GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Aplicación de la visión artificial a la seguridad de un robot

Autor: Concepción Góngora Luque

Director: Jaime Boal Martín-Larrauri

Director: José Antonio Rodríguez Mondéjar

Madrid

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

Aplicación de la visión artificial a la seguridad de un robot.

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018/2019 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.: Concepción Góngora Luque Fecha: 17 / 06 / 2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Jaime Boal Martín-Larrauri Fecha: 17 / 06 / 2019

Fdo.: José Antonio Rodríguez Mondéjar Fecha: 17 / 06 / 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.__ Concepción Góngora Luque _____DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Aplicacion de la visión artificial a la seguridad de un robot, 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 17 de junio de 2019

ACEPTA

Fdo……

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

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Aplicación de la visión artificial a la seguridad de un robot

Autor: Concepción Góngora Luque

Director: Jaime Boal Martín-Larrauri

Director: José Antonio Rodríguez Mondéjar

Madrid

….

Agradecimientos

En primer lugar, me gustaría agradecer este trabajo a todos los profesores que he tenido a lo largo de mi formación académica, en especial a Jaime por haberme ofrecido la oportunidad de realizar este proyecto, por su completa dedicación y por ofrecer siempre una solución a todos los problemas que he ido teniendo a lo largo del desarrollo del trabajo. Este trabajo es suyo.

También me gustaría agradecérselo a mis padres y mi hermano Pablo porque sin ellos no sería quien soy ni estaría donde estoy. A todos mis compañeros de clase por haberme ayudado de una manera o de otra a disfrutar de estos años de carrera. Desde prestarme apuntes hasta salir a tomar algo y pasar un buen rato después de tantas horas de estudio.

Finalmente a Sergio, por su apoyo, ayuda y sobretodo por enseñarme lo que de verdad importa.

Aplicación de la visión artificial a la seguridad de un robot

Autor: Concepción Góngora Luque.
Directores: Boal Martín-Larrauri, Jaime y Mondejar Rodríguez, Jose Antonio.

Palabras clave: Seguridad, IRB120, visión artificial, D435, SSIM, Algoritmo de Múltiples Gaussianas

Resumen del Proyecto

El presente proyecto define un mecanismo de seguridad para los robots industriales IRB 120 de la compañía ABB instalados en la minifábrica del laboratorio de automatización de la Universidad Pontificia de Comillas (ICAI). El objetivo principal de la aplicación consiste en proporcionar a dichos autómatas la visión artificial suficiente como para hacer posible una actividad colaborativa con los estudiantes.

Estado de la cuestión

Todo robot dentro de una cadena industrial automatizada en la Unión Europea deberá tener instalado su propio mecanismo de seguridad [1]. Principalmente se pueden encontrar dos tipos. Un funcionamiento con área de seguridad claramente definida, dentro de la cual ningún usuario podrá estar mientras el robot se encuentre en funcionamiento y actividad colaborativa entre robot y humano en la que ambos trabajen en un mismo área en armonía. Este segundo tipo, es bien sabido que aumenta la eficiencia y productividad en los trabajos realizados con respecto al primero. Para hacer posible la actividad colaborativa entre trabajadores y robots es necesario implementar al robot algún mecanismo que incluya elementos *software* y/o *hardware*. La seguridad de los robots industriales IRB 120 es del primer tipo y con este proyecto se pretende ofrecer a este tipo de robots la capacidad de trabajar en colaboración con los humanos de manera sencilla y barata.

Descripción del sistema

Para garantizar la seguridad del robot y los usuarios, el sistema implementará la inteligencia artificial suficiente como para que el robot se detenga o

adecue su velocidad dependiendo de los elementos que le rodean y la distancia a ellos, más en concreto aquellos elementos con movimiento. Un requisito fundamental será ofrecer una respuesta en tiempo real. Para cumplir con los objetivos el sistema está compuesto de tres partes; una cámara D435, producto de la empresa Intel, una aplicación para el análisis de información capturada implementada en Matlab y por último, un módulo para el control del robot IRB 120 de la compañía ABB implementado en su controlador IRC5. La misión de cada parte respectivamente es, toma de información del área de trabajo, análisis de la información y finalmente la última etapa consiste en la actuación del robot en función de los resultados obtenidos en los pasos previos. Para ello se debe realizar una comunicación entre los tres bloques. Bidireccional en el caso de la cámara y Matlab y unidireccional para Matlab y el controlador del robot.

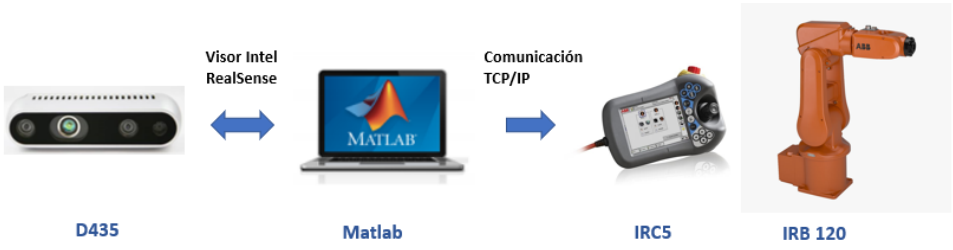


Figura 1: Diagrama de los bloques que componen el sistema desarrollado y la comunicación establecida entre ellos.

Dentro de la aplicación de visión artificial es la encargada de, a partir de una imagen capturada por una cámara, determinar el nivel de riesgo al que está expuesto el robot, para así comunicarle al robot la velocidad a la que debe trabajar. Esta aplicación divide el procesado de cada fotograma en seis etapas, representadas en la figura 5, desde la adquisición de la información del entorno hasta la determinación del índice de riesgo. El índice de riesgo podrá alcanzar cuatro valores. 0 para las situaciones en las que no hay ningún obstáculo con movimiento, 1 para situaciones en las que los obstáculos se encuentran a más de 30 cm, 2 para aquellas en las que los obstáculos están a más de 10 cm y se reduce la velocidad en un 70 %, y finalmente valor 3 para comunicar máximo riesgo y ordenar al robot su parada inmediata.

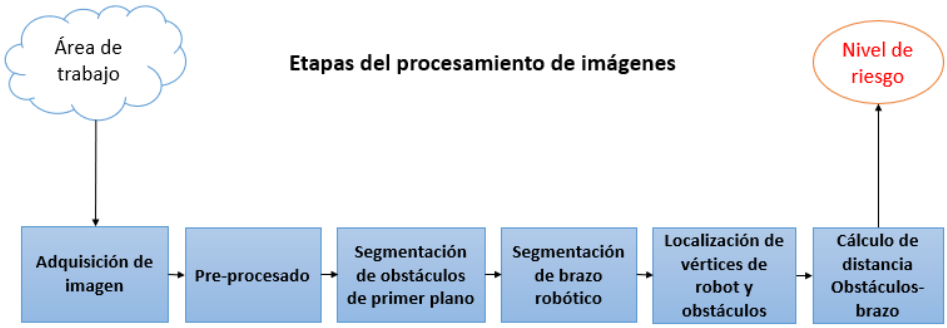


Figura 2: Etapas en el procesamiento de fotogramas.

Resultados

El modelo funciona correctamente cuando se trabaja en condiciones de iluminación perfectas (fuente de luz natural con intensidad media y sin la aparición de sombras o reflejos) y con tan solo dos niveles de riesgo (seguro y alarma), deteniéndose cada vez que se introduce un objeto nuevo en el área de trabajo con movimiento.

Al modificar las condiciones de iluminación, las sombras y reflejos hacen más inestable el sistema, especialmente en la etapa de detección de brazo. A pesar de las correcciones realizadas existen un par de casos en los que saltan falsas alarmas por obstáculos, detectando secciones del brazo como obstáculos con peligro.

Cuando se introducen dos niveles más de riesgo (medio y bajo) el robot no siempre modera la velocidad de manera adecuada, es decir, se registran algunos casos en los que el robot reduce su velocidad en un 70 % cuando debería hacerlo en un 20 % y viceversa.

Finalmente el tiempo de respuesta de la aplicación de procesamiento de imágenes aparece representado en la figura 6. Se observa como nunca supera los 0,2 segundos, y por tanto puede considerarse que ofrece una respuesta en tiempo real. Con esta misma frecuencia se envían los mensajes al robot, el cual lee el *buffer* de entrada cada 0.2 segundos también. Se observa un pequeño retraso en la reacción del robot desde que se envía un mensaje de cambio de estado para disminuir o aumentar la velocidad, debido al retardo introducido por el controlador y a las colas que en ocasiones se generan a la entrada pues se envía un mensaje por cada fotograma capturado.

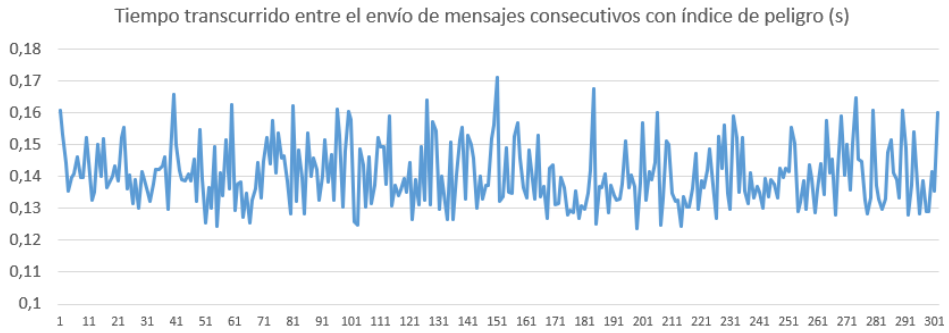


Figura 3: Tiempo requerido para el procesamiento de imágenes y la toma de decisión.

Artificial vision safety system applied to industrial robots

Concepción Góngora Luque

Keywords: Matlab, IRB120, Artificial vision, D435, SSIM, Gaussian mixture models

Abstract

The present Project describes a safety mechanism for the industrial ABB robots IRB 120 installed in the micro factory in the automation laboratory of the Universidad Pontificia de Comillas (ICAI). The main purpose of the system is to provide these automatons enough computer vision to make possible a joint activity with students.

State of the question

Every robot inside an industrial chain inside the European Union must have installed is own safety mechanism [1]. There are two major types, one in which the workspace is clearly defined and where no user is allowed to be while the robot is working and another type where robot and human work in the same area. This last kind of safety system it is well known for being much more efficient in terms of production than the previously mentioned system. In order to make this collaborative activity between humans and robots possible, it is necessary to implement some kind of mechanism in the robot that includes *software* and/or *hardware*. The safety system used nowadays is the first type, and this project will provide the robots the ability to work jointly with the employees in a simple and economic way.

System description

In order to guarantee the safety of both users and robots, the system will implement enough artificial intelligence to determine if the robot needs to stop or adequate its speed depending on the objects in its surroundings and

the distance to them, especially to those in movement. An essential requirement is to offer real time response. In order to comply with the objectives, the system is composed of three modules; an Intel D435 camera, a Matlab application that analyzes the information captured by the robot, and finally a control module for the ABB robot IRB 120 implemented in its IRC5 controller. The objective of each part respectively is to acquire information from the workspace, to analyze it and finally the action that the robot has to make depending on the results obtained in the previous modules. So as to communicate these three blocks, there is a bidirectional transmission between the camera and Matlab and unidirectional between Matlab and the controller.

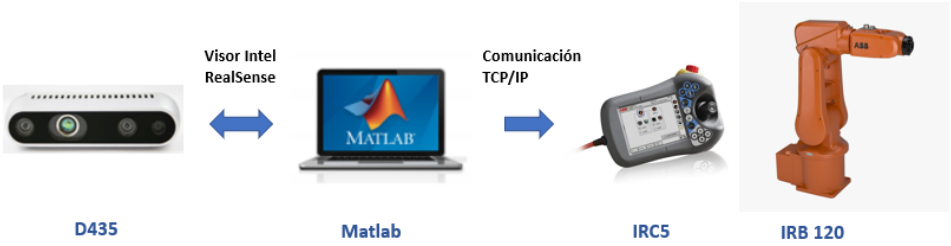


Figura 4: Diagrama de los bloques que componen el sistema desarrollado y la comunicación establecida entre ellos.

Inside the computer vision application it is the in charge of, from a certain image captured by the camera, determine the risk level to which the robot is exposed in order to communicate the robot at which speed has to work. This application separates the frame processing in 6 stages, represented in the figure 5, from the information acquisition from the environment to the risk index definition. This risk index can be 0 in situations where there is no obstacle in movement, 1 where obstacles are 30 cm away or more, 2 for situations where obstacles are between 30 and 10 cm away and it reduces its speed by a 70 % rate, and finally value 3 to communicate maximum danger and order the robot to stop immediately.

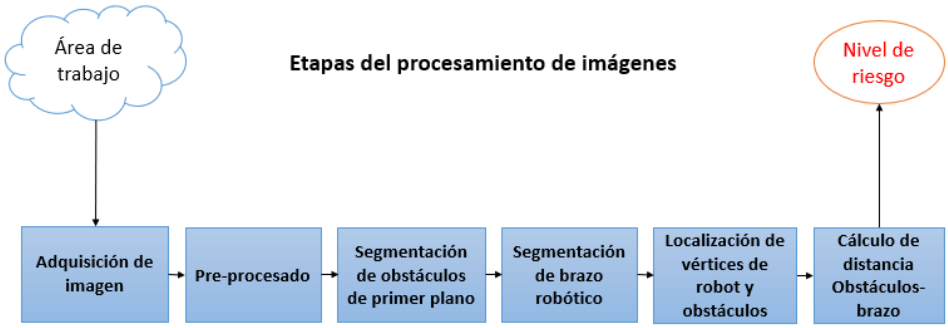


Figura 5: Etapas en el procesamiento de fotogramas.

Results

The system works accurately when working in ideal lighting conditions (medium intensity light source and without shadows or reflexes) and with only two risk levels (safe and alarm), stopping every time that a new object is introduced in the working area with some movement.

When lighting conditions do not remain unchanged, shadows and reflexes make the system more unstable, especially in the arm detection stage. Despite the corrections made, there still are some false positives in the object detection, detecting the arm as a dangerous object.

When the other two risk levels are introduced (high and low), the automaton does not adequate its speed correctly, there are some situations where the robot reduces its speed in a 70 % when it should be reduced in only a 20 % and the other way around.

Finally, the response latency of the image processing application is represented in the figure 6. As it can be seen in the picture, the delay is always less than 0.2 seconds, therefore it can be considered to be a real time response. With this same frequency, the messages are sent to the robot, who reads the input buffer *buffer* every 0.2 seconds. A small delay can be observed in the reaction of the robot since a message to change the state was sent in order to change its speed. This is due to the controller delay and to the queues that are made every time that a frame is captured.

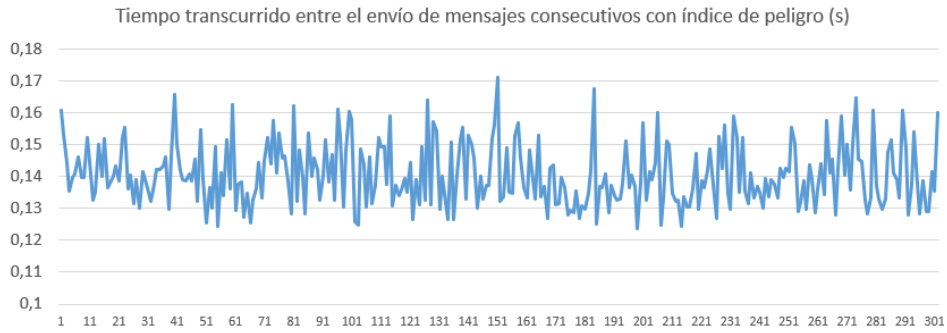


Figura 6: Tiempo requerido para el procesamiento de imágenes y la toma de decisión.

Bibliografía

[1] I.2. W.3. Seguridad Industrial.

Índice general

1. Introducción **1**

1.1. Estudio de los trabajos/tecnologías existentes 2

1.1.1. Robot industrial 2

1.1.2. Operación colaborativa Hombre-Robot 3

1.1.3. Sensores para la colaboración Hombre-Robot 5

1.1.4. Detección de objetos y posicionamiento con sensores . 7

1.2. Motivación del proyecto 14

1.3. Objetivos 15

1.4. Recursos/ herramientas empleadas 15

2. Arquitectura del sistema **19**

3. Análisis y tratamiento de imagen **27**

3.1. Adquisición de imágenes 28

3.2. Pre-procesado 33

3.2.1. Temperatura de color 34

3.2.2. Balance de blancos 35

3.2.3. Código de Matlab 36

3.2.4. Resultados del bloque de Pre-Procesado 37

3.3. Segmentación 38

3.3.1. Modelo de múltiples Gaussianas 39

3.3.2. Resultados obtenidos tras aplicar el Algoritmo de Múltiples Gaussianas 43

3.3.3. SSIM 44

3.3.4. Binarización de Mapa SSIM 47

3.3.5. Resultados parciales del Algoritmo SSIM 48

3.4. Reconocimiento 49

3.4.1. Reconocimiento de brazo robótico 49

3.4.2. Operaciones Morfológicas 53

3.4.3. Localización de coordenadas de objetos 55

3.4.4. Resultados parciales de la localización de obstáculos con Análisis de Blob 57

3.4.5. Resultados de localización del brazo del robot con vértices 61

3.4.6. Aproximación entre objetos. 62

3.5. Resultados y limitaciones del bloque de análisis de imágenes . 64

4. Comunicación entre Matlab y el Robot **69**

4.1. Protocolo de comunicación 70

4.2. Arquitectura de comunicación del sistema 71

4.2.1. Implementación del servidor (RAPID) 73

4.3. Implementación del cliente (Matlab) 74

4.4. Interfaz Gráfica de usuario de MATLAB 75

5. Procesado de datos y gestión de interrupciones por parte del Robot **77**

5.1. Lenguaje de programación Rapid 78

5.1.1. Interrupciones 78

5.2. Programa RAPID desarrollado 80

6. Resultados de los experimentos **85**

7. Conclusiones **89**

8. Futuros desarrollos **91**

A. Puesta en marcha del sistema **93**

A.1. Habilitar cámara 93

A.2. Uso de la interfaz de comunicación a través de GUI 94

B. Diseño de pieza **97**

C. Estudio Económico **99**

Bibliografía **105**

Índice de figuras

1. Diagrama de los bloques que componen el sistema desarrollado y la comunicación establecida entre ellos. 5

2. Etapas en el procesamiento de fotogramas. 6

3. Tiempo requerido para el procesamiento de imágenes y la toma de decisión. 7

4. Diagrama de los bloques que componen el sistema desarrollado y la comunicación establecida entre ellos. 10

5. Etapas en el procesamiento de fotogramas. 11

6. Tiempo requerido para el procesamiento de imágenes y la toma de decisión. 12

1.1. Hand Guidance, Hardware diseñado por Fanuc 4

1.2. Interacción física con DLR humanoid Justin [4] 5

1.3. Sensor ultrasónico HC-SR04 6

1.4. Velodyne ejemplo de sensor LIDAR [?] 6

1.5. Cámara ZED, tipo de cámara estéreo 7

1.6. Kinetic, ejemplo de cámara RGB-D 7

1.7. Diagrama de operación de diferenciación en el tiempo. [6] . . 8

1.8. Secuencia de tres imágenes, donde el flujo óptico es representado por el contorno de los píxeles entre el cuadro 0 y 1 así como el cuadro 1 y 2 [6] 9

1.9. Diagrama de bloques de método de detección de movimiento en primer plano a partir de la sustracción de fondo [6] 10

1.10. Distintas Gaussianas según la iluminación del píxel [16]. [7] . 11

1.11. Funciones Kernel tomadas a partir de unas mismas muestras, pero distinto ancho de banda [9]. 12

1.12. Diagrama de flujo del algoritmo propuesto por Zheng Yi [28] 13

1.13. Cadena de producción en la empresa china Changying Precision Technology Company con robots industriales de la compañía ABB. 16

1.14. Cámara D435 de la empresa Intel 17

2.1. Diagrama de los bloques que componen el sistema desarrollado y la comunicación establecida entre ellos. 20

2.2. Diagrama de flujo para iniciar conexión con la cámara y posterior toma de información. 21

2.3. Diagrama de flujo para el procesamiento de imágenes realizado por Matlab. 23

2.4. Diagrama de flujo para la comunicación entre Matlab y el Robot 25

3.1. Esquema representativo de la información de fotogramas de entrada y salida del bloque de toma y procesado de imágenes. 27

3.2. Diagrama de bloques con fases en la captura y análisis de imágenes. 28

3.3. Montaje de cámara estática fijada a la estructura del robot a partir de una pieza diseñada. 30

3.4. Dos imágenes tomadas consecutivamente con configuración inicial. Muy lenta, donde se observa como pasa de no haber nada a tener un brazo colisionando con el robot. 31

3.5. Comparativa de tiempo transcurrido para tomar cada imagen dentro de dos cadenas de 100 muestras con diferentes resoluciones. 32

3.6. Imagen con tonos anaranjados propios de una iluminación intensa 33

3.7. Imagen con tonos azulados propios de una iluminación débil . 33

3.8. Cada fuente luminosa emite una luz con distinto color o temperatura de color. Esta temperatura de color se mide en Kelvin y varía desde tonos rojizos a azules) [26] 35

3.9. Resultados Parciales obtenidos aplicando el balance de blancos aplicado a tres imágenes tomadas en situaciones de luz distintas. 37

3.10. Comparativa del resultado obtenido en la segmentación del brazo de la etapa posterior habiendo o no aplicado balance de blancos a la etapa anterior. En (1) aparece una segmentación ideal del brazo, en (2) la segmentación como resultado de aplicar balance de blancos y operaciones morfológicas y (3) igual pero sin aplicar balance de blancos. 38

3.11. Entrada y Salida del bloque de segmentación 39

3.12. Esquema aclaratorio para el funcionamiento del Modelo de Múltiples Gaussinaas [22] 40

3.13. Ejemplo de información memorizada típicamente como fondo. Ejemplo de una imagen de las 100 tomadas durante la ejecución del entrenamiento para determinar los valores de fondo. 41

3.14. Representación de información de entrada y resultado a la salida del bloque del algoritmo de SSIM en la aplicación . . . 45

3.15. Diagrama de bloques explicativo del Algoritmo SSIM en Matlab [29] . . . 46

3.16. Objetivo a alcanzar en la etapa de binarización del mapa SSIM. A la izquierda aparece un ejemplo de resultado obtenido a la salida del algoritmo SSIM y a la derecha el resultado de binarizar este resultado. . . . 47

3.17. Herramienta de Matlab para ajustar valores de color a las máscaras según histogramas. . . . 51

3.18. Mal funcionamiento de máscara de color al variar resolución de imágenes. . . . 52

3.19. Imagen ejemplo que se introduce al bloque de segmentación del brazo del robot. . . . 52

3.20. Resultado de aplicar a la figura 3.19 la máscara de color naranja diseñada para la pinza del robot. . . . 53

3.21. Resultado de aplicar a la figura 3.19 la máscara de color amarillo diseñada para la pinza del robot. . . . 53

3.22. Ejemplo de funcionamiento de operaciones morfológicas de dilatación y erosión. . . . 54

3.23. Etapas del proceso de segmentación de brazo del robot. . . . 55

3.24. Resultados de la localización de coordenadas de objetos dentro de una imagen con análisis de *Blob*. Para su representación se han dibujado bordes en las coordenadas encontradas. . . . 58

3.25. Resultado de localización de brazo del robot con método de análisis de *blob*. Sus coordenadas aparecen representadas en rojo y son inadecuadas pues esta recogiendo sección representada en verde sin ocupar. . . . 59

3.26. Objetivo de localización de brazo del robot, representado en rojo a alcanzar. 60

3.27. Aparece representado en blanco la segmentación del robot, en rojo la localización de sus coordenadas. Resultados obtenidos con el primer algoritmo diseñado específicamente para la aplicación. Se observa claramente su mal funcionamiento. . . 60

3.28. Resultados tras aplicar el segundo algoritmo que utiliza cinco vértices. Se observa que los resultados cumplen con los objetivos de este bloque de segmentación para el brazo orientado en las dos direcciones posibles. 62

3.29. Tiempo transcurrido entre mensajes con informe de riesgo enviado al robot. Se han utilizado 300 muestras. 64

3.30. Representación de etapas de análisis de imagen de entrada con tres obstáculos. Ejemplo de buen comportamiento del algoritmo. 65

3.31. Etapa final de localización de obstáculos en área de trabajo. . 65

3.32. Falsa alarma producida por cable de alimentación de la pinza del robot. 66

3.33. Correcto funcionamiento de la segmentación y localización pese a la aparición del cable de alimentación de la pinza. 66

3.34. Falsa alarma por confusión de la pinza por una sombra detectada con la máscara de color. 67

3.35. Ejemplo de fallo en el análisis de fotogramas por obstáculo no identificado. 68

4.1. Diagrama de capas del protocolo TCP/IP 71

4.2. Diagrama de bloques de elementos que intervienen en la comunicación TCP/IP. 72

4.3. Interfaz gráfica para el usuario 76

5.1. Diagrama de flujo del programa implementado en el controlador del robot 81

A.1. Pantalla para seleccionar que herramientas de RealSense SDK 2.0 descargar 94

A.2. Interfaz de usuario 95

Índice de tablas

1.1. Algoritmos de detección de movimiento en imágenes consecutivas tomadas con cámara estáticas. 14

3.1. Tabla representativa sobre posibilidades de configuración de imágenes de color suministradas por la cámara. Aparece representada las velocidad de captura posible en función del tamaño de la imagen tomada. 31

3.2. Tabla de resultados obtenidos del Modelo de Múltiples Gaussianas, al introducir imágenes del entorno capturadas en distintas condiciones de iluminación. 44

3.3. Resultados SSIM con distintas situaciones de iluminación. Mucho mejores que los obtenidos con el algoritmo de múltiples gaussianas. 49

3.4. Tabla de índice según distancia 63

5.1. Índice de riesgo. 77

5.2. Instrucciones de RAPID utilizadas para el manejo de interrupciones. 79

5.3. Instrucciones de RAPID utilizadas para el control del robot por el sistema de seguridad. 82

6.1. Resultados globales del funcionamiento del sistema 86

Capítulo 1

Introducción

La inteligencia artificial avanza, y cada vez más rápido. En particular, la industria española acababa el 2018 con cerca de 35.000 robots industriales trabajando a pleno rendimiento. Según la Federación Internacional de Robótica [11] se esperan instalar más de 1.7 millones de nuevos robots industriales en todo el mundo para finales de 2020. Se trata de un proceso de automatización sin precedentes, que no ha hecho más que empezar.

A su vez, es bien sabido que estos robots industriales pueden llegar a ocasionar graves accidentes por el impacto de su brazo ya sea al personal como con las instalaciones. Por ello, es lógico pensar que todo robot en funcionamiento deberá tener implementado su propio protocolo de seguridad. Actualmente la solución adoptada por la normativa de aplicación en toda la Unión Europea obliga a dotar a toda el área de alcance del robot de un perímetro de seguridad suficientemente dimensionado, el cual impida su acceso para todo usuario [24]. El robot tan solo podrá funcionar si el perímetro de seguridad se encuentra completamente desalojado. Con esta medida se nos plantea la duda sobre qué ocurría si se desprendiese alguna pieza de la plataforma y existiera posibilidad de impacto con ella, pero, no se encontrase ningún usuario para dar la orden de detección. Por otro lado, si se quiere realizar una modificación en el área de trabajo en la cual deba interrumpir un usuario, será necesario parar toda la línea de producción a la que pertenece dicho brazo y al detenerla lógicamente se produce una bajada en la eficiencia de ésta. En la última década ha surgido un nuevo modelo de robot con el propósito de aumentar la eficiencia y seguridad. Estos son conocidos como robots colaborativos o cobots y son capaces de compartir el espacio de trabajo con las personas. Representan una especie de robots seguros y cómodos para operar con el personal. Son capaces de adaptarse a la perfección a los empleados y al entorno en el que se encuentren.

En el presente trabajo se plantea un sistema para hacer posible esta colaboración Robot-Humano en los robots industriales IRB 120 de la compañía ABB

situados en el Laboratorio de Automatización de la Universidad Pontificia de Comillas (ICAI) a la vez que se mejora su seguridad.

En este primer capítulo se introducen todos los aspectos que han motivado la realización de este proyecto. Primero se hablará sobre la motivación de implementar visión artificial en la seguridad de un robot industrial; con necesidades de prestaciones cada vez mayores. Después se comentarán los diferentes tipos de soluciones que se han propuesto para implementar el sistema de seguridad automatizado.

1.1. Estudio de los trabajos/tecnologías existentes

Para realizar un correcto estudio sobre el estado actual de la cuestión, es preciso realizar un estudio de cada una de las partes involucradas en el presente trabajo. Se comenzará haciendo una breve introducción sobre la historia y actualidad de la robótica y más concretamente sobre la actividad colaborativa Robot-Humano. Se continuará con las posibilidades actuales para dotar a un robot de capacidad colaborativa con las personas y se citarán los sistemas de vigilancia en robots junto con los sensores que lo hacen posible. Finalmente, se profundizará en las técnicas de visión artificial y reconocimiento de objetos.

1.1.1. Robot industrial

La Organización Internacional de Normalización (ISO) [1] define Robot industrial como un manipulador reprogramable para usos múltiples, controlado automáticamente con tres o más ejes programables, ya sea fijo o móvil, para uso en aplicaciones de automatización industrial. En 1959 se desarrollaría el primer robot industrial por George Devol y Joseph Engelberger para que poco más de una década después se instalaran, en 1972, las primeras líneas de producción de robots en Europa. Los afortunados fueron Fiat en Italia, para soldaduras de puntos. Desde ese momento, se ha producido un crecimiento exponencial en el número de empresas que optan por automatizar las líneas de producción con robots industriales. Esto se debe en su mayoría, al aumento de la eficiencia y la disminución del coste para la empresa que suponen. En 2009 se lanzaría al mercado el robot industrial IRB 120 por ABB Suecia. Robot con el que se trabaja en el presente trabajo. Se trataría del robot más pequeño de la firma hasta el momento, con un peso de 25 kg. y manejando carga útil de hasta 3 kg. En la actualidad se encuentran funcionando en L'Oreal Canadá. Posteriormente, esta firma sacaría el IRBT 120 que se trataría de la versión rápida del anterior, pero con las mismas características de peso [2].

1.1.2. Operación colaborativa Hombre-Robot

Se entiende como robot colaborativo a aquel diseñado para la interacción directa con humanos en un espacio compartido con humanos [14]. El sistema de seguridad integrado que detiene el funcionamiento de un robot si este entra en contacto con una persona es una de las funciones que definen a los robots colaborativos en la actualidad. En 2008 UR (Universal Robots) diseñó y vendió el UR5, primer robot colaborativo, a Linatex, proveedor danés de caucho y plásticos técnicos para aplicaciones industriales. Con UR5 quedaría automatizada la supervisión de maquinaria de control numérico, trabajando por primera vez en la historia, simultáneamente en un espacio compartido con el personal de la empresa. Se pueden distinguir cuatro modos de funcionamiento cooperativo entre robot y personal, definidas a continuación:

Parada segura monitorizada

El robot se detendrá y permanecerá parado mientras algún individuo se encuentre dentro del espacio colaborativo. Existe la posibilidad de que el robot reinicie su trabajo de manera automática o manual cuando el área de trabajo se encuentre de nuevo desalojada. Este método de trabajo más que colaborativo se podría considerar de cooperación, pues pese a compartir un mismo espacio de trabajo, siempre será en tiempos diferentes. Se pueden dividir básicamente en dos. Mecanismos de seguridad con bloqueo, donde simplemente se bloquea el acceso al área de trabajo del brazo industrial mientras este se encuentra en funcionamiento y se consigue con jaulas o barreras de seguridad. Por otro lado, los mecanismos de seguridad sin bloqueo donde el robot se detendrá de manera automatizada en el instante en el que algún usuario entre en su zona de trabajo. Para conseguir una parada segura monitorizada sin bloqueo se suelen utilizar sensores como escáneres de láser, barreras fotoeléctricas y barreras sensibles o sistemas de visión artificial.

Guiado manual

Cuando la persona entra en el espacio de trabajo colaborativo, el robot se detiene y puede moverse únicamente si es guiado manualmente. Este tipo de operación exige una velocidad reducida de operación pues un fallo de cualquier tipo que ocasione un movimiento intempestivo del robot podría ser peligroso para los trabajadores. En octubre de 2017, OptoForce, principal proveedor en tecnologías robóticas de sensores de fuerza ofrecía una aplicación para los robots industriales KUKA de guiado manual. Con esta aplicación, los robots KUKA podrían ser guiados manualmente para moverse fácilmente y sin problemas en una dirección asignada y en la ruta seleccionada. Otro ejemplo de esta metodología de trabajo la encontramos en las

compañías Fanuc que disponen actualmente de un *hardware* de guiado manual. *Hand Guiadance* permite corregir la trayectoria del brazo de uno de sus robots ("CR-35iA") a partir de tan solo un par de piezas adicionales de hardware. Las principales ventajas que se encuentran son guiado intuitivo y que no necesita personal técnico con conocimientos previos sobre el robot pues se crean programas de forma rápida y sencilla.

Figura 1.1: Hand Guidance, Hardware diseñado por Fanuc

Limitación de fuerza y potencia

En este modo de trabajo la colisión entre el robot y la persona es posible, pero de ser así, el robot solo podrá ejercer una fuerza y/o potencia limitada. Es decir, pretenden reducir el riesgo limitando la fuerza o potencia que puede ejercer un robot en función de la zona del cuerpo donde se prevea el contacto. Es difícil de adaptar a cualquier sistema, pues mientras que los dos modos descritos previamente se podían implementar en prácticamente todos los robots industriales del mercado, este modo de operación requiere de robots específicamente diseñados para ello. Se consigue a partir de un diseño mecánico inherentemente seguro. Un ejemplo de robot con este modo de trabajo es PreciseFlex400/300, incluye características de seguridad que inhabilitan la potencia del motor cuando se encuentran una resistencia míni-

ma inesperado. Incluso trabajando a toda velocidad, su diseño seguro limita la fuerza en cuestión de milisegundos para cumplir con las normas de robot de colaboración ISO. S. Haddadin et al [4] proponen una serie de métodos de detección y reacción de colisión para reducir las fuerzas de contacto muy por debajo de cualquier nivel que sea peligroso para los humanos. Presentan evaluaciones de los impactos entre el robot y el brazo humano o el pecho hasta una velocidad máxima de 2.7m/s.

Figura 1.2: Interacción física con DLR humanoid Justin [4]

Control de velocidad y separación

Con control de velocidad y separación, la persona puede acercarse en todo momento y sin peligro al robot. La distancia entre ellos está controlada por el robot y la velocidad se adapta oportunamente. El robot es capaz de detectar obstáculos y reduce su velocidad o su trayectoria. Para ello, posee funciones de limitación de ejes y velocidades. Cuando no sea posible mantener la distancia de seguridad el robot se detendrá. En este caso, vuelve a existir la posibilidad de que ante una parada el robot pueda reiniciar su tarea automáticamente cuando haya desaparecido el riesgo.

1.1.3. Sensores para la colaboración Hombre-Robot

En esta sección se comentarán los tres tipos de sensores utilizados para obtener información sobre el área donde se encuentran que se han considerado más relevantes por la cantidad y calidad de información que proporcionan en relación con el precio.

Sensor ultrasónico

Sensor barato, pero con capacidad también reducida. Es compatible con toda superficie que tenga reflectividad acústica. Su funcionamiento consiste en emitir una onda ultrasónica en una dirección y esperar a que rebote contra el primer objeto que se encuentre en su camino y regrese para estimar la distancia en función del tiempo que ha tardado en recibir la respuesta. Su limitada resolución de rango y baja velocidad de respuesta hacen que no sea adecuado para tareas en las que se necesite una alta precisión como la colaboración en espacios reducidos entre robot y humanos.

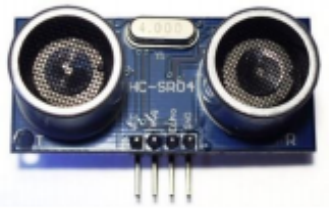


Figura 1.3: Sensor ultrasónico HC-SR04

Sensor láser y LIDAR

Son muy utilizados en la actualidad gracias a su precisión sobre todo en interiores. Abarcan desde simples sensores de rango que miden distancias en una única dirección, hasta sensores capaces de obtener información en 3D en áreas extensas de trabajo. Sus principales ventajas son buena resolución de ángulo y distancia y su elevada tasa de muestreo. Los LIDAR (Light Detection and Ranging) obtienen nubes de puntos del entorno con buena precisión y baja carga de datos, por lo que su velocidad de procesamiento es más alta. Funcionan calculando el tiempo en el que el haz de luz regresa al sensor tras ser disparado en una dirección, con ello se estima el objetivo más cercano en su trayectoria.

Figura 1.4: Velodyne ejemplo de sensor LIDAR [**?**]

Sensores de imagen

Dispositivos de excelencia en la robótica, han experimentado un avance en los últimos tiempos en los que se han desarrollado avanzadas tecnologías de procesamiento de imagen debido a su bajo precio y altas posibilidades de adaptabilidad y desarrollo. Tenemos las cámaras que son dispositivos de sobra conocidos, pero también podemos encontrar sensores RGB-D que capturan una imagen a color y también información sobre profundidad gracias a sensores de infrarrojos. Pese a su gran utilidad en interiores, el sensor infrarrojo es sensible a la luz solar y por esta razón esta tecnología no es aplicable en lugares abiertos.

Figura 1.5: Cámara ZED, tipo de cámara estéreo

Figura 1.6: Kinetic, ejemplo de cámara RGB-D

1.1.4. Detección de objetos y posicionamiento con sensores

En esta sección se hará un análisis de diferentes estudios realizados en los últimos años sobre la etapa de procesamiento de imágenes consecutivas con la intención de detectar nuevos elementos, tanto estáticos como dinámicos, dentro del área de trabajo. Nos limitaremos a estudios que traten con cámaras estáticas, pero, pese a trabajar con ellas, es habitual que la escena cambie constantemente debido a factores como el cambio en la iluminación, la aparición de sombras reflectadas, la aparición de elementos fuera del área de trabajo pero que aparecen en el fondo de la imagen, etc. Todos estos factores son considerados ruido y dificultan la detección de elementos en el área crítica. Aunque se han propuesto numerosos estudios que se citarán a continuación, el tema sigue siendo un reto. Los métodos de detección de movimiento se apoyan principalmente en tres principios: la diferencia temporal, el flujo óptico de una imagen y la sustracción de fondo.

Diferenciación temporal

El principio de detección de movimiento a partir de la diferencia temporal trata de encontrar movimiento calculando la diferencia entre dos bloques de fotogramas consecutivos tal y como podemos apreciar en el diagrama de bloques de la Figura 1.7. Este método tiene una complejidad baja, sin embargo, es muy sensible al umbral de error que se le asigne. Un umbral bajo causará mucho ruido pues será demasiado sensible a factores como el cambio en la iluminación o a las sombres entre otros. Por el contrario, un

umbral alto dejará objetos sin detectar. Este método también se ve afectado por la velocidad de los objetos en movimiento.

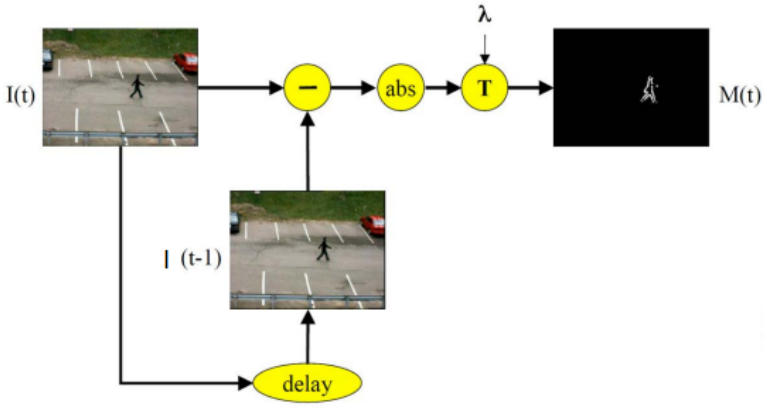


Figura 1.7: Diagrama de operación de diferenciación en el tiempo. [6]

Flujo óptico

Un recurso en el que se apoyan estudios como [25] para la detección de movimiento entre imágenes consecutivas, es el flujo óptico. Este método trata de determinar el patrón de movimiento existente entre dos imágenes consecutivas y su velocidad de movimiento como se puede observar en la 1.8: Secuencia de tres imágenes, donde el flujo óptico es representado por el contorno de los píxeles entre el cuadro 0 y 1 así como el cuadro 1 y 2. S.Blanco, et al [6] desarrollan un método basado en el cálculo del patrón de movimiento a partir de la estimación de las velocidades instantáneas que existen entre ellas. Las velocidades de los puntos se calculan mediante el desarrollo de Taylor de forma que el resultado quede en función de velocidades yx e yy. Para la ecuación que relaciona ambas velocidades se buscan valores de los parámetros que la hacen mínima. Para minimizar dichas ecuaciones, utilizan el método de los mínimos cuadrados o el método de Lucas-Kanade, que resuelve el sistema imponiendo unas restricciones considerando que el flujo situado en las proximidades del píxel sobre el que se trabaja es constante.

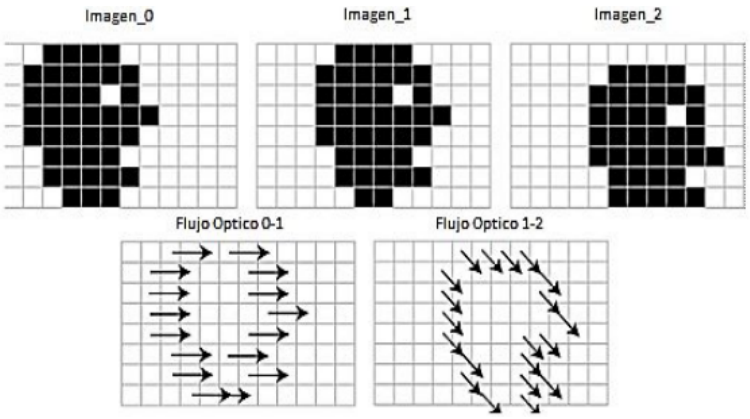


Figura 1.8: Secuencia de tres imágenes, donde el flujo óptico es representado por el contorno de los píxeles entre el cuadro 0 y 1 así como el cuadro 1 y 2 [6]

R. Nelson y Y. Aloimonos [18] también utilizan ciertas medidas de la divergencia del campo de flujo como señal cualitativa para evitar obstáculos. En este proyecto se demuestra como una cantidad denominada divergencia direccional del campo de movimiento 2D puede usarse como indicador confiable de la presencia de obstáculos en el campo visual de un robot que experimente un movimiento de rotación y translación generalizado.

Sustracción de fondo

Los algoritmos basados en la sustracción de fondo tratan de extraer objetos en movimiento en un primer plano que se desvían de un modelo de fondo mantenido y actualizado. Dividen dentro de una imagen lo que consideran fondo de lo que compone el primer plano para procesamiento posterior de la imagen para restarlo a toda nueva imagen que entra en el sistema como se aprecia en la 1.9. Todo modelo de fondo deberá tener la capacidad de evolucionar a cambios de fondo graduales y adaptarse a cambios de fondo repentinos. Se puede trabajar con cada píxel como variables aleatorias independientes modelando un color de fondo particular para cada uno o con la imagen como variable única aleatoria, modelando el fondo a partir de N imágenes del periodo de aprendizaje. En este proyecto nos centraremos en aquellos que tratan cada píxel como variable aleatoria e independiente. Es de esperar, que la complejidad del modelo se halle en la obtención de un modelo de fondo adecuado, por ello, a continuación se estudian los principales algoritmos de modelado de fondo.

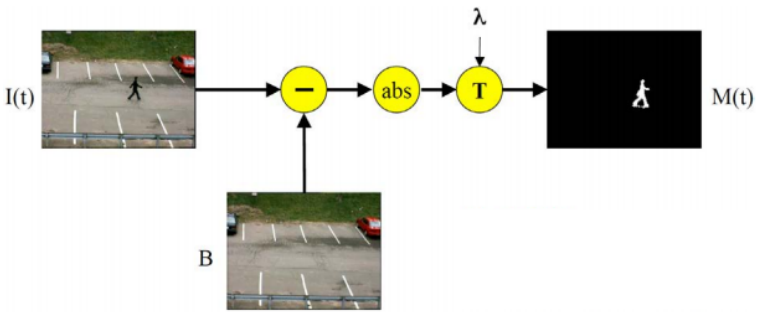


Figura 1.9: Diagrama de bloques de método de detección de movimiento en primer plano a partir de la sustracción de fondo [6]

- **Running Gaussian Average:**
  Se establece un modelo de fondo definido con una Gaussiana en cada píxel de la imagen. Este proceso de modelado se realiza durante el periodo de entrenamiento en el cual se asigna una función de densidad de probabilidad de color para cada píxel. La Gaussiana viene definida por la media (μ)que recoge los N valores observados como producto de los tres canales y su varianza (σ^2). Para determinar si un píxel pertenece o no al fondo se sigue el siguiente criterio:

$$\begin{aligned} |x - \mu| > k\sigma \rightarrow \text{Primer plano} \\ Resto \rightarrow Fondo \end{aligned} \tag{1.1}$$

  Con k como constante de precisión y con valor habitual entre 3 y 5. Pasado el periodo de entrenamiento se podrán actualizar los parámetros de la siguiente manera:

$$\begin{aligned} \mu_t &= \rho * x_t + (1 - \rho)\mu_{t-1} \\ \sigma^2 &= \rho(x_t - \mu_t)^2 + (1 - \rho)\sigma^2_{t-1} \end{aligned} \tag{1.2}$$

  Con x_t como valor del píxel de la imagen tomada, $/mu$ t y $/sigma_t$ como media y desviación estándar que caracterizan la función de densidad de probabilidad del píxel en ese mismo preciso momento, y $/ro$ es el parámetro de absorción que se refiere a la velocidad con la que se actualiza el modelo. Este algoritmo tiene una alta velocidad de procesado y requiere poca memoria. Wren et al [27] realizan un proyecto al que denominan PFinder y en el que tratan de determinar la silueta y postura de una persona apoyándose en trabajos como el de Martin Bichsel [5] en el que determina la aparición de personas a partir de su silueta con un modelo Gaussiano de sustracción de fondo. Este proyecto está diseñado para áreas en las que como máximo aparece un único

intruso en el área de trabajo y transforma la imagen de color (RGB) al espacio de luminancias y crominancias (YVU). Se trabaja con YVU para normalizar los cambios en la iluminación. Múltiples usuarios no causarán problemas en los algoritmos de seguimiento, pero si dificultan el sistema de reconocimiento de gestos.

- **Múltiples Gaussianas:**
  En este modelo estadístico se establece una Gaussiana por cada canal de color (RGB). Se pretende mejorar con respecto al modelo anterior la detección de objetos en secuencias en las que aparecen variaciones periódicas en el fondo. En un primer estudio Stauffer y Grimson [8] proponían este modelo para detectar objetos de primer plano mediante la suma ponderada de las tres funciones Gaussianas que tenía cada píxel. Su intención seria que las Gaussianas representasen tanto el fondo como el primer plano y se pudiera aplicar un criterio de ponderación para decidir cuales modelan el fondo y cuales el primer plano para que así pasaran a formar parte del fondo obstáculos estáticos y fuera del área de peligro. A modo de resumen, para adaptarse a cambios de color del píxel, cada píxel se modela con una media de entre 3 y 5 Gaussianas ponderadas según la frecuencia de aparición. Se considera que forman parte del fondo las Gaussianas con mayor frecuencia de aparición y por tanto ponderación más alta y el resto como de primer plano. Estos también se adaptan a cambios en la iluminación ya que cada Gaussiana modela el fondo con una iluminación determinada.

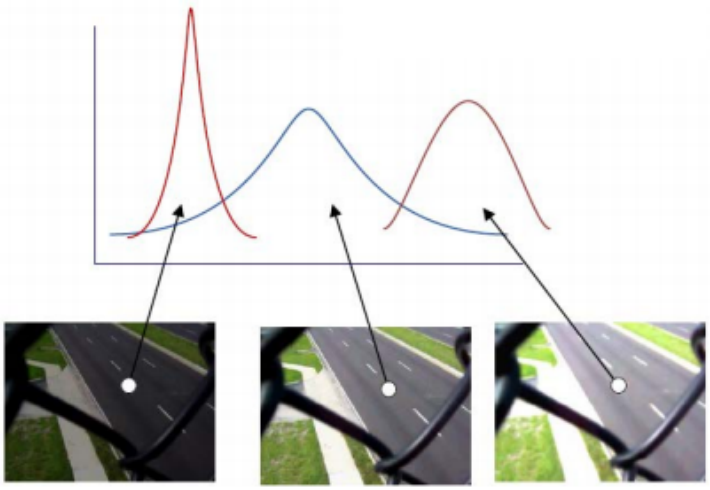


Figura 1.10: Distintas Gaussianas según la iluminación del píxel [16]. [7]

Una de las limitaciones de este modelo es que supone un fondo estático en escalas de tiempo corto. Es decir que no se adapta a escenas con

dinámica espaciotemporal, como el agua en movimiento el balanceo de los árboles o cualquier otro movimiento periódico de fondo. El proyecto de A. Chan et al [7] plantea una adaptación del anterior para escenas que contienen movimiento estocástico significativo.

- **Kernel Density Estimation (KDE):**

  Estima las funciones de densidad de probabilidad de un píxel basándose en la suma de funciones Kernel obtenidas a partir de algunas muestras. Una función Kernel es toda aquella que sea integrable, con área unitaria, y simétrica. La calidad de este método reside en el ancho de la función Kernel y esto se observa claramente en la figura 1.11 donde en la primera gráfica con un ancho de banda pequeño se obtiene una estimación demasiado variable, obteniendo picos mayores que en cualquier función de probabilidad que las componen. Por el contrario, en la segunda, un ancho demasiado grande enmascara la función de probabilidad. Finalmente, la última gráfica representa el resultado de trabajar con un ancho de banda óptimo.

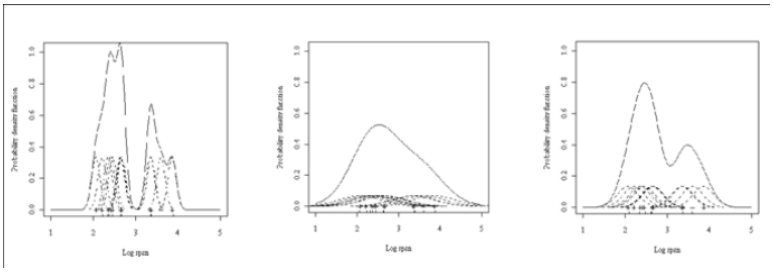


Figura 1.11: Funciones Kernel tomadas a partir de unas mismas muestras, pero distinto ancho de banda [9].

  Elgammal y Duraiswami [9] utilizan técnicas generales de KDE pues considera que al estimar una función de probabilidad directamente a partir de los datos sin ninguna suposición sobre las distribuciones subyacentes, se adapta mejor a cualquier escenario. Es decir, las funciones de densidad de probabilidad asociadas con el fondo y el primer plano pueden variar de una imagen a otra y, en general, no tendrán una forma paramétrica conocida.
  En ambos proyectos de Elgammal y Duraiswami [9] [10] utilizan el método de Kernel gaussiano por su simplicidad y suavidad. Teniendo en cuenta, que elegir la función gaussiana como núcleo, es diferente a ajustar la distribución a un modelo gaussiano o distribución normal. En este último, la distribución normal solo se usa como una función para ponderar los puntos de datos. A diferencia del ajuste paramétrico de una mezcla de gaussianas, la estimación de la densidad del núcleo

es un enfoque más general que no asume ninguna forma específica para la función de densidad.
Finalmente, otro trabajo interesante es el de Zheng Yi [28] que propone combinar la diferencia temporal y la sustracción de fondo para localizar objetos en movimiento en un primer plano. Se introduce un método de diferencia temporal para obtener la imagen de la diferencia asignando un umbral pequeño que como se ha visto, introduce mucho ruido, pero los contornos de los objetos en movimiento se conservan muy bien. Tal y como muestra la figura 1.13, esta imagen se combina realizando una operación "Lógica Y" con la imagen de primer plano obtenida aplicando un método de sustracción de fondo. Con esta operación se consigue eliminar por un lado el ruido de la imagen de diferencia temporal y también la sombra obtenida en la imagen de primer plano.

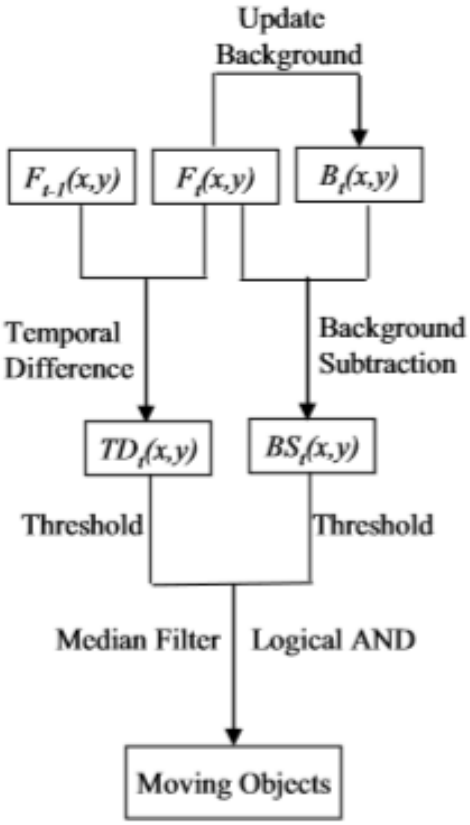


Figura 1.12: Diagrama de flujo del algoritmo propuesto por Zheng Yi [28]

Por último, Se ha recogido en la tabla 1.1 todos los estudios comentados con el algoritmo para detección de elementos dentro de imágenes, utilizado en cada uno de ellos. Se observa como los métodos de sustracción

de fondo son los típicamente empleados para resolver este problema.

	Flujo de movimiento	Sustracción de fondo	Diferenciación temporal
Zheng, 1999 [8]		✓	✓
M. Bichsel, 1996 [25]		✓	
C. Wren, et al. [16]		✓	
A.Elgammal, et al. [27]		✓	
A.Elgammal, et al. [5]		✓	
A. Chan, et al [18]		✓	
C. Stauffer y W. Grimson [6]		✓	
M.Watanabe et al. [4]	✓		
G.E.Silva Blanco, et al. [3]	✓		
R. Nelson y Y. Alaoimonos [12]	✓		
10	3	7	1

Tabla 1.1: Algoritmos de detección de movimiento en imágenes consecutivas tomadas con cámara estáticas.

1.2. Motivación del proyecto

El presente trabajo se centra en ofrecer un sistema de seguridad totalmente automatizado, resistente a la aparición de obstáculos en el recorrido del propio brazo, resistente a cambios de iluminación y presencia de sombras y adaptable a cualquier entorno de trabajo. Se ha diseñado para ser ejecutado en paralelo a cualquier aplicación con la que esté trabajando el robot como el sistema implementado por Lucía Díaz en un proyecto anterior [13] y es por ello que nuestro sistema está compuesto por los mismos elementos que el de ella, una cámara RGB-D, una aplicación en MATLAB y un robot industrial ABB IRB120.
El sistema de seguridad automatizado con control de velocidad y separación que se ha propuesto en el presente trabajo pretende proporcionar a una cadena de producción industrial un aumento en la eficiencia haciendo posible

la actividad colaborativa con humanos. Por otro lado, se puede adaptar a cualquier robot o cámara porque se trata de un sistema de visión artificial basado en un PC, es decir, independiente de la cámara y el robot industrial con la única limitación de la máscara de color.
Como sensor se ha utilizado una cámara RGB-D, pues por un lado ha sido el sensor escogido por otras compañeras para otros trabajos sobre el mismo Robot y por otro lado la información proporcionada, es suficiente para las necesidades del proyecto junto con un precio asequible.

1.3. Objetivos

El objetivo principal de este proyecto es la descripción e implementación de un sistema que proporcione seguridad a los robots ABB 120 instalados en el Laboratorio de Automatización de la Universidad Pontificia de Comillas (ICAI). Mediante una cámara RGB-D, el robot deberá ser capaz de distinguir personas u obstáculos (estáticos o en movimiento) en el área de trabajo y detenerse o alejarse para evitar posibles colisiones.
Se podría afirmar que este proyecto supondrá un primer paso para permitir el trabajo colaborativo entre los robots ABB 120 y los estudiantes de ICAI. Sin embargo, es importante destacar que dado que el software de detección será programado de forma independiente al robot, el alcance del proyecto será mayor, y podría ser implementado en distintos robots industriales.
Es decir, se implementará un sistema compuesto por una aplicación en MATLAB capaz de procesar imágenes de color y profundidad para dotar a cualquier robot de inteligencia artificial suficiente como para decidir si una trayectoria es segura, ser capaz de alejarse de objetos cuyo movimiento no sea seguro, o incluso como último paso, se pretende si es posible, dotar al robot de la capacidad de buscar rutas alternativas en las que no exista peligro de impacto. Cabe destacar que dichos robots ya están dotados de cierta visión artificial gracias a un software implementado por dos de mis compañeras de la Escuela. Estos robots disponen de un sistema que les proporciona la capacidad de ordenar piezas Lego en una cuadrícula según tamaño y color gracias a la misma cámara con la que se ha realizado este proyecto.

1.4. Recursos/ herramientas empleadas

En este punto, se realizará un breve resumen acerca de las herramientas, tanto software como hardware, utilizadas para la realización del proyecto.

IRB 120, robot industrial ABB

ABB es una empresa líder mundial en ingeniería eléctrica y automatización. La compañía es el producto de la unión en 1988 de Asea y BBC. Actualmente la sede central se encuentra en Zúrich (Suiza).

Tiene una gran oferta industrial desde interruptores de iluminación, hasta robots industriales, grandes transformadores eléctricos o sistemas de control capaces de gestionar grandes redes eléctricas o industrias.

Como ya se ha mencionado antes, ABB ofrece a sus clientes robots industriales con los que se mejora la productividad de las empresas, cuenta con una oferta de más de 25 tipos de robots distintos para las diferentes utilidades que se le puede dar en cada tipo de industria. Este gran abanico de posibilidades la convierte en la empresa líder de suministro de robots en el mundo.

Figura 1.13: Cadena de producción en la empresa china Changying Precision Technology Company con robots industriales de la compañía ABB.

El robot ABB IRB 120 es el mas pequeño y se puede utilizar para muchas aplicaciones pesa solamente 25 kg y puede manipular hasta 3 kg. con una área de trabajo de 580 mm. Su relación coste-efectividad, le permite ser una opción segura para conseguir altas producciones con una baja inversión. Será el robot al que se referirá todo el proyecto pues es sobre el que se ha implementado la aplicación de seguridad.

RobotStudio

El software de programación *offline* RobotStudio permite programar los robots en un ordenador sin necesidad de parar la producción. También se pueden preparar los programas de los robots anticipadamente, lo que implica un aumento de la productividad. Se compone de una copia virtual exacta de ABB del controlador utilizado para controlar los robots. Se utiliza para simulaciones reales.

Intel RealSense Depth Camera D435

Como sensor para captar el entorno de trabajo se ha escogido el modelo D435 de la empresa intel. Para la captación de imágenes de profundidad utiliza visión estéreo. Es alimentada por USB y consiste en un par de sensores de profundidad, sensor RGB y proyector infrarrojo. Llega a obtener fotogramas con una resolución de 1280x720. También tiene incluido un procesador de señal para modificar ajustes de la imagen y escala de datos de color. El proyector de infrarrojos lo utiliza para iluminar objetos y mejorar los datos obtenidos en la imagen de profundidad.

Figura 1.14: Cámara D435 de la empresa Intel

Matlab

MATLAB (abreviatura de MATrix LABoratoy) es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Sus prestaciones más comunes son la manipulación de matrices, representación de datos y funciones, implementación de algoritmos y creación de interfaces de usuario. Finalmente es un *software* muy popular en universidades y centros de investigación. Es por todo ello, que se ha decidido esta plataforma para el desarrollo de captura y análisis de imágenes.

Capítulo 2

Arquitectura del sistema

Una vez introducido el contexto tecnológico del que se parte y habiendo hablado sobre los diferentes aspectos que han motivado el desarrollo de este sistema de seguridad, en este capítulo se procederá a describir la arquitectura general del sistema, todo esto sirve como introducción para exponer el funcionamiento general de la aplicación y en particular el de cada bloque que constituye el sistema. A su vez, se comentará brevemente la interacción existente entre ellos y con más detalle en los capítulos 3 y 4.

Se recuerda que la aplicación de seguridad a diseñar, tiene como objetivo ser capaz de detectar movimiento dentro del área de trabajo del robot, identificar el movimiento ya sea del robot como de cualquier otro elemento así como el grado de peligro de colisión del brazo robótico, según la distancia. Finalmente comunicarle esta situación de riesgo al robot para que este disminuya su velocidad, se detenga o continúe su funcionamiento con normalidad. Para ello necesitamos un sensor que capte la situación del entorno de trabajo, un programa que sepa interpretar y sacar conclusiones de la información obtenida por el sensor y finalmente una comunicación con el controlador del robot para transmitirle la información del entorno. Todo ello funcionará en paralelo a un programa de automatización de tareas para el robot como podría ser el de ordenamiento de piezas según color [8]. En la figura 2.1 se representan los cuatro bloques que forman el sistema y la comunicación que se establece entre ellos. Las partes del sistema que aparecen representadas son:

- **D435:** Cámara encargada de capturar la información de la zona de trabajo del robot, y enviarla al programa de Matlab. Para ello, se establece una comunicación bidireccional con Matlab gracias al *driver* de Intel RealSense, esta herramienta permite al usuario acceder a la mayoría de las funciones de la cámara a través de una interfaz de usuario simple. Matlab enviará a la cámara la información de configuración y

solicitará información de los *frames* capturados. La cámara responderá con la información solicitada. Todo ello aparece detallado en la sección 3.1.

- **Matlab:** Plataforma encargada de realizar todo el procesamiento de la información contenida en cada fotograma, detallado en el capítulo 3. A su vez, se comunicará con el controlador IRC5 a partir de una comunicación TCP/IP explicada con mayor detalle en el capítulo 4.
- **IRC5 y IRB 120:** RobotWare 6.0 es un controlador integrado en el FlexPendant, facilita todas las necesidades para la programación y uso del robot. Realmente, se trata de una parte más del robot, pero se suministra de manera independiente [21]. El programa desarrollado para establecer la comunicación aparece explicado en el capítulo 5.

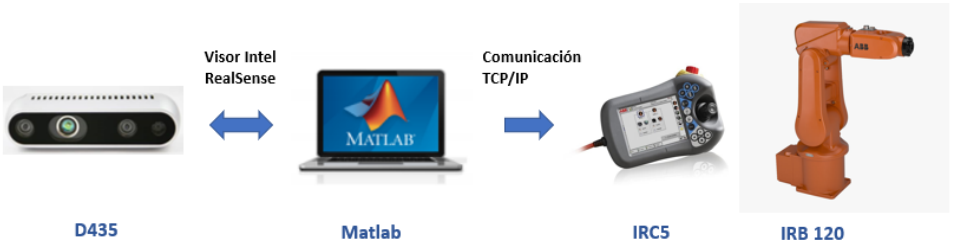


Figura 2.1: Diagrama de los bloques que componen el sistema desarrollado y la comunicación establecida entre ellos.

El funcionamiento del sistema se puede resumir a partir de tres diagramas de flujo. En el primero, aparece representada la comunicación entre la cámara y Matlab. En segundo lugar el diagrama de flujo representado en la figura 2.3, resume los pasos para el procesamiento de las imágenes y la determinación del índice de riesgo. Por último, la comunicación entre Matlab y el robot para comunicarle la situación de riesgo o peligro, representado en la figura 2.4.

Como ya se ha comentado, las tareas realizadas por Matlab para establecer y obtener información de la cámara aparecen representadas en la figura 2.2. Se observa como el primer paso es establecer los parámetros de la configuración como por ejemplo podrían ser; la resolución de imágenes o el tipo de imágenes (RGB, profundidad, infrarrojos). La discusión sobre la configuración utilizada en el proyecto aparece con más detalle en la sección 3.1. Posteriormente, se solicita la conexión con la configuración decidida como parámetro de entrada, para ello se realiza una llamada a la biblioteca compilada del visor Intel RealSense, esta petición aparece representada como mensaje tipo (1). Es importante tener en cuenta que trabajando con el sensor D435, no se puede abrir más de una conexión en paralelo. Por tanto solo se establecerá la

comunicación si no hay otra conexión establecida y los parámetros de configuración están en un orden y valor adecuado. A partir de ese momento, cada vez que se requiera información del entorno de trabajo, se le solicitará al sensor (mensaje (2)) y este responderá con la información del entorno (mensaje (3)), este paso de capturar imágenes se realizará en la primera etapa del procesamiento de imágenes. Finalmente, cuando se desee cerrar la comunicación (mensaje (4)) se deberá informar al visor para que este cierre la conexión con la cámara. Es importante destacar que tal y como se ha diseñado la aplicación, la configuración de la cámara es fija y la conexión no se cierra hasta que no se apaga el sistema, esto ha sido diseñado así para conseguir una velocidad mayor; de forma que si se deseara cambiar la configuración, es decir, tipo de fotogramas obtenidos, habría que cerrar y volver a abrir la conexión, perdiendo un tiempo medio aproximado de 0.5 segundos.

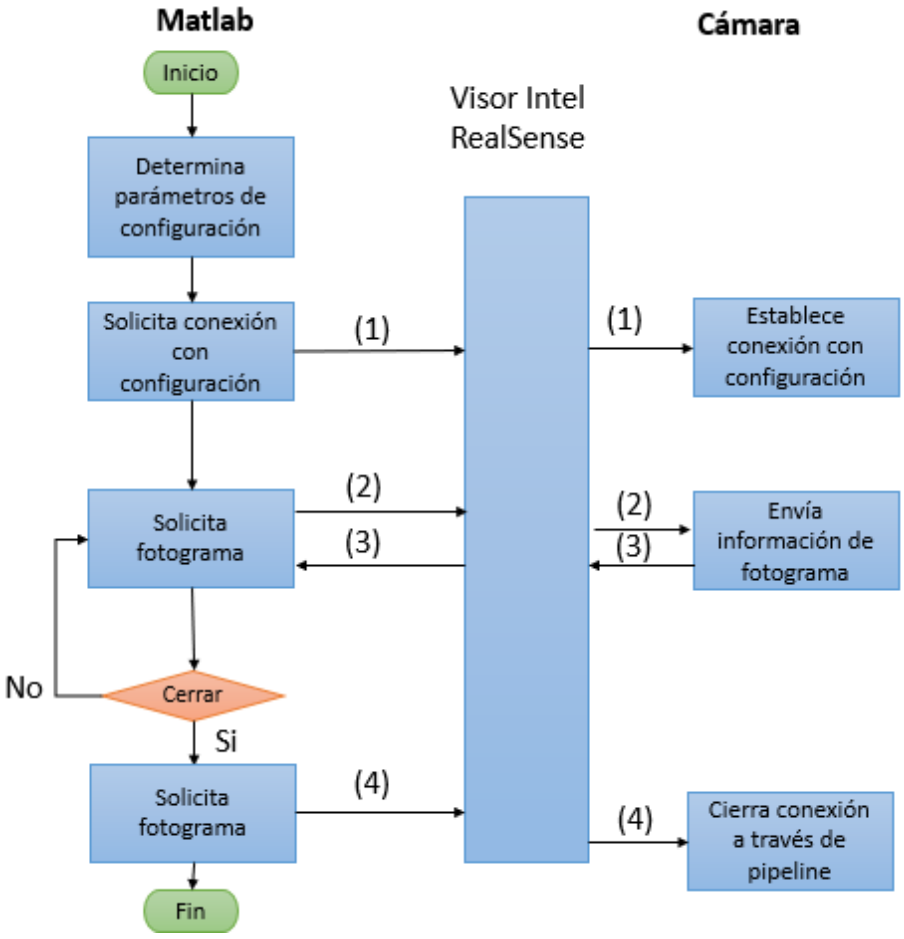


Figura 2.2: Diagrama de flujo para iniciar conexión con la cámara y posterior toma de información.

El siguiente flujo de tareas, ilustrado en la figura 2.3 constituye los pasos a realizar para el procesamiento de imágenes de entrada al sistema, tiene como objetivo determinar un nivel de riesgo a partir del estudio de la información contenida en cada imagen. Esta etapa se realiza íntegramente en Matlab, y no necesita el apoyo de ningún otro elemento o plataforma. Una vez obtenida una imagen a partir del proceso explicado en la figura 2.2, se procede a la etapa de pre-procesado para eliminar en cierto grado el ruido contenido en la imagen con el objetivo de aclarar la información contenida en ella para los pasos posteriores, este paso aparece explicado en la sección 3.2. El siguiente paso depende del estado del sistema. El sistema puede encontrarse básicamente en dos estados; entrenamiento o en marcha. El entrenamiento tiene como misión, memorizar un primer modelo de fondo al arranque del sistema para poder después detectar objetos con movimiento, para más información consultar la sección 3.3. Si se encuentra en este estado simplemente asume que está cogiendo una imagen representativa del fondo y modela los valores de fondo, por el contrario, si se encuentra en marcha procederá a la segmentación del área ocupada por el robot y por los obstáculos para posteriormente calcular la distancia existente de seguridad entre ellos y poder determinar un índice de riesgo. Este proceso de análisis de imágenes se repite en bucle durante toda la ejecución del sistema.

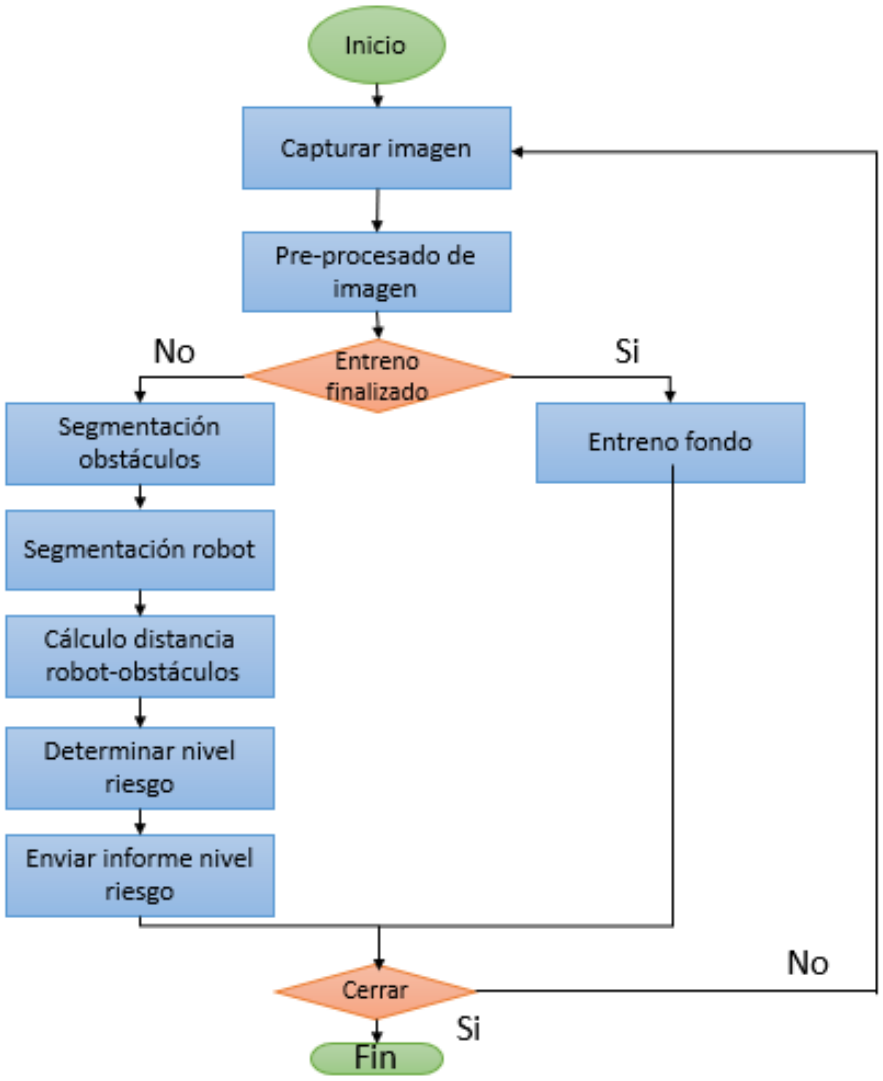


Figura 2.3: Diagrama de flujo para el procesamiento de imágenes realizado por Matlab.

Una vez explicados los pasos seguidos para la adquisición y tratamiento de imágenes, se detallará el otro diagrama de flujo del sistema representado en la figura 2.4, en éste entra en juego la comunicación Matlab-Robot y los pasos para traducir la información de un fotograma a un nivel de riesgo y así comunicárselo al controlador del robot industrial (mensaje 3). Como se verá con mayor detalle en el capítulo 4 se establece una comunicación TCP/IP entre Matlab y el controlador del robot. En un primer paso, el servidor abre el *socket* y lo conecta a la red local del laboratorio, pasa a esperar la solicitud de conexión por parte de un cliente y cuando el programa de Matlab

solicite la conexión se pasará a la etapa de entrenamiento del fondo para posteriormente poder detectar objetos en movimiento sobre este primer plano. Durante la etapa de entreno, el controlador tendrá que dirigir el brazo hacia un área no captada por la cámara para que no se interrumpa el periodo de entreno y permanecer a la espera de recibir un mensaje que le notifique que ha finalizado el entreno (mensaje 2). Cuando el entreno finaliza y se recibe el mensaje de confirmación, comienza a ejecutarse el programa principal, que puede ser cualquiera en función de la tarea automatizada que se esté realizando con él en paralelo al sistema de seguridad. En paralelo, Matlab estará constantemente captando información del entorno con el sensor, procesándolo y enviando informes de seguridad al robot (mensaje 3). El controlador a partir de un sistema de interrupciones leerá los informes y actuará al respecto si es necesario deteniendo el movimiento o bien continuará las tareas del programa principal si el informe indica que no se ha detectado ningún riesgo de colisión.

A partir de este punto, la memoria se centrará en el desarrollo extendido de los bloques citados en este apartado. En los capítulos siguientes se comentará como se han adecuado cada uno de ellos con la intención de cumplir el objetivo final de este TFG con un correcto funcionamiento. En el capítulo 6 se mencionan los resultados de la aplicación obtenidos junto con las conclusiones y futuros desarrollos. Finalmente esta memoria acaba con un capítulo en el que se detallarán los pasos a seguir para la puesta en marcha del sistema.

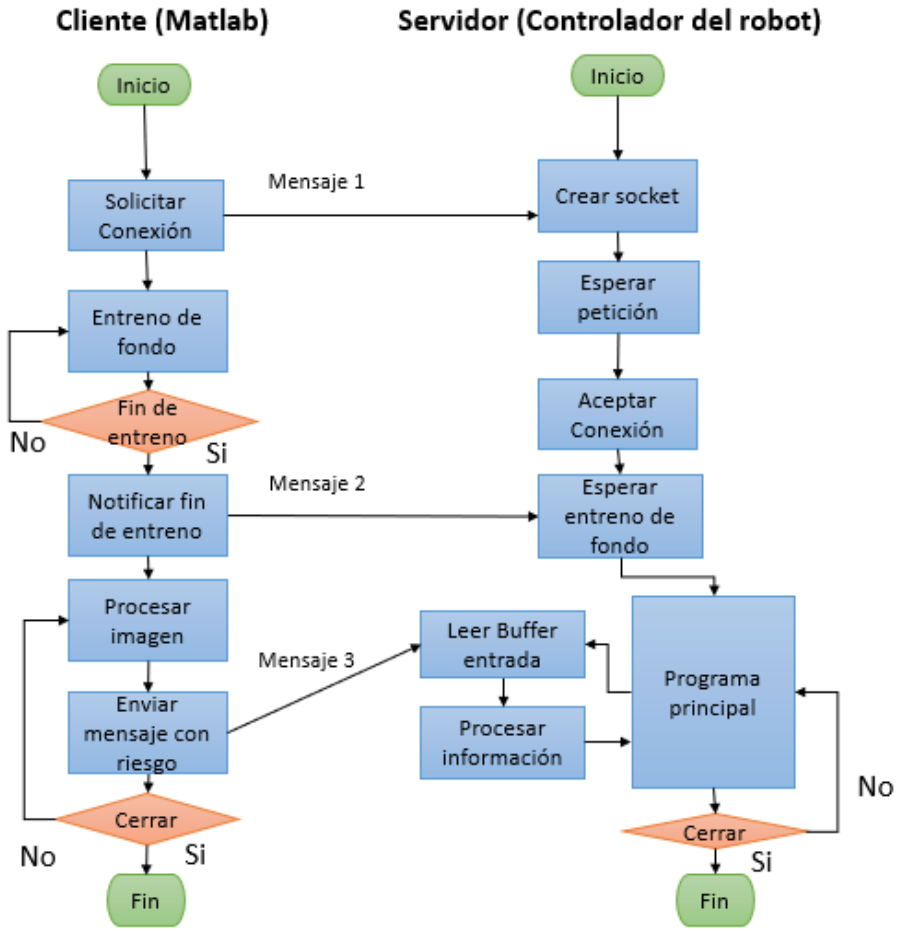


Figura 2.4: Diagrama de flujo para la comunicación entre Matlab y el Robot

Capítulo 3

Análisis y tratamiento de imagen

En este capítulo se describen con detalle cada una de las etapas de adquisición y procesamiento de las imágenes de entrada al sistema. El procesamiento tal y como se expone en el capitulo 2 se realiza desde Matlab. Como aparece representado en la figura 3.1, en este capitulo se explica el bloque de la aplicación encargado de, a partir de una imagen de entrada con información del entorno de trabajo del robot, detectar sobre ella objetos en movimiento y determinar su probabilidad de colisión a partir de un índice de colisión. Los objetos con movimiento constituirán el primer plano de la imagen y el índice determinará si el robot se encuentra en una situación con probabilidad de colisión o si por el contrario puede seguir trabajando con normalidad.

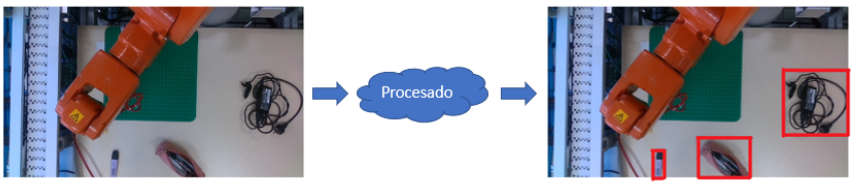


Figura 3.1: Esquema representativo de la información de fotogramas de entrada y salida del bloque de toma y procesado de imágenes.

Las fases del procesamiento de imágenes son básicamente seis: adquisición de imágenes, pre-procesado, segmentación del área ocupada por elementos del primer plano o obstáculos, segmentación del área ocupada por el robot, localización de vértices de las figuras y cálculo de distancias entre las figuras caracterizadas como obstáculos y robot para finalizar asignando a cada situación un nivel o índice de riesgo (el riesgo crecerá proporcionalmente al

valor del índice). Estas fases se explicarán con mayor detalle en cada una de las secciones que conforman el presente capítulo y aparecen representadas en la figura 3.2.

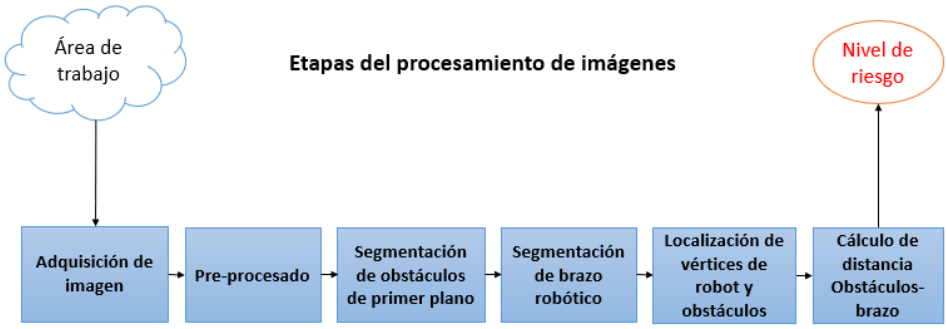


Figura 3.2: Diagrama de bloques con fases en la captura y análisis de imágenes.

3.1. Adquisición de imágenes

El primer paso es la toma de información sobre el entorno de trabajo. Para ello, se obtiene una imagen a partir de una cámara RGB-D estática, es decir, situada en un punto fijo a partir de una pieza, en la figura — se muestra el montaje realizado. Este tipo de dispositivos detectan profundidad en asociación a una imagen RGB, aumentando la información de una imagen convencional con datos relacionados con la distancia de cada objeto al sensor. Para acceder a la cámara se necesita el visor IntelRealSense D435 con el que se pueden acceder directamente a las imágenes capturadas por la cámara así como a la configuración y control de la cámara. IntelRealSense SDK 2.0 admite varios *wrappers* que hacen posible su acceso desde plataformas como Matlab, así como la modificación de su propia configuración. Como es bien conocido, un *wrapper* es una capa de código que traduce la interfaz existente en una biblioteca, en una interfaz compatible, con el fin de habilitar la interoperabilidad entre lenguajes de programación. La forma de acceder y utilizar esta biblioteca se encuentra explicada con mayor detalle en el capítulo 9 sobre la puesta en marcha del sistema. Algunas de las posibilidades de configuración de la cámara para el usuario son:

- **Formato** Se puede cambiar el formato de la información recibida. Se puede configurar para obtener entre otros, imágenes en formato RGB, YUYV u obtener imágenes que representen en cada píxel la distancia de la cámara a la sección representada por este píxel. La distancia es

calculada a partir de la disparidad obtenida entre los dos sensores que posee la cámara, efecto similar al realizado por el cerebro de un humano que posee dos ojos y la capacidad de conectar las dos imágenes percibidas por cada ojo para obtener una visión 3D. Todas estas posibilidades se encuentran dentro de la función `format.m` en el *wrapper* para Matlab.

- **Añadir metadatos a la imagen:** Con esta opción a los fotogramas se les pueden aplicar filtros de ganancia, exposición o incluso balance de blancos directamente desde la cámara a las imágenes. Estas oportunidades aparecen dentro de la clase definida `frame_metadata_value()` en el *wrapper* para Matlab.

- **Filtro para rellenar huecos:** Cuando se trabaja con imágenes en blanco y negro, existe la posibilidad de que la cámara de forma automática rellene los huecos. Esto significa que todo píxel con propiedades muy distintas a las de su alrededor, sean modificados. Esta posibilidad se puede encontrar dentro de la clase definida `hole_filling_filter()` en la librería del *wrapper* para Matlab.

- **Tipo de *stream*:** Existe la posibilidad de especificar el tipo de imagen que se quiere obtener. Entre sus posibilidades encontramos la imagen de color, infrarrojo, profundidad, con estabilización *gyro* es decir, con corrección de distorsión y alguna otra posibilidad más.

- **Dominio de la marca de agua:** Cada fotograma tiene una marca de tiempo, para identificar cuando se realizó la captura. Esta marca se puede sincronizar con el reloj del hardware de la cámara, con el reloj del sistema que se está utilizando (en nuestro caso Matlab) o con un sistema de cuenta.

- **Compensación de luz de fondo:** Algunas de las muchas posibilidades para aplicar a las imágenes son cambiar contraste, añadir ganancia, modificar la exposición, eliminar sombras, matizar la imagen o saturarla, opciones de auto exposición para ignora la luz de un único punto y que se centre en analizar la fotografía completa. Con esta última posibilidad se consigue algo similar al resultado observado cuando con un *smartphone* se toca un punto de la pantalla para cambiar el enfoque. En total existen 42 opciones diferentes para modificaciones en la luz de fondo de las imágenes. Todas estas opciones están dentro de la función `option()` en la librería del *wrapper* para Matlab.

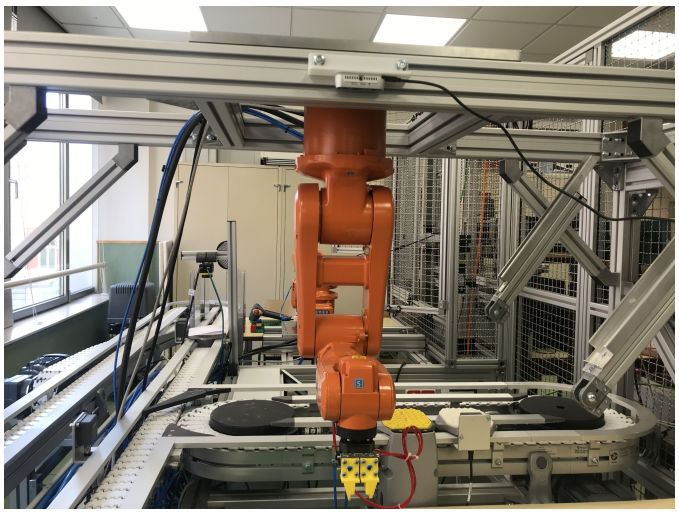


Figura 3.3: Montaje de cámara estática fijada a la estructura del robot a partir de una pieza diseñada.

Con la configuración preestablecida la cámara obtiene dos imágenes, una de profundidad y otra de color. Las imágenes de color con tamaño 640x480, a una velocidad de 30 fotogramas por segundo. La figura 3.4 demuestra como esta configuración no es adecuada. En ella aparecen representadas dos imágenes capturadas consecutivamente sin añadir procesamiento de ningún tipo entre su toma. Es decir, las únicas instrucciones ejecutadas en Matlab ha sido el necesario para la captura y posterior guardado en formato .png. Se han elegido estos dos frames como ejemplo para observar como sería imposible que el sistema generase una respuesta en un intervalo de tiempo aceptable, pues en la primera no aparece ningún obstáculo y en la inmediatamente posterior ya hay un obstáculo colisionando con el brazo, sin dejar margen para el procesamiento y envío de respuesta se ha producido un choque. Para mejorar la velocidad se ha cambiado la configuración inicial.

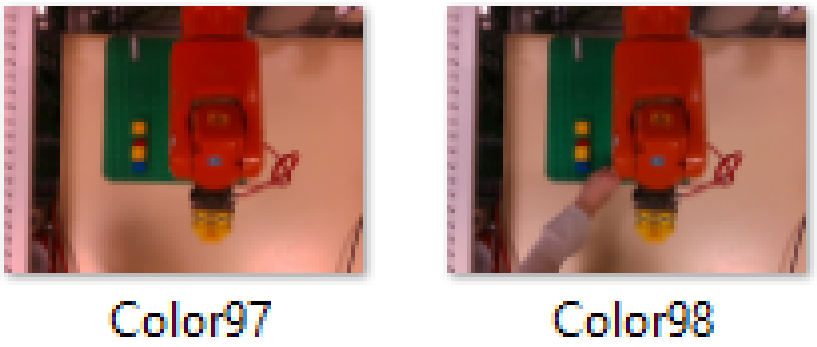


Figura 3.4: Dos imágenes tomadas consecutivamente con configuración inicial. Muy lenta, donde se observa como pasa de no haber nada a tener un brazo colisionando con el robot.

Es preciso mencionar que, pese a la gran cantidad de opciones de configuración expuestas, y que podrían ser útiles para la aplicación del proyecto, se han modificado tan solo dos de ellas (la resolución de imágenes y la velocidad de disparo). Esto, se debe a que cuando se realizó esta parte del proyecto, las versiones del visor RealSense disponibles no incluían la gran mayoría de posibilidades comentadas anteriormente y propias de la última versión descargada salida en este último mes de mayo. Las posibilidades de resolución y velocidad de disparo existentes en la versión 5.11.1.100 del visor aparecen representadas en la tabla 3.1.

Resolución	Velocidad de disparo (frames por segundo)		
	15	30	60
424x240	✓	✓	✓
640x480	✓	✓	
1280x720	✓		

Tabla 3.1: Tabla representativa sobre posibilidades de configuración de imágenes de color suministradas por la cámara. Aparece representada las velocidad de captura posible en función del tamaño de la imagen tomada.

La resolución de la cámara se ha adecuado teniendo en cuenta que el sistema debe generar respuestas en tiempo real, esto supone capturar y procesar la entrada, obteniendo un resultado dentro de un periodo de tiempo lo suficientemente pequeño como para ser considerado oportuno. Siempre y cuando la resolución sea lo suficientemente grande como para no obtener errores en el procesado de imagen por falta de información. En la figura 3.5 aparece representada la velocidad de computo para la obtención de estas imágenes con

resoluciones 424x240 y 640x480 representadas en azul y naranja y velocidades de 60 y 30 *frames* por segundo respectivamente. Para la realización de este gráfico se han usado 100 muestras y se aprecia con facilidad la diferencia de tiempos. El tiempo aumenta en un 50 % pasando de una velocidad media aproximada de 0.1 segundos a una velocidad de 0.05 segundos con la nueva configuración de la cámara. Esta diferencia de velocidad se debe a que la información que debe capturar es menor y por tanto se brinda la posibilidad de aumentar el numero de *frames*. También se prescinde de las imágenes de profundidad pues lógicamente capturar esta información consume tiempo adicional, en un principio se trabajaba tomando este tipo de imágenes y el tiempo de obtención de imágenes alcanzaba los 0.5 segundos.

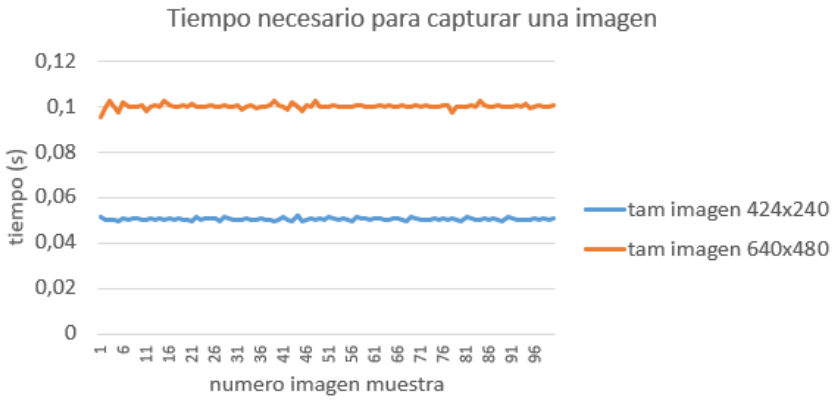


Figura 3.5: Comparativa de tiempo transcurrido para tomar cada imagen dentro de dos cadenas de 100 muestras con diferentes resoluciones.

Finalmente con todo ello, se ha escogido un tamaño de imagen de 424x240 píxeles en formato RGB. Esta información de imagen digital se almacena en una estructura constituida por tres matrices, cada una de ellas con esas dimensiones. Cada matriz se corresponde con la información de un canal de los tres colores primarios que constituyen el color (rojo verde y azul). Cada píxel se representa con 24 bits como estándar desde 2005, con 8 bits asignados a cada canal. En otras palabras, la imagen obtenida esta compuesta de tres imágenes donde cada una puede almacenar píxeles con valor comprendido entre 0 y 1. El cero significa que ese color no interviene en la mezcla y va creciendo proporcionalmente a su presencia en la mezcla. Los tres colores a cero componen el negro y por el contrario a uno, el blanco.

3.2. Pre-procesado

El siguiente paso, una vez adquirida la imagen es, aplicarle las transformaciones necesarias para mejorarla o hacerla mas entendible a los procesos posteriores de segmentación y localización. El punto más crítico de nuestro sistema y que se intentará resolver a continuación es la iluminación del área de trabajo. Las cámaras no siempre capturan los colores tal y como el ojo humano los percibe, debido, entre otros, a la luz reflejada o incidente en ellos. La luz reflejada por los cuerpos depende del tipo de fuente que la produce, su intensidad y del propio cuerpo que está siendo iluminado. El sistema a diseñar será puesto en marcha iluminado por fluorescentes o por luz natural, dependiendo del momento y debe ser capaz de funcionar en las dos situaciones. Como en fases posteriores se trabajará con colores, dependientes de la fuente de iluminación, en esta fase se elimina la dependencia con la fuente de iluminación del momento gracias a una representación cromática de los elementos de la imagen digitalizada.

El ojo humano bajo cualquier situación de iluminación, percibirá inequívocamente el color blanco del rojo y de cualquier otro. Por el contrario las cámaras digitales con frecuencia capturan imágenes con colores muy diferentes al que se está percibiendo a simple vista. Este hecho es observado con las fotografías recogidas en las pruebas iniciales con la cámara y representadas en las figuras 3.7, con tonos azulados, y la 3.6, que alcanza tonos anaranjados. Después de alguna prueba, se descubre que cuando la luz es natural y con alta intensidad, los colores de la imagen tienden a ser mas rojizos. Por el contrario, con luz artificial, los colores suelen tender a ser mas fríos, es decir, verdosos o azulados. Este problema aparece incluso en situaciones en las que la iluminación está cambiando constantemente como por ejemplo, un día nublado en el que las nubes ocultan y permiten el paso de la luz solar constantemente, donde se observará como dos imágenes consecutivas tienen colores totalmente diferentes.

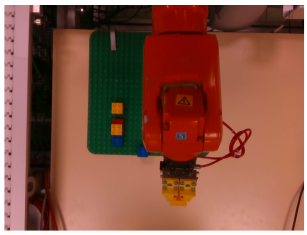


Figura 3.6: Imagen con tonos anaranjados propios de una iluminación intensa

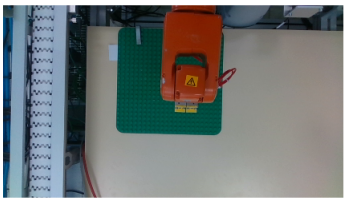


Figura 3.7: Imagen con tonos azulados propios de una iluminación débil

3.2.1. Temperatura de color

Antes de comentar la solución adoptada, es importante aclarar un concepto que explica por qué las imágenes digitales no siempre capturan el color tal y como un humano lo percibe. Se ha observado como dependiendo de la cantidad de luz, la temperatura del color tiende hacia tonos mas fríos o hacia tonos mas anaranjados. Este hecho se debe a la temperatura del color en ese momento. Si bien es así, se conoce como temperatura de color al dominio de alguno de los colores del espectro lumínico sobre los demás de modo que altera el color blanco hacia el rojo o hacia el azul en el espectro. Este dominio de rojo o azul hará que los tonos de nuestra fotografía se alteren y que ni el blanco no parezca blanco puro. La información de las imágenes y como consecuencia el sistema, estarán sujetos a la luz percibida por la cámara y la iluminación del momento. Nuestro cerebro a diferencia de la cámara, es capaz de realizar los cambios necesarios con el objetivo de que podamos entender el entorno que nos rodea. Al mirar un papel blanco, el cerebro humano asocia el papel con el color blanco y realiza lo que se conoce como equilibrio de blancos automático y así, percibirlo siempre blanco y no rojo o azulado como en el caso de las figuras 3.6 y 3.7 aunque la luz que esté reflejada en él tenga cierto color.

La luz blanca que nos ilumina es una mezcla de longitudes de onda con todos los colores del espectro. La proporción de cada color depende del tipo de fuente luminosa, se mide en Kelvin y es lo que se conoce como temperatura de color o temperatura a la que un supuesto cuerpo negro emitiría tal color. En la figura 3.8 se ve como el blanco o neutro se sitúa en los 5.500 K, que equivale a luz natural al medio día. La luz con temperatura menor se va haciendo gradualmente más amarillenta o anaranjada. Por el contrario objetos expuestos a luz con temperatura mayor se hacen mas azulados. La figura muestra como se vería un blanco puro dependiendo de la temperatura de color de la fuente de iluminación [26].

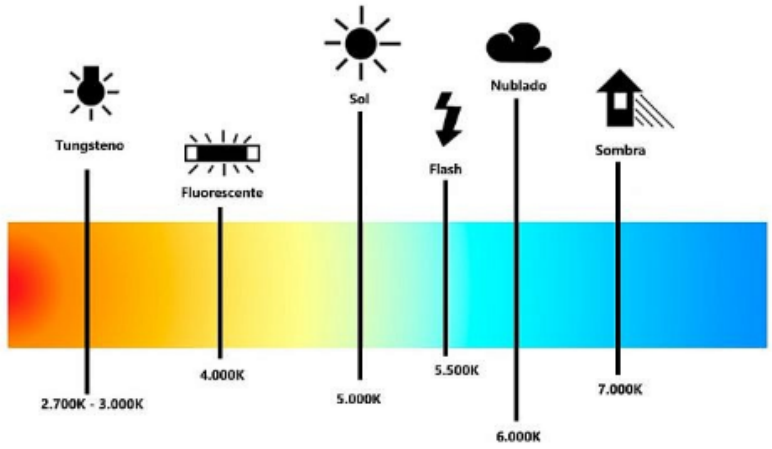


Figura 3.8: Cada fuente luminosa emite una luz con distinto color o temperatura de color. Esta temperatura de color se mide en Kelvin y varía desde tonos rojizos a azules) [26]

3.2.2. Balance de blancos

Para tratar de neutralizar las diferencias de color capturadas dependiendo de la temperatura del color procedente de la fuente de luz del momento, se utilizará el Balanceo de blancos. Se elimina el color de las influencias de iluminación. A partir de un punto blanco conocido, se ajusta el brillo de los colores básicos rojo, verde y azul. Como consecuencia la parte mas brillante de la imagen aparecerá como blanco y la más oscura como negro para obtener unos colores lo más parecidos a los propios de los elementos o que al menos las diferencias en los colores dependientes de las condiciones de iluminación de la sala sean despreciables para las máscaras de color utilizadas y explicadas en la sección 3.4.1.

Una vez se ha realizado una investigación sobre estudios en los que se ha utilizado este método para la neutralización de la iluminación como en [20], se llega a la conclusión de que suele funcionar razonablemente bien en condiciones de luz normales, y que por lo general resuelve bien situaciones en las que aparece un claro color dominante con una iluminación, hecho observado en la figura 3.6.

En este proyecto se realiza esta modificación cromática una vez capturada la imagen. Se podría haber realizado en la captura de imagen añadiéndole una serie de filtros a la cámara pero se ha elegido hacerlo post-producción por motivos de simplicidad. Se procede a explicar el método que utiliza el balance de blancos para ajustar la señal de crominancias; el usuario seleccionará manualmente al arranque del sistema un punto blanco. Para ello debe existir

una pegatina blanca o segmento de folio siempre presente e inamovible durante el funcionamiento del mismo. A partir de ese punto, cuyas coordenadas serán guardadas como variables globales, el sistema determinará que color se esta percibiendo cuando realmente debería estar tomándolo como blanco puro, determinando con esto la distorsión proveniente de la iluminación y la eliminará de todos los píxeles de la imagen.

3.2.3. Código de Matlab

Los pasos seguidos en el programa desarrollado en Matlab son los siguientes:

1. Aparece por pantalla una imagen del entorno de trabajo, igual durante toda la ejecución del programa pues se trabaja con cámara estática, sobre la que el usuario hará doble clic en un punto blanco. El usuario deberá seleccionar un único píxel y el programa guardará sus coordenadas como coordenadas del blanco. Se hará uso de la función de Matlab `getpts(ax)` que permite elegir un punto en los ejes x e y utilizando el ratón. Se debe validar que el usuario tan solo haya elegido un punto pues esta función permite seleccionar más de un punto. No es recomendable colocar el blanco en un lugar saturado como podría ser la luz del techo.

2. Este píxel en la imagen pasa a ser un punto de referencia. Con este se estima la iluminación de la escena en todo momento gracias a la función:

   ```
   gray-val=(A(y,x,1), A(y,x,2), A(y,x,2))
   ```

   Al estar realizando el balance de blancos en una imagen de entrada (A) representada en RGB, cada píxel se representa por superposición de tres canales. Se observa como la función recibe como entrada el valor del píxel en la posición seleccionada por el usuario (x,y) en cada uno de los canales (coordenada tercera).

3. Se utiliza el color seleccionado como referencia para la iluminación de la escena y se corrige el balance de blancos de la imagen.

   ```
   img_rgb_normalizada = chromadapt(A,gray_val);
   ```

   Este último paso se realiza con cada imagen capturada, justo en el momento previo al procesamiento.

3.2.4. Resultados del bloque de Pre-Procesado

Los resultados obtenidos en este paso, se muestran en la figura 3.9. En esta figura aparecen tres imágenes tomadas en condiciones de iluminación diferentes. La primera con la escena iluminada con luz artificial, la segunda con luz natural intensa y la última se trata de una situación casi perfecta con luz natural media o débil. Se observa como mejora ligeramente la situación, sobretodo en el caso de luz natural intensa. A pesar de no eliminar completamente los efectos dependientes de la temperatura de color. Tras realizar distintas comprobaciones, se llega a la conclusión de que elimina los tonos anaranjados relativamente bien, evitando problemas en la fase de segmentación del brazo del robot, pues como se verá, se apoya en el color anaranjado del robot.

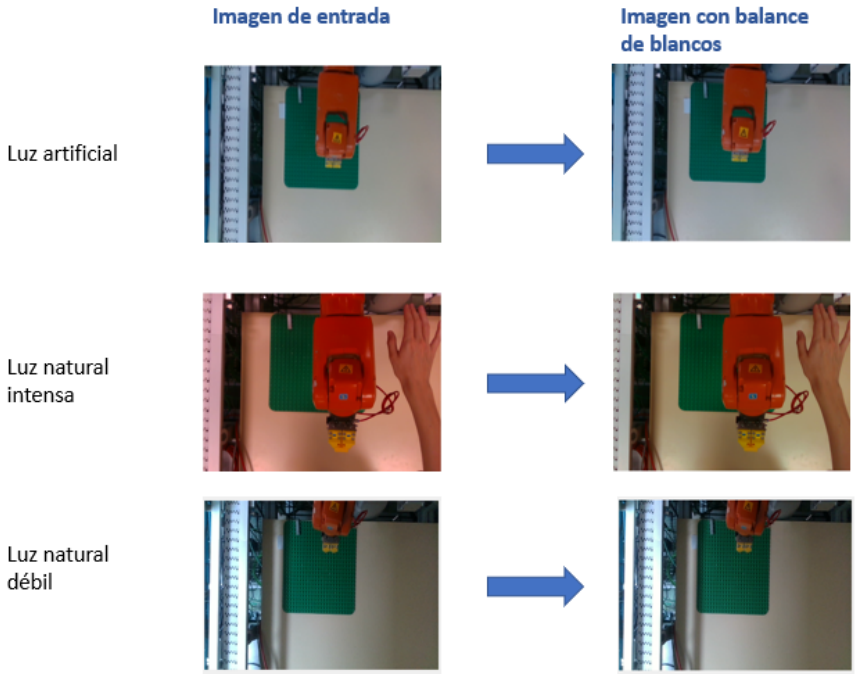


Figura 3.9: Resultados Parciales obtenidos aplicando el balance de blancos aplicado a tres imágenes tomadas en situaciones de luz distintas.

En la figura 3.10 se muestra otro ejemplo más claro sobre la mejora en los resultados al introducir este paso intermedio en el proyecto. En esta figura se observan tres máscaras binarias cuya procedencia y significado se verán en secciones posteriores. Estas tres máscaras binarias se han obtenido como resultado de diferentes etapas del procesamiento de un mismo fotograma. En este apartado tengamos en cuenta únicamente que la máscara 1 es obtenida por uno de los dos algoritmos de detección de primer plano, es decir

ha representado en blanco los píxeles con movimiento. Todos los píxeles que ha detectado pertenecen al robot, pues se ha elegido un fotograma sin obstáculos. La misión de aplicar la máscara de color será la de identificar estos píxeles como brazo pues de no ser así se interpretará que son obstáculos y que el robot corre peligro de colisión. Al aplicar la máscara de color a dos imágenes de un mismo fotograma pero con y sin realizar el balanceo previo de blancos se observa como en la que se ha aplicado este ajuste, máscara (2) se ha identificado todo el área ocupada por el brazo. Sin embargo en la que no se ha realizado este paso previo, máscara (3) se ha localizado el área ocupada por el brazo tan solo parcialmente. Este fallo se debe a que los colores de esa sección tenían distorsión y su valor se alejaba del naranja típico del brazo.

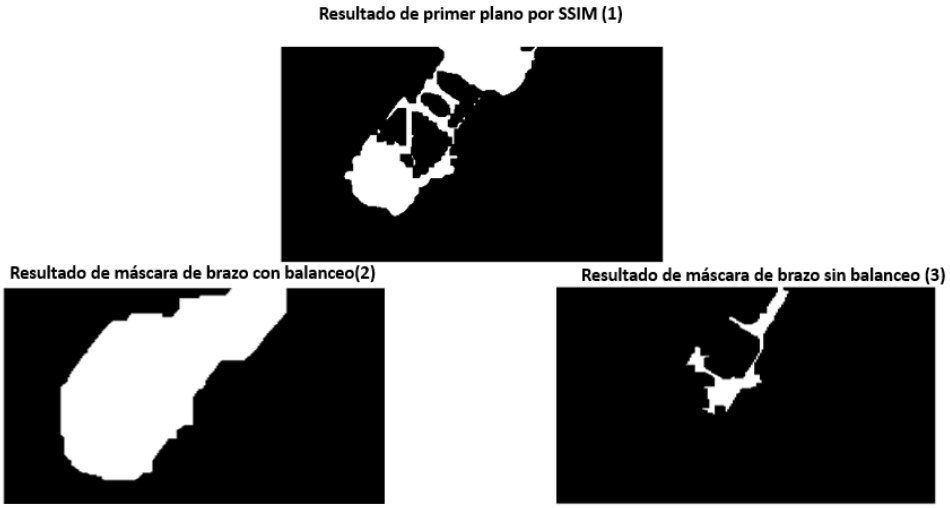


Figura 3.10: Comparativa del resultado obtenido en la segmentación del brazo de la etapa posterior habiendo o no aplicado balance de blancos a la etapa anterior. En (1) aparece una segmentación ideal del brazo, en (2) la segmentación como resultado de aplicar balance de blancos y operaciones morfológicas y (3) igual pero sin aplicar balance de blancos.

3.3. Segmentación

Después de la fase de preprocesado, en la que se intenta eliminar el ruido, el siguiente paso es la segmentación. Dentro del campo de la visión artificial se conoce como segmentación al proceso mediante el cual se divide una imagen digital en varias partes u objetos. Su propósito es simplificar la información contenida dentro de la imagen, para ello, se le asignan etiquetas a cada píxel en función de sus características. En el caso de esta aplicación se asignarán dos tipos de etiquetas; fondo y primer plano. El resultado de esta etapa será

una máscara binaria en la que los píxeles pertenecientes al fondo aparezcan con valor 0 y los del primer plano a 1. Pertenecerán al primer plano los píxeles que difieran del valor establecido para esa coordenada de píxel durante el periodo de aprendizaje del entorno de trabajo. Para definir el fondo de la imagen, existirá un periodo de entrenamiento o aprendizaje durante el cual se le asignará dicho valor a cada píxel. No obstante, el sistema deberá ser capaz de actualizar el modelo. Esto significa que el sistema actualizará los valores asociados como fondo para cada píxel, si así se requiere, por modificación del área de trabajo. Esto significa que si un objeto cumple un tiempo preestablecido dentro de la imagen, sin sufrir ningún movimiento, y no se encuentra en posición de peligro, deberá pasar a formar parte del fondo.

También es importante remarcar que en ningún momento se realizará un seguimiento del movimiento de los objetos, este planteamiento sería buena idea pues si el robot y el objeto se encuentran próximos pero realizan movimiento en sentidos opuestos, no habría riesgo de colisión. Sin embargo, en este proyecto por simplicidad, solo se estudia la información de los cambios en la escena.

Definir un modelo que cumpla con el objetivo planteado puede ser muy difícil cuando contiene formas sombras y objetos en movimiento. Teniendo en cuenta a su vez que los objetos estacionarios podrían sufrir variaciones de color e intensidad a lo largo del tiempo. Para conseguir adaptarse a estos dos problemas, se plantea combinar dos algoritmos; SSIM y el Modelo de Múltiples Gaussianas. Cada uno trabajará con objetivos diferente para conseguir una respuesta única por aplicación de operaciones lógicas entre los resultados parciales obtenidos con cada uno de ellos.

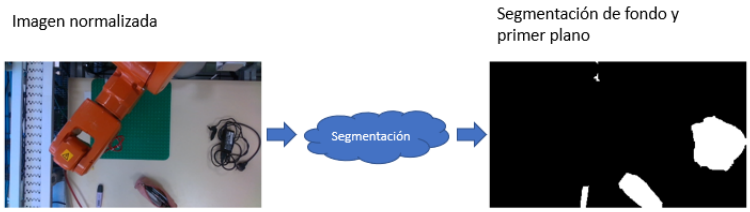


Figura 3.11: Entrada y Salida del bloque de segmentación

3.3.1. Modelo de múltiples Gaussianas

El primer algoritmo utilizado para la obtención del primer plano es el modelo de adaptación normal o Gaussiana. Es un modelo evolutivo, diseñado para maximizar el rendimiento de fabricación debido a la desviación estadística de los valores de los componentes. Como aparece representado en la figura 3.12 y proponía Wren et al [27], ajusta una o varias funciones de densidad de

probabilidad normales a cada píxel. Si en un momento determinado el color se encuentra fuera de esta distribución, será identificado como primer plano. Si durante un periodo de tiempo suficiente aparecen colores que difieren de la distribución en memoria, se adaptará la distribución a los valores que se han modificado para identificarlos así como fondo. Por tanto este modelo es flexible e irá cambiando durante la ejecución del programa.

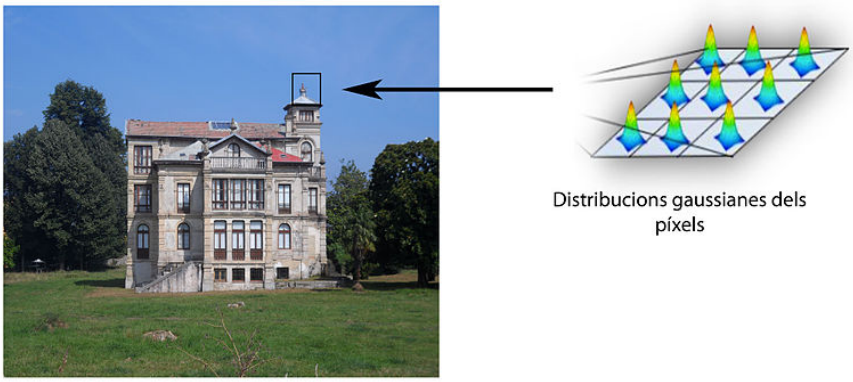


Figura 3.12: Esquema aclaratorio para el funcionamiento del Modelo de Múltiples Gaussinaas [22]

El fondo con el que trabajaremos será el ilustrado en la figura 3.13, aunque tal y como se ha planteado la aplicación, deberá funcionar correctamente con otro fondo. Se tienen en cuenta dos modificaciones de fondo; la primera periódica, ocasionada por ejemplo por el funcionamiento de la cinta transportadora blanca que aparece a la izquierda de la figura 3.13, y la segunda mucho más gradual, originada por cambios de iluminación sobre fondos estáticos. El algoritmo de distribuciones normales, no se ve afectado por ninguno de estos cambios, pues para movimiento periódicos asigna varias Gaussianas (típicamente entre 3 y 5) a un mismo píxel, recogiendo las diferentes fases del movimiento periódico como fondo. Para el segundo caso, el modelo es adaptativo y para cambios lentos y progresivos la distribución de probabilidad se adapta a ellos sin problema.

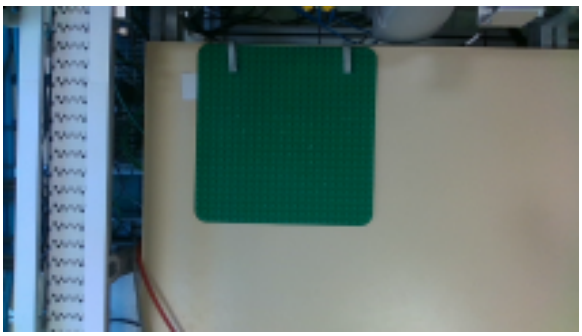


Figura 3.13: Ejemplo de información memorizada típicamente como fondo. Ejemplo de una imagen de las 100 tomadas durante la ejecución del entrenamiento para determinar los valores de fondo.

Los pasos que debe seguir nuestro sistema para implementar el algoritmo con Matlab son los siguientes:

1. Al arranque del sistema desde Matlab, se inicializa el objeto detector de primer plano. Asignándole los parámetros adaptados a las necesidades del proyecto. Las opciones de configuración del modelo son [22]:

   - **Tasa de aprendizaje:** Velocidad de aprendizaje para adaptar los parámetros del modelo. Controlando la velocidad con la que el modelo reconoce un elemento de primer plano estático como fondo. En el proyecto se ha establecido a 0.005, es decir un píxel reconocerá un valor como fondo, cambiando la distribución normal, tras aparecer este valor en 200 fotogramas consecutivos. Teniendo en cuenta que se requiere un tiempo medio aproximado para el procesamiento de cada fotograma de 0.15 segundos (consultar figura 3.29) se requiere un tiempo medio aproximado de 30 segundos en el aprendizaje de un objeto nuevo como parte del fondo.
   - **Número de Gaussianas:** Número de distribuciones asociadas a cada píxel. En el proyecto se han asignado tres. Cada una independiente a las anteriores. Si el valor de un píxel está dentro del área ocupada por alguna de las tres distribuciones será clasificado como fondo. Se han elegido tres, valor típico utilizado al usar este tipo de modelo, pues se comprobaba que con menor número de Gaussiana se recogían un gran número de falsas alarmas, fue el mínimo valor con el que el sistema funcionaba de manera coherente. Esto se debe a la variación constante de los tonos percibidos por la cámara por la temperatura de color, especialmente en situaciones por la mañana entre las 11:00 y 15:00 en las

que el sistema se encuentra iluminado de manera natural y con intensidad variando constantemente.

- **Número de muestras utilizadas para el entrenamiento:** Cuantas mas muestras se utilicen, más memoria tiene el sistema y para un mayor rango de colores los detecta como fondo. Por el contrario un numero bajo de memoria puede provocar la aparición de un numero elevado de falsas alarmas. Al usar un número fijo de Gaussianas superior a 1 estos dos problemas se solucionan. Aun así, el uso de un número elevado de muestras tiene efectos contraproducentes: la velocidad de procesado disminuye. En función de la aplicación sera más conveniente un caso u otro. En este proyecto se han asignado 100 muestras al entrenamiento, obteniendo un tiempo de entreno entorno a 5 segundos pues durante el periodo de entreno no se pasa por las fases de segmentación, localización y comunicación con el controlador del robot y como consecuencia el tiempo medio transcurrido entre fotogramas es de 0.05 segundos.

La instrucción para crear el objeto con las características especificadas en Matlab ha sido:

```
1    foreground_detector= vision.ForegroundDetector(
2                 'NumTrainingFrames',100,
3                 'InitialVariance',30^2,
4                 'LearningRate', 0.005,
5                 'NumGaussians', 3,  ...
                      'MinimumBackgroundRatio',0.4);
```

2. Una vez definido el modelo a utilizar, el siguiente paso es inicializarlo. Dependiendo del número de fotogramas asignados al entreno, se deberá asignar un periodo de tiempo para el entreno y aprendizaje del fondo. En este caso se han asignado los 100 primeros *frames* a entrenamiento, donde el brazo no se interpondrá en el aprendizaje.

3. Una vez finalizado el entrenamiento, el detector comienza a general resultados fiables y por tanto se puede poner en marcha el sistema de seguridad. Para el procesado de cada imágenes, se debe obtener el resultado parcial de este algoritmo llamando a la función a partir del objeto creado.

```
foreground_rgb = foreground_detector(frame_rgb);
```

El proceso de segmentación de este algoritmo no es perfecto y se recomienda usar apertura morfológica para eliminar el ruido y rellenar los huecos detectados. La operación de apertura se comentará con más detalle en el apartado

3.4.2 donde se comentan todas las operaciones morfológicas utilizadas en esta aplicación de procesamiento de imagen.

3.3.2. Resultados obtenidos tras aplicar el Algoritmo de Múltiples Gaussianas

Los resultados parciales obtenidos al aplicar este algoritmo aparecen representados en la tabla 3.2. Esta tabla contiene en la columna central la imagen de entrada y en tercera columna la respuesta del algoritmo. Se expone el comportamiento del algoritmo ante tres situaciones de iluminación; con sombras, con reflejos y prácticamente ideal. El fondo estudiado ha sido igual al de la figura 3.13. En dos de ellas se presenta el primer plano sin el brazo y en una aparece tambien el brazo. Sin embargo, la presencia o ausencia del brazo no es relevante en estos resultado. Lo que si es destacable es que este algoritmo capta con facilidad las sombras y reflejos como obstáculos de primer plano debido a que el valor de color de píxel cambia significativamente. El motivo de esta detección errónea es que se basa únicamente en el color del píxel. Se plantea como solución la combinación con el algoritmo SSIM explicado a continuación.

Finalmente, no se han representado resultados sobre la adaptación del algoritmo a cambios graduales en el fondo, pero siempre que los cambios sean lentos el algoritmo se adapta sin ningún problema. Por ello, se mantendrá este algoritmo y se combinará con el algoritmo SSSIM expuesto a continuación, que no se adapta a cambios de fondo.

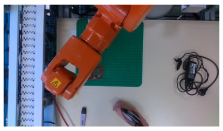
Tipo de ilumina-ción	Imagen de entrada	Máscara de salida
Intensidad luminosa fuerte con reflejos		
Intensidad luminosa ideal sin sombras ni reflejos		
Intensidad luminosa débil con sombras		

Tabla 3.2: Tabla de resultados obtenidos del Modelo de Múltiples Gaussianas, al introducir imágenes del entorno capturadas en distintas condiciones de iluminación.

3.3.3. SSIM

El segundo método utilizado es el del índice de similitud estructural, SSIM (*Structural Similarity Index Method* en inglés). Cuando se trabaja con imágenes digitales, aparece el problema mencionado en el apartado anterior debido a que estas habitualmente se ven altamente distorsionadas por elementos no estructurales como puede ser el cambio de luminancia o luminosidad, llegando a percibir como objeto una sombra o reflejo. Problema del que se ha hablado previamente y que se pretende resolver con el algoritmo expuesto en esta sección.

SSIM fue diseñado inicialmente para determinar un índice cuantitativo sobre la calidad de una imagen percibida por un humano. Fue desarrollado por Z.Wang y se basa en la idea de que el sistema visual humano es muy sensible a la información de elementos estructurales de una escena, como pueden ser las formas o figuras, y no lo es tanto para efectos no estructurales, como la iluminación. Por otro lado, los objetos de una escena presentan fuertes dependencias estructurales y es por ello que este algoritmo también ha sido usado en estudios sobre la identificación de objetos en imágenes. Las distorsiones no estructurales por cambios en la iluminación, como sombras y reflejos no

afectarán a las estructuras de los objetos que se pretenden localizar [29].

La idea de SSIM consiste en a partir de dos imágenes, una de ellas con calidad supuestamente perfecta, obtener la calidad de la otra comparando ambas. El índice SSIM local estima el error absoluto presente en cada píxel a partir de la diferencia con respecto a la imagen de referencia. Este error se fundamenta en tres elementos de imagen; la luminancia, el contraste y la estructura. Estos componentes son relativamente independientes, por lo que cambios en la luminancia o contraste no afectarán a la estructura de la imagen. Adicionalmente, se podrá poner mayor peso a cualquiera de los tres factores para calcular el índice de similitud entre pares de píxeles.

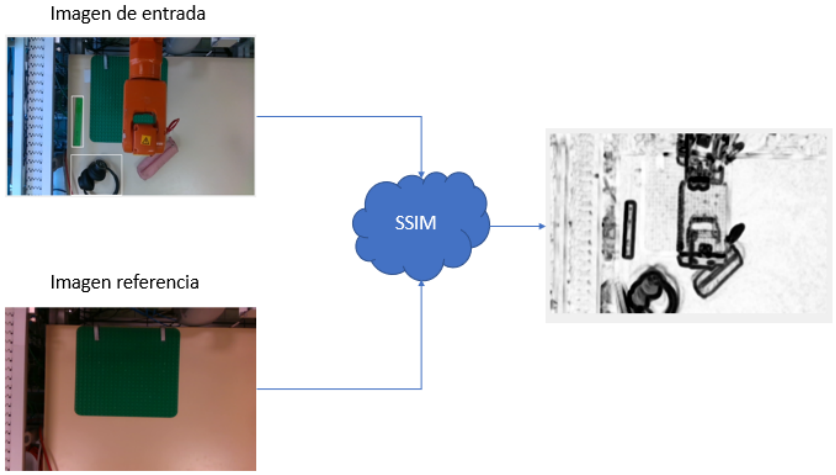


Figura 3.14: Representación de información de entrada y resultado a la salida del bloque del algoritmo de SSIM en la aplicación

A modo de resumen, la principal ventaja que se obtiene utilizando este algoritmo es que incorpora importantes fenómenos perceptivos que incluyen tanto el enmascaramiento de luminancia como términos de enmascaramiento de contraste. Consiguiendo que las regiones brillantes o sombreadas no sean captadas por este. Con el enmascaramiento por contraste lo que se consigue es que las distorsiones se vuelvan menos visibles cuando hay una actividad o textura significativa en la imagen. En la figura 3.15 aparece un diagrama básico de funcionamiento del algoritmo, con dos imágenes de entrada, un primer cálculo de índices de similitud locales obtenidos de comparar cada píxel de una imagen con la de referencia. Finalmente el último bloque calcula un índice global de similitud que no tendrá utilidad dentro de esta aplicación.

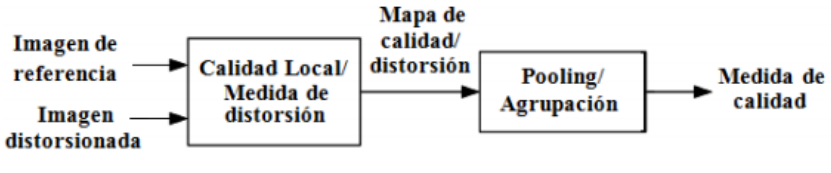


Figura 3.15: Diagrama de bloques explicativo del Algoritmo SSIM en Matlab [29]

Implementación del Algoritmo en Matlab

El código que se utiliza en el proyecto está ya programado en Matlab. Esta función calcula el índice SSIM comparando la imagen de referencia con la de entrada, píxel a píxel generando a la salida un mapa SSIM junto con el índice general de la imagen, fruto de la media del total de índices del mapa. En el caso de este algoritmo, a diferencia del de distribuciones normales, Matlab no tiene la herramienta para definir un objeto del sistema para utilizar este algoritmo con una configuración establecida en la creación del objeto y por tanto, se deberá llamar de la forma que se hace tradicionalmente a las funciones, introduciendo los argumentos de configuración cada vez que se invoca. A continuación, se exponen los argumentos que esta función puede tener a la entrada.

- Imagen de referencia: Esta imagen será obtenida al finalizar el periodo de entrenamiento, fruto del fondo memorizado para el modelo Gaussiano. Es asignada al inicio del programa y no se modifica. Al ser la referencia para el cálculo del primer plano durante toda la ejecución de la aplicación, este modelo no es adaptativo y captura los cambios del fondo como cambios de primer plano. Este problema no es relevante, pues sus resultados se combinan con los del algoritmo anterior que recordemos, si que se adaptaba a los cambios.
- Imagen de entrada: Representa la situación actual del área de trabajo del robot. A la que corresponden los índices de similitud. Varía en cada iteración.
- Exponentes para la componente de luminancia, contraste y estructural. Estas son opcionales y se utiliza con formato nombre-valor. En el proyecto no se han utilizado pues con la configuración estándar se obtenían resultados óptimos.

Un último detalle a comentar sobre el funcionamiento de este algoritmo en Matlab, es que trabaja con imágenes en escala de grises. Se podría utilizar

la imagen en RGB invocando tres veces a la función, una por cada canal, pero añadiría procesamiento innecesario ya que transformando la imagen a escala de grises y llamándola una única vez, se obtienen resultados muy positivos. Dentro del programa Matlab se llama a esta función con la siguiente instrucción:

```
[mssim,ssim_map]=ssim(img_gray,imagen_ref,'exponents', ...
                      [0.5,1,1]);
```

3.3.4. Binarización de Mapa SSIM

Como se observa en la figura 3.14 al inicio de la sección, el objetivo es obtener a la salida de este bloque una máscara binaria con píxeles que considere fondo a 0 y aquellos que haya clasificado como primer plano a 1 para poder ser combinada con la respuesta al Modelo de Múltiples Gaussianas. Sin embargo, el mapa devuelto por la función SSIM tiene asignado a cada píxel un valor comprendido entre 0 y 1,0 se corresponde con las zonas totalmente desiguales y 1 con las idénticas. Es muy difícil llegar a 1 pues cualquier diferencia por mínima que sea, la recoge.

Para pasar del mapa SSIM a la máscara se deberá realizar un paso intermedio de binarización representado en la figura 3.16. Este pequeño bloque consiste en un test para determinar en primer lugar un umbral de decisión, y a partir de éste, determinar que píxeles están por encima, clasificados como blanco (1), y cuales por debajo, representados en negro (0). Finalmente se invertirán todos los índices de la imagen pues el Algoritmo de Múltiples Gaussianas trabaja con los elementos del primer plano a 1.

Figura 3.16: Objetivo a alcanzar en la etapa de binarización del mapa SSIM. A la izquierda aparece un ejemplo de resultado obtenido a la salida del algoritmo SSIM y a la derecha el resultado de binarizar este resultado.

El umbral T puede ser local o global, esto dependerá de si es calculado para cada mapa o existe uno común utilizado durante todo el tiempo de funcionamiento de la aplicación. En el caso del presente proyecto, se utiliza uno

global, calculado durante el periodo de entrenamiento, pues el criterio para el cálculo del índice de similitud es igual para todas las imágenes de entrada y esto conlleva a considerar fondo a partir de un mismo valor. Se considera que un umbral demasiado pequeño hará que regiones del primer plano no sean detectados correctamente. Por el contrario, un umbral por encima del requerido introducirá ruido. De ahí surge la necesidad de buscar un algoritmo para obtener un valor adecuado para T. Se han estudiado algunos algoritmos como el de Otsu [19], analiza el histograma de niveles de gris de la imagen para determinar un umbral que segmente la imagen a partir de la mediana del mapa de índices. En el caso de la aplicación, este método no es coherente pues con el criterio de la mediana se detectarían píxeles pertenecientes al primer plano incluso para fotogramas en los que no se haya detectado ningún elemento como primer plano, pues los valores registrados rara vez alcanzarán el valor exacto de 1.

Finalmente, el método empleado para la binarización se ha diseñado específicamente para la aplicación. El umbral de binarización se obtiene calculando los índices medios de similitud entre cada par de fotogramas tomados consecutivamente durante el periodo de entrenamiento. Se selecciona el índice medio mayor entre todos los obtenidos durante el periodo de entreno, para evitar casos en los que ha habido movimiento dentro del periodo de entreno o se ha modificado el fondo. Se obtendrá un valor muy próximo a 1 pues se tratará de imágenes exactamente iguales a simple vista pero cuyos valores de píxeles no son exactamente iguales. Este valor próximo a 1 pero nunca 1, se tomará como 1 en el sistema y se dividirá entre dos para obtener el umbral de binarización. Los valores por debajo de este umbral serán considerados 0 en y los superiores como 1.

3.3.5. Resultados parciales del Algoritmo SSIM

Por último, con respecto a este algoritmo conocido como SSIM, los resultados parciales obtenidos aparecen representados en la tabla 3.3. Para comparar su funcionamiento con el del algoritmo anterior, se han estudiado los resultados obtenidos frente a las mismas situaciones expuestas en el caso anterior. Se aprecia con claridad su mejora en la detección de obstáculos con respecto al modelo de la normal. En este caso no se capturan las sombras y reflejos que aparecían en la tabla 3.2.

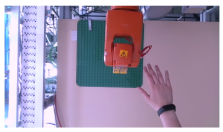
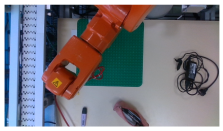
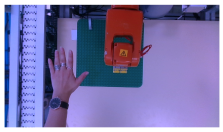
Tipo de iluminación	Imagen de entrada	Máscara de salida
Intensidad de iluminación fuerte con reflejos		
Intensidad de iluminación media sin reflejos ni sombras		
Intensidad de iluminación baja con sombras		

Tabla 3.3: Resultados SSIM con distintas situaciones de iluminación. Mucho mejores que los obtenidos con el algoritmo de múltiples gaussianas.

3.4. Reconocimiento

Una vez localizados los elementos críticos en la imagen, se comienza la siguiente fase del algoritmo: el reconocimiento. En esta fase en primer lugar se clasificarán las secciones de cada imagen pertenecientes al brazo del robot o al resto de objetos de primer plano. Se clasificarán como obstáculos, todo conjunto dentro del primer plano que no pertenezcan al robot. Por último, se calculará el área ocupada por cada uno para el posterior cálculo de distancias obstáculo-brazo. Las distancias, por simplicidad, se calculan teniendo en cuenta tan solo dos dimensiones, se ignora la dimensión con información sobre la profundidad de los elementos.

3.4.1. Reconocimiento de brazo robótico

El robot con el que se trabaja es naranja con la pinza amarilla. Para localizarlo, se buscarán conjuntos de píxeles con colores similares a los del brazo y cuya distribución tenga sentido con la del brazo y el sistema, pues por ejemplo la aparición de dos brazos no tendría sentido.

El método empleado buscará máscaras binarias de los dos colores predo-

minantes en el brazo, se realiza la intersección con operaciones morfológicas para la unión y formación de una única máscara binaria que identifique completamente al brazo. Por último se modela la forma del brazo a un rectángulo, calculando los vértices de este.

Máscara de color

Para hallar la posición del brazo, el primer paso es obtener una máscara binaria con los píxeles pertenecientes a la región ocupada por el brazo a uno y el resto a cero. Como ya se ha explicado, se buscarán estos píxeles según el color, amarillo si pertenece a la pinza y naranja si es parte del cuerpo. Para extraer los píxeles con valor de color dentro de un rango se utiliza la herramienta *ColorThreshorlder* de Matlab. Siguiendo los pasos explicados a continuación, se generan automáticamente dos funciones, una para cada color, con las que, a partir de una imagen de entrada con el balance de blancos aplicado, se obtienen como resultado las máscaras binarias.

Esta herramienta permite escoger una foto y un espacio de color para ajustar el comportamiento de las funciones. Los espacios de color posibles son: RGB, HSV, YCbCr y L*a*b. En nuestro caso se ha elegido YCbCr, debido a que cada píxel viene representado por tres componentes, una de luminancia (Y) y dos de crominancia diferencia de azul(Cb) y diferencia de rojo (Cr) y se pretende asignar mayor margen de valores a la componente de luminancia que a las de crominancias para así evitar de nuevo problemas derivados de la variación de iluminación. En la figura 3.18 se muestra el funcionamiento de esta herramienta, donde se ve como se asigna un margen de valores a cada componente que constituyen el color a partir de los histogramas. El píxel con las tres componentes dentro del rango asignado para cada una de ellas, se representará en blanco (1) y el resto con negro (0). Se han definido los valores de umbrales para cada histograma de la imagen como se muestra en la figura 3.18, segmentando los colores deseados. En nuestro caso no se ha utilizado el espacio RGB, pues a pesar de ser muy intuitivo, mezcla las componentes de luminancia en sus tres canales y en este proyecto se pretendía asignar un margen muy amplio para la componente de luminancia, y mucho mas estrecho para las crominancias, consiguiendo así un sistema aun más flexible a los cambios en la iluminación.

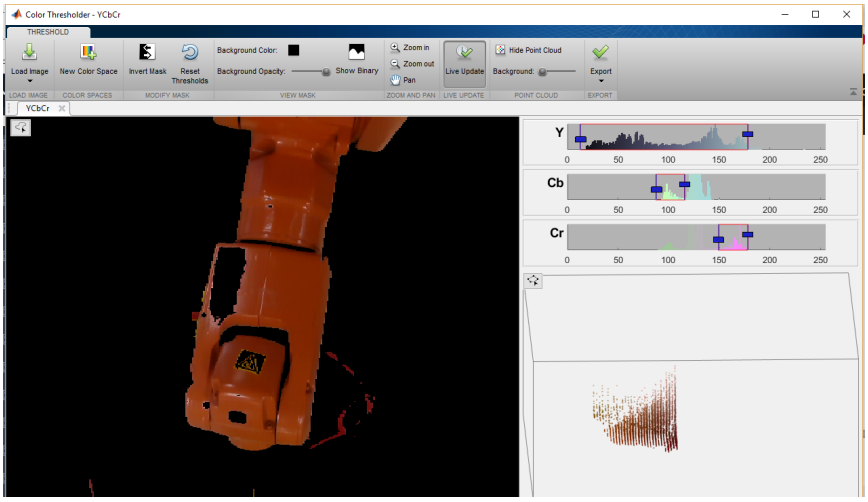


Figura 3.17: Herramienta de Matlab para ajustar valores de color a las máscaras según histogramas.

Otro factor a considerar es que cuando se varía la resolución de las imágenes tomadas, estas máscaras no funcionan correctamente, hecho representado en la figura 3.18, donde se ha introducido una imagen con resolución 480x640 y al aplicarle la máscara de color establecida para el sistema se ha obtenido ese resultado incoherente donde se ha detectado la mano como robot, hecho que no se observaba al trabajar con la resolución establecida para el sistema. Por tanto, si en algún momento se pretende modificar la resolución escogida, se deberán rehacer las máscaras con la herramienta de Matlab.

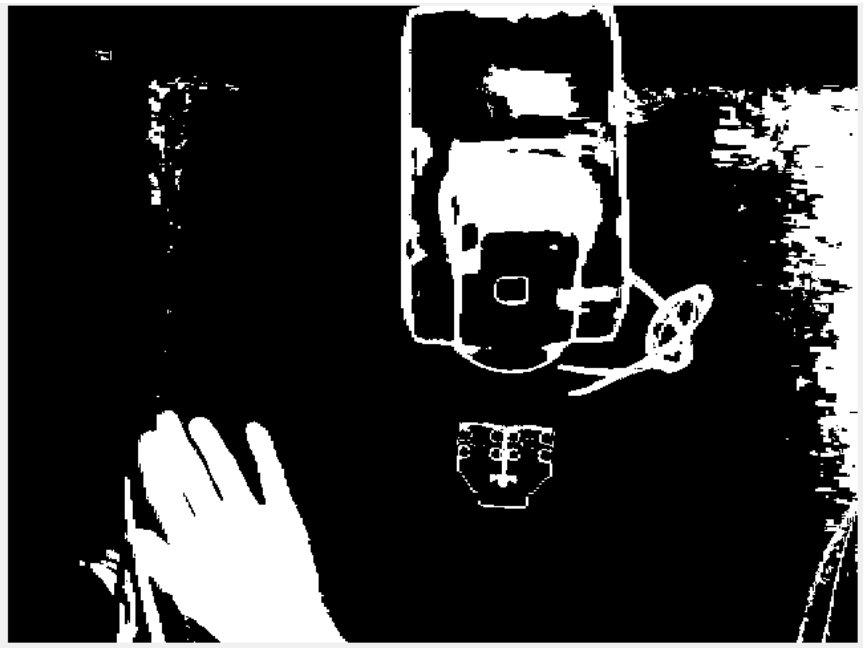


Figura 3.18: Mal funcionamiento de máscara de color al variar resolución de imágenes.

Para obtener una idea del funcionamiento, tras elegir el margen deseado de valores, se pulsa la opción de generar funciones y se obtienen dos funciones. Cada una de ellas, al ser invocada con argumento de entrada la imagen ilustrada en la figura 3.19 obtienen a la salida una matriz binaria con píxeles a 1 los que entran dentro del margen y a 0 el resto. Si representamos estas matrices aparece lo representado en las figuras 3.20 y 3.21 para las máscaras de naranja y amarillo respectivamente. Se puede ver como han identificado bastante bien el área ocupada por la pinza y por el brazo del robot.

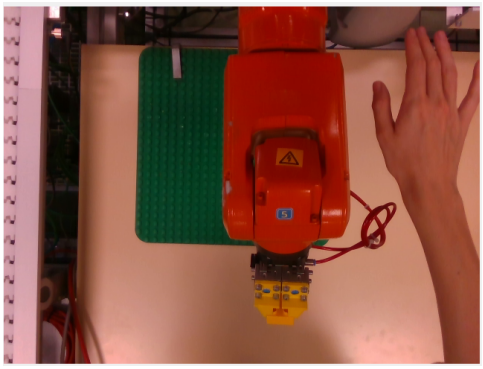


Figura 3.19: Imagen ejemplo que se introduce al bloque de segmentación del brazo del robot.

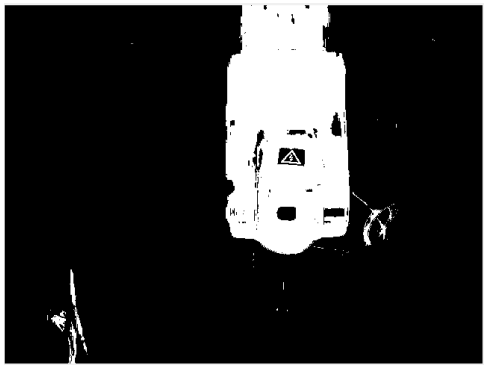


Figura 3.20: Resultado de aplicar a la figura 3.19 la máscara de color naranja diseñada para la pinza del robot.

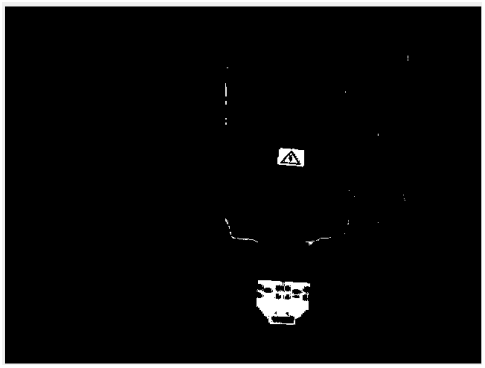


Figura 3.21: Resultado de aplicar a la figura 3.19 la máscara de color amarillo diseñada para la pinza del robot.

Finalmente, si la pinza tuviera otro color habría que modificar la máscara de la pinza, pues de no ser así, su movimiento sería clasificado como obstáculo a punto de colisionar con el robot. El resto de elementos amarillos o naranjas detectados en la región capturada serán desclasificados como elementos pertenecientes al robot en la última operación morfológica explicada a continuación en la que se selecciona tan solo el elemento de mayor tamaño de la imagen.

3.4.2. Operaciones Morfológicas

Las operaciones morfológicas son aquellas que alteran la forma o estructura de un objeto dentro de una imagen. Se utilizan sobre las máscaras binarias obtenidas previamente [23]. En este proyecto se utilizan cuatro tipos de operaciones morfológicas.

- **Dilatación:** Consiste en agregar píxeles a un objeto, es decir expandirlo. Es útil para asegurar que no hay peligro frente a cualquier ruido que se introduzca.
- **Erosión:** Operación dual de la dilatación, donde se extraen los píxeles que bordean al objeto, es decir lo reducen.
- **Apertura:** Erosión seguida de una dilatación. Separa objetos que están débilmente conectados para suavizar el contorno de las figuras.
- **Clausura:** Dilatación seguida de erosión, realiza operación inversa a la anterior, rellenando huecos en objetos dentro de la imagen.

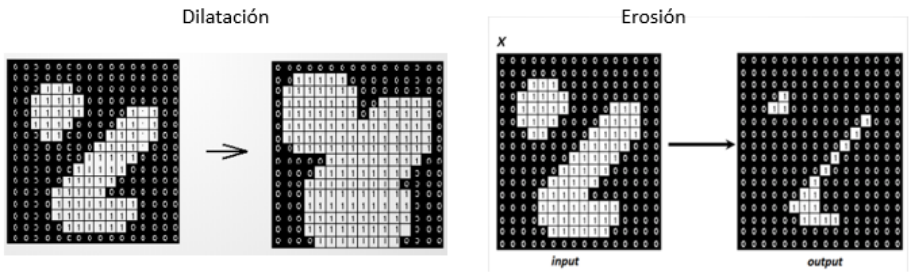


Figura 3.22: Ejemplo de funcionamiento de operaciones morfológicas de dilatación y erosión.

Las razones por la que se han utilizado estas máscaras en la localización del brazo son básicamente tres; eliminar ruido, dilatar el área ocupada por el brazo para crear un margen de seguridad y unir la pieza con el brazo pues al tener una pieza negra entre ellos, no vale con sumar ambas máscaras para obtener una única con la silueta del brazo como único elemento.

Para eliminar el ruido se aplicarán dos, una de erosión para limpiar píxeles sueltos y otra como último paso para quedarse únicamente con el elemento más grande de la máscara, que se supondrá que será el brazo. Para unir la pieza se usará una dilatación con una forma lineal de 30 unidades en línea. Finalmente para crear un margen de seguridad se usará también dilatación pero con una forma cuadrada como elemento dilatador.

Un ejemplo del proceso de segmentación del brazo del robot aparece ilustrado en la figura 3.23. Comenzando por arriba a la izquierda, la primera imagen muestra la imagen de entrada a este bloque. Esta imagen de entrada, tan solo ha sufrido modificación por el balanceo de blancos. La siguiente imagen se corresponde con la salida de la función de la máscara naranja, a esta matriz binaria se le aplica erosión para eliminar ruido que se aprecia ligeramente

en las esquinas, también se le aplica dilatación, después de haber limpiado el ruido, para ampliar el margen de seguridad de este elemento a proteger. Su resultado aparece en la figura 3. En la fila de abajo aparece el resultado de aplicar la máscara amarilla, la suma de ambos resultados y por último la dilatación para unir los dos elementos que forman el brazo robótico.

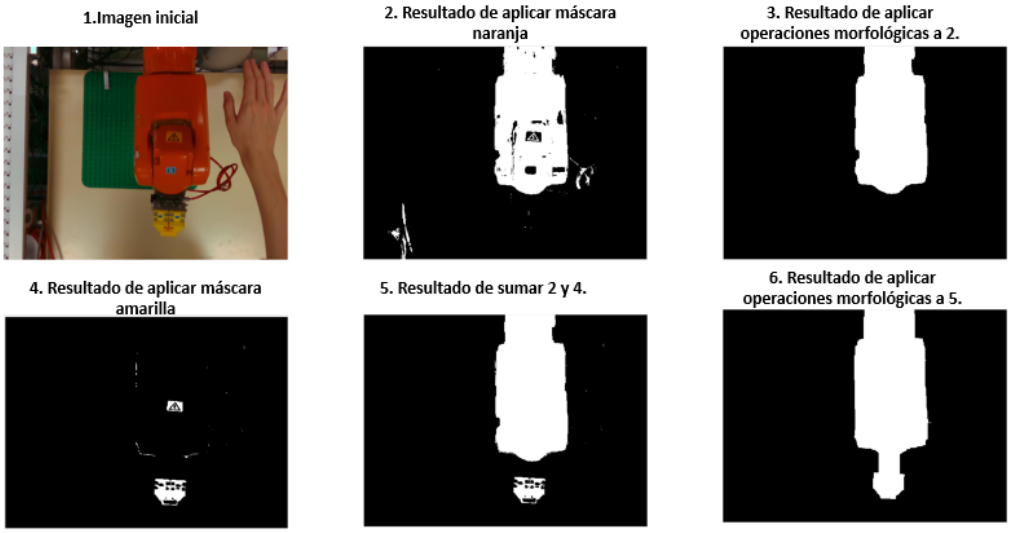


Figura 3.23: Etapas del proceso de segmentación de brazo del robot.

Finalmente, una vez obtenidas las tres máscaras de segmentación; resultado del brazo del robot y los dos algoritmos de primer plano el siguiente paso es combinarlas de tal forma que se consiga una única máscara con la segmentación del primer plano sin sombras localizadas ni movimiento del robot. Para ello, al trabajar con máscaras binarias se realizan dos operaciones lógicas básicas de *AND* y *OR*. La máscara binaria final estará compuesta por los bits a 1 que tengan valor 1 en la máscara resultante de los algoritmos de primer plano y valor cero en la de segmentación del brazo.

3.4.3. Localización de coordenadas de objetos

El último paso de la etapa de procesamiento se trata de localizar la ubicación en coordenadas x e y, de los objetos que se han detectado en el primer plano y el brazo para posteriormente calcular la distancia entre ellos. El objetivo final de este apartado es conocer la mínima distancia existente entre el robot y cualquier obstáculo para a partir de este valor, determinar el índice de riesgo. Para cumplir con este objetivo con la mínima cantidad de operaciones computacionales posible, se modelarán las regiones ocupadas por cada obstáculo como si fueran siempre polígonos, obteniendo los segmentos

que lo compone y finalmente calculando las distancias entre segmentos.

Análisis de *blobs*

Un método muy utilizado en el reconocimiento de elementos en imágenes, es el análisis de *blob* o manchas. Dentro de la visión artificial, este método está dirigido a detectar regiones en una imagen digital que difieren en relación con las regiones circundantes o píxeles que le rodean. En el caso de este sistema, al introducir a la entrada máscaras binarias con un fondo negro que ocupa siempre más del 60 % de la imagen y los obstáculos representados en blanco (unos). Estas regiones blancas serán identificados como manchas o *blobs*. Este métodos funciona muy bien en situaciones en las que los objetos a localizar se distinguen claramente del fondo, tal y como lo hacen los 1 de los 0 en las máscaras binarias con las que se trabajará. Las ventajas de estos métodos y por lo que se usaran son, su gran flexibilidad y su excelente rendimiento.

En el presente proyecto se utilizará para su desarrollo, una herramienta de Matlab, herramienta que devuelve características del blob como el centroide, el cuadro delimitador, la matriz de etiquetas y el recuento de blobs de la imagen de entrada. Para utilizar este modelo de identificación facilitado por Matlab, se deben seguir dos pasos:

1. Crear el `vision.BlobAnalysis` al inicio del sistema, cuando se define el resto de objetos del sistema, como por ejemplo, el objeto para el modelo de detección de fondo con distribución Gaussiana. Como con todo objeto del sistema en Matlab, se creará el objeto y se ejecutarán los datos a través de él como si se tratase de una función con único argumento de entrada, la máscara binaria sobre la que se desean identificar los elementos. La inmensa mayoría de herramientas en Matlab se usa en forma de objeto de sistema para permitir crear múltiples, persistentes, objetos o funciones reutilizables, cada uno con diferentes configuraciones, las cuales no deberán ser definidas cada vez que se requieran. Consiguiendo evitar la validación y verificación de entrada repetida, con un uso mas fácil y mejorando el rendimiento de la aplicación. En el momento de definir el objeto Matlab permite añadir argumentos de entrada para modificar la configuración del mismo, pero no posteriormente. Algunos de los opciones posibles para personalizar el modo de funcionamiento del objeto, son:

    - Número máximo de blob o elementos que se podrán identificar dentro de la imagen.
    - Área mínima en píxeles que debe poseer un elemento para ser identificado como mancha o *blob*.

- Excluir manchas que se encuentren tocando el borde de la imagen
- Orientación del rectángulo. Este modelo trata las manchas como si fueran rectángulos, por lo que se puede especificar previamente la orientación del rectángulo que lo delimitará.

Además, el modelo permite configurar, cuando se crea el objeto, qué desea obtener como salida cuando éste sea invocado como una función. La información que puede obtener es:

- Centroide: Matriz de dimensiones Mx2, donde M representa el número de elementos localizados y el vector que le acompaña, las coordenadas x e y de cada centro de los elementos localizados.
- *BoundingBox*: Matriz de dimensiones Mx4, donde M representa el número de manchas, las otras cuatro representan la posición x e y del vértice superior izquierdo y los dos valores de altura y ancho de la mancha.
- Cuenta: Devuelve número de manchas localizadas.
- Perímetro: Devuelve un vector con una aproximación de la longitud en píxeles de cada blob o elemento.

En el caso de esta aplicación se han habilitado las salidas de centroide y boundingBox, el primero se utilizará para identificar el número de elementos encontrados, si la dimensión de este vector es cero significará que no se han identificado elementos. Adicionalmente, se ha configurado el área mínima para que se identifique como elemento a 200 píxeles. Esta modificación evita un gran número de falsas alarmas, pues sin esta restricción, se observaban con frecuencia fallos mínimos en la segmentación del primer plano y se localizaban falsas alarmas por ruido. Esta modificación constituye un filtro más del ruido y supone una reducción significativa del número de falsas alarmas.

2. Por último se llama al objeto con argumento de entrada una máscara binaria de las vistas en el apartado anterior, de la misma manera que se llama a una función, en cada iteración del procesamiento de imágenes, obteniendo a la salida información del centro y vértices de los obstáculos identificados.

3.4.4. Resultados parciales de la localización de obstáculos con Análisis de Blob

Con este método, se ha observado que si la máscara de entrada está correctamente segmentada, los resultados serán siempre los mismos y sin posibilidad de fallo. Modelará cada obstáculo con un rectángulo a partir del vértice superior izquierdo y altura y ancho máximo del elemento identificado. Al

introducir las máscaras correspondientes a la segmentación de obstáculos, los resultados obtenidos se han representado con otro elemento de Matlab que pinta bordes a partir de los *boundingbox* sobre la imagen original y están representados en la figura 3.24. El modelado de las figuras en rectángulos, con sus lados orientados paralelamente a los lados de la imagen se considera adecuado para la detección de objetos no demasiado grandes, porque no ocupa secciones demasiado grandes. El peor de los casos se registra en la imagen 1 de esta figura, donde se ve como el rectángulo está ocupando sección de la imagen sin obstáculo, pero el espacio ocupado no es lo suficientemente grande como para generar problema o malentendidos.

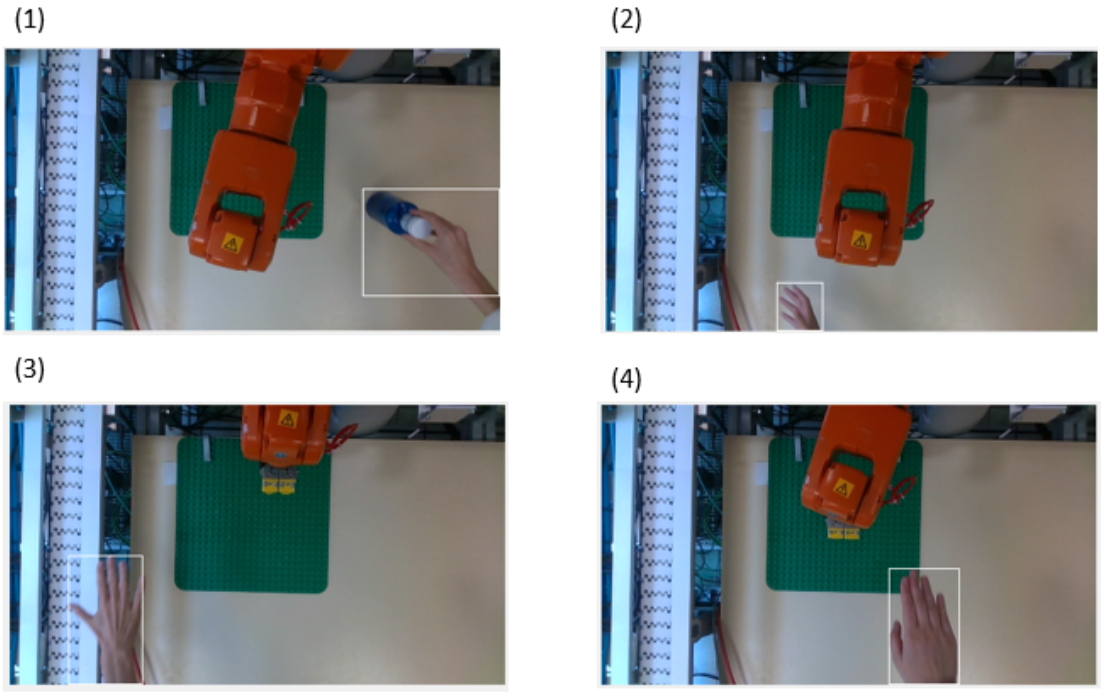


Figura 3.24: Resultados de la localización de coordenadas de objetos dentro de una imagen con análisis de *Blob*. Para su representación se han dibujado bordes en las coordenadas encontradas.

Por el contrario, los resultados obtenidos al aplicar, este método, no han sido del todo positivos cuando se introducía la máscara del brazo. Se observaba como la región o *boundingbox* identificada para este brazo ocupaba, en un alto porcentaje de ocasiones, el 90 % de la imagen, generando un gran número de falsas alarmas por peligro de colisión con un obstáculo que realmente se encontraba lo suficientemente lejos del brazo como para no ser necesario detener completamente el robot. Se recuerda que esta aplicación, con el objetivo de aumentar al máximo el rendimiento de robot, ante la aparición de

obstáculos, no siempre detendrá la ejecución principal del robot, intentará en este caso jugar disminuyendo la velocidad de trabajo en función de las distancias a los obstáculos. Este hecho, se aprecia con claridad en la figura 3.25, el brazo en esta ocasión está apuntando a la esquina inferior izquierda, se podrá encontrar trabajando orientado a cualquiera de las otras cuatro esquinas. Típicamente trabajará dentro del margen entre la esquina inferior izquierda y la esquina inferior derecha de la imagen, con aparición mayor o menor dentro del área capturada. En el caso de aparecer menos cuerpo del robot, la aproximación por análisis de blob será adecuada, el problema surge en el caso representado en la figura 3.25, cuando este cuerpo ocupa un área significativa de la imagen y se encuentra orientado diagonalmente a la imagen. En este caso al calcular el análisis de blob, el área ocupada por el robot será el rectángulo regular generado con el brazo como diagonal al mismo. Los bordes rojos de la figura delimitan el área que se está interpretando como robot con este análisis. Se observa como existe una región, marcada en verde que se podría eliminar de esta región. Es por ello que ha sido necesario buscar otro algoritmo para la localización del brazo del robot, que sea capaz de modelar su forma en un polígono, pero que sus lados no estén necesariamente orientados paralelamente a los lados de la imagen.

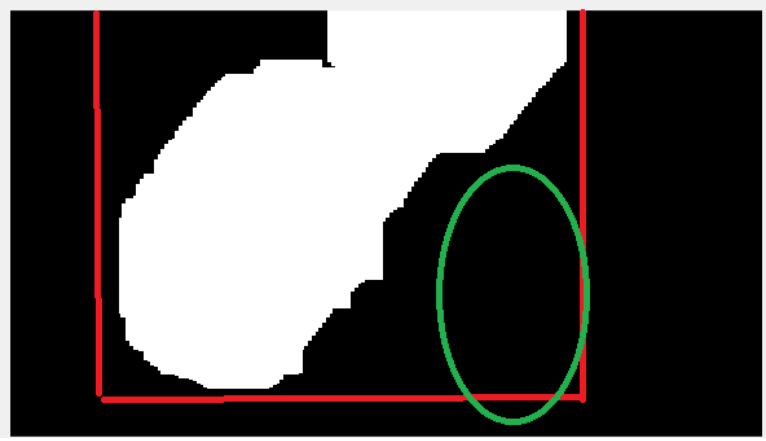


Figura 3.25: Resultado de localización de brazo del robot con método de análisis de *blob*. Sus coordenadas aparecen representadas en rojo y son inadecuadas pues esta recogiendo sección representada en verde sin ocupar.

Algoritmo alternativo para localización del brazo del robot

La alternativa al análisis de blob encontrada para la localización del brazo se ha diseñado específicamente para este proyecto. La solución encontrada obtiene como resultado algo similar a lo ilustrado en la figura 3.26. Se aprecia que existe un pequeño porcentaje de robot que se sale del área delimitada,

esto no será significativo pues se recuerda que se ha dilatado esta máscara para ampliar significativamente el área ocupada por el brazo en etapas anteriores. El algoritmo diseñado trata de buscar un polígono irregular que delimite lo más exacto posible el área ocupada por el robot. Para ello, en un principio se pensó utilizar tan solo cuatro vértices con las siguientes características: píxel blanco con menor valor de la coordenada x (el situado más a la izquierda de la matriz), píxel blanco con mayor valor de la coordenada x (situado más a la derecha de la matriz), píxel blanco con menor valor en la coordenada y (situado lo más arriba posible) y por último píxel blanco con valor mas grande en la coordenada y (el situado lo más próximo al lado inferior de la imagen).

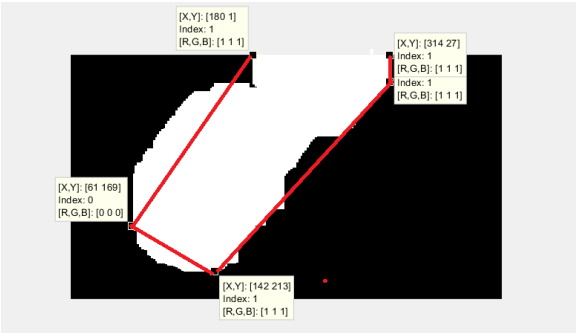


Figura 3.26: Objetivo de localización de brazo del robot, representado en rojo a alcanzar.

Este algoritmo obtendría un resultado aceptable para el caso de la figura 3.26, por el contrario al probar con otras posiciones del robot como las observadas en la figura 3.27 este sistema obtiene resultados mucho peores a los observados.

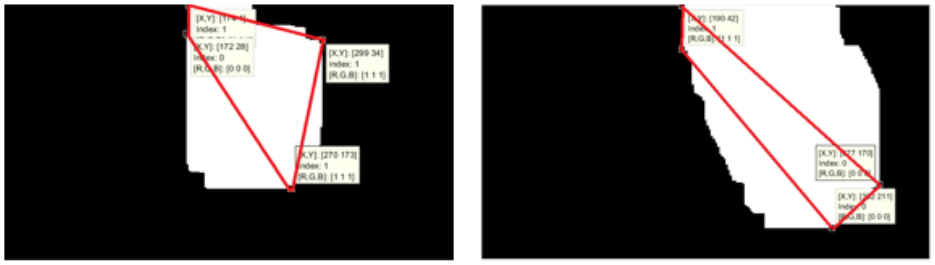


Figura 3.27: Aparece representado en blanco la segmentación del robot, en rojo la localización de sus coordenadas. Resultados obtenidos con el primer algoritmo diseñado específicamente para la aplicación. Se observa claramente su mal funcionamiento.

Se observa que añadiendo un quinto vértice en ambas se conseguiría una aproximación mucho más razonable. Finalmente se ha diseñado un algoritmo que modele su forma siempre a partir de 5 vértices, formados por píxeles con las siguientes características:

- Píxel con valor uno, con coordenada Y a 1 y con coordenada X con mínimo valor posible. Vértice superior izquierdo, con nombre X1.
- Píxel con valor a uno, con coordenada Y entre 1 y 5 y con coordenada X con máximo valor posible. Vértice superior derecho con nombre X1.
- Píxel con valor uno y con mayor valor posible en la coordenada Y. Vértice inferior, con nombre Y.
- Píxel con valor uno y con mayor coordenada posible del eje X. Vértice W2.
- Píxel con valor a uno y con menor coordenada posible del eje X. Vértice W1.

Estos cinco vértices se unirán siempre con el mismo orden; X2-W2, W2-Y, Y-W1, W1-X1, X1-X2. Formando un pentágono o un cuadrilátero en aquellas situaciones en las que dos de sus vértices coincidan. Siempre que esto ocurra serán dos vértices consecutivos, teniendo en cuenta las posiciones que puede tomar el robot. La longitud de los segmentos será irregular e independiente en cada fotograma.

3.4.5. Resultados de localización del brazo del robot con vértices

En la figura 3.28 se muestran los resultados obtenidos con el segundo algoritmo diseñado específicamente para esta aplicación. En esta figura se observa como el área clasificada como coordenadas del brazo es mucho mas reducida y se adapta mucho mejor a la región ocupada por el brazo que los resultados que se obtenían con el Análisis de Blob.

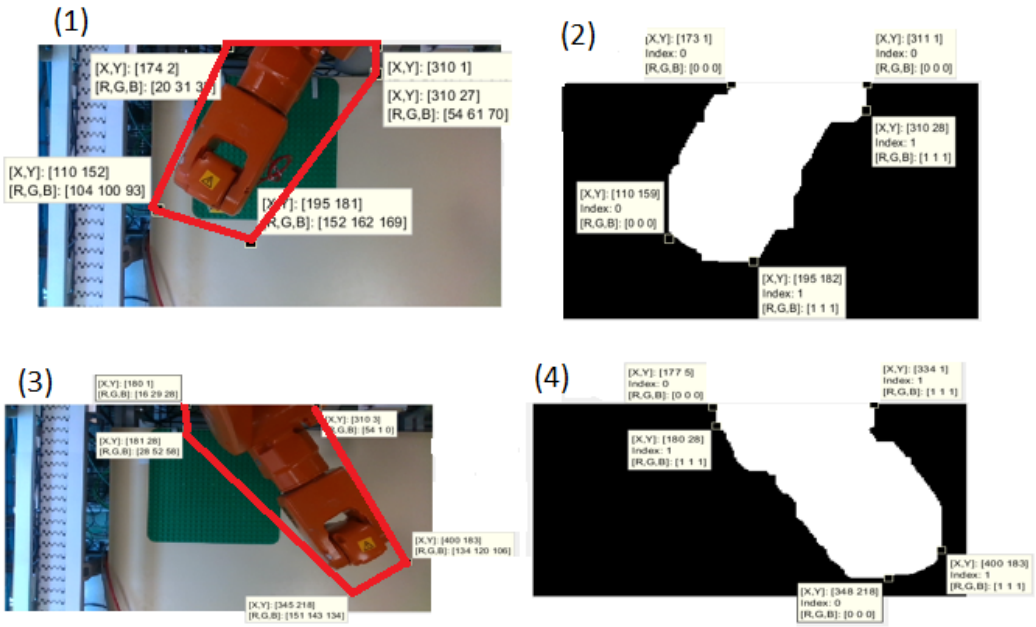


Figura 3.28: Resultados tras aplicar el segundo algoritmo que utiliza cinco vértices. Se observa que los resultados cumplen con los objetivos de este bloque de segmentación para el brazo orientado en las dos direcciones posibles.

Por último es relevante comentar un método alternativo para calcular la distancia entre obstáculos que en este proyecto no se ha llegado a desarrollar pero podría ser muy interesante, basado en el estudio de la intersección del brazo y los obstáculos y la dilatación del área ocupada por el robot. Este método consistiría en realizar la operación lógica AND entre la máscara del brazo dilatada y la de los obstáculos del primer plano. La máscara del brazo se dilatará en 4 niveles (correspondientes a cada nivel de seguridad). El nivel de riesgo será el mínimo nivel de dilatación en el que se produzca la intersección entre algún obstáculo y el brazo al realizar la operación AND.

3.4.6. Aproximación entre objetos.

Una vez calculadas las coordenadas de cada elemento, dentro de la imagen, el siguiente paso es calcular la mínima distancia para obtener un índice de peligro. Para calcular la distancia entre cada obstáculo y el brazo se utilizará la distancia mínima entre segmentos. Se recuerda que todos ellos se han modelado como polígonos con cada lado formado por un segmento de línea. El motivo por el que se ha decidido modelar los objetos como polígonos es para obtener menor número de operaciones computacionales. Se podría haber calculado la distancia mínima a partir de la distancia entre todos los píxeles

que conforman el borde de cada elemento, a partir de la distancia entre dos puntos una vez conocidas las coordenadas de ellos, pero la cantidad de operaciones a realizar crecería significativamente y con ello, el tiempo requerido para determinar esta distancia. Los pasos seguidos en la función que determina el índice de riesgo se siguen los siguientes pasos:

1. Comprobar si se han detectado obstáculos, si no es así no existe ningún riesgo ni distancia que calcular.

2. Comprobar si algún obstáculo está colisionando con el brazo. Para ello se compara, el número de elementos encontrado en el análisis de blob para la máscara que representa los obstáculos, con el número de elementos calculado también con el modelo de análisis de blob para la máscara resultante de sumar la del brazo con la de los obstáculos. Si el número es igual en ambas máscaras significará que algún obstáculo está colisionando con el brazo y por tanto se identifican como uno solo. En el caso de obtener colisión como resultado, el índice de peligro es máximo y se procede a comunicar al robot directamente sin hacer más cálculos. Por el contrario, si se obtiene que no están colisionando, se pasa a calcular la mínima distancia existente entre algún obstáculo y el brazo del robot.

3. Calcular la mínima distancia entre cada obstáculo y el brazo a partir de la distancia entre segmentos. Finalmente, para determinar el índice de riesgo se tiene en cuenta el obstáculo con menor distancia al brazo. La función encargada de determinar la distancia entre segmentos fue desarrollada por Vladimir J. Lumelsky [15], y se ha cogido para el presente proyecto, pues su código era abierto.

4. Una vez obtenida la distancia del fotograma que determinará el riesgo, dependiendo del valor en píxeles de esta distancia se establece un índice de riesgo. Para ver el valor del índice en función de la distancia consultar tabla 3.4.

ID	Nivel de riesgo
0	Ningún riesgo, no hay obstáculo
1	Riesgo bajo, presencia de obstáculos a 70 píxeles como mínimo (30 cm)
2	Riesgo medio, presencia de obstáculos a 30 píxeles como mínimo (10 cm)
3	Riesgo alto, distancia inferior a 30 píxeles

Tabla 3.4: Tabla de índice según distancia

La figura 3.29 representa el tiempo requerido por 100 muestras o fotogramas en realizar todos los pasos comentados en este bloque de Captura y análisis

de imagen para obtener índice de riesgo. Se concluye que el tiempo nunca supera los 0.17 segundos, una velocidad bastante razonable para la aplicación que se está diseñando. Este tiempo también marca la frecuencia con la que se envían informes sobre la seguridad del robot. Se enviarán aproximadamente seis informes por segundo, uno cada 0.2 segundos. Con esta velocidad se puede afirmar que Matlab está ofreciendo respuestas en tiempo real.

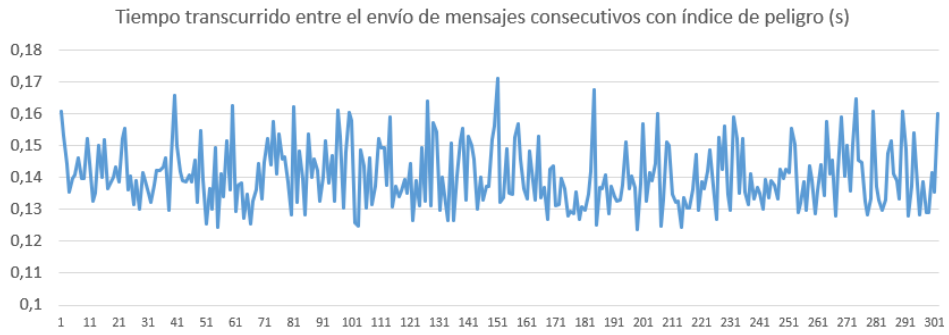


Figura 3.29: Tiempo transcurrido entre mensajes con informe de riesgo enviado al robot. Se han utilizado 300 muestras.

3.5. Resultados y limitaciones del bloque de análisis de imágenes

En primer lugar aparecen ilustradas las etapas de un proceso de análisis de un fotograma en el que justo en ese instante se han añadido tres obstáculos. Estos tres obstáculos son un estuche un teléfono y un monedero. En la figura 3.30 aparecen las etapas intermedias. En la primera imagen de este bloque aparece representada la segmentación del brazo del robot. Se aprecia como se ha detectado parcialmente el estuche, pero gracias a las operaciones morfológicas aplicadas posteriormente no genera ningún problema. La siguiente imagen representa la segmentación del primer plano realizada por el modelo de múltiples gaussianas, se observa que no saca resultados claros sobre el primer plano por la aparición de una sombra, pero es importante este resultado pues será el que posteriormente hará que nuestro algoritmo sea adaptativo y memorice estos elementos como elementos del fondo. La imagen 3 representa la segmentación del primer plano calculada por SSIM y la posterior binarización. Por último en la imagen 4 aparece el resultado final de la segmentación de obstáculos o primer plano obtenida de la combinación lógica de las otras tres. En la figura 3.31 aparece el resultado del siguiente paso, la localización de las coordenadas en píxeles de los elementos de la imagen. Estas coordenadas se han representando y unido con segmentos o bordes negros. Por último, comentar que estos elementos dejaron de ser

caracterizados como obstáculos tras un periodo aproximado de 5 segundos pues se trataba de tres elementos completamente estáticos.

Proceso general con varios objetos

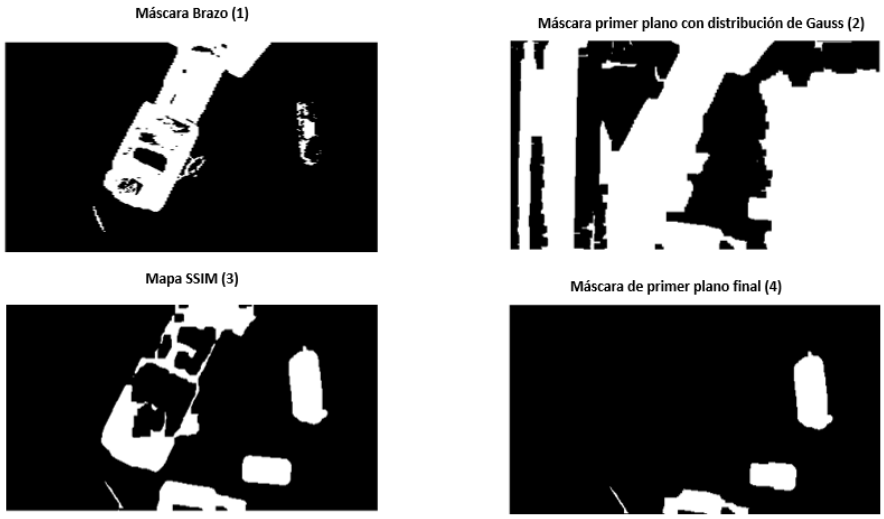


Figura 3.30: Representación de etapas de análisis de imagen de entrada con tres obstáculos. Ejemplo de buen comportamiento del algoritmo.

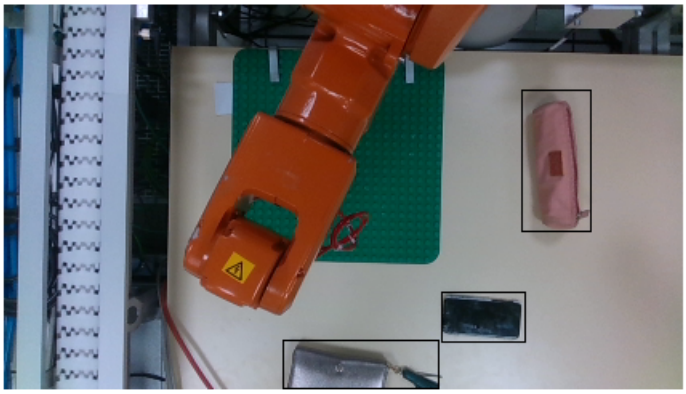


Figura 3.31: Etapa final de localización de obstáculos en área de trabajo.

Si bien el ejemplo comentado se trata de un caso de buen comportamiento del algoritmo, a continuación se comentan las limitaciones del mismo y la

solución buscada para que estas limitaciones sean menos apreciables desde la experiencia del usuario.

1. Se han observado algunas falsas alarmas ocasionadas por el cable rojo que sirve como fuente de alimentación al mecanismo de apertura y cierre de la pinza. Cuando el robot se encuentra en una posición en la cual el cable aparece muy despegado de la estructura del brazo se puede llegar a detectar como obstáculo. Esta falsa alarma pasa cuando el robot está con la orientación representada en la figura 3.32, en esta figura se aprecia como la máscara de segmentación (imagen de la derecha) no ha detectado el cable como parte del brazo y por el contrario, debido al movimiento asociado al robot, la máscara de segmentación final de obstáculos, sí que lo detecta.

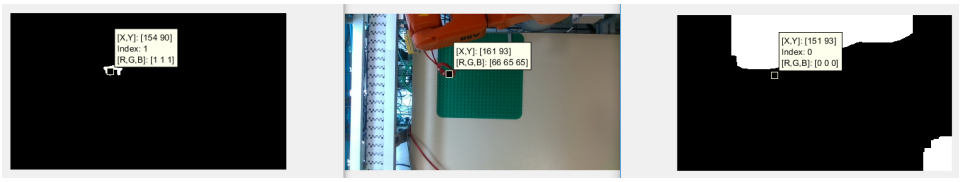


Figura 3.32: Falsa alarma producida por cable de alimentación de la pinza del robot.

   Esta limitación se puede solucionar haciendo que el cable vaya más incorporado a la estructura del robot. No obstante, en la figura 3.33 se observa como el robot ya se ha salido de esa esquina superior izquierda y pese a sobresalir el cable, no sobresale tanto y es detectado como parte del robot y ya en la máscara de segmentación de obstáculos no aparece rastro de este cable.

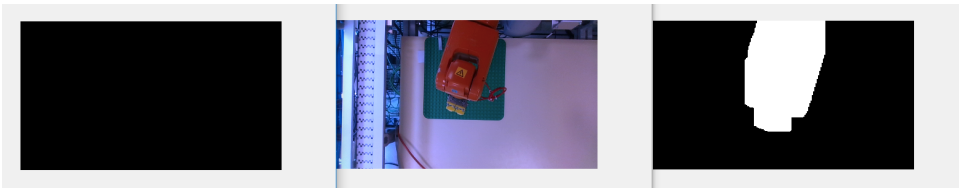


Figura 3.33: Correcto funcionamiento de la segmentación y localización pese a la aparición del cable de alimentación de la pinza.

2. Falsa alarma por pinza y sombra captada con la máscara amarilla. La otra falsa alarma que se ha observado en la comprobación del funcionamiento del algoritmo ha sido cuando en la zona de trabajo aparece solo la pinza amarilla, ejemplo observado en la imagen 3 de la figura 3.34. En la imagen 1 aparece la segmentación realizada con la máscara

de amarillo. Como se observa se ha recogido la pinza y una sombra, en este caso la sombra ocupa mayor número de píxeles que la pinza y por ello en las operaciones morfológicas posterior al eliminar el ruido y dejar únicamente el elemento encontrado más grande, se ha quedado con la sombra, localizando debido a su movimiento, la pinza como obstáculo. Esta limitación no se considera relevante pues surge tan solo en un caso muy atípico en el que el robot procede de su posición de entreno y lleva una postura completamente extendida y además a aparecido una sombra en ese preciso instante.

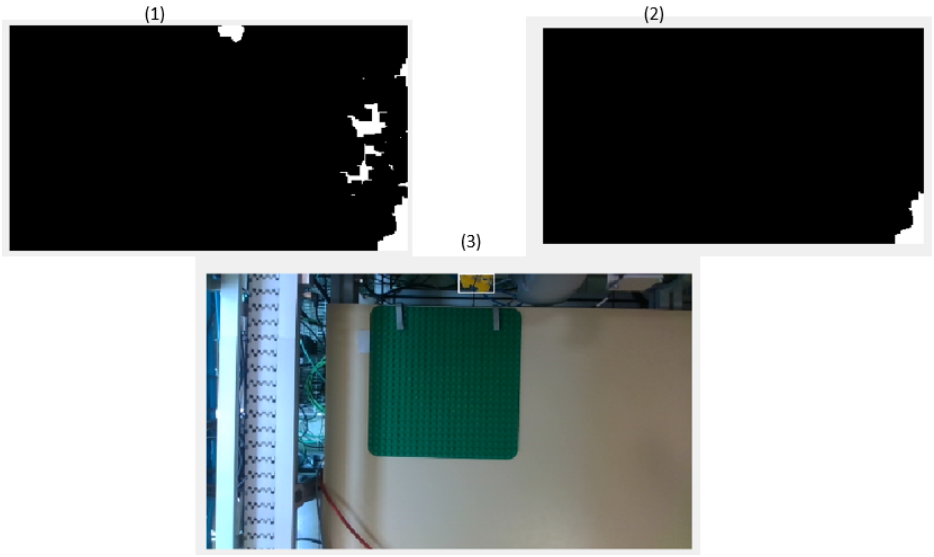


Figura 3.34: Falsa alarma por confusión de la pinza por una sombra detectada con la máscara de color.

3. Un último error observado, más grave pues se trata de una no identificación de un obstáculo. Este ejemplo aparece representado en la figura 3.35. En este caso el estuche rosa no se ha localizado como obstáculo por ser demasiado corto y encontrarse en el área de seguridad del brazo, se ha eliminado como parte del brazo. De este resultado se concluye que un obstáculo situado debajo del robot y que no sobresalga demasiado del perímetro de seguridad ocupado e identificado como robot no será detectado, lo bueno de este fallo del sistema es que para que el estuche llegue a ese punto habrá tenido que ser localizado primero separado al robot por el recorrido que ha debido seguir para llegar a ese punto.

Figura 3.35: Ejemplo de fallo en el análisis de fotogramas por obstáculo no identificado.

Pese a las tres limitaciones comentadas, se debe remarcar que se dejo el robot funcionando con distintos programas principales que realizaban diferentes recorridos, desplazándose el robot por todo el área de trabajo. con el sistema de seguridad trabajando en paralelo, la prueba se realizó dejando funcionar el robot durante tres minutos sin añadir obstáculos, el numero de falsas alarmas fue nulo. Las condiciones que se dieron fueron:

- Por la mañana a las 10:36 con luz artificial.
- Por la mañana 10:50 luz natural intensa únicamente, con tonos anaranjados.
- Por la mañana 11 con luz natural intensa y artificial
- Por la tarde a las 18:00 con luz artificial.
- Por la tarde a las 18:30 luz natural únicamente e intensidad baja.
- Por la tarde a las 18:43 con luz natural y artificial

Capítulo 4

Comunicación entre Matlab y el Robot

El *Flexpendant* es el elemento encargado de comunicar al hombre con la máquina y viceversa. Consiste en un mando, con una pantalla táctil y distintos botones con los que poder programar, configurar, e incluso monitorizar el estado del robot. Para poder integrar el sistema de seguridad en el robot y que este pueda actuar de diferente manera según el peligro que exista por colisión, debe existir una comunicación entre este controlador y Matlab, esta última plataforma como ya se ha comentado múltiples veces es la encargada de determinar el índice de riesgo. En este capítulo se especifican las características necesarias para establecer una comunicación entre Matlab y el controlador del robot. Se decidirá que protocolo implementar y finalmente se detallarán las instrucciones necesarias en ambos lados para establecer dicha comunicación.

La comunicación se ha planteado de la siguiente manera: Se tratará de una comunicación unidireccional con Matlab como origen y *FlexPendant* como destino. Como se especificaba en el capítulo 2, se mandarán tres tipos de mensajes; para solicitar conexión, para avisar de que el periodo de entrenamiento ha finalizado y puede comenzar a realizar las tareas automatizadas, y un mensaje de informe sobre la situación de seguridad. Los dos primeros se enviarán una única vez al arrancar el sistema, por el contrario, el informe del estado de riesgo del sistema se enviará periódicamente cada vez que finalice el procesado de un fotograma, indicando el grado de riesgo al que está expuesto el robot en ese preciso instante. El estudio de la frecuencia media de envío de mensajes está expuesto en la figura 3.29 donde se expone el tiempo medio transcurrido entre el envío de datagramas pues se corresponde con el tiempo en procesar un nuevo fotograma. Desde el otro lado, el controlador accederá al *buffer* de entrada para leer el primer mensaje situado en la cola cada 0.2 ms, una frecuencia de lectura más lenta haría que la cola a la

entrada se hiciera demasiado grande y la velocidad de respuesta fuera muy lenta por parte del robot. Si que se intentó aumentar la frecuencia de lectura de *buffer* de entrada desde el servidor, pero no fue posible por limitaciones temporales en el programa diseñado. Por otro lado, en esta aplicación, no se considera crítica la perdida de datagramas, ya que cada 0.2 ms como máximo, se volverá a mandar un nuevo informe sobre la situación de seguridad y por tanto la perdida de un datagrama solo supondrá un retraso de 0.4 ms en la respuesta del robot.

4.1. Protocolo de comunicación

El primer paso es discutir sobre el estándar de comunicación utilizado para el envío y recepción de mensajes entre ambos. El algoritmo de procesado de imagen se ejecuta en un ordenador de la red cableada propia del laboratorio, del mismo modo que lo hace el controlador. Esta red utiliza el protocolo IP con el dominio 192.168.56.0/24 y será la red que se utilice para la comunicación. El protocolo es no orientado a conexión y no posee ningún mecanismo para corrección de errores. El controlador IRC5 incorporado en el *flexpendant* define este tipo de comunicación basada en *sockets* como única opción para la comunicación entre un programa RAPID y un programa en lenguaje m.

El único margen que queda para personalizar la comunicación se encuentra en la capa de transporte. En ella, existen dos posibilidades para la comunicación entre *sockets*; TCP o UDP. La principal diferencia es que el primero es orientado a conexión y el otro no. Se acaba de comentar que no es crítica la perdida de información, pero no deja de ser deseable que esto no suceda. Por ello, se opta por utilizar TCP como protocolo, consiguiendo establecer una conexión reservada y única para la comunicación entre los dispositivos, tolerante a fallos entre otras.

A modo de conclusión, se utilizará como protocolo de control de transmisión el modelo TCP/IP; utilizado en servicios como HTTP o E-mail. Ofreciendo una transmisión de datagramas con confiabilidad a nivel de transporte pero no en la capa de red. En la figura 4.1 se muestra un esquema sobre las capas por las que está compuesto el protocolo a utilizar en el sistema, se muestra qué protocolo actúa en cada una de ellas. Cada una se encarga de determinados aspectos de comunicación y a su vez brinda un servicio específico a la capa superior.

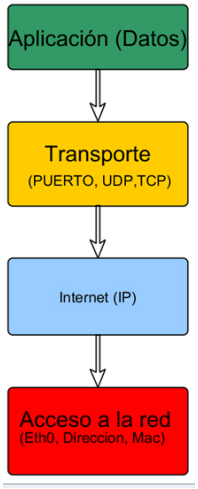


Figura 4.1: Diagrama de capas del protocolo TCP/IP

4.2. Arquitectura de comunicación del sistema

Como es frecuente en aplicaciones TCP, se utilizará una arquitectura cliente/servidor. El cliente se conectará al servidor para realizar una petición a procesar por este último. Para determinar que elemento realizará la función de cliente y cual la de servidor, se deben repasar las funcionalidades que desempeñan cada uno de ellos. El servidor realiza siempre una tarea solicitada por el cliente. Es básicamente un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de respuesta. El cliente que solicita la petición puede o no recibir una respuesta por parte del servidor, estableciendo así una comunicación unidireccional o bidireccional. Tiene sentido, que Matlab desempeñe el papel de cliente, cuyas peticiones serán los informes sobre la situación dentro del área de trabajo. Las tareas a realizar por parte del robot, serán detenerlo, disminuir la velocidad, iniciar la marcha del programa principal o simplemente continuar con ella sin realizar modificaciones. Podría ser útil que Matlab conociese información sobre el estado interno del robot como por ejemplo sobre el programa que está ejecutando de manera paralela al sistema de seguridad o si realmente está procesando de manera adecuada las tareas que le envía. Sin embargo, en el proyecto se ha limitado a enviar tareas sin esperar respuesta alguna.

La entrega de paquetes se realiza como ya se ha comentado, mediante *sockets*. Elementos formados por una interfaz de programación que constituye el mecanismo necesario para la entrega de paquetes de datos provenientes de la tarjeta de red, a los procesos apropiados. Al establecer una red TCP/IP entre ellos, es necesario establecer un *socket* de comunicación, dos si la comunicación fuese bidireccional, uno en cada plataforma para que ambos

recibieran por un puerto claramente definido. Se define con una dirección IP, en nuestro caso la del servidor y un puerto asociado a ella.

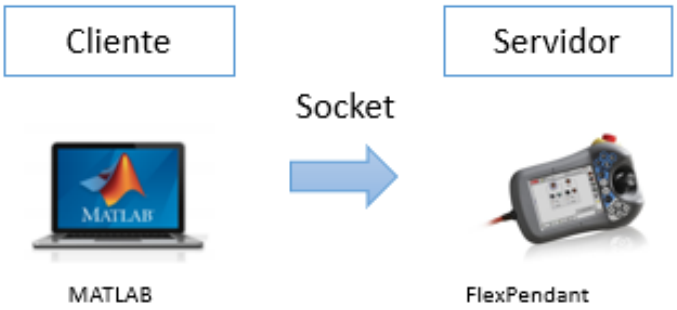


Figura 4.2: Diagrama de bloques de elementos que intervienen en la comunicación TCP/IP.

La comunicación entre el servidor y el cliente, en esta aplicación se puede dividir en cinco etapas:

1. Al arrancar el sistema en el servidor, en este caso el IRC5, habilitará el *socket* conectándolo a la red y permanecerá a la espera de que el cliente solicite la conexión al puerto y dirección adecuada.

2. El cliente o Matlab, inicia la comunicación entre ambos solicitando la conexión. Si el servidor en ese momento está escuchando, aceptará sin problema la conexión, en caso contrario la conexión será denegada.

3. Una vez establecida la comunicación, tal y como se ha diseñado la arquitectura, el controlador enviará al robot la instrucción de dirigirse a un punto conocido como punto de entrenamiento. Una vez en este punto volverá a esperar a recibir otro mensaje. Para ello permanecerá escuchando durante un tiempo indefinido sin realizar ninguna otra tarea hasta la entrada de un nuevo mensaje.

4. Una vez acabado el periodo de entreno, el cliente enviará un mensaje notificándolo. A partir de ese momento, se enviarán entre cinco y siete mensajes por segundo al controlador, estos mensajes informarán sobre el estado de seguridad de la zona de trabajo.

5. El servidor interpretará cada datagrama y actuará al respecto, pero en ningún momento responderá. Como consecuencia, el cliente no tendrá manera de saber si el servidor está realizando correctamente la tarea, simplemente, permanecerá enviando tareas hasta que el usuario decida detenerlo manualmente.

Una vez aceptada la comunicación por parte del servidor, el cliente podrá enviar mensajes aunque el otro extremo no esté escuchando. Estos mensajes se almacenarán en una cola de entrada hasta el momento en el que el servidor acceda a ella o bien se detenga la ejecución del programa. Este punto es muy importante para el presente proyecto en el que el servidor no puede estar escuchando constantemente, pues deberá realizar otras tareas en paralelo y éste será simplemente un mecanismo de seguridad adicional. Por otro lado, los dos extremos del enlace de comunicación deben estar conectados en todo momento durante la comunicación.

4.2.1. Implementación del servidor (RAPID)

Una vez justificada la elección del cliente y servidor, en este apartado se estudiará la implementación del servidor, es decir el bloque derecho del diagrama de bloques de la figura 4.2. La programación de este bloque se realiza en lenguaje RAPID, lenguaje propio de la compañía ABB y que permite fundar una comunicación basada en *socket* entre un ordenador y un programa de control del robot.

Para poder realizarse la conexión, es necesario conocer la IP y el puerto de enlace. La IP es única y fija para cada dispositivo. El puerto, que podría entenderse como el canal por el que circularán los datos, tiene una mayor flexibilidad. Es cierto que es posible configurar cualquier puerto, pero algunos de ellos están ya asignados para otro tipo de tareas. La utilidad de cada uno es:

- Entre el 0 y el 1023. Solo pueden ser utilizados por procesos del sistema.
- Entre el 1024 y 49151. Pueden ser utilizados por procesos de usuario.
- Entre el 49152 y 65535 Son dinámicos y privados.

Por tanto se escogerá un puerto que se encuentre en el segundo rango y más concretamente el 1024, típica elección para las comunicaciones entre aplicaciones. La rutina necesaria para abrir la conexión se muestra a continuación, donde en un primer lugar como cabe esperar se crea el servidor, al cual se le asocia un puerto y una dirección. Tras la ejecución de *SocketListen*, el servidor se encuentra a la escucha de peticiones para conexión. La siguiente instrucción detendrá el programa hasta la entrada de una conexión que aceptará.

```
1  PROC conexion()
2          SocketCreate server;
3          SocketBind server, "192.168.56.93", 1024;
4          SocketListen server;
```

```
5           SocketAccept server, client;
6           connected:=TRUE;
7   ENDPROC
```

Cuando se desee apagar el sistema, se debe cerrar el *socket* con la instrucción mostrada a continuación. Una vez cerrado si se quiere volver a crear una comunicación se deberá crear uno nuevo.

```
1       SocketClose server;
```

4.3. Implementación del cliente (Matlab)

Matlab funciona como cliente. El soporte del cliente TCP/IP, utilizado por la plataforma, establece una comunicación de Socket RAW. Se trata de un tipo de *socket* que no están ligados a ningún protocolo de capa de red o de transporte. Por el contrario, Matlab solo ofrece la posibilidad de modificar la parte del mensaje conocida como *payload*, no se puede leer ni escribir nada de la parte de cabeceras. El flujo de trabajo de la aplicación de puede dividir en 4 etapas:

1. Creación de conexión TCP/IP a un servidor. Se creará un objeto del sistema más, en este caso para la comunicación y se abrirá la conexión con este objeto. Las instrucciones para crearlo y abrir la conexión son las siguientes:

```
1           tcp=tcpip('192.168.56.93', 1024);
2           fopen(tcp);
```

2. Configuración de la conexión. Algunos de los parámetros que se pueden modificar para la comunicación son; *timeout* o tiempo de espera para completar las operaciones de lectura y escritura en segundos, *BytesAvailable* para conocer el número de bytes disponibles en la cola de entrada del servidor o *ConnectTimeout* para establecer el tiempo máximo en segundos de espera a que una solicitud sea aceptada o fallada. En esta aplicación no se modifica ningún parámetro de la comunicación, por tanto este paso se podría omitir y sus valores por defecto son 10 segundos para el *timeout*, 0 *bytes* en *BytesAvailable*, si el cliente en la aplicación recibiese mensajes sería aconsejable aumentar el *buffer* de entrada, y *ConnectTimmeout* infinito [17].

3. Envío de mensajes al cliente, para ello se utilizan operaciones de escritura disponibles con la función `fwrite()`. Esta función detiene el programa hasta que se ha enviado toda la información del mensaje.

4. Cerrar conexión. El último paso a realizar cuando se vaya a desconectar el sistema de seguridad, será cerrar la conexión. Para ello, Matlab dispone de la función `close()`, con el objeto del sistema creado como argumento de entrada.

4.4. Interfaz Gráfica de usuario de MATLAB

Adicionalmente, se ha creado una interfaz gráfica para facilitar al cliente el uso de la aplicación. Las opciones que se incluyen en ella, son básicamente dos botones. Uno de ellos permitirá al usuario encender o apagar la aplicación. El otro, facilita la posibilidad de que el usuario pueda hacer saltar la alarma manualmente adicionalmente al sistema de visión artificial y así detener el robot en situaciones en las que no se ha detectado peligro. El área de trabajo que está siendo capturada y analizada por la aplicación también se muestra en la interfaz con el objetivo de que un usuario pueda controlar la seguridad del robot desde el puesto de trabajo donde está ejecutándose Matlab. El aspecto de esta interfaz se ha representado en la figura 4.3. Para crear la interfaz se ha utilizado la herramienta de Matlab conocida como GUIDE. Este constituye un entorno de desarrollo GUI que proporciona herramientas para diseñar interfaces de usuario a alto nivel de programación, es decir deslizando bloques. La funcionalidad de los componentes (en el caso a tratar, únicamente botones) si que será necesario introducirla a nivel de código. El sistema de seguridad se puede arrancar con o sin la interfaz gráfica. Si se decide utilizar la interfaz gráfica, el envío de mensajes deberá realizarse a una frecuencia menor, pues para poder mostrar en la interfaz la imagen que está siendo capturada es necesario añadir una pequeña pausa pues de no ser así se bloqueaba por incompatibilidad de tiempos. Esta pausa como se verá en el capítulo de resultado, no viene nada mal, pues el controlador del robot es mas lento y Matlab a la velocidad inicial presentada en la sección 3.4.6 acumulaba muchos mensajes en la cola de entrada del servidor y en momentos en los que se pasaba de estado seguro a alarma había un pequeño retardo en la respuesta ya que había mensajes anteriores aún en cola.

Otro problema que presenta esta interfaz y que no se ha conseguido solucionar es que cuando el usuario pulsa el botón de alarma, el controlador tarda más en reaccionar a cuando se le envía un mensaje directo desde el bloque de procesamiento de imágenes. Concretamente cuando se probo sobre un programa principal que movía al robot a tres posiciones distintas consecutivamente y en bucle, al pulsar el botón de alarma el robot se detenía al llegar a una de estas tres posiciones mientras que cuando el programa automática-

mente analizaba esta situación de máximo riesgo, detenía el robot a mitad del camino.

Figura 4.3: Interfaz gráfica para el usuario

Capítulo 5

Procesado de datos y gestión de interrupciones por parte del Robot

La programación del robot se realiza en lenguaje RAPID, directamente sobre el controlador del robot. Consiste en un fichero de texto que se puede editar desde la plataforma de RobotStudio. Se recuerda que el objetivo de la aplicación consiste en, dependiendo de los resultados obtenidos en el procesamiento de imágenes, el robot sea capaz de detenerse o reducir su velocidad ante situaciones de peligro. El sistema diseñado en Matlab genera como salida para cada fotograma analizado un índice de riesgo de colisión. Este índice se transforma a un byte que es enviado al robot. El parámetro puede tomar 4 valores, el robot actuará en función al parámetro recibido y permanecerá en ese estado hasta que reciba un nuevo datagrama. En la tabla 5.1 aparece explicada la reacción del robot frente a la llegada de los mensajes con diferente índice de riesgo. Podrá reaccionar de cuatro maneras distintas: detenerse, reducir su velocidad actual en un 20 % o 70 % o simplemente continuar la marcha del programa principal sin alteración alguna.

ID	Objetivo
0	Todo seguro, no hay ningun objeto
1	Existen objetos a distancia superior a 30 cm, la velocidad debe reducirse en un 20 %
2	Existen objetos entre los 30 cm y 10 cm de distancia, riesgo medio, debe reducirse la velocidad en un 70 %
3	Situación de máximo riesgo. Detener el robot.

Tabla 5.1: Índice de riesgo.

El capítulo comienza con una breve introducción general sobre el lenguaje con el que se programa la parte del robot, la siguiente sección se centra en el funcionamiento de las interrupciones en este lenguaje, pues será la base del desarrollo de este programa dentro de la aplicación. Finalmente se explicará el programa utilizado.

5.1. Lenguaje de programación Rapid

RAPID (Robotics Apllication Programming Interactive Dialogue) es un lenguaje de programación textual de alto nivel diseñado por la empresa ABB para su uso exclusivo en el manejo de sus robots. Un programa RAPID está formado por un programa principal y un conjunto de módulos a los que irá llamando. Este programa nos permite predefinir el modo de funcionamiento del robot, utilizando funciones diseñadas por la compañía para su movimiento o el establecimiento de entradas y salidas digitales o analógicas. Otro concepto importante en este lenguaje es el de rutina, se trata de un conjunto de declaraciones de datos, seguidos de instrucciones utilizadas para implementar una tarea y que se invocarán desde el programa principal o *main*. Se pueden utilizar tres tipos básicos de rutinas:

- **Procedimientos:** No devuelven ningún valor. Se utilizan para limpiar el programa principal y son básicamente un conjunto de instrucciones a ejecutar.
- **Funciones:** Devuelven un valor de un tipo concreto y se utilizan de la misma forma que lo hacen las conocidas funciones en otros lenguajes como Java y C.
- **TRAP:** Conjunto de instrucciones a ejecutar cuando una interrupción salta. Cada interrupción será enlazada a una rutina TRAP. Una rutina de este tipo puede estar anidada a varias interrupciones, de manera que cuando alguna de ellas salte, se ejecutarán las instrucciones definidas en la rutina. Cuando finaliza la ejecución el puntero se coloca de nuevo en la instrucción donde estaba cuando salto la interrupción.

5.1.1. Interrupciones

RAPID permite interrumpir el curso de la ejecución del programa principal a partir de una o varias entradas o salidas, digitales o analógicas. Otra posibilidad es interrumpir el programa a partir del cambio de valor de una variable del sistema (*IPers*) o hacerlo con un temporizador para que salte tras cada periodo de tiempo programado. Incluso se puede definir una interrupción

a partir de la posición del robot (*TriggInt*). A la hora de programarlas se deben tener en cuenta unas características comunes a ellas:

- Toda interrupción es cancelada automáticamente cuando el puntero se sitúa al inicio de la rutina enlazada con la interrupción. Deberán ser creadas y reconectadas a la rutina TRAP cada vez que salten.
- Nunca se verán afectadas por caídas de alimentación ni arranques en caliente.
- Una interrupción no se puede conectar a más de una rutina a la vez. Lo que sí que se podrá hacer es dentro de una rutina, en el momento de reconectar la interrupción, podrá hacerlo a una rutina diferente. Por otro lado, si está permitido conectar varias interrupciones a una única rutina TRAP.
- Una vez conectada una interrupción a una rutina, si se desea cambiar de rutina se deberá eliminar la interrupción y volver a crearla.
- Una interrupción que llegue mientras otra está siendo gestionada, se almacenará en una cola o *buffer*. Las interrupciones que se encuentren en la cola o bien lleguen cuando el programa ha sido detenido, no se gestionarán en el rearranque.

Nuestro programa utilizará interrupciones temporizadas, es decir saltarán cada cierto tiempo, leerán el buffer de entrada de la comunicación TCP/IP y en función al primer mensaje en la cola el robot actuará de la manera que sea requerida. Las instrucciones mas relevantes para tratar las interrupciones en el presente proyecto, aparecen ilustradas en la tabla 5.2

Nombre	Funcionalidad
CONNECT	Conecta una interrupción a una rutina TRAP
IDelete	Cancela una interrupción
IDisable	Desactiva todas las interrupciones
IEnable	Habilita el uso de interrupciones
ITimer	Solicita una interrupción temporizada

Tabla 5.2: Instrucciones de RAPID utilizadas para el manejo de interrupciones.

5.2. Programa RAPID desarrollado

El sistema de seguridad se ha diseñado para trabajar en paralelo a un programa independiente automatizado del robot como podría ser uno que ordenase piezas según color. Se compone del programa principal y el módulo de seguridad que tendrá un comportamiento casi paralelo al principal, gestionado con interrupciones. En la figura 5.1 aparece representado el flujograma de este programa. Cuando se incorpora el sistema de seguridad diseñado en el presente proyecto, el primer paso que realice el controlador será desde el programa principal, habilitando un *socket* y conectándolo a la red para esperar la llegada de una petición por parte de un cliente, en nuestro caso Matlab. Se mantendrá en pausa esperando recibir un mensaje del cliente, que servirá a modo de confirmación sobre el otro componente de análisis del entorno. Tras haber recibido la petición de conexión, la aceptará y posicionará al robot para el entreno. Permanecerá en esta posición hasta que el entreno haya finalizado. Para ello recibirá un mensaje. Por tanto, una vez situado el robot en la posición de entreno, el controlador permanecerá escuchando hasta recibir un nuevo mensaje. Una vez recibido el mensaje que indique fin del entrenamiento, se comenzarán dos tareas en paralelo, por un lado, la del programa principal automatizado que podrá variar ya que es independiente al sistema de seguridad y otra con el método de seguridad.

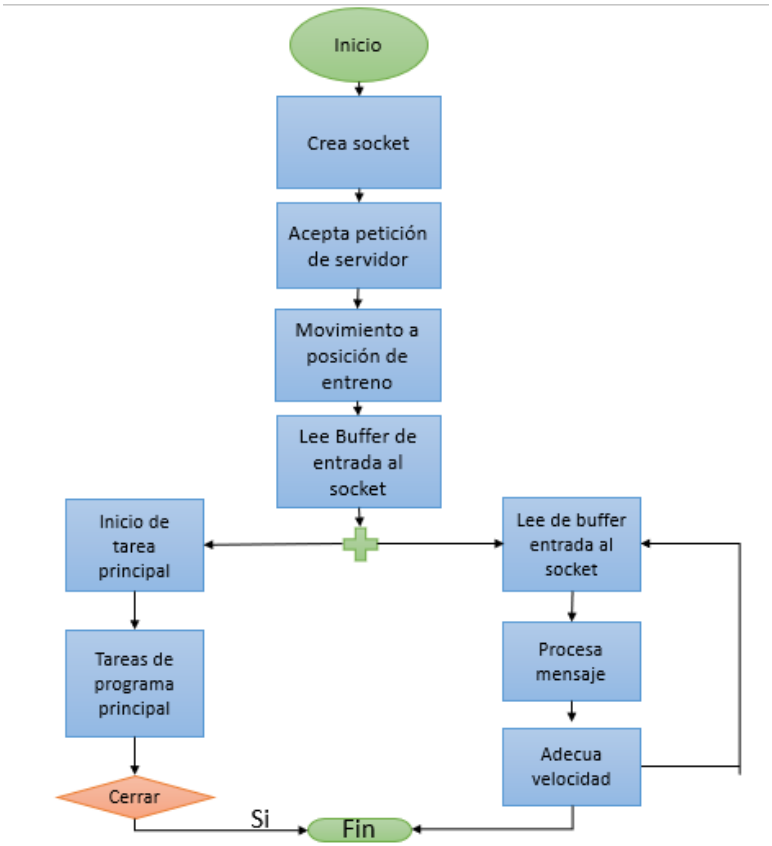


Figura 5.1: Diagrama de flujo del programa implementado en el controlador del robot

El procesamiento de imagen requiere un periodo de entrenamiento inicial para que los modelos estadísticos memoricen el fondo. El sistema se podría haber diseñado de tal manera que aprendiese el fondo con el robot realizando tareas dentro del área de trabajo. Para ello se necesitarían un mayor número de muestras para el entrenamiento y complicaría el sistema. Para reducir tiempo de entreno y simplificar el sistema, al arrancar el sistema, el robot se situará en una posición en la cual la cámara no lo captura y puede memorizar el fondo.

Con respecto al programa del robot, en primer lugar se deberá diseñar una interrupción temporizada que salte a una frecuencia lo más parecida posible a la frecuencia de recepción de mensajes. Junto a la interrupción de diseñará la rutina TRAP que gestione los mensajes recibidos cuando esta la invoca. La rutina TRAP puede iniciar, detener o reanudar un movimiento temporalmente. Para alcanzar esta funcionalidad, deben usarse las intrucciones StorePath, RestoPath y StartMove en la rutina TRAP mencionadas en la tabla 5.2.

Nombre	Funcionalidad
StopMove	Detiene el movimiento del robot instantáneamente.
StartMove	Vuelve a habilitar el movimiento del robot tras haber sido detenido
SpeedRefresh	Ajuste de velocidad para movimiento en curso
RestoPath	Restablece la trayectoria después de una interrupción
StorePath	Almacena la trayectoria cuando se produce una interrupción

Tabla 5.3: Instrucciones de RAPID utilizadas para el control del robot por el sistema de seguridad.

El código utilizado para inicializar la interrupción (intEscucha) y conectarla a la rutina TRAP(recibir_mensajes) se muestra a continuación:

```
1
2     IEnable; !Activa interrupciones
3     CONNECT intEscucha WITH recibir_mensajes;
4     ITimer\SingleSafe,0.2,intEscucha;
```

Para la rutina TRAP, los pasos comunes a realizar son leer la cola de entrada del *socket*, identificar el mensaje recibido y volver a conectar la interrupción. Dependiendo del mensaje identificado se ejecutará una instrucción de las resumidas en la tabla 5.3. Por otro lado, se observa que existe una variable booleana denominada alarma. Esta variable se utiliza pues no es igual pasar a un estado de riesgo 2 desde un estado de riesgo 3 o desde un estado de riesgo 0. Dependiendo de si proviene de un estado de riesgo 3 o de cualquier otro, habrá que leer la memoria de registro de trayectoria para restablecer el movimiento o simplemente modificar la velocidad.

```
1  TRAP recibir_mensajes
2    receivedData:="";
3    SocketReceive client, \Str:=receivedData\Time:=WAIT_MAX;
4    Identificador;
5    TEST ID
6      CASE 0:
7         IF(alarma) THEN
8             Enable_robot;
9             alarma:=FALSE;
10       ENDIF
11       SpeedRefresh 100;
12
13     CASE 1:
```

```
14       IF(alarma) THEN
15           Enable_robot;
16           alarma:=FALSE;
17       ENDIF
18       SpeedRefresh 50;
19
20      CASE 2:
21        IF(alarma) THEN
22          Enable_robot;
23          alarma:=FALSE;
24         ENDIF
25         SpeedRefresh 20;
26
27     CASE 3: !Se detiene el robot
28       alarma:=TRUE;
29       Stopmove;
30
31  ENDTEST
32  IDelete intEscucha;
33  CONNECT intEscucha WITH recibir_mensajes;
34  ITimer\SingleSafe,0.2,intEscucha;
35  ENDTRAP
```

Finalmente el módulo principal del programa principal se adjunta a continuación. En el se realizan los pasos comentados en el diagrama de bloques 5.1.

```
1  PROC main()
2    IF(encendido) THEN
3      INTRODUCIR AQU  PROGRAMA PRINCIPAL.
4    ELSE
5       SocketClose server;
6       SocketClose client;
7       conexion;
8       !Entreno.
9       MoveJ posicion_entreno, v100, z50, tool0;
10      SocketReceive client, \Str:=receivedData\Time:=WAIT_MAX;
11      IEnable; !Activa interrupciones
12      CONNECT intEscucha WITH recibir_mensajes; !Conecta ...
            se al con interrupcion
13      ITimer\SingleSafe,0.2,intEscucha;
14      encendido:=TRUE;
15   ENDIF
16 ENDPROC
```

Capítulo 6

Resultados de los experimentos

Los resultados que se presentan en esta sección son el producto de una serie de pruebas realizadas para medir la calidad de funcionamiento del algoritmo, es decir la cantidad de falsas alarmas, de que no salte alarma cuando debería o de la correcta moderación de velocidad. Para calcular el número de falsas alarmas se ha tenido trabajando al sistema en paralelo a distintos movimientos del robot en las distintas situaciones de iluminación durante dos minutos sin introducir obstáculos y se ha recogido el número de veces que ha saltado alarma. Para el estudio del funcionamiento de la moderación de velocidad se han introducido elementos nuevos estáticos en la escena y se ha estudiado el comportamiento del robot durante el periodo en el que se detectaba como obstáculo y no lo había memorizado como fondo. Para ello se ha desplazado el robot a dos puntos que difieren el uno del otro tan solo en una coordenada. Finalmente para la comprobación del funcionamiento del algoritmo con tan solo dos niveles de riesgo (seguro y detección) se ha comprobado su funcionamiento a simple vista, observando si cuando se introducían uno o varios elementos el robot se detenía.

De esta tabla se deduce que la moderación de velocidad no depende de la intensidad de iluminación o del número de elementos registrados, su bajo porcentaje de aciertos depende más bien del algoritmo para la localización de obstáculos pues en ocasiones es poco exacto. Se registra el mayor porcentaje de errores en el periodo de adaptación del sistema a cambios en el fondo. Es decir, cuando el obstáculo cuando lleva un largo periodo detenido y está pasando a pasar parte del fondo se detectan un gran número de fallos en la localización de las partes que aun no se han guardado como parte del fondo. Por otro lado, se podría asumir que el algoritmo es flexible a los cambios de iluminación, pues tan solo afecta en un porcentaje bajo de falsas alarmas que no ponen en peligro la seguridad del robot. Es preciso comentar que todas las falsas alarmas registradas por cambios en la iluminación se deben exclusivamente a la etapa de detección del brazo.

Situación	Ratio de acierto de los saltos de alarma con algoritmo	Proporción de falsas alarmas entre el total de situaciones sin peligro	Ratio de acierto en la moderación de velocidad
Un obstáculo & Iluminación artificial	100 %	0 %	36 %
Un obstáculo & iluminación natural intensa	100 %	8 %	38 %
Un obstáculo & Iluminación natural débil	100 %	3 %	29 %
Varios obstáculos & iluminación artificial	100 %	0 %	27 %
Varios obstáculos & iluminación natural intensa	100 %	8 %	10 %
Varios obstáculos & Iluminación natural débil	100 %	3 %	29 %

Tabla 6.1: Resultados globales del funcionamiento del sistema

Finalmente, el tiempo transcurrido entre mensajes enviados con informe sobre seguridad del robot aparece representado en la figura 3.29. El robot una vez enviado el mensaje tarda en reaccionar un tiempo cercano a un segundo, medida obtenida a ojo. Este retardo se debe a la acumulación de mensajes a la entrada del robot, pues como se vio en el capítulo 5 se envía un mensaje por cada fotograma procesado. Una buena solución podría ser la comunicación en el caso único de cambio de estado de seguridad, pero en el presente proyecto al haber simulado el trabajo con interrupciones temporizadas, cada 0.5 segundos el controlador leerá el buffer de entrada dando un error en el caso de no haber recibido nada, como no hay manera de establecer un tiempo en el que el estado de la seguridad cambie y el gestor de errores del controlador no nos permite volver a habilitar la interrupción, no se ha encontrado

una solución mejor que aceptar este retraso.

Capítulo 7

Conclusiones

Se ha diseñado un sistema de seguridad aplicando un algoritmo de visión artificial con el objetivo de permitir una actividad colaborativa entre los alumnos y los robots de la minifábrica de ICAI. Finalmente se ha desarrollado un algoritmo capaz de detenerse ante la presencia de obstáculos y recuperar su trayectoria cuando estos hayan desaparecido. Adicionalmente, se ha implementado una mejora que permite en función de la distancia a la que se encuentre de los obstáculos, adecuar la velocidad de sus movimientos. El algoritmo base que únicamente se detenía ante un obstáculo ha sabido responder frente a obstáculos en un porcentaje del 100 %. Por el contrario la mejora, pese a reaccionar siempre ante la aparición de obstáculos, se ha observado que adecua correctamente la velocidad en aproximadamente un 30 % de las ocasiones. Incluso, se ha llegado a registrar algún caso con colisión leve en el que inicialmente el robot no ha tendido a detenerse.

Por otro lado, existen algunas situaciones como se han estudiado, en las que el sistema registra una falsa alarma. Este tipo de sucesos tiene una aparición poco frecuente y suele ocurrir cuando el robot alcanza posiciones raras en las que la máscara de color, elemento menos versatil del sistema desarrollado, no es capaz de localizarlo correctamente.

En lo referente a la comunicación TCP/IP y el procesado de mensajes y actuación por parte del robot se puede clasificar como aceptable. Los mensajes son entregados correctamente, el controlador procesa adecuadamente la información enviada sin registrar ningún error. El único fallo que alcanza es la apariencia de pequeños retrasos en la respuesta del robot, nunca superiores a 1 segundo, por acumulación de mensajes en la cola de entrada del *socket*.

Se puede concluir que el objetivo base de la aplicación se ha cumplido correctamente, mientras que la moderación de distancias registra algunos fallos por motivos de incoherencia, en ocasiones en la etapa de localización de distancias. Otro inconveniente podría ser la escasa adaptabilidad de las máscaras

de color utilizadas para segmentar el movimiento del robot.

Capítulo 8

Futuros desarrollos

El proyecto aquí expuesto presenta numerosas vías de desarrollo futuro, las cuales no han podido ser desarrolladas debido a las limitaciones de tiempo, pero interesantes de implementar en futuros trabajos partiendo de la base creada con este TFG. Se comentarán dos tipos de mejoras, por un lado las mejoras posibles en el algoritmo. Por otro lado, las posibles ampliaciones en las aplicaciones del sistema.

En lo referente al algoritmo, y más concretamente al proceso de análisis de imágenes, las mejoras propuestas son:

- Al inicio de la fase de entrenamiento, el robot se sitúa en una posición fija. El recorrido a este punto es completamente aleatorio y depende de la posición del robot en el momento de arranque. Un desarrollo podría ser que el sistema tuviese la capacidad de determinar un recorrido seguro hacia esta posición de entreno, puesto que es, en cierto grado, poco seguro al encontrarse fuera del área de trabajo.
- Mejoras en la interfaz gráfica como por ejemplo permitir al usuario modificar las regiones que determinen los niveles de riesgo o velocidades del robot así como introducir dirección del robot al que se quiere conectar desde la interfaz.
- Las distancias entre el brazo del robot y cada obstáculo se han calculado en todo momento en 2D, pero es bien sabido que la distancia entre dos objetos en la zona de trabajo, así como en la vida real se debe estudiar con una dimensión mas de profundidad para obtener información más real. También sería interesante realizar un seguimiento de los obstáculos y el robot pues no es igual que dos elementos se encuentren alejandose o acercándose.
- Otra línea de desarrollo interesante podría ser habilitar la comunicación bidireccional entre Matlab y el robot, de tal manera que el robot

pudiera comunicarle a la plataforma encargada del análisis de imágenes sus intenciones de movimiento y en función de ello, calcular con mayor precisión el riesgo existente, pues no tienen el mismo peligro de colisión dos objetos que se acercan al que tienen si se alejan.

En lo referente al robot, mejoraría el comportamiento del sistema encontrar una opción que permita no estar mandando mensajes constantemente y enviar solo un informe cuando se produzca cambio en el estado de seguridad del sistema. Para ello se podría plantear usar alguna de las entradas digitales del sistema pues la información que se envía no es compleja. Con la entrada de una señal digital se podría detectar con facilidad el cambio de estado en la seguridad del robot.

Como nueva funcionalidad, sería interesante añadir al robot la inteligencia suficiente como para estudiar el riesgo en función al recorrido que va a realizar, adaptando el camino seguido para llegar a un punto en función de los obstáculos existentes.

Finalmente, para futuros desarrollos y ampliaciones en la funcionalidades de la aplicación de seguridad, sería interesante buscar una nueva posición de la cámara. Sería interesante localizar un punto desde el cual se obtuviera una visión más amplia, ya que la posición tomada en este trabajo tiene una perspectiva muy buena pero recoge información de un área reducida. Otra posibilidad a estudiar sería integrar la cámara en el brazo y trabajar con procesamiento de imágenes con cámaras dinámicas y rectificación del movimiento.

Apéndice A

Puesta en marcha del sistema

En este capítulo se detalla un manual para el usuario, con el objetivo de explicar el acceso a cualquier funcionalidad de la aplicación. Se explicarán detalladamente cada uno de los pasos para la puesta en marcha de la aplicación.

A.1. Habilitar cámara

Los pasos para poner en funcionamiento la cámara son dos:

1. Conexión física al ordenador con el que se pretende trabajar, mediante puerto USB. Esta cámara no tiene ningún mecanismo de acceso remoto.

2. Habilitar conexión con Matlab. Se consigue gracias a RealSense SDK 2.0, también conocido como libRealSense 2.xx. Para ello se deberá descargar el visor para Window 10, la descarga se puede realizar desde: `https://www.intelrealsense.com/developers/`. A continuación se ejecutará el instalador seleccionando la casilla de descarga del paquete de desarrollo con Matlab.

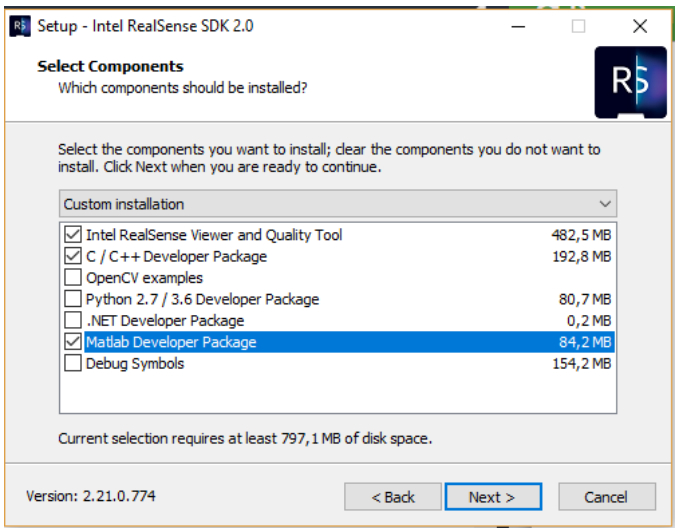


Figura A.1: Pantalla para seleccionar que herramientas de RealSense SDK 2.0 descargar

Finalmente para el uso de este visor, se debe situar la carpeta de funciones que típicamente se guarda dentro del directorio de "Program Files (x86).[en] una ruta que el programa de Matlab sea capaz de encontrar.

A.2. Uso de la interfaz de comunicación a través de GUI

Si se decide arrancar el sistema desde la interfaz de usuario, aparecerá por pantalla algo muy similar a lo mostrado en la figura A.2. El funcionamiento de esta interfaz es muy básico, tiene tan solo dos botones; uno para encendido y otro para hacer saltar la alarma. Al pulsar el botón de encendido, Matlab solicitará la conexión con el controlador del robot y con la cámara. Si ambas conexiones se han realizado con éxito, al usuario se le pedirá que seleccione con el ratón un punto blanco de una imagen mostrada a la derecha de la interfaz. Una vez seleccionado el punto blanco comenzará el periodo de entrenamiento. Durante este periodo el usuario no podrá pulsar en ningún momento el botón de alarma. Al acabar el entreno, el robot comenzará a realizar sus tareas principales. A partir de ese momento, el usuario podrá pulsar el botón y hará saltar la alarma aunque el sistema de inteligencia artificial no haya detectado ningún obstáculo.

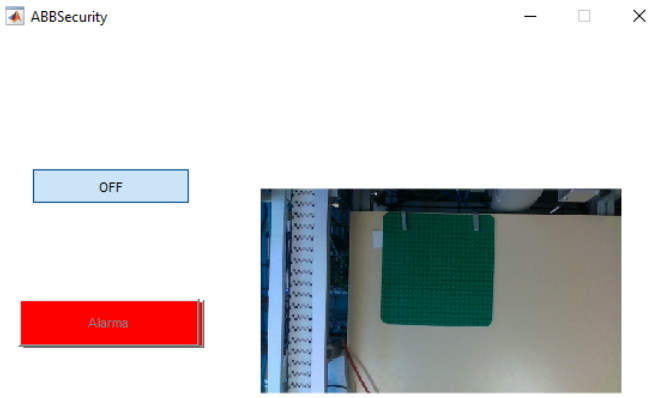


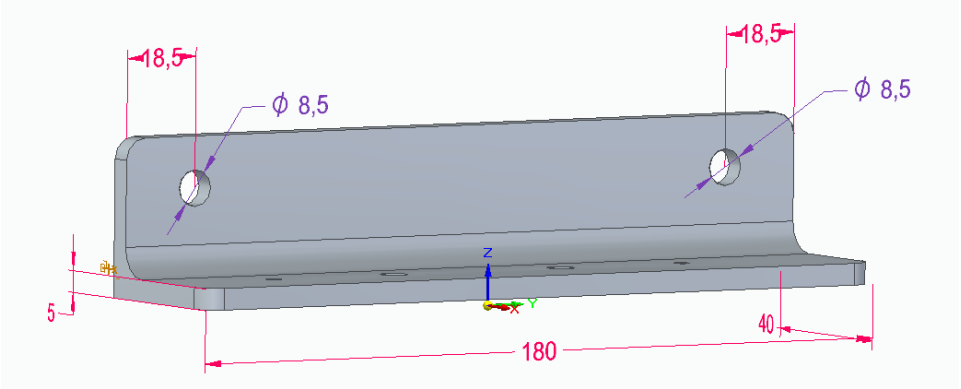
Figura A.2: Interfaz de usuario

Apéndice B

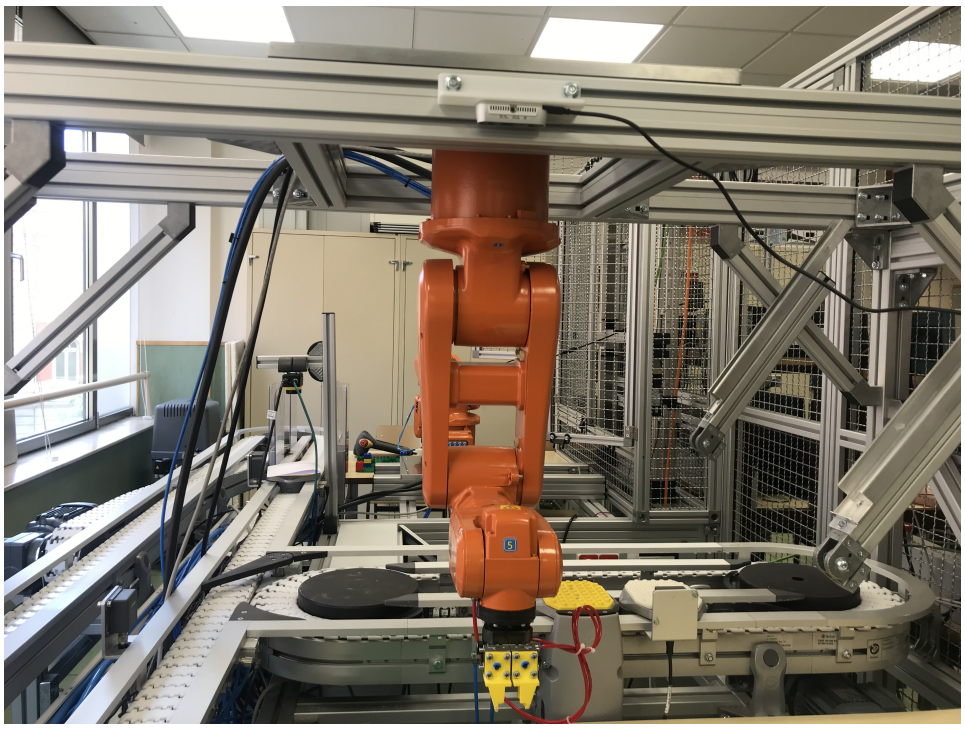
Diseño de pieza

En este anexo se comenta el diseño de la pieza empleada para sujetar la cámara puesto que se trabaja en todo momento con una cámara perfectamente fijada. Cualquier movimiento de la cámara produciría un comportamiento incoherente del sistema.

Como se vio en la capítulo 3 la pieza estará sujeta a la estructura del robot. Para ello se ha utilizado una pieza con la forma representada en la figura **??** y las medidas que aparecen en ella. El material usado para su fabricación ha sido..... Para su diseño se ha utilizado el programa Solid Edge.



La pieza una vez fabricada se ha colocado tal y como aparece representada en la figura **??**.



Apéndice C

Estudio Económico

En este anexo se estudia el coste económico del proyecto, teniendo en cuenta las herramientas necesarias para llevarlo acabo y la mano de obra que ha sido necesaria para el desarrollo e implementación del sistema. Los presupuestos, en especial la mano de obra se ha realizado de manera aproximada. Se estima un coste total de 4.000€ en mano de obra y 14.200€ en elementos que componen el sistema.

Elemento	**€/Unidad**	**Unidades**	€ Total
Robot IRB 120	14.000	1	14.000
Cámara Intel RealSense D435	153,89	1	153,89
Cable extensor USB 3.0.1.8m	4,5	1	4,5
Pieza de impresión 3D	20	1	20

Elemento	**Horas**	**€/hora**	**€ Total**
Programación del robot	80	10	800
Programación del sistema de visión artificial	130	10	1300
Investigación sobre el estado del arte	30	10	300
Sesiones de consultoría con el tutor	20	80	1600

Bibliografía

[1] I.Robots industriales manipuadores, 1994.

[2] ABB. *IRB 120*.

[3] UKIVA Asociación de Vision Industrial del Reino Unido. About UKIVA.

[4] The British Machine Vision Association and Society for Pattern Recognition. Aims and Objectives.

[5] Martin Bichsel. Segmenting simply connected moving objects in a static scene. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16:1138 – 1142, 12 1994.

[6] Gonzalo Silva Blanco, Francisco Avecilla Rangel, Alejandro Rivas Araiza, Manuel Toledano Ayala, Jesus Pedraza Ortega, and Manuel Ramos Arreguín. Desarrollo de un Sistema de Detección de Movimiento basado en Flujo Óptico en Raspberry Pi. *La Mecatrónica en México*, vol. 4(No. 2), May 2015.

[7] A. Chan, V. Mahadevan, and N. Vasconcelos. Generalized Stauffer-Grimson Background Subtraction for Dynamic Scenes. *Statiscal Visual Computing Lab UC San Diego*, 2008.

[8] W.E.L Grimson Chris Stauffer. Adaptive background mixture models for real-time tracking. *The Artificial Intelligence Laboratory Massachusetts Institude of Technology*, 1997.

[9] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, July 2002.

[10] Ahmed Elgammal, Ramani Duraiswami, and Larry Davis. Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25:1499–1504, 11 2003.

[11] FIR. 1.7 Millones de robots nuevos para transformar las fábricas del mundo en 2020, 2017.

[12] Mónica Gallego. Ai Guardman la primera camara de seguridad que funciona con IA. BigData magazine, July 2018.

[13] Lucía Díaz García. *Integración de Visión Artificial en un Robot Industrial*. Universidad Pontificia de Comillas (ICAI), Madrid, July 2018.

[14] I.15066. Robots and robotic devices, 2016.

[15] Vladimir J.Lumelsky. On fast computation of distance between line segments. Information Processing Letters, 1985.

[16] Ota Masaru. Obstacle Detection System with Stereo Cameras for Level Crossings. *Railway Technology Avalanche*, 2004.

[17] Matlab. *MathWorks*.

[18] R. C. Nelson and J. Aloimonos. Obstacle avoidance using flow field divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1102–1106, Oct 1989.

[19] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[20] Pedro M.R Caleiro, Antonio J. R Neves, and Armando J.Pinho. Colorspaces and color segmentation for real-time object recognition in robotic applications. *Revista Do Detua*, 4(8), June 2007.

[21] ABB Robotics. *Especificaciones del producto Controller Software IRC5*, v edition.

[22] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252 Vol. 2, June 1999.

[23] UC3M. *Visión por Computador*.

[24] I.2. W.3. Seguridad Industrial.

[25] M. Watanabe, N. Takeda, and K. Onoguchi. A moving object recognition method by optical flow analysis. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 528–533 vol.1, Aug 1996.

[26] Wikipedia. Temperatura de color.

[27] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[28] Zheng Yi and Fan Liangzhong. Moving Object Detection Based on Running Average Background and Temporal Difference. *Ningbo Institute of Technology*, 2010.

[29] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.