer las comunicaciones entre ellos y ejecutar la lógica de usuario soportada dentro de un bucle infinito.

El objetivo de esta primera iteración es obtener una arquitectura básica que iremos completando y refinando en cada una de las sucesivas iteraciones. Se decide que el *sprint* abarque la construcción y pruebas de las siguientes *historias*:

Captura de imágenes. La Raspberry tiene la opción de conectar cámaras USB. Se construye un módulo de captura de vídeo, utilizando las funciones de la clase `VideoCapture` de OpenCV, que proporciona los distintos frames necesarios para dar soporte al resto de módulos del sistema.

Para la medida de rendimiento se implementa el cálculo de los FPS que da el sistema a la salida y que nos va a determinar si se obtienen resultados aceptables para un sistema de tiempo real. Los primeros resultados son mucho peores de lo esperado. A una resolución de 320x240 píxeles y sólo realizando el proceso de captura y visualización de los frames capturados en pantalla, se observa un lag de 2 segundos y una tasa de 2 FPS.

Si estos valores son inadmisibles en un sistema en tiempo real con una arquitectura tan básica, y sin realizar ningún tipo de procesado en la imagen, cuando se deba realizar algún tratamiento a la imagen o computo adicional, el sistema no será usable, con una sensación de bloqueo del sistema para el usuario.

Se vuelven a realizar las pruebas con otra cámara USB más moderna (Logitech S720) y afortunadamente los resultados mejoran sensiblemente, a 6 FPS y hasta 10 FPS para imágenes obtenidas en blanco y negro.

Detección de una hoja de papel. Mediante segmentación de la imagen obtenida, se detecta una hoja de papel y se obtiene la posición de sus 4 esquinas en píxeles de pantalla. Esta funcionalidad se encapsula dentro de la clase `PaperDetector` Durante las primera fase de pruebas nos pone en situación de los numerosos factores que influyen en este tipo de sistemas. El primero de ellos es la iluminación. El tipo de luz (natural, fluorescente, incandescente,...), intensidad de la luz natural (luz por la mañana, al mediodia o por la tarde), dirección de la luz (crea sombras en uno u otro sentido), los reflejos producidos por las superficies se convierten en interferencias en forma de grandes manchas en la imagen. Incluso se han detectado variaciones en la detección al encontrase varias personas entre el sistema y una ventana.

Iteración 2

Los objetivos de esta iteración venian condicionados por los malos resultados obtenidos en el *sprint* anterior. Hay que conseguir mejorar enormente los resultados anteriores o la viabilidad del proyecto se verá seriamente afectada. Se estableció la duración de la iteración entre el 8 de enero y el 3 de febrero.

Optimización de la captura de imágenes. Durante las pruebas de la iteración anterior, con sólo cambiar la cámara USB se obtuvieron grandes mejoras, por lo que la elección de la cámara era un elemento importante. Tras consultar en diversos foros especializados, se decide que la mejor opción es utilizar la ***Raspberry Pi Camera Board***. Esta cámara está especialmente diseñada para utilizarse en la Raspberry, y se conecta directamente al conector CSI de la placa mediante un cable plano flexible de 15 pines. Dispone de un sensor de 5MP de resolución y puede llegar a grabar vídeo a 1080p a 30 fps.

El gran inconveniente de es que la cámara no incluye drivers *video4linux*, por lo que cualquier biblioteca para lectura de cámaras web estándar, OpenCV incluido, no es capaz de obtener los frames producidos. Se investiga la manera que tienen las aplicaciones para acceder a la información de la cámara, y se encuentran unos drivers en desarrollo, basados en la API MMAL, que ha liberado la Universidad de Córdoba.

Tras utilizar en el módulo de captura los nuevos drivers para la *Raspberry Camera Board*, se realizan de nuevo los test de rendimiento, y obtenemos casi 30 PFS, que es un buen resultado para aplicaciones en tiempo real.

Calibrado de la cámara. Se construye un programa externo para el calibrado de la cámara basado en un patrón de tablero de ajedrez. Esta primera versión, devuelve un fichero XML con los parámetros intrínsecos de la cámara y los coeficientes de distorsión obtenidos, pero no tiene en cuenta el error de reproyección.

Sistema básico de cálculo de homografías. En esta primera fase, los cálculos son los necesarios para realizar el registro y visualizarlo en la pantalla del ordenador. Dentro del proyecto, se crea la clase `CameraModel` encargada de leer los ficheros de calibración y almacenar los parámetros propios de la cámara que serán utilizados para el cálculo de los parámetros extrínsecos.

Para el cálculo y almacenamiento de la posición de los folios respecto a la cámara se construye la clase `Paper`.

Las pruebas realizadas al módulo de cálculo de la posición y rotación del papel, son satisfactorias, y se puede comprobar que el dibujado de ejes de coordenadas, o distintos poliedros en el espacio 3D es correcto y perfectamente alineado con el papel.

Dibujado mediante OpenCV. En la iteración anterior se realizaron una serie de funciones auxiliares para dibujado de contornos, puntos y polígonos mediante OpenCV que sirviesen de soporte para el *«modo debug»*. Para la iteración actual, estas funciones se ha aumentado con dibujado de ejes de coordenadas, cubos y ortoedros para validar los cálculos de la posición y rotación de los folios. También permite recibir una ventana con la imagen por medio de SSH. Con todas estas funciones se ha creado la clase estática `DrawCV`.

Iteración 3

El siguiente paso es incorporar el proyector al sistema. Acoplando la Raspberry Pi sobre el proyector y colocando la raspiCam junto a la lente obtenemos un conjunto compacto y reducido. El sistema se monta sobre un trípode y se coloca en dirección a la mesa.

Los objetivos son, por tanto, obtener un sistema de calibrado. Debido a la complejidad de la tarea, se establece como la única tarea a realizar en este *sprint*.

Aunque el *sprint* inicialmente se planteo con una duración aproximada de un mes, la construcción y las numerosas pruebas realizadas para garantizar el correcto funcionamiento finalmente se llevaron a cabo en cerca de dos meses, del 3 de febrero de 2014 al 31 Marzo de 2014.

Calibrado del sistema cámara-proyector Un proyector se calibra usando los mismos algoritmos que una cámara ya que puede considerarse como una *«cámara invertida»*. Sin embargo como el proyector no ve, y el método no es tan directo como en el caso de una cámara. Además es necesario realizar los cálculos para la transformación entre el sistema de referencia de la cámara y el sistema de referencia del proyector.

El estudio de distintas técnicas de calibrado nos llevó a considerar dos procedimientos. El primero, siguiendo el enfoque de Zhang, en el se utilizan patrones planos, y el propuesto por Daniel Moreno y Gabriel Taubin, basado luz estructurada.

Aunque según los *papers* consultados, el método basado en luz estructurada, es más preciso que otras técnicas de calibrado, el procedimiento es más costoso de realizar. El método de Zhang, permite un procedimiento más flexible, no necesita una preparación exhaustiva de la escena y tiene una precisión aceptable para los objetivos de nuestro proyecto.

Con el código para el calibrado de cámaras de la iteración anterior, realizamos una ampliación para añadir la funcionalidad del calibrado del proyector.

Para calibrar el proyector, es necesario obtener un conjunto de coordenadas 3D-2D correspondientes. Las coordenadas se determinan utilizando la cámara situada en una posición con una vista similar a la que tendría el proyector. El método consiste en realizar una proyección de un plano de calibrado y establecer la correspondencia entre lo proyectado y lo que ve la cámara.

Ahora dispondremos de los puntos 3D en el sistema de referencia global (el patrón), y también su proyección (puntos 2D en la imagen) en el sistema de referencia de la cámara y en el sistema de referencia del proyector.

Finalmente, aplicando un procedimiento de calibrado estéreo como el que proporciona OpenCV obtenemos la transformación entre la cámara y el proyector.

El resultado, con los parámetros intrínsecos y extrínsecos, se escribe en ficheros YAML para cargarlos en el sistema. Mientras que la cámara y el proyector mantengan su posición y rotación entre ellos, no es necesario realizar una nueva calibración y es posible mover todo el sistema.

Entre las decisiones tomadas en este *sprint*, fue realizar el calibrado como una aplicación externa al sistema. Esto nos permite la reutilización del módulo para otros sistemas estéreo.

Otras decisión fue la resolución de imagen con la que trabajaría el sistema. El proyector tiene una resolución nativa de 854x480 en formato 16:9, pero la Raspberry Pi no la soporta. Se eligió una resolución de 1280x720, pero con este tamaño de imagen, el rendimiento disminuía significativamente. También se utilizó 800x600 en formato 16:9, pero la distorsión que realizaba el proyector para mostrar la imagen en formato panorámico no podíamos medirla para realizar una corrección, y los puntos calculados no correspondían con los objetos situados sobre la mesa.

La resolución que se estableció finalmente fue de 800x600 en formato 4:3. Se perdía algo de superficie de proyección, pero con aumentar un poco la distancia entre el proyector y la mesa se compensaba esta área, y además, el rendimiento del sistema con esta resolución era adecuado y entraba dentro de los parámetros para un sistema de tiempo real.

Iteración 4

El periodo de la iteración transcurrió entre el 31 Marzo de 2014 y el 28 de Abril de 2014.

Desde la iteración 2, se arrastraba un pequeño *bug* en el cálculo de los parámetros extrínsecos del papel. Entre los objetivos planificados en esta iteración se decidió que también se corrigiese este bug, quedando finalmente el *sprint backlog* con los siguientes elementos:

Corrección de la orientación del papel En el cálculo de los parámetros extrínsecos del papel, dependiendo de la posición del papel, el sistema confundía la orientación y consideraba que el papel se encontraba apaisado cuando realmente no era así.

La corrección del *bug* fue relativamente sencilla. Tras una serie de pruebas rotando la hoja de papel para comprobar en que casos se producía el error, se observó que el problema era debido a como OpenCV enumera las esquinas del papel.

Detector de documentos mediante descriptores de imágenes Se eligió utilizar SURF como detector y descriptor de características debido que es más rápido y robusto que otros algoritmos. Aun así, la implementación de esté modulo se realizó utilizando los *wrappers* que proporciona OpenCV para la construcción de detectores y descriptores, y que tiene como ventaja, cambiar el algoritmo aplicado sin necesidad de modificar la implementación, únicamente modificando en el constructor un *string* con el método que queramos utilizar (BRISK, ORB, SURF,...).

Optical Flow Para la estimación y descripción del movimiento, se implementó *Optical Flow* (Lucas-Kanade) que proporciona herramientas para detección, segmentación y seguimiento de la hoja de papel en la escena a partir de un conjunto de imágenes. En etapas posteriores, debido a la oclusión que se realiza sobre el papel, se decidió que no cumplía las expectativas como metodo de tracking y se rechazó.

Iteración 5

A estas alturas del desarrollo del proyecto, la carga de trabajo que debe soportar la Raspberry Pi es muy significativa. Se aprecia que el rendimiento se ve afectado al incluir la detección de documentos, ya que es una tarea costosa para el sistema, y aun queda por implementar el módulo de interacción natural, que se trata también de una tarea con gran carga de trabajo.

Durante la reunión de planificación, se acuerda que se realicen las siguientes tareas en el periodo del 28 de Abril de 2014 al 12 de Mayo de 2014:

Migración del sistema al servidor Como ya se venía observando, en cada iteración el rendimiento general del sistema en la Raspberry era cada vez menor. No sólo tenia que soportar los módulos de GrayAR, sino que también debía soportar la ejecución de *BelfegAR*. El objetivo de la iteración anterior de *Belfegar*, fue crear una arquitectura cliente-servidor que permitiese la delegación de tareas entre la Raspberry Pi y un servidor. Este servidor, seria el encargado de realizar la ejecución de los módulos más costosos del sistema y descargar a la Raspberry Pi, para realizar las tareas de captura y representación más holgadamente.

Gracias a que se tuvo en cuenta que este caso pudiese ocurrir, y la decisión temprana construir un sistema modular y desacoplado, la migración al servidor y la conexión mediante la unidad de delegación de tareas de *Belfegar* fue prácticamente transparente y sin realizar modificaciones en los módulos creados.

Aprovechando esta arquitectura, también se separó el programa de calibración, lo que supuso una reducción considerable del tiempo empleado para realizar el calibrado del sistema cámara-proyector.

Implementación de un histórico de percepciones Al igual que en ARToolKit, se desarrolla una función de tratamiento del histórico de percepciones para estabilizar el *tracking*. Este histórico se implementa almacenando las últimas 4 percepciones similares y realizando una media ponderada, en la que las percepciones recientes tienen más peso que las antiguas. Para determinar si son percepciones próximas se establece un umbral. Mediante el uso de esta técnica eliminamos gran parte del efecto tembloroso en la proyección.

Optimización del proceso de detección de rectángulos La función para detectar hojas de papel implementada en la iteración 1 es muy básica. No permite que exista ninguna oclusión y es muy sensible a la iluminación. Tras un estudio de técnicas alternativas, se opta por implementar el cálculo de la envoltura convexa del contorno. Además de ser un algoritmo más reducido, la mayor ventaja que aporta este método es que tolera ciertos solapamientos en los bordes del papel.

Para optimizar más la función, se sustituye el método de binarización de la imagen. Se cambia la umbralización adaptativa por el método de Canny. Con este cambio, se obtiene un algoritmo más robusto, más tolerante a la iluminación y que devuelve bordes más finos, con lo que también aumenta la precisión del sistema.

Iteración 6

Esta iteración supone la construcción del sistema de interacción natural de usuario. Al finalizar esta iteración, se podrá realizar acciones sobre los botones que está desarrollando *BelfegAR* y que permitirán ofrecer una experiencia de usuario muy enriquecida al mostrar

información opcional a petición del usuario. Para este desarrollo se estimó una duración entre el 12 de Mayo de 2014 y el 23 de Junio de 2014.

Segmentación de manos Para extraer el contorno de la mano de la imagen se decide utilizar un filtrado por el color de la piel. Existen numerosos artículos sobre este tema, y todos concluyen que los mejores resultados se obtienen realizando la búsqueda en el espacio de color HSV *(Hue, Saturation, Value)*. Para convertir las imágenes a este espacio de color, es necesario que la captura se realice en color. Hasta esta iteración, la adquisición de imágenes se estaba efectuando en escala de grises, ya que consumía menos recursos, y el color no era una propiedad que necesitásemos.

Teniendo en cuenta que los procesos de cálculo están en el servidor, obtener las imágenes a color no supuso una pérdida de rendimiento en el sistema. Además, con sólo realizar la conversión a escala de grises de la imagen, los métodos ya implementados y que no necesitan la imagen a color no tienen que ser modificados.

La segmentación implementada, se basa en la selección del los píxeles de la imagen que pertenezcan al rango habitual del color de la piel. Para aumentar la tolerancia del algoritmo, se le aplica a la imagen una serie de filtros, y finalmente se obtiene una máscara con la región de la mano en la imagen.

Cálculo de la posición del dedo Una vez obtenida el contorno de la mano, se debe buscar su posición. En un principio se pensó en implementar el reconocimiento de todos los dedos de la mano, pero se optó por sólo reconocer el índice. Esta decisión, vino determinada por la funcionalidad de interacción que se iba a implementar, que era la pulsación de un botón, por lo que examinar la posición del resto de los dedos es, en principio, innecesaria.

Iteración 7

Con el sistema de interacción de usuario funcionando el sistema, se advierte que el módulo de detección de papeles funciona correctamente cuando no hay solapamientos o si la oclusión se encuentra muy próxima a los bordes. En la reunión del *sprint* se considera que se debe implementar una nueva mejora de la detección de papeles, y establece esta iteración entre el 23 de Junio de 2014 y el 14 de Julio de 2014.

Detección de papeles con solapamiento Analizando los solapamientos que se realizan sobre el papel en función de las posiciones de la mano, se observa que normalmente quedan visibles partes de los lados del papel. Se decide realizar la búsqueda de segmentos rectos mediante la transformada de Hough y generar las posibles combinaciones de 4 segmentos del conjunto de rectas detectadas. Aquellos que cumplan las restricciones de formar una hoja de papel, serán los posibles candidatos.

Este segundo método de detección es más costoso que el anterior, por lo que se determinó que de forma predeterminada, el sistema utilizase la detección basada en la envoltura convexa cuando no existiesen solapamientos, y en caso de fallo, se aplicase esta nueva función.

Los resultados de las pruebas fueron muy satisfactorias. Esta nuevo algoritmo detectaba el papel aún existiendo grandes solapamientos, tanto en bordes como en esquinas del documento.

Iteración 8

Durante es periodo del 14 de Julio de 2014 al 25 de Agosto de 2014, prácticamente no se efectuaron modificaciones al sistema. Se realizaron algunas correcciones de *bugs* y se ajustaron algunos parámetros en la configuración de los algoritmos de visión por computador. La mayor parte de esta iteración fue destinada a la realización, tanto de la presente memoria, cómo del informe técnico a presentar tras la finalización del *Proyecto ARgos*.

Iteración 9

El 15 de Septiembre, recibimos de la Asociación ASPRONA una serie de formularios y partes de trabajo para que se implementase un caso real mediante los documentos que ellos utilizaban habitualmente. Tras una reunión, se establece el nuevo *sprint* entre el 15 de Septiembre de 2014 y el 28 de Noviembre de 2014. Las tareas que se realizan son la preparación y entrenamiento del sistema para adecuarlo a la documentación recibida.

Para facilitar el cambio de configuración y no tener que recompilar constantemente, se implementa un gestor de configuración. Este gestor nos permite la modificación de los distintos parámetros del sistema sin la necesidad de recompilar nuevamente el proyecto cada vez que se realiza un cambio. También, de cara a la versión final, se ajustan el sistema para que no sea necesario utilizar una base de fondo negro, que se estuvo utilizando para mejorar el contraste entre papel y mesa.

Con el sistema preparado, definimos el flujo uso de cara a la «demo final» y se graba un vídeo de demostración.

El 15 de Octubre el proyecto es presentado en el **5º Congreso Nacional CENTAC de Tecnologías de la Accesibilidad** con gran aceptación por parte del público asistente. Se concertó una reunión el 28 de Octubre con la Asociación ASPRONA, donde tras la demostración, expresaron interés por el sistema y el potencial que tiene para la adaptación y ayuda a personas con discapacidad en el puesto de trabajo.

Finalmente, el 25 de Noviembre de 2014 en otra reunión mantenida, fue expuesto a responsables de Indra y la fundación Adecco.

5.6 Tecnologías y herramientas utilizadas

En esta sección se listan y detallan los recursos software y hardware empleados en la construcción de la plataforma. Además de una breve explicación del recurso, se enuncia la versión utilizada y sobre qué plataformas opera.

5.6.1 Hardware

- **Raspberry Pi.** Se trata de la unidad de detección y despliegue del sistema. En el Anexo A pueden verse las características oficiales del modelo empleado.

- **Raspberry Pi Camera Board** Se trata de una minicámara de 5MP con un sensor Omnivision 5647, que se conecta directamente a la Raspbery Pi a través de un conector CSI.

- **Proyector portátil.** Se trata del medio principal que permite el despliegue de gráficos del sistema. El proyector empleado es un Optoma PK320 que soporta una resolución panorámica máxima de 854x480.

- **Amplificador de Audio** Se trata de un circuito integrado LM386 montado en una placa de test con un altavoz de 1W para proporcionar audio al sistema.

- **Tarjeta de memoria:** Se emplea una MicroSDHC Trascend de 32GB clase 10, como unidad de almacenamiento de la Raspberry Pi. Proporciona una velocidad de lectura y escritura de 20 MB/s y 17 MB/s respectivamente.

- **Equipo informático** Para el desarrollo del proyecto ha sido necesario hacer uso de un computador conectado por red a la plataforma Raspberry Pi, para poder acceder remotamente a ella y de esta forma ejecutar las pruebas. El equipo es un Intel Core i7-2600K 3.4 GHz 4 núcleos y dos hilos por núcleo 16 GB de RAM y Nvidia GeForce GTX 560 Ti.

5.6.2 Software

A continuación se enumeran las diversas herramientas software empleadas y diferenciadas por categorías:

Lenguajes de programación

- **C++** - El lenguaje empleado para el desarrollo del proyecto ha sido C++ [Str13], debido a la eficiencia y velocidad de ejecución que proporciona a la hora trabajar en aplicaciones y sistemas en tiempo real. También por ser el estándar referente en bibliotecas gráficas y de visión artificial.

Sistemas Operativos

- **Debian** - Es una distribución de GNU/Linux desarrollada y mantenida por una comunidad de voluntarios. Es una de las famosas y un gran número de distribuciones están basadas en ella. Para el desarrollo del proyecto se ha utilizado la versión *unstable*.
- **Raspbian** - Es una distribución de GNU/Linux basada en Debian Wheeze especialmente diseñada y optimizada para la ejecución en la placa Raspberry Pi con CPU ARMv6

Aplicaciones de desarrollo GNU

- **GNU Emacs** - Editor y entorno de desarrollo. Se ha utilizado la generación del código fuente y la escritura de la documentación. Se han empleado, además, algunos *plugins* como ECB [1], que añade al editor funcionalidades propias de un entorno de desarrollo completo. Versión 24.3.1
- **GNU Make** - Herramienta para la compilación incremental, con soporte multiproceso.
- **GNU GCC** - La colección de compiladores GNU. En concreto se ha utilizado el compilador de C++ (g++) en su versión 4.5.2.
- **GNU GDB** - Se trata del depurador por excelencia de los sistemas GNU/Linux. Se ha utilizado la versión 7.2.
- **GNU GPROF** - Es una herramienta para hacer profiling (ver Sección 6.2) para compiladores de la familia gcc. Se ha utilizado la versión 2.21.
- **GNU CMAKE** - Se trata de una herramienta análoga a make, aunque de más alto nivel, para la automatización de generación de código. Se ha utilizado principalmente para la compilación de la biblioteca RaspiCam. La versión de CMake es la 2.8.3.

Documentación y gráficos

- **Doxygen** - Sistema de documentación de código fuente. Compatible con C++. Versión 1.8.9.1.
- **InkScape** - Programa de edición de imágenes vectoriales. Versión 0.48.5-3
- **Draw.io** - Servicio web para dibujado de diagramas y gráficos vectoriales.
- **GIMP** - Herramienta de manipulación de gráficos. Versión 2.8
- **LibreOffice Draw** - Potente herramienta de dibujado vectorial perteneciente a la suite ofimática LibreOffice. Utilizada para la generación de diagramas para la documentación. Versión 3.5.4.

[1] `http://ecb.sourceforge.net/`

- **LaTeX** - Es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. Elegido para la generación de la documentación mediante la distribución Texlive. Versión 2014.20141024-1
- **AUCTeX** - Para la elaboración de este documento se ha empleado este paquete [2] que permite integrar las funciones de LaTeXen los menús del editor Emacs. Versión 11.86-10.2.
- **esi-tfg** Clase LaTeX del grupo ARCO desarrollada por David Villa, y que proporciona una plantilla con la que se ha maquetado este documento.

Bibliotecas

- **OpenCV** - Biblioteca libre que proporciona funciones dirigidas principalmente para el desarrollo de aplicaciones de visión por computador en tiempo real. La versión utilizada es la 2.4.9
- **RapidXML** - Se trata de un intérprete o *parser* para archivos XML. Versión 1.13
- **RaspiCam** - Es una biblioteca para la utilización de la cámara Raspberry Pi Board desarrollada en C++ por el grupo de investigación «Aplicaciones de la Visión Artificial» de la Universidad de Córdoba. La versión utilizada es la 0.1.1

Control de Versiones

- **Git** - Sistema de control de versiones distribuido. Como repositorio central se ha utilizado la plataforma Bitbucket.

[2] `http://www.gnu.org/software/auctex/`

Capítulo 6

Arquitectura

En este capítulo se mostrarán los resultados obtenidos al aplicar la metodología descrita en la sección 5 a través de un enfoque top-down, comenzando con una descripción general, continuando con las decisiones de diseño y terminando con los detalles de implementación.

GrayAR se ha diseñado como una arquitectura con 6 subsistemas (Figura 6.1) y una herramienta externa para calibrar cámaras y proyectores (Figura 6.2). A continuación se indica el cometido de cada uno de los módulos:

- *Sistema de Captura:* Es el encargado de la capturar y proveer de imágenes al sistema por medio de la cámara de la Raspberry Pi ó a través de cámaras USB.
- *Sistema de Tracking y Registro:* Su misión es detectar y calcular la *pose* de los documentos mostrados al sistema.
- *Sistema de Identificación de Documentos:* Este módulo asume la tarea de identificar los documentos en base a su contenido.
- *Sistema de Interacción Natural de Usuario:* Proporciona los mecanismos para implementar el paradigma de «pantalla táctil» como interfaz del sistema.
- *Sistema de Modelos Matemáticos:* Consiste en la implementación de los modelos matemáticos, de los que se sirven el resto de módulos para realizar sus cálculos.
- *Sistema de Soporte y Utilidades:* Es un conjunto de utilidades internas de GrayAR. Incorpora el *log* del sistema, funciones para dibujar y un gestor de configuración.
- *Sistema de Calibrado (calibrationToolbox):* Implementado como una aplicación externa, su función es obtener los parámetros intrínsecos y extrínsecos de la cámara y el proyector.

El diseño de los componentes que forman GrayAR mantiene, una independencia minimizando el acoplamiento en su definición. El encargado de establecer la relación entre ellos y controlar el bucle de ejecución es la clase `Core`.

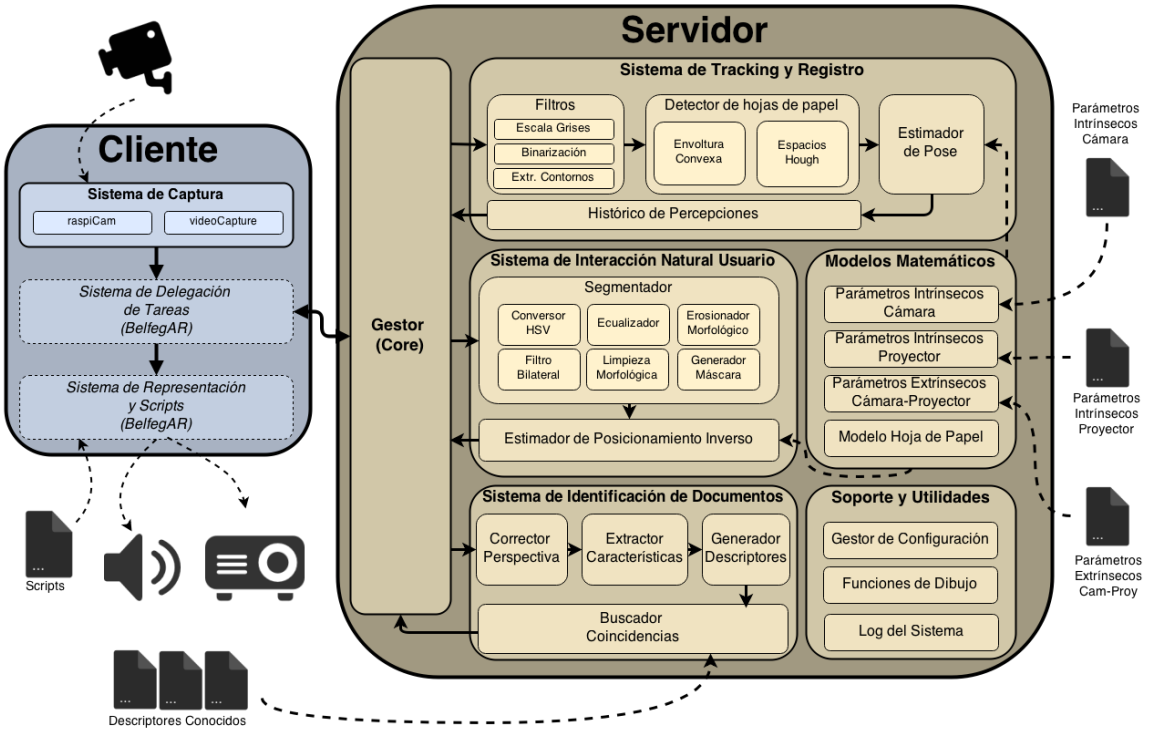


Figura 6.1: Diagrama de la estructura de GrayAR en el contexto de *ARgos*

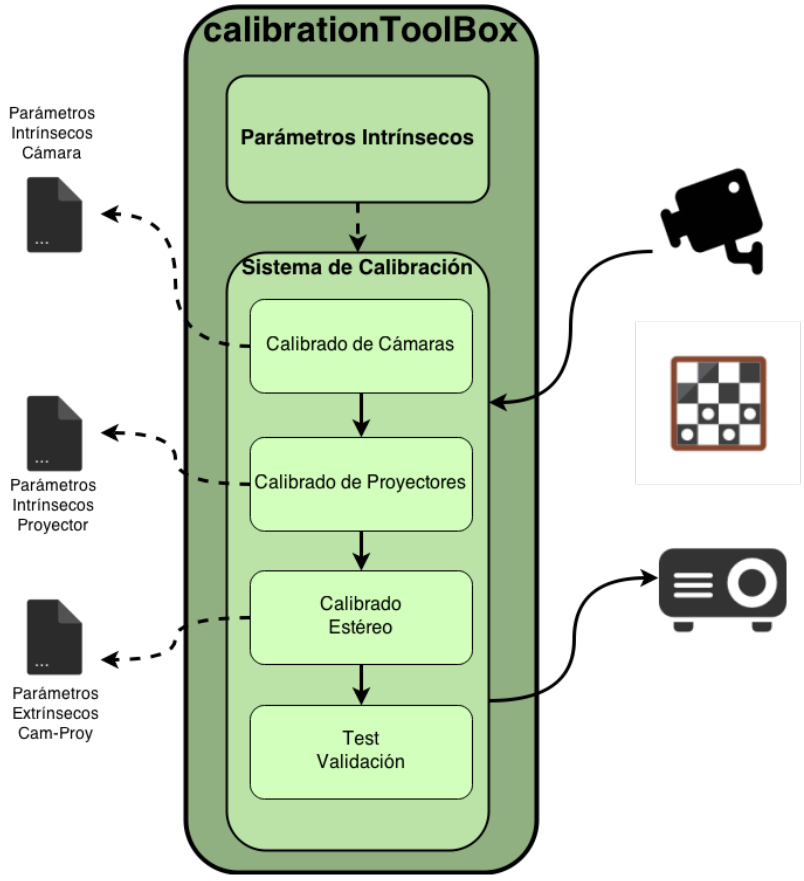


Figura 6.2: Diagrama de la estructura de calibrationToolbox

En este capítulo se procederá a explicar en detalle los sistemas de GrayAR sin tener en cuenta la distinción entre la parte del cliente y la del servidor, ya que es totalmente transparente al sistema y no corresponde al alcance del proyecto.

6.1 Módulo externo de calibración (calibrationToolbox)

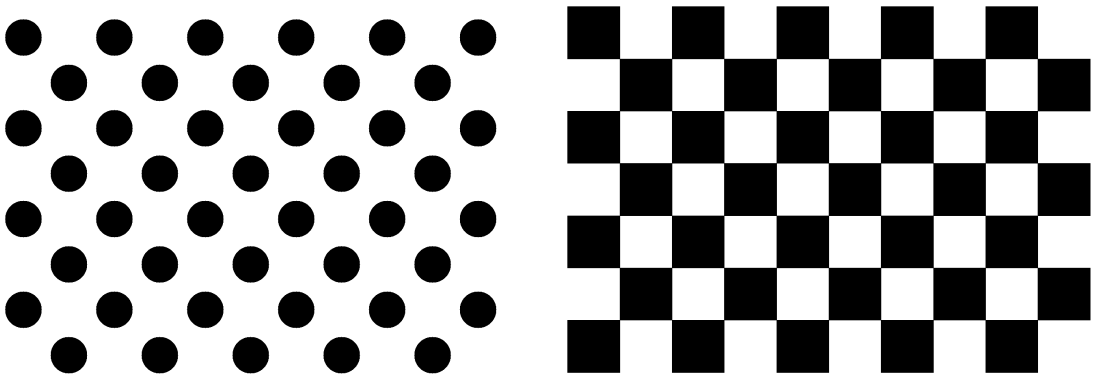
Como ya se explicó en el capitulo 4, los objetivos de realizar el proceso de calibración son la estimación de los parámetros intrínsecos y extrínsecos de la cámara. Los parámetros intrínsecos se refieren a las características internas de la cámara, como por ejemplo, su distancia focal, distorsión, y el centro de la imagen. Los parámetros extrínsecos describen su posición y orientación dentro de un espacio de referencia. Conocer los parámetros intrínsecos es un primer paso esencial, ya que permite calcular la estructura de la escena en el espacio euclídeo y elimina la distorsión de lentes, la cual afecta a la precisión.

Para ubicar objetos en el mundo real, establecemos un sistema de referencia, denominado sistema de referencia global. Un objeto en una imagen es medido en términos de coordenadas de píxeles, los cuales están en el sistema de referencia de la imagen. El sólo conocer la distancia en píxeles entre puntos en una imagen, no nos permite determinar la distancia correspondiente a los mismos puntos en el mundo real. Por lo tanto, es necesario establecer las ecuaciones que unan el sistema de referencia global con el sistema de referencia de la imagen, de manera de establecer la relación entre los puntos en coordenadas en el espacio 3D y los puntos en coordenadas de imagen 2D.

Desafortunadamente, no se puede establecer esta relación directamente, haciéndose necesario establecer un sistema de referencia intermedio, llamado sistema de referencia de la cámara. Por lo tanto, se deben encontrar las ecuaciones que unan el sistema de referencia de la cámara con el sistema de referencia de la imagen, y las ecuaciones que unan el sistema de referencia del global con el sistema de referencia de la cámara. Al resolver el sistema de ecuaciones generado se obtiene la relación buscada.

Básicamente, el proceso consiste en obtener una serie de imágenes en los que se encuentre visible un patrón plano (de dimensiones conocidas), con distintas orientaciones y distancias de la cámara. De cada patrón encontrado en las imágenes obtenemos una ecuación de homografía que establece la relación entre los puntos en coordenadas en el espacio 3D y los puntos en coordenadas de imagen 2D. Aunque en teoría con dos imágenes sería suficiente para resolverlo mediante un sistema lineal de ecuaciones, el objetivo es obtener el mayor número de ellas, ya que en la práctica existe gran cantidad de ruido en las imágenes adquiridas. Se recomienda por tanto, para obtener buenos resultados, al menos 10 imágenes correctas del patrón en diferentes posiciones.

Como soporte al cálculo de los parámetros, OpenCV proporciona tres tipos de patrones para la calibración:



(a) Patrón de Círculos Asimétricos (b) Patrón Tipo Tablero de Ajedrez

Figura 6.3: Tipos de patrones de calibración

- Tablero de ajedrez
- Distribución de círculos simétrica
- Distribución de círculos asimétrica

En principio, cualquier objeto caracterizado apropiadamente podría ser utilizado como patrón para la calibración. Existen otros métodos que basan sus referencias en objetos tridimensionales o que requieren de patrones de calibración consistentes, en al menos dos planos ortogonales.

La principal ventaja de la utilización de patrones planos frente a otras técnicas es su flexibilidad. No necesita de una preparación exhaustiva de la escena, ni es necesario conocer las posiciones de los mismos. También resulta mucho más complicada la construcción y distribución de objetos 3D precisos para realizar una calibración.

6.1.1 Descripción general del calibrado de GrayAR

El proceso de calibrado de la cámara esta basado esencialmente por el enfoque de Zhang [ZFN02]. Se utiliza un patrón tipo tablero de ajedrez, en la que se alternan cuadrados blancos y negros, de dimensiones conocidas. El patrón se imprime y se pega sobre una superficie plana rígida. A continuación se obtiene una serie de imágenes en los que se encuentre visible el patrón desde varias posiciones.

Se realiza el cálculo de las homografías entre el patrón y sus imágenes. Estas transformaciones proyectivas 2D producen un sistema de ecuaciones lineales que al resolverse obtiene los parámetros de la cámara. Esta fase generalmente es seguida por una etapa de refinamiento no lineal, basado en la minimización del error total de reproyección.

Se ha diseñado y construido como una utilidad a parte del proceso principal, ya que una vez calibrado el sistema, genera unos ficheros YAML con los parámetros intrínsecos y extrínsecos que se cargan en el proyecto. Mientras que la cámara y el proyector mantengan su

posición y rotación entre ellos, no es necesario realizar una nueva calibración y es posible mover todo el sistema.

El módulo implementado está basado en un *plugin* para *openFrameworks* que han realizado Álvaro Cassinelli, Niklas Bergström y Cyril Diagne a partir de un complemento desarrollado por Kyle McDonald. Es capaz de calibrar cámaras y proyectores, consiguiendo los parámetros intrínsecos de ambos, además de los extrínsecos en cuestión de varios minutos.

El proceso de calibrado está divido en 4 fases:

- **Calibrado de la cámara.** Aunque la cámara y el proyector podrían ser calibrados de forma simultánea, es mejor comenzar primero por calibrar la cámara. Siguiendo en método de Zhang, los parámetros intrínsecos de la cámara se calculan encontrando las coordenadas, en el plano de imagen, de las esquinas de los cuadrados de un patrón de calibración para cada una de las orientaciones capturadas.
- **Calibración del proyector.** El sistema proyecta un patrón de círculos asimétrico en una posición fija. La cámara se utiliza para calcular la posición 3D de los círculos proyectados, primero según el sistema de coordenadas de la cámara, y después según el sistema de coordenadas del patrón proyectado. Con esto parámetros se calculan los parámetros intrínsecos del proyector porque tiene puntos 3D (los círculos proyectados) en coordenadas reales, y sus respectivas proyecciones en el plano de imagen del proyector. El procedimiento de cálculo de homografías es el mismo que para las cámaras, ya que el modelo matemático del proyector, es el de una cámara invertida.
- **Calibración del sistema estéreo (cámara-proyector).** Una vez que tengamos un error de reproyección suficiente (para el proyector), se puede comenzar a mover el patrón de los puntos proyectados a lo largo de toda la superficie de proyección con el fin de explorar mejor el espacio (y obtener una calibración más precisa). En esta fase, podemos ejecutar la calibración estéreo de OpenCV para obtener los parámetros extrínsecos cámara-proyector . Después de varias capturas, el proceso converge, y los datos se almacenan en un fichero YAML.
- **Verificación del calibrado.** Una vez se han obtenido todos los parámetros de calibrado del sistema, se inicia una fase de test para comprobar la fiabilidad del calibrado realizado. Utilizando el patrón de tablero de ajedrez, el sistema proyectará, de forma dinámica, 4 círculos situados en las respectivas esquinas del patrón. Si el proceso se ha realizado correctamente y con un error de reproyección contenido, las proyecciones deben coincidir con las esquinas y realizar un desplazamiento acorde, según movamos el patrón dentro de la zona de proyección

Gracias a la decisión de separar el proceso, permite poder iniciarlo en cualquiera de las fases descritas. Podremos por tanto, calibrar sólo la cámara, o partiendo de una calibración

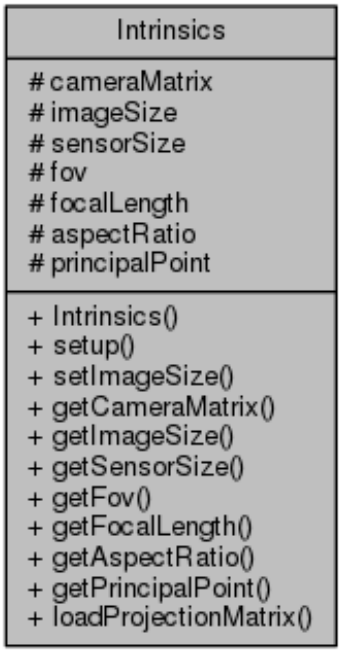


Figura 6.4: Clase Intrinsics

previa de la cámara calcular los parámetros del proyector o simplemente validar una calibración anterior, iniciando el proceso en la fase de verificación.

6.1.2 Definición del modelo de proyección y calibrado

La idea principal tras los procesos de calibrado de cámaras es describir el modelo de proyección que relaciona los sistemas de coordenadas que permiten obtener los parámetros de la cámara.

La clase `Intrinsics` almacena la geometría y las características internas de la cámara. La matriz intrínseca o matriz de la cámara se representa como un objeto matriz `cv::Mat` de dimensiones 3x3 formada por los siguientes parámetros:

$$cameraMatrix = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

Los parámetros f_x y f_y representan la distancia focal en términos de distancia. γ representa el coeficiente de asimetría entre los ejes X e Y, pero por simplificar el modelo, tomaremos que tiene valor 0. Finalmente, c_x y c_y representan las coordenadas en píxeles del punto principal, que sería idealmente en el centro de la imagen.

A partir de la matriz de la cámara y el tamaño de la imagen capturada, la clase `Intrisics` calcula el resto de parámetros propios como son el campo visual `(fov)`, la distancia focal

(focalLength), el centro óptico (principalPoint) y la relación de aspecto (aspectRatio) mediante la función cv::calibrationMatrixValues.

```
cv::Mat cameraMatrix;        // (fx 0 cx, 0 fy cy, 0 0 1)
cv::Size imageSize;          // Size of the image
cv::Size sensorSize;         // Size of the image
cv::Point2d fov;             // Field of view
double focalLength;          // Focal length
double aspectRatio;          // Aspect Ratio
cv::Point2d principalPoint; // Principal point (center)
```

Listado 6.1: Atributos de la clase Intrinsics

OpenGL asume que no existe distorsión en la cámara por lo que a la hora de calcular la matriz de proyección debemos tener en cuenta que los parámetros de la matriz de la cámara deben estar corregidos. De esta forma, se utilizaran dos instancias de la clase Instrinsics: distortedIntrinsics y undistortedIntrinsics. La primera instancia almacena los parámetros «reales» mientras que en undistortedIntrinsics se encuentran los parámetros corregidos mediante las funciones cv::getOptimalNewCameraMatrix y cv::initUndistortRectifyMap.

Aunque el proceso de calibrado para la cámara y el proyector sigue el mismo enfoque, hay ciertas particularidades propias de cada dispositivo en el proceso de calibrado. La clase Calibration implementa toda la funcionalidad común, que es la mayoría, mientras que se han definido las clases CameraCalibration y ProjectorCalibration que heredan de Calibration y terminan de definir los procesos específicos.

Finalmente la clase CameraProjectorCalibration encapsula un objeto de CameraCalibration y otro de ProjectorCalibration con los parámetros intrínsecos de ambos dispositivos e incluye los vectores de rotación (rotCamToProj) y traslación (transCamToProj) que definen las transformaciones necesarias entre el sistema de referencia de la cámara y el sistema de referencia del proyector.

CalibrationCore es la encargada de inicializar la calibración mediante la configuración establecida y define el proceso a realizar en cada una de las fases que lo compone.

6.1.3 Calibración de la cámara

En esta primera fase el objetivo es obtener los parámetros intrínsecos de la cámara. De forma continua, el sistema realiza capturas de imagen. Durante este proceso, se debe mostrar a la cámara el patrón impreso en distintas posiciones y orientaciones.

Para cada imagen capturada, se realiza el siguiente proceso:

```
//Intrinsics
Intrinsics distortedIntrinsics;
Intrinsics undistortedIntrinsics;
cv::Mat distCoeffs;

// Calibration parameters
vector<vector<cv::Point2f> > imagePoints;
vector<vector<cv::Point3f> > objectPoints;
float reprojectionError;
vector<float> perViewErrors;
vector<cv::Mat> boardRotations;
vector<cv::Mat> boardTranslations;

// Pattern Configuration
CalibrationPattern patternType;
cv::Size patternSize;
cv::Size addedImageSize;
cv::Size subpixelSize;
float squareSize;

// Auxiliar calibration variables
cv::Mat grayMat;
bool fillFrame;
cv::Mat undistortBuffer;
cv::Mat undistortMapX, undistortMapY;
bool ready;
```

Listado 6.2: Atributos de la clase Calibration

- Se realiza una **búsqueda de las esquinas de los cuadrados del patrón de calibración** en la imagen. Para detectar las esquinas se utiliza la función `cv::findChessboardCorners`. A esta función se le debe proporcionar la imagen actual donde buscar el patrón, las dimensiones del patrón y un vector de `cv::Point2f` donde se almacenaran los puntos de la imagen de las esquinas detectadas. Adicionalmente, para mejorar esta detección se emplea `cv::cornerSubPix`, que se encargará de ubicar estas esquinas en medidas de subpíxeles. Si se ha encontrado el tablero de ajedrez en la imagen, el vector de puntos devuelto se guardan en el vector `imagePoints` junto con las extracciones de los *frames* anteriores.
- **Se generan los puntos de las esquinas del patrón en el sistema de coordenadas globales** (`objectPoints`) correspondientes a la imagen actual, en función del tipo, dimensiones y el valor de la medida real del lado del cuadrado del patrón que se ha impreso a través de la función `updateObjectPoints`.
- Para **realizar la asociación entre los puntos de la imagen y los del objeto** se utiliza la función `cv::calibrateCamera` que estima los parámetros intrínsecos y extrínsecos de la cámara para cada una de las vistas. Recibe los `objectPoints` y los `ima-`

```
    // Threshold parameters
    circleDetectionThreshold = 160;

    // Application
    diffMinBetweenFrames = 4.0;
    timeMinBetweenCaptures = 2.0;

    //Boards parameters
    numBoardsFinalCamera = 20;
    numBoardsFinalProjector = 20;
    numBoardsBeforeCleaning = 3;
    numBoardsBeforeDynamicProjection = 5;
    maxReprojErrorCamera = 0.20;
    maxReprojErrorProjectorStatic = 0.25;
    maxReprojErrorProjectorDynamic = 0.43;

    // Image Size
    projectorFrame.create(cv::Size(800,600), CV_8UC1);

    // --- INITIAL MODE ---
    setState(CAMERA); //CAMERA, PROJECTOR_STATIC, DEMO_AR
```

Listado 6.3: Configuración de la clase CalibrationCore

gePoints de todas las capturas que llevamos realizadas y el tamaño de la imagen capturada (addedImageSize), y devuelve la matriz de la cámara (cameraMatrix), los coeficientes de distorsión (distCoeffs) y un vector con los vectores de rotación boardRotations y traslación boardTranslations que se ha estimado para cada vista del patrón. Es decir, cada rotación del vector k-ésimo junto con el vector correspondiente k-ésimo de la traslación, convierte los puntos del patrón de calibración desde el sistema de coordenadas globales (en el que se especifican los puntos del objeto) al sistema de referencia de la cámara en la vista k-ésima. La función devuelve también el error de reproyección cometido para cada una de las vistas.

```
    cv::Mat cameraMatrix = cv::Mat::eye(3, 3, CV_64F);
    distCoeffs = cv::Mat::zeros(8, 1, CV_64F);
    int calibFlags = 0;
    float rms = cv::calibrateCamera(objectPoints, imagePoints,
       addedImageSize, cameraMatrix, distCoeffs, boardRotations,
       boardTranslations, calibFlags);
    cout << "RMS reprojection error: " << rms << endl;
```

Listado 6.4: Calibrado de la cámara mediante cv::calibrateCamera

- La función `cv::checkRange` **comprueba la integridad de los elementos de la matriz de la cámara y de los coeficientes de distorsión**, asegurando que son valores numéricos válidos.
- A partir de los parámetros intrínsecos obtenidos, se **actualiza el error de reproyección general** de la calibración (`updateReprojectionError()`), y se **calcula la matriz de proyección de la cámara corregida** (`undistortedIntrinsics`) aplicando los valores de distorsión.
- Cada cierto número de iteraciones **se eliminan las calibraciones que tienen un error de reproyección mayor que el umbral definido** en la variable `maxReprojErrorCamera`

Alcanzado el número de calibraciones correctas por debajo del error definido, se procede al **almacenamiento de los últimos parámetros calculados** en un fichero de tipo YAML. Para ello se usa un objeto de tipo `cv::FileStorage` para escribir el fichero `calibrationCamera.yml` con la estructura y formato definido.

Una vez finalizada la calibración de la cámara, el sistema continua para obtener los parámetros intrínsecos del proyector.

6.1.4 Calibración del proyector

El cálculo de los parámetros intrínsecos de proyector se realiza mediante el mismo enfoque que en la cámara. Considerándolo como una *«cámara inversa»* se puede aplicar el mismo modelo matemático de cámara *pinhole*.

En esta fase, el sistema proyecta continuamente un patrón de círculos asimétricos con un tamaño y en una posición fija. El objetivo es que la cámara detecte el patrón y pueda establecer, una primera aproximación de la matriz de proyección del proyector. Posteriormente, estos parámetros serán refinados en la siguiente fase del proceso de calibrado, mediante la proyección del patrón dinámicamente.

En primer lugar, Se carga el fichero con los parámetros intrínsecos de la cámara y se establecen los puntos del patrón asimétrico en la imagen que se va a proyectar (`imagePoints`) y en el sistema de referencia global (`objectPoints`) utilizando parámetros y distancias fijas.

Se busca el patrón asimétrico en la imagen capturada, para ello se utiliza la función `cv::findCirclesGrid` que intenta determinar si la imagen de entrada contiene una cuadrícula de círculos. Si es así, la función localiza los centros de los círculos. Para la detección de los círculos esta función se apoya en un detector de regiones *(Blob detector)* que configuramos mediante los siguiente parámetros para adaptar la búsqueda en función del tamaño de la imagen y el tamaño de las regiones que se van a detectar.

A continuación, se aplica una binarización de la imagen de entrada que permite la segmentación de los círculos proyectados y realiza una separación del patrón de tablero de ajedrez.

Encontrado el patrón de círculos, obtenemos los vectores de rotación `boardRot` y translación `boardTrans` de los puntos proyectados en el sistema de referencia de la cámara mediante la función `cv:solvePnP`

Para realizar la asociación entre los puntos de la imagen y los del objeto se utiliza la función `cv::calibrateCamera` que estima los parámetros intrínsecos y extrínsecos de la cámara para cada una de las vistas. Recibe los `objectPoints` y los `imagePoints` de todas las capturas que llevamos realizadas y el tamaño de la imagen `addedImageSize` y devuelve la matriz de la cámara `cameraMatrix`, los coeficientes de distorsión `distCoeffs` y un vector con los vectores de rotación y traslación que se ha estimado para cada vista del patrón.

Nuevamente, al igual que en el proceso de calibrado de la cámara, se comprueba que el error de reproyección no es superior al umbral definido.

Este proceso se repite hasta capturar un número de patrones definido en la configuración. Una vez alcanzado el numero de patrones capturados y con un error inferior al umbral pasamos a la siguiente fase de calibrado.

6.1.5 Calibración del sistema estéreo cámara-proyector

En esta fase se refina la calibración de los parámetros intrínsecos del proyector calculados en la fase previa y se realizan los cálculos para obtener los parámetros extrínsecos del sistema estéreo cámara-proyector.

A partir de ahora, se genera el patrón de círculos asimétricos de manera dinámica. A partir de la posición y las dimensiones del patrón de tablero de ajedrez observado por la cámara, con la función `setDynamicProjectorImagePoints` se calcula y genera un patrón de círculos asimétricos que se proyectará junto al de ajedrez.

Se dispone de los puntos 3D en el sistema de referencia global (el patrón), que son los círculos proyectados, y también su proyección (puntos 2D en la imagen) en el sistema de referencia de la cámara y en el sistema de referencia del proyector. Esto significa que se puede utilizar un procedimiento de calibración estéreo como el que proporciona OpenCV.

Consideraremos en nuestro sistema estéreo que la posición y orientación relativa entre la cámara y el proyector es fija. En los paso previos, se ha calculado la pose de un objeto respecto a la cámara (R_1, T_1) y la pose para el mismo objeto pero respecto al proyector (R_2, T_2), entonces esas poses se relacionan entre sí. Esto significa que, dado (R_1, T_1), es posible calcular (R_2, T_2) conociendo la posición y orientación de la segunda cámara con

respecto a la primera cámara. Esto es lo que hace la función descrita. Calcula (R, T) de modo que:

$$R_2 = R * R_1 T_2 = R * T_1 + T, \tag{6.2}$$

Opcionalmente, se calcula la matriz esencial E:

$$E = \begin{bmatrix} 0 & -T_2 & -T_1 \\ T_2 & 0 & -T_0 \\ -T_1 & T_0 & 0 \end{bmatrix} * R \tag{6.3}$$

donde t_i son componentes del vector de traslación T : $T = [T_0, T_1, T_2]^T$. Y la función también puede calcular la matriz F fundamental:

$$F = cameraMatrix_2^{-T} * E * cameraMatrix_1^{-1} \tag{6.4}$$

Mediante la función `cv::stereoCalibrate` se estima la matriz de transformación entre dos cámaras que formen un par estéreo. En nuestro caso la cámara y el proyector.

La función `cv::stereoCalibrate` también permite obtener los parámetros intrínsecos de cada dispositivo, sin embargo debido a la alta dimensionalidad del espacio de parámetros y el ruido que pueden introducir las imágenes de entrada, la función puede divergir de la solución correcta. Por tanto utilizaremos los parámetros intrínsecos calculados en las fases previas para cada uno de los dispositivos de forma individual, proporcionándolos a la función e indicándolo mediante el *flag* CV_CALIB_FIX_INTRINSIC. Esto acción, simplifica los cálculos que tiene que realizar la función y que se realicen cálculos erróneos de unas matrices de proyección que ya se han obtenido previamente con bastante precisión.

Como hemos indicado, debemos proporcionar los puntos del patrón de calibración y las proyecciones de estos puntos observados respecto a la cámara y el proyector. `setDynamicProjectorImagePoints()` nos genera los puntos del patrón asimétrico y almacena la proyección de los puntos respecto de la cámara. La función `calibrationProjector.calibrate()` obtiene los parámetros intrínsecos y la proyección de los puntos del patrón respecto del proyector.

Al estar diseñado el bucle del programa mediante un enfoque *update/draw*, tenemos que proceder en dos iteraciones para dar tiempo a la imagen proyectada a refrescarse antes de tratar de detectarlo. De lo contrario podemos estar detectando el viejo patrón proyectado, pero utilizando los puntos de imagen más recientes.

Con todos los parámetros necesarios calculados:

- ***objectPoints:*** Puntos del patrón generado dinámicamente.

- ***auxImagePointsCamera:*** Proyecciones de los puntos del patrón desde el punto de vista de la cámara
- ***calibrationProjector.imagePoints:*** Proyecciones de los puntos del patrón desde el punto de vista del proyector
- ***cameraMatrix:*** Matriz de parámetros intrínsecos de la cámara
- ***cameraDistCoeffs:*** Coeficientes de distorsión de la cámara
- ***projectorMatrix:*** Matriz de parámetros intrínsecos del proyector
- ***projectorDistCoeffs:*** Coeficientes de distorsión del proyector
- ***calibrationCamera.getDistortedIntrinsics().getImageSize():*** Tamaño de la imagen utilizada.

`cv::stereoCalibrate` devuelve la matriz de rotación `rotation3x3`, el vector de translación `transCamToProj` entre el sistema de referencia de la cámara y el proyector y el error de reproyección obtenido. La función `cv::Rodrigues` transforma la matriz de rotación en el vector de rotación `rotCamToProj` que almacena el proceso de calibrado.

Cada cierto número de iteraciones, configurable en el sistema, se eliminan las capturas del patrón que el error de reproyección supere un determinado umbral. Y finalmente al alcanzar un numero suficiente de calibraciones correctas, se toman los valores de los parámetros intrínsecos y extrínsecos del proyector a los que converge el proceso como los finales, guardándolos en un fichero de tipo YAML al igual que se realizó con los parámetros intrínsecos de la cámara mediante un objeto de tipo cv::FileStorage.

6.1.6 Verificación del proceso de calibrado

Para finalizar, el sistema pasa al modo de validación del proceso de calibrado, donde de manera visual, se comprobará la precisión de la calibración realizada.

- Se cargan los ficheros YAML y se obtienen las instancias de los objetos.
- Se busca el patrón de tablero de ajedrez en la imagen y se extraen los puntos de las esquinas `chessImgPts`.
- Calcula los vectores de rotación `cv::Mat boardRot` y traslación `cv::Mat boardTrans` del patrón a la cámara.
- Selecciona los `ObjectPoints` correspondientes a las cuatro esquinas del patrón y se almacenan en `vector<cv::Point3f>auxObjectPoints`.
- Compone los vectores de rotación y traslación del objeto a la cámara con vectores cámara al proyector para obtener las transformaciones objeto a proyector.
- Proyecta con la función `cv::projectPoints` los puntos de las cuatro esquinas del patrón `auxObjectPoints`, mediante las transformaciones objeto-proyector y los parámetros intrínsecos del proyector.

- Establece los puntos obtenidos en la proyección `vector<cv::Point2f>out` como la salida que tiene que representar el proyector y que deben coincidir físicamente si se ha realizado una buena calibración.

6.2 Subsistema de captura

El módulo de captura de vídeo se encarga de crear y proporcionar fuentes de vídeo de diversa naturaleza para dar soporte al submódulo de *tracking* y registro, al submódulo de interacción natural de usuario, y al submódulo de identificación de documentos.

Es capaz de dar soporte multicámara. Se pueden crear tantas fuentes de vídeo como se disponga en el sistema, que será de alguno de los siguientes tipos:

- **RaspiCam:** Para un rendimiento óptimo en la Raspberry Pi, al menos la cámara principal debería ser de este tipo.
- **Cámara USB:** En caso de no disponer de una raspiCam o si se quieren incluir cámara adicionales para obtener capturas desde otra posición (videoconferencia,...).

OpenCV es la biblioteca de visión por computador más utilizada. Las estructuras que proporciona implementan todas la operaciones necesarias para el tratamiento de imágenes y su uso está ampliamente extendido y aceptado. OpenCV utiliza un tipo de matriz propio, denominado `(cv::Mat)`, adaptado a las necesidades de esta tecnología. Por este motivo, GrayAR utiliza esta estructura internamente para representar imágenes y su tratamiento en las diferentes funciones del sistema.

El uso la cámara raspiCam *(Raspberry Pi Camera Board)*, presenta un inconveniente. El dispositivo no es compatible con *video4linux* y se debe utilizar la API MMAL (Multi-Media Abstraction Layer) sobre OpenMAX, para acceder a los datos de la cámara y transferirlos a la pantalla o codificarlo como imágenes o vídeos. Esto se traduce en que no es posible la utilización de OpenCV para la gestión del dispositivo y la estructura de datos para almacenar la imagen no es compatible con `cv::Mat`.

La solución definida en el submódulo de captura es la implementación en dos componentes: el `VideoCapture`, que proporciona la interfaz de creación y gestión de fuentes de vídeo basada en la biblioteca de visión artificial OpenCV y `RaspiCam` para la gestión de la cámara propia de la Raspberry.

Con la raspiCam se proporcionan 2 aplicaciones: `raspivid` y `raspistill`, para la captura de vídeo e imágenes respectivamente. Las aplicaciones utiliza la API MMAL (Multi-Media Abstraction Layer), para acceder a los datos de la cámara y transferirlos a la pantalla o codificarlo como imágenes o vídeos. A partir del código fuente disponible de estas aplicaciones, Pierre Raufast modificó el programa, para que una vez obtenidos los datos de la cámara, utilizar el *buffer* de memoria para construir estructuras de tipo `CvMat` propias de OpenCV.

Partiendo del trabajo de Pierre, otras personas, como Chris Cummings o Rafael Muñoz Salinas[1], han desarrollado esta solución en C++ para poder obtener las imágenes de la cámara en objetos compatibles con OpenCV. Es precisamente la biblioteca de Rafael Muñoz, la que se ha utilizado a modo de drivers de la raspiCam.

El rendimiento que ofrece en el uso de la raspiCam es de hasta 30 FPS para los modos YUV420, RGB y en escala de grises con imágenes de hasta 1280x960 píxeles. En modo BGR y con un tamaño de 1280x960 el rendimiento cae hasta los 14 fps, ya que el proceso de conversión del espacio de color consume bastantes recursos.

Por defecto, cuando se utiliza esta biblioteca con OpenCV las imágenes obtenidas están en el espacio de color BGR, con lo que el desempeño es menor. Al utilizar una arquitectura cliente-servidor, se decide que la tarea de conversión sea delegada, y sea el servidor el encargado de cambiar al espacio de color de OpenCV cuando la imagen es recibida.

Gracias al diseño, en el que las fuentes de vídeo concretas tienen que respetar un interfaz común, se obtiene un acceso homogéneo. Así, el acceso a los recursos que proporciona cualquier fuente es común a todas ellas:

- La utilización de la biblioteca raspiCam proporciona una interfaz sencilla, sin necesidad de dependencias y compatible con los objetos de OpenCV. De esta forma, todos los subsistemas reciben la misma estructura de datos, independiente del dispositivo.
- Utilizando la misma interfaz de OpenCV, no es necesario un aprendizaje previo para el acceso a las fuentes de vídeo y obtenemos una transparencia que simplifica la utilización de cara al usuario y le abstrae de la implementaciones propias. En resumen, a una cámara se accede siempre igual, independientemente del tipo que sea.
- Para la realización de este proyecto se ha compilado una versión de OpenCV con soporte de archivos de vídeo basado en FFMPEG y con soporte de dispositivos de vídeo basado en Video4Linux2. Gracias a esto, OpenCV encapsula el acceso a cualquier tipo de fuente de vídeo, por lo que obtener las imágenes desde un archivo de vídeo se realiza de la misma forma que de una cámara.

6.3 Subsistema de *tracking* y registro

El cálculo del registro requiere posicionar el sistema cámara-proyector, mediante su posición y rotación, relativo a las hojas de papel que se encuentren en la escena capturada. Los métodos de *tracking*, en general, son los encargados de obtener una estimación de la trayectoria que realiza un objeto.

En GrayAR, se ha optado por implementar un sistema de *tracking* visual basado en una aproximación ***bottom-up***[San07], en la que se calculan los seis grados de libertad de la cámara a partir de lo que se está percibiendo en la imagen.

[1]Grupo de Investigación «Aplicaciones de la Visión Artificial» de la Universidad de Córdoba

El módulo tiene como entrada el *frame* actual, donde va a realizar la búsqueda de cuadriláteros candidatos a ser hojas de papel; los parámetros de calibración de la cámara y el proyector; las dimensiones de la hoja de papel y por último, si la pose se debe calcular para representarse por pantalla o en el proyector.

La salida consistirá en un vector con las hojas de papel detectadas (definidas por las esquinas), así como sus parámetros extrínsecos correspondientes. Estos parámetros extrínsecos, consistentes en una matriz de rotación y un vector de translación, son los que aplicados a un objeto virtual 3D producen la transformación necesaria, para al realizar su representación esté correctamente situado respecto al sistema de referencia del papel.

Conociendo las posiciones 2D de las aristas y vértices que definen la hoja de papel, y el modelo de proyección de la cámara es posible estimar la posición y rotación 3D de la cámara relativamente al documento. Aprovechando que conocemos la estructura de formato normalizado (según ISO 120/DIN 476) de una hoja de papel, con un tamaño previamente conocido nos permite definir un sistema de coordenadas local de cada hoja detectada, de modo que obtengamos la matriz de transformación 4x4 del sistema de coordenadas de la hoja al sistema de coordenadas de la cámara.

Figura 6.5: Imagen de entrada en el módulo paperDetector

El enfoque utilizado es similar al de ArUCo ó ARToolkit, mediante un algoritmo de detección de bordes y un método de estimación de la orientación. Sobre la imagen obtenida se

inicia el primer paso de búsqueda de hojas de papel. La imagen se convierte a blanco y negro para facilitar la detección de cuadriláteros; primero se convierte a escala de grises, y después se binariza eligiendo un parámetro de umbral *«threshold»* que elige a partir de qué valor de gris (de entre 256 valores distintos) se considera blanco o negro. A continuación el algoritmo de visión por computador extrae componentes conectados de la imagen previamente binarizada,cuya área es suficientemente grande como para ser una hoja de papel. A estas regiones se les aplica un algoritmo de detección de contornos, obteniendo a continuación los vértices y aristas que definen la región de la hoja en 2D.

6.3.1 Conversión a escala de grises

El primer paso en el proceso de detección, consiste en convertir el *frame* capturado por la cámara a escala de grises mediante la función `cv::cvtColor`. Esta función convierte una imagen de un espacio de color a otro. La constante `CV_BGR2GRAY` sirve para indicar que queremos pasar de una imagen BGR (estándar de OpenCV) a otra en escala de grises.

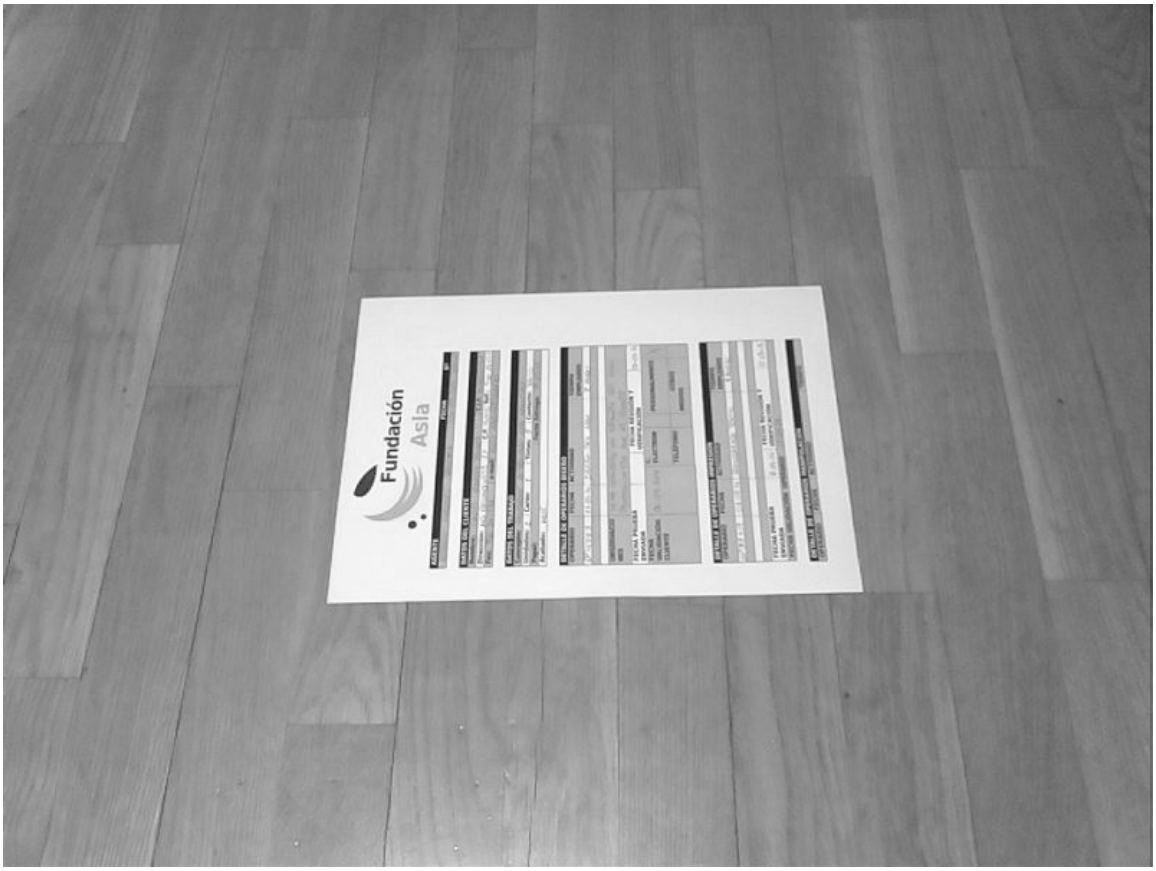


Figura 6.6: Imagen en escala de grises

6.3.2 Umbralización o binarización

La umbralización consiste en definir un valor umbral y compararlo con cada uno de los píxeles de la imagen. A los píxeles que estén por debajo del umbral se les asigna un valor, y a los que estén por encima otro. De esta forma se divide toda la población de valores

en tan sólo dos grupos, reduciendo considerablemente la complejidad de la información a analizar.

En GrayAR, se han desarrollado tres soluciones distintas para tratar de resolver este problema del *thresholding* básico (fijo, adaptativo y el método Canny). En un principio, sólo se implementaron los métodos fijos y adaptativo, pero en una fase de optimización realizada posteriormente, se incluyo el método Canny.

La ventaja aportada por el algoritmo de Canny respecto a los anteriores es que proporciona bordes más precisos, y es tolerante a las variaciones de iluminación, produciendo menores índices de ruido en la imagen binarizada.

Figura 6.7: Binarización por el método de Canny

6.3.3 Extracción de contornos

El siguiente paso del proceso es la detección de contornos, una operación un tanto distinta a las anteriores por dos motivos principales. El primero es que no consiste en iterar aplicando una misma función de forma monótona sobre todos los píxeles de la imagen. Y el segundo, es que el resultado que se obtiene no es una nueva imagen, sino un colección de conjuntos de puntos sobre la imagen.

Para este paso se parte de la imagen resultante del paso anterior. Es decir, de una imagen que tan sólo consta de píxeles con valor cero o uno. Sobre ella se aplica un algoritmo que

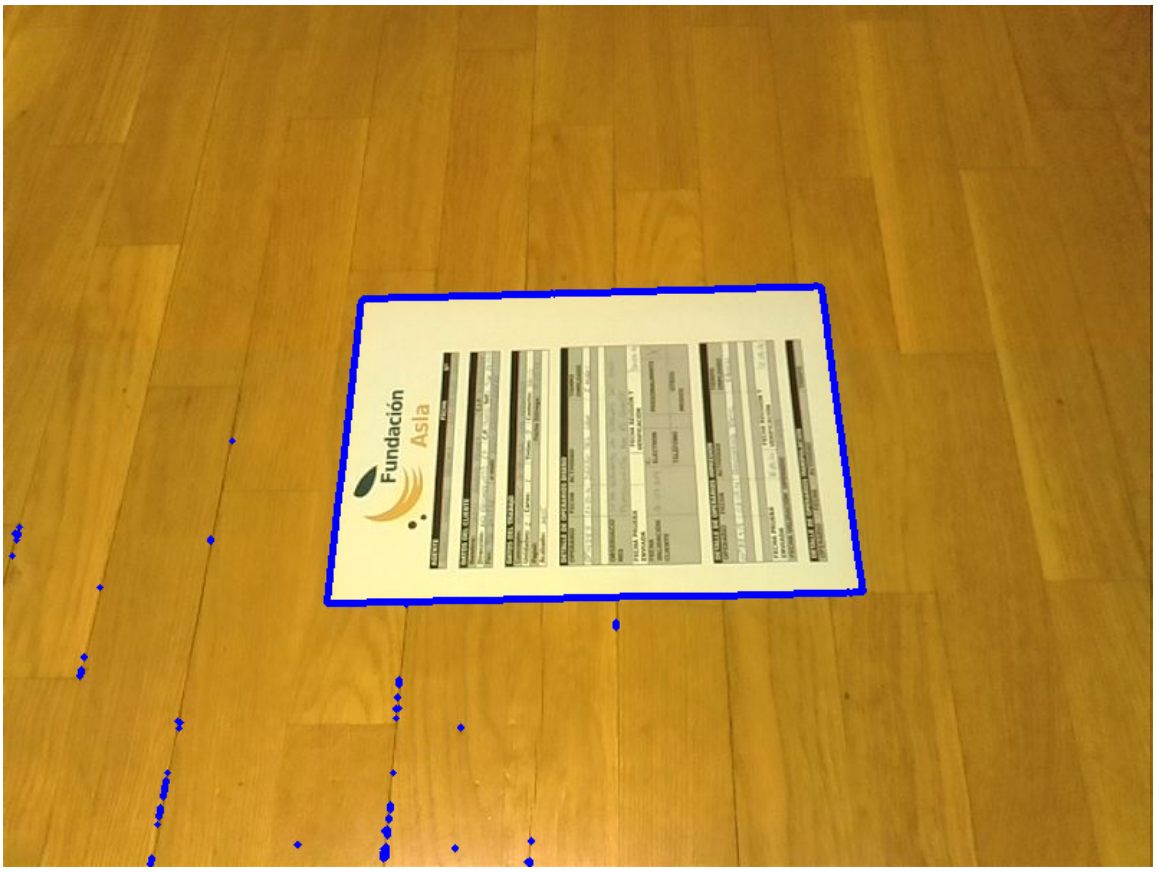


Figura 6.8: Contornos encontrado en la imagen binarizada

la barre, empezando por su esquina superior izquierda, a la busca de un primer píxel a uno, y que cuando lo encuentra es capaz de seguir la cadena de píxeles con valor uno que se encuentran unidos a él hasta volver al píxel de partida. Esa cadena de píxeles a uno encontrados se denomina contorno. Y es más, el algoritmo es capaz de encontrar todos los contornos presentes en la imagen, ya que cuando termina con uno empieza de nuevo el proceso hasta asegurarse de haber barrido la imagen por completo.

En la documentación, así como la referencia de implementación, se refiere al *paper* original «Topological structural analysis of digitized binary images by border following» de Satoshi Suzuki and Keiichi Abe.

GrayAR utiliza la función `findContours` que implementa este algoritmo. Básicamente hay un contador de contornos encontrados y un *buffer* de píxeles recorridos. El contador se inicializa a cero y el *buffer* con una copia de la imagen original. Se barre el *buffer* de arriba abajo y de izquierda a derecha. Una transición de un píxel 0 a otro 1 indica que se ha detectado un borde exterior, momento en que se suma uno al contador de contornos encontrados y se buscan todos los píxeles 1 vecinos del encontrado. Si el píxel no tiene vecinos a 1 se cambia el valor del píxel por el del contador con signo contrario y se empieza otra vez con el siguiente píxel. En caso contrario se cambia el valor del píxel por el del contador, excepto que su valor sea mayor que 1, lo que significa que ya ha sido visitado, y se

continúa buscando vecinos a 1 hasta retornar al píxel inicial. Una transición de un píxel con valor igual o mayor que 1 a otro 0 indica que se ha detectado un borde interior, momento en que se repite el mismo proceso usado para los contornos exteriores.

El algoritmo presenta una pequeña dificultad en los bordes de la imagen, y para solventarlo presupone que la imagen está rodeada por los cuatro lados de píxeles con valor 0. Es decir, que el *buffer* que utiliza tiene una fila más por encima y por debajo, y una columna más a derecha e izquierda, que la imagen original.

Gracias a este algoritmo, no sólo se obtienen los contornos exteriores, como ocurre con otras implementaciones, sino también los interiores a otros, y retornarlos clasificados jerárquicamente.

Otra ventaja de esta implementación, al devolver todos y cada uno de los puntos que forman el contorno, es que permite tomar algunas decisiones tempranas, como por ejemplo descartar los contornos que no tengan un mínimo de puntos. Es decir, aquellos que no son susceptibles de formar la hoja de papel.

6.3.4 Aproximación a polígonos

Los contornos de los que se parte para este paso son colecciones de puntos del paso anterior. El objetivo de este método es obtener el polígono que mejor se ajuste a cada uno de los contorno de entrada.

En primer lugar, sobre aquellos contornos que cumplan con la restricción de superficie, se calcula la **envoltura convexa del contorno**. El objetivo del cálculo de la envoltura es la de obtener un contorno *limpio*, ya que tras este paso se han eliminado puntos interiores debidos a reflejos, sombras o ruido en la imagen que se hubiesen detectado.

Matemáticamente se define la envoltura convexa de un conjunto de puntos X de dimensión n como la intersección de todos los conjuntos convexos que contienen a X.

Dados k puntos $x_1, x_2, \dots x_k$ su envoltura convexa C viene dada por la expresión:

$$C(X) = \left\{ \sum_{i=1}^{k} \alpha_i x_i \;\middle|\; x_i \in X,\, \alpha_i \in \mathbb{R},\, \alpha_i \geq 0\,, \sum_{i=1}^{k} \alpha_i = 1 \right\} \tag{6.5}$$

Es decir, teniendo un conjunto de puntos en el plano, su envoltura convexa está definida por el polígono convexo de área mínima que cubre todos los puntos (esto es, todos los puntos están dentro del polígono).

OpenCV proporciona la función `cv::convexHull` para encontrar la envoltura de una conjunto de puntos 2D utilizando el algoritmo de Sklansky con una complejidad $O(NlogN)$ en su actual implementación.

A continuación, la conversión de una colección de puntos en un polígono se realiza mediante el algoritmo de Douglas-Peucker. El algoritmo admite como entrada una lista de puntos y una distancia máxima permitida. Define un segmento que va desde el primer punto de la lista hasta el último, y calcula la distancia más corta que hay desde dicho segmento a todos y cada uno del resto de puntos de la lista. Si encuentra un punto a una distancia mayor que la máxima pasada como parámetro divide la lista y el segmento en dos, utilizando el punto encontrado como nuevo extremo. Y vuelve a empezar el proceso comprobando cada nueva lista contra cada nuevo segmento de forma individual, creando nuevas listas y segmentos si fuera necesario, de forma recursiva.

Esta implantación se realiza en GrayAR mediante la función `approxPolyDP`. GrayAR utiliza como distancia máxima para al algoritmo el 10 % del número de puntos que tiene cada contorno de forma individual.

Una vez convertidos los puntos a polígonos se descartan los que no tengan cuatro lados.

En este punto, en los casos en los que no existan grandes solapamientos sobre el papel, debemos obtener una serie de polígonos que se corresponden con cada una de las hojas detectadas en la imagen.

Gracias a la utilización del cálculo de la envoltura convexa como paso previo, nos permite que existan pequeños solapamientos en los bordes, y aún así, se siga detectando el contorno del papel. Sino, las oclusiones supondrían una interrupción en la conectividad de los contornos, y por lo tanto, el fallo en el proceso de detección.

6.3.5 Búsqueda de cuadriláteros en el espacio de Hough

Mientras se este realizando alguna interacción con el documento, nuestra mano, dedos e incluso parte del brazo estarán solapando gran parte de la hoja que queremos detectar. Tras aplicar el detector de bordes a la imagen recibida, obtenemos segmentos no conectados que las funciones anteriores no serán capaces de clasificar como cuadriláteros candidatos a ser una hoja de papel.

En estos casos, la búsqueda de la hoja de papel en la imagen se debe realiza mediante detectores de líneas basados en la transformada de Hough.

Esta función se invocará sólo cuando no se hayan detectado cuadriláteros mediante el algoritmo anterior, ya que esta función es costosa computacionalmente debido al tamaño del espacio de búsqueda.

El resultado será, al igual que la función anterior, un vector con todos los cuadriláteros que corresponden a las hojas de papel detectadas.

La transformada de Hough es una técnica para detectar bordes llevando los puntos al espacio paramétrico donde se transforman en rectas.

Figura 6.9: Polígonos en encontrados en la imagen

Basándose en lo anterior, la recta $y = m*x+n$ se puede representar como un punto (m, n) en el espacio de parámetros. Sin embargo, cuando se tienen rectas verticales, los parámetros de la recta (m, n) no están definidos. Por esta razón se utilizan los parámetros que describen una recta en coordenada polares, denotados (ρ, θ).

El parámetro ρ representa la distancia entre el origen de coordenadas y el punto(x, y), mientras que θ es el ángulo del vector director de la recta perpendicular a la recta original y que pasa por el origen de coordenadas.

Usando esta parametrización, la ecuación de una recta se puede escribir de la siguiente forma:

$$y = (-\frac{\cos\theta}{\sin\theta}) * x + (\frac{\rho}{\sin\theta}) \tag{6.6}$$

que se puede reescribir como

$$\rho = x * \cos\theta + y * \sin\theta \tag{6.7}$$

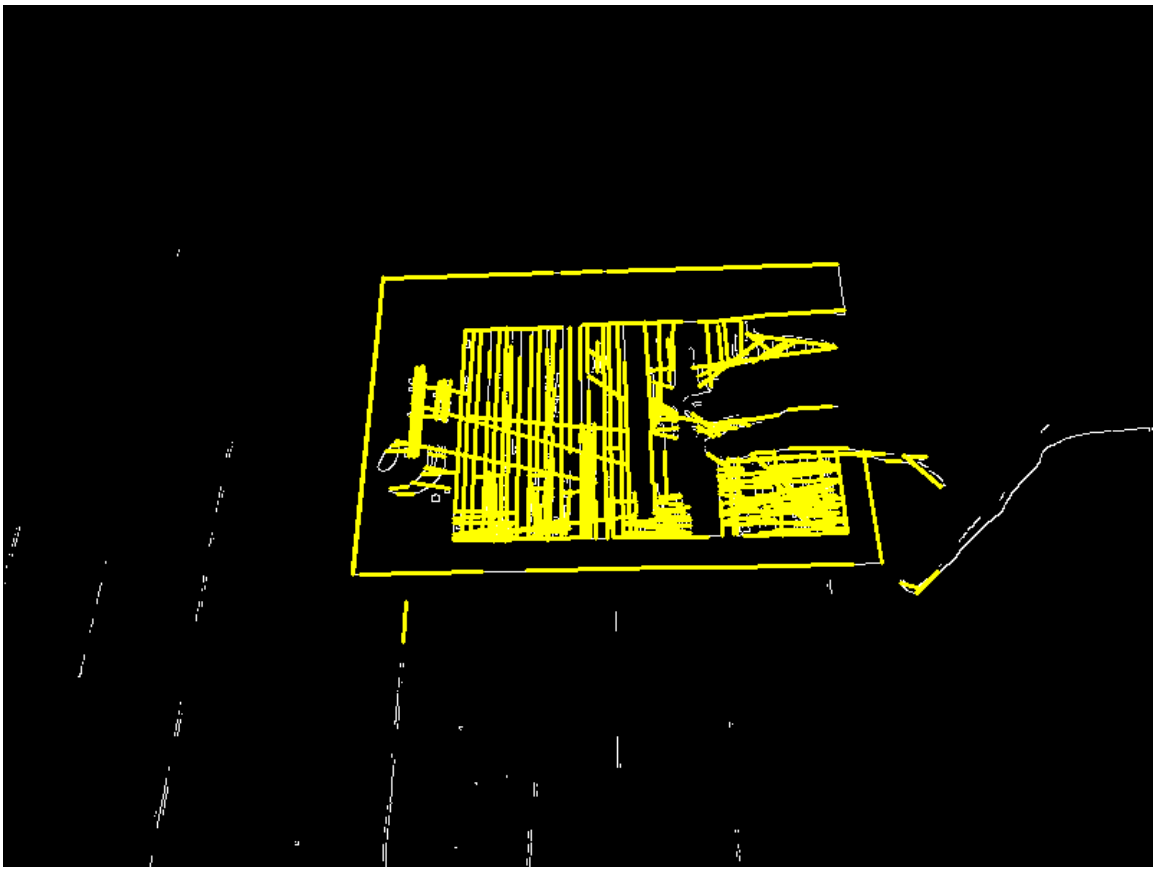


Figura 6.10: Detección erronea al existir solapamiento

Entonces, es posible asociar a cada recta un par (ρ, θ) que es único si $\theta \in [0, \pi)$ y $\rho \in \mathbb{R}$ ó $\theta \in [0, 2\pi)$ y $\rho \geq 0$. El espacio (ρ, θ) se denomina espacio de Hough para el conjunto de rectas en dos dimensiones.

Para un punto arbitrario en la imagen con coordenadas (x_0, y_0), las rectas que pasan por ese punto son los pares (ρ, θ) con $\rho = x * \cos\theta + y * \sin\theta$ donde ρ (la distancia entre la línea y el origen) está determinado por θ. Esto corresponde a una curvas sinusoidal en el espacio (ρ, θ), que es única para ese punto. Si las curvas correspondientes a dos puntos se intersecan, el punto de intersección en el espacio de Hough corresponde a una línea en el espacio de la imagen que pasa por estos dos puntos. Generalizando, un conjunto de puntos que forman una recta, producirán sinusoides que se intersecan en los parámetros de esa línea. Por tanto, el problema de detectar puntos colineales se puede convertir en un problema de buscar curvas concurrentes.

Las zonas de este espacio donde más líneas se cortan se convierten en parámetros de posibles rectas en el espacio de la imagen.

Se seleccionan un conjunto de las rectas detectadas para ser examinado en función de las siguientes características:

Figura 6.11: Imagen de entrada

- El segmento se encuentra dentro o en la proximidades de la zona de proyección del sistema. Con esta restricción descartaremos aquellos segmentos que sabemos de antemano que no van a ser hojas de papel, como por ejemplo, otros objetos que se encuentren en la imagen o los bordes de la superficie donde se encuentra en prototipo.
- El segmento es mayor que un valor umbral.
- Sólo se tienen en cuenta los segmentos más próximos a los límites de proyección. El detector de Hough puede encontrar rectas en el interior de la hoja de papel que estamos buscando. Así sólo seleccionaremos los posibles candidatos a formar el borde del papel.
- Los ángulos que formen dos rectas contiguas estará comprendido dentro del rango de 75^o a 105^o.
- Los ángulos que formen dos rectas opuestas deben ser cercanos a 180^o ó 360^o.

Tras la selección anterior obtendremos un conjunto de segmentos de rectas extraídas de la imagen. Cuatro segmentos se consideran que formarán un cuadrilátero candidato si cumplen los siguientes criterios:

- Las intersecciones de los cuatro segmentos están dentro de la imagen y próximos a la zona de proyección.

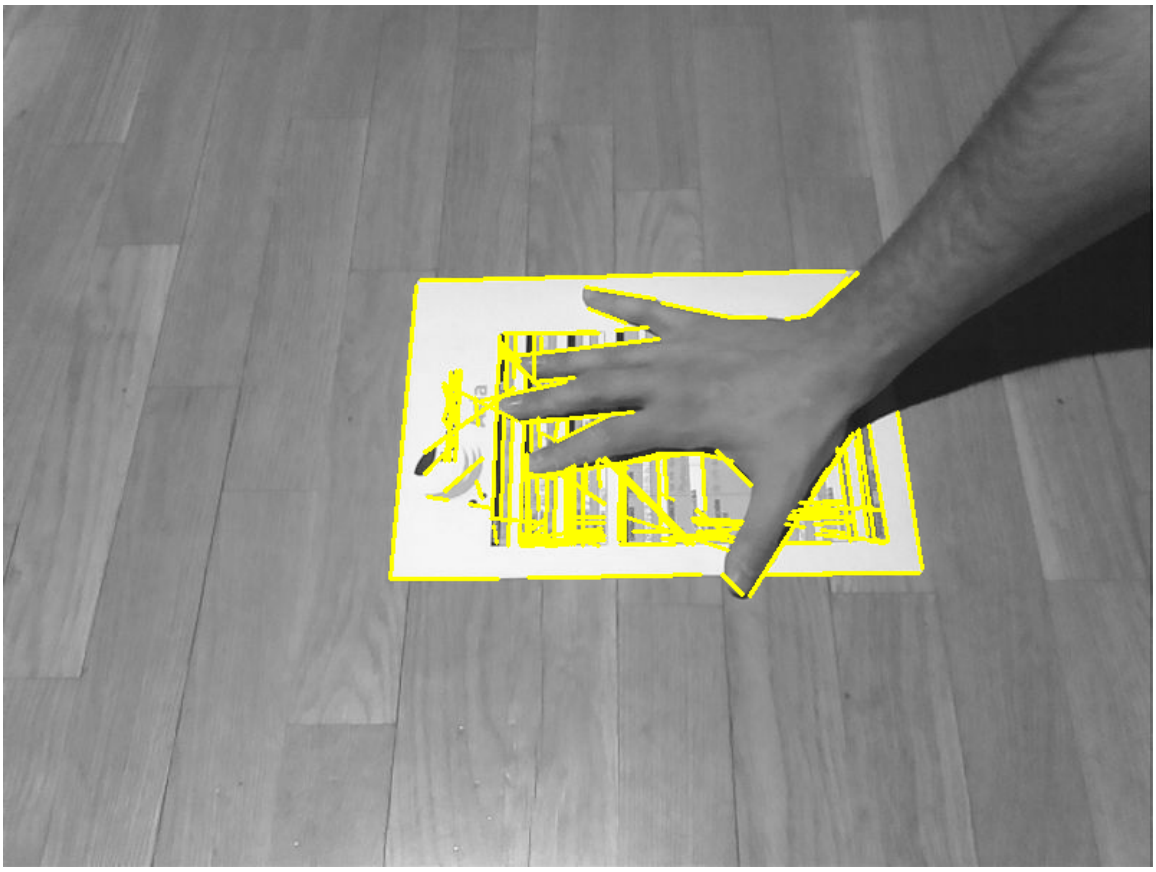


Figura 6.12: Segmentos dentro de la zona de proyección

- Sólo intersecan en cuatro puntos.
- El área del cuadrilátero delimitado por las intersecciones esta entre 47000 y 49000 píxeles, ya que garantiza que el cuadrilátero que se detectó representa una hoja de papel.

Para ello se generan las posibles combinaciones de 4 segmentos del conjunto de rectas detectadas y se analizan los criterios anteriores.

6.3.6 Estimación de la pose

EL reto principal en un sistema de realidad aumentada es la estimación de la *pose*. Básicamente consiste en obtener, en función del sistema de referencia del dispositivo de representación, *donde* se debe colocar el objeto virtual, que *orientación* tiene y con que *tamaño*, por medio de matrices de translación, rotación y escalado respectivamente.

La solución a este problema se ha encapsulado mediante la clase `Paper`. Esta clase representa una hoja de papel detectada por el sistema, mediante el almacenamiento de los puntos correspondientes a sus esquinas. Se ha diseñado mediante el patrón *Decorator* aplicado a un vector de `cv::Point2f`, añadiendo las funciones necesarias calcular los parámetros extrínsecos de la cámara respecto a las cuatro esquinas del papel almacenadas en el vector.

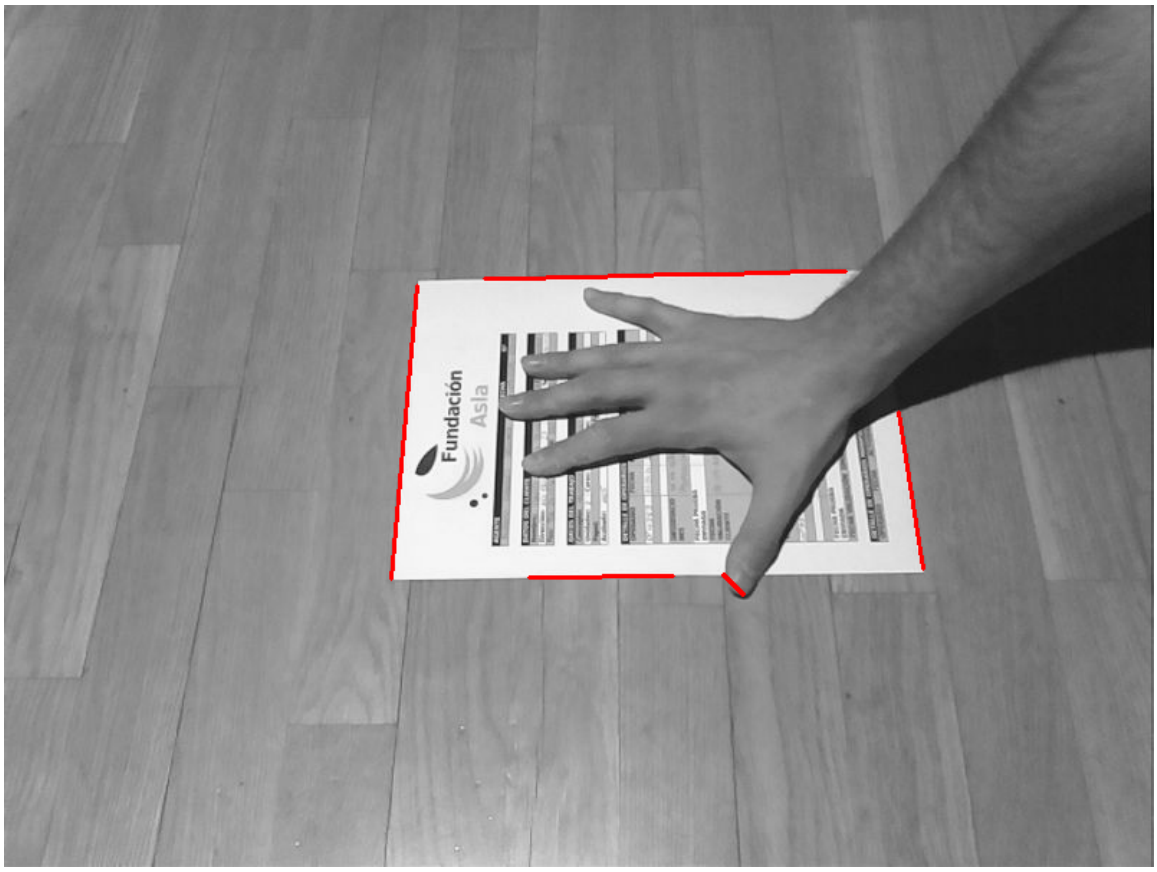


Figura 6.13: Segmentos más externos y mayores que un umbral

Para convertir las coordenadas del mundo a coordenadas de pantalla, de espacio tridimensional a bidimensional, se utilizará la fórmula 4.12 desarrollada en la sección 2:

$$q = K[R|t]Q \tag{6.8}$$

donde

$$q = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{6.9}$$

$$[R|t] = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & T_1 \\ R_{2,1} & R_{2,2} & R_{2,3} & T_2 \\ R_{3,1} & R_{3,2} & R_{3,3} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

Por tanto, q representa el espacio bidimensional en coordenadas homogéneas, y Q el espacio tridimensional. f_x es la distancia focal en el eje X, f_y la distancia focal en el eje Y, (c_x, c_y) marcan el punto principal, el punto donde el eje de visión corta el plano de visión (normalmente suele estar en el centro de la imagen, o muy cerca).

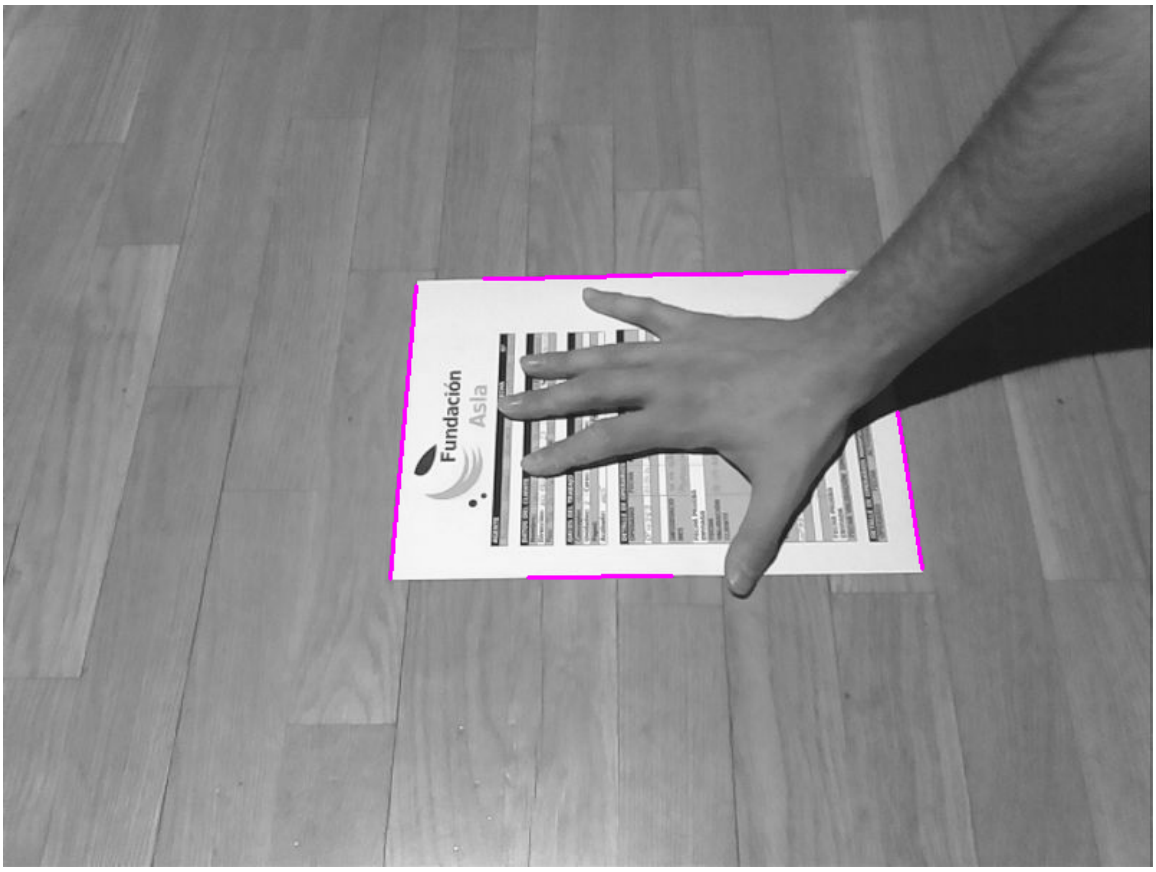


Figura 6.14: Segmentos que forman 90º ó 180º

El cálculo de la pose en función del dispositivo de representación, se ha definido en la función `calculateExtrinsics`. Esta función es una interfaz común para obtener la *pose* tanto para la cámara como para el proyector. Recibe como argumentos las dimensiones de la hoja de papel y un objeto de la clase `cameraProjectorSystem` que encapsula las matrices de proyección de ambos dispositivos. Adicionalmente, y como parámetros opcionales, admite dos valores *booleanos*. El primero, `setYPerperdicular` indica si considera el sistema de coordenadas cartesianas con el eje Y vertical, y el segundo, para definir el dispositivo de representación, y por tanto seleccionar una u otra matriz de parámetros intrínsecos a la hora de realizar los cálculos.

Mediante la función de OpenCV `cv::solvePnp`, se calcula la *pose* (rotación y traslación $[R|t]$) de la hoja de papel. Dado un conjunto de puntos del objeto (`ObjPoints`), sus proyecciones en la imagen correspondiente (`ImagePoints`), así como la matriz de la cámara (ó del proyector) (K) (`camMatrix`) y los coeficientes de distorsión (`distCoeff`). Esta función encuentra una *pose* que minimiza el error de reproyección, es decir, la suma de los cuadrados de las distancias entre las proyecciones `imagePoints` observados y los `objectPoints` proyectados.

Figura 6.15: Cuadrilátero formado por lo segmentos y que cumple las condiciones

Se establece el centro del papel como centro de coordenadas $(0, 0, 0)$, por lo que las esquinas que se le proporcionan a la función son:

$$ObjPoints = \begin{bmatrix} -w/2 & -h/2 & 0 \\ w/2 & -h/2 & 0 \\ w/2 & h/2 & 0 \\ -w/2 & h/2 & 0 \end{bmatrix} \tag{6.11}$$

Siendo h y w la altura y anchura de la hoja de papel respectivamente.

Se debe conocer cómo OpenCV enumera las esquinas del papel. Dado un cuadrilátero, la primera esquina corresponde a la que se encuentra situada más a la derecha de la imagen. Las consecuencias son que para ciertas posiciones, las dimensiones de los segmentos que forman los lados del papel tienen un valor, y en otras posiciones, esos mismos segmentos tienen otra medida. En resumen, para el cálculo de los parámetros extrínsecos, considera una distribución distinta en entre los puntos de la imagen y los puntos del objeto.

La solución implementada ha sido comparar la longitud de los segmentos detectados, para determinar en que posición se están enumerando las esquinas, y aplicar una enumeración acorde al establecido en los puntos del objeto.

OpenGL trabaja con matrices de modelo *(ModelViewMatrix)* y de proyección *(Projection-Matrix)* para realizar las transformaciones necesarias en el proceso de representación de un objeto 3D. La función `getGetModelViewMatrix` es la encargada de realizar las operaciones sobre las matrices de rotación y translación para obtener una matriz tipo `float` equivalente y compatible con OpenGL.

Las ventajas al utilizar el patrón *Decorator* en la clase `Paper`, nos permite extender su funcionalidad de una manera más flexible que con la herencia. La parametrización de los ejes cartesianos y del dispositivo de representación, permiten la utilización de distintos sistemas de representación gráfica, como OGRE3D u OpenGL; y dispositivos como cámaras o proyectores, sin la necesidad de realizar modificaciones en la clase.

6.3.7 Refinamiento de la *pose*

El posicionamiento calculado puede variar en cada frame, aun cuando el papel no se haya movido realmente. Esto puede ser debido a variaciones en la iluminación que alteran sensiblemente los procesos de detección o errores debido a la naturaleza imprecisa de los métodos utilizados.

Este refinamiento de las percepciones se basa en la capacidad del sistema de almacenar las últimas *n* percepciones. Este parámetro es configurable en tiempo de compilación.

Cuando se recibe una última percepción, se calcula una media ponderada con las cuatro últimas percepciones, de forma que la percepción refinada es la resultante de la siguiente fórmula:

$$P_{actual} = P_1 * 0{,}5 + P_2 * 0{,}25 + P_3 * 0{,}15 + P_4 * 0{,}1 \tag{6.12}$$

Siendo P_1 la percepción más reciente, y el resto las percepciones almacenadas en el histórico. De esta forma, se da más peso a la percepción más actual, pero se tiene en cuenta las anteriores. El resultado es un movimiento más fluido y suave, aunque da la sensación de haber un *delay* al momento de representar la posición. Este retraso es lógico, al tener en cuenta percepciones pasadas.

6.4 Subsistema de identificación de documentos

EL objetivo de este módulo es la identificación rápida de documentos empleando algoritmos de recuperación de imágenes, comparando el documento que está siendo analizado con una base de datos de documentos conocidos por el sistema.

El prototipo construido ha sido configurado para funcionar con varios documentos proporcionados por la asociación ASPRONA. Estos documentos son partes de trabajo reales que se utilizan en un taller de serigrafía que tiene la asociación. Otra sugerencia realizada, y que han puesto de manifiesto que sería de gran utilidad de cara una implantación real, es el reconocimiento de facturas.

Como ya se ha explicado en el capitulo del estado del arte, las técnicas para la identificación de documentos se pueden dividir en dos categorías:

- Basados en la **búsqueda de coincidencias de características locales.**
- Basados en **combinación coincidencias de características locales y la distribución del contenido** dentro de la página.

La solución implementada en GrayAR, debido a los requerimientos anteriores, corresponde a la primera de las técnicas. Es la más adecuada para la identificación de documentos estructurados o semi-estructurados, como pueden ser el caso de formularios, facturas, billetes de tren o tickets.

A la hora de elegir entre los detectores basados en la extracción de características invariante que dispone OpenCV, se ha preferido SURF frente otros descriptores como SIFT ó BRISK por las siguientes razones:

- **Velocidad de cálculo** considerablemente superior al resto de los detectores, sin ocasionar perdida del rendimiento.
- **Más robusto** ante posibles transformaciones de la imagen.
- **No necesita ningún paso previo de segmentación** y la identificación es para obtener documentos similares (no idénticos) a la imagen utilizada como consulta.

A continuación se detallan los pasos que realiza el algoritmo.

6.4.1 Eliminación de la transformación de perspectiva

Este paso tiene como entrada la imagen en escala de grises obtenida por la cámara y los cuadriláteros candidatos encontrados en el módulo de detección de hojas de papel. Para cada hoja de papel, extrae la región de la imagen que cubre cada una de ellas eliminando la transformación de perspectiva, es decir, la deformación que se produce en el papel debido a la perspectiva.

GrayAR utiliza dos funciones de OpenCV para este paso:

- `cv::getPerspectiveTransform` calcula una matriz que multiplicada por un punto en el cuadrilátero origen devuelve un punto equivalente en el cuadrado destino.
- `cv::warpPerspective` acepta una imagen, un cuadrilátero, una matriz de transformación, y retorna una nueva imagen del área cubierta por el cuadrilátero sobre la que se ha aplicado la matriz de transformación para eliminar la deformación debida a la perspectiva.

El resultado de este proceso es una imagen de cada una de las hojas de papel encontradas. Una para cada cuadrilátero candidato. GrayAR utiliza un tamaño de 600x420 píxeles para

(a) Imagen de Entrada (b) Cuadrilátero extraído

Figura 6.16: Eliminación de la transformación de perspectiva

las imágenes destino ya que estas dimensiones mantienen la proporcionalidad de tamaño del folio.

Aunque SURF es invariante a la rotación, translación y escala, realizar este paso previo nos proporciona varias ventajas. Por una lado, se puede realizar la identificación de varios documentos dentro de la misma imagen, y al extraer de la imagen inicial el documento a identificar, estamos eliminado *ruido* que podría influir en el resultado de la detección, aumentado la robustez del algoritmo mediante el análisis exclusivo de la región a identificar dentro de la imagen general.

6.4.2 Extracción de Características

El proceso de extracción consiste en la búsqueda de zonas en la imagen con diferente apariencia que las que están a su alrededor, denominadas características *(features)*. Normalmente las características suelen ser bordes, esquinas o zonas más brillantes u oscuras en función del algoritmo utilizado en particular.

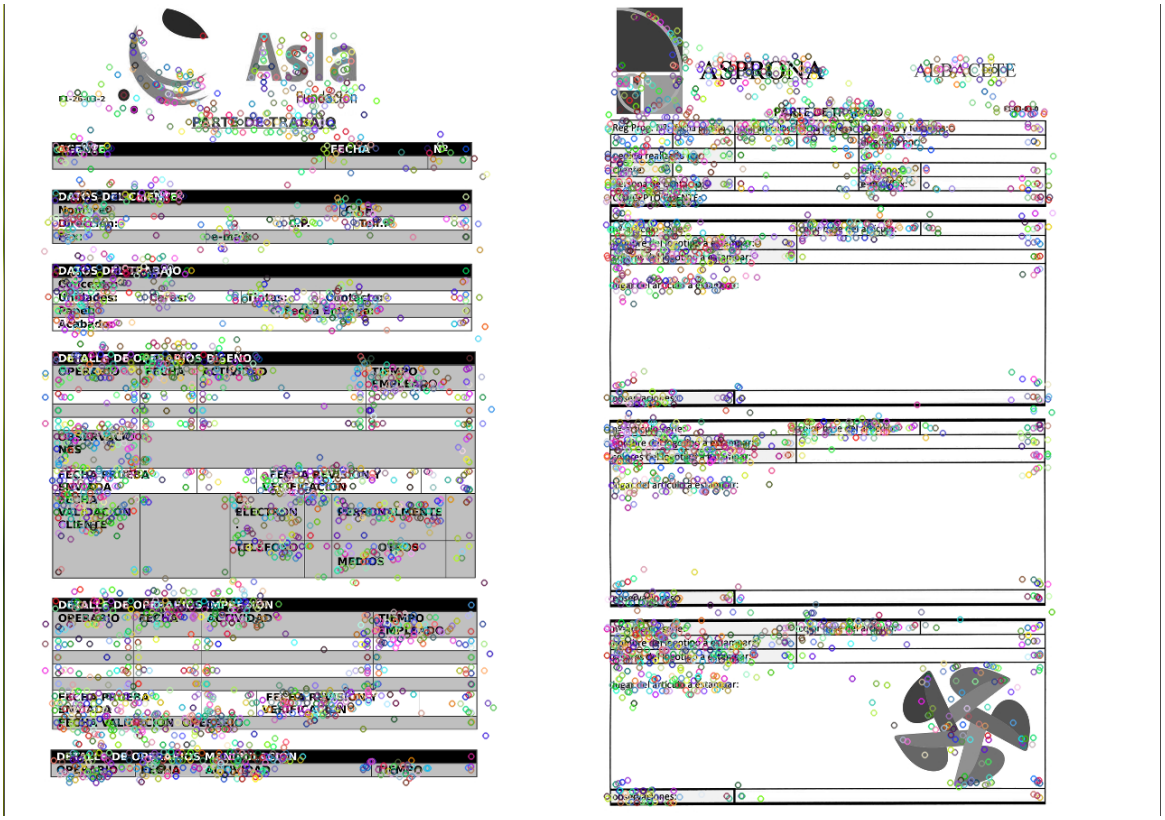
OpenCV proporciona la clase wrapper **FeatureDetector** con una interfaz común que permite cambiar fácilmente entre diferentes algoritmos.

La función **detect** extrae los puntos clave de la imagen que cumplan el umbral para el valor del determinante de la matriz hessiana, definido en el constructor del extractor. Estos punto se almacenan en una estructura tipo `vector<cv::KeyPoint>`.

KeyPoint es una clase genérica que ha sido diseñada para almacenar puntos significativos (con la ubicación, orientación, escala y alguna información adicional).

6.4.3 Generación de Descriptores

La clase **DescriptorExtractor** es la encargada de calcula el vector que describe la característica de un punto significativo para la posterior comparación entre otros puntos de



(a) KeyPoints en el formulario ASLA (b) KeyPoints en el formulario ASPRONA

Figura 6.17: Puntos de interés

interés. SURF utiliza el histograma de gradientes, calculado a partir de la cuantificación de los gradientes dentro de una área local. Una zona se divide en subregiones y se calcula el histograma de gradiente en cada una de ellas.

Mediante la función **compute** calcula los descriptores de los puntos clave detectados en la imagen. Un descriptor se representa como un vector de dimensión fija de un tipo básico. La mayoría de los descriptores siguen este patrón, ya que simplifica el cálculo de las distancias entre los descriptores. Por lo tanto, un conjunto de descriptores se representa como un `cv::Mat`, donde cada fila es un descriptor de un punto clave.

6.4.4 Búsqueda de Coincidencias

Para buscar una coincidencia entre varias imágenes, se extraen los descriptores de la imagen de consulta y se accede a la estructura de descriptores de las imágenes referencia con los datos del vector de consulta calculado, devolviendo el vector almacenado más similar.

Si el vector de consulta está vacío, significa que no se han podido extraer descriptores del documento. En el sistema, esto ocurre cuando se coloca la hoja por la parte posterior, que es totalmente blanca y el extractor no puede encontrar puntos de referencia.

La implementación de la búsqueda se realiza mediante la clase **DescriptorMatcher**. El buscador se configura para que utilice la biblioteca FLANN (Fast Library for Approximate Nearest Neighbors) que contiene una colección de algoritmos para la búsqueda rápida los vecinos más cercanos en grandes conjuntos de datos.

El método de los k vecinos más cercanos *(K-nn)*, es un clasificador supervisado que calcula la probabilidad de que un elemento pertenezca a una clase conocida.

Para su utilización en GrayAR, se configura la clase **Flann** con dos parámetros que especifican el algoritmo a utilizar y su parametrización.

- El primero es **IndexParams** donde se define el índice de búsqueda que se construirá. Para SURF, se recomienda la utilización de árboles k-dimensionales (kd-tree) aleatorios como estructuras de búsqueda, que permiten realizar búsquedas en paralelo.
- El segundo es **SearchParams**, en el que se indica el número de veces que los árboles del índice deben ser recorridos de forma recursiva. Valores altos dan mayor precisión, pero también conllevan un mayor tiempo de procesado. Se utiliza el valor por defecto de 32.

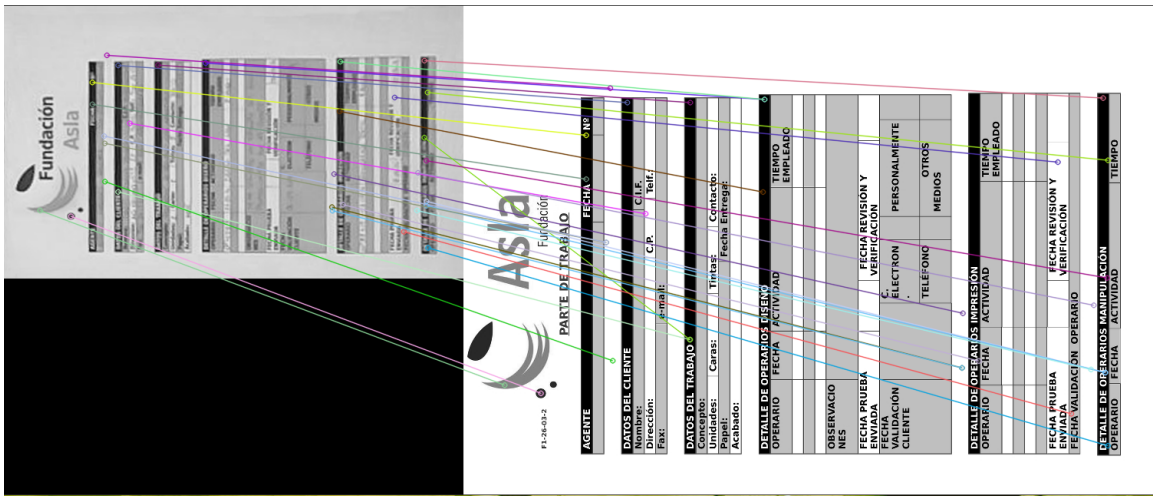
Dos descriptores se consideran coincidentes si la distancia euclídea entre ellos es baja. Para ello se incluye una condición de filtrado basada en NNDR *(Nearest Neighbor Distance Ratio)* con los resultados del buscador FLANN para conseguir búsquedas más robustas y precisas. Una coincidencia de descriptores se eliminará si la distancia desde el descriptor de la consulta a su primer vecino más cercano es mayor que un valor umbral, entre 0 y 1, multiplicado por la distancia al segundo vecino más cercano. Este umbral, por lo general se establece en 0,8 y obliga a que las coincidencias sean únicas. Este requisito se puede hacer más estrictos, reduciendo del valor de este umbral.

Finalmente, al comparar todas las imágenes almacenadas con la imagen de referencia, la que mayor número de coincidencias haya obtenido, será la candidata para devolver su índice como resultado de la identificación.

6.5 Subsistema de interfaz natural de usuario

Interfaz natural de usuario es un término genérico para una variedad de tecnologías que permiten a los usuarios interactuar con los ordenadores en términos humanos. Algunas de estas tecnologías son las basadas en visión por computador y que son capaces desde interpretar expresiones naturales como gestos hasta proporcionar información contextual que se proyecta dentro del campo de visión del usuario.

En GrayAR se ha creado una interfaz de usuario basada en zonas de resaltado y botones virtuales que son proyectados directamente sobre el documento. El usuario podrá interactuar directamente en el espacio físico sin utilizar sistemas de mando o dispositivos de entrada



]

Figura 6.18: Búsqueda de coincidencias de descriptores Imagen 1

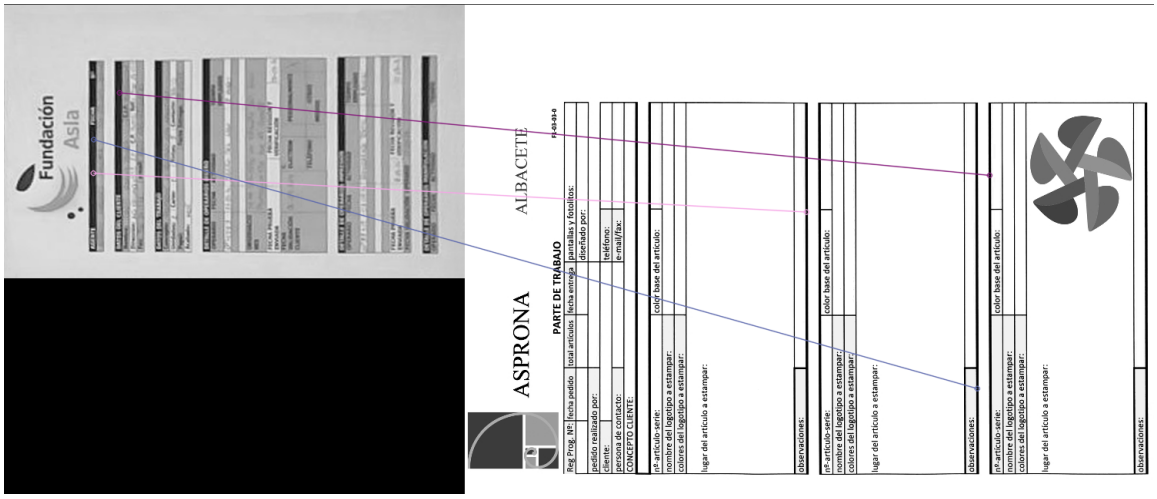


Figura 6.19: Búsqueda de coincidencias de descriptores Imagen 2

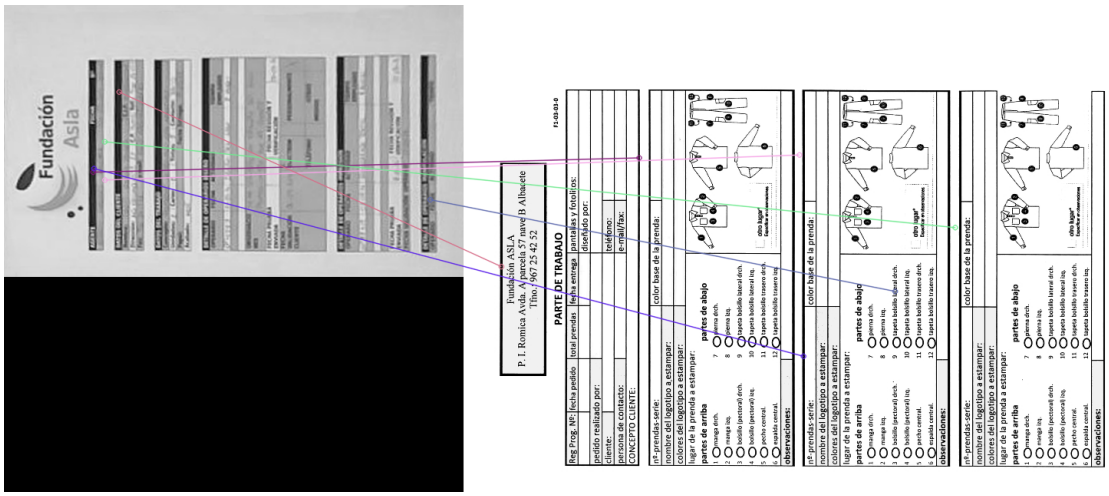


Figura 6.20: Búsqueda de coincidencias de descriptores Imagen 3

tradicionales como sería un ratón, teclado, etc. siendo sustituidos por funciones más naturales como el uso de movimientos gestuales con las manos.

Se ha realizado la implementación del paradigma de superficie táctil interactiva, ampliamente utilizada en los actuales dispositivos portátiles como tablets o smartphones, en la que al pulsar con un dedo sobre la pantalla, reaccione instantáneamente a las acciones del usuario.

Al disponer únicamente de la imagen obtenida por la cámara del sistema, la detección de una pulsación se abstrae a la localización del extremo del dedo índice dentro de una región de interés. Todo ello, realizado mediante técnicas de visión por computador.

6.5.1 Segmentación de la mano

El objetivo de este paso es generar de una máscara que discrimine o diferencie entre los píxeles de una imagen que corresponden al color de la piel y los que no pertenecen. Con está mascara se puede identificar y extraer la mano que aparezca en la imagen y que esté realizando alguna acción sobre el documento a tratar.

Aunque existan diferentes colores de piel, muchos estudios han demostrado que la mayor diferencia se presenta en la intensidad y no el su crominancia, por lo tanto en lugar de utilizar el espacio de color RGB, se utilizará el **espacio de color HSV** (Hue, Saturation, Value), que es más adecuado se aproxima a la percepción humana.

Se utilizan los canales de color H y S (matiz y saturación) para detectar la piel. En el canal H, la piel se compone de zonas pequeñas (representadas en negro), y otras mayores (representadas en gris). En primer lugar, se realiza una **ecualización del histograma** del canal H para conseguir una distribución uniforme. Es decir, que exista el mismo número de píxeles para cada nivel de gris del histograma de del canal.

A continuación, se aplica una **erosión morfológica** al canal para reducir los contornos y separar las zonas similares próximas, a la vez que se eliminan las zonas más claras separadas y amplían los pequeños detalles oscuros.

Las operaciones morfológicas, consisten en la alteración de los píxeles de salida de una imagen dependiendo del valor del píxel de entrada y una relación de vecindad definida por un elemento estructural. El elemento estructural define el tamaño y la forma con la que se aplica la operación. En GrayAR se aplica un elemento en forma de elipse de tamaño 7x7 píxeles.

Para finalizar el tratamiento previo, se aplica un **filtro bilateral** tanto al canal H como al S. El filtra bilateral es una herramienta muy útil en visión por computador ya que permite suavizar las zonas homogéneas de una imagen manteniendo los bordes.

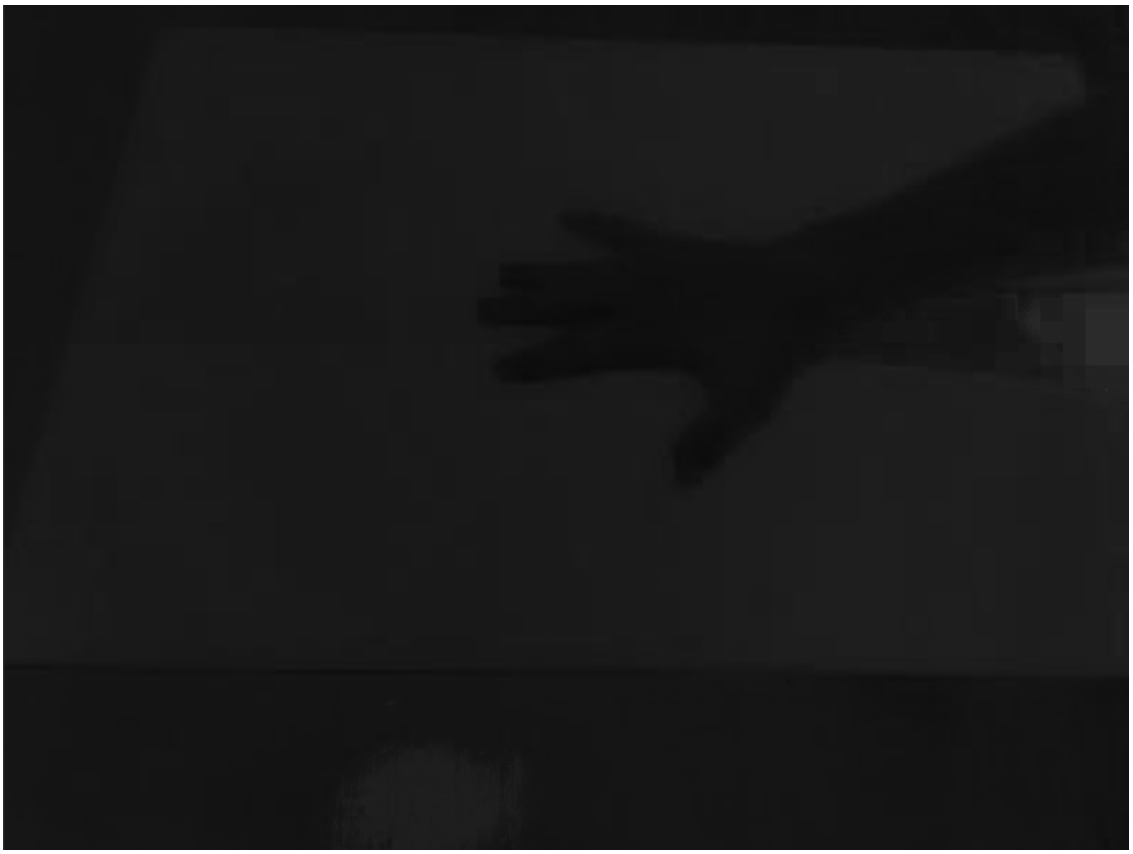


Figura 6.21: Canal H (Espacio de Color HSV)

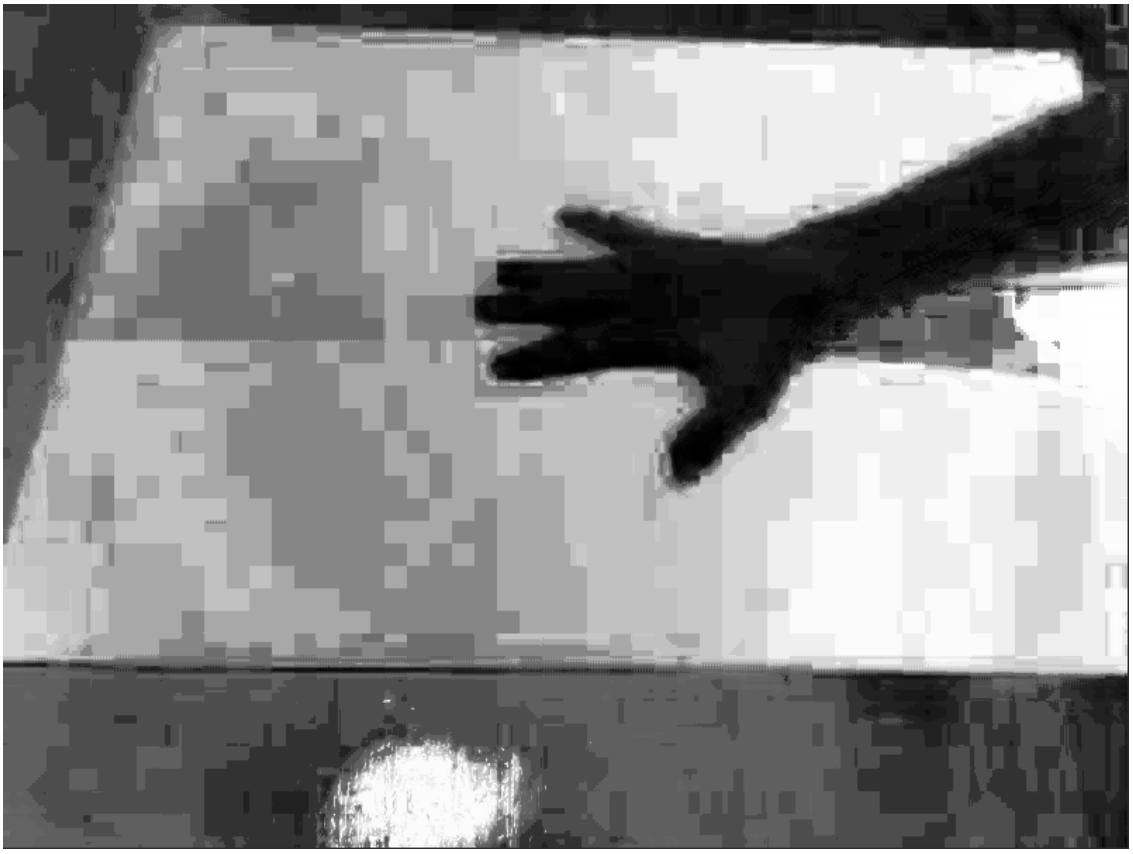


Figura 6.22: Ecualización del Histograma

Figura 6.23: Erosión Morfológica

Figura 6.24: Aplicación de Filtro Bilateral

Figura 6.25: Máscara generada por la mano extraída

En este momento la imagen se encuentra preparada para seleccionar píxeles perteneciente a la piel. Se unifican los tres canales y seleccionamos los píxeles que se encuentren entre los umbrales $0 < H < 25$ y $80 < S < 255$

Finalmente se realiza una **limpieza morfológica** mediante las operaciones de apertura y cierre. La apertura suaviza los contornos, elimina pequeñas protuberancias y elimina las conexiones más débiles. El cierre, rellena vacíos en los contornos y elimina pequeños huecos en la imagen.

El resultado de este tratamiento es una imagen en escala de grises, donde las zonas más claras pertenecen a píxeles de piel y los contornos negros son zonas indeterminadas. Estableciendo un umbral, se seleccionan los píxeles candidatos y **se genera la máscara** que corresponderá a la región ocupada por la mano en la imagen.

6.5.2 Análisis de la imagen para la detección de la mano

Una vez que disponemos en una imagen binaria la región de la mano, se aplica un **detector de contornos** sobre la imagen. En caso de no ser único, el contorno que mayor longitud tenga entre los encontrados, corresponderá a la mano.

Según la disposición del prototipo en relación a posición de los documentos (situado a la izquierda de los papeles), establecemos la restricción de que la mano aparecerá por la par-

te derecha del documento para interactuar con los botones, que también serán proyectados sobre la zona derecha del documento. Además, para la simulación de la pulsación es necesario tener el dedo índice extendido. Por tanto, el extremo del dedo índice corresponde con el **punto del contorno situado más a la izquierda** dentro de la imagen. Seleccionando este punto podemos situar la zona de pulsación dentro de la pantalla.

6.5.3 Cálculo de un punto 3D a partir de un punto en la imagen

La función anterior obtiene el punto del extremo del dedo índice en coordenadas (x, y) de la pantalla. Este dato no sirve para saber donde está colocado el extremo del dedo en relación a las dimensiones reales del papel.

La opción de calcularlo mediante proporcionalidad de los píxeles que forman el papel y sus dimensiones no es válida, ya que la distorsión de la perspectiva generaría un error muy grande.

Para solventar el problema utilizaremos los parámetros intrínsecos y extrínsecos de la cámara calculados de antemano y partiendo de la relación entre los puntos del mundo real y los de la cámara:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K(R \begin{bmatrix} X \\ Y \\ Z_{const} \end{bmatrix} + t) \tag{6.13}$$

donde K es la matriz de la cámara, R la matriz de rotación, t el vector de rotación, y s es el factor de escala de la homografía. $Zconst$ representa la altura donde se están representado los componentes gráficos en relación al papel, en GrayAR es 0.

$$R^{-1}K^{-1}s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z_{const} \end{bmatrix} R^{-1}t \tag{6.14}$$

Resolviendo la ecuación anterior sustituyendo los puntos (x, y) de la pantalla en (u, v) para conseguir s, obtenemos el punto 3D relativo al papel, es decir, situando el centro de coordenadas $(0, 0, 0)$ en el centro de la hoja.

6.6 Subsistema de soporte y utilidades

6.6.1 Gestor de configuración

Existen muchos parámetros configurables en GrayAR. En las primeras versiones del proyecto estos parámetros estaban implementados en variables dentro del programa y cada vez que era necesario modificar estos parámetros, por ejemplo para la realización de pruebas, era necesario recompilar el código. Para los archivos que se encuentran en la Raspberry, el

tiempo de compilación al modificar el valor de uno de esto parámetros podía incluso a tardar varios minutos. Otro problema que surgió al crecer el proyecto era que estos parámetros estaban repartidos entre varios ficheros, lo que suponla tener que recordar donde estaba localizado cada parámetro y el consecuente riesgo de dejarse alguno sin actualizar que invalidaría las pruebas realizadas.

Lo primero que se realizo fue sacar todos los parámetros configurables a un fichero XML que es leído al inicio del programa permitiendo que todos los parámetros sean accesibles por cualquier módulo dentro del programa.

Las ventajas son apreciables a instante, el proceso de pruebas es mucho más rápido, ya que no es necesario volver a compilar cada vez que se realiza un cambio en la configuración. Todos los parámetros se encuentran en un único fichero, que al ser XML, tiene una estructura clara y legible para las personas, y que permite una edición y modificación sencilla. También permite tener varios ficheros con distintas configuraciones y que se cargue en el programa uno u otro en función de las necesidades del momento.

La necesidad es la de tener una clase centralizada y accesible desde cualquier parte del programa que sea capaz de leer un fichero de configuración en XML y almacene aquellas variables parametrizables de

Para implementar el gestor de configuración se ha creado la clase `ConfigManager`. Esta tiene la función `loadConfiguration` a la que se le indica el fichero XML con la configuración a cargar

Se ha optado por seguir el principio de diseño KISS que establece que la mayoría de sistemas funcionan mejor si se mantienen simplesen lugar de hacerlos complejos; por ello, la simplicidad debe ser mantenida como un objetivo clave del diseño, y cualquier complejidad innecesaria debe ser evitada. Las ventajas genéricas que ofrece un patrón *Singleton* frente a la implementación de un clase estática como puede ser que las funciones se puedan sobrescribir en las clases herederas, que la inicialización asíncrona mientras que una clase estática generalmente se inicializa cuando se carga por primera vez o que el *Singleton* pueda ser manejado polimórficamente sin forzar a los usuarios a asumir que sólo existe una instancia, no suponen un mejora para la funcionalidad que necesitamos implementar.

6.6.2 Funciones de representación para depuración

Aunque la creación de funciones para la representación gráfica están fuera del alcance de este proyecto, a la hora de probar y depurar los distintos algoritmos implementados resulta muy practico disponer de algunas funciones de dibujado ya implementadas para corregir errores y tener una visión preliminar de como puede quedar finalmente.

A partir de las primitivas de dibujado que ofrece OpenCV se han implementado una serie de funciones para facilitar la representación de contornos, esquinas, y una vez obtenidos los

parámetros extrínsecos de las cámara y el proyector, dibujado de ejes, cubos u ortoedros que permiten validar si los cálculos se han realizado correctamente.

6.6.3 Log del sistema

GrayAR proporciona la clase `Log` que es la encargada de mantener el registro de eventos con los sucesos que ocurren en el sistema. Es una clase estática que realiza la escritura en la salida estándar con resaltado en distintos colores en función del tipo de mensaje. La clase además proporciona algunas funciones para imprimir vectores y matrices:

- **plain:** Mensaje sin colores.
- **info:** Mensaje de información. Color azul.
- **error:** Mensaje de error. Color rojo.
- **success:** Mensaje de éxito. Color verde.

Capítulo 7

Resultados

En este capítulo se introducirán los recursos y costes empleados por GrayAR. Se presentarán estadísticas del proyecto y algunas medidas de rendimiento que se han calculado junto a sus resultados.

Para finalizar, se muestra el funcionamiento del sistema desarrollado dentro del *Proyecto ARgos* mediante varios casos de usos.

7.1 Estadísticas del repositorio

La Tabla 7.1 muestra el número de líneas de código de GrayAR. Prácticamente, la totalidad las líneas implementadas pertenecen al lenguaje C++. El lenguaje YAML se ha utilizado para la escritura de los parámetros intrínsecos y las matrices de transformación del proceso de calibrado y XML para la definición de la configuración del sistema. La herramienta utilizada para la contabilización ha sido ***cloc***.

Lenguaje	**Archivos**	**Espacios en blanco**	**Comentarios**	**Líneas de código**
C++	37	1273	1238	3714
Cabeceras C/C++	42	1018	1473	3412
YAML	4	0	0	873
make	2	27	0	70
XML	2	14	18	22
Bourne Shell	1	1	0	2
Total:	**88**	**2333**	**2729**	**8093**

Cuadro 7.1: Número de líneas del código fuente de GrayAR.

7.2 Recursos y costes

El desarrollo del presente Trabajo de Fin de Grado abarcó desde el día 18 de Noviembre de 2013, hasta el 25 de Noviembre de 2014, aproximadamente. No se ha tenido en cuenta el tiempo dedicado a la documentación de la presente memoria, ni los periodos vacacionales, por lo que el tiempo total de desarrollo y pruebas ha supuesto una duración de 43 semanas. Con una dedicación media de 7 horas diarias (5 días a la semana), se traduce en 1505 horas de trabajo.

Se ha supuesto un precio de *30 €/hora* tomando como referencia la media de cobro actual de mercado en este sector [1].

La Tabla 7.2 muestra un desglose de gastos aproximados para el desarrollo de GrayAR. La tabla incluye todo el hardware necesario para la puesta en marcha del sistema, así como el sueldo del programador calculado teniendo en cuenta lo anterior. Se muestra también el coste total del proyecto.

El coste obtenido es meramente informativo, ya que a la hora de calcular un presupuesto detallado se deben de tener en cuenta otros muchos elementos, como impuestos o gastos derivados del puesto de trabajo (luz, acceso a internet,...).

Recurso	**Cantidad**	**Coste**
Sueldo programador (1505 horas)	1	45.150 €
Computador de trabajo	1	499 €
Raspberry Pi (Modelo B)	1	40 €
Cámara Raspberry Pi	1	24,95 €
Proyector Optoma PK320	1	325 €
MicroSDHC Trascend 32GB	1	14,75 €
Altavoz WaveShare LM386	1	10 €
Webcam Logitech HD C525	1	26 €
Total		46.089,7 €

Cuadro 7.2: Desglose de costes de GrayAR.

7.3 *Profiling*

El *profiling* ó análisis de rendimiento, es la observación del comportamiento de un programa utilizando información obtenida desde el análisis dinámico del mismo. Tiene por objetivo averiguar el tiempo dedicado y el consumo de recursos en la ejecución de diferentes partes del software. Se utiliza principalmente para detectar *«cuellos de botella»* del sistema, dónde sea posible llevar a cabo una optimización,

Debido a los limitados recursos y capacidades de los que dispone la Raspberry Pi, durante el desarrollo de GrayAR, se han realizado numerosas mediciones del sistema con el objetivo de tomar decisiones que consiguiesen producir un sistema lo más optimizado posible.

7.3.1 Rendimiento de cámaras utilizadas

La elección de la cámara era un factor muy importante para garantizar la viabilidad del proyecto. El proceso de captura de imágenes debería ser lo más liviano posible, ya que tiene un uso intensivo a lo largo de la ejecución. Además debe tener un rendimiento, capaz de proporcionar una sensación de tiempo real en el sistema.

[1] Información obtenida de `https://freelance.infojobs.net/freelancers/programador/informatica`.

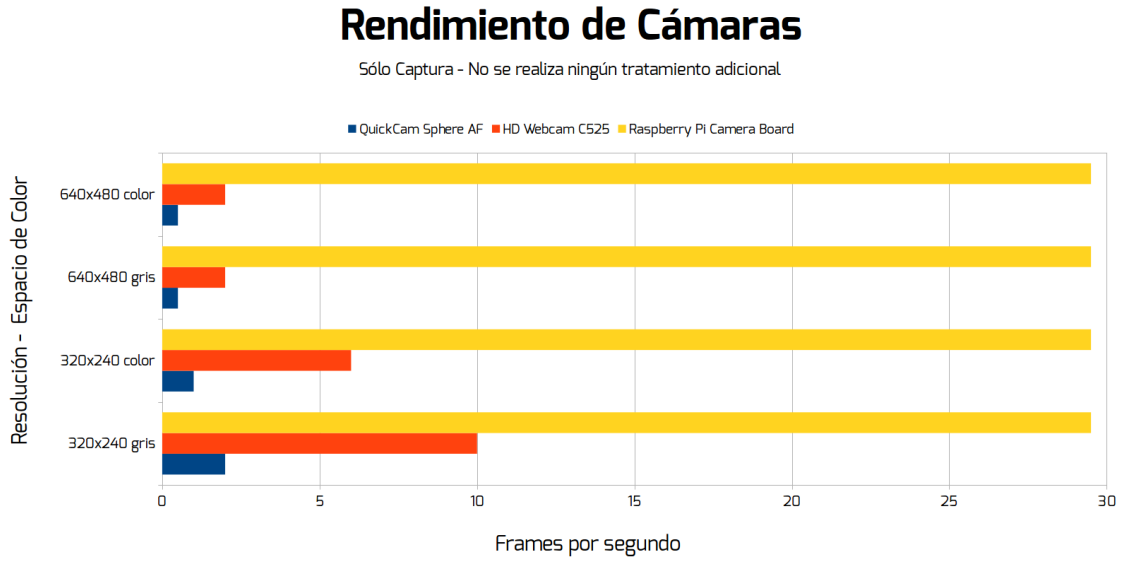


Figura 7.1: Tasa de FPS por modelo de cámara

En principio se utilizaron cámaras USB, pero la utilización de drivers *video4linux* para la captura de las imágenes, no tenia el rendimiento esperado, presentado las imágenes con *lag* de hasta 2 segundos.

La elección final, fue la utilización de la Raspberry Pi Camera Board, que ofrece una tasa de 29,5 FPS hasta una resolución de 1280x960 a escala de grises y 14 FPS con esa misma resolución, pero esta vez en el espacio de color BGR (propio de OpenCV).

En la figura 7.1 se muestran las medidas realizadas, en *frames por segundo* (FPS), del proceso de captura y representación en vivo de los fotogramas capturados. No se realizar ningún tratamiento adicional a la imagen.

7.3.2 Histórico de percepciones

La utilización de métodos totalmente visuales para la detección de la hoja de papel, provoca, que incluso cambios en la iluminación, devuelva valores distintos en el cálculo de la posición, aún cuando la hoja no se haya movido. Este efecto, se aprecia como un temblor en la proyección, y será más acusado cuanto mayor sea el error cometido.

La detección de bordes más finos, ayuda a minimizar este error, pero en última instancia, la inclusión de un histórico de percepciones permite la estabilización del *tracking*, eliminando gran parte de este incómodo efecto, basándose en las percepciones anteriores.

En la figura 7.2 se muestran diferentes valores de la componente de la posición en el eje X en 40 frames de captura. Lo interesante de esta figura no son los valores de estas posiciones (intencionadamente distintos), sino la forma de la trayectoria. Empleando el histórico, la trayectoria resultante es mucho más suave.

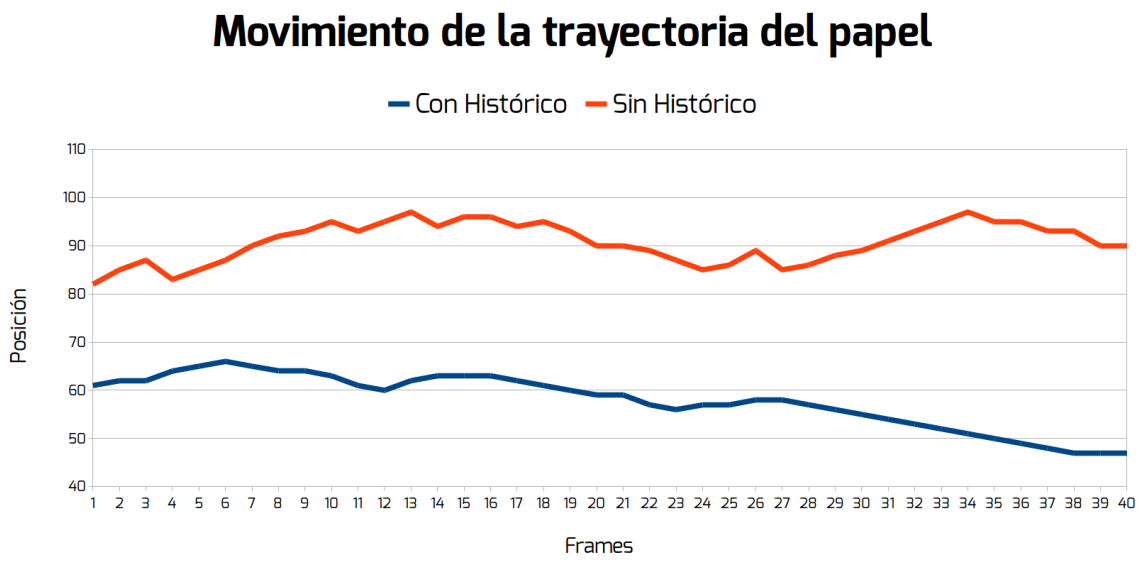


Figura 7.2: Comparativa de la trayectoria (eje X) de una hoja de papel detectada a largo de 40 frames

7.3.3 Rendimiento general del sistema tras cada iteración construida

En cada iteración el software crecía en funcionalidades y complejidad. La Raspberry Pi tenía que soportar, cada vez, una carga superior. Además, se le debe añadir las tareas de *BelfegAR*, que también crecía iteración tras iteración.

En la figura 7.3 se muestra el rendimiento medido del sistema, en FPS, al finalizar cada *sprint*. Se puede observar los malos resultados de la primera iteración al utilizar una cámara web y la recuperación experimentado en el sistema a partir de la iteración 5, que corresponde con la migración de los subsistemas más complejos al servidor.

7.4 Prototipo hardware construido

El prototipo construido emplea una cámara de bajo coste como entrada al módulo de visión por computador y un cañón de proyección portátil para mostrar información visual, directamente alineada sobre el documento del mundo físico. La ejecución del software del sistema la realiza la unidad Raspberry Pi, que se encarga de tomar como entrada las imágenes obtenidas por la cámara y generar la salida para el cañón de proyección. También se ha incluido un amplificador con altavoz incorporado para la reproducción de mensajes y alertas del sistema.

Para obtener un conjunto compacto y reducido, se acopla la Raspberry Pi sobre el proyector y la raspiCam junto a la lente. En la figura 7.4 se muestran por separado los componentes hardware empleados, así como su disposición final en el prototipo.

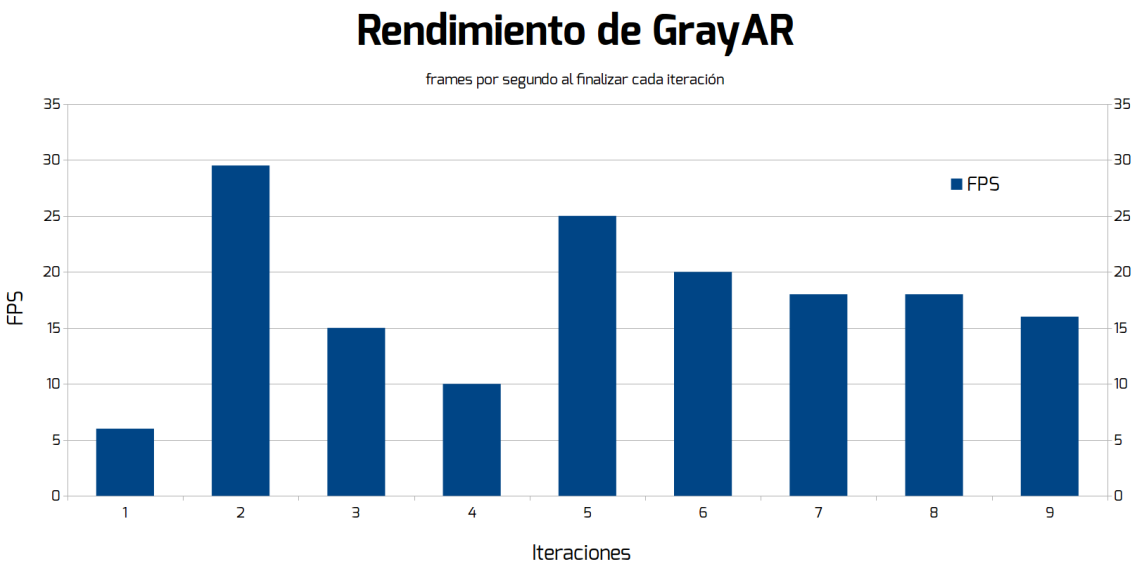


Figura 7.3: Tasa de frames del sistema en cada iteración

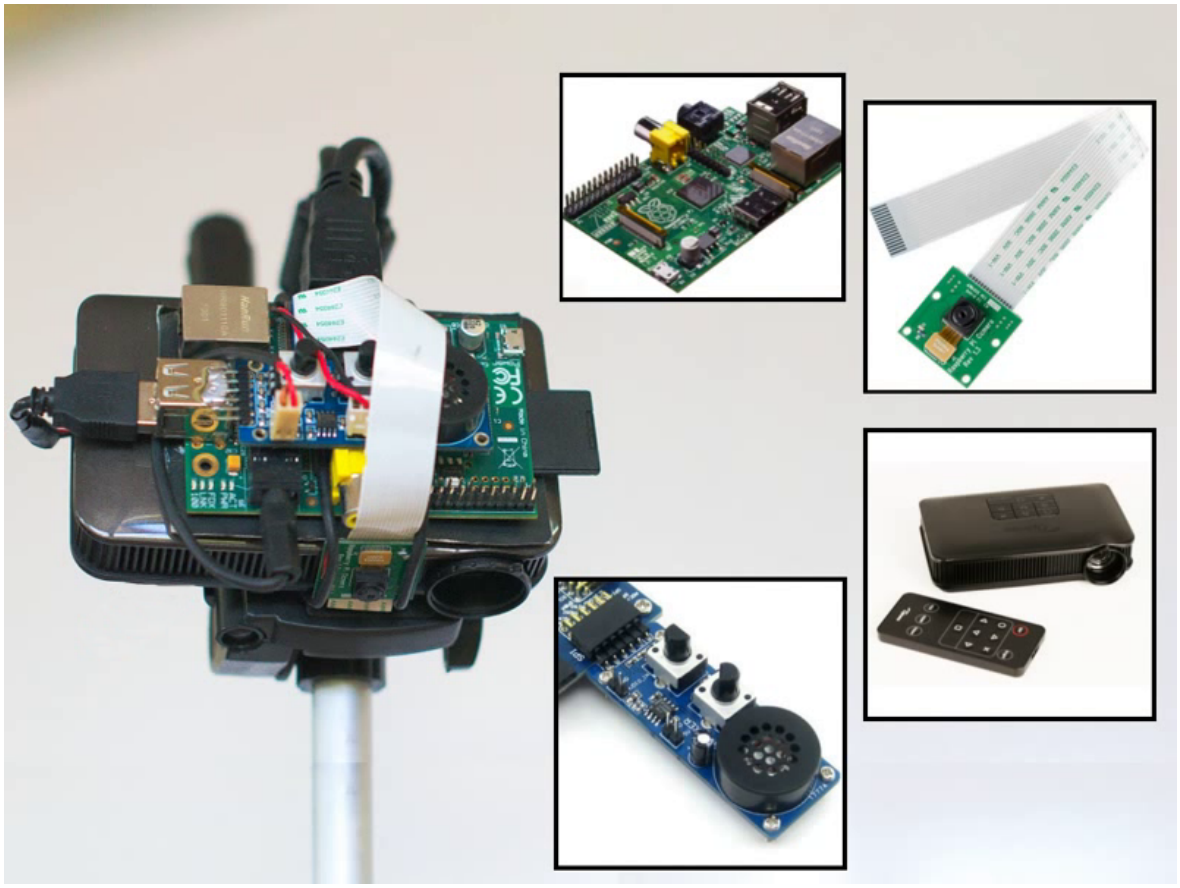


Figura 7.4: Hardware de *ARgos*.

Figura 7.5: Ubicación del prototipo en un escenario real

Finalmente, el sistema se monta sobre un trípode y se dirige en dirección a la mesa. Siendo la colocación final de uso, la que se aprecia en la figura 7.5

7.5 Uso del sistema

A continuación, se introducirá un caso de uso concreto del sistema *ARgos*, mediante imágenes del funcionamiento real del prototipo. Los componentes dibujados son tarea de *Belfegar*, que a partir del procesado de la imagen y los cálculo realizado en GrayAR, *«dibuja»* mediante OpenGL ES, la salida que debe proyectarse, de manera que aparezcan siempre correctamente alineados y sin deformaciones debidos a la perspectiva, independientemente de la colocación del papel en la mesa.

7.5.1 Prerrequisitos

Previamente, el usuario ha calibrado el sistema utilizando la herramienta que se proporciona, y de la que obtiene una serie de ficheros, que deben ser copiados en el cliente y el servidor. En estos ficheros se encuentran los parámetros intrínsecos de la cámara y el proyector, así como las matrices de transformación entre el sistema de coordenadas de la cámara y el del proyector.

Asimismo, el usuario ha proporcionado una imagen de cada uno de los documentos posibles a identificar por el sistema, y las acciones a realizar o representar en ellos. Esta configuración se realiza por medio de los *scripts* implementados en *BelfegAR*.

7.5.2 Inicialización del sistema

Tras el encendido del prototipo, el registro de eventos muestra por la salida especificada los sucesos correspondientes al cliente y el servidor durante esa etapa (véase Figura 7.6 y Figura 7.7, respectivamente).

Para el cliente (y detallando el registro correspondiente a *ARgos*), el registro muestra que la conexión con el servidor se ha realizado correctamente, la inicialización del contexto OpenGL ES y la carga de imágenes que se usarán más tarde por el sistema. Para el caso del servidor, el registro muestra la carga de los ficheros de calibrado y se generan los descriptores de imágenes de los documentos conocidos por el sistema. Finalmente, carga en memoria los *scripts* de acciones y muestra la IP y el puerto donde está escuchando conexiones de clientes.

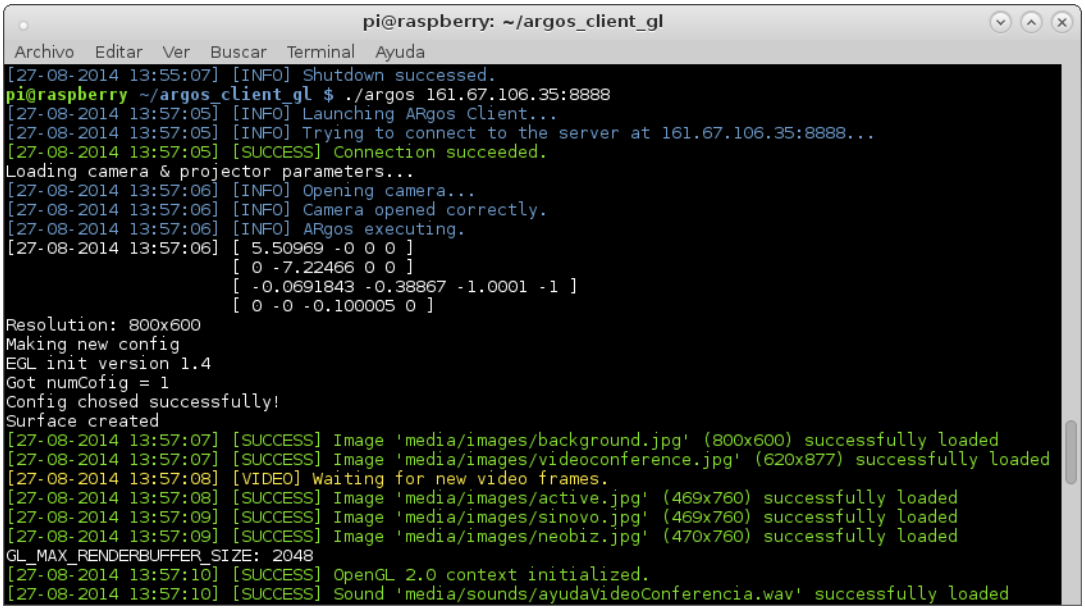


Figura 7.6: Inicialización del cliente.

Una vez iniciado el sistema, el proyector muestra una imagen con información sobre un supuesto usuario ficticio (véase Figura 7.8).

7.5.3 Bucle principal

Al finalizar la introducción, el sistema avanza hacia el próximo estado, entrando en el bucle principal de detección e identificación de documentos, delegación de tareas y renderizado. Este estado es fácilmente reconocible por proyectar las cuatro esquinas que delimitan el área de trabajo sobre la superficie (véase Figura 7.9).

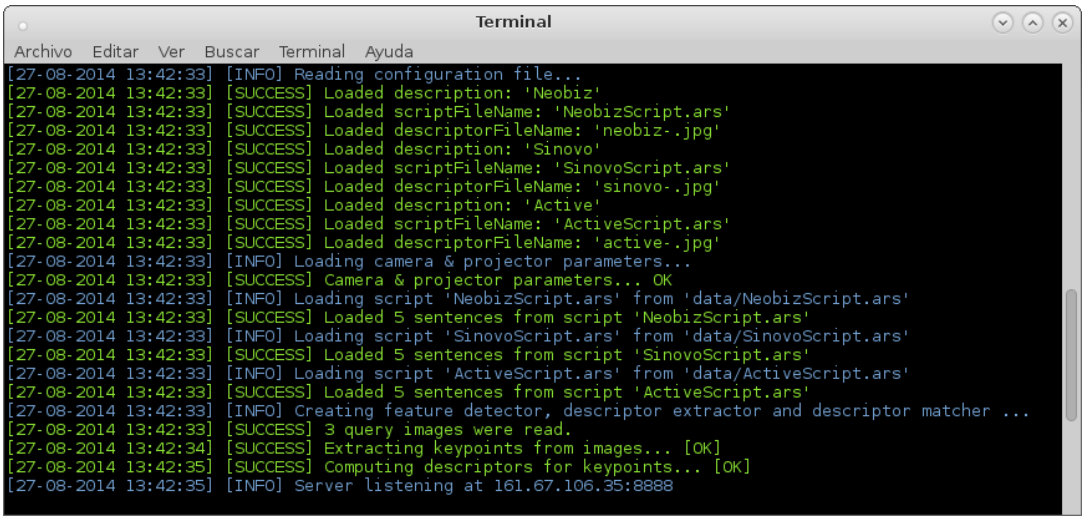


Figura 7.7: Inicialización del servidor.

A partir de este momento, el usuario puede empezar a colocar los documentos sobre la superficie de trabajo. Tras situar el primer documento, el sistema detecta la hoja de papel, calcula su posición en el espacio 3D e identifica el documento por su contenido. Con esta información, *BelfegAR* renderiza varios componentes gráficos, como zonas de resaltado, correspondientes a ese documento. Para este caso concreto, se presentan varios botones que el usuario puede pulsar e iniciar un vídeo que traduce a lengua de signos el documento. (véase Figura 7.10).

Después, el usuario coloca un segundo documento. Para este caso, el sistema renderiza otros componentes gráficos basados en el contexto del formulario detectado (véase Figura 7.11). Además, se presentan varios botones que el usuario puede pulsar para obtener información adicional de forma ampliada al darle la vuelta al documento, y utilizar la parte trasera como «pantalla de proyección». (véase Figura 7.12).

Entre las acciones a realizar en el tercer documento, está definida la posibilidad de realizar una videollamada. Tras la pulsación del botón correspondiente y volteando nuevamente el documento (véase Figura 7.13).

La Figura 7.14 muestra como se han empezado a recibir fotogramas del servidor y son renderizados sobre el propio documento junto a una imagen de fondo.

Durante toda la ejecución del sistema se realiza un registro de los eventos ocurridos. La Figura 7.15 y la Figura 7.16 muestran respectivamente los registros de eventos del cliente y del servidor mientras se ejecuta el bucle principal del sistema. En ellos, se puede apreciar las matrices de transformación de cada documento detectado o la comunicación realizada entre cliente y servidor.

Figura 7.8: Arranque de *ARgos*.

Figura 7.9: *ARgos* esperando documentos.

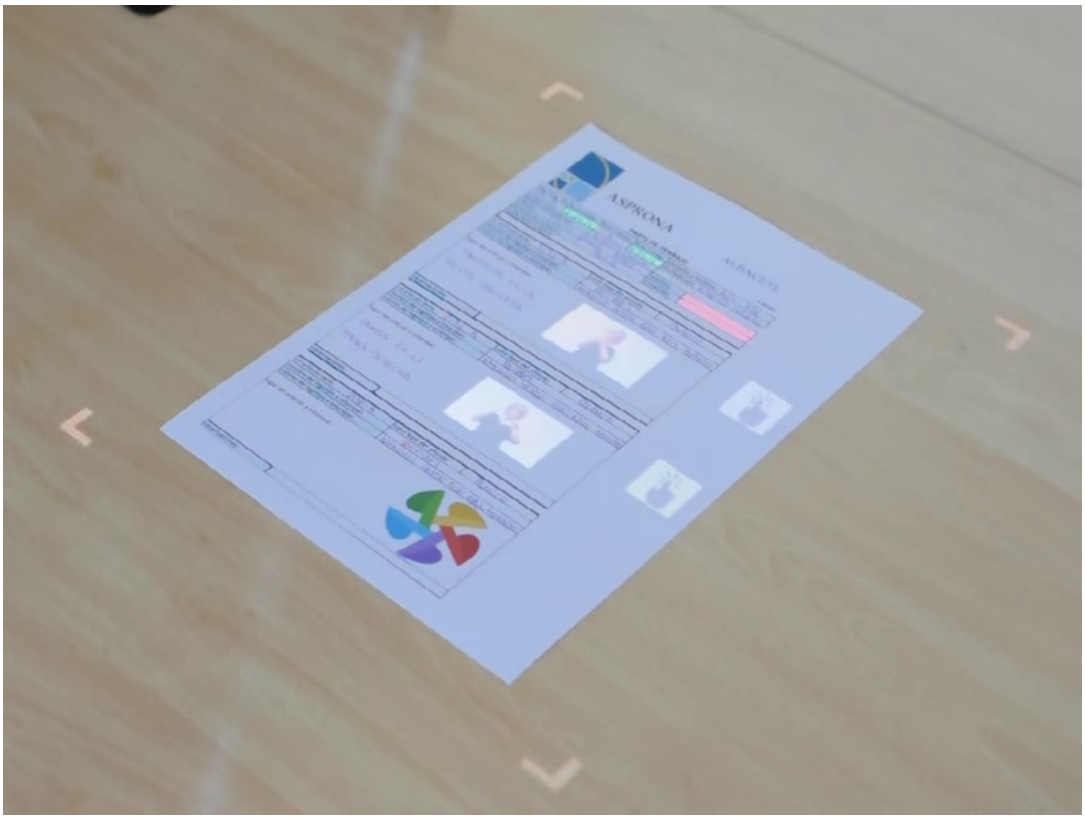


Figura 7.10: Primer documento detectado y componentes gráficos desplegados.

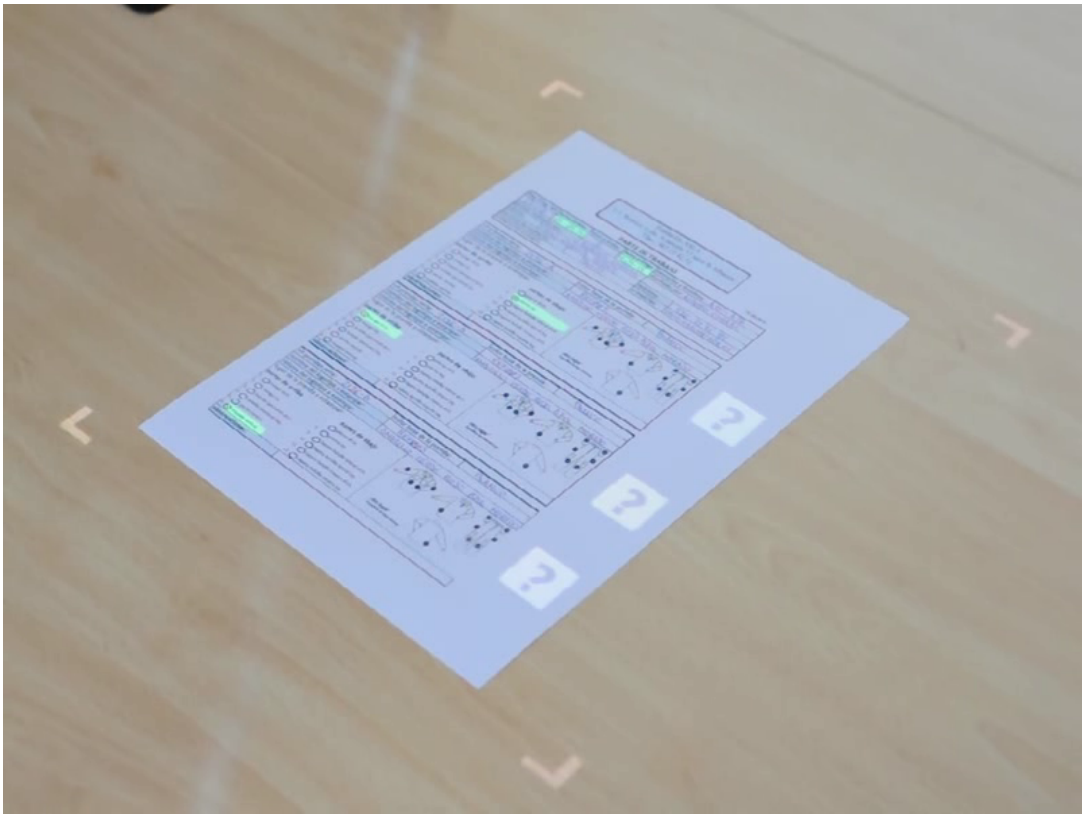


Figura 7.11: Segundo documento detectado y componentes gráficos desplegados.

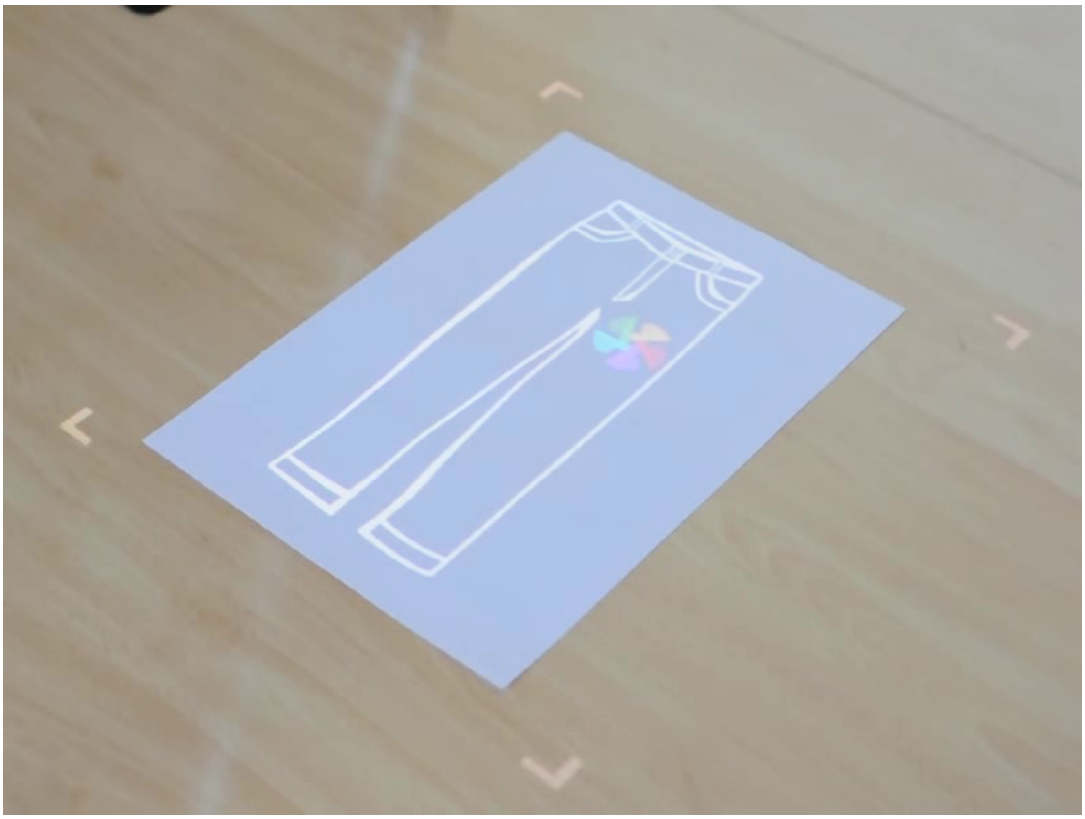


Figura 7.12: Información mostrada al pulsar un botón de interacción.

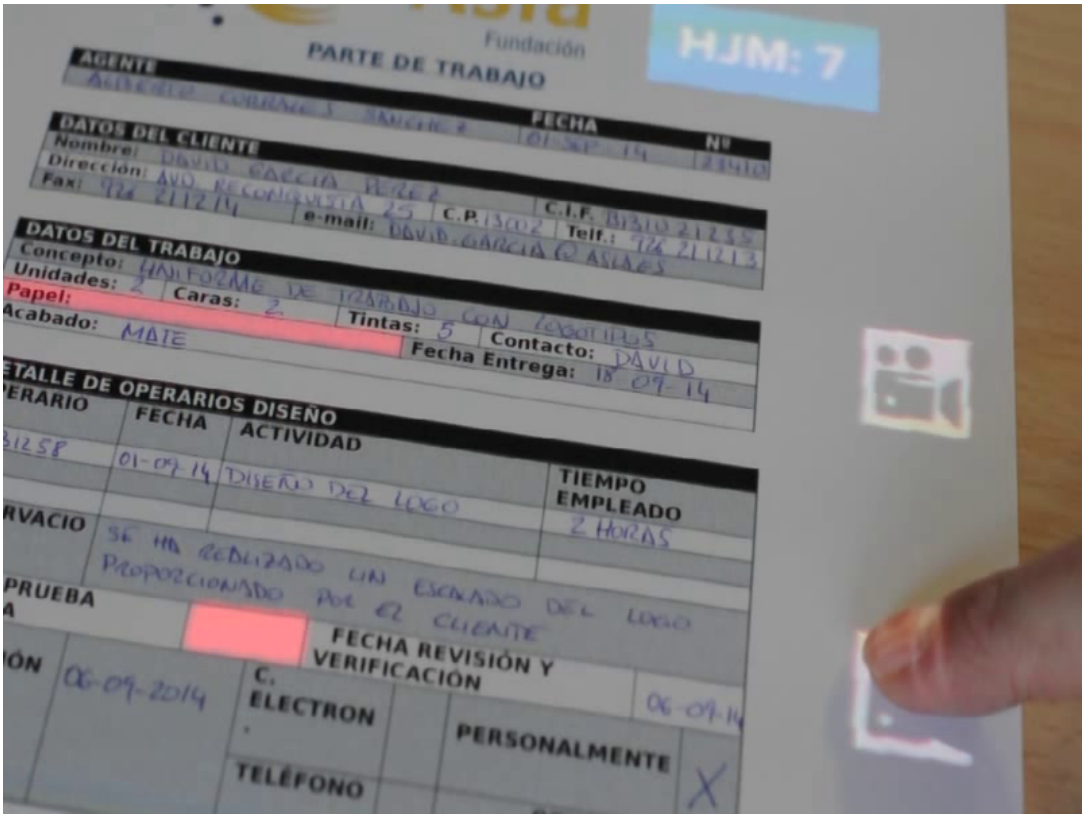


Figura 7.13: Tercer documento detectado e interacción con los botones virtuales.

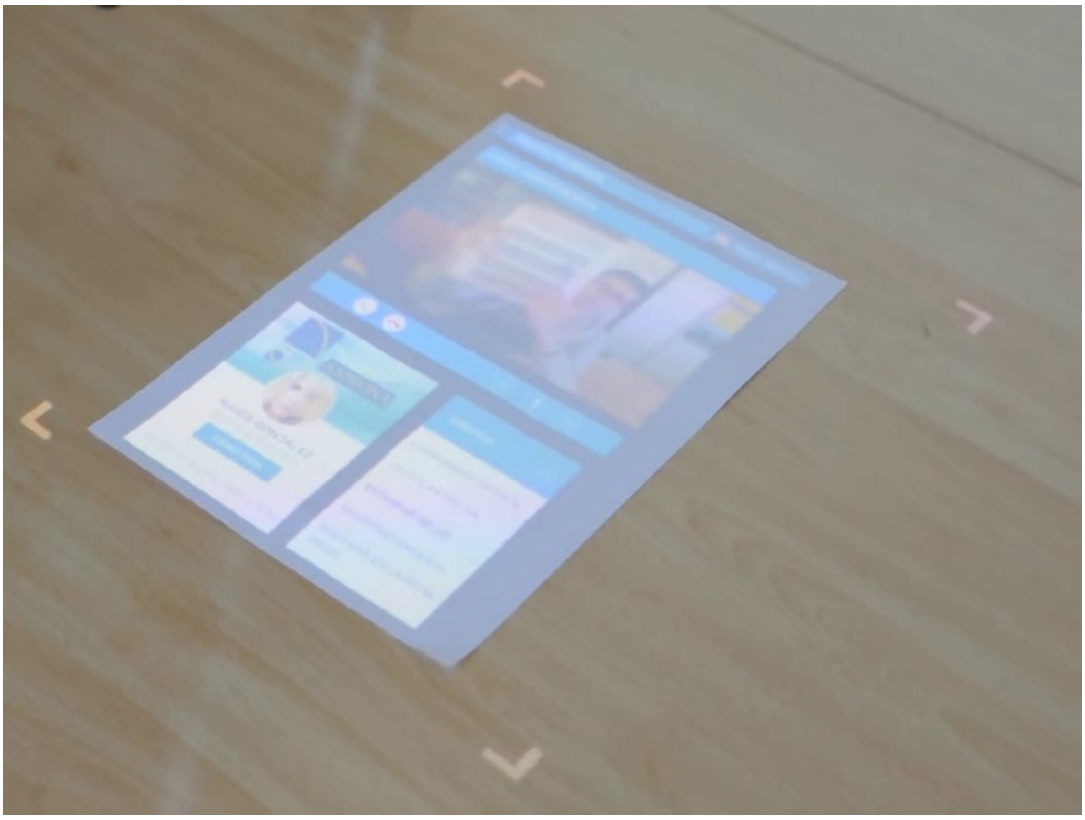


Figura 7.14: Videollamada posicionada y alineada sobre la superficie del documento.

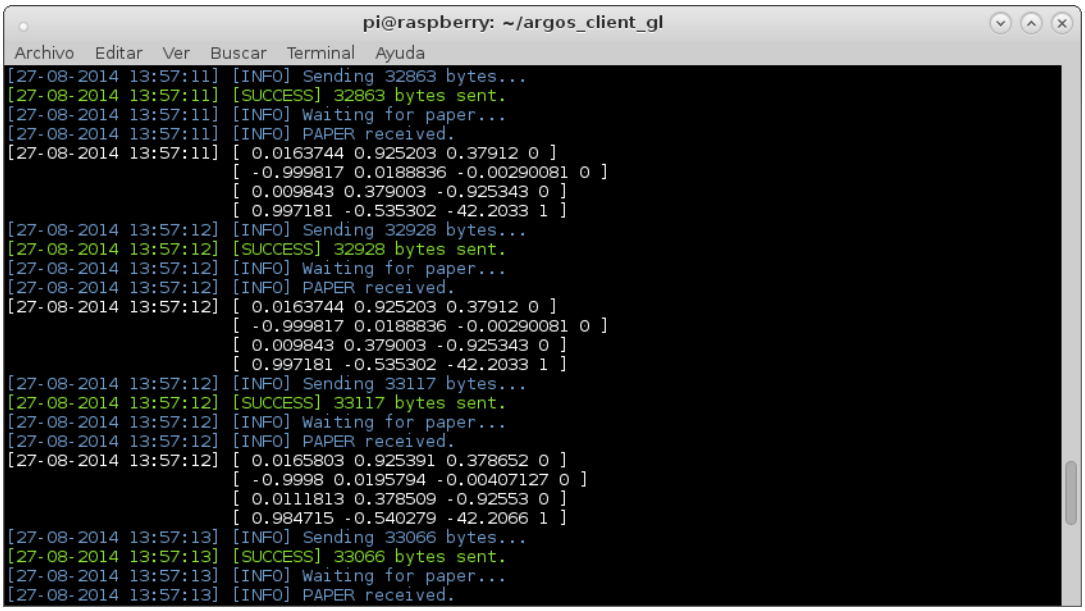


Figura 7.15: Registro de eventos del cliente mientras se ejecuta el bucle principal del sistema.

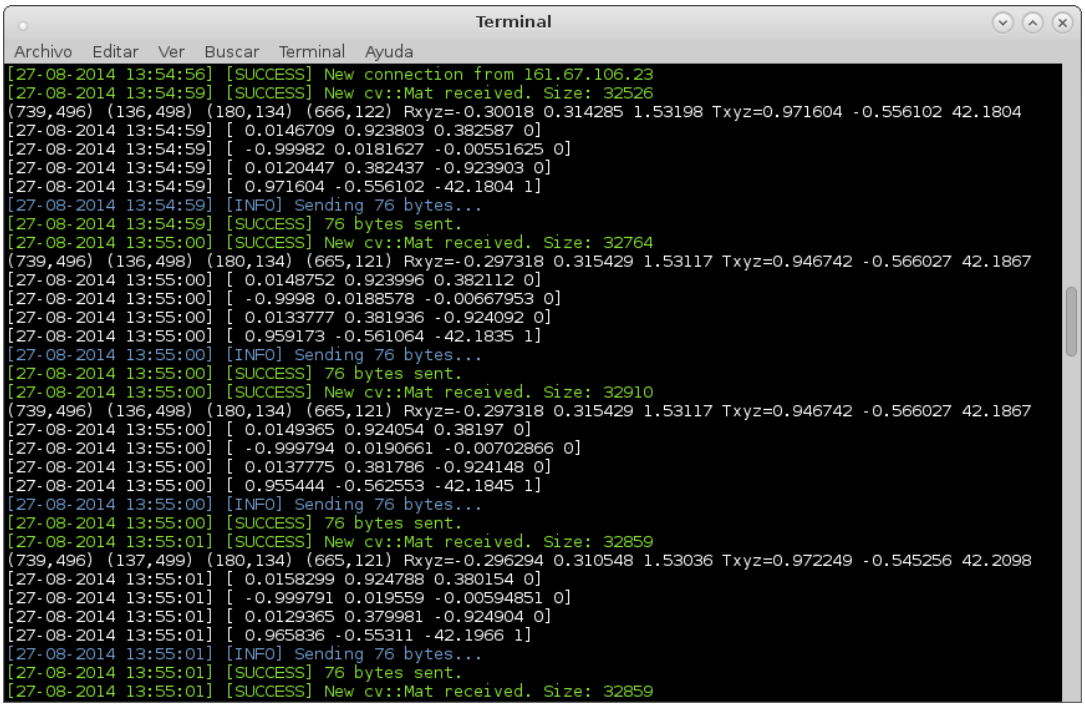


Figura 7.16: Registro de eventos del servidor mientras se ejecuta el bucle principal del sistema.

Capítulo 8

Conclusiones y propuestas

En este capítulo se realiza un análisis sobre los objetivos alcanzados durante el desarrollo del presente Trabajo Fin de Grado, aportando en cada caso, las ventajas que ha supuesto la utilización o la elección de una determinada implementación respecto a otras opciones disponibles. A continuación, se exponen nuevos puntos de vista, que se pueden tratar en futuros trabajos para mejora ó ampliación del sistema, indicando una posible implementación y una estimación del coste temporal en caso de realizarlos.

8.1 Objetivos alcanzados

El objetivo principal de GrayAR es el desarrollo de los sistemas de captura, tracking, registro e identificación de documentos dentro del *Proyecto ARgos*. Con tal fin, se ha construido un prototipo real basado en una Raspberry Pi, con una cámara de bajo coste y un proyector portátil, que montado sobre un trípode, muestra información visual alineada sobre un documento que se encuentre dentro de la zona de proyección. Para efectuar esta visualización, se tiene en cuenta el posicionamiento 3D relativo entre el documento y el sistema cámara-proyector, para que el registro de la amplificación visual sea perfecto.

Además, el sistema responde a distintas acciones que el usuario pueda efectuar sobre el espacio físico, ampliando la información relacionada que sea relevante a la acción realizada.

Por tanto, el objetivo principal queda satisfecho, como así ocurre también con aquellos objetivos específicos, definidos en el capítulo 2 y detallada su solución entre los capítulos 5 y 6, que exponemos a continuación a modo de resumen.

Captura y preprocesado de imágenes. El sistema cuenta con un módulo capaz de obtener imágenes de los distintos tipos de cámaras que se le pueden acoplar, ofreciendo una interfaz común, a pesar de disponer de drivers y APIs diferentes. Se ha provisto de los filtros necesarios para el tratamiento de las imágenes, contando con funciones de escalado, umbralización, detección de bordes y detección de características.

Para el cálculo de los parámetros intrínsecos y extrínsecos tanto de la cámara como del proyector, se ha desarrollado una aplicación externa que permite un calibrado rápido y sencillo del sistema, mediante un patrón de tipo tablero de ajedrez, cada vez que sea

necesario. Una vez realizada la calibración, obtenemos tres ficheros XML que son lo necesarios para poder realizar los cálculos de registro en el prototipo.

Sistema de identificación de documentos. La identificación de documentos en GrayAR se ha realizado mediante un algoritmo de recuperación de imágenes basado en descriptores y comparará el documento que está siendo analizado con una base de datos de documentos conocidos por el sistema.

Estos documentos se proporcionan en el fichero de configuración, así como las acciones posibles que se pueden realizar sobre él. Este tipo de configuración permite un sistema adaptable y personalizable a cada organización y usuario.

Implementación de técnicas de tracking y registro. Para conseguir una experiencia totalmente inmersiva, se han desarrollado técnicas de tracking y registro que permiten el cálculo de la *pose* (rotación y translación del documento en el espacio 3D) en tiempo real. Esto nos permite, que podamos ofrecer al usuario la información proyectada sobre la hoja de papel, independiente de su posición, rotación o plano en el que se encuentre. Siempre y cuando, no se salga de los límites de la zona de proyección.

Utilización de paradigmas de interacción natural con el usuario (NUI). El usuario puede interactuar directamente en el espacio físico sin utilizar sistemas de mando tradicionales, como ratones o teclados. Sólo es necesario encender el prototipo, y según el usuario vaya colocando documentos, los desplace por la zona de proyección, realice interacciones señalándolo ó girándolo para mostrar la cara posterior, el sistema responderá automáticamente a la acción realizada.

La respuesta se le ofrece al usuario amplificada de varias formas (visual y/o auditiva) y siempre adaptada a la escena que se esté produciendo en ese mismo instante.

Facilitar la gestión documental a personas con necesidades especiales. GrayAR proporciona toda la información necesaria al sistema de representación *(BelfegAR)*, para que realice el dibujado de la información visual relevante al contexto directamente sobre el espacio del papel.

Para complementar a la representación visual, se ha añadido a la Raspberry Pi un amplificador de audio LM386 montado sobre una placa de test con un altavoz de 1W, que hace las funciones de «tarjeta de sonido», permitiendo la reproducción de avisos, alertas o lecturas de texto para aquellas personas que necesiten de amplificación auditiva para la utilización del sistema.

Se debe basar en componentes de bajo coste. Para la construcción del prototipo se han seleccionado aquellos componentes, que dentro de unas prestaciones mínimas requeridas, fuesen lo más económicos posibles. El coste final (ver Sección 4.4) es muy reducido respecto a otros sistemas que puedan realizan funciones parecidas. El proyector,

la parte más cara del sistema con un valor de 350€, lo podemos considerar económico, teniendo en cuenta que el precio medio de otros modelos es de 1000€.

Respecto al software empleado, todos los recursos y bibliotecas utilizadas son de licencia libre, y por tanto no están sujetos a pagos de licencias.

Dispositivo multiplataforma (hardware y software) El desarrollo de GrayAR se ha realizado exclusivamente en C++, siguiendo el estándar C++11. Los fichero de configuración y de calibrado están en XML. El uso de bibliotecas externas ha sido mínimo, eligiendo las que no poseían dependencias de terceros, y siempre con licencias libres multiplataforma. Estas decisiones garantizan la portabilidad del sistema, tanto hardware como software (sistemas operativos), en forma de paquete único.

8.2 Propuestas de trabajo futuro

El *Proyecto ARgos* no ha finalizado aún. GrayAR comprende las funciones de realidad aumentada y visión por computador que se han construido hasta el día de hoy en el *Proyecto ARgos*. Algunas de la propuestas aquí reflejadas serán desarrolladas en la versión final de *ARgos* y servirán de contexto para la realización de la tesis del Máster en Tecnologías Informáticas Avanzadas prevista para en este curso académico.

Debido a la construcción modular y desacoplada del sistema, las ampliaciones o mejoras aquí planteadas son de ámbito local y solo afectarán al subsistema o módulo referido, siendo transparente los cambios realizados para el resto de la aplicación.

Tracking mediante aproximaciones *top-down*. Estas técnicas se basan en la estimación mediante modelos de movimiento basados en filtros bayesianos para predecir la posición de la cámara. A partir de la posición obtenida, se buscan referencias en la imagen que puedan corregir y ajustar la predicción.

Los filtros bayesianos a utilizar pueden clasificarse en dos tipos. Aquellos que trabajan con modelos de movimiento gausianos, se denominan Filtros de Kalman, y los que, por las características del ruido no pueden ser modelados mediante modelos gausianos, y se implementan mejor mediante Filtros de Partículas.

Estos métodos proporcionan robustez al proceso de tracking, ya que permite seguir detectando la hoja de papel, aun cuando exista una gran oclusión de la misma.

Para la implementación de esta mejora, habría que valorar, además, el tiempo para analizar ambos enfoques y determinar cual de ellos se adapta mejor a las características de su desplazamiento por la zona de proyección.

Detección de páginas de texto mediante LLAH o similares. La implementación actual de detección de documentos en GrayAR está basado en la búsqueda de coincidencias de características locales, ya que estos métodos obtienen mayor precisión de acierto en

la detección de documentos con información semiestructurada, como es el caso de facturas o formularios.

Sería interesante, como ampliación al sistema de detección, que se pudiesen reconocer documentos con alto contenido textual sin una estructura predeterminada de antemano. Para ello, es necesario la implementación de algún algoritmo de detección basado en las características inherentes de la estructura del documento y el texto que contiene.

LLAH [NKI09] tiene una alta escalabilidad y el esquema de indexación y recuperación empleado es extremadamente rápido. Los autores han confirmado que LLAH tiene una tasa de acierto del 99 % y con un tiempo de procesamiento de 50 ms en una base de datos de 20 millones de páginas.

Existen varias implementaciones optimizadas sobre LLAH para salvar las limitaciones iniciales del algoritmo. Sería un trabajo futuro la tarea de buscar sólo las publicaciones que se hayan hecho sobre LLAH e intentar crear una implementación conjunta con todas las optimizaciones.

Calibrado del sistema cámara-proyector mediante luz estructurada. Otra técnica para el calibrado de proyectores es el propuesto por Daniel Moreno y Gabriel Taubin, basado luz estructurada, en el paper ***«Simple, Accurate, and Robust Projector-Camera Calibration»***.

La implementación, consistiría en los siguientes pasos a realizar [MT12]:

- Detectar la localización de las esquinas de un patrón de tablero de ajedrez en cada una las orientaciones capturados.
- Estimar los componentes de luz directa y global.
- Decodificar los patrones de luz estructurada.
- Calcula una homografía local para cada una de las esquinas del patrón detectado.
- Trasladar las posiciones de las esquinas a coordenadas del proyector mediante homografías locales.
- Obtener los parámetros intrínsecos de la cámara utilizando las posiciones de las esquinas en la imagen.
- Obtener los parámetros intrínsecos del proyector con las posiciones trasladadas al sistema de referencia del proyector.
- Ajustar los parámetros intrínseco de la cámara y el proyector y obtener los parámetros extrínsecos del sistema en conjunto.
- Opcionalmente, se puede realizar una optimización conjunta de todos los parámetros (intrínsecos y extrínsecos).

Este método no requiere ningún equipamiento especial y, según sus autores, es más preciso que otras técnicas de calibrado, ya que utilizan el modelo pinhole completo con distorsión radial.

En base a la experiencia en el desarrollo del proceso de calibrado en GrayAR, esta historia se podría plantear para realizarse en un sprint de 3 ó 4 semanas de duración.

Utilización de cámaras con sensor de profundidad. Sustituyendo la cámara principal, por otra equipada con sensores de infrarrojos (IR) capaces de medir la profundidad de la escena, obtendríamos una mayor precisión y la posibilidad de aumentar la variedad de gestos reconocidos.

Por ejemplo, realizar la acción de click (o doble click) con los dedos sobre la superficie de la mesa, ó la simulación de una superficie multitáctil.

El coste temporal de esta propuesta puede ser mayor que las anteriores. Habría que estudiar las bibliotecas existentes para la utilización del dispositivo (por ejemplo, kinect), proporcionar un nuevo sistema de calibrado para la cámara IR, y también, seria necesaria la modificar los métodos de reconocimiento gestual, que tendrían un enfoque distinto, debido a la información adicional de la cámara de profundidad.

Capítulo 9

Conclusions and proposals

In this chapter, we analyse the objectives that we have reached in the development of this end-of-degree project, providing in each case, the advantages that the use and choice of a specific implementation regarding to other available options have made. The following are different point of views which can be studied in future projects to improve or expand the system, indicating a possible implementation and an estimation of the temporary cost in case of developing the project.

9.1 Achieved objectives

The main objective of GrayAR is the development of capture, tracking and registration systems and the identification of documents that belong to the *ARgos Project*. For this reason, a real prototype based on a Raspberry Pi board with ARM architecture, a low-cost camera and a portable projector has been built. It shows, mounting on a tripod, visual information aligned on a document which is in the projection area. To execute this visualisation, the main part is the relative 3D position between the document and the camera-projector system, in order to achieve a perfect visual amplification.

Moreover, the system responds to different actions that the user can do in the physical space, expanding information related to the desired action.

Therefore, the main purpose is satisfied, as it occurs with the specific objectives, defined in chapter 3 and detailed their solution in chapters 5 and 6. These are explained below as a synopsis.

Image capture and processing. The system is provided with a module that obtains different kind of attachable cameras, offering a common interface, despite of having different drivers and APIs. It has been provided with the required filters to process images, with scaling functions, thresholding, edge detection or characteristics detection. An external application has been developed to calculate the extrinsic and intrinsic parameters of both, camera and projector. This application allows a rapid and easy calibration of the system, using a checkerboard pattern, whenever it is needed. Once the calibration is done, we obtain three XML files, which are necessary to do the calculation of the registration in the prototype.

System of document identification. The identification of documents in GrayAR has been made by an image-retrieval algorithm based on descriptors and it will compare the document which is being analyzing with a document database, known for the system.

These documents are located in the configuration file, as well as the possible actions that can be made on it. This kind of configuration allows an adaptable and personalised system in each organisation and user.

Implementation of tracking and registration techniques. The tracking and registration module will have real-time pose calculation features (rotation and translation of the object in the 3D space) to reach a fully immersive experience. This can lead to offer the user the information projected on a sheet of paper, without a dependence of the position, rotation or plane in which is located, as long as it does not exceed the limits of the projection area.

Use of Natural User Interface (NUI) paradigms. The user will directly interact with the physical space without using control systems or traditional input devices such as mouse, keyboard, etc. The only need is turning the prototype on and the system will automatically respond to the action that is done when the user places documents, moves them around the projection area or points or rotates them to show their back side.

The response is showed to the user amplified in different ways (visual and/or auditive), and it is always adapted to the scene that is happening at that moment.

Facilitation of document management for people with special needs. GrayAR will give all the required information to the representation system *(BelfegAR)*, in order to draw the relevant visual information right on the paper area.

A LM386 audio amplification has been added to the Raspberry Pi to complement the visual representation. It has been built on a test board with a 1W loudspeaker, which has already «sound card» functions. It allows the reproduction of warnings, alarms or reading of texts to those people who need the hearing amplification for the use of the system.

It has to be based on low-cost components. The selected components of the prototype structure have been chosen with a minimum price, but with a minimum required qualities as well. The final cost is very low compared to other systems that can execute similar functions.

The projector, which is the most expensive part of the system, costs 325€, and can beconsiderate an economic price, taking into account that the average price of other models is 1000€.

Regarding to the used software, every resource and library that have been used are freely licensed, and therefore, they are not subject to licence payment.

Multi-platform device (hardware and software). The development of GrayAR has been exclusively done in C++, following the C++11 standard. The configuration and calibration files are done in XML. The use of external libraries has been minimised, choosing the ones that did not depend on third parties, and always having a multi-platform free licence. These decisions guarantee the system portability, as much hardware as software (operating systems), in a unique pack and integrated form.

9.2 Proposals for a future project

The *ARgos Project* has not finished yet. GrayAR comprises the augmented reality and computer-based vision functions that have been developed until now in Argos project. Some of the proposals here reflected have been developed in the final version of Argos and will be the context to develop the thesis in the Advanced Information Technologies Masters, planned for the next academic year.

Due to the modular and disconnected construction of the system, the extensions and improvements that are reflected here are local and they will only concern to the subsystem or the referred module, with transparent changes for the rest of the application.

Tracking with top-down approximation. These techniques are based on the approximation of movement models based on Bayesian filters to predict the position of the camera. From this position, references are searched on the image to correct and adjust the prediction.

The Bayesian filters that have been used can be classified into two types. One of them works with Gaussian movement models and are called Kalman Filters, the others cannot be model with Gaussian models because of the noise/interference characteristics and are implemented better with Particle Filters.

These methods provide robustness/strength to the tracking process, considering that they let the sheet of paper be still detectable, even when there is abig occlusion of this. The time to analyse both focuses and determine which one is better for the characteristics of its movement around the projection area should be valued to implement this improvement.

Document image recognition using LLAH or similar. The current implementation of documents detection in GrayAR is based on the search for the coincidences in the local characteristics, since these methods are precisely correct on the detection of documents with a semi-structured information, such as receipts and forms.

It would be interesting, as an extension of the detection system, to be able to recognise documents with a high textual content without a predetermined structure as a first sight. For this, it is necessary to implementate a detection algorithm based on the inherent characteristics of the structure of the document and the text that this one has.

LLAH [NKI09] has a high scalability and its indexing and recovery/reclamation scheme is extremely fast. The authors have confirmed that LLAH has achieved a hit rate of 99 percent and it has a processing time of 50 ms more in a database with more than 20 million of pages.

Several optimized implementations about LLAH exist to overcome the initial limitations in the algorithm. The task of looking for the exclusive publications about LLAH could be a future project, and it would be interesting to create a coordinated implementation with all the optimizations.

Calibration of the camera-projector system using structured light. Another technique for the calibration of projectors is the technique proposed by Daniel Moreno and Gabriel Taubin, based on structured light, in the paper called ***«Simple, Accurate, and Robust Projector-Camera Calibration»***.

The implementation would consist of the following steps to perform [MT12]:

- Detect checkerboard corner locations for each plane orientation.
- Estimate global and direct light components.
- Decode structured-light patterns.
- Compute a local homography for each checkerboard corner.
- Translate corner locations into projector coordinates using local homographies.
- Calibrate camera intrinsics using image corner locations.
- Calibrate projector intrinsics using projector corner locations.
- Fix projector and camera intrinsics and calibrate system extrinsic parameters.
- Optionally, all the parameters, intrinsic and extrinsic, can be optimized together.

This method does not require any special equipment and, according to their authors, is more precise than other calibration techniques, considering that they use the complete pinhole model with radial distortion.

On the basis of the experience in the development of the calibration process in GrayAR, this issue could be carried out in a three-or-four-weeks sprint.

Use of cameras with depth sensor. Replacing the main camera with other equipped one with sensors IR capable of measure the depth of the scene, we would obtain a higher precision and the possibility of enlarge the variety of recognized gestures. For example, the execution of the click (or double click) action with the fingers over the table surface or the simulation of a multi-touch surface.

The temporary cost of this proposal could be higher than the previous ones, since we should study the existing libraries for the execution of the device (e.g. kinect) and

provide a new calibration system for the IR camera. The modification of the gesture recognition methods would be necessary as well, and they would have a different perspective due to the incorporation of additional information of the depth camera.

ANEXOS

Anexo A

Características Raspberry Pi

SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)
CPU:	ARM 1176JZF-S a 700 MHz (familia ARM11)
Juego de instrucciones:	RISC de 32 bits
GPU:	Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1, (con licencia), 1080p30 H.264/MPEG-4 AVC
Memoria (SDRAM):	512 MiB (compartidos con la GPU)
Puertos USB 2.0:	2 (vía hub USB integrado)
Entradas de vídeo:	Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF
Salidas de vídeo:	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD
Salidas de audio:	Conector de 3.5 mm, HDMI
Almacenamiento integrado:	SD / MMC / ranura para SDIO
Conectividad de red:	10/100 Ethernet (RJ-45) via hub USB
Periféricos de bajo nivel:	8 x GPIO, SPI, I2C, UART
Reloj en tiempo real:	Ninguno
Consumo energético:	700 mA, (3.5 W)
Fuente de alimentación:	5 V vía Micro USB o GPIO header
Dimensiones:	85.60mm x 53.98mm (3.370 x 2.125 inch)
Sistemas operativos soportados:	GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux. RISC OS

Cuadro A.1: Características del modelo (B) de Raspberry Pi empleado en ARgos.

Anexo B

Manual de calibrado

El proceso de calibrado del sistema se ha creado como una utilidad independiente del sistema principal de *ARgos*. Una vez calibrado, la aplicación proporciona los ficheros YAML, con los parámetros intrínsecos y extrínsecos, que necesita *ARgos* para el calculo del registro.

El proceso está separado en una aplicación cliente *(calibrationToolBox_client)* que está alojada en la Raspberry Pi, y otra aplicación servidor *(calibrationToolBox_server)* que se podría ejecutar en cualquier otro equipo.

En un principio se diseñó como un único programa que ejecutaba en la Raspberry Pi, pero si se tomaban muchas capturas para conseguir una calibración más precisa, el sistema se ralentizaba, y el tiempo de calibrado crecía excesivamente.

Se ha empleado esencialmente el enfoque de Zhang [ZFN02] para calibrado de cámaras. Se utiliza un patrón tipo tablero de ajedrez, en la que se alternan cuadrados blancos y negros, de dimensiones conocidas. El patrón se imprime y se pega sobre una superficie plana rígida. Se debe dejar una zona despejada a continuación del tablero de ajedrez, ya que en esta zona se proyectará el patrón que utiliza el proyector para su calibrado. Se debe prestar atención en que orientación es pegado, ya que si se colocase al revés, la proyección dinámica se realizaría fuera del panel.

Antes de iniciar el proceso de calibrado, debemos definir los parámetros de configuración;

- Tomamos la medida del lado de un cuadrado para proporcionarla al programa de calibrado, de esta forma las unidades de medida del sistema de referencia se corresponden a medidas reales.

- Además del tamaño del lado debemos indicar el número de esquinas en vertical y horizontal que tiene el patrón que estamos utilizando. Gracias a esto, no estamos atados a realizar el calibrado con un patrón particular, que tenga unas medidas concretas. Basta indicar las características del patrón que vamos a utilizar para realizar un calibrado correcto.

- Los otros dos parámetros de la configuración, corresponden al desplazamiento (en los ejes X e Y) de la proyección del patrón de círculos asimétrico. El objetivo es ajustar la visualización de la proyección para que el patrón proyectado aparezca junto al patrón impreso.
- Los últimos parámetros son las dimensiones de las imágenes, tanto de la cámara como del proyector. Se pueden utilizar resoluciones distintas, pero en GrayAR, se he optado por utilizar la misma en ambos dispositivos.

A continuación, colocamos la superficie bajo la cámara, en distintas orientaciones y distancias, para obtener una serie de imágenes en los que se encuentre visible el patrón.

Cuando el proceso disponga de al menos 20 imágenes válidas, realizará los cálculos para obtener los parámetros intrínsecos de cámara y los grabará en el fichero **calibrationCamera.yml**

Se considera una captura válida aquella que al aplicar los cálculos de calibrado, devuelve un error de reproyección menor de 0.25.

El siguiente paso consiste en calibrar el proyector. Al comienzo de este estado, se proyecta un patrón de círculos asimétrico, en una posición fija.

Colocar la superficie, de tal forma, que el patrón proyectado quede junto al del tablero de ajedrez. Volver a mover el patrón el espacio de la proyección hasta que el sistema adquiera 5 imágenes más. Para la aceptación de estas 5 imágenes, el error de reproyección cometido no debe ser mayor de 0.35.

En ese momento, la calibración pasa al modo dinámico, y los puntos proyectados se generan en función de la posición que se encuentre el tablero de ajedrez.

Tras capturar 15 imágenes con los puntos dinámicos, se refinan los cálculos correspondientes a los parámetros intrínsecos del proyector, y finalmente, se calculan los parámetros extrínsecos (matrices de rotación y traslación) entre la cámara y el proyector.

Los parámetros de calibrado se almacenan en los ficheros **calibrationProjector.yml** y **cameraProjectorExtrinsics.yml**.

Una vez realizada la calibración se iniciará un test para comprobar que se ha hecho correctamente. El test consiste en proyectar un círculo sobre las cuatro esquinas exteriores del patrón de tablero de ajedrez. Al mover el patrón los puntos proyectados deben siempre permanecer alineados con las esquinas.

Tanto el número de imágenes que se deben detectar antes de pasar a la siguiente fase, como los errores de reproyección máximos permitidos por la cámara y el proyector, son también configurables. Tras realizar numerosas pruebas variando estos parámetros, los mejores resultados manteniendo el compromiso *precisión-tiempo de calibrado-consumo de recursos*, se llega a la conclusión que los valores antes descritos son los que mejor se ajustan.

Tal y como se describe, el proceso completo de calibrado se puede realizar en varios minutos. Asimismo, mientras que la cámara y el proyector mantengan su posición y rotación entre ellos, no es necesario realizar una nueva calibración y es posible desplazar todo el sistema.

Anexo C

Manual de usuario

El presente anexo constituye un manual de usuario, incluye la instalación del sistema con sus dependencias, hasta su configuración inicial y puesta en marcha.

C.1 Instalación del cliente

Suponiendo que contamos con una Raspberry Pi con la distribución *Raspbian* correctamente instalada, la instalación de la aplicación cliente se divide en dos fases; instalación de dependencias y compilación del sistema.

C.1.1 Dependencias

El sistema requiere las siguientes dependencias para su funcionamiento:

```
libx11-dev
libfreetype6-dev
libsoil-dev
libsdl-1.2-dev
libboost-all-dev
libsdl-mixer-1.2-dev
libopencv-dev
```

Dichas dependencias pueden ser instaladas mediante el uso del comando `apt-get`:

```
# apt-get install libx11-dev libfreetype6-dev libsoil-dev \
    libsdl-1.2-dev libboost-all-dev libsdl-mixer-1.2-dev \
    libopencv-dev
```

Para utilizar la cámara CSI se requiere además la biblioteca «RaspiCam» [1]. Una vez descargada se procede a su compilación e instalación:

```
$ tar xvzf raspicamxx.tgz
$ cd raspicamxx
$ mkdir build
$ cd build
$ cmake ..
$ make
# make install
# ldconfig
```

[1] `http://www.uco.es/investiga/grupos/ava/node/40`

C.1.2 Compilación

Una vez instaladas las dependencias, se procede a compilar el sistema. Para ello, simplemente se ejecuta desde el directorio raíz de la aplicación:

```
$ make
```

C.2 Instalación del servidor

La instalación del servidor sigue el mismo proceso de antes, solo que puede ser llevada a cabo en cualquier computador.

C.2.1 Dependencias

El sistema requiere las siguientes dependencias para su funcionamiento:

```
build-essential
cmake
git
libgtk2.0-dev
pkg-config
libavcodec-dev
libavformat-dev
libswscale-dev
python-dev
python-numpy
libtbb2
libtbb-dev
libjpeg-dev
libpng-dev
libtiff-dev
libjasper-dev
libdc1394-22-dev
```

Dichas dependencias pueden ser instaladas mediante el uso del comando `apt-get`:

```
# apt-get install build-essential cmake git libgtk2.0-dev \
    pkg-config libavcodec-dev libavformat-dev libswscale-dev \
    python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev \
    libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
```

Después, se pasa a compilar la biblioteca OpenCV:

```
$ git clone https://github.com/Itseez/opencv.git
$ cd opencv
$ mkdir release
$ cd release
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
  -D CMAKE_INSTALL_PREFIX=/usr/local ..
$ make
# make install
```

C.2.2 Compilación

Una vez instaladas las dependencias, se procede a compilar el sistema. Para ello, simplemente se ejecuta desde el directorio raíz de la aplicación:

```
$ make
```

C.3 Guía de uso

En este apartado, se introduce el funcionamiento del sistema de tal forma que sirva de guía rápida al usuario.

C.3.1 Instalación de recursos

Los recursos como imágenes, sonidos, vídeos y fuentes, deben de respetar las rutas de directorios existentes:

- Los archivos de imagen deben de ir en el directorio `media/images` del cliente.
- Los archivos de sonido deben de ir en el directorio `media/sounds` del cliente.
- Los archivos de vídeo deben de ir en el directorio `media/videos` del cliente.
- Los archivos de fuentes *true-type* deben de ir en el directorio `media/fonts` del cliente.
- Los archivos de *script* ARS y configuración XML deben de ir en el directorio `data` del servidor.

C.3.2 Calibración del sistema

El proceso de calibrado del sistema se ha creado como una utilidad independiente del sistema principal de *ARgos*. Una vez calibrado el sistema, la aplicación proporciona los ficheros XML, con los parámetros intrínsecos y extrínsecos, que necesita *ARgos* para el calculo del registro.

C.3.3 Descripción de documentos

La descripción de los documentos que se puedan identificar en el sistema se realiza proporcionando una imagen en formato JPEG de una plantilla del documento.

C.3.4 Fichero de configuración

El archivo de configuración permite cambiar algunos parámetros del sistema. El archivo se encuentra en la carpeta `data` del servidor y presenta una estructura tal que:

```
<config>
  <paper_size width="21.0" height="29.7" />
  <calibration_files camera="calibrationCamera.xml"
```

```
    projector="calibrationProjector.xml"
    extrinsics="CameraProjectorExtrinsics.xml" />
  <argos_papers>
    <paper id="0" description="Neobiz"
      scriptFileName="NeobizScript.ars"
      descriptorFileName="neobiz-.jpg" />
    <paper id="1" description="Sinovo"
      scriptFileName="SinovoScript.ars"
      descriptorFileName="sinovo-.jpg" />
  </argos_papers>
  <output_display type="projector" />
</config>
```

El elemento `paper_size` indica las dimensiones de los documentos sobre los que se va a trabajar. En este caso, representa el tamaño de un papel en formato A4, es decir, 21.0x29.7 cm.

El elemento `calibration_files` indica los nombres de los archivos de calibración generados en el apartado anterior.

El elemento `argos_papers` define la lista de documentos con los que va a trabajar el sistema. Estos documentos mantienen una asociación con un archivo de *script* ARS y su imagen correspondiente.

El elemento `output_display` indica al sistema si la visualización la va a realizar un proyector o un monitor para realizar los cálculos del registro en función del dispositivo.

C.3.5 Ejecución

El arranque del sistema requiere ejecutar de forma separada tanto el servidor como el cliente, y en ese orden para obtener la dirección de conexión.

Ejecución del servidor. Se debe indicar el número de puerto y la interfaz de red donde escuchará el servidor:

```
$ ./argos_server <port> <iface>}
```

Ejecución del cliente. Se debe indicar la dirección IP y el puerto proporcionados por el servidor:

```
$ ./argos_client <ip>:<port>
```

Una vez ejecutado solo queda interactuar en la superficie de trabajo con los documentos definidos en los descriptores.

Anexo D

Código fuente

Dada la extensión del código fuente, éste no ha sido incluido en la documentación. Se incluye por tanto de forma digital en el CD adjunto a este documento.

D.1 *ARgos* Cliente

Para la aplicación cliente, es decir, aquella que se ejecuta en la Raspberry Pi, proyecto presenta la siguiente estructura de directorios

- **include:** Contiene los archivo de cabecera o *headers* C++ del proyecto.
- **libs:** Contiene las biblioteca externas `freetypeGlesRpi` y `glm`.
- **media:** Contiene los recursos del sistema categorizados en subdirectorios:
  - **fonts:** Contiene las fuentes *true-type* a usar por el sistema.
  - **images:** Contiene todas las imágenes del sistema.
  - **sounds:** Contiene todos los archivos de audio del sistema.
- **shaders:** Contiene todos los programas *vertex shader* y *fragment shader* usados por OpenGL ES 2.0.
- **src:** Contiene los archivos de código fuente C++ del proyecto.
- El directorio raíz contiene los archivos de calibración de la cámara y el *Makefile* necesario para la construcción del proyecto.

D.2 *ARgos* Servidor

Por otra parte, la aplicación servidora o aquella que se ejecuta en el computador externo presenta la siguiente estructura de directorios

- **data:** Contiene los archivos de configuración XML y *scripts* del sistema.
- **include:** Contiene los archivo de cabecera o *headers* C++ del proyecto.
- **src:** Contiene los archivos de código fuente C++ del proyecto.
- El directorio raíz contiene el *Makefile* necesario para la construcción del proyecto y el archivo de descriptores empleado para el reconocimiento de documentos.

D.3 *CalibrationToolbox* Servidor

- **data:** Contiene los archivos de calibrado YAML.
- **include:** Contiene los archivo de cabecera o *headers* C++ la aplicación.
- **src:** Contiene los archivos de código fuente C++ de la aplicación.

D.4 *CalibrationToolbox* Cliente

- **include:** Contiene los archivo de cabecera o *headers* C++ de la aplicación.
- **libs:** Contiene las biblioteca externas `freetypeGlesRpi` y `glm`.
- **src:** Contiene los archivos de código fuente C++ de la aplicación.

Anexo E

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.

- I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

5. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

6. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

7. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations

of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

8. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

9. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

10. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Referencias

[AAB84] E. Adelson, C. Anderson, y J. Bergen. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[AiJD13] Olivier Augereau, icholas Journet, y Jean-Philippe Domenger. Semi-structured document image matching and recognition. En *Proceedings 20th Document Recognition and Retrieval Conference, part of the IS&T-SPIE Electronic Imaging Symposium*, 2013.

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, y Angela Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *J. ACM*, 45(6):891–923, Noviembre 1998.

[AY11] Mitsuru Ambai y Yuichi Yoshida. CARD: Compact And Real-time Descriptors. En *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, páginas 97–104, Washington, DC, USA, 2011. IEEE Computer Society.

[Azu97] Ronald. T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[BA04] Kent Beck y Cynthia Andres. *Extreme Programming Explained: Embrace Change (Segunda Edición)*. Addison-Wesley Professional, 2004.

[BET08] H. Bay, A. Ess, y T. Tuytelaars. Speeded-Up Robust FeatureS. *Computer Vision and Image Understanding*, 110(3):346–359, Junio 2008.

[BL07] Matthew Brown y David G. Lowe. Automatic Panoramic Image Stitching Using Invariant Features. *Int. J. Comput. Vision*, 74(1):59–73, Agosto 2007.

[Bou00] Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm, 2000. url: `http://robots.stanford.edu/cs223b04/algo_tracking.pdf`.

[Can86] J. Canny. A Computational Approach To Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[CCSC13] Wan-Yu Chen, Jia-Lin Chen, Yu-Chi Su, y Liang-Gee Chen. Intelligent document capturing and blending system based on robust feature matching with an active camera. En *IEEE International Conference on Consumer Electronics (ICCE)*, páginas 131–132, Enero 2013.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, y Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. En *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, páginas 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.

[CMC03] Andrew I. Comport, Éric Marchand, y François Chaumette. A Real-time Tracker for Markerless Augmented Reality. En *Proceedings of the 2Nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '03, páginas 36–, Washington, DC, USA, 2003. IEEE Computer Society.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, y Vahab S. Mirrokni. Locality-sensitive Hashing Scheme Based on P-stable Distributions. En *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, páginas 253–262, New York, NY, USA, 2004. ACM.

[DKB11] Michael Donoser, Peter Kontschieder, y Horst Bischof. Robust Planar Target Tracking and Pose Estimation from a Single Concavity. En *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, páginas 9–15. IEEE Computer Society, 2011.

[DM10] Amaury Dame y Éric Marchand. Accurate real-time tracking using mutual information. En *ISMAR*, páginas 47–56. IEEE, 2010.

[dT13] Organización Internacional del Trabajo. Estudio: Factores para la inclusión laboral de las personas con discapacidad, 2013.

[EAH08] Berna Erol, Emilio Antúnez, y Jonathan J. Hull. HOTPAPER: Multimedia Interaction with Paper Using Mobile Phones. En *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, páginas 399–408, New York, NY, USA, 2008. ACM.

[EHH04] J. Ehnes, K. Hirota, y M. Hirose. Projected Augmentation - Augmented Reality Using Rotatable Video Projectors. En *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '04, páginas 26–35, Washington, DC, USA, 2004. IEEE Computer Society.

[GEHL03] Jamey Graham, Berna Erol, Jonathan J. Hull, y Dar-Shyang Lee. The Video Paper Multimedia Playback System. En *Proceedings of the Eleventh ACM Inter-*

national Conference on Multimedia, MULTIMEDIA '03, páginas 94–95, New York, NY, USA, 2003. ACM.

[HBESB11] Nate Hagbi, Oriel Bergig, Jihad El-Sana, y Mark Billinghurst. Shape Recognition and Pose Estimation for Mobile Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1369–1379, 2011.

[Hec01] David L. Hecht. Printed Embedded Data Graphical User Interfaces. *Computer*, 34(3):47–55, Marzo 2001.

[HF09] S.J. Henderson y S. Feiner. Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turre.t. En *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium.*, páginas 135–144, 2009.

[HS81] Berthold K. P. Horn y Brian G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17(1-3):185–203, 1981. url: `http://dblp.uni-trier.de/db/journals/ai/ai17.html#HornS81`.

[HS88] Chris. Harris y Mike. Stephens. A Combined Corner and Edge Detector. En *Proceedings of the 4th Alvey Vision Conference*, páginas 147–151, 1988.

[HYC02] Jun-Wei Hsieh, Shih-Hao Yu, y Yung-Sheng Chen. Morphology-Based License Plate Detection from Complex Scenes. En *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3 - Volume 3*, ICPR '02, páginas 30176–, Washington, DC, USA, 2002. IEEE Computer Society.

[HZ03] Richard Hartley y Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, edición 2, 2003.

[IKN+09] Kazumasa Iwata, Koichi Kise, Tomohiro Nakai, Masakazu Iwamura, Seiichi Uchida, y Shinichiro Omachi. Capturing Digital Ink As Retrieving Fragments of Document Images. En *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ICDAR '09, páginas 1236–1240, Washington, DC, USA, 2009. IEEE Computer Society.

[JKK+02] Keechul Jung, Kwang In Kim, Takeshi Kurata, Masakastu Kourogi, y JungHyun Han. Text Scanner with Text Detection Technology on Image Sequences. En *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3 - Volume 3*, ICPR '02, páginas 30473–, Washington, DC, USA, 2002. IEEE Computer Society.

[KB99] H. Kato y M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. En *Augmented Reality, 1999.*

(IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on, páginas 85–94, 1999.

[KHP93] Tapas Kanungo, Robert M. Haralick, y Ihsin T. Phillips. Global and local document degradation models. En *ICDAR*, páginas 730–734, 1993.

[KM94] F. Kishino y P. Milgram. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems.*, 1(12):1321–1329, 1994.

[KM09] Georg Klein y David Murray. Parallel Tracking and Mapping on a Camera Phone. En *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, páginas 83–86, Washington, DC, USA, 2009. IEEE Computer Society.

[LCS11] Stefan Leutenegger, Margarita Chli, y Roland Y. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. En *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, páginas 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society.

[LD08] Xu Liu y David Doermann. Mobile Retriever: Access to Digital Documents from Their Physical Source. *Int. J. Doc. Anal. Recognit.*, 11(1):19–27, Septiembre 2008.

[LF06] Vincent Lepetit y Pascal Fua. Keypoint Recognition Using Randomized Trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1465–1479, Septiembre 2006.

[LK81] Bruce D. Lucas y Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. En *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, páginas 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[LL12] Qiong Liu y Chunyuan Liao. PaperUI. En *Proceedings of the 4th International Conference on Camera-Based Document Analysis and Recognition*, CBDAR'11, páginas 83–100, Berlin, Heidelberg, 2012. Springer-Verlag.

[Low04] D. Lowe. Distinctive image features from scale-invariant key-points. *Intl. Journal of Computer Vision*, 60:91–110, 2004.

[MCUP02] Jiri Matas, Ondrej Chum, Martin Urban, y Tomás Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. En Paul L. Rosin y A. David Marshall, editors, *BMVC*. British Machine Vision Association, 2002.

[ML09] Marius Muja y David G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. En Alpesh Ranchordas y Helder Araújo, editors, *VISAPP (1)*, páginas 331–340. INSTICC Press, 2009.

[Mor12] Jorge Moraleda. Large Scalability in Document Image Matching Using Text Retrieval. *Pattern Recogn. Lett.*, 33(7):863–871, Mayo 2012.

[MT12] Daniel Moreno y Gabriel Taubin. Simple, Accurate, and Robust Projector-Camera Calibration. En *Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 3DIMPVT '12, páginas 464–471, Washington, DC, USA, 2012. IEEE Computer Society.

[NKI06] Tomohiro Nakai, Koichi Kise, y Masakazu Iwamura. Use of Affine Invariants in Locally Likely Arrangement Hashing for Camera-based Document Image Retrieval. En *Proceedings of the 7th International Conference on Document Analysis Systems*, DAS'06, páginas 541–552, Berlin, Heidelberg, 2006. Springer-Verlag.

[NKI09] Tomohiro Nakai, Koichi Kise, y Masakazu Iwamura. Real-Time Retrieval for Images of Documents in Various Languages Using a Web Camera. En *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ICDAR '09, páginas 146–150, Washington, DC, USA, 2009. IEEE Computer Society.

[NY09] U. Neumann y S. et al. You. Modeling and video projection for augmented virtual environments., 2009.

[OCLF10] M. Ozuysal, M. Calonder, V. Lepetit, y P. Fua. Fast Keypoint Recognition Using Random Ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010.

[Ort12] Raphael Ortiz. FREAK: Fast Retina Keypoint. En *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, páginas 510–517, Washington, DC, USA, 2012. IEEE Computer Society.

[PAK10] Apostolos P. Psyllos, Christos-Nikolaos E. Anagnostopoulos, y Eleftherios Kayafas. Vehicle Logo Recognition Using a SIFT-based Enhanced Matching Scheme. *Trans. Intell. Transport. Sys.*, 11(2):322–328, Junio 2010.

[Pin01] Claudio S. Pinhanez. The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. En *Proceedings of the 3rd International Conference on Ubiquitous Computing*, UbiComp '01, páginas 315–331, London, UK, UK, 2001. Springer-Verlag.

[PJN99] Jun Park, Bolan Jiang, y Ulrich Neumann. Vision-Based Pose Computation: Robust and Accurate Augmented Reality Tracking. En *Proceedings of the 2Nd*

IEEE and ACM International Workshop on Augmented Reality, IWAR '99, páginas 3–, Washington, DC, USA, 1999. IEEE Computer Society.

[PLC03] Hanchuan Peng, Fuhui Long, y Zheru Chi. Document Image Recognition Based on Template Matching of Component Block Projections. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1188–1192, Sept 2003.

[PLW09] Youngmin Park, Vincent Lepetit, y Woontack Woo. ESM-Blur: Handling & Rendering Blur in 3D Tracking and Augmentation. En *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, páginas 163–166, Washington, DC, USA, 2009. IEEE Computer Society.

[PS10] Julien Pilet y Hideo Saito. Virtually Augmenting Hundreds of Real Pictures: An Approach Based on Learning, Retrieval, and Tracking. En *Proceedings of the 2010 IEEE Virtual Reality Conference*, VR '10, páginas 71–78, Washington, DC, USA, 2010. IEEE Computer Society.

[PYN99] Jun Park, Suya You, y Ulrich Neumann. Natural Feature Tracking for Extendible Robust Augmented Realities. En *Proceedings of the International Workshop on Augmented Reality : Placing Artificial Objects in Real Scenes: Placing Artificial Objects in Real Scenes*, IWAR '98, páginas 209–217, Natick, MA, USA, 1999. A. K. Peters, Ltd.

[RD05] Edward Rosten y Tom Drummond. Fusing Points and Lines for High Performance Tracking. En *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, páginas 1508–1515, Washington, DC, USA, 2005. IEEE Computer Society.

[RLK+04] H. Regenbrecht, T. Lum, P. Kohler, C. Ott, M. Wagner, W. Wilke, y E. Mueller. Using augmented virtuality for remote collaboration. *Presence: Teleoperators & Virtual Environments.*, 13(3):338–354, 2004.

[ROW+03] H. Regenbrecht, C. Ott, M. Wagner, T. Lum, P. Kohler, W. Wilke, y E. Mueller. An augmented virtuality approach to 3D videoconferencing. En *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium.*, páginas 290–291, 2003.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, y Gary Bradski. ORB: An Efficient Alternative to SIFT or SURF. En *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, páginas 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[San07] David Marimón Sanjuan. *Advances in Top-Down and Bottom-Up Approaches to Video-Based Camera Tracking*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2007.

[SB01] Ken Schwaber y Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, edición 1st, 2001.

[SKR99] Didier Stricker, Gundrun Klinker, y Dirk Reiners. A Fast and Robust Line-based Optical Tracker for Augmented Reality Applications. En *Proceedings of the International Workshop on Augmented Reality : Placing Artificial Objects in Real Scenes: Placing Artificial Objects in Real Scenes*, IWAR '98, páginas 129–145, Natick, MA, USA, 1999. A. K. Peters, Ltd.

[ST94] Jianbo Shi y Carlo Tomasi. Good features to track. En *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 94, páginas 593–600. IEEE, 1994.

[Ste92] J. Steuer. Defining virtual reality: Dimensions determining telepresence. *Journal of communication.*, 42(4):73–93, 1992.

[Str13] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, edición 4rd, 2013.

[UG09] Raúl Ubeda González. Métodos ágiles para el desarrollo de software. Master's thesis, Universitad Politécnica de Cataluña, 2009.

[UM11] Hideaki Uchiyama y Eric Marchand. Toward Augmenting Everything: Detecting and Tracking Geometrical Features on Planar Objects. En *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, páginas 17–25, Washington, DC, USA, 2011. IEEE Computer Society.

[US09] Hideaki Uchiyama y Hideo Saito. Augmenting Text Document by On-line Learning of Local Arrangement of Keypoints. En *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, páginas 95–98, Washington, DC, USA, 2009. IEEE Computer Society.

[US11] H Uchiyama y H Saito. Random Dot Markers. En *Proceedings of the 2011 IEEE Virtual Reality Conference*, VR '11, páginas 35–38, Washington, DC, USA, 2011. IEEE Computer Society.

[WRM+08] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, y Dieter Schmalstieg. Pose Tracking from Natural Features on Mobile Phones. En *Proceedings of the 7th IEEE/ACM International Symposium on Mixed*

and Augmented Reality, ISMAR '08, páginas 125–134, Washington, DC, USA, 2008. IEEE Computer Society.

[YSSIT07] Gehua Yang, Charles V. Stewart, Michal Sofka, y Chia ling Tsai. Alignment of challenging image pairs: Refinement and region growing Starting From Single Keypoint Correspondence. *IEEE TRANS. PATTERN ANAL. MACHINE INTELL*, 11(23):1973–1989, 2007.

[ZFN02] Xiang Zhang, Stephan Fronz, y Nassir Navab. Visual Marker Detection and Decoding in AR Systems: A Comparative Study. En *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, ISMAR '02, páginas 97–, Washington, DC, USA, 2002. IEEE Computer Society.

Este documento fue editado y tipografiado con LaTeX
empleando la clase **esi-tfg** que se puede encontrar en:
`https://bitbucket.org/arco_group/esi-tfg`