Implementación de un sistema de seguridad para evitar colisiones en un entorno dinámico para un robot industrial

Autor: Rubén Zaera Faus
Tutor: Carlos Ricolfe Viala

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
anys
Escuela Técnica Superior de Ingeniería del Diseño

It's dangerous to go alone!

Take this.

Agradecimientos

A todos aquellos profesores que me han guiado hasta hoy, a mis padres y hermanos por apoyarme siempre y, sobretodo, a mi novia por hacer de editora y estirarme de las orejas cuando lo necesitaba. También quiero hacer especial mención a los compañeros con los que he trabajado durante mis prácticas, he aprendido mucho de vosotros. Gracias a todos.

Resumen

Este trabajo trata sobre el diseño y programación de una aplicación informática, la cual, a partir de la información de la mesa de trabajo que se observa mediante una cámara, permite la generación de una trayectoria aplicable a un robot industrial para ejecutar un pick and place sin colisionar con los obstáculos detectados.

El proyecto se divide en tres sistemas conectados a través de conexión TCP/IP. Estos sistemas son: el robot IRB 140 de ABB, el sistema de visión con cámara de Sherlock y un ordenador con el programa principal en código de Matlab. Es con este último con el que se controla la ejecución y se administra la información. Además, el proyecto posee una interfaz que permite al usuario interactuar con las diversas funcionalidades. Tales como los parámetros de generación de la trayectoria, la visualización de los objetos en el área de trabajo o movimiento del robot en modo manual.

Abstract

This paper deals with the design and programming of a computer application, which, based in the information observed through a camera about the working area, generates a trajectory applicable to an industrial robot. By this method, the robot will execute a pick and place without colliding with the detected obstacles.

The project is divided in three systems connected together by a TCP/IP connection. Those systems are: The IRB 140 ABB robot, the Sherlock vision system with camera and a PC with the main programme, codded in Matlab. It's through the last one, which the rest of the project is controlled.

In addition, the project has an interface what allows the user to interact with the most of its functionalities. Such as the trajectory parameters, the object visualization in the working area or the manual movement of the robot.

Índice

Índice de Figuras

Índice de Tablas

Capítulo I: Introducción

1. Objeto

En el presente trabajo se pretende diseñar un sistema en el que un robot industrial pueda trabajar en un entorno dinámico sin que colisione. Como entorno se considerará un área cuadrada, delimitada manualmente, en la que se pueda modificar sus condiciones: obstáculos, posición del objetivo, movimiento del robot en su interior, etc. En el caso de que se halle en peligro, el robot parará y dará una señal de aviso.

Todo lo expuesto se diseñará en el ámbito del laboratorio de control donde se encuentra el Robot ABB 140T y el sistema de visión artificial, con las herramientas y las capacidades que este brinda.

2. Motivación

Los avances en la automatización de los procesos industriales, siempre se ha complementado con un avance en la seguridad en la industria. Esto se ha debido a que, procesos más complejos, necesitan de mayores barreras para impedir ser perjudiciales para los trabajadores y para el entorno físico en general.

En base a esta idea, durante las últimas décadas se han desarrollado aplicaciones industriales basadas en la tecnología de la visión artificial, se han desarrollado protocolos de seguridad más efectivos y se ha ampliado las capacidades de los robots industriales, tanto sensitivamente, como por la flexibilidad ofrecida. Así, ha aparecido una nueva generación de brazos robots colaborativos, de siete ejes y, además, con sistemas de anticolisiones más avanzadas. Todo ello, se ha orientado en mantener el control del proceso en cualquier situación, pero de una forma rígida, es decir, el programa no variará el trayecto que se ha programado de antemano, aunque sea posible físicamente.

Este proyecto quiere ir más allá de eso. El sistema que se va describir, trata el control de un brazo robot industrial, pero en entornos en los que no es posible conocer todas las situaciones que se pueden dar. Es decir, aquéllas en las que no se conocen de antemano, ni los obstáculos que puede haber en la zona de trabajo, ni sus posiciones. O bien porque la posición depende de un humano o bien porque las posiciones son puramente aleatorias. Estos serán, por tanto, los dos tipos de escenarios que se trabajarán: el trabajo conjunto con un humano, siendo un sistema de seguridad para un proceso colaborativo entre ambos, y el trabajo con posiciones caóticas, teniendo la capacidad de diferenciar el objeto que se desea y la de generar la trayectoria para esquivar al resto.

Para ambas finalidades, este sistema será capaz de decidir y calcular trayectorias diferentes cuando existan obstáculos en el camino, sin necesidad de reprogramar las trayectorias para poder llevarlas a cabo. Es decir, es un sistema que se habitúa al entorno y decide cómo ejecutar el proceso por sí mismo. Una cualidad muy útil en la industria del futuro.

3. Objetivos

Los puntos propuestos a cumplir para ejecutar la idea inicial serán:

- El área de trabajo será modificable manualmente y el interior, tanto los obstáculos como el objetivo, podrá disponerse aleatoriamente sin problemas.
- El robot deberá entrar en un área limitada con seguridad, moverse por ella en una trayectoria 2D para coger un objeto y sacarlo fuera de los límites.
- Se intentará en todo lo posible que el robot no salga fuera, por seguridad, ni tampoco que choque con ningún obstáculo. En cualquiera de los casos, el brazo robot parará completamente a la espera de una acción humana.
- Se intentará que el movimiento, así como el cálculo sea lo más rápido y fluido posible.
- El cálculo de las trayectorias será en tiempo real. En el caso de no cumplirse, el programa deberá prepararse para que sea modificable en este aspecto en un futuro.
- Todo podrá ser controlado por una interfaz gráfica de usuario que permita: Un modo en manual, un modo de visualización del entorno y un modo de funcionamiento automático.

Capítulo II: Conceptos generales

En este capítulo se expondrá en contexto los conceptos útiles para entender la aplicación y todo lo que la envuelve. Toda la información estará comprimida y focalizada en la idea del proyecto.

1. Sistema de Visión Artificial

Se considera un sistema de visión artificial al conjunto que engloba: las cámaras, la iluminación y el procesador, como *hardware* y la programación y el post-procesamiento, como *software*. Todo ello con la función de captar la información de forma visual de un proceso automatizado. Esta información, puede utilizarse para modificar el propio proceso o para guardarla en una base de datos.

Actualmente, y de forma general, este tipo de sistemas se utilizan como sensores, inspectores de calidad o escáneres de códigos. Como ejemplos se pueden apreciar: la selección de fruta en buen estado; los escáneres *datamatrix* o de código barras y las comprobaciones de puntos de soldadura en piezas de metal.

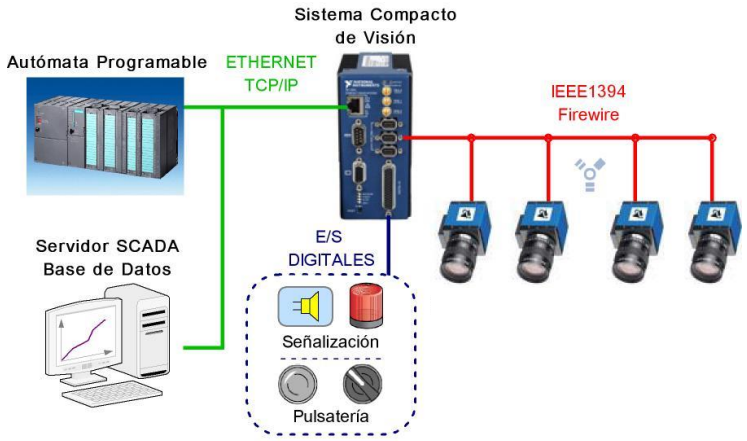

Ilustración 1. Esquema general de un sistema de visión artificial

El funcionamiento general de un sistema de visión artificial es el siguiente:

1. Se capta la imagen, para ello la iluminación debe ser la correcta y la foto debe estar enfocada.
2. La imagen es transmitida al programa de procesamiento.
3. Se aplican los algoritmos de detección necesarios para extraer la información deseada. Este paso depende del fabricante y del software utilizado, ya que el programa incluirá unas u otras herramientas.
4. La información extraída es enviada al autómata o a una base de datos.

La adquisición de imágenes es influida por las entradas desde el autómata y desde las entradas digitales. De esta manera se puede adaptar mejor a las circunstancias.

2. Robots Industriales

Actualmente la robótica y la automatización están presentes en toda la industria, sobre todo en aquellos procesos de repetición continua, es decir, aquéllos que no varían más allá de unos límites bien establecidos.

Existen una gran variedad de máquinas autómatas y actuadores industriales, pero el que interesa para este proyecto es el robot manipulador de brazo de 6 ejes electromecánicos.

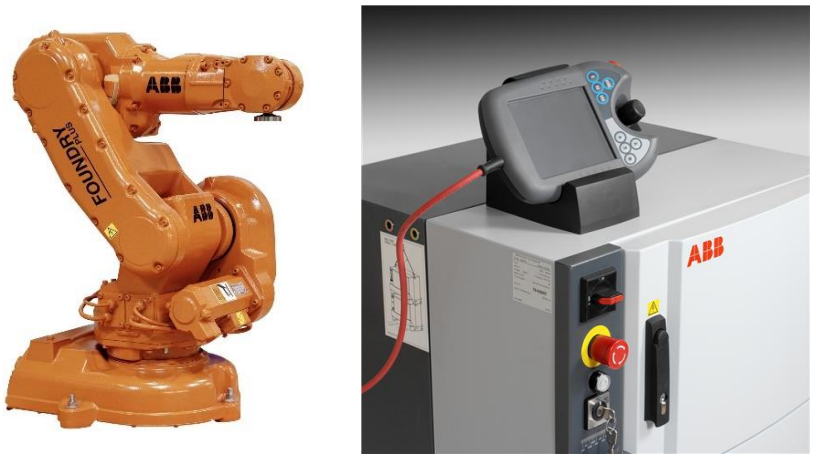

Ilustración 2. Robot IRB 140 y armario utilizados en el proyecto

La mayor característica que posee este tipo de autómata es su capacidad de movimiento en área tridimensional. Es por esta capacidad por la que se popularizó en su momento y por la que se ha seguido utilizando hasta el presente.

Los robots comerciales se componen de:

- Brazo robot: Estructura actuadora. Se compone de 6 ejes con motores electromecánicos
- Armario de control: Puede tener diferentes tamaños, sirve para almacenar todo el hardware de control y seguridad del robot
- Maleta: Dispositivo portátil que, unido al armario de control, permite al usuario acceder a la programación, configuración e información del robot. Es con este dispositivo con el que se maneja el robot de forma manual.
- Software: El programa utilizado, así como el lenguaje de programación depende del fabricante y modelo del robot. Aunque todos ellos operan internamente de la misma forma, se calculan: las posiciones, velocidades y aceleraciones para ejecutar un movimiento de un punto A, a un punto B. Como ejemplos están el lenguaje RAPID de ABB o el KRL de KUKA, los dos, dialectos del lenguaje de programación C.

Por último, queda explicar, que un robot comercial, está preparado para ejecutar varios tipos de movimientos, entre los que se encuentran:

- Absoluto: Se define la posición mediante el ángulo, en radianes de los ejes.
- Simple: Se mueve de un punto A, a un punto B, sin restricciones. Todos los ejes se mueven a la vez.
- Lineal: Se mueve de un punto A, a un punto B, de forma completamente lineal. Este tipo de movimientos, se deben de ejecutar conociendo que todos los puntos intermedios son posibles para el robot.
- Circular: Se mueve de un punto A, a un punto B, pasando por un punto C. Durante este trayecto se traza una curvatura. Le ocurre lo mismo que al movimiento lineal, se deben conocer que todos los puntos intermedios son posibles.

3. PLC (*Programable Logic Controler*)

Un PLC, es el centro de control de cualquier sistema automático moderno, el cual ha sustituido por completo a los armarios de relés utilizados hace tiempo. Lo que le ha hecho tan popular, ha sido la capacidad de ser reprogramado, así como la flexibilidad que aporta la programación, es decir, es útil en cualquier situación y además reutilizable. Si se compara frente a otros tipos de controladores tiene la ventaja de ser más rápido y robusto, diseñado y construido especialmente para aplicaciones industriales y de control E/S. El control de entradas y salidas simultáneas, mayores rangos de temperatura o resistencia a las vibraciones, son algunas de sus características físicas.

Entre las capacidades internas que posee un PLC, cuenta con, por ejemplo, operaciones aritméticas, manejo de señales analógicas o cálculo y aplicación de PID's.

El problema de utilizar este dispositivo es que la aplicación que se quiere diseñar no requiere de demasiada potencia de computo ni ser tan robusta, dado que únicamente se mandan datos de forma síncrona con el sistema periférico de visión y el robot. En resumen, sería sobredimensionar el proyecto, tanto en potencia, como económicamente. En su lugar se utilizará un PC, ya que, uno sencillo y económico puede ejecutar el programa sin problemas.

4. Seguridad en la Industria

La seguridad, es uno de los pilares en los que se fundamenta la industria automatizada actual. Las empresas, deben pasar controles exhaustivos sobre sus protocolos de emergencia, sobre la seguridad, sus estructuras e instalaciones, etc.

Con el fin de mejorar la industria en este ámbito, el presente proyecto se centrará en la seguridad de líneas de producción industrial automatizadas en las que existen agentes físicos y humanos (energía cinética, electricidad...).

Primero cabe explicar que, una línea de producción genérica en la que exista movimientos de robots o de actuadores mecánicos debe estar aislada de cualquier agente externo, el cual puede ser material o humano. El primero, por los daños materiales que se puedan ocasionar y el segundo para evitar cualquier accidente en el que pueda estar involucrada una persona. Para ello, se utilizan elementos estructurales tales como vallas y únicamente se puede acceder al interior a través de puertas especiales para esta función, las cuales se bloquean mientras la línea está en funcionamiento. Además, cuando una de las puertas está abierta, se aprieta la seta de emergencia u ocurre un fallo, cae el voltaje de todo el interior y todos los actuadores se bloquean (contactores de motores y relés de emergencia apagados). Esto sirve para, que en el caso de que una persona quiera entrar al área de seguridad, deba consignar la puerta mediante un candado personal. De esta manera, nadie, excepto ese operario, puede cerrarla y activar la línea de nuevo.

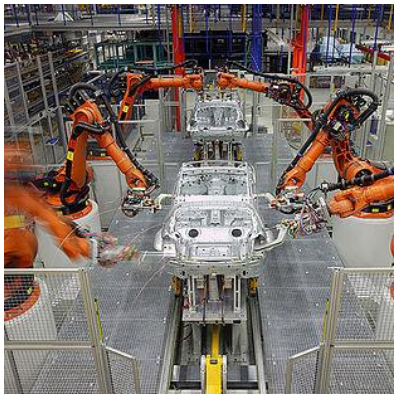

Ilustración 3. Ejemplo de estación de trabajo completamente automatizada

Además de las líneas independientes en las que no hay contacto con los trabajadores, existen las estaciones de trabajo las cuales son afectadas por los dos factores, el humano y el físico. En este caso se recurre a la secuenciación; primero actúa el operario, después la línea (bridas, robots, fresadoras, etc.). Es decir, el operario nunca podrá estar dentro mientras la estación esté en movimiento, ni la estación se podrá activar mientras se esté trabajando en ella. Para ello se utilizan diversos dispositivos como puertas automáticas, escáneres de proximidad, escáneres láser, etc. Todo ello, con la función de captar la posición del trabajador o de aislarlo de la estación de trabajo.

Como se puede intuir, toda la secuencia de acciones relacionadas con la seguridad en este tipo de procesos se encuentra ligada a la programación del mismo. Un error en el código puede suponer un fallo grave en la seguridad. Es por ello que se exigen sistemas de comunicación redundantes y un control estricto desde el controlador sobre cualquier ámbito que pueda suponer un error. Por ejemplo, fallos de comunicaciones con dispositivos o sensores, puertas o botones estropeados

En cuanto a lo que respecta al proyecto, la seguridad es una de las prioridades. Lo que se pretende conseguir, a efectos terministas, es que una persona y un robot estén trabajando a la vez en una misma estación de trabajo sin que exista un riesgo de accidente. Todo ello mediante la visión artificial y cálculos de trayectoria paralelos a la secuenciación del proceso, como se explicará en este documento.

Por último, comentar que en los últimos años han estado apareciendo un nuevo tipo de robots industriales, los colaborativos. Estos robots pueden ser programados para tener una mayor sensibilidad y así reaccionar mejor a estímulos externos. En consecuencia, nos encontramos frente a una gama de robots que tienen la capacidad de trabajar junto a un operario. Sin que esto entrañe algún tipo de riesgo para el mismo. Todo ello provoca que el método de seguridad que se ha comentado no sea totalmente necesario en este nuevo tipo de estaciones colaborativas.

5. Comunicaciones Industriales

Hoy en día, la industria en general, es más compleja dada la organización, la planificación y el control de los procesos que se requieren. Es por ello, que la administración y transporte de información por toda la fábrica es mayor que nunca: entre el personal; entre las máquinas; entre las máquinas y el personal, tanto para diagnóstico como para control de producción; desde dentro la fábrica hacia el exterior y viceversa. Dada esta complejidad, se necesitan sistemas de comunicación y protocolos preparados para ello.

Junto a la industria, estos sistemas de comunicación, han estado evolucionando durante las últimas décadas, creando una gran variedad de estándares que se utilizan actualmente, como Ethernet, TCP, OPC, PROFIBUS, ControlNet o Interbus. Aunque el más utilizado es el estándar Ethernet, el resto son muy utilizados, pero dependiendo de los buses de campo o las tarjetas de comunicación presentes. De la misma forma, también dependen de la aplicación en la que se precise la comunicación, ya que en la estructura de la transmisión de información en planta de una fábrica posee diferentes niveles y en cada uno de estos niveles son preferibles unos u otros estándares. Las redes en los distintos niveles en esta estructura son: red de planta (Nivel de gestión), red de línea (Nivel de control), red de sistema (Nivel de Proceso) y red de campo (Nivel de campo y E/S).

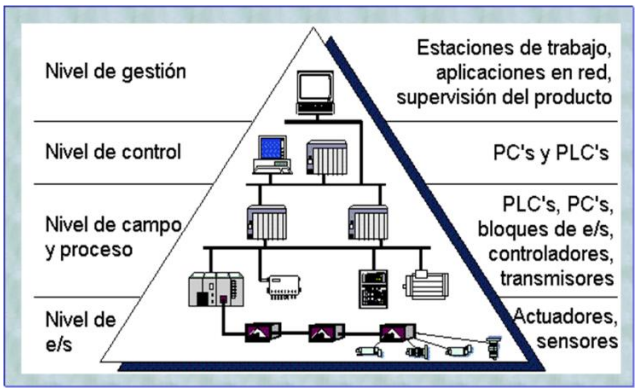

Ilustración 4. Esquema general de las comunicaciones en la industria

- Nivel de gestión: Se trata de la red que no controla directamente el proceso, pero sí obtiene la información del mismo. Es utilizado sobre todo para planificar el funcionamiento general.
- Nivel de control: Es en este nivel donde se encuentran los PC's de control y los PLC's. Además, es en esta red a través de la cual se conecta la línea con los dispositivos periféricos de control y los robots. En este nivel se utiliza la conexión ethernet.
- Nivel de proceso: En este nivel se encuentra la red del sistema. Aquí se encuentran las conexiones entre PLC's de una misma línea de producción y las conexiones con los dispositivos HMI de esa misma línea.
- Nivel de campo y E/S: Es el nivel más cercano al proceso. Se trata de la interconexión de los distintos sensores y actuadores con el PLC.

En cuanto a la aplicación que se detallará en el proyecto, esta, se encontrará dentro del nivel de control y, para sus comunicaciones, se utilizará el conjunto de protocolos TCP/IP.

Capítulo III: Descripción del Sistema

1. Descripción General

El sistema desarrollado consta de tres partes diferenciadas y, en cada una de ellas, se ejecuta una acción concreta. La primera es el sistema de visión artificial, el cual analiza las imágenes y capta los datos que se desean (límites del área, obstáculos y objetivo a mover). A continuación, estos datos son enviados al programa principal mediante conexión TCP/IP

En la segunda parte, en el programa principal, la información es gestionada para generar la trayectoria que debe seguir el robot. Como se explicará más adelante, esta trayectoria se construye en pequeños tramos a partir del algoritmo de campos atractivo/repulsivo.

Por último, la trayectoria es mandada al robot en pequeños paquetes que, al descomprimirlos, le informa del tipo de movimiento que debe ejecutar.

Todo ello está gestionado desde una pantalla de usuario, que permite modificar las variables principales, así como visualizar los datos actuales. El visionado grafico permite tres modos, cada uno con una utilidad diferente: Modo manual, modo de visualización y modo automático

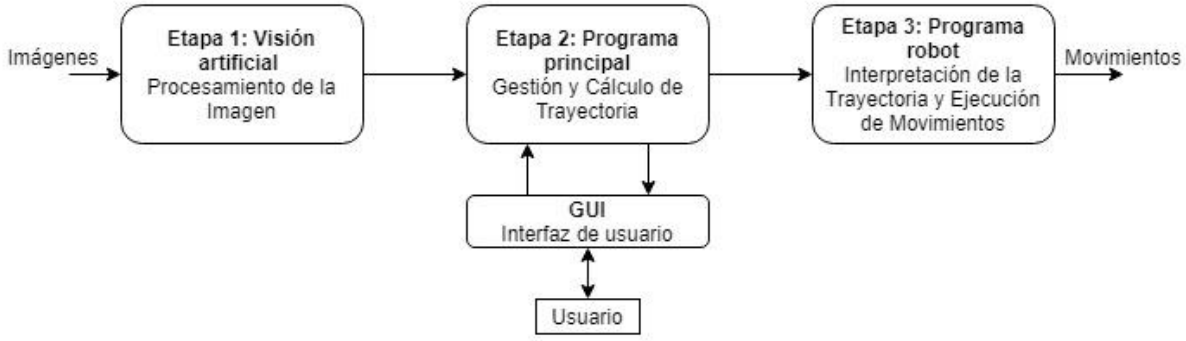

Ilustración 5. Esquema general del proyecto

1.1 Componentes. *Hardware*

1.1.1 Sistema de visión artificial

- Cámara: Cámara situada a 2 m de distancia de la mesa de trabajo, es con ella con la que se captarán las imágenes para el proyecto. Sería válida cualquier cámara de poca resolución mientras sea compatible con el programa.
- PC para ejecutar el sistema de visión artificial: Es un PC del laboratorio con la licencia de Sherlock, está conectado a la cámara y se accede a él mediante un escritorio remoto.
- Iluminación: Consiste en el luz ambiente y el las lámparas del laboratorio.
- Hoja de calibración: Utilizada en el proceso de homografía, explicado más adelante.

1.1.2 Programa Principal

- Portátil con la licencia de MatLab: Se ha utilizado un portátil personal para la ejecución del programa principal, pero sería válido cualquiera que tuviera instalado el programa Matlab y una salida Ethernet.

1.1.3 Robot ABB

- Robot ABB IRB 140: En su conjunto consta del robot, un armario y una maleta
- Mesa de trabajo: Superficie plana y de color blanco en la que se va a trabajar
- Objetos limitadores: Dos objetos circulares fáciles de detectar, es decir, que contrasten con la mesa de trabajo.
- Obstáculos: Se han utilizado diferentes objetos para esta función. Para esta tarea es válido cualquier objeto que no sea del color de la mesa de trabajo.
- Objeto "*Target*": Se ha elegido una lata pintada de negro, dada la facilidad de ser detectada y de ejecutar un "*pick and place*"

1.2 Programas. *Software*

1.2.1 Sherlock®

Software para sistemas de visión artificial, propiedad de Teledyne Dalsa©. Se trata de una herramienta versátil a la hora de procesar imágenes, dada la cantidad de algoritmos de imagen y filtros que posee. También permite la comunicación con otros equipos mediante TCP/IP y servidor OPC.

La licencia incluye la cámara y el equipo de visión artificial.

1.2.2 MATLAB 2015a ®

Software de cálculo, programación y análisis, propiedad de The MathWorks©. Se trata de una potente herramienta multidisciplinar, es usada por estudiantes, ingenieros y científicos.

De todos los *add-ons* que posee Matlab, los que se han utilizado para la elaboración del proyecto han sido:

- Instrument Control Toolbox: Para las conexiones TCP/IP
- GUIDE: Entorno de desarrollo de GUI (*Grafic User Interfaces*)

1.2.3 RobotStudio®

Software de programación de movimientos para los robots ABB, propiedad de ABB©. Posee un lenguaje de programación propio llamado RAPID con unas funciones predeterminadas que ayudan en la tarea.

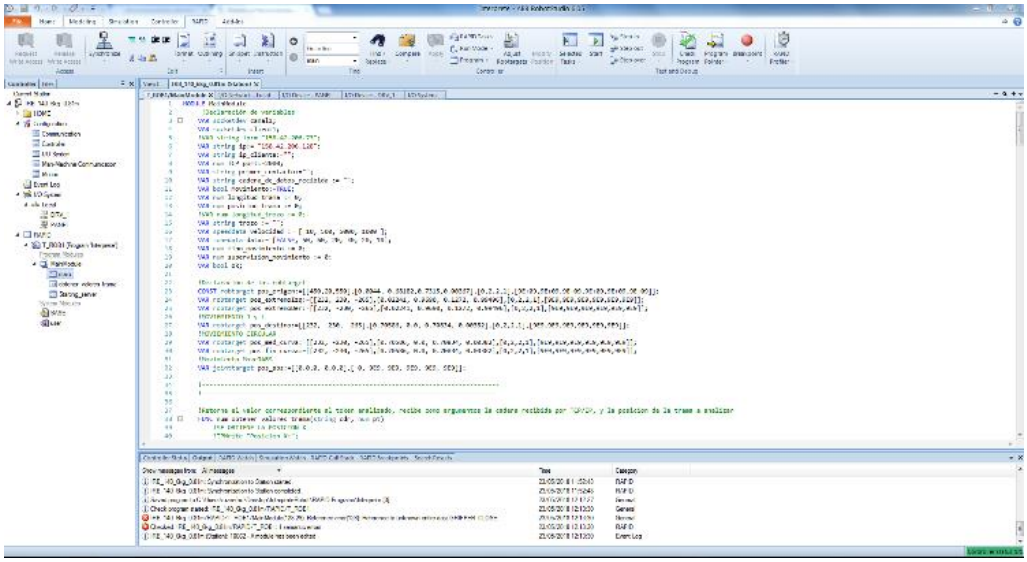

Ilustración 6. Entorno de trabajo del RobotStudio

1.3 Disposición general

En el laboratorio se divide en dos partes. En la primera, que es el área de seguridad, se encuentran el robot, el PC con Sherlock, la mesa de trabajo y la cámara, que enfoca a la mesa desde arriba, a 2 m de altura. Desde esa posición se puede observar un área suficiente para trabajar correctamente.

Y en la segunda parte, que es el área de trabajo, se encuentra el armario del robot y el portátil con el programa MATLAB. Es donde debe posicionarse el usuario mientras el robot este en movimiento.

2. Sistema de Visión Artificial. Sherlock

En términos generales, la parte de visión artificial del proyecto tratará de captar y transmitir los datos que requiera el programa principal. Todo ello de la forma más rápida y fiable posible.

2.1 Programa y Comandos

El programa se ha escrito de forma que pueda detectar y enviar tres tipos de datos, según se requiera para los cálculos en Matlab; calibración del área de trabajo, detección de obstáculos y detección del objeto a mover.

Los datos comunicados dependen del comando recibido:

- "$0" Desconexión. Devuelve "$1" si se ha ejecutado correctamente
- "$1" Conexión. Devuelve "$1" si se ha si se ha ejecutado correctamente
- "$2" Calibración. Devuelve "a,b,c,d", correspondiendo a los datos de los límites de seguridad: X mínimo, Y mínimo, X máximo, Y máximo, respectivamente.
- "$3" Búsqueda de obstáculos y objetivo. La primera respuesta devuelve "Nenv,aux,target,X_f,Y_f" , siendo, respectivamente, número de veces que se enviarán datos, número de obstáculos en el último envío, si se ha detectado el objetivo (1 ó 0) y el punto en coordenadas respecto a la cámara. A continuación se recibirán "paquetes" de datos, tantas veces como se haya expresado en Nenv. Cada uno de ellos tiene la estructura: "P_{1x},P_{1y},P_{1s},P_{2x},P_{2y},P_{2s},P_{3x},P_{3y},P_{3s}" Siendo las posiciones en el plano (X,Y) y valores de las máximas diagonales (S) de tres obstáculos diferentes.

2.2 Modo Calibración del Entorno. Estado Inicial

Lo primero que se pedirá desde el programa principal será la calibración del entorno, para ello enviará el comando "$2". Lo que se pretende en este modo es, por un lado, saber cómo es el entorno sin obstáculos ni *target*, por el otro, conocer los límites del espacio de trabajo.

2.2.1 Foto inicial

La cámara hace una foto del entorno cuando no hay nada en la mesa de trabajo, a excepción de las dos marcas que delimitan el área donde por donde se va a mover el robot.

La foto de referencia será de utilidad en el modo de captación de obstáculos. Se utilizará para compararla con la imagen actual y así detectar los objetos extraños.

2.2.2 Calibración del entorno

Sobre la foto de referencia se aplica un filtro para resaltar las marcas y se ejecuta la herramienta *Connectivity-Binary*. A partir de este proceso se obtendrán las coordenadas de los

límites del área de trabajo respecto al sistema de referencia de la cámara. Además, en base a estos datos, el programa moverá el centro de ordenadas a la esquina más cercana al (0,0), y solo se captará la información de dentro del área. Toda la información que se envíe posteriormente estará basada en este nuevo sistema de referencia.

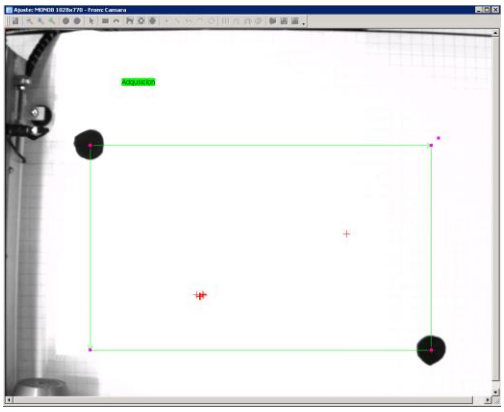

Ilustración 7. Imagen utilizada para ejecutar la calibración del entorno. Se sitúa un *roi* entre los puntos centrales de las marcas

2.3 Modo Continuo

Una vez calibrado el entorno se podrá llamar al modo continúo mediante el comando "$3". En este modo la cámara detectará los objetos extraños y distinguirá el *target*. Sólo lo hará si los objetos se encuentran dentro de los límites marcados en el modo de calibración.

2.3.1 Búsqueda de obstáculos

Primero, el programa empieza haciendo una foto de la situación actual del entorno. A continuación, se le resta a la imagen inicial (obtenida en la calibración) mediante el algoritmo de resta.

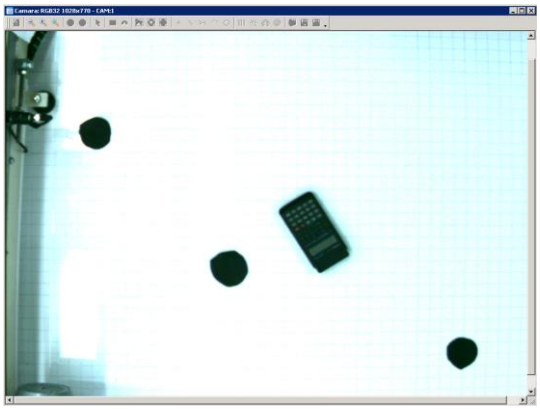

Ilustración 8. Imagen ejemplo. A partir de ella se obtendrá la posición del obstáculo (Calculadora) y la posición del punto final.

El algoritmo de resta calcula la diferencia entre dos imágenes del mismo tamaño y con la misma escala de colores, en este caso mono 8 (valores entre 0 a 256). De esta forma se obtiene una imagen donde los pixeles que son iguales resultarán más oscuros (valores cercanos a 0),

mientras que los que son diferentes resultarán grises o blancos. Para que el algoritmo funcione correctamente debe de haber un claro contraste, es decir, la mesa de trabajo debe ser de color blanco o similar y, a su vez, los obstáculos y el *target* no deben serlo.

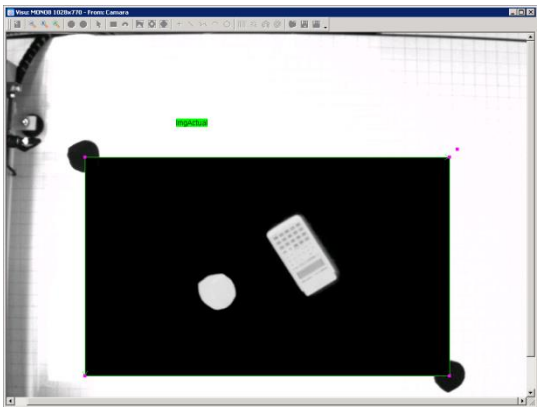

Ilustración 9. Imagen resultante de la resta de la imagen 8 y la de calibración.

A continuación, a la imagen resultante, se le aplica un filtro de contraste que separe claramente ambos rangos (cercanos a 0 y alejados de 0), de esta forma, se genera una imagen binaria con *blobs* blancos en donde están presentes los obstáculos.

Por último, se aplica el algoritmo *Connectivity-Binary* sobre la imagen binaria para así obtener las coordenadas y tamaños de los *blobs* de la imagen. Además, este algoritmo permite discernir entre los objetos útiles (áreas grandes) y los errores que hayan podido suceder (áreas pequeñas).

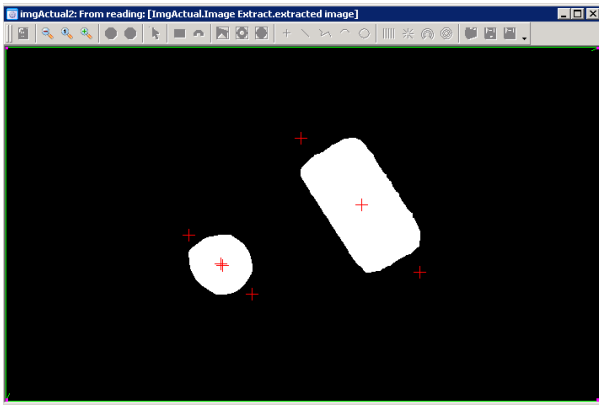

Ilustración 10. Imagen resultante de aplicar un filtro de contraste a la imagen 9.

Toda la información obtenida a partir del algoritmo es almacenada en una matriz de puntos, así como el tamaño de todos los puntos. Además, se calcula la cantidad total de envíos necesarios (tres puntos por envío) y la variable auxiliar (número de puntos en el último envío).

2.3.2 Búsqueda del objetivo

Una vez se ha obtenido la información de todos los obstáculos, se busca el objeto que se quiere mover con el robot. Para ello, se empezará a partir de la imagen binaria con la posición de los obstáculos. En esta imagen se aplicará el algoritmo *Search-Geometric* el cual buscará, entre

todos los *blobs*, cual es el que más se parece al *target*. En el caso de no encontrar ninguno, lo registrará.

Los datos que se enviarán al programa de MATLAB serán: si lo ha encontrado y su posición. Posteriormente, como el *target* también ha sido registrado como un obstáculo, el programa principal eliminará su información de la matriz de obstáculos.

2.4 Homografía y Calibración de la Cámara

El sistema en cuestión trabaja con dos referencias distintas: la del robot y la de la cámara. Cada una con sus ejes de coordenadas y escalas propias. Para poder comunicar la información de una a otra de forma coherente se deberá utilizar la herramienta matemática de transformación homográfica. En este apartado se detallará el proceso y partes para llevar a cabo las transformaciones correctas.

2.4.1 Calibración de la cámara

Primero, se configura el entorno de la cámara. Por defecto el sistema de coordenadas tiene como unidad un pixel, pero lo que deseable es que las medidas obtenidas se expresen en milímetros. Para efectuarlo se deberá aplicar una transformación inicial mediante la herramienta de calibración de imágenes de Sherlock. Esta herramienta permite generar una transformación imagen-realidad de forma manual, que afecta a toda la información de las imágenes en las que se aplique.

En este caso se ha configurado la calibración sabiendo que cada recuadro de la mesa de trabajo corresponde a un centímetro.

2.4.2 Homografía para el sistema de referencia del robot

Una vez el programa principal ha recibido los datos, los ha gestionado y los ha mostrado por pantalla, se deberá ejecutar otra transformación. En esta se traducirá la información de la cámara para que sea coherente con el sistema de referencia del robot. Para este paso se ha recurrido a un código en MATLAB que, en base a varios puntos definidos desde los dos sistemas de referencia, permita la generación automática de una matriz de transformación.

Una vez calculada la matriz de transformación homográfica, esta se definirá dentro del programa principal de MATLAB, en el apartado de configuración inicial. Como se explicará más adelante.

3. Robot ABB

La filosofía utilizada en la programación del robot ABB IRB 140 es la de ejecutar movimientos en base a la interpretación de cadenas de caracteres enviadas por comunicación TCP/IP. Es decir, no va a ejecutar cálculos internos ni modificar la información que le llegue.

3.1 Programa y Comandos

Para la elaboración de este programa he utilizado un código previo, recibido de mi tutor de TFG. Este programa permitía ejecutar cualquier comando de movimiento en base a la configuración que se le enviaba mediante TCP/IP.

Para su inclusión en el proyecto se le ha añadido: la configuración inicial, la conexión y desconexión con el programa principal, la apertura y cierre del *gripper* y la determinación de los parámetros velocidad y zone desde el programa principal.

3.1.1 Comandos

Toda la información que le llega al robot se encuentra en una cadena de caracteres que se analiza cada vez que es recibida ($c m x y z q1 q2 q3 q4 fc1 fc4 fc6 fcx v z). Una vez ejecutado el movimiento o acción se envía una respuesta en forma de otra cadena de caracteres ($0#).

- Variable c. Se trata del comando principal, dependiendo del número puede ejecutar hasta 6 acciones diferentes.
  - "$0". Ejecuta un movimiento lineal (MoveL)
  - "$1". Ejecuta un movimiento libre (MoveJ)
  - "$2". Ejecuta un movimiento circular. Es la configuración del punto intermedio (MoveC)
  - "$3". Ejecuta un movimiento absoluto (MoveAbs) ($3 0 a b c d e f g v z) siendo las variables (a-g) los ángulos en radianes de los ejes.
  - "$4". Ejecuta un movimiento circular. Es la configuración del punto final (MoveC)
  - "$5". Modo desconexión. Debe enviarse "$5 0"
  - "$6". Acción del *gripper*. Debe enviarse "$6 0" para abrir y "$6 1" para cerrar.

- Variable m. Sirve como auxiliar en el caso de los modos 5 y 6 y, en el resto, para el control de movimiento.
    - 0. ConfJ apagado
    - 1. ConfJ encendido
    - 2. ConfL apagado
    - 3. ConfL encendido
- Variables x y z. Indican la posición final del movimiento
- Variables q1 q2 q3 q4. Cuaterniones. Indican la orientación finalista del robot respecto a su base. Dado que el robot se va a mantener constantemente la orientación, estos datos no variarán.
- Variables cf1 cf4 cf6 cfx. Configuración de los ejes. Indican en que cuadrantes se va a trabajar.
- Variable v. Parámetro de velocidad. Se indica la velocidad a la que se quiere mover el robot
- Variable z. Parámetro zone. Indica la cercanía al punto de destino para que se considere que se ha llegado a dicho punto.

3.2 Orientación. Cuaterniones

Dada la necesidad de simplificar el proyecto, se ha optado por utilizar la misma orientación durante todos los movimientos, esta es, una orientación perpendicular a la mesa de trabajo. De esta forma, el robot siempre se direccionará hacia abajo, ya preparado para ejecutar el movimiento de cogida y el de dejada.

Para obtener los cuaterniones de esta orientación, se pueden utilizar las ecuaciones o se pueden obtener de forma experimental desde el FlexPendant, esto es, al mover el robot de forma manual es posible observar las variables de posición y orientación desde la pantalla.

En este caso, se ha decidido obtenerlo de forma experimental, dada la facilidad del proceso. Además no es necesaria la precisión en este ámbito.

El resultado: [q_1=0.00441; q_2=-0.68182; q_3=0.73150; q_4=0.00367]. Estos serán usados en todos los movimientos del proyecto.

3.3 Cuadrantes. Configuración de Ejes

Al igual que se tiene que definir una orientación de trabajo, se debe designar los cuadrantes respecto al que se va a mover el robot. Los parámetros son cuatro: cf1, cf4, cf6 y cfx.

Los tres primeros parámetros designan el rango de trabajo de los ejes 1, 4 y 6, respectivamente. Correspondiendo cada rango a un giro de 90° sobre su eje. El parámetro puede tomar el rango de valores: [-4,3]. En este caso se han utilizado: cf1=0, cf4=2 y cf6=2, validados de forma experimental.

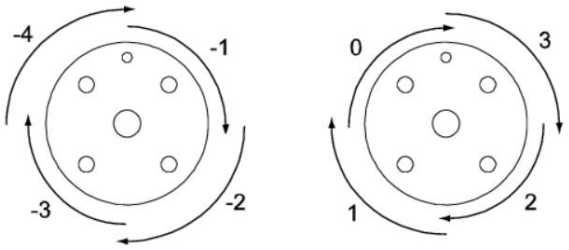

Ilustración 11. Diagrama de la configuración de los ejes.

El último parámetro se utiliza para determinar la posición del resto de ejes a la hora de posicionarse en un punto concreto, existiendo 8 posicionamientos diferentes para cada punto. Mediante este parámetro se designará cómo se comportará el robot a la hora de moverse. Para elegirlo hay que tener en cuenta lo máximo que los ejes se pueden mover en sentido horario y anti horario, ya que, unas configuraciones restringirán el movimiento más que otras. En este caso se ha utilizado al configuración 1, dado que permite al robot atacar desde arriba sin movimientos de ejes excesivos y permite el trabajo en la mesa de forma satisfactoria.

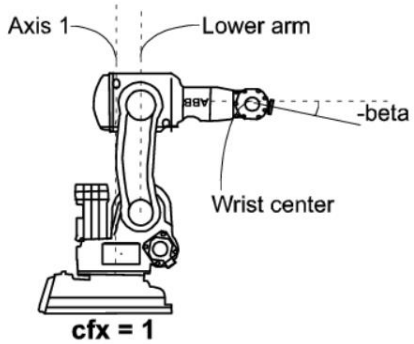

Ilustración 12. Imagen de muestra de la posición de la configuración cfx=1.

Mediante el uso de la configuración de los ejes y de la orientación del robot, se conseguirá que siempre se conozca la manera en la que se va a posicionar el robot y se podrá prever cuando habrá una colisión con un objeto, debido a que siempre trabajará posicionado perpendicularmente a la mesa.

4. Programación MATLAB

El programa escrito en MATLAB es el principal. Es el que, a partir de los datos obtenidos de la visión artificial, organizará la información y generará las trayectorias que, posteriormente, serán enviadas al robot. Todo este proceso estará influido por la información que el operario modifique a través pantalla de usuario.

4.1 Concepción Modular

La programación modular consiste en simplificar cada utilidad en una función diferente, las cuales son llamadas desde el programa principal cuando son necesarias. De esta forma, si se desea modificar una parte del código será mucho más fácil trabajar, ya que, mientras no se modifique la información de entrada y salida de las funciones, no será necesario modificar el resto del programa, sólo las funciones afectadas.

Todo el programa se ha escrito con esta filosofía, pensado para futuras mejoras y para una mejor organización.

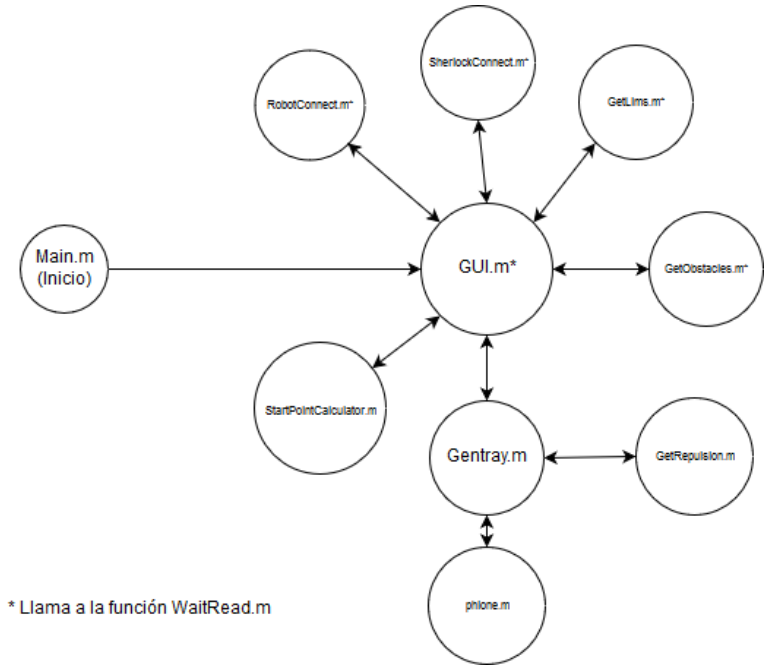

Ilustración 13. Diagrama de módulos utilizados. Las flechas indican a que funciones se llaman.

4.1.1 Tabla de Módulos

Tabla 1. Módulos

Ámbito	Función	Entradas	Salidas	Descripción
General	**Main**			Llama a la función de la pantalla.
	GUI			Función de control de la pantalla. Se utiliza como vínculo entre el usuario y el programa. Es este módulo el encargado organizar el programa, es decir, guarda el valor de todas las variables y llama a las funciones cuando sea necesario.
Conexiones	**RobotConnect**	ip, port	res, t	Función que se llama a la hora de conectarse al robot. Se le introduce el valor de la IP y el puerto y devuelve si se ha realizado (*res*) y el valor del canal tcp/ip, *t*, el cual es recibido de la función de Matlab, *tcpip()*.
	SherlockConnect	ip, port	res, s	Función que se llama a la hora de conectarse al sistema de visión. Se le introduce el valor de la IP y el puerto y devuelve si se ha realizado (*res*) y el valor del canal tcp/ip, *s*, el cual es recibido de la función de Matlab, *tcpip()*.
	WaitRead	t ó s	res	Esta función deja el programa parado hasta que se reciba algo por el canal *t* o por el *s*, en el caso de que sea incorrecta muestra un mensaje por pantalla y para la acción que se esté ejecutando.
Información Sistema de Visión	**GetLims**	t, s, RestPoint	Xmin, Ymin, Xmax, Ymax, res	Esta función se encarga de enviar el robot al punto *RestPoint* y de recibir los datos de los límites, los cuales descomprime y devuelve. Si no se reciben correctamente o no son coherentes, devolverá un 1 en la variable *res*.
	GetObstacles	s	points, Nobs, TargetPoint, Target	Esta función se llama para recibir la información completa del estado de la mesa de trabajo. La matriz de obstáculos (*points*), el número de objetos (*Nobs*), La posición del objetivo (*TarjetPoint*) y si se ha visualizado el objetivo (*Target*).
Cálculos de trayectoria	**StartPointCalculator**	x0, y0, points, N, seg, Target, dy	res, xf, yf	A partir de la posición inicial (una de las esquinas), la matriz de obstáculos, el número de obstáculos, el índice de seguridad, la posición del objetivo y el tamaño del lateral del área, se calcula el punto por el cual puede empezar a moverse el robot. A su vez envia si se ha ejecutado correctamente (*res*).
	GenTray	pos, xf, xy, dif, seg, p0, Katr, Krep, objects, limxMax, limyMax	x, y, col, colpoint	Esta función se encarga de secuenciar las acciones de generación de la trayectoria. Crea un vector utilizando el valor del ángulo *phi* y el módulo del parámetro *dif*. Este es aplicado a la posición actual para calcular el siguiente punto que se mandará al robot. A su vez, se encarga de la seguridad, utiliza la matriz de puntos para comparar las distancias con el punto calculado. Envía la resolución en la variable de salida *col* y el punto del obstáculo en el la variable *colpoint*.
	GetRepulsion	Krep, p0, points, pos, seg	fx, fy	Mediante los parámetros *Krep*, *p0* y el índice de seguridad y junto con la matriz de obstáculos y la posición actual, se calculan las fuerzas de repulsión que afectan en la posición actual.
	Phione	fx, fy	phi	Mediante la fuerza de repulsión se calcula un vector de ángulo *phi*.

4.2 Pantalla de Usuario

La pantalla de usuario está programada mediante la herramienta de Matlab: GUIDE. Esta será la única puerta de acceso a la información del sistema, por parte del usuario.

Servirá para conectarse al robot y al sistema de visión artificial, modificar la mayoría de los parámetros, visualizar los movimientos y cambiar entre los modos.

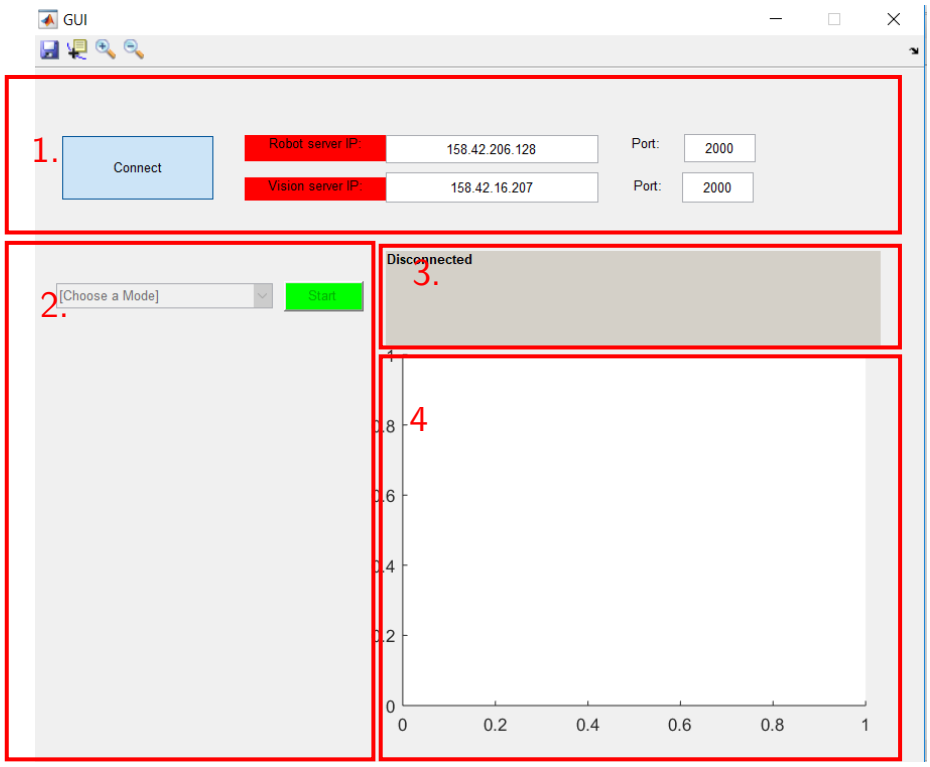

Ilustración 14. Esquema de la pantalla de usuario. Se encuentra dividida en cuatro apartados.

4.2.1 Conexión-Desconexión

Esta sección (Ilustración 14, apartado 1) permite seleccionar la IP y el puerto de las conexiones a los servidores de los sistemas periféricos (robot y visión artificial). Una vez se han aceptado ambas conexiones se permitirá modificar el resto de la pantalla.

4.2.2 Selección de modos. Parámetros

En esta sección (Ilustración 14, apartado 2) se permite cambiar de un modo a otro mediante el menú desplegable. Además, dependiendo del modo seleccionado, el apartado de parámetros cambia.

Los parámetros se dividen en dos secciones. La primera, la de arriba, sirven para el modo manual y automático para modificar los parámetros del algoritmo. La segunda, sirve para modificar la visualización y para ver la cantidad de objetos detectados.

4.2.3 Descripción de modo

Este cuadro de comentario (Ilustración 14, apartado 3) cambia su contenido dependiendo del modo en el que se encuentra. Contiene una pequeña descripción en inglés.

4.2.4 *Display* de información

El último apartado (Ilustración 14, apartado 4) muestra un gráfico con la información de los movimientos y obstáculos en tiempo real.

- Modo Manual: Muestra el punto inicial y final del movimiento mediante dos puntos, así como la trayectoria ejecutada en tiempo real mediante una línea. Las trayectorias antiguas se mantienen hasta que se cambia de modo.
- Modo de Visualización: La grafica adopta los límites marcados. Después muestra los obstáculos junto con su área de seguridad. En el caso de detectar el objetivo, lo mostrará con una cruz.
- Modo automático: Muestra por pantalla la información de los modos anteriores, a la vez.

Los obstáculos que se muestran en los modos Automático y de Visualización, no se muestran en tamaño proporcional. Para corregir este hecho existe un parámetro de visualización interno, el cual se multiplica al valor del tamaño del obstáculo para que se ajuste a la realidad. El valor que se utiliza para el algoritmo no es afectado por este parámetro.

4.3 Parámetros Generales

A continuación, se mostrarán los principales parámetros. Los cuales son modificables desde la pantalla de usuario y afectan al funcionamiento del programa en general.

La tabla 2 está ordenada según a qué aspecto del proyecto afecta el parámetro. En ella también se expresa cuando son modificables por el usuario, junto con una pequeña descripción.

4.3.1 Tabla de parámetros generales

Tabla 2. Parámetros generales

Ámbito	Parámetro	Habilitado/ Modificable	Descripción
Conexión	**Robot IP**	Sin conexión	Define la IP del servidor del robot a la que se intentará conectar
	Robot Puerto	Sin conexión	Define el puerto de la comunicación con el robot a la que se intentará conectar
	Visión IP	Sin conexión	Define la IP del servidor del sistema de visión a la que se intentará conectar
	Visión Puerto	Sin conexión	Define el puerto de la comunicación con el sistema de visión a la que se intentará conectar
General	**Modo**	Conectado	Se modifica mediante el *PopUp menu* de la pantalla permite cambiar de modo mientras no se esté ejecutando el programa. Tanto el programa como lo que aparece por pantalla dependen de este parámetro.
Modo Manual	**X**	Modo manual (Modo=2)	Define el punto final X del movimiento manual, no puede salir del rango preestablecido desde el programa: [420,650]
	Y	Modo manual (Modo=2)	Define el punto final Y del movimiento manual, no puede salir del rango preestablecido desde el programa: [600,-600]
Robot	**Z**	Modo automático (Modo=4)	Valor de la altura en la que trabajará el robot. Se le envia directamente este parámetro por lo que está referenciado al sistema de referencia del robot.
	Error	Modo automático (Modo=4)	Valor de error en trayectoria. En el programa del robot se denomina como *zone.* Cuanto mayor sea el valor, antes considerará el robot que ha llegado a la posición que se le ha mandado.
Algoritmo	**Dif**	Modo manual (Modo=2) / Modo automático (Modo=4)	Valor utilizado por el algoritmo para designar lo lejos que estará el siguiente punto respecto al actual. Se mide en mm.
	Katr	Modo manual (Modo=2) / Modo automático (Modo=4)	Valor utilizado por el algoritmo para designar la atracción de las cargas, en este caso el punto final.
	Krep	Modo manual (Modo=2) / Modo automático (Modo=4)	Valor utilizado por el algoritmo para designar la repulsión de las cargas, en este caso los obstáculos.
Algoritmo y visionado	**Índice de seguridad**	Modo de Visualización (Modo=3) / Modo automático (Modo=4)	Define el valor de la seguridad en base a la cual se parametriza el programa. Se utiliza para: calcular un choque o una salida del área; en la visualización por pantalla y en el cálculo de trayectorias. El valor 1 representa que se utilizan los valores tan y como se reciben. Por ejemplo, si se define un 1.2, los objetos aumentaran un 20% su tamaño y afectarán más su influencia sobre el algoritmo.
	Radio de acción	Modo de Visualización (Modo=3) / Modo automático (Modo=4)	Se suma al radio de los obstáculos. Se utiliza para definir el área de acción de los objetos. Fuera de este radio los obstáculos no tienen efecto sobre el movimiento.

4.4 Programa

En general, el código se divide en cuatro partes: comunicación, modo manual, modo de visualización y modo automático.

4.4.1 Funciones generales

- Conexión/Desconexión

El sistema funciona con dos servidores a los que debe conectarse el programa: el robot y la visión artificial. Los dos deben encontrarse en la misma red, en este caso la red ethernet de la universidad. Para establecer las conexiones, el usuario utilizará el apartado de conexión de la pantalla (Imagen, apartado 1). En ella se puede configurar la IP y puerto a donde se va a conectar y, en el caso de que se haya efectuado correctamente, aparecerá un mensaje de confirmación y enviará el robot a posición de conexión. En ese instante, se desbloquearán el resto de opciones.

En el caso de que la conexión no se efectuara correctamente, se enviaría la señal de desconexión a ambos servidores. De esta forma, todo el sistema volverá al estado inicial y mostrará un mensaje de error.

Para que este proceso se ejecute de forma correcta, los servidores deben de estar iniciados y esperando una conexión. Será entonces cuando se puede proceder al intento de conexión.

- Cambio de modo

Una vez ejecutada correctamente la conexión, se habilitará el objeto de selección de modo. Internamente, el cambio de modo se efectúa habilitando o deshabilitando los botones y *displays numéricos* dependiendo del valor de la variable de control del modo. Una vez se ha cambiado visualmente, el programa espera a que se seleccione el botón de inicio para ejecutarse. En el caso de empezar a trabajar, el cambio de modo queda deshabilitado hasta que haya finalizado.

La variable popupmenu1.Value (variable de control de modo) puede tener cuatro estados: 1-Sin Modo, 2-Modo Manual, 3-Modo de Visualización y 4-Modo Automático.

- *Environment calibration*

Esta función es propia de los modos: automático y de visualización. Al seleccionar el botón en la sección de parámetros de la pantalla, envía al robot a una posición en la que no interfiera con la cámara. Esta posición está pre-programada dentro del código y no puede ser modificada por el usuario.

Seguidamente muestra por pantalla un mensaje de aviso y envía el comando '$2' al sistema de visualización, el cual devuelve todos los datos necesarios para modificar el *display de información* y así ajustarlo a las medidas registradas y, de igual manera, la información de las dimensiones del área de seguridad. Por último, habilita el botón de inicio del modo.

En el caso de no haber detectado correctamente la información, o que esta no fuera coherente, se mostrará un mensaje de error y no se permitirá la activación del modo.

- *Initial Position*

Esta función se utiliza únicamente en el modo manual. Sirve para enviar el robot a la posición de seguridad para después moverlo manualmente desde el programa. Al igual que la posición para la calibración, esta no puede ser modificada por el usuario.

Una vez el robot ha llegado a la posición inicial, aparecerá un mensaje de confirmación y habilitará las funcionalidades del modo manual.

- *Start/Stop*

Una vez se ha seleccionado el modo y se han ejecutado las funciones de *environment calibration* o de *initial position*, se podrá ejecutar el programa principal. En el caso del modo manual ejecutará el movimiento desde la posición actual a la determinada en los parámetros, en el caso del modo de visualización empezará a recibir datos sobre la posición de los objetos en la mesa y los mostrará a través del *display* de información y, en el caso del modo automático, empezará a ejecutar el *pick* and *place* guiado para evitar los obstáculos.

Una vez se pone en marcha el programa el botón cambia a un botón de paro, si es seleccionado, el programa esperará un ciclo de escaneo y envía la señal de paro tanto al robot como al sistema de visualización. En cuanto se haya ejecutado este proceso, la pantalla volverá a la posición anterior a iniciar el programa. El mismo proceso se ejecuta automáticamente cuando los modos manual y automático han terminado su tarea.

4.4.2 Modo manual

En este modo no se tiene en cuenta la información del sistema de visión, su función es crear trayectorias a partir de los datos que se introducen por pantalla. Los movimientos siempre serán entre dos puntos en el plano XY, de forma lineal y, además, no se curvará y no permitirá movimientos que no pueda ejecutar el robot.

La utilidad principal de este modo es probar el generador de trayectorias de forma directa, eso es, el usuario es el que decide los parámetros y a donde se dirige. También se puede usar para poder ejecutar movimientos concretos con el robot en automático, es decir, sin tener que utilizar la maleta en manual.

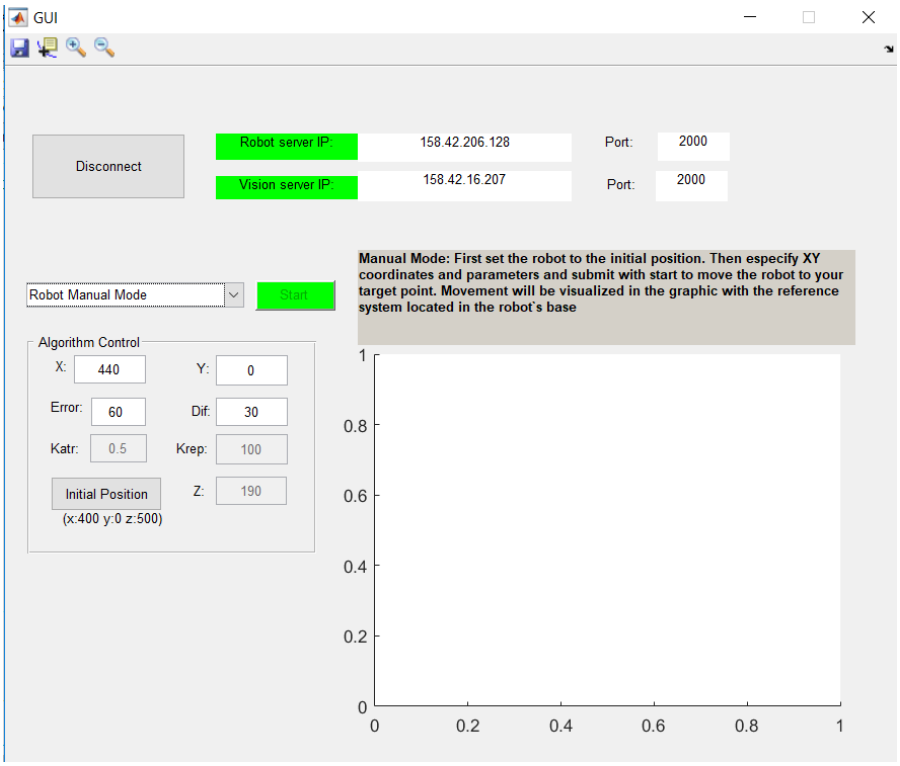

Ilustración 15. Pantalla de usuario. Modo Manual

El funcionamiento es el siguiente. Primero, se lleva al robot a una posición de seguridad inicial pre-programada, en este caso x:400, y:0, z:500. Después, se elige el punto al que se quiere mover, siempre en coordenadas XY respecto al robot, y de la misma forma se eligen los parámetros de diferencial y de error. Siendo estos la distancia de cada subtramo y el error de trayectoria, respectivamente. Por último, se presiona el botón de inicio para ejecutar el movimiento. Mientras está en marcha mostrará la trayectoria en el display en tiempo real, así como el punto de inicio y final de cada movimiento.

4.4.3 Modo de visualización

Este modo sirve para comprobar el funcionamiento correcto de la adquisición de datos por parte de la cámara. Además, permite visualizar la mesa de trabajo en tiempo real, pero con cierto retraso temporal.

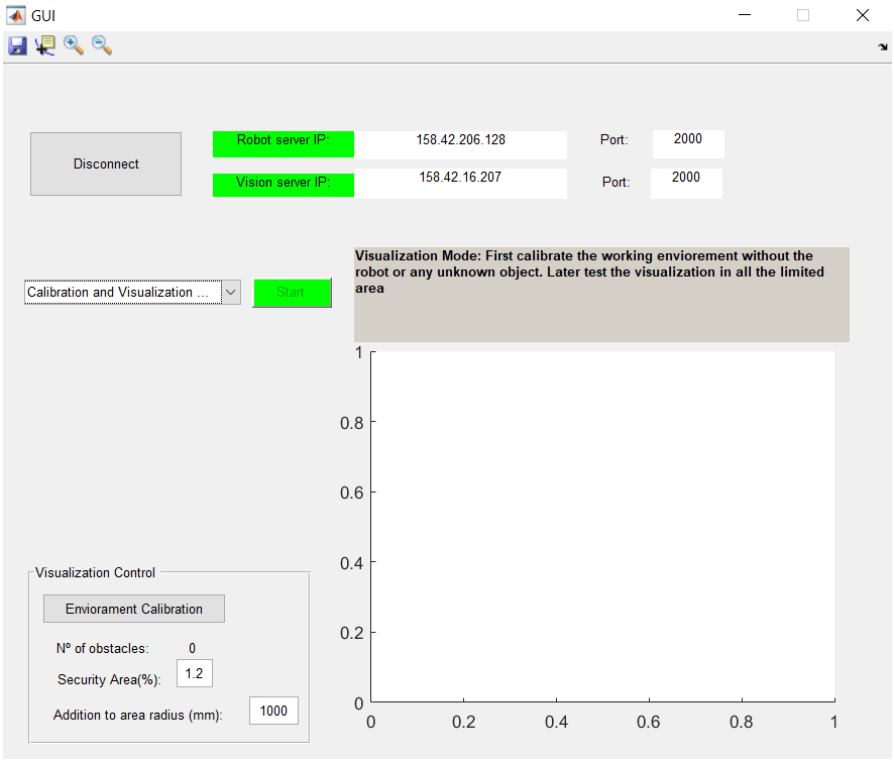

Ilustración 16. Pantalla de usuario. Modo de visualización

La ejecución del modo es el siguiente:

- Calibración del entorno: La calibración es lo primero que se deberá hacer, para ello se despejará la mesa de trabajo y se dispondrán las marcas donde se quieran especificar los límites de visionado. Después se seleccionará el botón *Enviorement Calibration*, con el que empezara la secuencia de calibrado.
- Configuración: En el apartado de parámetros se podrán modificar tanto el índice de seguridad, como la adición en el área de acción de cada obstáculo sobre el algoritmo. En el caso del índice de seguridad, un 1 significa el uso del 100% del valor del radio del objeto (S_x) recibido del sistema de visión y, en el caso del área de acción, es el valor en mm que se suma mismo radio. Ambos parámetros afectan a como se observa en pantalla y a cómo afectan los obstáculos al algoritmo, ver Capítulo III, Apartado 5
- Inicio: Al iniciar el programa se envía el comando: "$3" Búsqueda de obstáculos y objetivo, ver Capítulo III, Apartado 2.4. De forma continua se recibe el estado de los objetos en la mesa de trabajo. Los cuales se reconstruyen de forma visual a través de una gráfica en el *display* de información.

4.4.4 Modo Automático

Este es el modo principal de trabajo. Su función será coordinar los datos obtenidos de la cámara, mostrarlos, convertirlos en movimiento y mandarlos al robot. Todo ello para entrar en el área de seguridad, esquivar los obstáculos, ejecutar el *pick* sobre el *target*, salir del área de trabajo teniendo en cuenta los obstáculos y ejecutar el *drop* en la posición de inicio del robot.

El usuario tendrá la capacidad de modificar todos los parámetros del algoritmo, así como visualizar por pantalla los obstáculos y los movimientos de la misma forma que se visualiza en los modos: manual y de visualización.

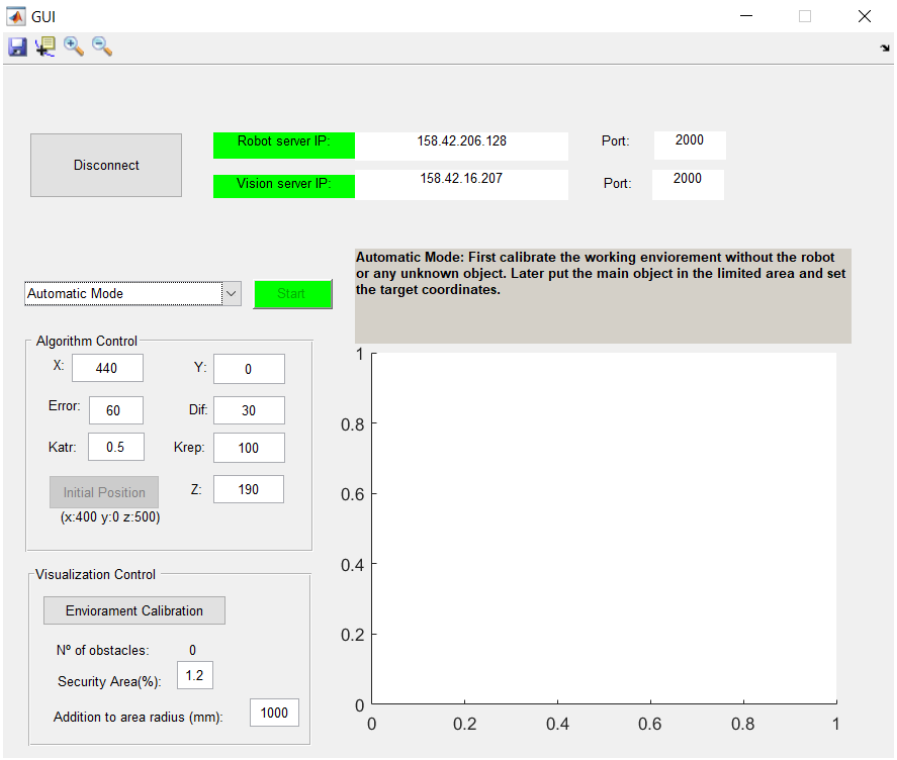

Ilustración 17. Pantalla de usuario. Modo automático

La secuencia de acciones es la siguiente:

- Calibración del entorno: Antes de poder iniciar el modo se deberá calibrar el entorno en el que se va a trabajar, es decir, aprender cómo es la mesa sin objetos y sus límites. Para ello se ejecutará la función *Environment Calibration*. De esta forma se determinarán los límites de seguridad para el robot y el área de visionado.
- Configuración: Los parámetros utilizados en este modo son: Error y Z para el robot; Dif, Katr, Krep, para la generación de la trayectoria e índice de seguridad y radio del área de efecto para la percepción y visionado de los objetos.

- Inicio. Búsqueda de objetivo y obstáculos: Al iniciar el programa se envía el comando: "$3" Búsqueda de obstáculos y objetivo, ver Capítulo III, Apartado 2.4. En base a esta información el robot calculará un punto por donde empezará moverse en el plano. En el caso de que no exista un punto que cumpla los requisitos, mostrará un mensaje de error y pedirá repetir la acción, ver Capitulo III, Apartado 5.1. El paso de la búsqueda de obstáculos se ejecuta una vez, al contrario de lo que ocurre en el modo de visualización.
- Movimiento al punto inicial: Se mandan tres órdenes al robot para que se mueva al punto inicial. Primero a un punto cercano a la esquina límite del área de trabajo, pero a una altura de seguridad determinada en el programa. En la siguiente orden el robot se dirigirá a la altura de trabajo, el valor del parámetro *Z*. Por último, se dirigirá al punto de inicio calculado anteriormente.
- Creación de la trayectoria: Tal como se explicará de forma detallada más adelante (ver Capítulo III, Apartado 5.2), el algoritmo no crea la trayectoria de una vez, sino que crea un vector para ejecutar el siguiente movimiento. Y es en base a este vector mediante el cual se calcula el próximo punto al que debe dirigirse el robot. Todo este proceso se ejecuta cuando el robot llega a la posición calculada en la ejecución anterior y mientras el robot no haya terminado el *pick* and *place*.
- Ejecución del movimiento: Cada punto se envía al robot para que ejecute el movimiento de forma lineal entre ellos con un error determinado en el parámetro Error. En cuando llegue a una posición cercana a la posición del *target*, se ejecutará el movimiento de pick para coger el objeto e invertirá las posiciones inicial y final dentro del algoritmo. Todo ello con la función de llevar el objeto fuera del área delimitada por las marcas, es decir, el robot creará una trayectoria contraria a la que ha hecho hasta ese momento.

En cuanto a la seguridad se refiere, el programa compara la posición de cada punto nuevo con la posición de los obstáculos, es decir, si el robot se va a mover dentro del área de un obstáculo se parará, dará una señal de error y se quedará esperando la acción del usuario, pero obligará a ejecutar el movimiento desde el principio. Lo mismo ocurre en el caso de salir del área de trabajo.

5. Algoritmo de Trayectoria y Cálculos Internos

5.1 Cálculo del punto inicial

En base a la posición de los obstáculos y al tamaño del área de trabajo, el programa calculará un punto en un lateral, a partir del cual sea seguro empezar a mover el robot. Para ello se ejecuta el módulo *StartPointCalculator(x0,y0,points,N,seg,Target,dy)* Siendo *x0, y0* la esquina del área desde la que se empieza a buscar, *points* la matriz de obstáculos, *N* el número de obstáculos, *seg* el índice de seguridad, *Target* la existencia del objetivo y *dy* el tamaño del lateral. La función del código es comprobar varios puntos separados por un espacio dy/10 para encontrar uno que no se encuentre cerca de un obstáculo. Si la distancia del punto de inicio al obstáculo es menor a Sx*seg*0.5, entonces el punto no será válido.

```
res=0;
i=1;
if(Target==1)
 while(1)
     %Start For
        while(i<=N)
            dist=sqrt((points(i,1)-x0-10)^2+(points(i,2)-y0)^2)
            if(dist<=(points(i,3)*seg*0.5))
                i=N;
                y0=y0+dy/10;
                res=1;
                if(y0>=dy)
                    res=2;
                end
                                                end
            i=i+1;
        end
    %End For
    %Response:
        if(res==0)
           xf=x0;
           yf=y0;
           break;
        elseif(res==1)
            res=0;
            i=1;
        elseif(res==2)
            xf=0;
            yf=0;
            uiwait(msgbox({'Start point could not be found','Please, check the limits of
the workspace and press start'},'Error','warn','modal'))
            res=1;
            break;
        end
 end
else
    res=2;
    xf=0;
    yf=0;
    uiwait(msgbox({'Target could not be found','Please, check the sistem an press start to
try again'},'Error','warn','modal'))
end
end
```

Ilustración 18. Fragmento de código. Búsqueda del punto inicial

5.2 Cálculo de la trayectoria

El algoritmo de cálculo de la trayectoria del robot es la base del proyecto. Este cálculo permite obtener el punto siguiente al que debe dirigirse el robot, en base a los parámetros definidos por el usuario y el punto actual.

5.2.1 Ecuaciones

El principio en el que se basa el algoritmo, es el campo atractivo y repulsivo en un plano de dos dimensiones. A partir de este campo, se obtendrá un vector de fuerza (1) que guie al robot hasta el punto final, pero esquivando los obstáculos en el trayecto.

Los componentes X e Y de fuerzas atractivas y repulsivas se calculan de forma independiente (2), ya que posteriormente, se utilizará la razón entre X e Y para obtener una dirección (3), que a su vez, se utilizará para calcular el siguiente punto que se enviará al robot.

(1) $$\vec{F} = \overrightarrow{F_D} - \overrightarrow{F_O} = -\Delta Ud - \Delta Uo$$

(2) $$\begin{cases} F_x = F_{dx} - F_{rx} \\ F_y = F_{dy} - F_{ry} \end{cases}$$

(3) $$\varphi = \tan^{-1}\frac{F_y}{F_x}$$

- Fuerza atractiva

Para el cálculo de la fuerza atractiva se utilizará como modelo el campo atractivo eléctrico (4) al cual afectan la posición final (5) y actual (6), es decir, se simulará que ambas son partículas con carga diferente, con lo que se podrá calcular un vector de fuerza. El módulo de esta fuerza ejercida será proporcional a la constante de atracción (8) y la distancia entre los puntos actual y final (7), es decir, tendrá un módulo menor cuanto más cerca se encuentre. Además, como lo que se desea es descomponerlo en dos vectores X e Y para el cálculo del valor de la dirección φ, se aplicará la misma ecuación, pero para el vector unitario de la distancia de cada uno de los ejes (9).

(4) $$U_p(q_d, q) = \frac{\varepsilon P}{2}|q_d - q|$$

(5) $$q_d = (x_d, y_d)$$

(6) $$q = (x, y)$$

(7) $$|d| = \sqrt{(x_d - x)^2 + (y_d - y)^2}$$

(8) $$K_{atr} = \frac{\varepsilon P}{2}$$

(9) $$\begin{cases} F_{dx} = K_{atr}\left(\frac{x_d - x}{|d|}\right) \\ F_{dy} = K_{atr}\left(\frac{y_d - y}{|d|}\right) \end{cases}$$

- Fuerza repulsiva

En contraposición a la fuerza atractiva, se encuentra la fuerza repulsiva, esta es, el sumatorio del efecto repulsivo de todos los obstáculos sobre el punto actual del robot (10). En este cálculo, los obstáculos pueden afectar de tres formas diferentes sobre la trayectoria y, este efecto, dependerá de la distancia que los separa del punto actual. Las maneras en las que se afectará a la trayectoria serán: la primera, si el robot se encuentra fuera del área de acción del obstáculo, la fuerza ejercida sobre el total será nula; la segunda, si el robot se encuentra en el área de acción, pero no en la de seguridad, se calculará una fuerza que se sumará al total, la cual dependerá de la distancia y los parámetros y, por último, si el robot se encuentra en el área de seguridad del obstáculo, se generará una fuerza de varios ordenes de magnitud mayor que el resto, mediante esta fuerza se condicionará la orientación del siguiente movimiento para que se aleje del obstáculo lo máximo posible.

(10) $$\begin{cases} F_{rx} = \sum_{i=0}^{N_{obs}} F_{xi} \\ F_{ry} = \sum_{i=0}^{N_{obs}} F_{yi} \end{cases}$$

(11) $$\overrightarrow{Fr} = \begin{cases} 0 \quad si\ |d| \geq p \\ k\left(\frac{1}{p} - \frac{1}{d_0}\right)\left(\frac{P_{obs} - P_{rob}}{|d|^2}\right) \quad si\ d_0 < |d| < p \\ -k_0\left(\frac{P_{obs} - P_{rob}}{|d|^2}\right) \quad si\ |d| \leq d_0 \end{cases}$$

5.2.2 Parámetros

Tabla 3. Parámetros de generación de trayectoria

Ámbito	Parámetro	Descripción
Repulsión	ox	Valor de la coordenada x del obstáculo. Se obtiene de la matriz de obstáculos
	oy	Valor de la coordenada y del obstáculo. Se obtiene de la matriz de obstáculos
	os	Valor del radio del obstáculo. Se obtiene a partir de la matriz de obstáculos
	d_0	Valor de la distancia de seguridad de un obstáculo. Es el producto del valor de su radio y el valor del índice de seguridad. Este último lo elige el usuario.
	p	Se trata del área de influencia del obstáculo, fuera de este valor no afecta a la trayectoria. Es la suma del radio del obstáculo y *p0,* el cual es la adición al área de influencia.
	K	Valor de la constante de repulsión del obstáculo. Es directamente proporcional al valor de su radio y al valor de *Krep*. Este último lo elige el usuario
	F_{ox}	Valor de la fuerza repulsiva de un obstáculo en el eje x
	F_{oy}	Valor de la fuerza repulsiva de un obstáculo en el eje y
	F_{rx}	Sumatorio de las fuerzas en el eje x de todos los obstáculos
	F_{ry}	Sumatorio de las fuerzas en el eje y de todos los obstáculos
	N_{obs}	Numero de obstáculos. Será el número de veces que se repetirá el cálculo de fuerzas de repulsión.

Ámbito	Parámetro	Descripción
Atracción	Katr	Parámetro de la constante atracción del punto final. Lo elige el usuario
	xd	Valor de la coordenada x del punto final
	yd	Valor de la coordenada y del punto final
	dmin	Valor de distancia mínimo respecto al punto final, si el robot se encontrara más cerca, se ejecutaría la acción de coger o dejar. Tiene un valor de 100 mm.
	F_{dx}	Fuerza de atracción en el eje X
	F_{dy}	Fuerza de atracción en el eje Y
Posición actual	x	Valor x actual
	y	Valor y actual
	F_x	Resultado de la fuerza en el eje x
	Fy	Resultado de la fuerza en el eje y
	Phi (ϕ)	Valor del ángulo de la fuerza

5.3 Cálculo de seguridad del movimiento

La seguridad del movimiento se divide en dos secciones: por un lado se encuentra la seguridad frente a los obstáculos y, por el otro, la seguridad frente a los bordes del área de trabajo. Si se cumple que el robot va a salirse del área o va a golpear un obstáculo, detectado anteriormente, se detendrá. La protección frente a estas dos condiciones se resuelve de la misma forma, esta es, se compara el siguiente punto de la trayectoria con las condiciones citadas, antes de haberse ejecutado el movimiento.

Por una parte, el proceso de parará si el punto se encuentra dentro del área crítica de un obstáculo, es decir, si el valor de la distancia al objeto es menor que su o_s. Esta comprobación se efectúa para todos los objetos.

Por la otra parte, el sistema se parará y generará un error si el punto tiene una de las coordenadas X e Y menor que cero o si el mismo punto es mayor que el valor X máximo o el valor Y máximo. Estos últimos se obtienen a partir del calibrado del entorno.

Existe una diferencia entre el valor de la distancia de seguridad y el valor de cercanía crítica a un obstáculo. Esta es el índice de seguridad (12), es decir, el robot solo se parará si la distancia al objeto es menor a o_s, pero no lo hará si es menor a la distancia de seguridad y mayor a la crítica. En este caso solo aplicaría la tercera condición del algoritmo de repulsión.

(12) $$d_0 = o_s \cdot i_{seg}$$

```
for i=1:size(objects,1)%Compare between next distances and objects' sizes
        d=sqrt((objects(i,1)-x)^2+(objects(i,2)-y)^2);
        if(d<=objects(i,3))
            colpoint=i
            col=1;
        end
    end
    if(((x<=0)||(x>=limxMax))||((y<=0)||(y>=limyMax)))%Out of limits
        col=3;
    end
```

Ilustración 19. Fragmento de código. Comparación de las áreas de peligro con el próximo punto de la trayectoria.

6. Gestión de la Información y Comunicaciones

6.1 Flujo de Información

El orden en el que se adquiere la información de las variables y se utilizan para calcular nueva información en el modo automático, es el siguiente:

1. El programa principal envía el comando *"$2"* al sistema de visión. El cual ejecutará la secuencia de obtención de los límites y devolverá los datos *XLim, YLim*, además se guardará la foto de referencia. Para ello, se utiliza el módulo *GetLims.m*
2. El programa principal envía el comando *"$3"* al sistema de visión. Este ejecutará la búsqueda de los obstáculos y devolverá: la matriz de obstáculos, en número de obstáculos, la existencia del objetivo y la posición del objetivo, si lo hubiera. Para ello se utiliza el módulo *GetObstacles.m*
3. A partir de la información de los obstáculos se calcula el punto inicial, a partir del cual empezará el robot y se guarda como posición inicial. Se gestiona mediante el módulo *StartPointCalculator.m*
4. A partir de este paso y hasta completar completamente el movimiento, el programa se ejecuta de forma repetitiva. Es en este paso donde se ejecuta el generador de trayectorias utilizando la información de la parametrización, la posición actual, la matriz de obstáculos y la posición final. A partir de esta información, Se calcula el siguiente punto. Este proceso se gestiona desde los módulos: *GenTray.m* para la fuerza atractiva, *Phione.m* para la dirección y *GetRepulsion.m* para la fuerza de repulsión.
5. El siguiente punto es comparado con los estándares de seguridad impuestos en el programa. Si es correcto, se manda al robot y, una vez se ha ejecutado y se ha recibido la confirmación, se repite el paso 4. Se gestiona mediante el módulo *GenTray.m*
6. Al llegar a la posición del objeto se ejecuta el *pick* y se cambia el punto final por el punto inicial, calculado al principio. A continuación se vuelve al paso 4.
7. Al volver al punto inicial, el robot ejecuta el *place* del objeto y detiene el modo automático

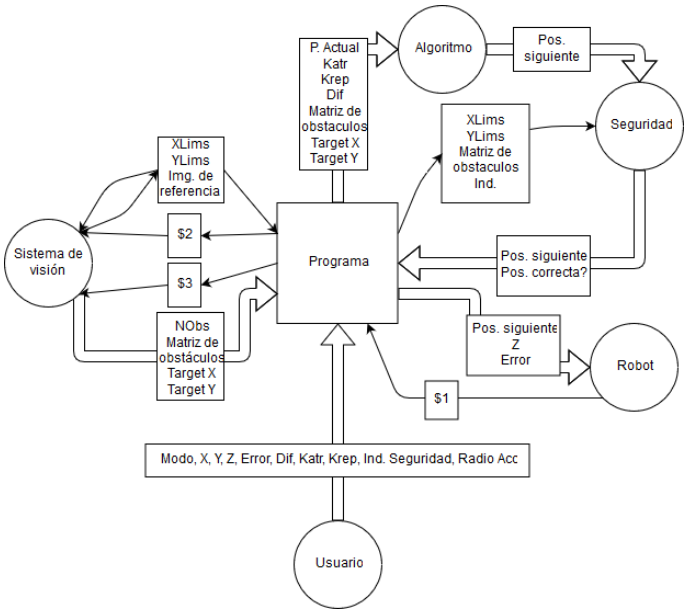

Ilustración 20. Diagrama del flujo de información en el proyecto.

6.2 Descompresión de la información de los obstáculos

En esta función, los datos observados desde el sistema de visión se descomprimen y se almacenan la posición y tamaño de los obstáculos en una matriz de tamaño n (13). Siendo n el número de obstáculos mas 1, X e Y la posición y S el tamaño de la diagonal mayor del objeto en milímetros.

$$\begin{bmatrix} x_1 & y_1 & s_1 \\ x_2 & y_2 & s_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & s_n \end{bmatrix} \tag{13}$$

$$[x_f \quad yf] \tag{14}$$

De la misma forma, se almacena la posición del *target* en la matriz de puntos finales (14). La cual, a su vez, se elimina de la matriz de obstáculos, ya que el programa de visión cuando detecta un obstáculo no distingue si es el *target* o no y, si no se filtraran a posteriori, se encontrarían registradas ambas posiciones en el mismo espacio. Para evitar que esto ocurra, primero se busca cuál de todos los obstáculos registrados es el más cercano al *target*, después se modifica la posición en la matriz, cambiándose a una en la que no se efectiva: [-100,-100] y, por último, se anula el tamaño del obstáculo en la matriz (parámetro o_s=1). De esta forma tampoco afectará al posterior uso en el algoritmo de trayectoria.

Capítulo IV: Experimentación

En este capítulo se explicará cómo se ha puesto en práctica todo lo programado en el proyecto, que problemas han surgido y cómo se han solucionado.

1. Solución Robot y el Movimiento Intermitente

El problema que ha surgido a la hora de implementar la generación de trayectorias en el robot ABB 140T, ha sido la falta de movimiento continuo, esto ocurre porque, para generar la trayectoria se van enviando pequeños tramos del total. Si los movimientos que se envían al robot son muy cortos o muy rápidos, el robot se frenará al terminar cada uno antes de empezar el siguiente. Este es un efecto que no es deseable.

Para solucionar este problema se ha recurrido a las variables del robot que afectan a la ejecución de la trayectoria: *velocidad*, *error* y *dif*. Configurando estos tres parámetros correctamente se modifica la velocidad del robot, el error de la trayectoria (valor de *zone* del robot) y el tamaño del vector de movimiento, respectivamente. Con ello se conseguirá que el movimiento sea un poco más lento y los tramos calculados sean un poco más grandes. Además el robot trabaja mejor con un error respecto a la trayectoria.

Los valores que se han obtenido experimentalmente han sido: una velocidad de 30, un error de 50 y un vector de movimiento de 3 cm.

2. Solución Parametrización del Generación de Trayectorias

El otro problema que se ha encontrado durante la experimentación ha sido la parametrización del generador de trayectorias. Este problema ha surgido porque, si el generador no se encuentra configurado correctamente, la trayectoria creada tendrá cambios bruscos en su dirección debido a la generación de fuerzas demasiado intensas.

Para la solución de este problema, se han adoptado dos puntos:

El primero, ha sido trabajar con un radio de acción de varias magnitudes mayor a las distancias utilizadas, de esta forma todos los obstáculos influirán sobre el movimiento.

El segundo, ha sido equilibrar las fuerzas atractivas y repulsivas para conseguir el tipo de movimiento deseado. Esto se ha conseguido modificando los parámetros K_{atr} y K_{rep} mediante un código externo al programa principal. Se trata de una código con el mismo generador de trayectorias al que se le puede diseñar un escenario como se desee, es decir, se pueden definir la matriz de obstáculos, el punto final, los parámetros y el tamaño del área de trabajo. Este código está incluido dentro del proyecto, como una herramienta útil para el trabajo de prueba y error.

3. Resultados

El resultado final del proyecto ha resultado satisfactorio, ya que, se ha conseguido ejecutar un *"pick and place"* de forma correcta, además, se han probado los modos de visualización y manual. En este apartado se muestran los resultados de todas las pruebas efectuadas.

La primera prueba, ha consistido en un ejemplo sencillo del modo automático. Se ha posicionado un obstáculo entre el punto de inicio y el objetivo. El resultado se puede visualizar en:

https://www.youtube.com/watch?v=TA8Ob_WqhoA

La segunda prueba, también del modo automático, ha consistido en posicionar tres objetos medianos entre el punto de entrada y el objetivo, de manera que la trayectoria deba pasar entre ellos (Ilustración 21). En el caso de tapar completamente cualquier trayectoria, el robot se para y pide volver a empezar con el área completamente libre.

El resultado se puede visualizar en:

https://www.youtube.com/watch?v=7BMi2fobzAM

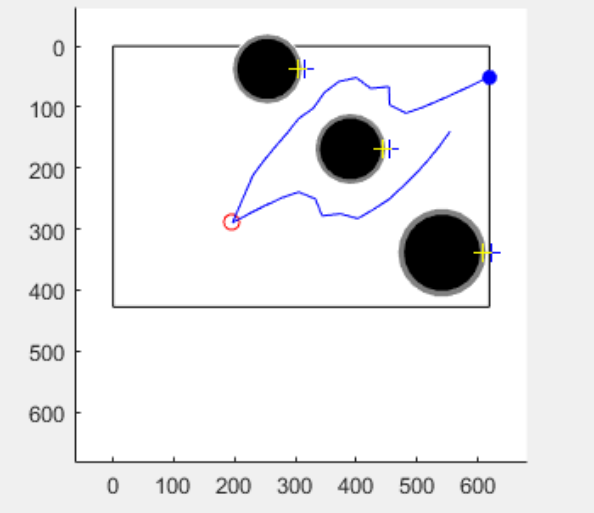

Ilustración 21. Trayectoria generada de forma automática para esquivar tres obstáculos mientras coge un objeto y lo lleva al punto de inicio.

Por último, las pruebas de los modos de visualización y manual han sido efectivas. Han sido dos herramientas útiles para la configuración de los parámetros para el correcto funcionamiento del modo automático.

Capítulo V: Conclusiones

A partir de los resultados obtenidos se puede decir que la aplicación ha sido un éxito. Tanto la generación de la trayectoria, como la ejecución de los movimientos son satisfactorias.

En esta aplicación se ha volcado todo el conocimiento adquirido durante la carrera de Ingeniería Electrónica y Automática Industrial. Creando un método para realizar movimientos en un entorno automatizado y dinámico con una mayor seguridad que antes.

Para concluir, queda decir que el programa queda abierto a futuras mejoras. Ya que, cada proceso es diferente y necesita de un programa con capacidades distintas. Pero, para ello, se ha dado la facilidad de mejorar y cambiar las diferentes funciones mediante la programación modular, por ejemplo, si se necesita un generador de trayectorias diferente, es suficiente con cambiar la respectiva función. No será necesario modificar el resto.

1. Márgenes de Mejora

1.1 Reacción rápida en tiempo real

Dada la lentitud en la gestión de imágenes con la que se ejecutaba el sistema de visión, no ha sido posible implantar uno de los puntos iniciales del proyecto: la reacción en tiempo real. El cual fue descartado. En este caso, se ha podido solucionar el problema de la ejecución intermitente del movimiento en el robot cuando se dan tramos pequeños, pero el tema de los tiempos cortos en el sistema de visión, no. Es por ello que será necesario implantar un sistema diferente o ejecutar un algoritmo diferente para disminuir este retardo.

Otro de los problemas que surge de esta idea es que el robot se encuentra en el área de trabajo mientras se ejecuta la adquisición de los datos. Sería necesaria un algoritmo que lo elimine de la imagen y algún modo de guardar los obstáculos que está tapando a la cámara en ese momento.

1.2 Programa integrado en dispositivo

El programa principal se encuentra escrito en Matlab. Aunque es una herramienta de trabajo útil, una aplicación profesional para la industria debería formar parte de un dispositivo independiente a un PC. Además, la integración del sistema de visión al mismo dispositivo ayudaría a reducir retardos.

1.3 Algoritmo de generación de trayectorias diferente

El método que he elegido para la generación de trayectorias ha sido sencillo y útil. El único inconveniente que ha surgido, es que es necesario modificar los parámetros en el momento que cambia un poco el entorno. Hay otras técnicas, como Voronoi, que permiten la generación de trayectorias de una forma más robusta. Eso sí, a cambio de una necesidad de mayor capacidad de computación. El programa permite integrar un generador de trayectorias diferente, si es necesario.

Capítulo VI: Presupuesto

1. Tabla del Presupuesto

Materiales y Licencias			
Descripción	**Unidades**	**Coste**	**Total**
Robot ABB IRB 140	1	15000	15000
Cámara y ordenador. Sistema de visión	1	600	600
PC	1	800	800
Licencia Matlab Student	1	69	69
Licencia Sherlock	1	60	60
Trabajo			
Descripción	**Horas**	**€/h**	**Total**
Montaje	20	15	300
Diseño y Programación	120	15	1800
Documentación	100	15	1500
		Total Bruto	20.129,00 €
		IVA (21%)	4.227,09 €

TOTAL PRESUPUESTO	24.356,09 €

Bibliografía

[1] Sherlock User Reference Manual

[2] https://www.dspace.espol.edu.ec/bitstream/123456789/10740/11/MATLAB_GUIDE.pdf

[3] http://developercenter.robotstudio.com/BlobProxy/manuals

[4] http://www.inocua.org/site/Archivos/seguridad/LSI_Cap02.pdf

[5] https://new.abb.com/products/robotics/es/robots-industriales/irb-140

[6] https://es.mathworks.com/help/matlab/

ANEXO

Guía de usuario

En esta pequeña guía se explicará cómo utilizar la aplicación con sus tres modalidades.

Conexión/Desconexión

Lo primero que se deberá hacer una vez se ha iniciado el programa será conectarse a los sistemas periféricos. Para ello iniciamos los programas del robot y del sistema de visión y se configuran las IP y puertos que se van a utilizar.

Se debe tener en cuenta que el robot se encuentre en modo automático y listo para recibir la información. En el FlexPendant debe de aparecer un texto que lo indique. De la misma forma el sistema de visión debe de estar correctamente conectado a la red y se deberá reiniciar el programa Sherlock en el caso de que no fuera así.

Una vez se ha tenido todo esto en cuenta apretaremos el botón de conexión de la pantalla de usuario. En ese instante, si todo ha ido bien, el robot comenzará a moverse a la posición inicial y los botones de la pantalla se desbloquearán.

En el caso de no haberse conectado correctamente se desconectará automáticamente de ambos periféricos y volverá a la situación inicial. Además, mostrará un mensaje de error.

El botón ahora ha cambiado a *desconexión*. Si se selecciona se cortarán las comunicaciones con los periféricos y se volverá a la posición de inicio.

Selección de modo

Existen tres modos: Manual, Visualización y Automático, cada uno de ellos tiene una pequeña descripción en la misma pantalla y ocultan ciertos botones cuando están activos. Para cambiar entre ellos se utiliza el menú pop-up abajo del botón de conexión

Calibración del entorno

Este es un proceso que se debe efectuar cuando se quiera visualizar algo desde el programa. Sirve para establecer los límites de la mesa de trabajo, así como para tomar la referencia de la mesa limpia. El proceso es el siguiente:

1. Selecciona el botón *Enviorment Calibration.* El robot irá a la posición de calibración, es decir, fuera del rango de visión de la cámara. Cuando haya terminado mostrará un mensaje de confirmación.
2. Limpia la mesa de trabajo y coloca las marcas circulares donde se desee que se encuentren las esquinas del área de trabajo. Pulsa OK en el mensaje de confirmación
3. Los límites de la gráfica han cambiado y se muestra un mensaje.

Modo Manual

Este modo permite controlar el punto final al que se dirigirá el robot. Para ejecutarlo se lleva al robot a la posición inicial apretando el botón *initial position*. Una vez allí se mostrará un mensaje de confirmación. A partir de ese momento se permitirá pulsar el botón *start*.

El movimiento se determina mediante los parámetros X, Y, Z. Estos determinan el punto final. Se debe tener en consideración que este punto está referenciado al sistema de coordenadas del robot y, además, su valor no puede estar fuera de los límites físicos del robot.

ADVERTENCIA: En este modo, el robot no tiene en cuenta por donde se mueve. Depende completamente del usuario

Modo Visualización

Este modo necesita de la calibración del entorno para funcionar. Una vez ejecutado se activará el botón *start*.

Una vez ejecutado el programa no parará de recibir información hasta que se haya seleccionado el botón *stop*. Mientras tanto toda la información recibida de la cámara aparecerá en la gráfica de la aplicación de forma continua. Un círculo negro para el área de seguridad de los obstáculos y un círculo rojo para el objetivo.

Este modo también permite la modificación del radio de acción y del índice de seguridad. Estos modifican como se observan los obstáculos

Modo Automático

Este modo necesita de la calibración del entorno para funcionar. Una vez ejecutado se activará el botón *start*.

Este modo se divide en dos fases. La primera, es la captación de información y elección del punto inicial y la segunda, la generación de la trayectoria y movimiento del robot.

En el caso de que durante la fase de movimiento encontrara algún problema de seguridad, saldría de esta fase y volvería al principio de la fase de obtención de información.

La visualización de la información es la misma que en el modo de visualización, pero en este modo se visualiza también la trayectoria seguida por el robot.

En cualquier momento se puede detener la ejecución del movimiento apretando el botón *stop.*