ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

GESTIÓN DE UNA MINIFÁBRICA MEDIANTE REALIDAD AUMENTADA

Autor: Álvaro Bolinches Tejedor

Director: José Antonio Rodríguez Mondéjar

Madrid

Junio 2019

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Gestión de una Minifábrica mediante Realidad Aumentada

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2018/19 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Álvaro Bolinches Tejedor Fecha: 10/07/2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: José Antonio Rodríguez Mondéjar Fecha: 10/07/2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. **Álvaro Bolinches Tejedor**

DECLARA ser el titular de los derechos de propiedad intelectual de la obra:

Gestión de una minifábrica mediante realidad aumentada, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.
d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción

de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 10 de julio de 2019

ACEPTA

Fdo. Álvaro Bolinches Tejedor

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

GESTIÓN DE UNA MINIFÁBRICA MEDIANTE REALIDAD AUMENTADA

Autor: Álvaro Bolinches Tejedor

Director: José Antonio Rodríguez Mondéjar

Madrid

Junio 2019

Agradecimientos

Querría agradecer en primer lugar a mi madre, esa persona que ha luchado tanto durante todo el año para salir adelante, luchado en silencio, para que yo pudiese seguir adelante con mis tareas mi trabajo, mi vida, y siempre será mi espejo donde mirarme para poder enfrentarme al resto de cosas en este mundo.

También a mi padre, que con su apoyo he podido conducir este periodo complicado y que, hoy, ya podemos decir juntos, que está terminado, que todas las adversidades de este año parecen superarse, y que, a partir de ahora, gracias a su apoyo podré vivir mi propia experiencia en la vida.

Por último, a María, mi pilar, cuando tenía que aguantar la dificultad, el dolor que no podía expresar, el miedo, las buenas y malas experiencias juntos ya después de muchos meses compartiendo nuestra vida, allí está ella para ayudarme, aconsejarme y hacer que todas las cosas sean sencillas. Gracias de verdad.

Gracias a todos vosotros, a las demás personas cercanas que no nombré que me han ayudado en todo momento sin yo pedírselo.

GESTIÓN DE UNA MINIFÁBRICA MEDIANTE LA REALIDAD AUMENTADA

Autor: Bolinches Tejedor, Álvaro.
Director: Rodríguez Mondéjar, José Antonio.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Proyecto de investigación que trata de descubrir las diferentes posibilidades en el mundo de la realidad aumentada con un análisis exhaustivo, y así poder dar una solución real y sencilla aplicando estas técnicas a la gestión y mantenimiento de una fábrica.

Palabras clave: Algoritmo de reconocimiento, Gestión, Realidad aumentada, Smartphone.

1. Introducción

El aumento de la complejidad de los procesos de fabricación, la reducción de tiempo para realizar las tareas de mantenimiento predictivo y correctivo hace necesario dotar al operario de mantenimiento de herramientas más sencillas y que agilicen las tareas.

Por tanto, con la parte de investigación se quiere apoyar teóricamente a la realización de nuevas aplicaciones en el mundo de los servicios con técnicas de realidad aumentada, y gracias a la investigación de este proyecto sea más sencillo poder realizar nuevas soluciones a distintos problemas.

2. Definición del proyecto

El proyecto presentado consiste en la aplicación de las técnicas de realidad aumentada a la gestión y mantenimiento de una fábrica [1]. Para realizar un estudio y resolución óptimo se va a centrar el caso de estudio en la minifábrica existente en la planta 2ª de la Escuela Técnica Superior de ICAI, siendo extrapolable para otras infraestructuras.

En primer lugar, para la hora de investigar los algoritmos de reconocimiento se utilizarán diferentes informes y artículos científicos de cada uno de los métodos para así sintetizar y recoger las características y parámetros más importantes para este proyecto.

Una vez realizada la parte de investigación del proyecto, se va a realizar un pequeño sistema para demostrar el funcionamiento de estos algoritmos.

3. Descripción del sistema

Para el sistema a desarrollar se ha escogido el algoritmo *Speeded Up Robust Features* (SURF) [2] como algoritmo de identificación de las etiquetas, las cuales podrían ser cualquier tipo de etiqueta (polígonos, etiquetas lingüísticas, etc.), pero, para el sistema a realizar se utilizarán finalmente códigos de barras y códigos QR.

El sistema podrá utilizarse en modo administrador en el que se puedan añadir etiquetas nuevas, quitarlas, editar ciertos parámetros, etc. Este modo debe existir como seguimiento ya que la planta puede crecer en cuanto a número de máquinas, por ejemplo, y se quiere un sistema final que sea escalable al tamaño futuro de la minifábrica.

Mientras que en donde se realizará la lectura de la etiqueta, reconocimiento de la máquina y, por tanto, la parte de gestión de la planta de la minifábrica. En este modo entrará toda la operativa de gestión y lo que se quiera implementar en proyectos futuros.

4. Resultados

La investigación de los algoritmos ha arrojado distintos resultados, en primer lugar, el algoritmo ORB es claramente el más eficiente y robusto cuando se utilizan dispositivos con una potencia de procesamiento limitada, en cambio, cuando las especificaciones no son un problema, tanto los algoritmos ORB o SURF obtienen los mejores resultados. Por otro lado, una vez finalizada la solución de la minifábrica se obtiene una herramienta integrada e intuitiva para cualquier tipo de usuario.

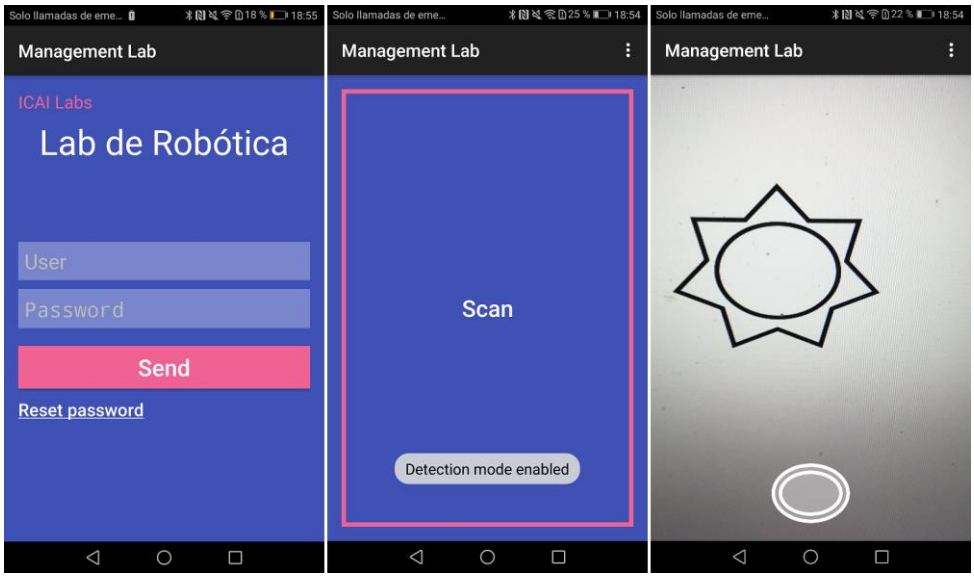

Figura 1. Página principal de la aplicación

5. Conclusiones

Con el desarrollo final de la herramienta se ha obtenido una solución final que aporta un grado de mejora a la organización de cualquier estructura de máquinas, además, con la solución de la realidad aumentada se ha conseguido mejorar el mantenimiento predictivo aportado una sencillez tanto al usuario como al administrador a la hora de tratar con un gran número de dispositivos.

Además, con la investigación se ha podido aclarar la posibilidad de utilización de cada uno de los algoritmos en trabajos futuros, ya que en la investigación se ha definido las virtudes y defectos de cada uno de los algoritmos y así poder adecuar los distintos proyectos futuros a los mejores algoritmos o los que han dado mejores resultados.

6. Referencias

[1] Rodríguez Mondéjar, J.A. “Mundo Industrial: Automatización, sistemas de control y supervisión”. Universidad Pontificia de Comillas, septiembre 2017.

[2] Bay, H; Tuytelaars, T; Van Gool, L. “SURF: Speeded Up Robust Features”. *ETH Zurich, Katholieke Universitet Leuven*, 2006.

MANAGEMENT OF A MINI FACTORY USING AUGMENTED REALITY

Author: Bolinches Tejedor, Álvaro.
Supervisor: Rodríguez Mondéjar, José Antonio.
Collaborating Entity: ICAI – Universidad Pontificia Comillas.

ABSTRACT

Research project that tries to discover the different possibilities in the world of augmented reality with an exhaustive analysis, and thus be able to give a real and simple solution applying these techniques to the management and maintenance of a factory.

Keywords: Augmented reality, Management, Recognition algorithm, Smartphone.

1. Introduction

The increase in the complexity of manufacturing processes, the reduction in time to perform predictive and corrective maintenance tasks makes it necessary to provide the maintenance operator with simpler tools that speed up tasks.

The research part of the project aims to make other contributions in the future to new applications in the world of services with augmented reality techniques, and thanks to the research of this project it will be easier to find new solutions to different problems.

2. Project definition

The project presented consists of the application of augmented reality techniques to the management and maintenance of a factory [1]. To carry out an optimal study and resolution, the case study will focus on the existing mini factory on the 2nd floor of the ICAI School of Engineering, which can be extrapolated to other infrastructures.

First, when investigating the recognition algorithms, different reports and papers of each of the methods will be used to synthesize and collect the most important characteristics and parameters for this project.

Once the research part of the project has been completed, a small system will be developed to demonstrate the functioning of these algorithms.

3. System's definition

For the system to be developed, the Speeded Up Robust Features (SURF) algorithm [2] has been chosen as the label identification algorithm. These labels could be of any type (polygons, linguistic labels, etc.), but for the system to be carried out, barcodes and QR codes will finally be used.

On the one hand, the system can be used as an administrator mode where you can add new labels, remove them, edit certain parameters, etc. This mode should exist as a follow-up to the mini-factory as the plant can grow in terms of number of machines, for example, and a final system that is fully scalable to the future size of the mini-factory is desired.

While where the reading of the label, recognition of the machine and, therefore, the plant management part of the mini factory will take place. In this mode, all the plant management operations and what you want to implement in other future projects will come into play.

4. Results

Research into the algorithms has yielded different results, firstly, the ORB algorithm is clearly the most efficient and robust when using devices with limited processing power, whereas when specifications are not an issue, both the ORB and SURF algorithms get the best results. On the other hand, once the mini-factory solution is completed, an integrated and intuitive tool is obtained for any type of user.

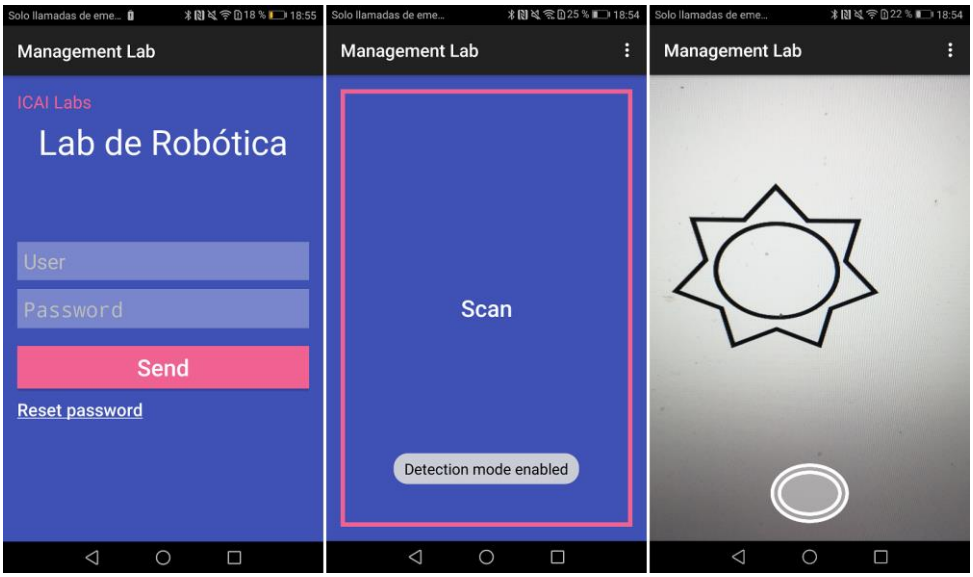

Figure 1. Application home page

5. Conclusions

With the final development of the tool we have obtained a final solution that provides a degree of improvement to the organization of any machine structure, in addition with the augmented reality solution has been able to improve predictive maintenance provided a simplicity to both the user and the administrator when dealing with a large number of devices.

In the research it has been possible to clarify the possibility of using each of the algorithms in future work, since the research has defined the virtues and defects of each of the algorithms and thus be able to adapt the different future projects to the best algorithms or those that have given better results.

6. References

[1] Rodríguez Mondéjar, J.A. "Mundo Industrial: Automatización, sistemas de control y supervisión". Universidad Pontificia de Comillas, September 2017.

[2] Bay, H; Tuytelaars, T; Van Gool, L. "SURF: Speeded Up Robust Features". *ETH Zurich, Katholieke Universitet Leuven*, 2006.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

1.1 CONTEXTO Y ANTECEDENTES

En una fábrica actual se pueden encontrar uno o varios procesos automatizados compuestos por sensores, accionamientos, un control, etc. Existen numerosas técnicas para la consecución de estos procesos, desde autómatas programables, a lenguajes de programación especializados en estos sistemas o formas de gestión y supervisión (pantallas, organizaciones jerárquicas, etc.) [1].

Estos procesos automatizados pueden ser modelados por el estándar ANSI/ISA-95 [2], y como se dice en el estándar (1990), "El propósito es crear un estándar que definirá la interfaz entre las funciones de control y otras funciones empresariales basadas en el modelo de referencia de Purdue University para CIM publicado por ISA. La interfaz inicialmente considerada es la interfaz entre los niveles 3 y 4 de ese modelo mientras que se considerarán interfaces adicionales, según corresponda. El objetivo es reducir el riesgo, el costo y los errores asociados con la implementación de estas interfaces. El estándar debe definir el intercambio de información que sea robusto, seguro y rentable. El mecanismo de intercambio debe preservar la integridad de la información de cada sistema y el alcance del control.".

Para entender mejor este modelo y contextualizar correctamente este proyecto se va a explicar este estándar mediante una pirámide de niveles, la cual será incremental empezando desde el nivel de un proceso o una acción concreta, hasta la planificación de la empresa de producción.

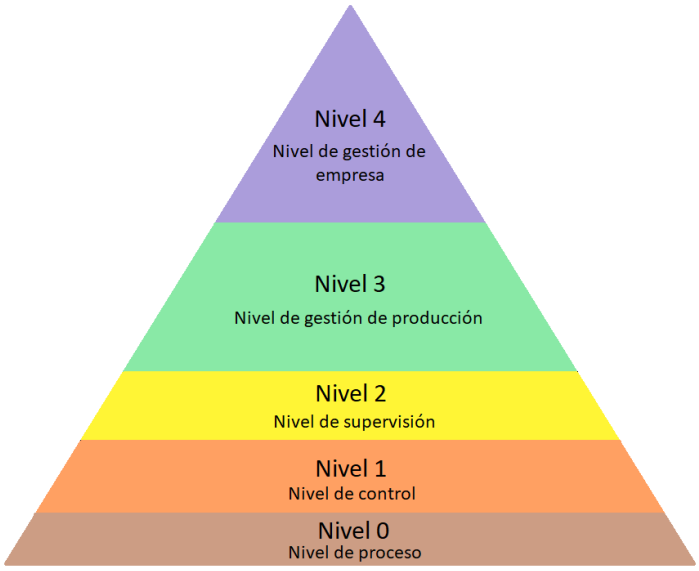

Figura 1. Modelo funcional ISA de un sistema automatizado

Como se puede observar en la pirámide anterior, el modelo ISA-95 establece cinco niveles en su estándar, los cuales son:

- Nivel 0 (Proceso): Actividad concreta, flujos de trabajo.
- Nivel 1 (Control): Control del directo del proceso, recogida de datos de sensores, controladores, etc.
- Nivel 2 (Supervisión): Gestión en tiempo real de la producción o de la planta en su conjunto, visualización, alarmas, etc.
- Nivel 3 (Gestión de producción): Gestión de órdenes, de recursos, gestión de calidad, mantenimiento.
- Nivel 4 (Gestión de la empresa): Por un lado, objetivos de producción y temas relacionados con la parte técnica, y por el otro, áreas como la administración, finanzas, recursos humanos, etc.

Por tanto, una vez contextualizado una planta de una fábrica, se puede afirmar que este proyecto estaría situado en el Nivel 2, en donde se va a situar la solución para la gestión en tiempo real de la producción.

1.2 ORIGEN DEL PROYECTO

Este proyecto se origina por la necesidad de una gestión más sencilla de laboratorios, fábricas o espacios donde se tienen dispositivos, máquinas o distintos elementos que se quieren tener controlados y verificados.

Además, a raíz de este proyecto, se planteó la investigación de la realidad aumentada para añadir un grado de investigación a este proyecto. Es por ello, que este proyecto se podría dividir en dos partes claramente diferenciadas, un componente de investigación y uno de desarrollo.

1.3 MOTIVACIÓN DEL PROYECTO

La motivación principal de este proyecto es lograr una solución sencilla a la hora de la gestión diaria en una fábrica. Es bien sabido que las gestiones actuales son realizadas con programas complejos y completos como SCADA [4] los cuales requieren un gran conocimiento técnico, por tanto, con este proyecto se busca realizar una solución que sea sencilla y limpia para el operario.

Por tanto, el fin último es proveer de un diseño técnico completo que incluya desde el mantenimiento de la solución (añadir, eliminar, modificar elementos de los dispositivos) hasta la propia operación de la solución donde las personas a cargo del control puedan acceder a información importante únicamente con una tableta y estando delante de la propia máquina.

Además, con esta solución se pretende integrar la plataforma al laboratorio de la planta 2ª de la Escuela Técnica Superior de Ingeniería ICAI para una gestión rápida por parte del profesorado y los alumnos de la universidad de los dispositivos existentes como robots, cámaras, etc. Por otro lado, otra de las motivaciones del proyecto, como se ha podido ver, es realizar un análisis detallado de los algoritmos existentes de reconocimiento para saber cuál es el más adecuado en qué casos, por tanto, se buscará ayudar a otras soluciones de reconocimiento de objetos con la investigación que se haga para esta solución particular.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En primer lugar, para la hora de investigar los algoritmos de reconocimiento se utilizarán diferentes informes y artículos científicos de cada uno de los métodos para así sintetizar y recoger las características y parámetros más importantes para este proyecto.

Para la realización de la aplicación móvil se utilizará inicialmente el lenguaje de programación Python o Java, todo dependerá del algoritmo que se escoja finalmente, si tiene como requisito utilizar algún lenguaje en concreto o si se puede adaptar al lenguaje que se quiera. Por tanto, para la aplicación móvil se utilizará alguno de los IDE más conocidos como son PyCharm si se utilizase el lenguaje Python o Eclipse IDE si se utilizase Java.

El siguiente objetivo es la creación e integración de una base de datos, si esta base de datos tuviese que ser integrada a la universidad se tendría que utilizar MySQL, es decir, la base de datos tendría que ser relacional, es decir, una colección de datos organizados en tablas y filas. En el caso de que no fuese necesario y el propio laboratorio pudiese tener una base de datos propia, sería conveniente utilizar una base de datos NoSQL como pueda ser MongoDB el cual es un sistema de base de datos orientado a documentos y en este caso cada una de las instancias se guardarían como si fuesen un objeto JSON, lo cual simplifica enormemente el diseño final de la base de datos, o utilizar un repositorio Cloud para alojar los documentos.

Por último, para la realización de la consola o interfaz hombre máquina es conveniente que el lenguaje de programación sea el mismo que el de la aplicación móvil integrada en la *tablet* del laboratorio. Esto es debido a que, aunque inicialmente la interfaz hombre maquina se realizara en una versión de escritorio, se valorará al final del proyecto la posibilidad de integración en la aplicación móvil, ya que en principio la persona de operación y la de mantenimiento serán distintas y por tanto las dos interfaces deberían estar separadas, aunque podría darse el concepto de que la persona sea la misma como en el caso del laboratorio de la universidad.

Capítulo 3. ESTADO DE LA CUESTIÓN

Centrándonos en el nivel de supervisión, este se caracteriza por distintas funciones, como pueda ser la visualización del estado actual de un proceso o maquinaria, sus características, etc. [1].

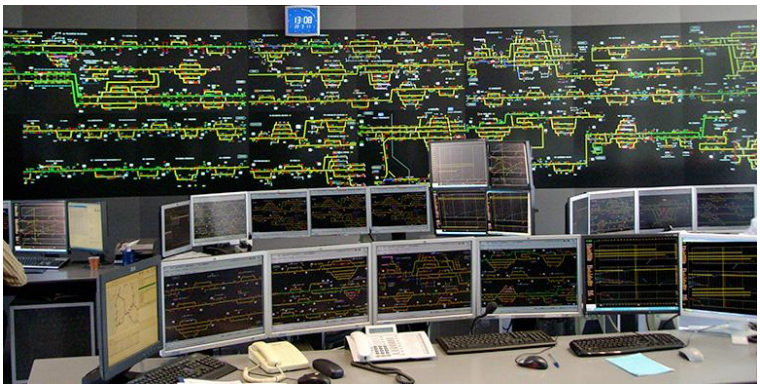

Figura 2. Ejemplo de centro de control. Fuente: CAF

Además, existen otras funciones como la gestión y listado de alarmas, el comportamiento de los datos supervisados, ya sean datos tanto de entrada como de salida, o la intercomunicación entre los niveles 0 y 2.

Además de la parte de nivel de gestión, y como se comentó al inicio, este proyecto abordará la aplicación de las técnicas de realidad aumentada para la propia gestión de la planta de la mini fabrica.

Para ello, se ha seleccionado el reconocimiento de objetos como técnica a la hora de añadir un punto de valor a la solución final, el reconocimiento de imágenes se engloba dentro de la visión artificial, la cual se define como “disciplina científica que destaca por el poder de procesar, analizar y comprender imágenes del mundo real” [3].

Existen numerosas técnicas y algoritmos a la hora de detección de objetos tanto en imágenes como en videos, estos se pueden dividir en dos grandes grupos, métodos basados en apariencias:

- Detección de bordes.
- Búsqueda "divide y vencerás".
- Detección y emparejamiento por escala de grises.
- Emparejamiento del gradiente.
- Histogramas de respuesta.

Y, en segundo lugar, métodos basados en características:

- Arboles de interpretación.
- Conjetura y prueba.
- Alineación.
- Agrupamiento de la pose.
- Invarianza.
- Invariantes geométricas para votar por hipótesis de objetos.
- Scale-Invariant feature transform (SIFT).
- Speeded Up Robust Features (SURF).
- Oriented Fast and Rotated Brief (ORB).

Debido a que se busca un algoritmo que sea válido para cualquier tipo de etiquetas, se necesitan algoritmos que detecten puntos clave independientemente de la etiqueta, para lo cual los algoritmos SIFT y SURF son ideales ya que a través de un conjunto de referencia se puede realizar una aproximación puntual de la imagen.

Esta determinación de este grupo de algoritmos es inicial, lo cual quiere decir que más adelante en el proyecto se irán analizando las distintas posibilidades hasta encontrar el que sea perfecto para las necesidades.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

En primer lugar, este proyecto busca que se simplifique de una manera rápida y sencilla todos los trabajos recopilados sobre la realidad aumentada. Como se sabe, numerosos son los algoritmos y métodos provistos para este tema, pero ninguno de los trabajos previos realizados engloba una explicación matemática y práctica de ellos.

Por otro lado, se quiere proveer una solución de gestión, sencilla y veloz, simplemente con un teléfono o cualquier aparato digital poder controlar un laboratorio, fábrica, etc. Es por ello por lo que se realizará una aplicación en donde gracias al algoritmo de reconocimiento que se escoja finalmente, la identificación las etiquetas, las cuales podrían ser cualquier tipo de etiqueta (polígonos, etiquetas lingüísticas, etc.), se realizará de una manera rápida para así tener una gestión más sencilla por parte de los administradores del sistema.

4.2 OBJETIVOS

Los siguientes objetivos expresan qué se persigue con el proyecto, contestan a las preguntas planteadas en el anterior apartado. Para ello, el proyecto se ha organizado en cuatro grandes objetivos los cuales a su vez pueden estar subdivididos:

Objetivo 1: Algoritmos para lectura de etiquetas en entornos industriales.

El primer objetivo del proyecto es investigar y analizar los distintos algoritmos existentes para conseguir una lectura de etiquetas correcto. Es requisito indispensable que el algoritmo escogido sea universal, es decir, que valga tanto para todo tipo de etiquetas (polígonos, imágenes, códigos de barras, códigos QR, etiquetas lingüísticas, etc.).

- Subobjetivo 1.1: Investigación de todos los algoritmos existentes, y listado de los algoritmos válidos según los requisitos, sus parámetros, y sus características.

- Subobjetivo 1.2: Comparación de los distintos algoritmos y sus parámetros para saber cuál es el idóneo para la solución final dependiendo de la tasa de acierto y el número de falsos positivos.
- Subobjetivo 1.3: Establecimiento de los parámetros adecuados del algoritmo escogido y pruebas finales.

Objetivo 2: Desarrollo de una aplicación móvil para mostrar en un dispositivo móvil.

Se realizará una aplicación móvil donde se encapsulará el algoritmo de identificación finalmente escogido, esta aplicación estará optimizada para una tablet genérica que llevará un sistema operativo Android.

- Subobjetivo 2.1: Modo mantenimiento, en el que se puedan añadir etiquetas nuevas, quitarlas, editar ciertos parámetros, etc. Este modo debe existir como seguimiento de la minifábrica ya que la planta puede crecer en cuanto a número de máquinas, por ejemplo, y se quiere un sistema final que sea totalmente escalable al tamaño futuro de la minifábrica.
- Subobjetivo 2.2: Modo operación, en donde se realizará la lectura de la etiqueta, reconocimiento de la máquina y, por tanto, la parte de gestión de la planta de la minifábrica. En este modo entrará en juego toda la operativa de gestión de la planta y lo que se quiera implementar en otros proyectos futuros.

Objetivo 3: Desarrollo de base de datos y consola asociada de mantenimiento.

En este tercer objetivo principal, se desarrollará una base de datos de acuerdo con los requisitos necesarios para que pueda estar integrada en el sistema de la Escuela, si no fuese posible o necesario, se optimizará de acuerdo con unos requisitos que se plantearán.

- Subobjetivo 3.1: Toma de requisitos e implantación de la base de datos en la minifábrica, además se realizarán pruebas de acceso.
- Subobjetivo 3.2: Realización de pruebas extensas de la base de datos, distintas operaciones como consulta, inserción, borrado, etc.

- Subobjetivo 3.3: Desarrollo de una consola asociada de mantenimiento. Es la consola o interfaz hombre máquina para introducir contenidos en la base de datos, por ejemplo, un informe con las características de un dispositivo que aparecerá en la tableta del usuario. Este desarrollo se realizará inicialmente en una versión de escritorio, aunque con la posibilidad de añadirlo a la versión móvil más adelante.

Objetivo 4: Carga de información y pruebas.

El último objetivo por realizar es la carga de información real en la base de datos, la puesta a punto de la aplicación móvil (carga de etiquetas, visualización del contenido, etc.) y la realización de pruebas unitarias a determinar según el avance del proyecto.

Aparte, habrá unas pruebas estándar como la tasa de éxito del reconocimiento de las etiquetas, el análisis de los falsos positivos, el tiempo de respuesta de la base de datos, el tiempo de respuesta de la aplicación, etc.

4.3 METODOLOGÍA

La metodología principal para el desarrollo de la solución software será Scrum [5], la cual se caracteriza por ser una metodología incremental, en vez de una ejecución completa, además también se caracteriza por ser secuencial en las distintas fases del proyecto, pero con solapamiento entre distintas actividades de la misma fase, como se verá más adelante en el cronograma que se adjuntará.

Por otro lado, a la hora de abordar la parte técnica del trabajo con los algoritmos de reconocimiento se utilizarán técnicas numéricas en la experimentación, para ello se realizarán pequeños scripts de cada uno de los algoritmos y de esta manera se podrán comprobar entre ellos.

Por ejemplo, se podrían utilizar distintos estadísticos conocidos, como pueda ser la tasa de acierto, el tiempo miedo a la hora de acertar la etiqueta, o el porcentaje de falsos positivo.

$$Tasa\ de\ acierto = \frac{numero\ de\ reconocimientos\ correctos\ de\ la\ etiqueta}{total\ de\ pruebas\ para\ una\ misma\ etiqueta}$$

$$Tiempo\ medio\ de\ conversión = \frac{\sum tiempo\ para\ el\ reconocimiento\ de\ la\ etiqueta}{total\ de\ pruebas}$$

$$\%\ Falsos\ positivos = \frac{numero\ de\ falsos\ positivos}{numero\ de\ reconocimientos\ de\ una\ etiqueta}$$

Una vez que se tiene claro la metodología Scrum para el desarrollo software y las distintas posibilidades de técnicas numéricas para la experimentación de los distintos algoritmos, se ha de incluir un cronograma (plan de trabajo) con diferentes fases, duración y secuenciación.

Para la realización de este plan de trabajo se ha utilizado Microsoft Project, en principio el proyecto estará compuesto por cinco fases: Estudio de Algoritmos de Reconocimiento, Desarrollo de una Solución móvil, Integración de Datos, Carga de Datos y Pruebas, y en paralelo con esas cuatro fases una fase llamada “Project Management”.

4.4 PLANIFICACIÓN Y RECURSOS A EMPLEAR

4.4.1 PLANIFICACIÓN INICIAL

Para la realización de este proyecto se ha realizado una planificación inicial con Microsoft Project, mostrada a continuación.

	de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Inicio	0 días	lun 01/01/18	lun 01/01/18	
2		**Estudio de Algoritmos de Reconocimiento**	**40 días**	**lun 01/01/18**	**vie 23/02/18**	
3		Investigación de los diferentes algoritmos	15 días	lun 01/01/18	vie 19/01/18	1
4		Comparación teórica de los distintos algoritmos	15 días	lun 22/01/18	vie 09/02/18	3
5		Realización de un script para cada algoritmo seleccionado	10 días	lun 12/02/18	vie 23/02/18	4
6		**Desarrollo de una Solución móvil**	**40 días**	**lun 12/02/18**	**vie 06/04/18**	
7		Desarrollo móvil versión 1.0	20 días	lun 12/02/18	vie 09/03/18	5CC
8		Comparación práctica de los distintos algoritmos	5 días	lun 12/03/18	vie 16/03/18	7
9		Desarrollo móvil versión 2.0	15 días	lun 19/03/18	vie 06/04/18	8
10		**Integración de Datos**	**25 días**	**lun 09/04/18**	**vie 11/05/18**	
11		Requisitos para la estructura de datos	5 días	lun 09/04/18	vie 13/04/18	9
12		Desarrollo de la base de datos	5 días	lun 16/04/18	vie 20/04/18	11
13		Desarrollo de la integración hombre máquina	10 días	lun 23/04/18	vie 04/05/18	12
14		Pruebas unitarias de la base y la integración	5 días	lun 07/05/18	vie 11/05/18	13
15		**Carga de Datos y Pruebas**	**15 días**	**lun 14/05/18**	**vie 01/06/18**	
16		Carga de datos reales	5 días	lun 14/05/18	vie 18/05/18	14
17		Pruebas del modo mantenimiento	10 días	lun 21/05/18	vie 01/06/18	16
18		Pruebas del modo operación	10 días	lun 21/05/18	vie 01/06/18	16
19		**Project Management**	**110 días**	**lun 01/01/18**	**vie 01/06/18**	
20		Documentación del proyecto	110 días	lun 01/01/18	vie 01/06/18	1
21		Reuniones semanales de seguimiento	110 días	lun 01/01/18	vie 01/06/18	1
22		Reuniones mensuales de fase	110 días	lun 01/01/18	vie 01/06/18	1
23		**Finalización del Proyecto**	**6 días**	**lun 21/05/18**	**lun 28/05/18**	
24		Finalización de la memoria final	5 días	lun 21/05/18	vie 25/05/18	16
25		Realización de la presentación final	5 días	lun 21/05/18	vie 25/05/18	16
26		Reunión final del proyecto	1 día	lun 28/05/18	lun 28/05/18	24;25
27		Fin	0 días	vie 01/06/18	vie 01/06/18	22;26;20;21;17;18

Figura 3. Plan de Trabajo del Proyecto Fin de Máster

Observando el cronograma en la imagen anterior, se comienza con la investigación de los distintos algoritmos, después se realiza una versión de "cartón piedra" de la aplicación en paralelo con la preparación de los distintos *scripts* de los algoritmos seleccionados para realizar las pruebas.

Una vez lista esa versión 1.0 y probado los algoritmos, se empezará a realizar una segunda versión de la aplicación que detecte las etiquetas de una manera más intuitiva. La tercera fase sería empezar con la parte de base de datos, trazar un pequeño documento de requisitos,

colecciones que se necesiten y a la par realizar la interfaz hombre máquina. Cuando todas las fases estén finalizadas se realizará una carga de datos y pruebas.

En paralelo con las cuatro fases comentadas, habrá una fase llamada "Project Management" en donde se realizará un control individual y periódico con el director del avance del proyecto, así como el avance con la memoria final.

Por último, habrá una sexta fase dedicada a la finalización de la memoria final añadiendo problemas dedicados y posibles mejoras en otros proyectos para hacer aumentar la funcionalidad de la solución integrada.

En el ANEXO A se detalla de una manera más gráfica la planificación inicial.

4.4.2 RECURSOS A EMPLEAR

En primer lugar, para la hora de investigar los algoritmos de reconocimiento se utilizarán diferentes informes y artículos científicos de cada uno de los métodos para así sintetizar y recoger las características y parámetros más importantes para este proyecto.

Para la realización de la aplicación móvil se utilizará inicialmente el lenguaje de programación Python o Java, todo dependerá del algoritmo que se escoja finalmente, si tiene como requisito utilizar algún lenguaje en concreto o si se puede adaptar al lenguaje que se quiera. Por tanto, para la aplicación móvil se utilizará alguno de los IDE más conocidos como son PyCharm si se utilizase el lenguaje Python o Eclipse IDE si se utilizase Java.

El siguiente objetivo es la creación e integración de una base de datos, si esta base de datos tuviese que ser integrada a la universidad se tendría que utilizar MySQL, es decir, la base de datos tendría que ser relacional, es decir, una colección de datos organizados en tablas y filas. En el caso de que no fuese necesario y el propio laboratorio pudiese tener una base de datos propia, sería conveniente utilizar una base de datos NoSQL como pueda ser MongoDB el cual es un sistema de base de datos orientado a documentos y en este caso cada una de las instancias se guardarían como si fuesen un objeto JSON, lo cual simplifica enormemente el diseño final de la base de datos.

Por último, para la realización de la consola o interfaz hombre máquina es conveniente que el lenguaje de programación sea el mismo que el de la aplicación móvil integrada en la *tablet* del laboratorio. Esto es debido a que, aunque inicialmente la interfaz hombre maquina se realizara en una versión de escritorio, se valorará al final del proyecto la posibilidad de integración en la aplicación móvil, ya que en principio la persona de operación y la de mantenimiento serán distintas y por tanto las dos interfaces deberían estar separadas, aunque podría darse el concepto de que la persona sea la misma como en el caso del laboratorio de la universidad.

Capítulo 5. ALGORITMOS DE RECONOCIMIENTO

Como se comentó en los apartados anteriores, este proyecto está compuesto por una parte de investigación y por otra de diseño, diseño teniendo en cuenta el estudio realizado en este Capítulo 5.

Para entender el concepto del reconocimiento, y, por tanto, de la realidad virtual, se empezará a explicar desde un ejemplo sencillo para luego entrar en los distintos algoritmos de manera matemática y práctica.

5.1 LAS CARACTERÍSTICAS

Para comenzar con esta sección de investigación se plantea un pequeño ejercicio, obsérvese la siguiente figura.

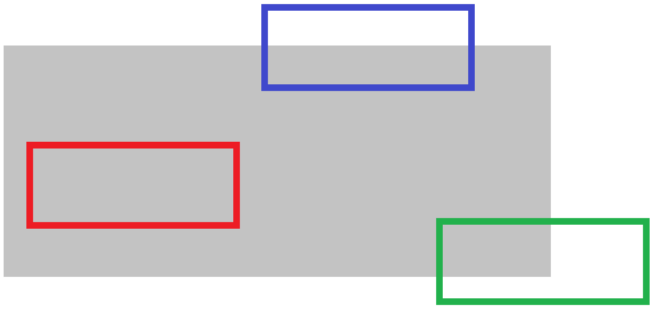

Figura 4. Ejercicio práctico sobre las características

Si se tiene en cuenta cada uno de los rectángulos por separado y se tuviese el fondo gris, ¿Serían capaces de adivinar la posición de cada uno de los tres rectángulos con exactitud?

Está claro que si se coge el rectángulo rojo en donde el fondo es todo gris es imposible adivinar de donde salió, ya que hay mucha área gris.

Teniendo en cuenta el rectángulo azul, las posibilidades se reducen ya que cualquier movimiento vertical descartaría todas las posiciones menos una, en cambio también hay

infinitas posibilidades en todo el borde superior, ya que se mueva hacia donde se mueva siempre parecerá el mismo.

Por último, cogiendo el rectángulo verde, el cual corresponde a una esquina, por mucho que lo muevas en cualquier posición será siempre distinto, por tanto, es único. Es único porque es una región de la imagen la cual tiene una máxima variación cuando la mueves, mientras que en los otros dos casos o la variación es nula o no es máxima, y de esta manera sencilla se explica lo que es una característica en el reconocimiento de imágenes y como a este objetivo se le llama detección de características.

Ampliando este concepto un poco más, para explicar una característica propia en una imagen, lo que una persona haría sería coger la región y explicar lo que tiene alrededor, por tanto, lo que realmente está haciendo es describir la característica en la que se ha fijado, y este concepto en realidad virtual se conoce como descripción de características.

Estos dos conceptos de detección y descripción de características son fundamentales, ya que son los pilares en los que se sustentan cualquier algoritmo o modelo de reconocimiento.

Por otro lado, es importante diferenciar dos conceptos que muchas veces en los modelos de reconocimiento se solapan o se confunden en su significado:

- Una esquina se puede definir como la intersección de dos bordes, o un punto en el plano (X, Y) en el cual hay dos direcciones dominantes y con diferente vecindad local, generalmente un máximo local.
- Un punto de interés, en cambio, es un concepto más generalista, se podría definir como un punto característico que se ha identificado como diferente a los demás. Por ejemplo, una esquina sí sería un punto de interés, pero también lo sería un punto donde su intensidad es máxima, un punto donde la tangente es máxima, o un punto donde la curvatura es máxima.

Por tanto, se ha de diferenciar el concepto, ya que los algoritmos más modernos no utilizan esquinas, matemáticamente hablando, en sus desarrollos.

5.2 DETECCIÓN DE ESQUINAS

El primer paso en la realidad aumentada, y concretamente en los algoritmos de reconocimiento fueron los detectores de esquinas. El primer detector conocido y publicado fue el Detector de Esquinas de Harris & Stephens, mientras que luego llegaron otros como el método de Plessey o el de Shi-Tomasi.

5.2.1 ALGORITMO DE HARRIS & STEPHENS

Concretamente, en 1988 se publicó un artículo llamado *A Combined Corner and Edge Detector* [7], en donde se plasmó la idea fundamental del ejercicio que se planteó en el anterior subapartado, pero de una manera matemáticamente más correcta.

Concretamente buscaban la diferencia en el desplazamiento en todas las direcciones de un plano, quedando la ecuación de la siguiente manera:

$$E(u,v) = \sum_{x,y} w(x,y) \cdot [I(x+u, x+y) - I(x,y)]^2$$

Por tanto, la función objetivo es maximizar la ecuación, por lo que viendo los tres términos de la función lo que se debe maximizar es el segundo término. Siguiendo el artículo, aplicando lo desarrollos de Taylor y las derivadas se llega a la siguiente expresión:

$$\mathrm{E(u,v)} \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Donde:

$$M = \sum_{x,y} w(x,y) \cdot \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

Y, por tanto, I_x y I_y son las derivadas en x e y respectivamente.

Una vez que se tiene la ecuación para maximizar el segundo término, generaron una ecuación con los autovalores de la matriz M para determinar si un punto (u,v) es una esquina o no:

$$R = det(M) - k \cdot (trace(M))^2$$

Donde:

$$det(M) = \lambda_1 \cdot \lambda_2$$
$$trace(M) = (\lambda_1 + \lambda_2)^2$$

Con esta función demostraron que si el valor de |R| es pequeño ($\lambda_1 \sim \lambda_2 \sim 0$) la región es llana, que si R es negativa ($\lambda_1 >> \lambda_2$ o viceversa) la región es un borde, mientras que si R es grande ($\lambda_1 \sim \lambda_2 >> 0$) la región es un borde.

Una manera más rápida de verlo es con la siguiente gráfica.

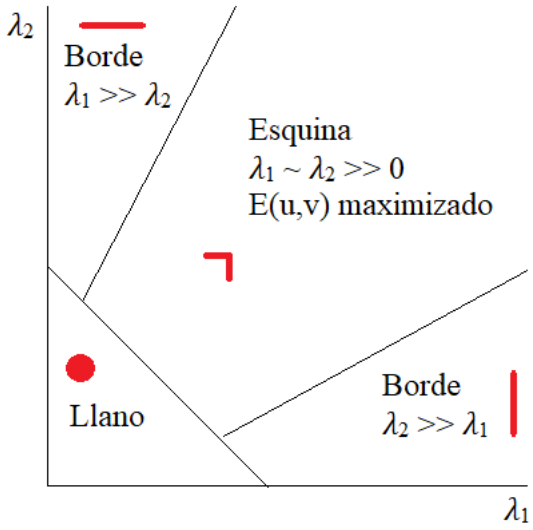

Figura 5. Representación gráfica del Algoritmo de Harris [7]

5.2.2 MÉTODO DE SHI-TOMASI

Para ver la evolución de los primeros algoritmos de reconocimiento, el siguiente método que se utilizó mucho fue el método de Shi-Tomasi publicado en un artículo llamado *Good Features to Track* el cual tenía pequeños cambios del algoritmo original de Harris & Stephens.

Como se vio, el algoritmo de Harris se reducía a la evaluación del determinante y la traza de M para decidir si el punto es una esquina o no.

$$R = det(M) - k \cdot (trace(M))^2$$

Mientras, que en su estudió, Shi & Tomasi proponen que la evaluación se realiza con los mínimos de los autovalores.

$$R = min(\lambda_1, \lambda_2)$$

Por tanto, la evaluación de esquina se reduce a que, si es mayor que un valor de *threshold* sea considerado como una esquina, por tanto, si se representa en una gráfica como en el caso de Harris quedaría la siguiente figura.

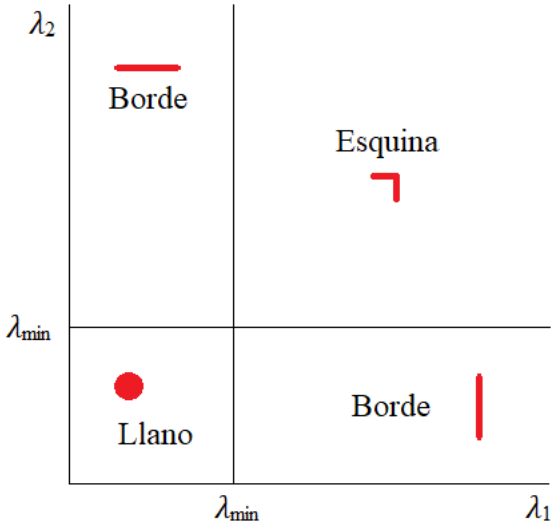

Figura 6. Representación gráfica del Algoritmo de Shi-Tomasi [8]

Por tanto, como se puede observar en la Figura 6 como con el algoritmo de Shi-Tomasi se reduce la complejidad matemática y programática a la hora de evaluar un posible punto de interés.

5.2.3 FEATURES FROM ACCELERATED SEGMENT TEST (FAST)

Por último, en el apartado de detectores de esquinas, se debe hacer hincapié en el algoritmo FAST propuesto por Edward Rosten y Tom Drummond en el artículo *Machine Learning for High-speed Corner Detection* [9] ya que es el algoritmo de esquinas más utilizado, sobre todo en *Machine Learning*.

El método FAST está englobado en los detectores AST, sigla que significa la prueba de segmento acelerada en inglés.

Los detectores AST se basan en un principio muy básico:

- Se coge el píxel *p* candidato a ser punto de interés y se debe evaluar los píxeles en un radio *r*.
- Se escoge un *threshold t*.
- Entonces, si los *n* píxeles consecutivos del círculo son todos más luminosos o menos luminosos que $I_p + t$ o $I_p - t$, el candidato *p* es considerado como una esquina.

Para realización de este principio, se utilizan arboles de decisión cortos para que la mayoría de los detectores del tipo AST sean muy eficientes.

En concreto, FAST es un método particular en donde tiene unos valores que se utilizan siempre por defecto:

- El radio *r* toma siempre el valor 3 (por lo que se tendrá un círculo de 16 píxeles).
- El valor de *n* píxeles consecutivos puede ser algo más variable, entre 9 y 12, aunque ya con 9 píxeles consecutivos se logran resultados muy fiables.
- Como árbol de decisión se utiliza el algoritmo ID3, el cual se encarga de probar cada subsecuencia como crea conveniente según los resultados de los demás conjuntos.

Una manera gráfica de ver cómo funciona este algoritmo es con la siguiente imagen obtenida del documento original de Edward Rosten y Tom Drummond donde se ve un ejemplo de un posible punto de interés, *p*.

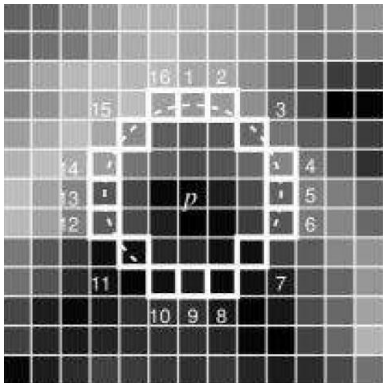

Figura 7. Representación gráfica del algoritmo FAST.

5.2.4 UTILIZACIÓN PRÁCTICA DE LOS DETECTORES DE ESQUINAS

La utilización de este algoritmo hoy en día está más limitada, pero originalmente era el algoritmo utilizado para reconocimiento de imágenes, concretamente se enfocaba más en reconstrucción de frames, alineaciones y registro digital de imágenes.

Se utilizaba tanto para la realización de mosaicos en 2D como en modelado de escenas 3D como en la reconstrucción de dichos frames, como también captura de movimiento.

Además, estos algoritmos de detección de esquinas son utilizados para reconocimiento de paisajes y estructuras, más que para reconocimiento de personas o seres vivos. Aunque se sigue utilizando en ciertas aplicaciones de indexación de imágenes, recuperación basada en contenido y seguimiento de vídeo.

Por último, hay que indicar que este detector de puntos de interés se utiliza hoy como parte del algoritmo ORB, el cual se detalla en la 5.6.

5.3 SCALE-INVARIANT FEATURE TRANSFORM (SIFT)

Como se vio en el apartado anterior, los detectores de esquinas fueron los primeros algoritmos en la visión artificial o aumentada, estos algoritmos de los años 90 se caracterizan por ser invariantes a las rotaciones ya que las esquinas por mucho que se las oriente siguen siendo únicas en el espacio, lo que estos algoritmos no podrían salvar era el cambio de escalado.

Se puede observar este problema con un ejemplo sencillo, observando la siguiente figura se puede ver en la izquierda un cuadrado en el que se ha detectado la esquina con una ventana de tamaño *x*. Con esta misma ventana se ha intentado identificar la esquina de este cuadrado que ha sido ampliada, y como se puede observar es imposible identificarla.

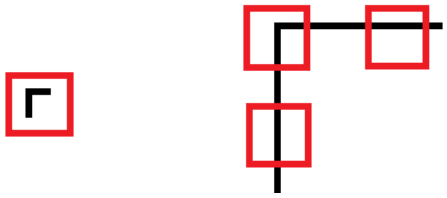

Figura 8. Problema de escalado en los algoritmos de detección de esquinas

Por tanto, en 1999 David Lowe empezó a trabajar en un nuevo algoritmo que intentaba salvar este problema de escalado publicando su primer artículo sobre SIFT, aunque no fue hasta 2004 cuando publicó definitivamente el algoritmo en el artículo *Distinctive Image Features from Scale-Invariant Keypoints* [11].

La principal motivación de SIFT es la de caracterizar puntos de interés independientemente de los cambios en la escala o el tiempo. Este algoritmo tiene cuatro pasos principales:

- Primero determina aproximadamente la localización y la escala de los puntos de interés, que a partir de ahora se llamaran *keypoints*.
- Seguidamente se refinará la localización y la escala.
- Se determinará la orientación u orientaciones de cada *keypoint*.
- Por último, se caracterizarán sus descriptores de cada *keypoint*.

5.3.1 ALGORITMO SIFT

5.3.1.1 Paso 1: Aproximación de la localización de los keypoints

El algoritmo SIFT basa su desarrollo matemático en la función gaussiana y el operador laplaciano, aunque en la práctica utilizará la diferencia gaussiana.

De una manera sencilla, SIFT identifica un punto en la imagen como:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Siendo la función *L(x,y,σ)* el resultado de la aplicación (convolución) del filtro gaussiano a la imagen para una escala determinada *σ*. Para obtener el filtrado de una imagen se obtiene con la convolución de la imagen con un filtro, siendo el resultado la respuesta al filtro.

Atendiendo al desarrollo estricto de SIFT, para poder identificar esquinas más grandes como la de la Figura 8 se necesitará una ventana más grande y para ello se utiliza un filtro que pueda cambiar de escala *σ*, siendo este resultado *L(x,y,σ)* conocido por el nombre de laplaciano de una función gaussiana, LoG (*Laplacian of Gaussian*).

Por tanto, se puede encontrar un máximo local a lo largo del espacio y para cualquier escala, lo que nos dará una lista de puntos *(x,y,σ)*, lo que significa que para un punto *(x,y)* se puede encontrar un punto de interés a una escala *σ*.

El problema reside en que el cálculo de la función LoG es complicado y poco eficiente, es por ello por lo que SIFT utiliza otro método de cálculo llamado diferencia de gaussianas DoG (*Difference of Gaussian*), en donde se utiliza dos escalas σ_1 y σ_2 de la siguiente manera.

$$D(x, y, \sigma) = G(x, y, \sigma_1) * I(x, y) - G(x, y, \sigma_2) * I(x, y) = \left(G_{\sigma_1} - G_{\sigma_2}\right) * I(x, y)$$

$$= L(x, y, \sigma_1) - L(x, y, \sigma_1)$$

Este proceso se realiza mediante las diferentes octavas de la imagen, una octava es una relación 2/1, por tanto, utilizando las octavas, las escalas adjuntas en la pirámide Gaussiana se diferencian en un factor *k*.

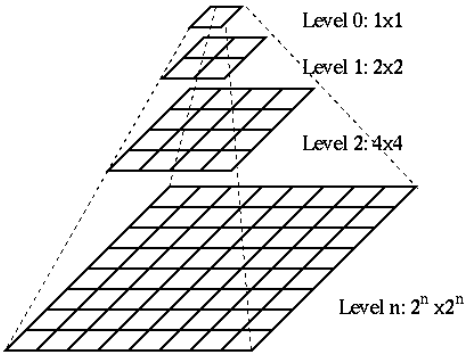

Figura 9. Pirámide Gaussiana (Fuente: http://fourier.eng.hmc.edu)

De esta manera, se utiliza el concepto de pirámide Gaussiana en el cálculo de la diferencia gaussiana, DoG.

> Para cada octava de espacio de escala, la imagen inicial se convoluciona repetidamente con funciones gaussianas para producir el conjunto de imágenes del espacio de escala que se muestra a la izquierda de la imagen. Las imágenes gaussianas adyacentes se restan para producir las imágenes de la diferencia gaussiana de la derecha. Después de cada octava, la imagen gaussiana es muestreada por un factor de 2, y el proceso es repetido (Lowe, 2004, p.6).

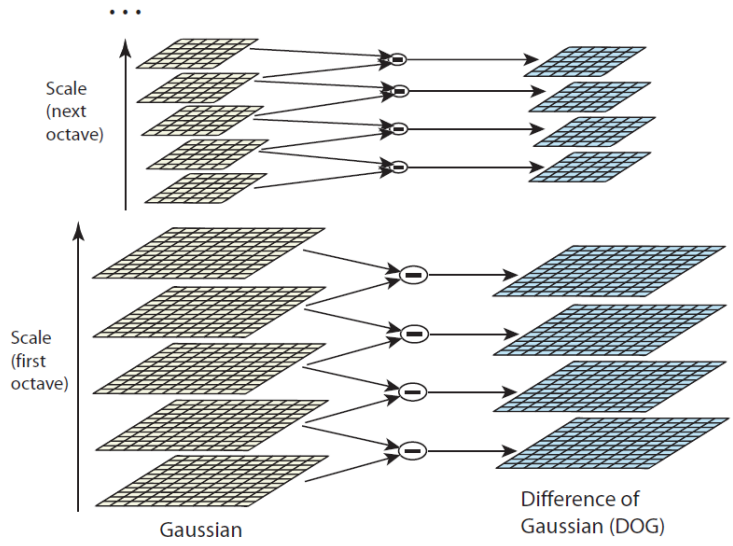

Figura 10. Diferencia gaussiana [11]

La diferencia gaussiana da como resultado una aproximación bastante exacta laplaciano de una función gaussiana, LoG, definido como $\sigma^2\nabla^2 G$ estudiada por Lindeberg (1994), siendo el efecto de escala σ^2 necesario para la invarianza de escalado, aunque no se va a estudiar la función LoG al estar fuera de alcance de SIFT.

Por tanto, siguiendo el algoritmo y una vez que se ha calculado la función DoG, se busca un máximo local en función del espacio y la escala.

Para calcular si un píxel *p* es un punto de interés se compara con sus 8 vecinos, y con diferentes escalas, generalmente se compara con los 9 píxeles de la siguiente escala y con los 9 píxeles de la escala previa, y si resulta ser un máximo local se considera como un *keypoint* potencial.

5.3.1.2 Paso 2: Refinado de la localización y la escala

En este segundo paso se realizarán dos operaciones, la primera será afinar más el valor de los potenciales *keypoints* obtenidos en el paso anterior, y después se comprobará si los *keypoints* pueden ser falsos positivos debido a ser bordes ya que la diferencia gaussiana generalmente también coge estos puntos.

En primer lugar, para obtener un resultado más refinado se utilizará la expansión de la serie de Taylor para el espacio-escala y así conseguir un resultado $|D_{extremal}|$ más exacto, esto se puede utilizar ya que la localización del *keypoint* y su escala es discreto y por tanto se puede interpolar para conseguir una mejor exactitud.

Entonces, Lowe definió un valor *threshold* de 0.03, por tanto, si $|D_{extremal}| < 0.03$ el punto de interés provisional se desecha por ser considerados extremos débiles o puntos de bajo contraste.

El segundo problema con el que nos encontramos son los numerosos falsos positivos que se obtienen con los bordes ya que estos puntos singulares también se obtienen con la función de la diferencia gaussiana DoG. Por tanto, para eliminarse SIFT utiliza la matriz de la 2ª derivada que se conoce como la matriz hessiana.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Igual que en los algoritmos de detección de esquinas, los autovalores de la matriz *H* dan mucha información sobre los puntos locales, de hecho, son el máximo/mínimo valor de curvatura de la superficie *D(x, y)* de la función DoG [12].

Además, como se comentó en el algoritmo de Harris, es sabido que un borde tiene el valor de curvatura máxima alto, pero el de curvatura mínima bajo, es decir uno de los dos autovalores es claramente mayor que el otro. En cambio, un punto de interés real ("esquina") tendrá los ambos autovalores del mismo orden de magnitud, así de esta manera se discriminarán los bordes y se obtendrán los *keypoints*.

Por tanto, con estas dos operaciones se eliminarán cualquier *keypoint* de bajo contraste y los que fuesen bordes para quedar así los *keypoints* más robustos.

5.3.1.3 Paso 3: Determinación de la orientación

El siguiente paso es asignar a cada uno de los *keypoints* resultantes una orientación dominante como orientación principal y así hacer invariante la rotación de la imagen.

En este proceso se escoge una vecindad alrededor del *keypoint*, cuyo tamaño de vecindad dependerá de la escala, y se calcula el tamaño del gradiente y dirección en toda la región.

Una vez calculado el gradiente y dirección de todos los puntos vecinos se transfiere los resultados a un histograma de 36 rangos que cubre los 360 grados como se puede observar en la Figura 11.

Una vez que se obtiene el histograma se asigna como orientación dominante al rango de mayor valor, aunque en el caso de que existan un pico mayor al 80% también es considerado para el cálculo de la orientación, es decir, se creara un descriptor de características (una práctica habitual en los algoritmos como se comentó en el Apartado 5.1) para cada una de las orientaciones obtenidas para la misma escala y localización.

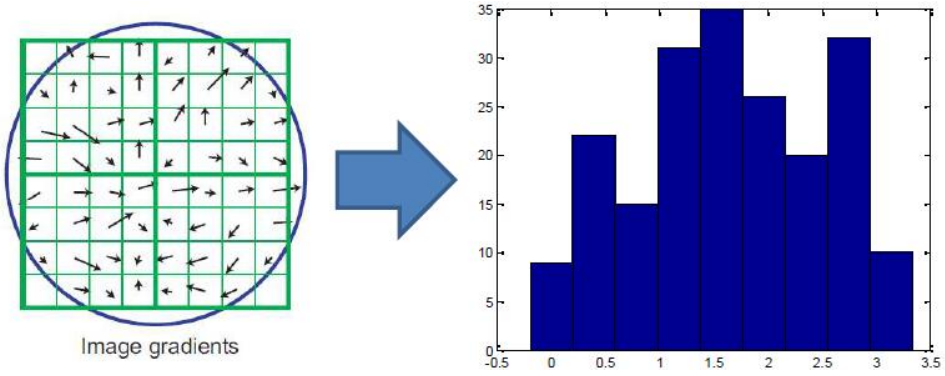

Figura 11. Gradiente e Histograma de una vecindad de un keypoint (Fuente: Departamento de Computer Science and Engineering del IIT Bombay)

Por ejemplo, en el caso de esta figura se elegirán hasta tres rangos para generar su descriptor de características, aunque notase que este es un ejemplo visual ya que no hay 36 rangos sino 9 pero sirve como ejemplo grafico de la operación.

5.3.1.4 Paso 4: Descripciones de cada keypoint

En este último paso se realizará la descripción de características de cada punto de interés según su/s orientación/es.

Para ello se escoge la vecindad de 16x16 píxeles alrededor del punto de interés y a su vez se divide en 16 sub-bloques de tamaño 4x4. Una vez que se tienen los bloques se construye un histograma de orientaciones para cada bloque.

> El descriptor del punto de interés se crea calculando primero la magnitud y la orientación del gradiente en cada punto de muestra de imagen en una región alrededor de la ubicación del punto clave, como se muestra a la izquierda de la Figura 12. Estos son ponderados por una ventana gaussiana, indicado por el círculo superpuesto.
>
> Estas muestras se acumulan en histogramas de orientación que resuman los contenidos en subregiones 4x4, como se muestra a la derecha, con la longitud de cada flecha corresponde a la suma de las gradientes cercanas a esa dirección dentro de la región. Esta figura muestra un conjunto de descriptores de 2x2 calculado a partir de

un conjunto de muestras de 8x8, mientras que los experimentos en este documento usan descripciones 4x4 calculadas a partir de una matriz de muestra de 16x16 (Lowe, 2004, p.15).

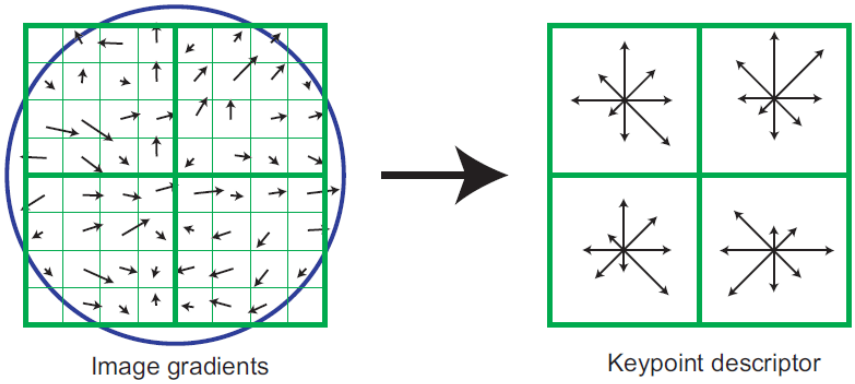

Figura 12. Descriptores de características de un keypoint [11]

5.3.2 RESULTADOS

En este punto se deben contestar ciertas preguntas del procedimiento anterior para dejar claro cómo funciona el algoritmo SIFT, como por que se utiliza la diferencia de gaussianas, DoG, porque hay que mirar a los extremos locales, o cuantas octavas se deben utilizar para tener resultados fiables.

En primer lugar, se utiliza la diferencia de gaussianas como aproximación a la función filtro de laplaciano de una función gaussiana, la cual matemáticamente sería el filtro apropiado de un filtro invariante a las rotaciones.

Los extremos locales son utilizados porque un máximo local indica píxeles negros mientras que un mínimo local india puntos brillantes, por tanto, con el cálculo de los autovalores se puede discriminar los extremos locales como se ha explicado.

En cuanto a las octavas, Lowe estableció empíricamente un numero de 3 escalas por octava es con la que se obtiene una repetibilidad máxima teniendo en cuenta el muestreo, la rotación, etc.

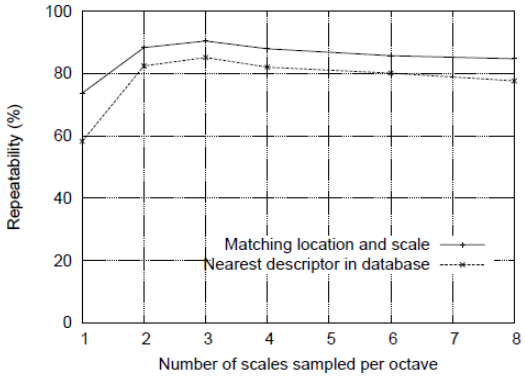

Figura 13. Porcentaje de keypoints que se detectan en la misma ubicación y escala en una imagen transformada en función del número de escalas muestreadas por octava [11]

5.3.2.1 Correspondencia entre keypoints de distintas imágenes

El resultado final buscado en este algoritmo es la correspondencia entre *keypoints* de dos imágenes que se quiere comparar, y se obtiene buscando el punto más próximo en el espacio *(x,y)* de los descriptores de los *keypoints*.

El problema reside en que en algunos casos el segundo resultado más próximo puede estar realmente cerca del primero, esto puede ser debido al ruido u a otras razones. En estos casos lo que se realiza es el ratio entre la distancia al más cercano y al segundo más cercano, y si el ratio es superior a 0.8 se desecha.

$$R = \frac{D_1}{D_2} \begin{cases} R < 0.8 \; se \; acepta \\ R > 0.8 \; se \; desecha \end{cases}$$

Lowe demostró con una muestra de 40.000 puntos de interés en una imagen con un cambio de escala aleatorio, una rotación del 30% y un ruido superior al 2%, que realizando esta prueba se descartaban en torno al 90% de los falsos positivos mientras que únicamente un 5% de las coincidencias correctas se descartaban.

Por tanto, los *keypoints* que se obtienen con el resultado ya son considerados los puntos de interés finales del algoritmo de SIFT, los cuales son invariantes a la rotación, traslación y escala.

5.3.3 UTILIZACIÓN PRÁCTICA DEL ALGORITMO SIFT

El algoritmo SIFT se utiliza en distintas aplicaciones prácticas como la identificación de localizaciones en distintas imágenes, reconocimiento en 2D y 3D, rastreo de movimiento, etc. aunque hay que tener en cuenta que este algoritmo esta patentado.

Comentando varios casos:

- Modelado de escenas 3D, en donde SIFT se utiliza para realizar imágenes 2D desde diferentes ángulos para modelar los objetos.
- Aplicaciones de acople de imágenes mediante lo que se conoce como la "costura" de imágenes, ya que mediante SIFT se obtienen los puntos de interés de todas las imágenes que se tienen para poder acoplarlas y sincronizarlas.
- La técnica de Morfometría basada en características, FBM (en inglés, *Feature-based morphometry*) utiliza el filtro de diferencia de gaussianas, DoG, para analizar y clasificar resonancias magnéticas del cerebro en seres humanos.

En definitiva, SIFT es un algoritmo utilizado actualmente a pesar de tener ya más de una década, pero es bastante fiable y con tasas de acierto altas, aunque tiene un problema esencial que es la iluminación ya que los cambios de brillo e intensidad en los píxeles hace que su eficacia sea menor.

5.4 SPEEDED UP ROBUST FEATURES (SURF)

El siguiente algoritmo se desarrolló un par de años después de SIFT, ya que, aunque SIFT es un algoritmo robusto no es demasiado rápido para lo que los usuarios necesitaban, es por ello que en 2006 se presentó el algoritmo SURF en el artículo *SURF: Speeded Up Robust Features* por Herbert Bay Tinne Tytelaars y Luc Van Gool en la Conferencia Europea de visión computacional [13].

Se demostró en aquel momento que SURF era comparativamente más rápido que SIFT ya que utilizaba un nuevo método para comparar y detectar los puntos de interés lo cual lo hace varias veces más rápido que SIFT.

El algoritmo SURF está basado en los mismos principios que SIFT, pero los detalles del algoritmo son distintos, aunque las partes en las que se divide el algoritmo son similares:

- Detección de los puntos de interés.
- Descripción de los píxeles con vecindad local.
- Comparación de los píxeles con los resultados de las imágenes.

5.4.1 ALGORITMO SURF

5.4.1.1 Paso 1: Detección de los keypoints

En SIFT, Lowe aproximo el laplaciano gaussiano (LoG) mediante la diferencia de gaussianas (DoG) que quitaba complejidad al cálculo y le otorgaba velocidad, aunque no lo necesario para el procesamiento de imágenes más complicadas, es por ello por lo que SURF va un poco más allá y utiliza un filtro cuadrado (*Box Filter*) como aproximación de la función LoG.

Una ventaja fundamental de esta nueva aproximación es que la convolución que se realiza con el filtro se calcula de una manera todavía más sencilla con la ayuda de imágenes integrales, donde además se puede realizar en paralelo para diferentes escalas.

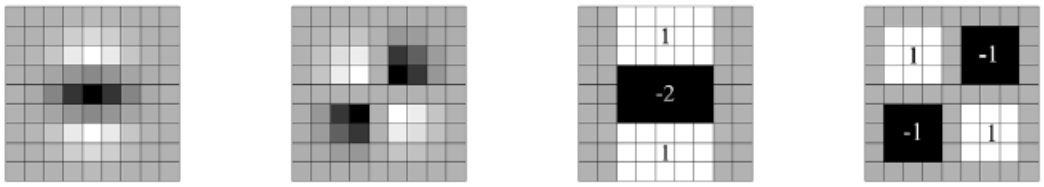

Figura 14. Comparación de la gaussiana de segundo orden tipo SIFT (a la izq.) con la aproximación del algoritmo de SURF (a la drcha.) [13]

Este ejemplo que se muestra en esta figura se puede ver como SURF aproxima una sección de una imagen en rectángulos en vez de evaluarlo individualmente los píxeles como en el algoritmo de SIFT. En la figura, las zonas grises corresponden a un valor 0, mientras que las zonas negras son negativas y las claras positivas, y de esta manera está demostrado que es más fácil calcular la convolución y obtener el resultado.

Para poder calcular estos rectángulos se utilizan imágenes integrales[1] que facilitan el cálculo de estos rectángulos:

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

Estas imágenes hacen que se pueda calcular el área de un rectángulo arbitrario con exactitud si se tiene la referencia del origen, para ello solo es necesario realizar la suma en valor absoluto de los distintos valores del rectángulo como se puede ver en la Figura 15, quedando:

$$\sum A - B - C + D$$

Siendo los valores de *A*, *B*, *C* y *D* los vértices del área rectangular, y por tanto el área S(*x*,*y*) quedaría evaluada de una manera mucho más rápida con el rectángulo (*A*, *B*, *C*, *D*).

[1] Una imagen integral es una imagen en donde cada punto x = $(x,y)^T$ guarda la suma de todos los píxeles de un área rectangular entre el origen y el valor de x.

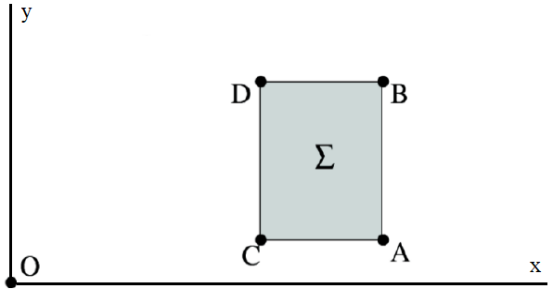

Figura 15. Área rectangular para el cálculo de una imagen integral

Además, SURF utiliza la matriz hessiana (similar a SIFT) para hallar los puntos de interés, concretamente se utiliza el determinante de la matriz hessiana para medir el cambio alrededor del punto o puntos, aunque otra novedad de SURF es que utiliza el determinante con el parámetro de escala como elección, no solo el píxel como se hace en SIFT.

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

5.4.1.2 Paso 2: Asignación de la orientación

Para conseguir una invarianza en la rotación se debe encontrar la orientación del punto de interés, para ello se utiliza el Wavelet de Haar [14].

El Wavelet de Haar se define como una secuencia de funciones de forma cuadrada que juntas forman el wavelet, lo cual permite que una función objetivo sobre un intervalo se represente en términos de una base ortonormal, esencial en este caso para la orientación.

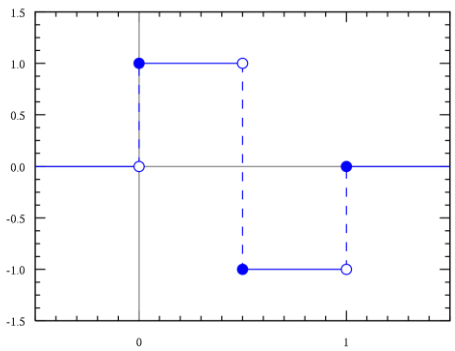

Figura 16. Wavelet de Haar (Fuente: Wikipedia)

Por tanto, se calculan las respuestas del Wavelet[2] de Haar tanto en las direcciones del eje *x* y eje *y*, dentro de una vecindad de tamaño 6σ alrededor del *keypoint* donde σ es la escala en la que se detectó este *keypoint*.

Estas respuestas se ponderan mediante una función gaussiana centrada en el *keypoint* y se trazan como puntos en el plano con la respuesta horizontal en la abscisa y la respuesta vertical en el eje de ordenadas.

La orientación principal se puede estimar si se calcula la suma de todas las respuestas calculadas dentro de una ventana de tamaño $\pi/3$, para sumarlas se suman todas sus componentes horizontales y verticales por separado, quedando como resultado un vector de orientación.

Esta misma operación se realiza en cada una de las ventanas de tamaño $\pi/3$ alrededor del *keypoint* dando como resultado distintos vectores de orientación local. Una vez que se tienen todos, el vector más largo es el que definirá la orientación del punto de interés.

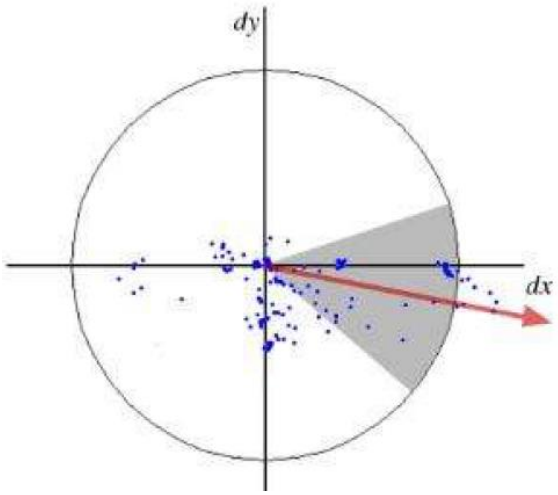

Figura 17. Vector de orientación local para una ventana especifica.

[2] El término original francés ondelette, introducido por Jean Morlet y Alex Grossmann, ha sido traducido al inglés como wavelet, y también al español como ondículas, ondeletas u onditas (Fuente: Wikipedia).

Se debe tener en cuenta el desplazamiento que se realiza con la ventana, ya que no es lo mismo deslizar la ventana 5º y tener 36 ventanas con 36 orientaciones locales que desplazar la ventana 30º cada vez con lo que se tendrían 6 ventanas con 6 orientaciones locales. Por lo cual, se debe tener un *match* entre la resolución, la fiabilidad y la velocidad de cálculo para conseguir la orientación adecuada sin hacer el algoritmo lento.

5.4.1.3 Paso 3: Descriptor de características

El descriptor de SURF está basado en las respuestas del Wavelet de Haar, como se comentó anteriormente, el objetivo principal de un descriptor es dar una descripción que sea única y robusta a un punto de interés o puntos. En el caso de SURF, para describir correctamente la vecindad se utilizará nuevamente las respuestas del wavelet en ambas direcciones horizontal y vertical.

Primero se construye una región cuadrada de tamaño 20*s*, donde *s* corresponde al tamaño, después se trocea en pequeñas regiones cuadradas de 4x4, y para cada una, se extrae la respuesta al Wavelet de Haar y se suavizan mediante un filtro gaussiano.

Para cada región se suman los resultados de las respuestas horizontales y verticales como a su vez el valor absoluto obteniendo así una descripción en un vector *v* lo suficientemente precisa.

$$v = \sum dx, \sum dy, \sum |dx|, \sum |dy|$$

De esta manera *v* es un vector único, que describe de una manera precisa el punto de interés, pero a su vez es robusto al ruido, robusto a las deformaciones y a la orientación, que en conjunto con la orientación calculada en el paso anterior hace que el *keypoint* esté perfectamente definido para su posterior comparación con otras imágenes o con frames de video.

5.4.2 RESULTADOS

El último paso de cualquier algoritmo, o mejor dicho el objetivo final de cualquier algoritmo de computación visual es conseguir *matching* entre distintas imágenes independientemente si han sido modificadas.

En este caso, para poder realizar ese *matching* se deberá comparar los descriptores hallados cuando se realizó el análisis del algoritmo en las distintas imágenes. Para realizar esta comparación se puede realizar de dos maneras:

- La primera, se obtienen los *keypoints* y los descriptores de la primera imagen, y realizar el mismo paso en la segunda imagen. Una vez que se tienen ambas cosas se pueden comparar los descriptores de las dos imágenes y luego ver la correspondencia entre los puntos.
- La segunda posibilidad es obtener los *keypoints* de la primera imagen y sus descriptores asociados. Después, se compara el descriptor de la primera imagen directamente con los puntos de la segunda imagen donde se cree que esta su posible *match* correspondiente.

Cualquiera de las dos maneras funciona correctamente, pero dependiendo de la que se utilice tiene sus ventajas y sus inconvenientes. En el primer caso, los números de falsos positivos son claramente inferiores que, en el segundo, pero, por otro lado, el tiempo de ejecución del programa de reconocimiento es mayor.

5.4.3 UTILIZACIÓN PRÁCTICA DEL ALGORITMO SURF

De manera breve, SURF se utiliza sobre todo en aplicaciones relacionadas con el reconocimiento de objetos obviamente, pero también de personas y sus caras ya que es más potente que SIFT.

También se utiliza en aplicaciones de realización de escenas 3D y escaneo de lugares, o también para el seguimiento y *tracking* de objetos, aunque es importante remarcar que como SIFT este algoritmo está también patentado.

5.5 BINARY ROBUST INDEPENDENT ELEMENTARY FEATURES (BRIEF)

El algoritmo de BRIEF aparece después de los algoritmos SIFT y SURF, publicado en el artículo *Binary Robust Independent Elementary Features* [15] por Michael Calonder, Vincent Lepetit, Christoph Strecha, y Pascal Fua.

Este algoritmo se introdujo como alternativa del algoritmo SIFT ya que requiere menos complejidad computacional y puede lograr una tasa de acierto similar a los algoritmos SIFT y SURF.

Esta complejidad se debe a que SIFT utiliza 128 bytes para los descriptores, y desde que se utilizan los números de coma flotante [16] utiliza 512 bytes. Además, el algoritmo SURF utiliza como mínimo 256 bytes con la coma flotante, por tanto, si se necesitan miles de descriptores para una única imagen se necesita un sistema de memoria bastante moderno que no todas las memorias soportan, y cuanta más memoria necesaria más tiempo de cálculo.

Es verdad que hoy en día no se necesita esas dimensiones de tantos bytes ya que se utiliza la compresión para reducir el tamaño, aunque el tiempo de cálculo sigue siendo inamovible, por ejemplo, se utilizan métodos como PCA o LDA o de *hashing* como LSH[3] que reducen drásticamente grandes volúmenes de datos.

Con estos métodos se convierten los descriptores de coma flotante a código binario (0 y 1), y con este código binario se utiliza la distancia de Hamming [17] para realizar el emparejamiento entre dos imágenes, lo cual le aporta mayor velocidad ya que con Hamming se aplica un XOR y un conteo de bits, y de esta manera se reduce drásticamente el tiempo de emparejamiento entre dos imágenes.

[3] En inglés, *Locality-sensitive hashing*.

Pero el problema principal no se resuelve, que es el del cálculo de los descriptores, ya que todo el proceso de compresión o *hashing* y la distancia de Hamming para reducir tanto tamaño como tiempo se calcula una vez teniendo las descripciones.

En este punto entra en juego el algoritmo BRIEF, ya que intentará calcular directamente los descriptores con código binario. Para ello, BRIEF intentará calcular individualmente los bits comparando las intensidades de un par de puntos a lo largo de las mismas líneas.

Con este método, el algoritmo 128 bits o como máximo con 256 bits obtendrá resultados verdaderamente fiables. Además, se pueden comparar códigos binarios entre sí mediante el cálculo de la distancia de Hamming lo que otorga al algoritmo una velocidad extrema.

El algoritmo define el valor test τ en el área p de tamaño *S x S* como:

$$\tau(p;x,y) := \begin{cases} 1 & si\ p(x) < p(y) \\ 0 & cualquier\ otro \end{cases}$$

donde p(x) es la intensidad del píxel.

Para el cálculo del descriptor de BRIEF un conjunto de n_d parejas *(x,y)* que definirían un conjunto binario, y el descriptor vendría dado por:

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1}\ \tau(p;x,y)$$

siendo n_d el número de bits necesarios para el correcto cálculo de las descripciones, ya sea n_d = 128, 256, o inclusive 512 en casos extremos. Por tanto, se referirá más adelante a BRIEF-*k*, donde *k* son el número de bytes necesarios para recoger el descriptor:

- Para casos de 128 bits se hablará de BRIEF-16.
- Para casos de 256 bits se hablará de BRIEF-32.
- Para casos de 512 bits se hablará de BRIEF-64.

Siendo el algoritmo BRIEF-32 el más común.

5.5.1 RESULTADOS Y APLICACIONES PRÁCTICAS

Para poder comentar los resultados de BRIEF se va a comparar este algoritmo con los dos anteriores estudiados, SIFT y SURF. Se compararán con dos medidas esenciales en estos algoritmos, la invarianza de rotación y la invarianza de escala [18].

	Invarianza rotacional	**Invarianza de escala**
SIFT	Bien	Bien
SURF	Bien	Bien
BRIEF-16	Limitado	Bastante bien
O-BRIEF	Bien	Bastante bien
BRIEF-32	Limitado	Bien
D-BRIEF	Muy bien	Muy bien

Tabla 1. Propiedades invariantes de los algoritmos SIFT, SURF y BRIEF

Teniendo en cuenta lo que se comentó en la página anterior y observando la tabla resumen anterior, se pueden sacar distintas conclusiones de BRIEF.

En primer lugar, se estudia la primera especificación de BRIEF-16 (128 bits) el cual está limitado a la invariabilidad en el plano, ya que se trata de una versión con una cantidad de bits reducidos.

La segunda variante, O-BRIEF, utiliza la especificación de 16 bytes, pero con una pequeña diferencia, se tiene una estimación orientativa de la orientación. Esta estimación puede darse, por ejemplo, mediante la detección de los puntos de interés o, cuando se utiliza un dispositivo para capturar la imagen por el sensor de gravedad del dispositivo.

Además, aumentando el número de bits a 256, BRIEF-32, se obtiene una gran mejora en la escala con respecto con BIEF-16 ya que al aumentar los bytes la información de escala si mejora exponencialmente.

La última variante que se comenta es lo que se ha llamado D-BRIEF (32 bits) por los autores del algoritmo [18], esta versión de BRIEF intenta que la invariabilidad quede solventada con la construcción de una base de datos, en donde se puede comparar con todas las versiones giradas y escaladas.

Esta solución no es complicada de aplicar ya que el tiempo de detección no se ve muy afectado, aunque en este caso entra en juego la variable de hardware ya que existirán conexiones entre el dispositivo y una base de datos.

Finalmente, el algoritmo BRIEF tiene usos parecidos a SIFT y SURF con pequeñas variaciones, se utiliza en distintas aplicaciones prácticas como la identificación de localizaciones en distintas imágenes, reconocimiento en 2D, mientras que tiene una tasa de acierto inferior en 3D por lo que temas de rastreo de movimiento no se utiliza tanto.

Por este motivo es posible que este algoritmo no se utilice en nuestra solución, ya que al capturar códigos existan pequeñas variaciones al tomar la imagen.

5.6 ORIENTED FAST AND ROTATED BRIEF (ORB)

El último algoritmo que se va a tratar es el que introdujo Ethan Rublee, Vincent Rabaud, Kurt Konolige y Gary R. Bradski en el artículo "*ORB: An efficient alternative to SIFT or SURF*" en 2011 [19].

El algoritmo ORB es básicamente una fusión del detector de puntos de interés FAST y el descriptor BRIEF con muchas modificaciones para mejorar el rendimiento. Además, este algoritmo, a diferencia de SIFT y SURF, no está patentado, sino que desde el principio fue *open source*.

Como se vio en la 5.2.3, FAST es un detector de puntos de interés en los detectores AST, sigla que significa la prueba de segmento acelerada en inglés. FAST tiene en un principio muy básico, el cual se puede resumir con las siguientes líneas:

- Se coge el píxel *p* candidato a ser punto de interés y se debe evaluar los píxeles en un radio *r*.
- Se escoge un *threshold t*.
- Entonces, si los *n* píxeles consecutivos del círculo son todos más luminosos o menos luminosos que $I_p + t$ o $I_p - t$, el candidato *p* es considerado como una esquina.

El principal problema de FAST es que no es invariante en la orientación, pero ORB soluciona este problema:

- En primer lugar, se calcula la intensidad media del centroide del punto con la esquina localizada en el centro.
- La dirección del vector desde este punto de esquina al centroide del punto calculado da la orientación.
- Por último, para mejorar la invariancia de rotación, se puede calcular los momentos con *(x, y)* que deben estar en una región circular de radio *r*, donde *r* es el tamaño del punto.

Para lograr esto, en primer lugar, se define el momento del punto como:

$$m_{pq} = \sum_{x,y} x^p y^q \ I(x,y)$$

Con este momento se puede calcular por tanto el centroide del punto:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

Y una vez que se tiene el centroide *C*, se puede calcular el vector desde este punto de esquina, *O* al centroide del punto calculado, *OC*, el cual será la orientación:

$$\theta = atan2(m_{01}, m_{01})$$

A esta modificación, los autores lo llamaron oFAST (*Fast Keypoint Orientation*), lo que hace que FAST sea más robusto.

Como segunda pieza del puzle, ORB utiliza para el análisis de los descriptores el algoritmo BRIEF, Pero como ya se vio en el apartado anterior funciona mal con la rotación, por lo tanto, lo que ORB hace es "dirigir" la descripción según la orientación de los puntos de interés calculados con oFAST, en el paso anterior.

Teniendo en cuenta los cálculos que se realizaron para BRIEF. El algoritmo define el valor test τ en el área p de tamaño *S x S* como:

$$\tau(p; x, y) := \begin{cases} 1 & si\ p(x) < p(y) \\ 0 & cualquier\ otro \end{cases}$$

Y después, para el cálculo del descriptor de BRIEF se debe escoger un conjunto de n_d parejas *(x,y)* que definirían un conjunto binario, por lo que el descriptor vendría dado por:

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1} \ \tau(p; x, y)$$

Se vio que BRIEF puede tomar diferentes distribuciones, pero en este caso ORB escoge el tamaño $n_d = 256$. Aunque, el principal problema reside en el gran número de pares de puntos que pueden existir, entonces ORB realiza los siguientes pasos:

- Se ordenan los pares de puntos por su distancia ponderada y se escoge los que están a 0.5, formando un vector T.
- Establecer un umbral *Th*.
- El siguiente paso es utilizar un algoritmo voraz [20]. Este tipo de algoritmo intenta llegar a la solución óptima mediante la elección óptima en cada paso.
    - Se escoge la primera en un vector R y se elimina de T.
    - Se toma la siguiente prueba de T y se compara con las del vector R, en el caso de que su correlación absoluta sea mayor que el umbral *Th*, se deshecha la prueba, de lo contrario se añade a R.
    - Se debe repetir este paso hasta rellenar los 256 pares de puntos, en caso de no conseguir la solución óptima, se debe bajar el valor del umbral *Th* y repetir el proceso.

Este proceso definido para ORB se conoce como rBRIEF, *Rotation-aware* BRIEF.

Por último, para el cálculo final de los descriptores, se utiliza como método de compresión el *hashing* LSH[4] multi-sonda que claramente mejora la eficiencia del LSH tradicional visto en BRIEF [19][21].

[4] En inglés, *Locality-sensitive hashing*.

5.6.1 RESULTADOS Y UTILIZACIÓN PRÁCTICA

De manera teórica, ORB es mucho más rápido que SURF y que SIFT, y también el descriptor de ORB descriptor funciona mejor que SURF. ORB es una buena opción en dispositivos de baja potencia para costuras panorámicas, reconocimiento en 3D, etc.

Existen numerosas pruebas realizadas con ORB se ha demostrado que el tiempo de procesamiento de ORB es claramente inferior al del resto de algoritmos. Por ejemplo, se toma el experimento realizado en el artículo original [19] en donde se utiliza las siguientes imágenes.

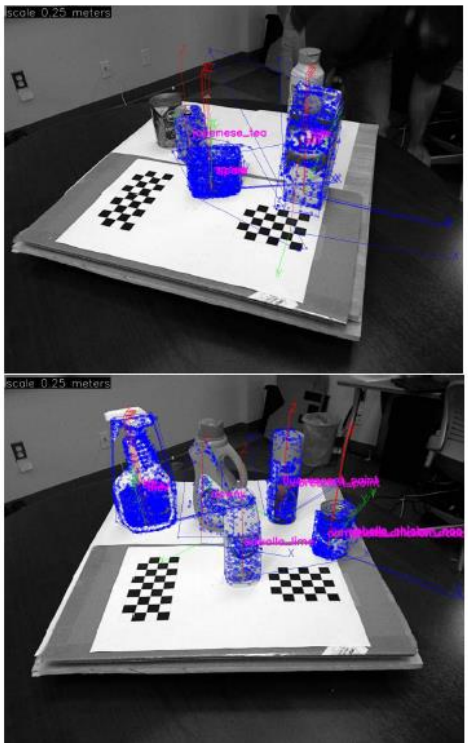

Figura 18. Dos imágenes en teoría similares en donde ha habido una rotación y, en donde la imagen superior ha perdido dos objetos que están en la inferior [19]

En este experimento, se compara el tiempo necesario para detectar las diferentes características (puntos de interés y descriptores) para los algoritmos ORB, SURF y SIFT. Para ello, se tomarán un total de 2686 imágenes como las mostradas en la figura anterior en 5 escalas distintas.

Según el artículo:

> El algoritmo de ORB fue capaz de detectar y calcular más de 2×10^6 puntos de interés en aproximadamente 42 segundos. Comparando con los algoritmos de SIFT y SURF, utilizando los mismos datos, para el mismo número de características (aproximadamente 1000), y el mismo número de escalas, obtenemos los siguientes tiempos. (Rublee, 2011, p.6).

Detector	ORB	SURF	SIFT
Tiempo por frame (ms)	**15.3**	217.3	5228.7

Tabla 2. Análisis de tiempo de detección por frame [ms][19]

En definitiva, se ha definido con este nuevo algoritmo una nueva posibilidad en la aplicación de la solución final del proyecto. ORB ha demostrado tener una gran eficiencia frente a otros algoritmos ya existentes, además, ORB es software libre por lo que para aplicaciones futuras se puede modificar para cada propia solución.

Siguiendo con este proyecto, en los siguientes apartados se va a realizar una prueba empírica de los algoritmos tratados hasta ahora para decidir cuál será el correcto algoritmo. Más concretamente se realizarán dos experimentos.

- El primero, consistirá en un experimento más teórico, es decir, se utilizará la fotografía de Lenna para comprobar cada uno de los algoritmos.
- Con el segundo experimento se utilizarán ya las etiquetas que probablemente se utilicen en la minifábrica, como posiblemente serán los códigos de barras y los códigos QR. En este experimento se comprobará primero la velocidad de detección, y los falsos positivos utilizando códigos degradados, girados y a diferente escala.

De esta manera se decidirá finalmente que algoritmo se utilizará.

5.7 COMPARATIVA ENTRE ALGORITMOS

Una vez analizado los distintos algoritmos se va a realizar una comparativa entre ellos para ver in situ las diferencias entre ellos. Los algoritmos que se utilizarán en la comparativa son:

- Detección de esquinas de Shi-Tomasi.
- FAST.
- SIFT.
- SURF.
- BRIEF.
- ORB.

Además, se utilizará también la fotografía de Lenna, al haberse convertido de facto en un estándar científico siendo la siguiente.

Figura 19. Imagen original de Lenna

Por tanto, con esta imagen se procesarán los seis algoritmos para observar el resultado final de cada uno de ellos.

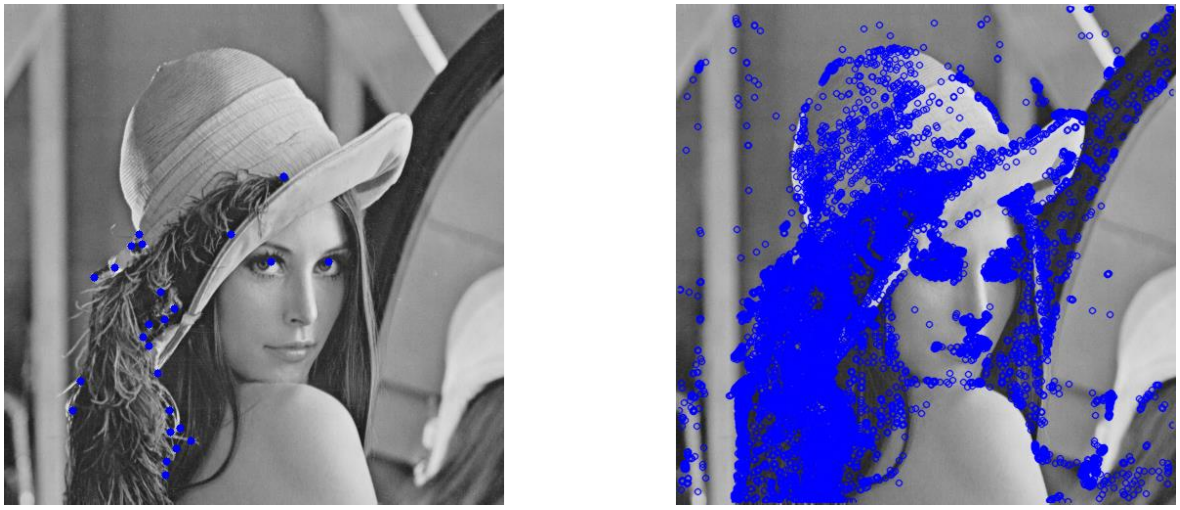

Figura 20. Puntos de interés con los algoritmos de Shi-Tomasi (izq.) y FAST (drcha..).

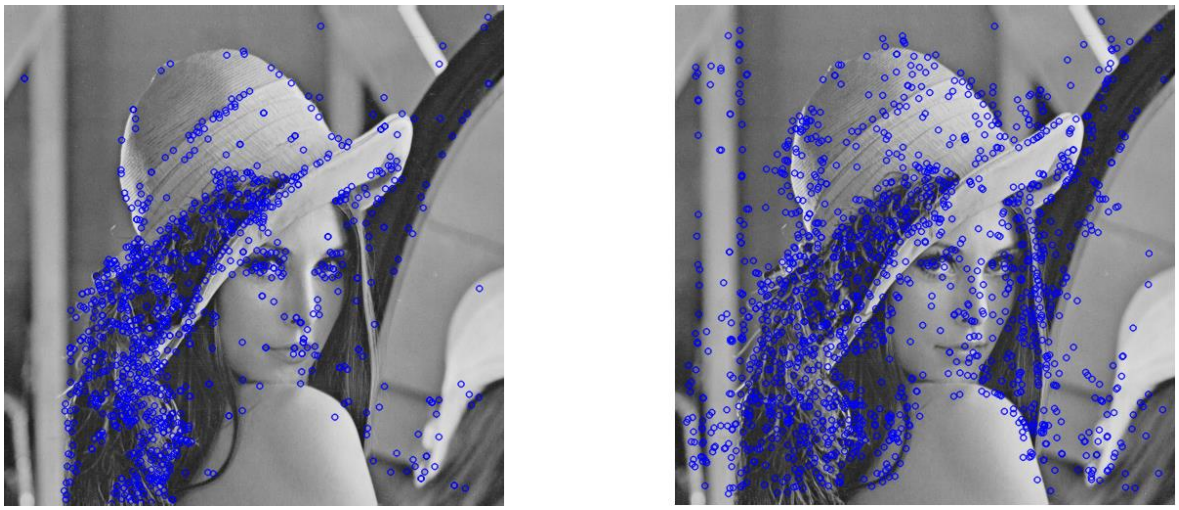

Figura 21. Puntos de interés con los algoritmos SIFT (izq.) y SURF (drcha.).

Figura 22. Puntos de interés con los algoritmos BRIEF (izq.) y ORB (drcha.).

Observando las diferentes imágenes se puede observar como la detección de esquinas es menos robusto que el resto de los algoritmos. Además, FAST si es más robusto que Shi-Tomasi, pero sacrifica un tiempo elevado de ejecución.

Por otro lado, en cuanto a los algoritmos de SIFT y SURF, sus similitudes son bastante amplias, pero se puede ver como SURF está más optimizado ya que para el mismo tiempo de ejecución calcula un mayor número de puntos de interés.

Por último, el algoritmo BRIEF consigue los tiempos de respuesta más pequeños ya que es mucho más selectivo en cuanto a puntos de interés y optimiza más su búsqueda, mientras que ORB es un paso más que BRIEF en donde el tiempo de ejecución es muy levemente superior, pero en cuanto al número de puntos de interés es mucho más preciso y su tasa de acierto claramente mayor.

En definitiva, con esta comparativa no está claro el algoritmo que coger, si SIFT, SURF u ORB (mejora del BRIEF), por lo que se realizará un pequeño experimento entre los algoritmos para elegir el óptimo para la solución técnica.

5.8 EXPERIMENTO ENTRE ALGORITMOS PARA LA APLICACIÓN

En este último apartado de la investigación, se va a centrar el estudio en el caso práctico que se está tratando y no un caso genérico. Como se comentó al principio de la memoria, se va a realizar un diseño para la gestión de una minifábrica, es por ello por lo que se van a utilizar ciertas etiquetas.

Como se verá en el diseño, se utilizarán básicamente dos tipos de etiquetas, y adicionalmente se realizará una prueba con etiquetas no estándar.:

- Código de barras ASCII.
- Códigos QR.
- Etiquetas no estándar.

Realmente se puede utilizar cualquier tipo de código de identificación, ya que estos algoritmos se caracterizan por analizar los puntos clave, las características o los descriptores, por lo que la etiqueta podría ser desde formas poligonales hasta emoticonos, lo único que se utilizará este tipo de etiquetas al ser un estándar en la industria.

Por tanto, el código de barras es lineal en donde los grosores de las barras codifican los diferentes caracteres mientras que los códigos QR son bidimensionales, por lo que pueden codificar un mayor número de caracteres.

Para la comparativa se van a utilizar las siguientes etiquetas, las cuales significan lo mismo, “robot 1” (se pueden escanear desde cualquier smartphone para comprobar que es correcto ese contenido).

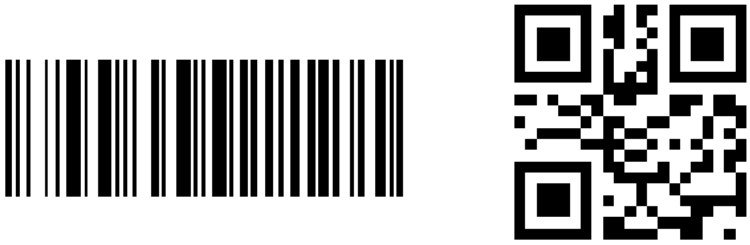

Figura 23. Código de barras y código QR del experimento

Una vez se tienen los dos códigos se va a utilizar los diferentes algoritmos para comprobar su utilidad, los que se utilizarán para el experimento serán (como en el apartado anterior):

- Detección de esquinas de Shi-Tomasi.
- FAST.
- SIFT.
- SURF.
- BRIEF.
- ORB.

Para dicha prueba, se realizarán distintos programas en código *Python* por cada uno de los algoritmos que se quieren comprobar. A continuación, puede encontrar un esquema sencillo de la prueba realizada.

```
import codes
import algorithms. Algorithms

# Read the codes to analyze
labels = ['qrcode', 'barcode']
codes = []
for label in labels:
    codes.append(codes.imread('codes/' + label + '.png'))

# Algorithm for each code
ind = 0
for code in codes:
    # CORNER DETECTION
    corner_code = Algorithms.corner_detection(code)
    codes.imwrite("keypoints/" + labels[ind] + "_corner.png", corner_code)
    # FAST
    fast_code = Algorithms.fast(code)
    codes.imwrite("keypoints/" + labels[ind] + "_fast.png", fast_code)
    # BRIEF
    brief_code = Algorithms.brief(code)
    codes.imwrite("keypoints/" + labels[ind] + "_brief.png", brief_code)
    # SURF
    surf_code = Algorithms.surf(code)
    codes.imwrite("keypoints/" + labels[ind] + "_surf.png", surf_code)
    # SIFT
    sift_code = Algorithms.sift(code)
    codes.imwrite("keypoints/" + labels[ind] + "_sift.png", sift_code)
    #ORB
    orb_code = Algorithms.orb(code)
    codes.imwrite("keypoints/" + labels[ind] + "_orb.png", orb_code)
    ind += 1
```

En este código se puede observar cómo los dos códigos se guardan en un directorio llamado *"codes"*, mientras que los códigos se guardaran en el directorio *"keypoints"*.

Además, para cada código (sentencia *for*), se ejecuta cada uno de los seis algoritmos para obtener los puntos de interés, los cuales son los puntos críticos para un buen rendimiento de la aplicación final.

5.8.1 RESULTADOS PARA LOS BARCODES

Primeramente, se va a analizar el funcionamiento de los algoritmos con los códigos de barras, para ello se organizan en tres figuras para poder observar bien los puntos de interés marcados por cada algoritmo.

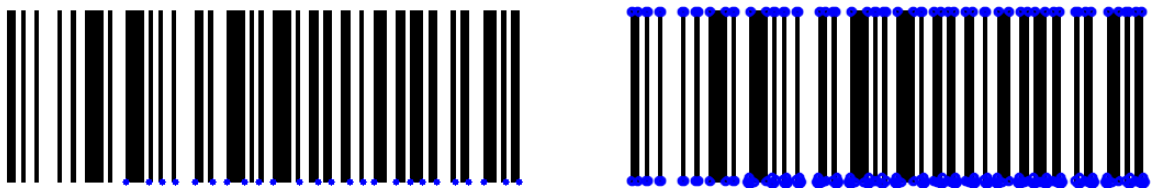

Figura 24. Puntos de interés con los algoritmos de Shi-Tomasi (izq.) y FAST (drcha..).

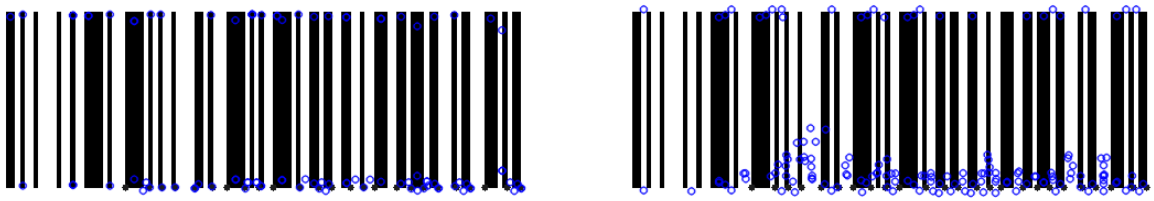

Figura 25. Puntos de interés con los algoritmos SIFT (izq.) y SURF (drcha.).

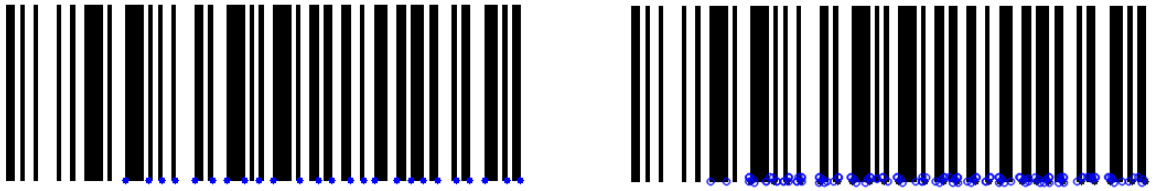

Figura 26. Puntos de interés con los algoritmos BRIEF (izq.) y ORB (drcha.).

Observando las distintas soluciones para los seis algoritmos probados se pueden sacar diferentes conclusiones:

- El algoritmo de detección de esquinas es el más débil, lo que sorprende también es que el algoritmo BRIEF tiene esa misma debilidad para este tipo de códigos.
- El algoritmo ORB, aun siendo el más potente y el más nuevo, tampoco es muy robusto para este tipo de códigos ya que hereda parte del potencial de BRIEF, por lo que, aunque se puede ver como saca un número más exacto de puntos, no es el más robusto.
- El algoritmo FAST si saca un buen número de puntos de interés a lo largo de las diferentes barras del código, lo que ocurre es que el tiempo de detección es excesivamente mayor ya que necesita muchos puntos de interés para tener buen rendimiento, lo que comúnmente se llama "matar moscas a cañonazos".
- Por último, quedan los algoritmos de SIFT y SURF, los cuales tienen puntos de interés a lo largo de todo el código, aunque SURF tiene mayor detalle, aunque sacrifica un poco la velocidad ya que es algo más lento en la detección.

En definitiva, con este tipo de código, los algoritmos SIFT y SURF son óptimos, mientras que ORB es un buen algoritmo, aunque puede que pueda dar algún tipo de falso positivo con las distintas detecciones.

5.8.2 RESULTADOS PARA LOS CÓDIGOS QR

Para afinar más la decisión final se realizará la misma casuística para los códigos bidimensionales o QR.

Por tanto, cogiendo como referencia el QR de la Figura 23 se pasará esa imagen por cada uno de los distintos algoritmos para obtener así la imagen con los puntos de interés sobreimpresos, y así, de esta manera poder realizar un análisis más definitivo y poder tomar una decisión final.

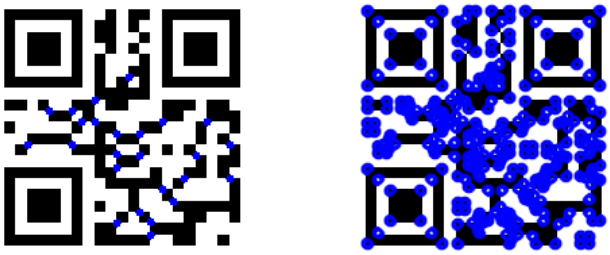

Figura 27. Puntos de interés con los algoritmos de Shi-Tomasi (izq.) y FAST (drcha..).

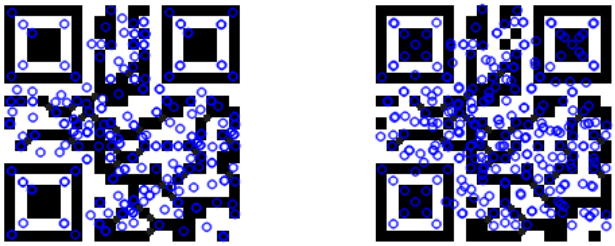

Figura 28. Puntos de interés con los algoritmos SIFT (izq.) y SURF (drcha.).

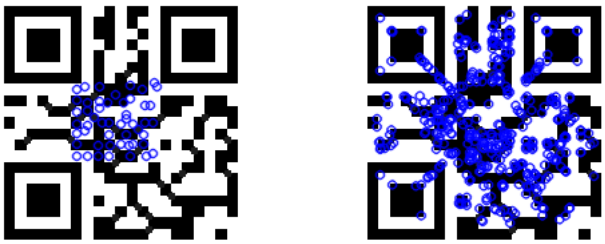

Figura 29. Puntos de interés con los algoritmos BRIEF (izq.) y ORB (drcha.).

A diferencia con los códigos de barras, el algoritmo ORB es mucho más robusto y probablemente el más eficiente para estos códigos.

BRIEF sigue siendo poco robusto para este tipo de códigos, aunque si se observa la imagen de Lenna para reconocimiento de objetos si es potente (véase Figura 22).

Por otro lado, tanto los algoritmos de SURF y SIFT para los códigos QR si dan buen resultado en cuanto a robustez, mientras que Shi-Tomasi y FAST les ocurre lo mismo que con el barcode, el primero es muy débil y el segundo es ineficiente con este tipo de códigos.

5.8.3 DECISIÓN FINAL

Para la decisión final se ha utilizado distintas imágenes degradas de las etiquetas originales (véase Figura 23) para ver el comportamiento de los distintos algoritmos con diferentes cambios de luz, pequeñas rotaciones, o pequeñas vibraciones humanas a la hora de capturar el código. Para ello, se utilizarán hasta 6 variantes originales, metiéndole a la imagen diferentes contrastes, brillos, y pequeños ruidos gaussianos que distorsionen ligeramente la imagen para simular la realidad[5].

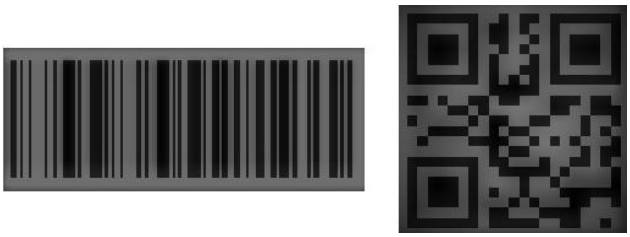

Figura 30. Código de barras y código QR del experimento degradados

Además, se toman hasta 9 etiquetas distintas para ver si los algoritmos son capaces de conseguir acertar la etiqueta de "robot 1". Las etiquetas utilizadas son:

- "robot 1" (que es la etiqueta correcta).
- "robot 2".
- "robot 3".
- "cámara 1".
- "cámara 2".
- "cámara 3".
- "ordenador 1".
- "ordenador 2".
- "ordenador 3".

[5] Se ha realizado una simulación de cada uno de los algoritmos ya que es inviable realizar una simulación real con cada uno de los algoritmos, ya que no se puede producir seis diferentes variantes de la solución final en el tiempo que se dispone.

Con estas 9 etiquetas guardadas en base de datos, se capturan las 6 variantes de etiquetas degradas para saber la tasa de éxito de cada uno de los algoritmos.

A estas 6 variantes se capturan hasta 3 veces ya que se rotan aleatoriamente entre 8-10 grados, por tanto, se capturan 18 variaciones de cada una de las dos etiquetas dando los siguientes resultados:

	Barcode	**QR Code**	**Lenna**	**Time (s)[6]**
Shi-Tomasi	44%	44%	56%	0.18
FAST	69%	81%	81%	0.13
SIFT	**87%**	87%	**94%**	0.22
SURF	**87%**	94%	87%	0.08
BRIEF	62%	84%	81%	**0.05**
ORB	69%	**100%**	**94%**	**0.05**

Tabla 3. Resultados del experimento

Observando los resultados, se escoge el algoritmo SURF frente a SIFT ya que, aunque tengan resultados muy similares es más rápido en la detección.

También se escoge SURF frente a ORB ya que, aunque para QR tenga un acierto del 100%, como se vio anteriormente, en el código de barras no es tan robusto.

Por tanto, para el sistema que se desarrolle se utilizara el algoritmo de *Speeded Up Robust Features,* SURF, desarrollado por Herbert Bay Tinne Tytelaars y Luc Van Gool.

[6] Tiempo aproximado, ya que para cada algoritmo se pone un flag de tiempo antes y después del cálculo, lo que ocurre es que la programación de Python puede meter un delay en cada sentencia código,

Capítulo 6. SISTEMA DESARROLLADO

Una vez realizada la parte de investigación del proyecto, se va a realizar un pequeño sistema para demostrar el funcionamiento de estos algoritmos.

Se ha escogido el algoritmo *Speeded Up Robust Features* (SURF) como algoritmo de identificación de las etiquetas, las cuales serán código de barras para una administración más sencilla por parte de los administradores del sistema.

Por otro lado, el sistema desarrollado tendrá dos modos de desarrollo diferentes dependiendo del usuario, como se vio en los objetivos:

- Modo mantenimiento, en el que se puedan añadir etiquetas nuevas, quitarlas, editar ciertos parámetros, etc. Este modo debe existir como seguimiento de la minifábrica ya que la planta puede crecer en cuanto a número de máquinas, por ejemplo, y se quiere un sistema final que sea totalmente escalable al tamaño futuro de la minifábrica.
- Modo operación, en donde se realizará la lectura de la etiqueta, reconocimiento de la máquina y, por tanto, la parte de gestión de la planta de la minifábrica. En este modo entrará en juego toda la operativa de gestión de la planta y lo que se quiera implementar en otros proyectos futuros.

Además, aparte de la utilización del algoritmo de reconocimiento, se utilizará como sistemas de almacenamiento:

- Base de datos de acuerdo con los requisitos necesarios para que pueda estar integrada en el sistema de la Escuela, si no fuese posible o necesario, se optimizará de acuerdo con unos requisitos que se plantearían.
- Repositorio Cloud de archivos tipo Google Drive para almacenar los distintos archivos que mostrar en la aplicación.

- Interfaz entre la solución móvil y la base de datos y el repositorio Cloud, para ello se realizarán distintas implementaciones para facilitar el acceso.

6.1 ANÁLISIS DEL SISTEMA

Como se ha comentado anteriormente, el sistema desarrollado tendrá dos modos de desarrollo diferentes dependiendo del usuario que esté utilizando la aplicación, para ello deberá existir la posibilidad de una autenticación en la página inicial para entrar y luego una autenticación para entrar en modo administrador.

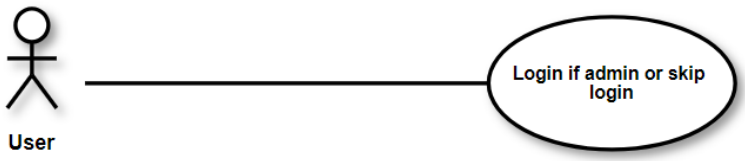

Figura 31. Caso de uso inicial

Existen dos posibilidades:

- Una primera, en donde la página de inicio sea una propia autenticación, y en donde la persona se pueda autenticarse o simplemente saltarse dicha autenticación siendo así en un usuario mientras que si se autenticase pasaría a ser administrador.
- Otra segunda opción sería que la aplicación directamente estuviese en modo operación (directamente con la cámara donde se realizará la lectura de la etiqueta, reconocimiento de la máquina y, por tanto, la parte de gestión de la planta de la minifábrica, etc.), y hubiese una opción de menú en donde el usuario pudiese realizar la autenticación si quisiese.

En definitiva, el primer caso de uso refleja la problemática en discernir si una persona que utilice la aplicación se le considera un simple usuario en donde puede consultar todos los recursos en modo operación, o se trata del administrador que entraría en modo mantenimiento en donde podría realizar cambios en la herramienta. Finalmente se optará a tener dos autenticaciones, una inicial y otra de administrador.

El segundo caso de uso se puede definir como la operativa de un usuario (no administrador). Este usuario tiene los permisos de lectura y de visualización de la herramienta, pero en ninguno de los casos podrá realizar modificaciones en la aplicación.

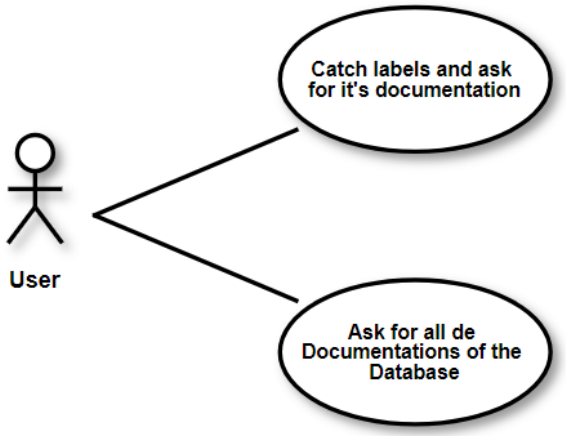

Figura 32. Caso de uso para un usuario (sin autenticación de administrador)

En este caso se visualizan dos funciones adquiridas por el usuario:

- En primer lugar, utilizará el modo operación, como ya se conoce, para la lectura de las etiquetas, en consecuente, el reconocimiento de cada una de las máquinas para así poder pedir la información relacionada con esa máquina.
- Como segunda opción, el usuario será capaz de ver el registro de todos documentos alojados sobre la minifábrica de una manera ordenada para el usuario, con esta funcionalidad no necesitará estar físicamente al lado de la máquina en cuestión ya que de esta manera puede consultar su información remotamente.

Por tanto, este segundo caso refleja la actividad posible que puede realizar el usuario en la solución, que principalmente se basa en permisos de lectura, es decir, puede realizar toda la operativa de la herramienta, pero sin poder modificar ninguno de los parámetros o recursos de la aplicación.

Por último, se concibe el caso de uso del administrador, el cual tendrá un grado mayor de permisos que el usuario, este administrador está pensado para profesores o responsables de un laboratorio, por un lado, y por otro, para gerentes y personal autorizado de una fábrica.

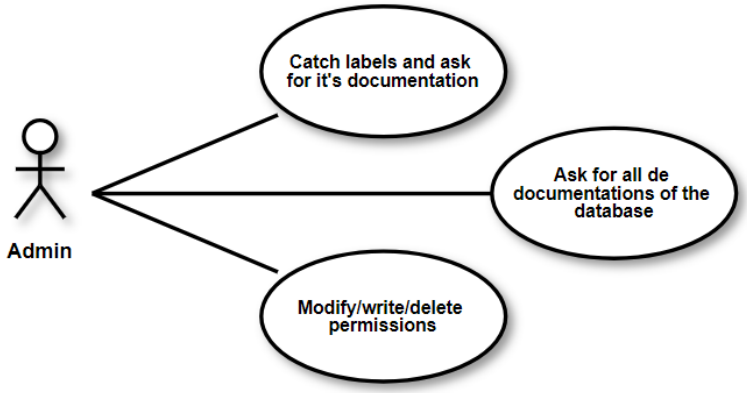

Figura 33. Caso de uso para un administrador

Este tercer caso de uso presenta tres actividades para el administrador:

- En primer lugar, podrá realizar las mismas dos tareas que el usuario estándar, es decir, utilizará el modo operación, como ya se conoce, para la lectura de las etiquetas y así poder obtener la información del dispositivo, y, por otro lado, será capaz de ver el registro de todos documentos alojados sobre la minifábrica de una manera ordenada.
- En cambio, al administrador se le añade un tercer caso, que consiste en los permisos de modificación, creación o borrado, es decir, el administrador podrá modificar la información de cualquier dispositivo (su nombre, sus características, o cualquier parámetro), podrá añadir nuevos documentos relacionados o borrar los existentes, y por último podrá borrar dispositivos que ya no se encuentren en la fábrica/laboratorio para así tener actualizado la lista de dispositivos.

6.2 DISEÑO TÉCNICO

En este segundo apartado se trata el análisis del sistema de una manera más técnica, para ello, se han utilizado los diagramas de actividad y de secuencia para mostrar el pleno comportamiento de la herramienta.

En el diagrama de actividad se muestra el flujo descrito en los casos de uso anteriores, en este diagrama se visualiza de una manera clara y breve como se tratan las partes interesadas.

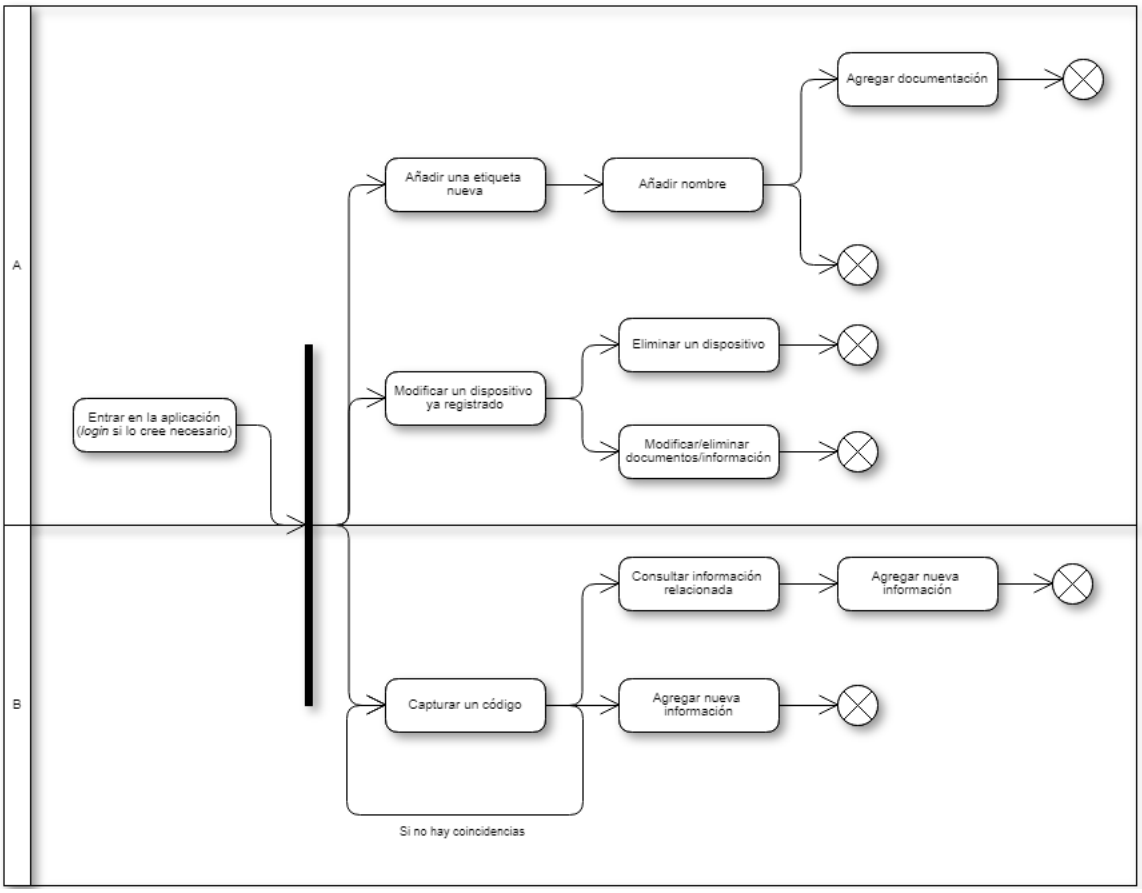

Figura 34. Diagrama de actividad o de flujo

El diagrama se ha dividido en dos partes, A y B. En la primera parte se la entrada a la herramienta, con una autenticación de usuario. Según si se identifica más adelante como administrador, o no, podrá acceder a la parte B únicamente o a la parte A y B.

- La parte A se identifica como la parte de la aplicación que únicamente es para el administrador, entonces, cuando se registra tiene la posibilidad por ejemplo de añadir nuevas etiquetas para nuevos dispositivos con la operativa que conlleva, o la modificación de los datos.
- En la parte B puede trabajar cualquiera de los dos perfiles, tanto el usuario como el administrador, esta parte se trata del modo de operación en donde la persona podrá realizar la identificación de cada dispositivo por su código, o en su defecto, podrá consultar todos los documentos.

Adicionalmente, se ha definido el diagrama de secuencia para poder explicar las interacciones que ocurren dentro del sistema.

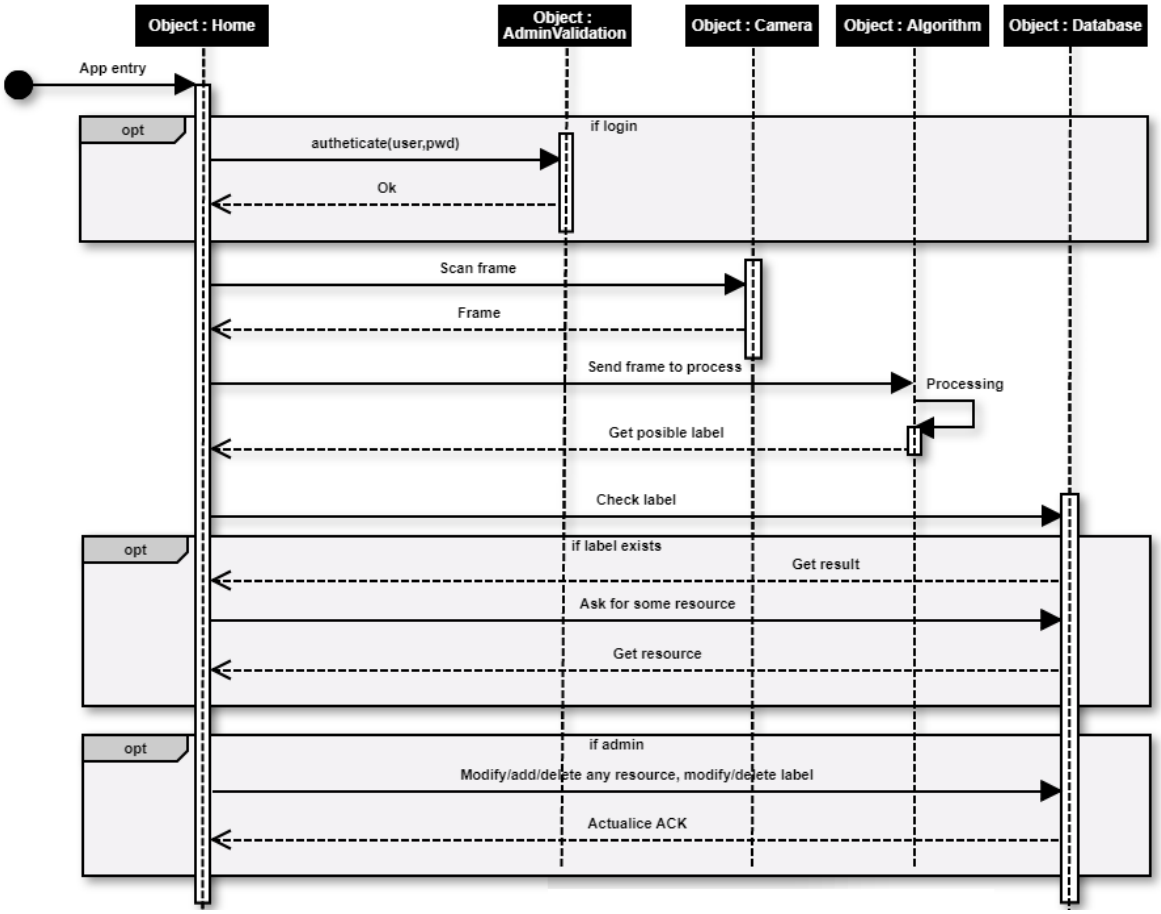

Figura 35. Diagrama de secuencia

Observando el diagrama de la página anterior, se definen cinco objetos dentro del sistema:

- Objeto principal.
- Validación de administrador.
- Cámara.
- Algoritmo de reconocimiento.
- Base de datos/repositorio.

Este tipo de diagrama se centra en las líneas de vida de cada objeto y como se relacionan entre sí los distintos objetos del sistema. Como se puede observar, cualquier persona que utilice la herramienta estará situado en el objeto principal e interaccionará directamente con los distintos objetos.

Por ejemplo, a la hora de validar el usuario para ser administrador, el objeto principal se comunicará con el validador mediante el método de autenticación el cual le pasará las credenciales (usuario, contraseña), y éste devolverá al objeto principal si la autenticación ha sido exitosa o no mediante un token.

Otra de las operativas sería a la hora de detectar los dispositivos, en este caso el objeto principal estará enteramente ligado a la cámara (de hecho, desde el punto de la programación, en ese punto serían el mismo objeto), la cual escaneara a tiempo real el entorno en busca de reconocer una de las etiquetas y le mandará la respuesta al objeto principal, y éste se lo enviará en tiempo real el frame escaneado al algoritmo. Este objeto algoritmo lo procesará y buscará si hay evidencias de una etiqueta y le devolverá el resultado al objeto principal (por eso es crítico elegir un algoritmo que sea realmente rápido).

En definitiva, mediante el diagrama de secuencia se puede saber cómo está estructurada la programación de la solución, la cual será desgranada en el siguiente capítulo de desarrollo e implementación en donde se mostrará ilustrativamente el funcionamiento final de la aplicación.

Por último, para apoyar las distintas actividades de la solución, se necesitarán diferentes clases, las cuales se representan en el siguiente diagrama de clases.

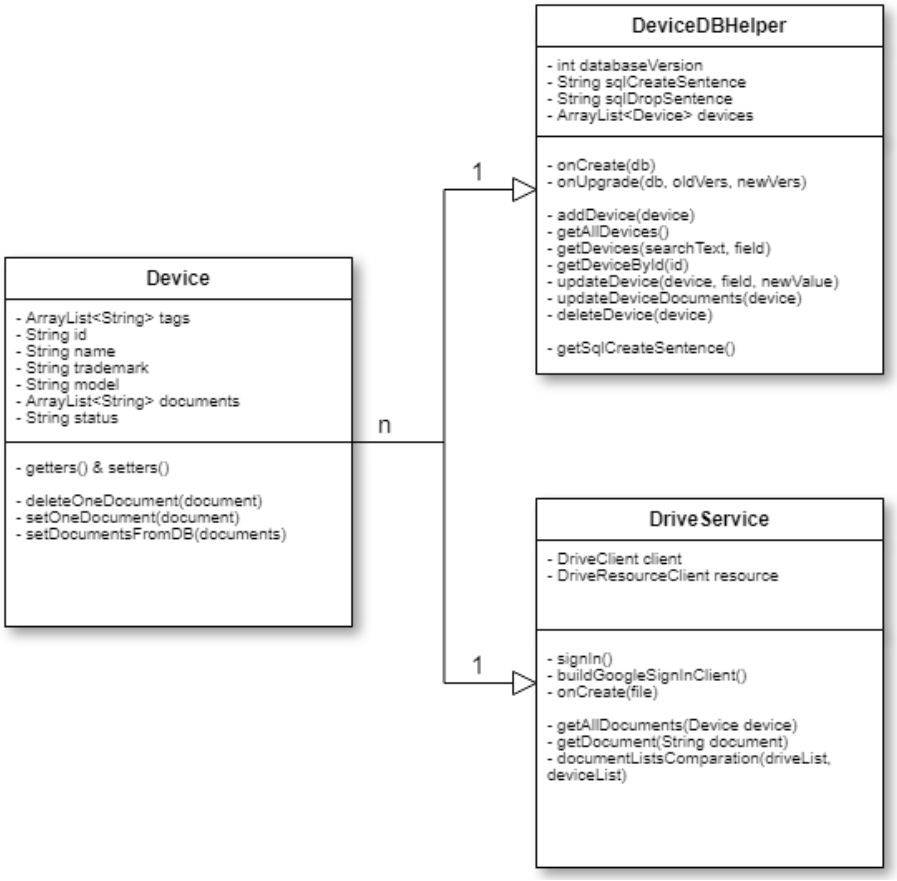

Figura 36. Diagrama de clases

Como se puede observar en la figura anterior, se utilizarán tres clases, una para definir los dispositivos, en la cual existirán numerosos objetos. Por otro lado, se definen las clases supletorias de conexión a base de datos y conexión a repositorio, las cuales únicamente contaran con un único objeto. Dichas clases se relacionan en cuanto a que cada uno de los objetos *dispositivo* contarán con el mismo objeto *DBHelper* o *Drive* para realizar conexiones a la base de datos o al repositorio.

Se debe aclarar, que **al uso no se trata de una relación propia** de Java, pero, para ilustrar el motivo por el que se han creado las clases se establece dicha relación.

Capítulo 7. DESARROLLO E IMPLEMENTACIÓN

Teniendo en cuenta el Capítulo 6. , se desarrolla un pequeño desarrollo para poner en práctica la investigación realizada. En la siguiente figura se muestra el esquema del desarrollo realizado.

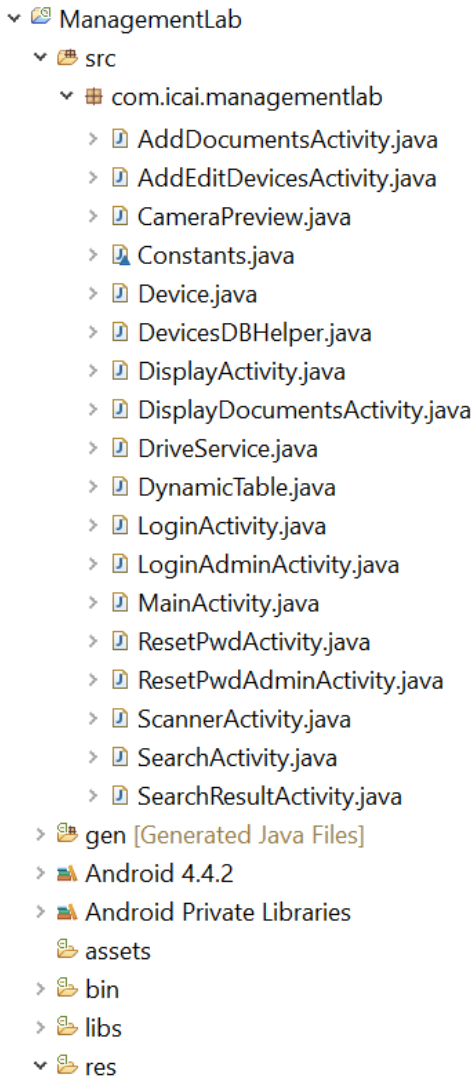

Figura 37. Directorio de la aplicación

7.1 ACTIVIDADES DESARROLLADAS

Para mostrar el desarrollo de la aplicación se va a explicar cada una de las clases (o actividades en Android) realizadas.

1. ***Login Activity & Login Admin Activity***: Actividades utilizadas para el logueo, la actividad *Login Activity* solo se utiliza para entrar a la App mientras que la actividad *Login Admin Activity* se utiliza para realizar tareas de administrador. Por tanto, existen credenciales de usuario y administrador.

```
if(!user.isEmpty() && !password.isEmpty()) {
	if(user.compareTo(user_cred) == 0 && password.compareTo(pwd_cred) == 0) {
		Toast.makeText("Detection mode enabled", Toast.LENGTH_LONG).show();
		intent = new Intent(getApplicationContext(), MainActivity.class);
		intent.putExtra(SELECTED_MODE, Constants.detection_mode);
		startActivity(intent);
	}
	else {
		Toast.makeText("Login failed", Toast.LENGTH_LONG).show();
		clearFormData();
	}
}
```

Por tanto, la aplicación se inicia con la petición de credenciales al usuario siempre.

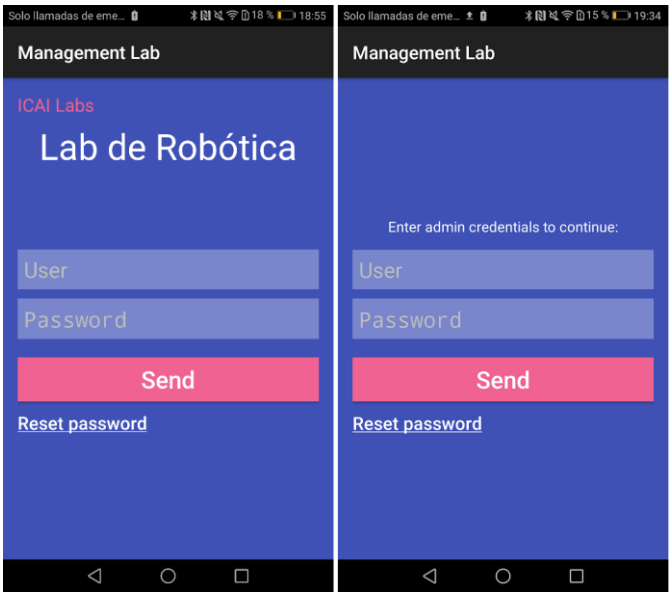

Figura 38. Pantallas de autenticación de usuario y administrador

2. ***Main Activity***: Actividad principal, en donde se puede elegir realizar un escaneo, o en el menú elegir alguna de las opciones disponibles: Añadir dispositivo, realizar una búsqueda o cerrar sesión.

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
        super.onActivityResult(requestCode, resultCode, data);
        if(resultCode == 100){
              Intent intent = new Intent(this, DisplayActivity.class);
              String scan_result = (String) data.getExtras().get("result");
              intent.putExtra(DEVICE_ID, scan_result);
              intent.putExtra(SELECTED_MODE, selected_mode);
              startActivity(intent);
         }
    }
```

Por tanto, esa doble función quedaría de la siguiente manera, en donde se tienen dos botones para elegir una de las dos funcionalidades.

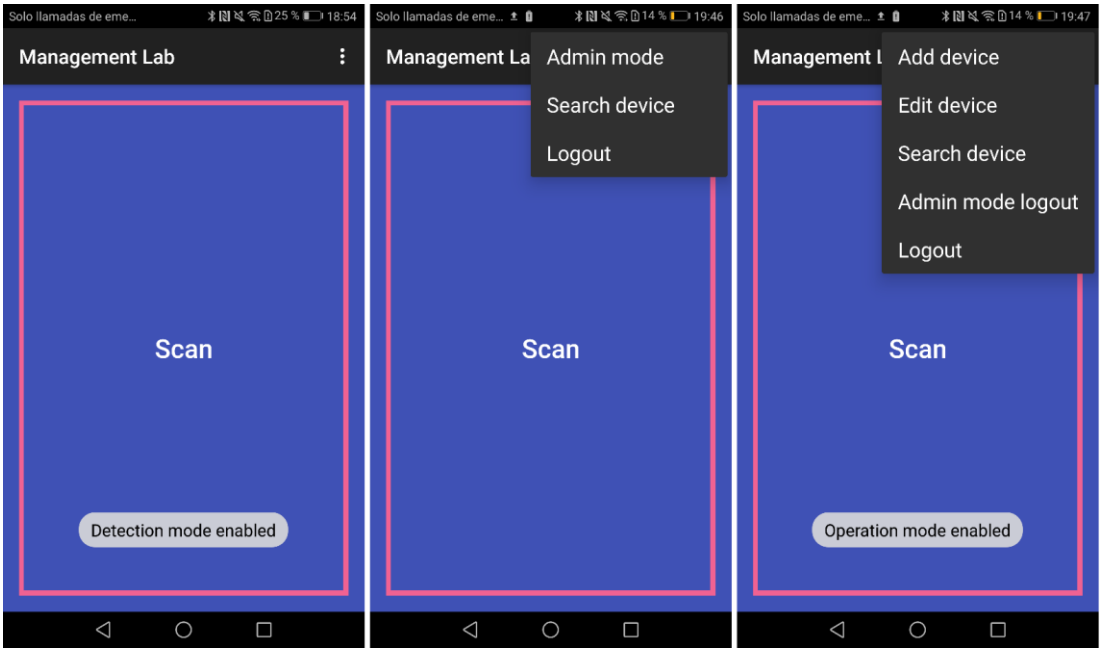

Figura 39. Pantallas principales de la aplicación

3. ***Scanner Activity***: En esta segunda actividad es donde entra en juego el resultado de este proyecto. Como se concluyó en la 5.8.3, el algoritmo escogido es el algoritmo SURF frente a SIFT ya que, aunque tengan resultados muy similares es más rápido en la detección, y frente a ORB debido a que tiene fallos con los *barcode*.

   Esta actividad es la parte principal del proyecto, ya que aquí se establecen todos los parámetros y variables para hacer funcionar SURF adecuadamente. Para este proyecto se ha optimizado el algoritmo para dos tipos de etiquetas, el código de barras y el QR. Esta actividad tiene dependencias con las clases *CamaraPreview* y *ScanAlgorithmHelper*.

   Para establecer los parámetros ideales se tomaron hasta un total de 150 escaneos de manera programática, estos escaneos se realizaron básicamente para afinar:

   - Los puntos de interés deben ser los justos para que el tiempo sea mínimo, ya que el tiempo de procesamiento por hardware será la parte más importante.
   - La luz formará parte importante del código, por lo que se ha decidido antes de utilizar el algoritmo SURF realizar un pequeño preprocesado de la imagen la cual eliminará puntos de luz y convertirá la imagen a escala de grises.

   También se han ajustado otros parámetros como la rotación, los autovalores, y parámetros matemáticos.

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_scanner);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        autoFocusHandler = new Handler();
        camera = getCameraInstance();
        scanner = new ImageScanner();
        scanner.setConfig(0, Config.X_DENSITY, 3);
        scanner.setConfig(0, Config.Y_DENSITY, 3);
        cameraPreview = new CameraPreview(this, camera, previewCb, autoFocusCb);
        FrameLayout preview = (FrameLayout)findViewById(R.id.cameraPreview);
        preview.addView(cameraPreview);
```

Como se puede observar en la figura anterior, la función *onCreate()* inicializa la cámara, tanto la clase *ImageScanner()* donde se establece el algoritmo SURF, como la función *getCameraInstance()* y la clase *CamaraPreview()* en donde se establece las características principales de una cámara.

En la siguiente figura se muestra una captura de la actividad de escaneo. Además, se debe indicar que existen dos modos de captura, un modo captura único, en el que el usuario capta la imagen y el algoritmo procesa dicha captura y un modo de captura continuo en donde el algoritmo consulta la base de datos continuamente.

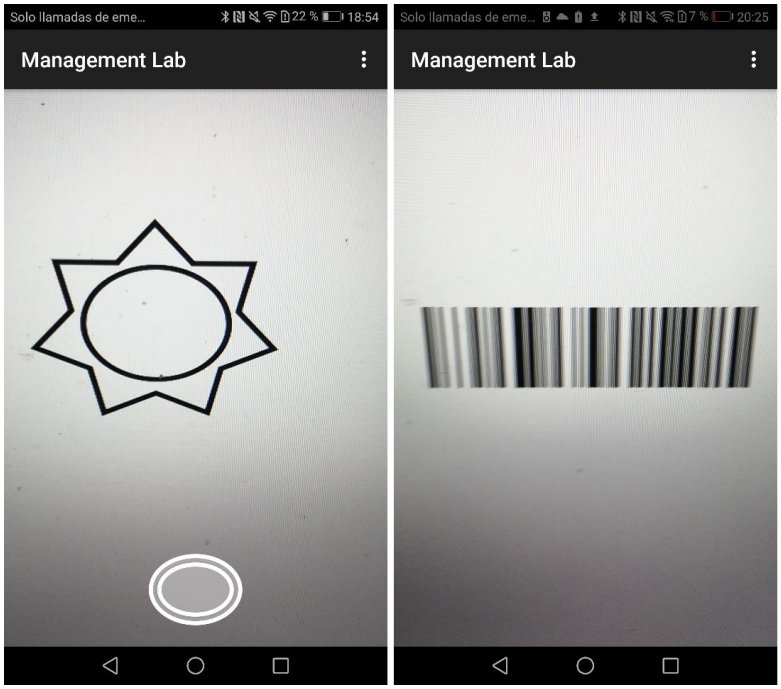

Figura 40. Pantallas de escaneo

En este caso se trata de un código en forma de estrella el cual se va a capturar en modo único, por lo que se puede observar que existe un botón traslucido para obtener dicha imagen. Por otro lado, en el modo de captura continua dicho botón desaparece y las imágenes obtenidas se procesan en tiempo real.

3.1. Camara Preview: Para la resolución del escaneo se debe preparar la cámara y establecer los permisos y las características del funcionamiento de la cámara.

```
public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback
{
    private SurfaceHolder surfHolder;
    private Camera camera;
    private PreviewCallback;
    private AutoFocusCallback autoFocusCallback;

    public CameraPreview(Context context, Camera camera,
                         PreviewCallback previewCb,
                         AutoFocusCallback autoFocusCb) {
        super(context);
        this.camera = camera;
        previewCallback = previewCb;
        autoFocusCallback = autoFocusCb;

        surfHolder = getHolder();
        surfHolder.addCallback(this);
        surfHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }
```

A su vez, esta clase tiene distintos métodos, el más importante es *surfaceChanged()* el cual se utiliza en casos de que haya cambios de tamaños de imagen ya que se ha habilitado el autofocus de la cámara.

3.2. Scan Algorithm Helper: Una vez que se obtiene la captura, se debe tomar los puntos de interés, y, la actividad conecta con la base de datos existente (*DBDevices*) y compara con los puntos de interés obtenidos cuando se carga una nueva etiqueta. Para ello, se utiliza esta clase que resuelve estos

Una vez que se obtiene el resultado, la actividad devuelve el valor obtenido a la actividad principal, *MainActivity.java* para que entre en juego una nueva actividad que muestre los datos del dispositivo escaneado.

En la actividad principal se tiene la función *onActivityResult()* en donde se espera que la actividad de escaneo devuelva un código de respuesta y un objeto *Intent* [22] el cual llevará como parámetro el identificador del dispositivo obtenido en el escaneo.

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == 100){
        Intent = new Intent(getApplicationContext(), DisplayActivity.class);
        String scanContent = (String) data.getExtras().get("result");
        intent.putExtra(SCAN_CONTENT, scanContent);
              startActivity(intent);
    }
}
```

Por tanto, observando el código, mediante a variable *scan* se obtiene el valor *result* obtenido de la aplicación del algoritmo SURF, y este resultado se pasa a la actividad de Display.

Una vez que se tiene esa variable se llama a otra clase (*DeviceDBHelper*) para obtener de la base de datos los datos que se tienen del dispositivo.

4. ***Display Activity***: En esta nueva actividad se mostrarán los datos obtenidos de la etiqueta resultante. Analizando el código se puede observar la función *getSelectedMode ()* la cual es una función del modo que esté el usuario se tienen los métodos de lectura y escritura de la base de datos de las etiquetas.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_display);
       setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
       // Instance DB
       db = new DevicesDBHelper(this);
       // Get mode app, detection or operation mode
       selected_mode = getSelectedMode();
       Intent = getIntent();
       String scan = intent.getStringExtra(MainActivity.SCAN_CONTENT);
    if(!scan.isEmpty()) {
       device = db.getDeviceById(device_search_id);
       ...
    }
}
```

En este caso, el resultado quedaría como se muestra en la siguiente figura, en donde, además, según si el usuario está con credenciales de administrador, o no, podrá realizar operaciones operativas o simplemente visualizar.

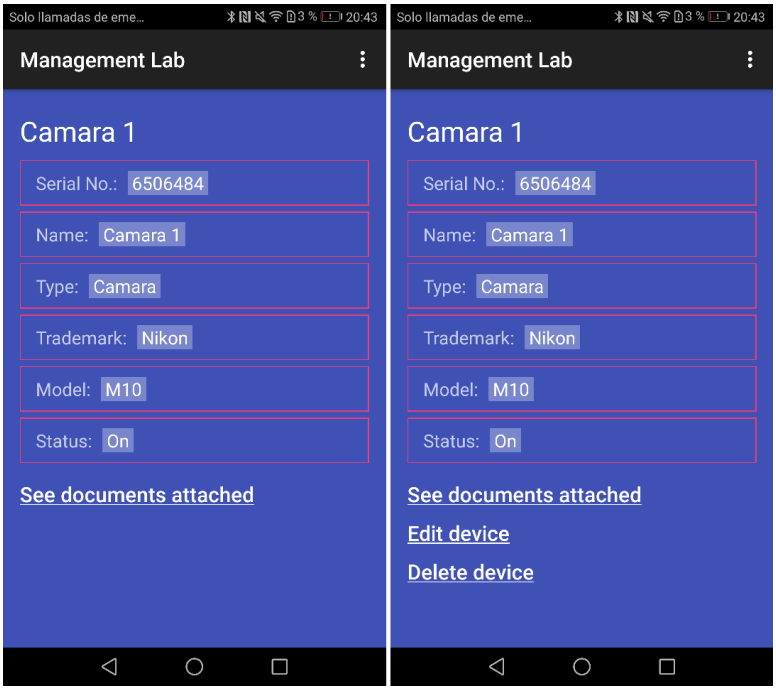

Figura 41. Resultado de la obtención de una etiqueta según el modo

Por tanto, en esta figura se puede observar los datos pertenecientes a este dispositivo, el cual se trata de una cámara. Y en el caso de tener el modo administrador, se podría editar el dispositivo o eliminarlo.

5. ***Search Activity***: Esta actividad se utiliza para la búsqueda manual de etiquetas, lo cual es útil cuando no se está propiamente en el laboratorio, o simplemente se quiere buscar un dispositivo sin tener que escanearlo.

   Para esta operativa se habilita un botón en el menú de cada una de las actividades el cual se utiliza para realizar la búsqueda y, al tratarse de una función de usuario, estará accesible para cualquier modo de operación, ya sea el modo de detección o el modo de administrador.

   Se podrán realizar búsquedas sobre el nombre o el tipo de dispositivos. Adicionalmente, se podrán mostrar todos los dispositivos si se escribe la *wild-card* "*".

Una vez se llega a la página de búsqueda, se pueden observar diferentes elementos:

- Un campo de texto el cual será el valor que buscar.
- Un *dropdown* en donde se podrá elegir si se busca por nombre o por tipo de dispositivo, por ejemplo, el nombre "Ordenador" o el tipo "Cámara".
- El botón que inicia la búsqueda.

El aspecto de la actividad y el *dropdown* quedarían de la siguiente manera:

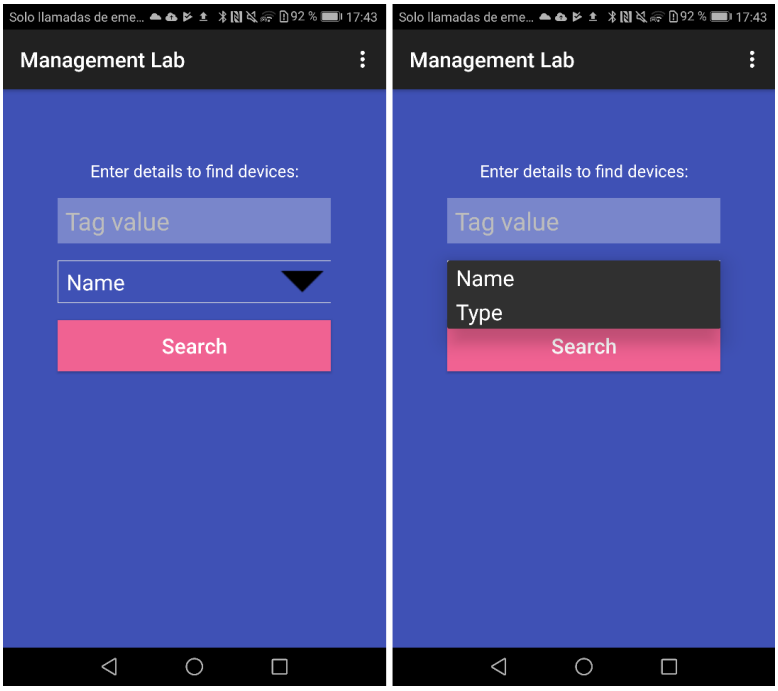

Figura 42. Actividad de búsqueda según nombre o tipo de dispositivo

El resultado obtenido es exactamente igual al del caso de la detección de una etiqueta, de hecho, son actividades gemelas, solo que el nombre es distinto y por tanto las llamadas desde la actividad de *Search* cambian:

```
    public void onClick(View v){
        if(v.getId()==R.id.search_button){
                    Intent intent = new Intent(this, SearchResultActivity.class);
                    String sch_text = findViewById(R.id.searchtxt)).getText();
                if(!search_text.isEmpty()) {
                    String sch_type = findViewById(R.id.type)).getSelectedItem();
                    intent.putExtra(SEARCH_TEXT, sch_text);
```

```
                    intent.putExtra(SEARCH_TYPE, sch_type);
                    intent.putExtra(SELECTED_MODE, selected_mode);
                    intent.putExtra(EDIT_MODE, edit_mode);
                    startActivity(intent);
                }
            }
        }
```

Al utilizar un *Intent* nuevo, se realiza una llamada a la actividad *SearchResultActivity* para mostrar los datos, en caso de que no obtenga ningún resultado en la base de datos mostrará que no obtuvo ningún resultado.

6. ***Search Result Activity***: Actividad en donde se muestran los resultados obtenidos en una tabla. Se podrá escoger una de las opciones obtenida para obtener los detalles del dispositivo escogido.

   Además, como se comentó anteriormente, si se utiliza el código de wild-card, es decir, el código *"*"*, la aplicación mostrará todos los dispositivos como se puede ver en la siguiente imagen.

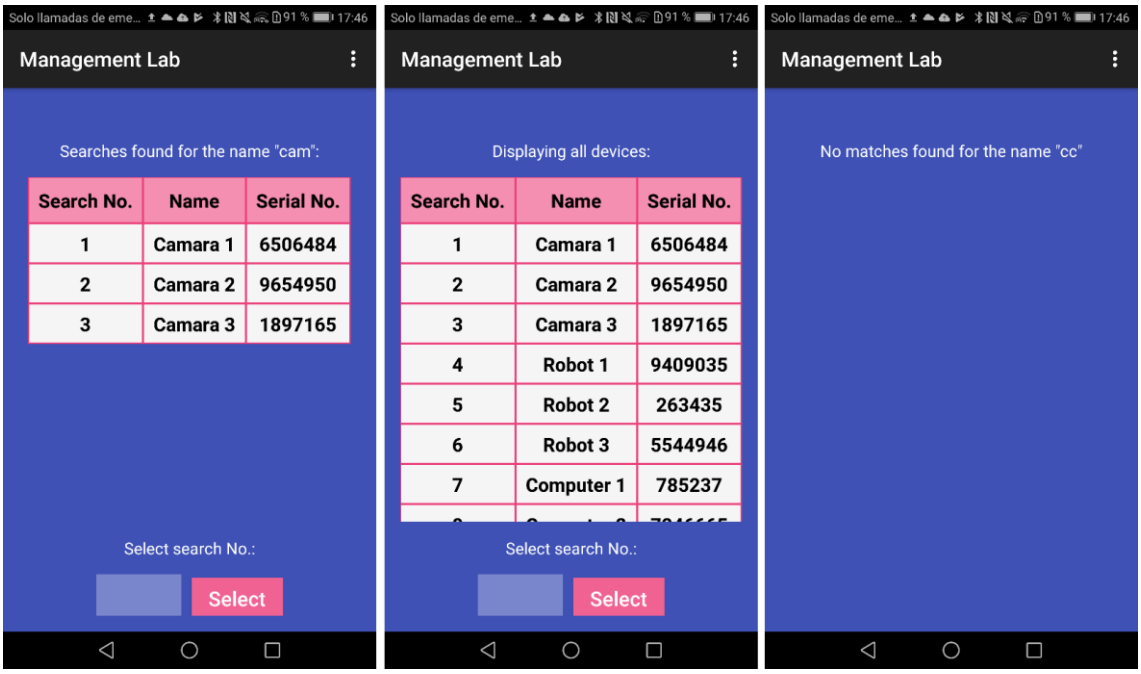

Figura 43. Resultado de la búsqueda

Si se observa la figura anterior, se puede ver tres opciones diferentes, en la primera, se obtiene cuando se busca tanto por *nombre* o *tipo* alguna cadena, en ese caso *"cam"* y se obtiene en este caso los posibles resultados, en este caso las cámaras. Otro caso es la utilización de la *wild-card*, y se observa que en la tabla se muestran todos los dispositivos. Y, por último, se puede ver como la aplicación muestra en el caso de no encontrar ningún dispositivo, en este ejemplo cuando se busca la cadena *"cc"*.

7. ***Add Edit Activity***: Actividad en la que se añade un dispositivo nuevo o se edita un dispositivo ya existente. Esta actividad es llamada desde otras actividades y dependiendo del origen se tratará de añadir un dispositivo o editarlo.

   Esta actividad, únicamente se puede acceder cuando se está en modo administrador, ya que se trata de una actividad de operación.

   La manera de acceder a esta actividad es mediante el menú desplegable existente en la aplicación en el caso de añadir un dispositivo, o en el caso de editar habrá distintas maneras de activar la actividad de edición.

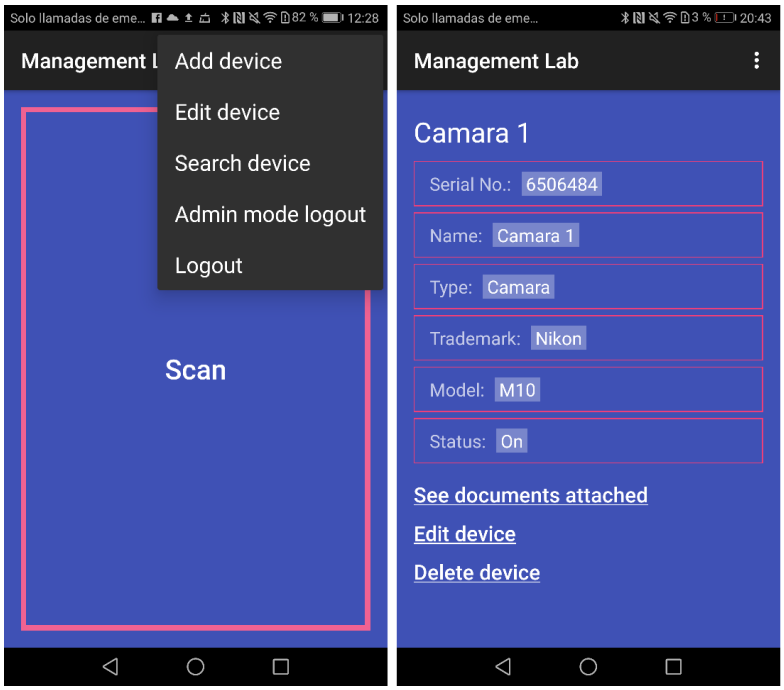

Figura 44. Formas de acceder a la actividad de edición

Una vez en la actividad de añadir un dispositivo nuevo, el cuestionario que rellenar sería el que se muestra en la siguiente figura. Además, cabe mencionar que si se pone un *Serial No.* o un *Nombre* ya existente lo que ocurrirá es que no se podrá guardar el nuevo dispositivo.

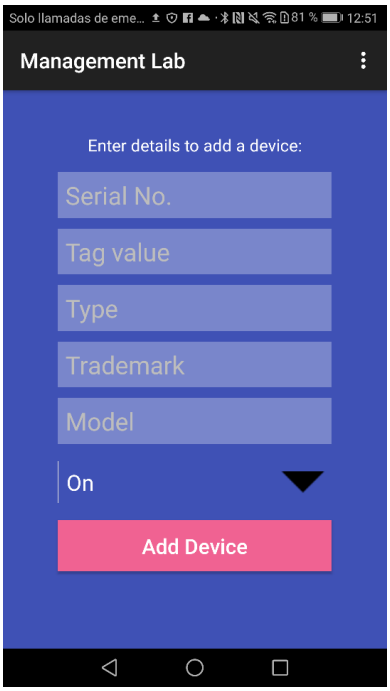

Figura 45. Formulario de añadir dispositivo

Una vez guardado el dispositivo, se deberá confirmar que se guarde y si se quiere añadir documentos al nuevo dispositivo o no, para ello se mostrarán diferentes *pop-ups* que preguntan al usuario sobre las distintas opciones a seguir, es decir, primeramente preguntará si quiere añadir el dispositivo, una vez que se acepte, se volverá a preguntar si se quiere añadir algún documento relacionado o no, y se añadirá un mensaje en donde se diga si se añade el dispositivo o no.

En el siguiente flujo de imágenes se puede ver el flujo de añadir un dispositivo, por ello se pone un nombre de dispositivo ya escogido para ver cómo reacciona la aplicación.

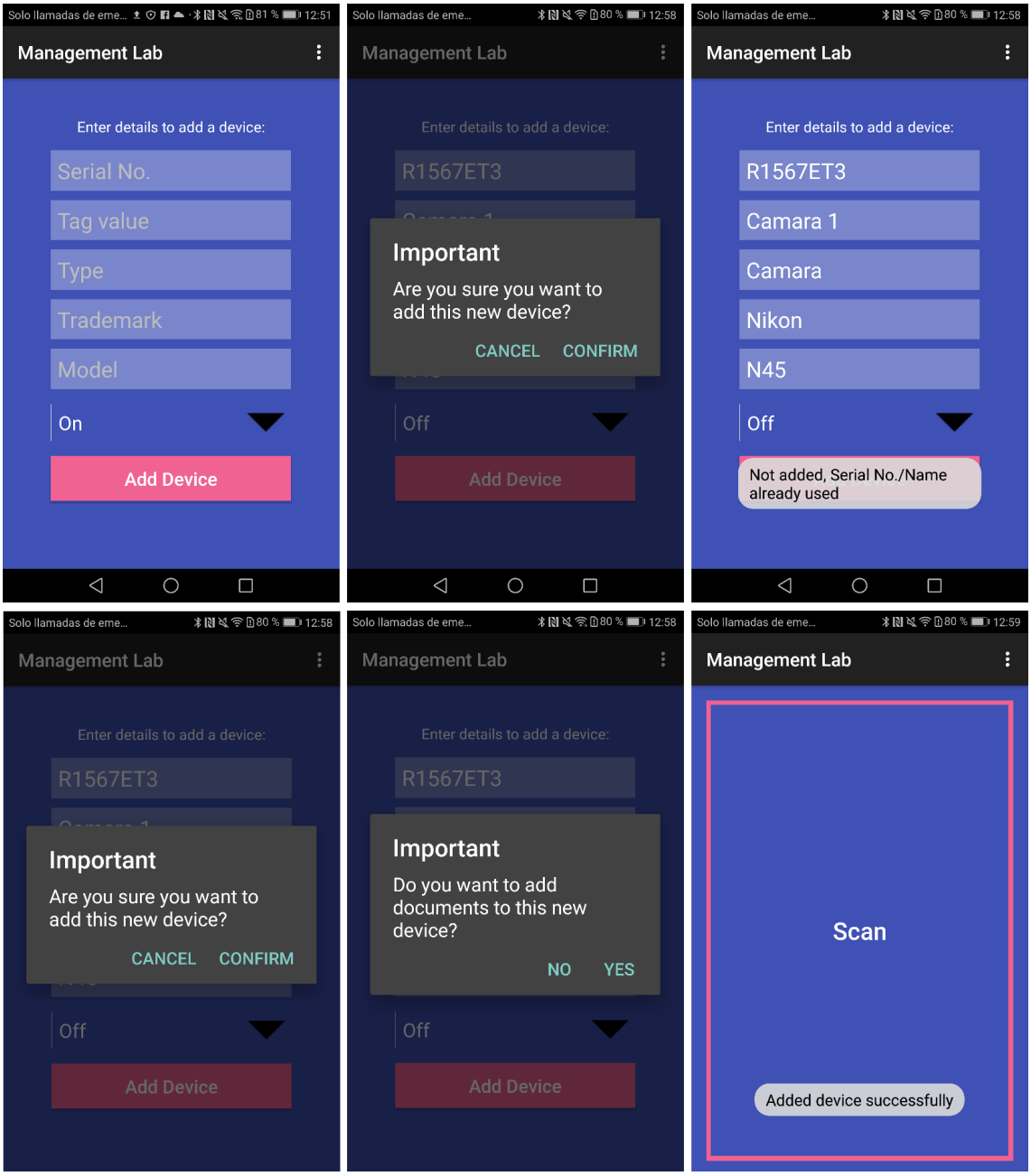

Figura 46. Flujo para añadir un dispositivo sin nuevos documentos

En la primera opción se ha puesto como nombre *"Camara 1"* la cual ya existe, por lo que la aplicación informa que no se puede añadir el dispositivo, mientras que en la tercera pantalla ya se establece el nombre como *"Camara 7"* y ya si se añade el dispositivo.

Por otro lado, teniendo en cuenta la subactividad de editar, el funcionamiento es similar, el formato de edición es el mismo, lo único que cambia es que los valores actuales están grabados en los campos de texto como *hint*, y el Serial No. no estará disponible para editar.

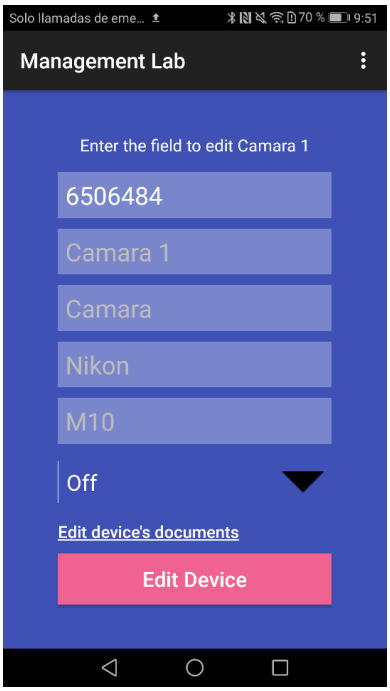

Figura 47. Formulario de editar un dispositivo

La manera de poder editar un dispositivo son dos maneras (ambas, únicamente habilitadas en modo administrador):

- La primera opción es directamente desde el menú desplegable, en donde se podrá realizar una búsqueda de un dispositivo para editarlo directamente.

- La segunda opción es en la página de la actividad Display, en cuyo caso, si se está en modo administrador, se podrá observar un link que te redirigirá al modo de edición. En caso de que el usuario no esté en modo administrador este link no existirá.

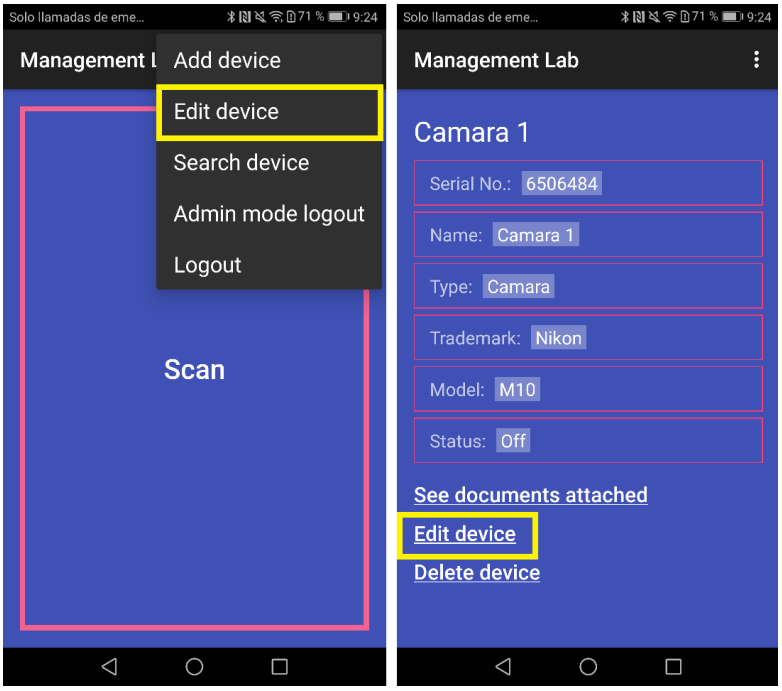

Figura 48. Modos para acceder a editar un dispositivo

Una vez se esté en el formulario de edición del dispositivo se deberá cambiar alguno de los campos visibles (excepto Serial No.) para que la aplicación detecte algún cambio en el objeto. En caso de que se haya cambiado algún dato se deberá confirmar que se guarde dicha actualización, para ello se mostrarán un *pop-up* que preguntará al usuario sobre si quiere guardar la edición o no, y se añadirá un mensaje en donde se diga si se añade el dispositivo o no.

Adicionalmente, como se puede ver en el formulario se puede editar los documentos que ya están relacionados con el dispositivo, en ese caso se activará una nueva actividad *Display Documents Activity*.

En el siguiente flujo de imágenes se puede ver el flujo de editar un dispositivo, por ello se pone otro nombre de dispositivo ya escogido para ver cómo reacciona la aplicación, si deja editarlo o no.

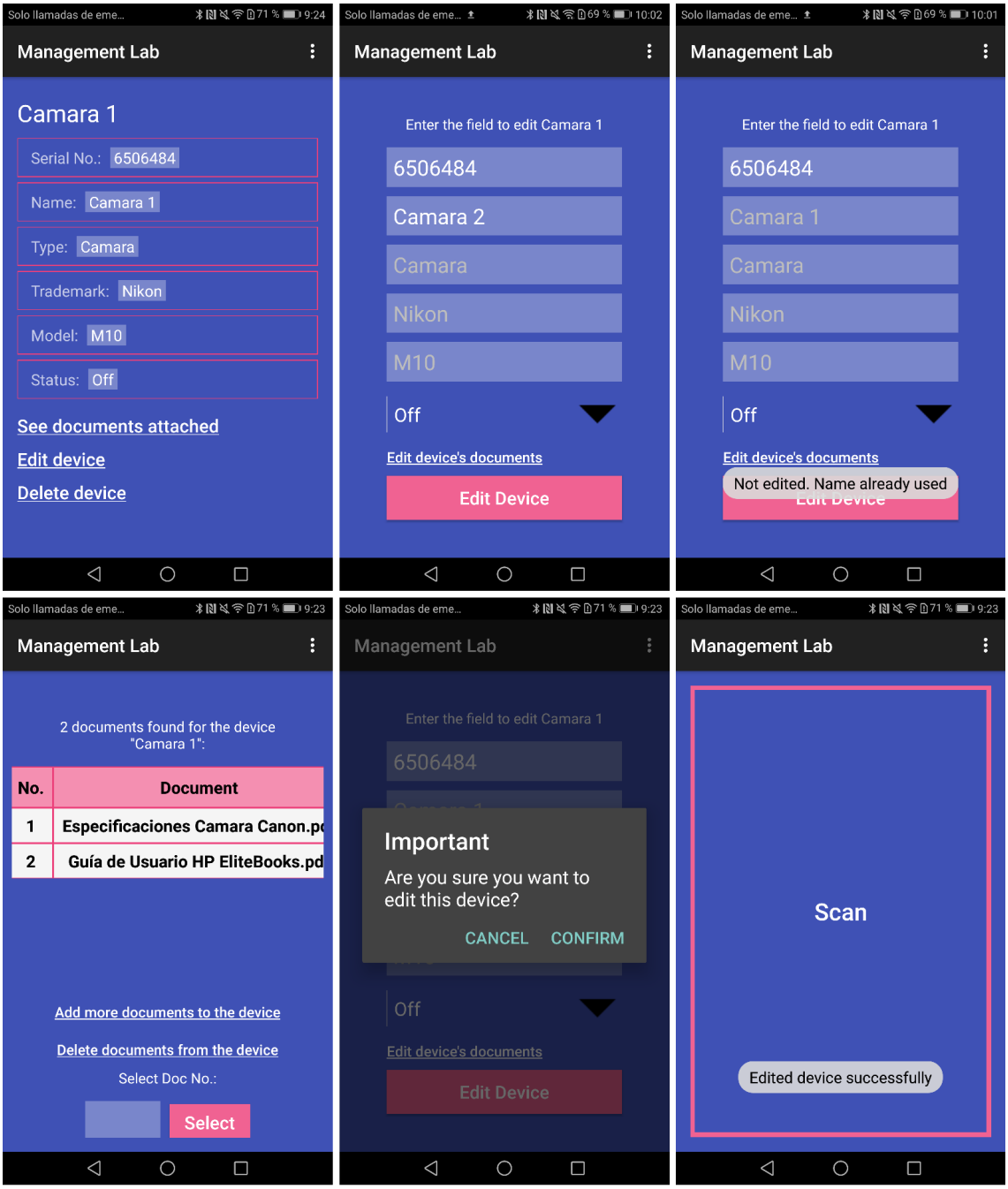

Figura 49. Flujo para editar un dispositivo ya existente

En la primera opción se ha puesto como nombre *"Camara 2"* la cual ya existe, por lo que la aplicación informa que no se puede editar el dispositivo, mientras que en la quinta pantalla se establece el estado a *"Off"* y ya si se edita el dispositivo con ese nuevo estado.

8. ***AddDocuments Activity & DisplayDocuments Activity***: Estas dos actividades están relacionadas con los documentos asociados a los dispositivos.

   Por un lado, ***Display Documents Activity*** es la actividad en donde se muestran los documentos del dispositivo seleccionado para visualizarlos.

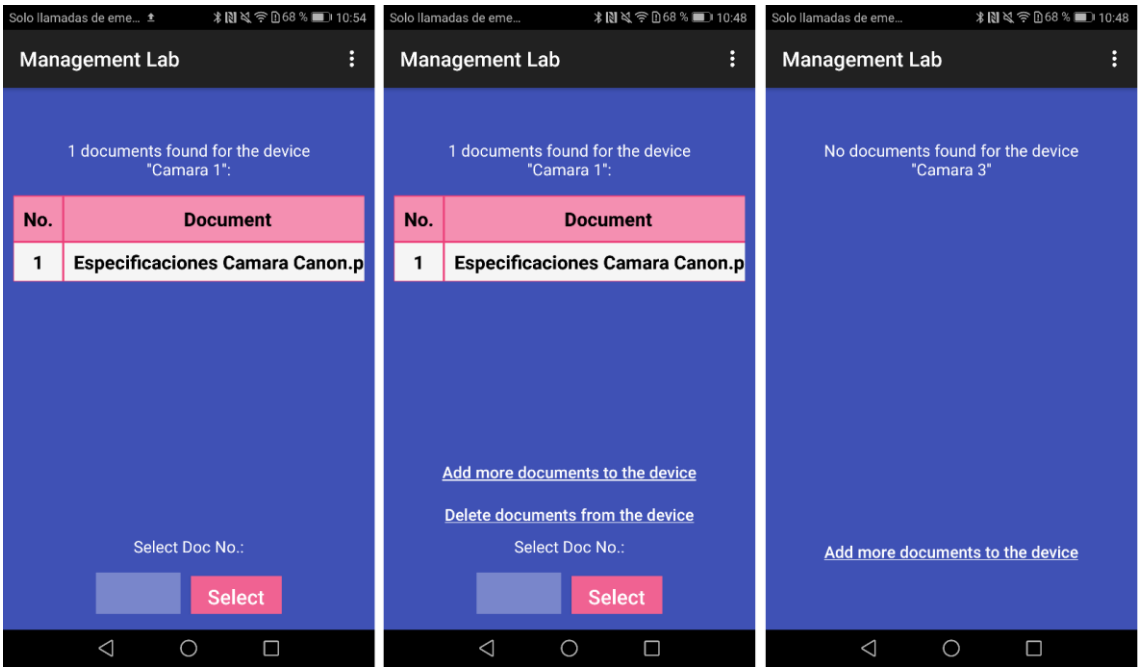

Figura 50. Pantallas principales de la actividad de visualización de documentos

Como se puede ver en las pantallas anteriores, dependiendo si se está en modo administrador o no, se podrá realizar las opciones de añadir o eliminar documentos, en caso negativo únicamente se podrán visualizar dichos documentos.

En caso de que no exista ningún documento, como se puede ver en la tercera pantalla de la figura, la aplicación avisa de esa casuística, y en caso de que el modo administrador esté activado, se podrá añadir nuevos documentos a dicho dispositivo.

Esta actividad se puede iniciar de dos maneras:

- La primera cuando se realiza una captura o una búsqueda de un dispositivo y donde se muestra la opción de visualizar sus documentos. Esta opción siempre es visible para el usuario ya que no requiere de permisos de administrador.

- La segunda opción, únicamente está disponible cuando está el modo administrador, ya que se puede ver los documentos del dispositivo cuando se está en el formulario de edición.

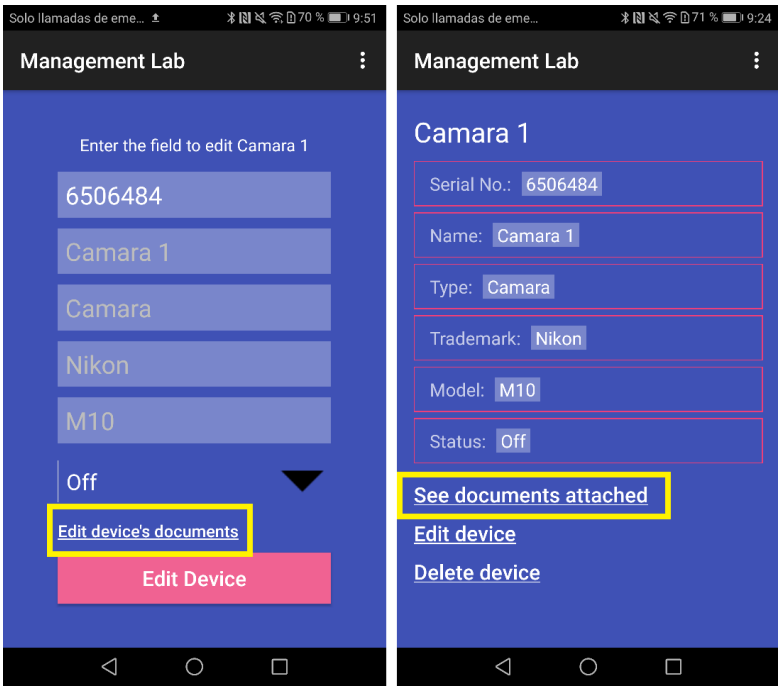

Figura 51. Modos para acceder a visualizar documentos

La segunda actividad, ***Add Documents Activity***, es la encargada de añadir documentos a cada uno de los dispositivos.

Esta actividad únicamente está disponible para el modo administrador, y, únicamente es accesible desde la actividad comentada anteriormente y, cuando se añade un nuevo dispositivo, que la aplicación te da la opción de añadir documentos a ese dispositivo.

La aplicación recoge el nombre de todos los dispositivos conectados al repositorio Cloud, y se realiza una comparación con los que ya hay añadidos para no mostrarlos en la lista, y ya el usuario escogerá que documento o documentos se añaden.

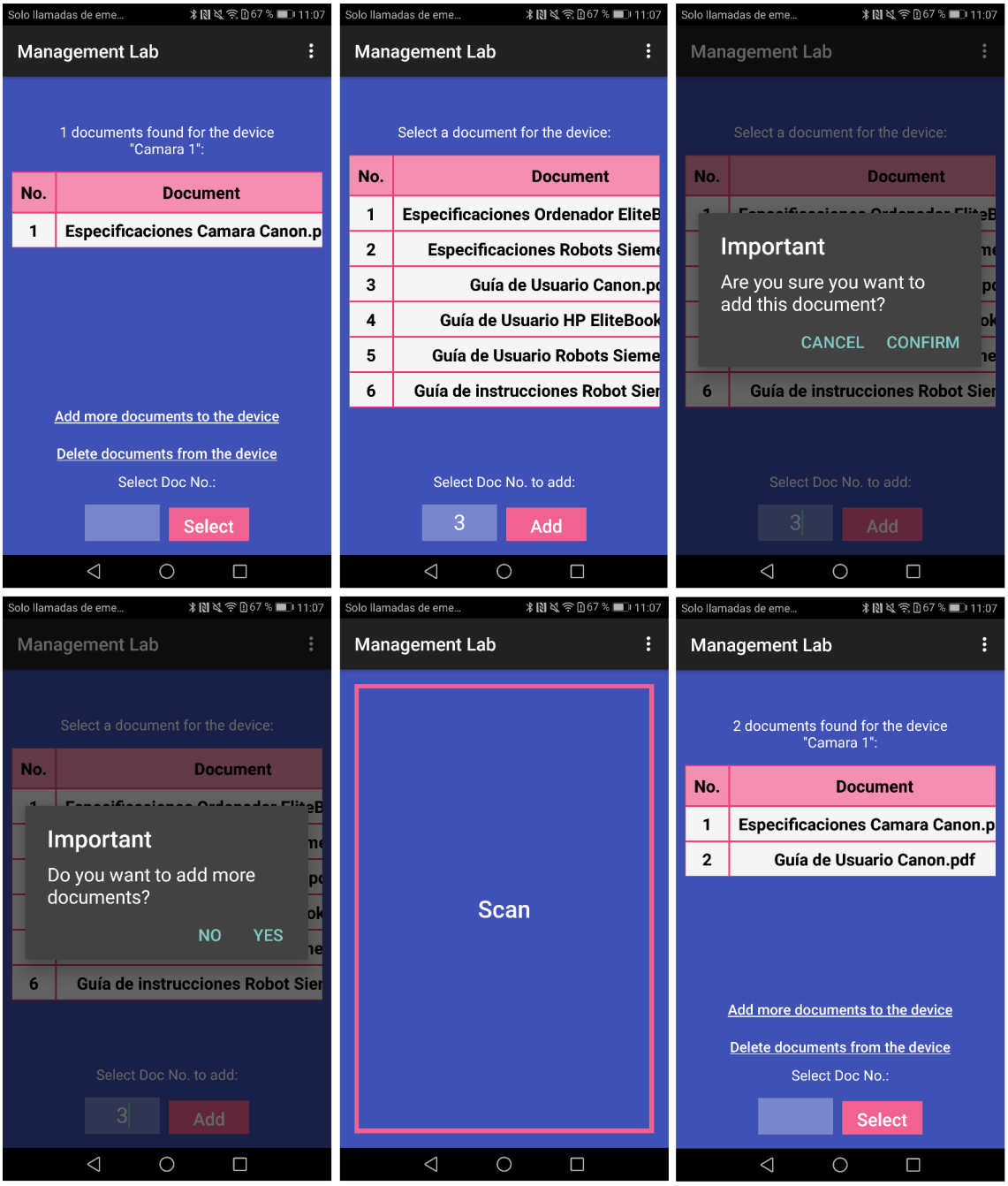

Figura 52. Flujo para añadir nuevos documentos a un dispositivo

Por tanto, en la siguiente figura se puede observar las maneras de acceder a la actividad de añadir documentos.

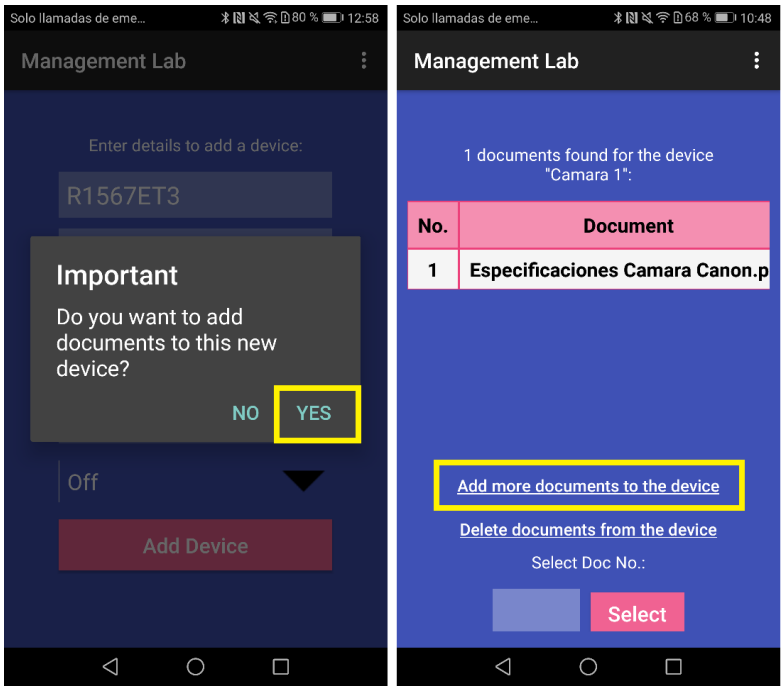

Figura 53. Modos para acceder a añadir nuevos documentos a un dispositivo

9. ***ResetPwd Activity & ResetPwdAdmin Activity***: Estas actividades son utilizadas para cambiar la contraseña o recuperar la contraseña que había por defecto.

   Las actividades de autenticación se habilitan nada más entrar en la aplicación con la autenticación de usuario para entrar en modo detección, y una vez dentro, la autenticación de administrador se requiere cuando se quiera entrar en modo operacional o administrador.

   En ambos casos comentados, para cambiar la contraseña se requerirá la contraseña de administrador actual, por lo que la contraseña de administrador nunca puede ser olvidada, ya que, aunque se resetee la contraseña de administrador, se necesitaría saber la original para poder realizar cambios de contraseña.

La actividad de reseteo de contraseña se podrá iniciar cuando el usuario en cualquiera de las dos autenticaciones requiera resetear o cambiar la contraseña.

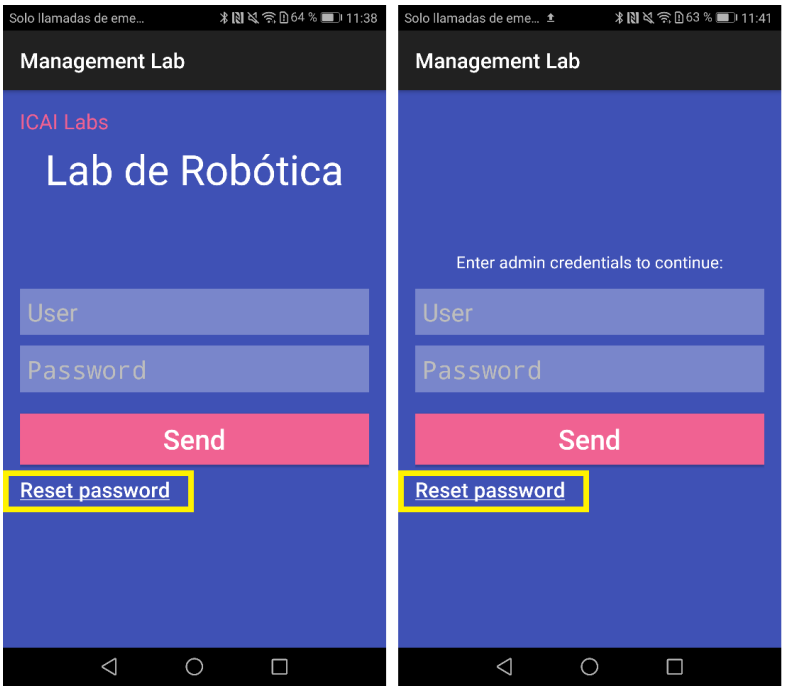

Figura 54. Modos para acceder a visualizar el cambio/reseteo de contraseña

El flujo de la actividad se muestra en la figura de la siguiente página.

Para dicho flujo, se puede observar cómo se requiere en primer lugar la contraseña actual de administrador, y una comprobación doble de contraseña para no poner una contraseña no deseada.

También, el link inferior se utiliza para resetear la contraseña a la contraseña que se estableció por defecto.

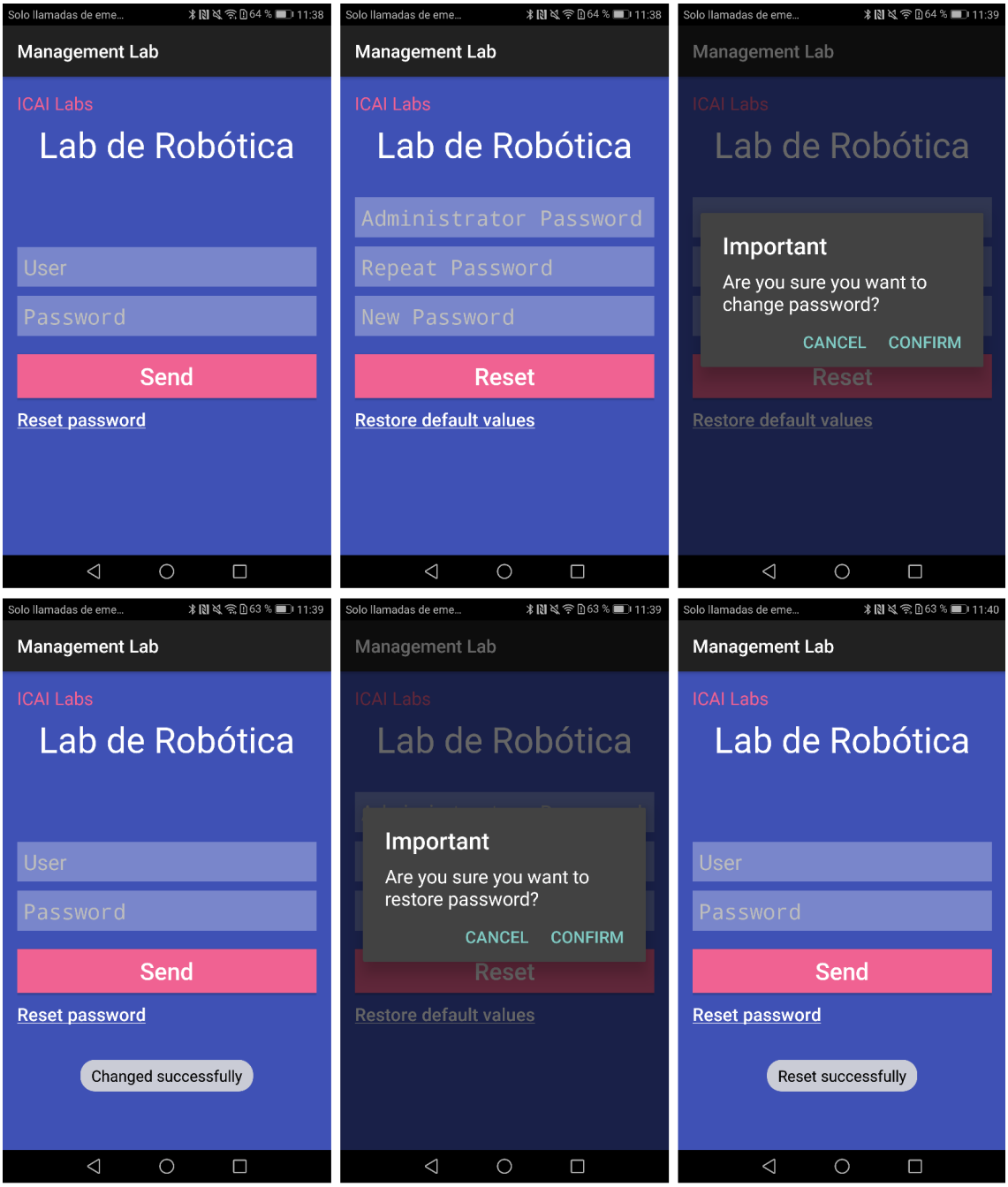

Figura 55. Flujo para el cambio/reseteo de contraseña

Por tanto, esta era la última de las actividades realizadas para el desarrollo de la solución. En los dos siguientes apartados se comentarán las clases java utilizadas de apoyo y la base de datos desarrollada.

7.2 CLASES DESARROLLADAS

En este apartado se van a comentar las clases Java desarrolladas las cuales son tres,

- ***Constants***: Constantes importantes de la aplicación, en esta clase java se escriben las constantes que son globales en toda la aplicación, por ejemplo, el nombre de la base de datos.

  Está implantación está considerada como una buena práctica ya que se reúne el valor de cualquier constante en un único sitio, por lo que si ese valor de la constante sufre modificaciones no obliga a una revisión del código ni puede dar a errores la aplicación.

  Se trata de una clase sin métodos, únicamente una clase final estática para que las actividades o clases obtengan el valor de cada atributo necesario.

- ***Device***: Clase definida para los dispositivos, con sus atributos y sus métodos. Se trata de la clase principal de la aplicación donde se define a los dispositivos. En esta clase se tiene como métodos los típicos *getters()* and *setters()*, y adicionalmente tres métodos específicos para tratar los documentos, en donde se elimina algún documento, se añade alguno, o se recoge los documentos de la base de datos.

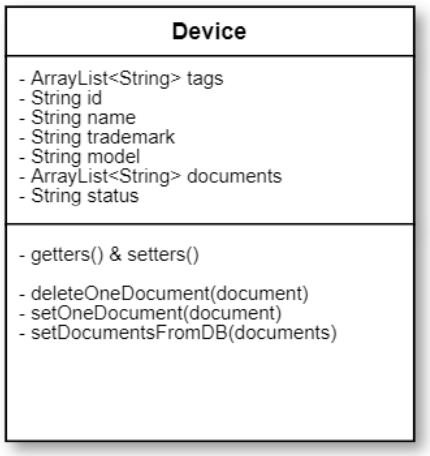

Figura 56. Diagrama de clase para Devices.java

- ***Drive Service***: Clase utilizada para la conexión con el repositorio online. Esta clase tiene tres métodos por defecto, los cuales corresponden a la conexión con Drive, y luego tiene tres métodos privados para realizar las conexiones a Drive para los documentos.

  El primer método recoge todos los documentos de un dispositivo del repositorio para mostrarle al usuario los documentos.
  El siguiente método recoge un único documento, el cual habrá seleccionado el usuario para visualizar y la clase recogerá dicho documento para mostrárselo en un editor PDF.
  Por último, el último método compara las listas existentes, es decir, compara los documentos de un dispositivo con todos los documentos alojados en el repositorio par

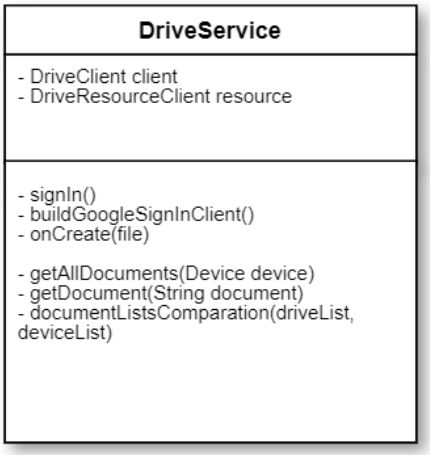

Figura 57. Diagrama de clase para DriveService.java

7.3 BASE DE DATOS DESARROLLADA

En primer lugar, se ha desarrollado una base de datos relacional (tipo SQL) ya que los datos a tratar son todos del tipo texto. Finalmente, dicha base de datos se encuentra alojada dentro de la aplicación y se tiene una clase *java* para realizar las conexiones.

Dicha clase se llama *DeviceDBHelper.java* y aloja distintos métodos como agregar, actualizar, eliminar dispositivos, añadir documentos, etc.

Como se puede ver en la siguiente figura, tiene métodos para añadir un dispositivo, obtener todos los dispositivos, obtener un único dispositivo de distintas maneras, actualizar el campo de un dispositivo, actualizar los documentos y, por último, eliminar un dispositivo.

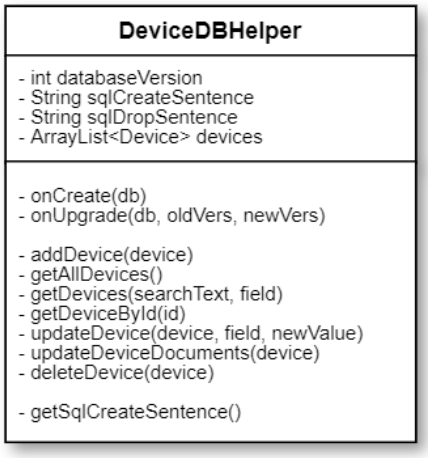

Figura 58. Diagrama de clase para DeviceDBHelper.java

Con esta clase para gestionar la base de datos y la clase *Device*, se puede inferir la estructura de la base de datos del tipo SQL, para ello, dicha base de datos tendrá cada uno de los datos de la clase *Device*, siendo el campo *id* clave única (PK).

Por tanto, la base datos tiene la siguiente estructura:

Barcode tag	QR tag	ID	Name	Type	Trademark	Model	Status
e3syLDQ […]sezYsNi5 fSx7wxfX0=	Mn0sTkke zgw**[…]**s Mk LDE=	3FE11 G887U	Laptop 1	Computer	Hewlett Packard	EliteBook ej15	Off

Tabla 4. Estructura de la base de datos

7.4 IMPLEMENTACIÓN DE LOS FLUJOS DE LA APLICACIÓN

A continuación, en este apartado, se muestran los diferentes flujos existentes en la aplicación, estos flujos son las distintas pantallas para cada uno de los flujos establecido en el diseño técnico. Por tanto, se pueden comparar estos flujos de pantallas con el diagrama de actividad realizado en la Figura 34.

Se representarán, por tanto, los siguientes flujos:

- Autenticación de usuario y reseteo de la contraseña.
- Escanear etiqueta.
- Búsqueda de un dispositivo y visualización de sus documentos.
- Añadir un dispositivo con/sin nuevos documentos.
- Edición de un dispositivo existente.
- Añadir documentos a un dispositivo existente.
- Eliminación de documentos.
- Eliminación de un dispositivo.

Debido al tamaño de las pantallas, y que se quiere mostrar lo suficientemente grande, se mostrará un flujo por página bien rotulado debajo, excepto los dos últimos que se pueden representar en la misma página.

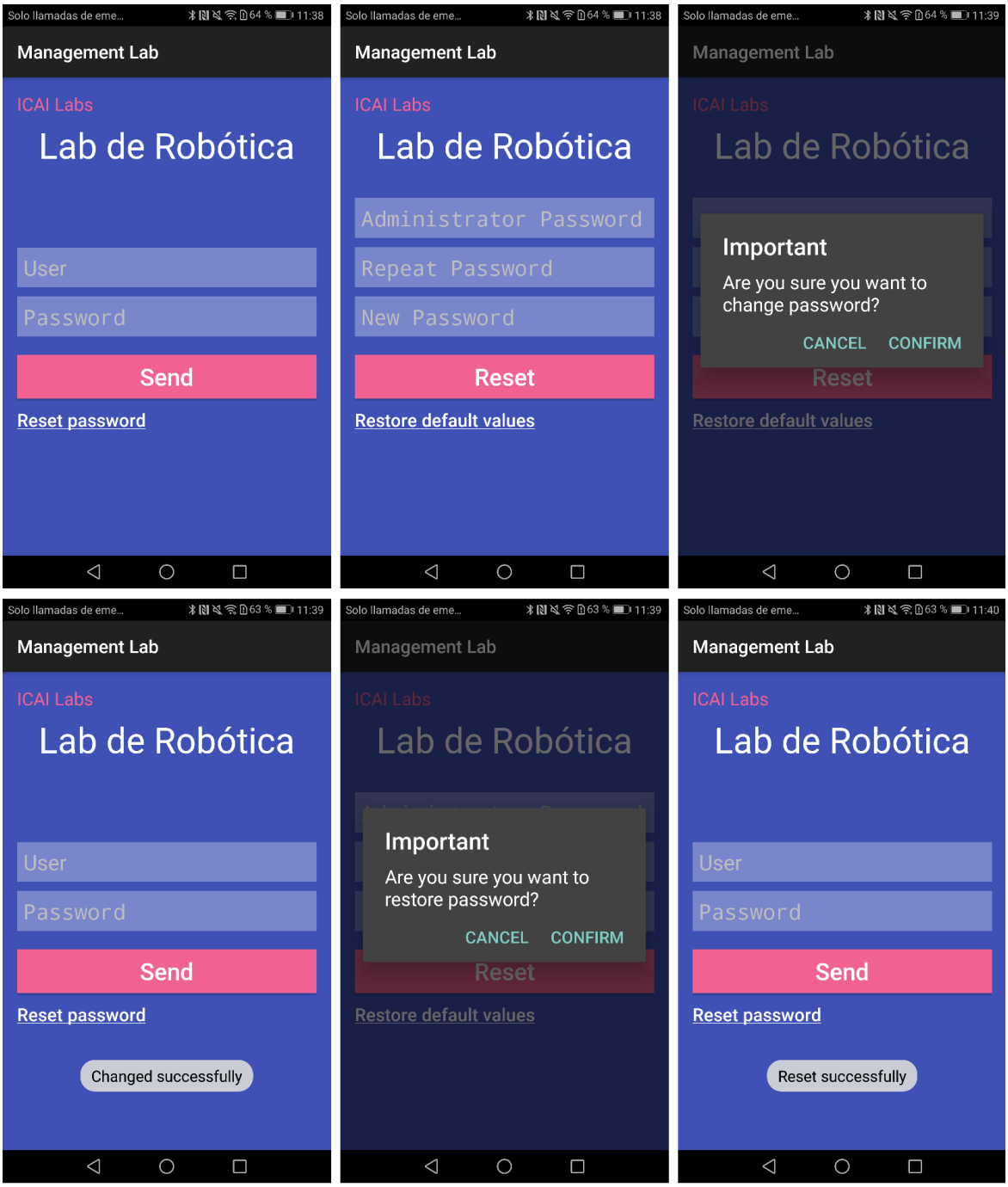

Figura 59. Flujo para la autenticación de usuario y reseteo de la contraseña.

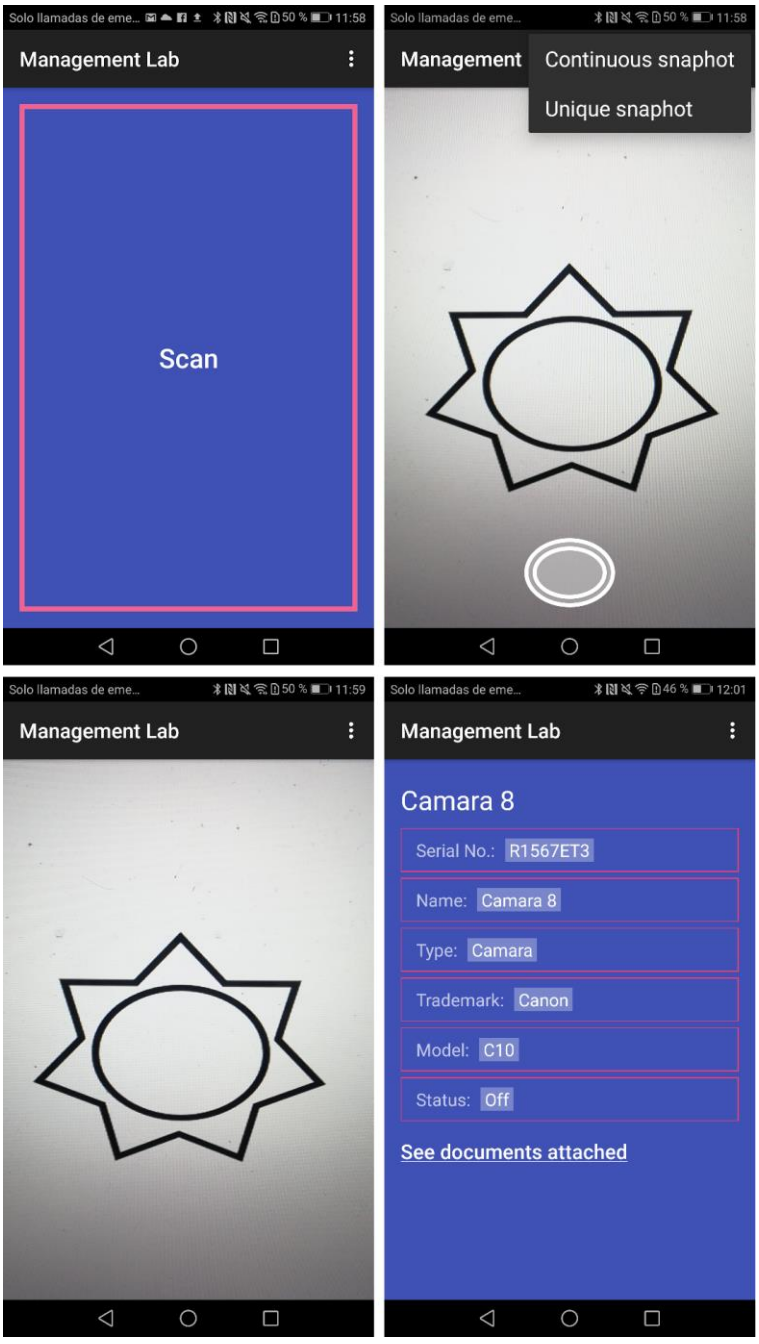

Figura 60. Flujo para el escaneo de etiquetas de los dispositivos

Figura 61. Flujo para la búsqueda de dispositivos

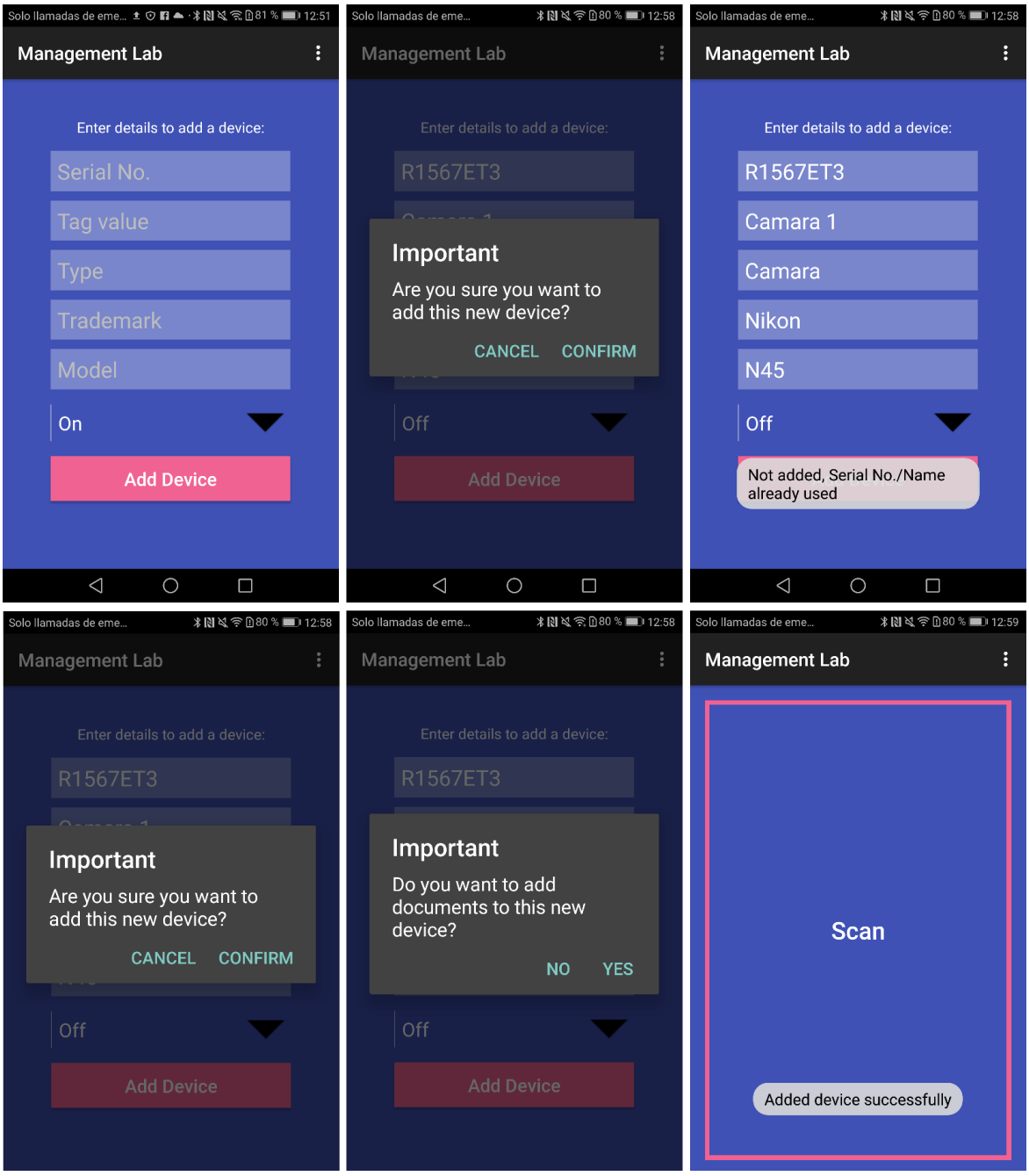

Figura 62. Flujo para añadir un dispositivo sin nuevos documentos

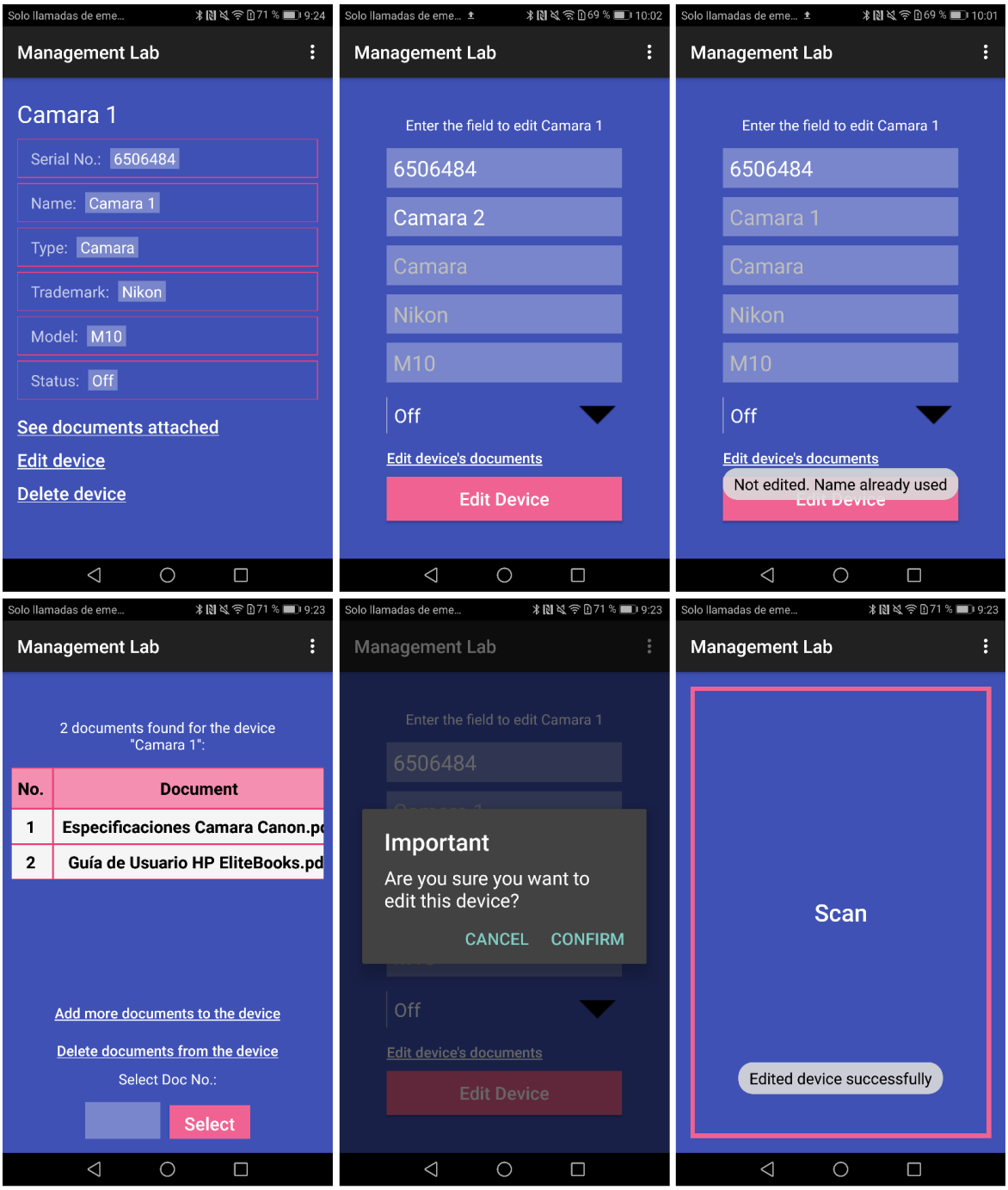

Figura 63. Flujo para editar un dispositivo ya existente

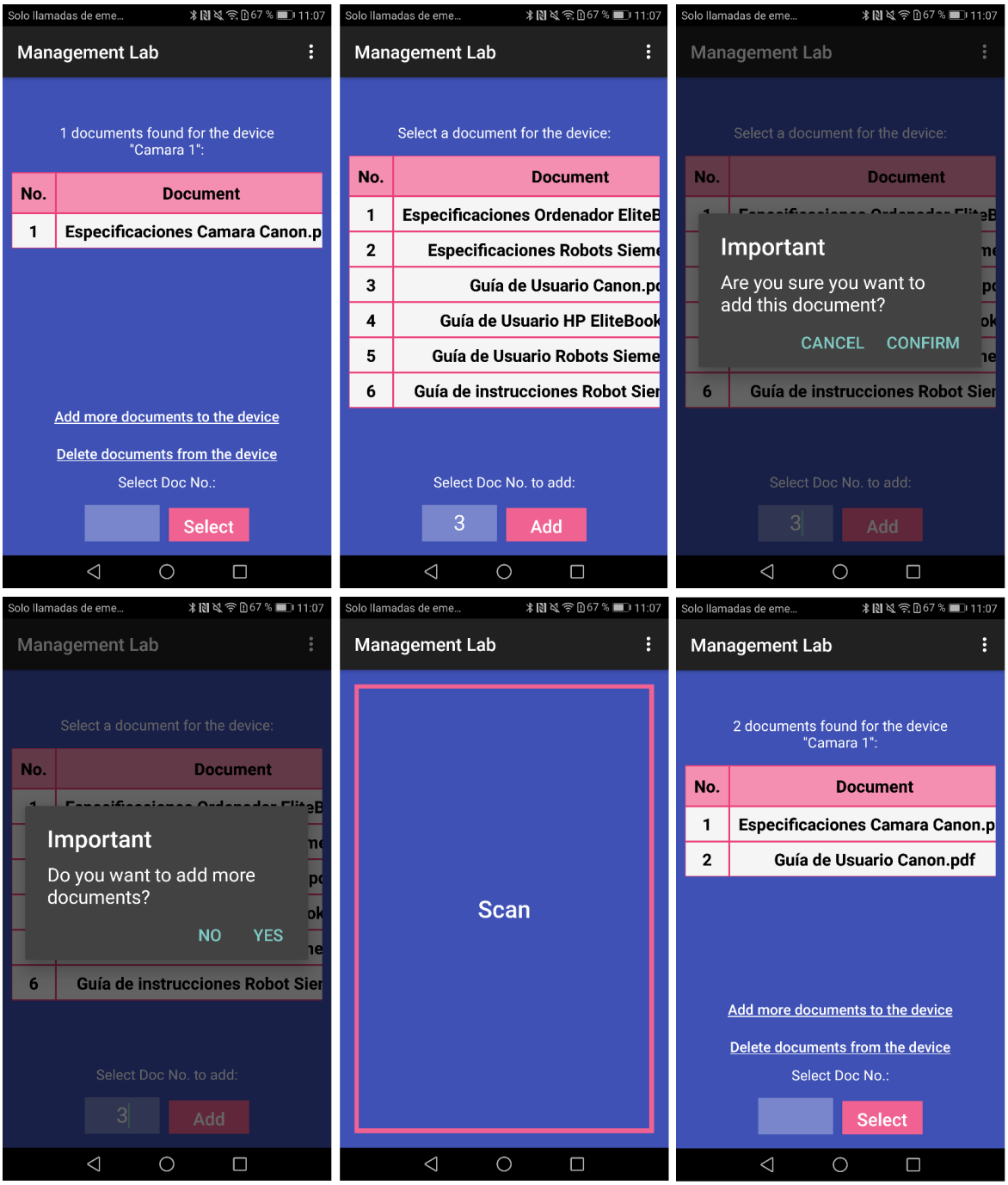

Figura 64. Flujo para añadir nuevos documentos a un dispositivo

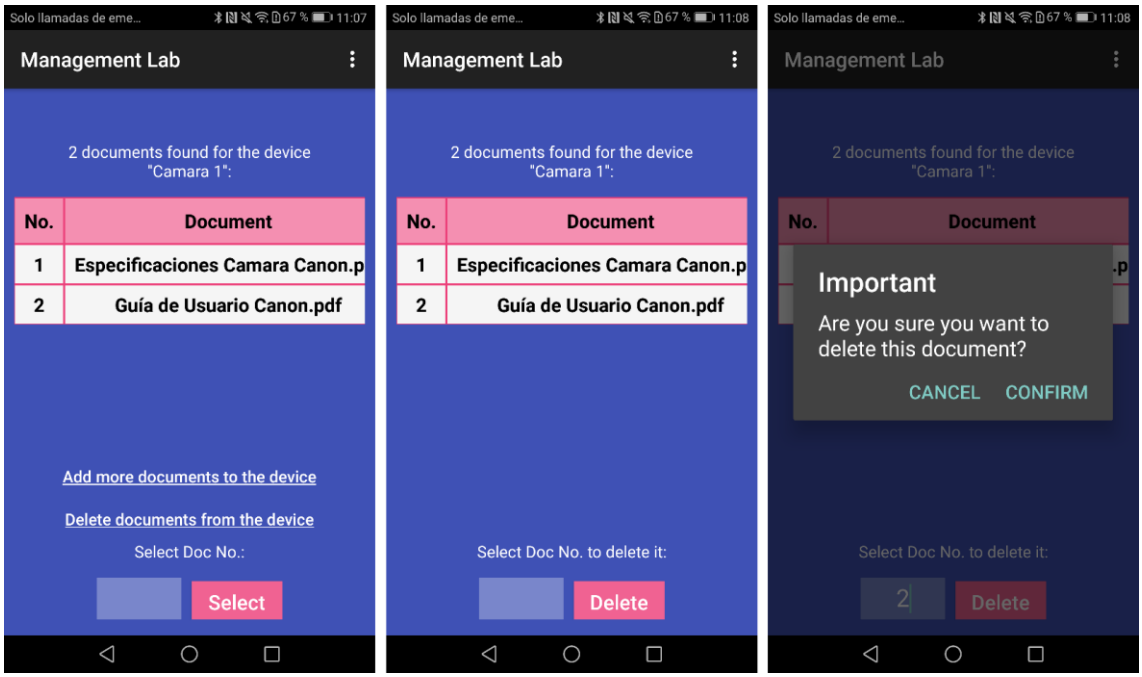

Figura 65. Flujo para eliminar un documento asociado a un dispositivo

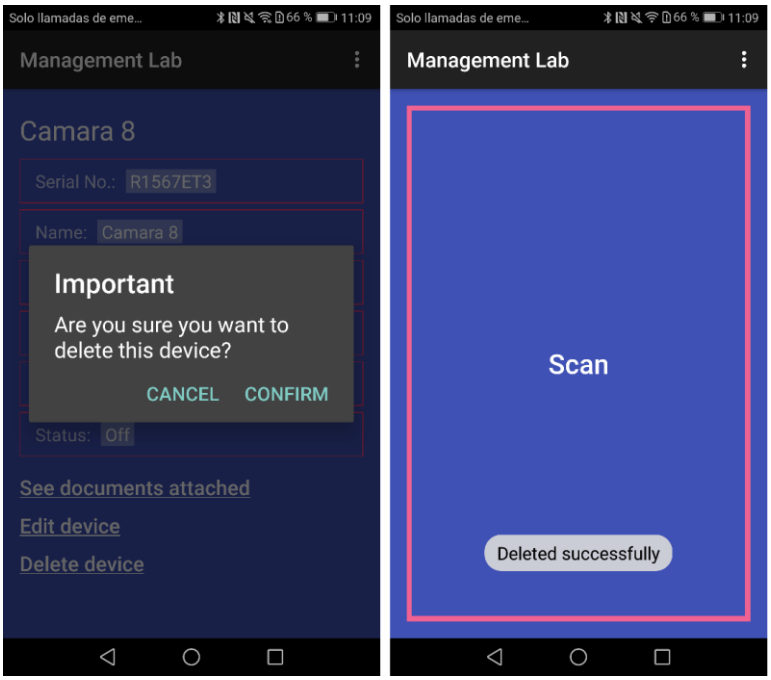

Figura 66. Flujo para eliminar un dispositivo

Capítulo 8. ANÁLISIS DE RESULTADOS

8.1 REQUISITOS DEL SISTEMA

Para analizar los resultados del proyecto se toman distintos códigos, con distintos valores para probar si la aplicación responde correctamente.

Además, aparte de utilizar diferentes luces artificiales también se imprimió los códigos de barras y códigos QR con una pequeña degradación.

Figura 67. Ejemplo de código de barras degradado para su impresión

Para realizar los resultados se ha utilizado como SDK el Android Kit Kat (Android 4.4) ya que más del 95% de los dispositivos son compatibles.

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,6%
4.2 Jelly Bean	17	98,1%
4.3 Jelly Bean	18	95,9%
4.4 KitKat	19	95,3%
5.0 Lollipop	21	85,0%
5.1 Lollipop	22	80,2%
6.0 Marshmallow	23	62,6%
7.0 Nougat	24	37,1%
7.1 Nougat	25	14,2%
8.0 Oreo	26	6,0%
8.1 Oreo	27	1,1%

Figura 68. Distribución de las APIs de Android.

Además, para compilar la aplicación y realizar las pruebas se ha utilizado dos versiones de Android totalmente distintas, las versiones *KitKat* (4.4) y *Nougat* (7.0), mediante un teléfono Sony Xperia T3 y un Huawei P9 Lite.

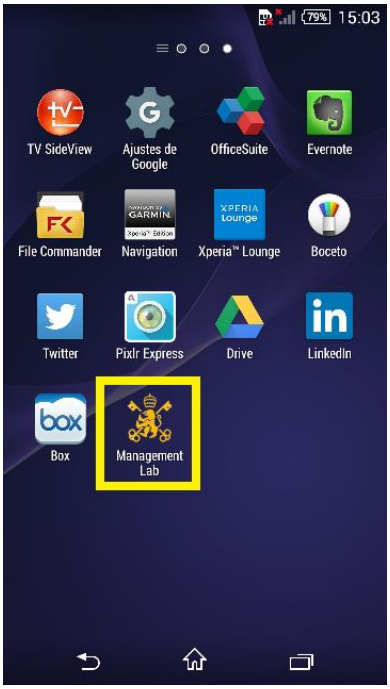

Figura 69. Terminal Sony Xperia T3 con la aplicación instalada

Por otro lado, se muestran los ajustes de ambos teléfonos para identificar las versiones.

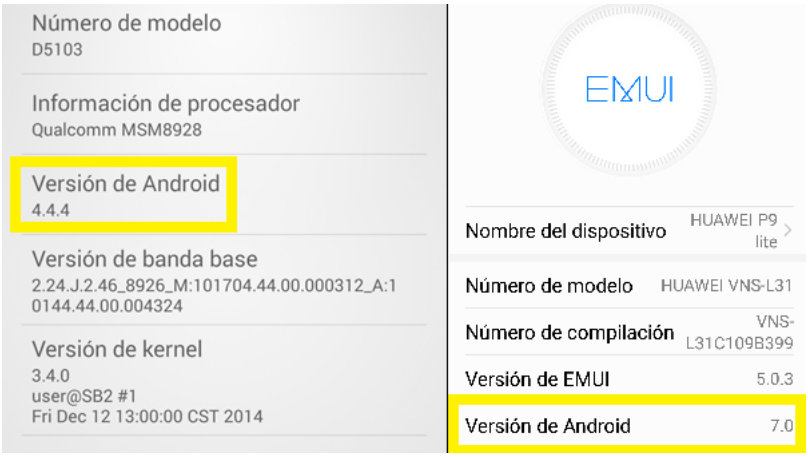

Figura 70. Información de los dos terminales (izquierda Sony, derecha Huawei)

Por tanto, se realiza un banco de pruebas con distintas etiquetas las cuales son:

- "robot 1", "robot 2" y "robot 3".
- "cámara 1", "cámara 2" y "cámara 3".
- "ordenador 1", "ordenador 2", "ordenador 3".

8.2 ANÁLISIS DE LOS RESULTADOS

Con estas 9 etiquetas guardadas en base de datos, se imprimen las 9 etiquetas y se realizan capturas en diferentes lugares con mayor y menor luminosidad. A estas 9 variantes se realizan hasta 10 capturas aproximadamente ya que se rotan la cámara aleatoriamente entre 5-8 grados con distinta luz, por tanto, se capturan distintas variaciones de cada una de las dos etiquetas dando los siguientes resultados:

	Precision	**Time (s)**[7]
Barcode	100%	1s – 2s
QR Code	100%	2s - 3s
Star *	100%	4s - 5s

Tabla 5. Resultados de la prueba en el terminal

Adicionalmente hay una etiqueta llamada *"star"* la cual se comentará más adelante.

Lo que si se observa es que el código QR es ligeramente más lento a la hora de capturarse que el caso de un código de barras. En esto influye dos grandes temas, el hardware del teléfono, en cuanto a que el procesamiento es más complejo más tardará, y el otro es con el algoritmo ya que para los códigos QR necesita por sí solo un mayor número de puntos de interés, lo que hace que el número de líneas de código a ejecutar sea mayor y por tanto mayor tiempo de ejecución.

[7] Tiempo aproximado ya que no se puede calcular con exactitud.

Aunque lo realmente importante es que no ha habido por el momento ningún falso positivo en más de 100 capturas que se han realizado en un periodo de tiempo corto, por tanto, se puede concluir que la elección del algoritmo es la adecuada y que la aplicación funciona correctamente gracias a la tasa máxima de acierto.

8.2.1 RESULTADOS CON ETIQUETAS NO ESTÁNDAR

En este apartado se quiere analizar el comportamiento del algoritmo más allá de las etiquetas de la aplicación. Por ello, como se comentó en el 5.8, se realizará un experimento con etiquetas no estándar, por ejemplo, la siguiente:

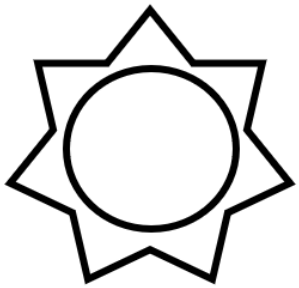

Figura 71. Etiqueta no estándar

Con el algoritmo SURF se obtienen los *keypoints* y los descriptores de la imagen y se asigna a un dispositivo, concretamente se ha elegido a “robot 3”.

Una vez asignada, se deben realizar unas pequeñas modificaciones en una aplicación duplicada que se llamará *TestCodeApp* en las que se realizará unas pequeñas modificaciones:

- Se añade dicha etiqueta no estándar al dispositivo elegido. Para ello, se realiza una modificación en la actividad de *AddEditDevice*, en donde se captura dicha etiqueta anterior con la cámara y se guarda la matriz de *keypoints* en los dos campos disponibles en la base de datos (*barcode tag* y *QR tag*) para realizar la comparación posterior.

- Se modifica el modo de detección único de la cámara, es decir, el modo que el usuario captura la imagen. Se modifica en cuanto a que, anteriormente, para evitar fallos de captura, el usuario si capturaba la imagen, dicha imagen se comparaba con un rectángulo y un cuadrado (forma de los *barcode* y QR), por lo que, si son similares al 50%, se seguirá procesando la imagen, y si no, se entenderá un error de captura por parte del usuario.
- Es decir, se elimina esa comprobación previa, para que cualquier captura que se tome, se analice, y así poder capturar la etiqueta no estándar y realizar la comparación con todos los dispositivos alojados en la base de datos.

Por tanto, con dichas modificaciones, se captura la etiqueta y se realiza el mismo paso en la segunda imagen, es decir, se obtienen los *keypoints* y los descriptores de la imagen.

Una vez que se tienen ambas cosas se pueden comparar las dos imágenes (valor obtenido, por el valor de la base de datos) y luego ver la correspondencia entre los puntos, y como se puede comprobar se obtiene el resultado esperado.

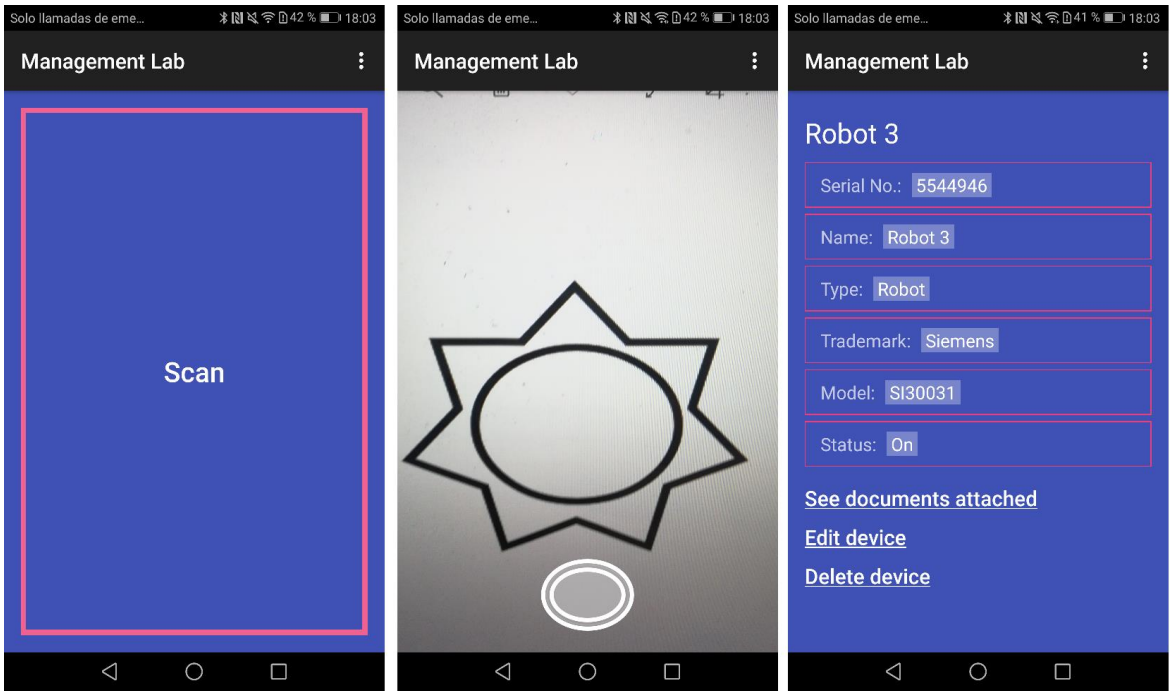

Figura 72. Captura con la aplicación de una etiqueta no estándar

Además, si se observa la Tabla 5, realizando distintos ángulos, iluminaciones, se obtiene un resultado 100% en un tiempo similar. Este resultado, es el esperado, ya que la codificación de puntos es mucho más sencilla, el algoritmo SURF establece un punto de interés en cada vértice y vértice inferior, y con el circulo establece varios puntos de interés cada 15º, o 20º aproximadamente. Por tanto, si se combinan diferentes polígonos, la distinción de etiquetas es también muy sencilla de detectar para el algoritmo.

En definitiva, independientemente de la etiqueta que se utilice, el algoritmo es capaz de identificar el dispositivo.

Capítulo 9. CONCLUSIONES Y TRABAJOS FUTUROS

En primer lugar, cabe destacar el trabajo de investigación realizado en este trabajo final, en el que se ha intentado explicar de una manera sencilla para el usuario las distintas posibilidades existentes en el mundo de la realidad aumentada, y las soluciones que se pueden aportar con cada una de las diferentes herramientas.

En este caso, el trabajo se ha centrado en el algoritmo *Speeded Up Robust Features* (SURF) como algoritmo de identificación de las etiquetas, ya que con las pruebas que se realizaron inicialmente se obtuvieron resultados claramente mejores, además, para la utilización de código de barras el algoritmo está bastante optimizado como se pudo ver en las comparaciones del Capítulo 5.

En cuanto a la herramienta realizada para el laboratorio de la universidad, y por consiguiente para cualquier minifábrica de tamaño medio, se ha obtenido una solución final que aporta un grado de mejora a la organización de cualquier estructura de máquinas, además con la solución de la realidad aumentada se ha aportado una sencillez y automatización intuitiva tanto para el usuario y el administrador a la hora de tratar con un gran número de dispositivos.

Con la investigación se quiere poder aclarar la posibilidad de utilización de cada uno de los algoritmos en trabajos futuros, ya que en la investigación se define claramente las virtudes y defectos de cada uno de los algoritmos y así poder adecuar los distintos proyectos futuros a los mejores algoritmos o los que den mejores resultados.

Por último, con esta solución se plantea como la piedra inicial de cara a una posible automatización global del laboratorio ya que esta herramienta soluciona el nivel de supervisión (ver Figura 1) con lo que con trabajos y mejoras futuras (nivel 3 en adelante) se podría conseguir una solución integrada mucho más potente.

Capítulo 10. REFERENCIAS

[1] Rodríguez Mondéjar, J.A. "Mundo Industrial: Automatización, sistemas de control y supervisión". *Universidad Pontificia de Comillas*, septiembre 2017.

[2] Instrumentation, Systems and Automation Society. "ANSI/ISA-95, Enterprise-Control System Integration". *ISA*, 1990.

[3] H. Ballard, Dana; Brown, Christopher M. "Computer Vision". Prentice Hall, 1982.

[4] Office of the Manager National Communications System, "Supervisory Control and Data Acquisition (SCADA) Systems", *National Communications System*, octubre 2004.

[5] Schwaber, K; Sutherland, J. "The Definitive Guide to Scrum: The Rules of the Game", *The Scrum Guide™*, julio 2003.

[6] Willis, Andrew; Sui, Yunfeng. "An Algebraic Model for fast Corner Detection". *12th IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[7] Harris, Chris; Stephens, Mike. "A Combined Corner and Edge Detector". *Proceedings of the 4th Alvey Vision Conference pp. 147-151*, 1988.

[8] Shi, J; Tomasi, C. "Good Features to Track". *9th IEEE Conference on Computer Vision and Pattern Recognition*. Springer, 1994.

[9] Rosten, Edward; Drummong, Tom. "Machine Learning for High-Speed Corner Detection". *IEEE International Conference on Computer Vision pp. 1508-1511*, 2005.

[10] Mitchell, T.M. "Machine Learning". *McGraw-Hill Chapter 3 – Decision Tree Learning*, 1997.

[11] Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints". *International Journal of Computer Vision*, 2004.

[12] Wikipedia. *Principal Curvature* (s.f.). En *Wikipedia*. Recuperado el 24 de abril de 2018 de https://en.wikipedia.org/wiki/Principal_curvature#Generalizations.

[13] Bay, H; Tuytelaars, T; Van Gool, L. "SURF: Speeded Up Robust Features". *ETH Zurich, Katholieke Universitet Leuven*, 2006.

[14] Haar A. "Zur Theory Orthogonal Function Systems", *Mathematische Annalen, 69, pp. 331-371*, 1910.

[15] Calonder, M; Lepetit, V; Strecha, C; Fua, P. "Binary Robust Independent Elementary Features". *CVLab, EPFL, Lausanne, Switzerland*, 2010.

[16] Wikipedia. *Coma Flotante* (s.f.). En *Wikipedia*. Recuperado el 15 de junio de 2018 de https://es.wikipedia.org/wiki/Coma_flotante.

[17] Wikipedia. *Distancia de Hamming* (s.f.). En *Wikipedia*. Recuperado el 15 de junio de 2018 de https://es.wikipedia.org/wiki/Distancia_de_Hamming.

[18] Calonder, M; Lepetit, V; Özuysal, M; Trzcinski, T; Strecha, C; Fua, P. "BRIEF: Computing a local binary descriptor very fast". *CVLab, EPFL, Lausanne, Switzerland*, 2011.

[19] Rublee, E; Rabaud, V; Konolige, K; Bradski, G.R. "*ORB: An efficient alternative to SIFT or SURF*". Willow Garage, Menlo Park, California, 2011.

[20] Wikipedia. *Algoritmo voraz* (s.f.). En *Wikipedia*. Recuperado el 20 de junio de 2018 de https://es.wikipedia.org/wiki/Algoritmo_voraz.

[21] Lv, Q; Josephson, W; Wang, Z; Charikar. M; Li, K. "*Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search*". Department of Computer Science, Princeton University, 2007.

[22] Android. *Intents y filtros de intents* (s.f.). En *Developer Android*. Recuperado el 7 de septiembre de 2018 del sitio web https://developer.android.com/guide/components/intents-filters?hl=es-419.

ANEXO A. PLAN DE TRABAJO COMPLETO

En este anexo se incluye el cronograma asociado al plan de trabajo comentado anteriormente (Figura 3) para poder ver gráficamente el plan de trabajo completo siendo las fases las siguientes:

- 1: Estudio de algoritmos de reconocimiento.
- 2: Desarrollo de una solución móvil.
- 3: Integración de datos.
- 4: Carga de datos y pruebas.
- 5: Project *management*.
- 6: Finalización del proyecto.

Las cuales quedan marcadas en amarillo en el siguiente cronograma.

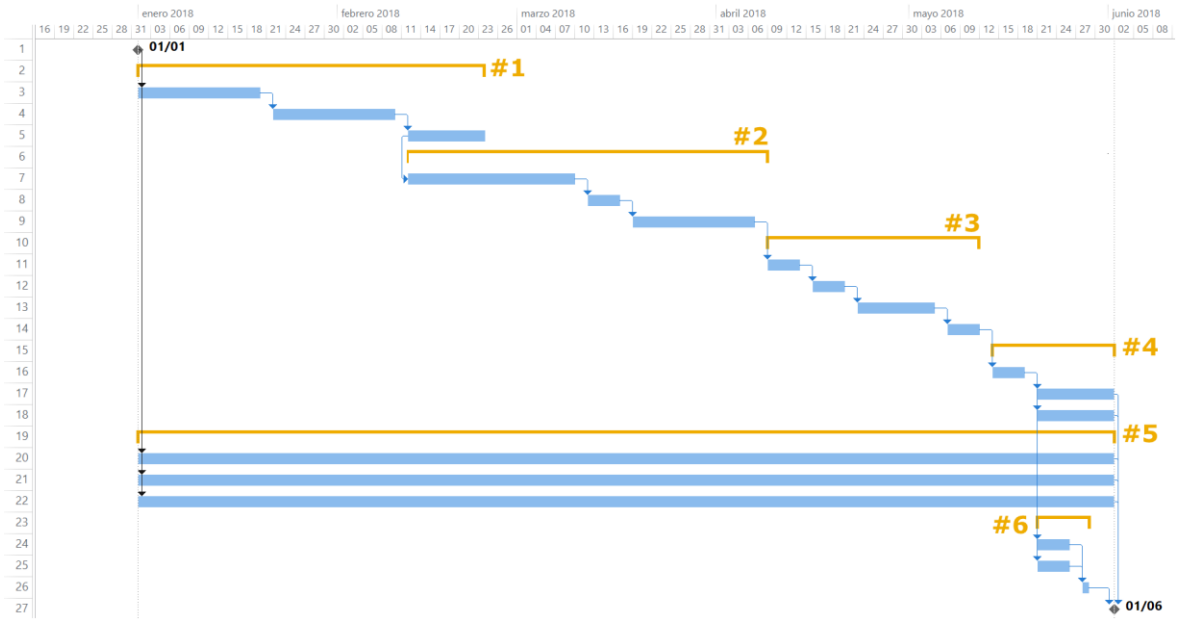

Figura 73. Cronograma de Trabajo del Proyecto Fin de Máster (Diagrama de Gannt)