Trabajo Fin de Grado

Protocolo de comunicación trabajador-robot mediante imágenes

José Angel Castilla Berduque

Grado de Ingeniería Electrónica y Automática

Tutor: Gerard Masferrer Caralt

Vic, Enero de 2015

Índice

1. RESUMEN 5
2. INTRODUCCIÓN 8
2.1. Antecedentes 9
2.2. Objetivos 9
2.3. Estructura de la memoria 9
2.4. Metodología de trabajo 10
3. HERRAMIENTAS UTILIZADAS 11
3.1. Recursos utilizados 12
3.2. Matlab 12
3.2.1. Image Processing toolbox 13
3.3. Connecting Sockets 14
3.4. Robot Studio 15
3.5. Robot ABB IRB120 16
4. ANALISIS Y TRATAMIENTO DE IMAGEN 17
4.1. Procesado digital de imágenes 18
4.2. Diseño y definición de imágenes 18
4.2.1. Zona de trabajo 19
4.2.2. Selección de herramienta 21
4.2.3. Selección de modo 23
4.2.4. Puntos discontinuos 25
4.2.5. Trazo continuo 27
4.2.6. Comunicación con el robot 29
4.2.6.1. Protocolo TCP/IP 29
4.2.6.2. Script comunicación con el Robot 31
5. PROGRAMACIÓN DEL ROBOT 34
5.1. Lenguaje de programación RAPID 35
5.2. Programa del robot 36
5.2.1. Módulo CalibData 37
5.2.2. Módulo Main 37
6. RESULTADOS Y CONCLUSIONES 43
6.1. Resultados y conclusiones 44
6.2. Avances futuros 45
7. BIBLIOGRAFIA 46

7.1. Libros y artículos 47
7.2. Direcciones de Internet 47
8. ANEXO 48
8.1. Código de análisis de imagen 49
8.2. Código programación robot 53

Indice de Figuras

Figura 1 Zona de trabajo Matlab 12
Figura 2 Image Processing toolbox 13
Figura 3 Connecting sockets 14
Figura 4 Robot Studio 15
Figura 5 Robot ABB IRB120 16
Figura 6 Rango de movimiento del robot ABB IRB120 16
Figura 7 Zona de trabajo 19
Figura 8 Zona definición herramienta 21
Figura 9 Zona definición modo 23
Figura 10 Zona definición puntos mediante cruces 25
Figura 11 Zona de definición del trazo 27
Figura 12 Arquitectura cliente-servidor 30
Figura 13 Flujograma cliente servidor 30
Figura 14 Flujograma comunicación Matlab 31
Figura 15 Flujograma programa Robot Studio 36

1. RESUMEN

Universitat de Vic
Escola Politècnica Superior

Resumen Trabajo Fin de Grado
Grado de Ingeniería Electrónica y Automática

Título: Protocolo de comunicación trabajador-robot mediante imágenes
Palabras clave: Trabajador, imagen, Matlab, RobotStudio, robot
Autor: José Angel Castilla Berduque
Tutor: Gerard Masferrer Caralt

La idea del proyecto viene del concepto de “fábricas del futuro”, donde las barreras entre robots y humanos se rompen para que la colaboración entre ambos sea como en un equipo.

Para la realización de este proyecto se ha utilizado el brazo robótico IRB120 de la marca ABB de 6 Grados de libertad, Matlab y el software Robot Studio.

El Objetivo principal de este proyecto es establecer el protocolo de comunicación trabajador-robot mediante imágenes. El trabajador debería poder controlar el robot mediante dibujos realizados en la mesa de trabajo.

En el desarrollo de la comunicación trabajador-robot cabe distinguir tres partes:

- *El análisis y tratamiento de imágenes para el cual se ha utilizado el software Matlab.*
- *Transmisión de los datos desde Matlab al robot.*
- *Programación de las acciones a realizar por el robot mediante el software “Robot Studio”.*

Con el protocolo de comunicación desarrollado y las imágenes realizadas por el trabajador el robot es capaz de detectar lo siguiente:

- *la herramienta que debe utilizar (rotulador, boli o ventosa)*
- *si lo que tiene que dibujar en la mesa de trabajo son puntos o trazo continuo.*
- *la localización de los puntos o del trazo continuo en la mesa de trabajo.*

Se ha alcanzado el objetivo propuesto con éxito, el protocolo de comunicación trabajador-robot mediante imágenes ha sido establecido. Mediante el análisis y tratamiento de imágenes se puede conseguir la información necesaria para que el robot pueda ejecutar las acciones requeridas por el trabajador.

Universitat de Vic
Escola Politècnica Superior

Final Project Summary
Degree in electronics engineering and automatic

Title: Communication protocol worker-robot through images.
Keywords: Worker, image, Matlab, RobotStudio, robot
Author: José Angel Castilla Berduque
Tutor: Gerard Masferrer Caralt

The project idea is the concept of "factories of the future" where the barriers between robots and humans break so that the collaboration between them is like a team.

For this project we used the robotic arm IRB120 brand ABB with 6 Degrees, Robot Studio and Matlab software.

The main objective of this project is to establish the communication protocol worker-robot through images. The worker should be able to control the robot by drawings made on the workbench.

For the development of the communication protocol worker-robot three parts can be distinguished:

- *The analysis and processing of images using “Matlab” software.*
- *Data transmission from Matlab to the robot.*
- *Programming actions to be performed by the robot using the "Robot Studio" software.*

With the communication protocol developed and the images drawn by the worker the robot is able to detect the following:

- *the tool to use (pen, marker or vacuum pad)*
- *if the points to draw are continuos or discontinuos*
- *the location of the points to draw.*

The target has been achieved successfully, the communcation protocol worker-robot through images has been stablished. Through analysis and image processing can get the necessary information so that the robot can perform the actions required by the worker.

2. INTRODUCCIÓN

2.1. Antecedentes

La industria está cambiando vertiginosamente y principalmente en la robótica. Las empresas buscan flexibilidad y productividad para poder adaptarse rápidamente a las necesidades de mercado.

En el campo de la robótica uno de los problemas es la complejidad de la programación del robot y la poca flexibilidad que presenta.

Los robots son muy útiles para tareas repetitivas y pesadas pero no aportan la flexibilidad y autonomía necesarias para adaptarse al cambio.

En la búsqueda de esa flexibilidad requerida se encuentra necesario que exista una completa interacción de los robots con los seres humanos en las cadenas productivas.

Las empresas demandan ligereza, costes más bajos y flexibilidad, así como un uso sencillo de la robótica y de su programación.

2.2. Objetivos

El Objetivo principal de este proyecto es establecer el protocolo de comunicación trabajador-robot mediante imágenes. El trabajador debería poder controlar el robot mediante dibujos realizados en la mesa de trabajo.

2.3. Estructura de la memoria

Para la realización de la memoria se ha dividido el trabajo en los capítulos que se detallan a continuación:

Capítulo 2: Herramientas utilizadas:

En este apartado se describen el software utilizado en el análisis y tratamiento de la imagen y programación del robot.

Capítulo 3: Análisis y tratamiento de imagen

En este capítulo se describe los análisis y tratamientos de imagen realizados para obtener la información requerida por el robot.

Capítulo 4: Programación del robot

En este capítulo de describe todas la programación realizada en Rapid para que el robot ejecute las acciones requeridas.

Capítulo 5: Resultados y Conclusiones

Se exponen las conclusiones obtenidas y siguientes pasos.

Capítulo 6: Bibliografía

Se detallan todas las fuentes bibliográficas utilizadas.

Capítulo 7: Anexo

Se anexan el código de los programas realizados.

2.4. Metodología de trabajo

1. Profundizar en el manejo de software de tratamiento y procesamiento de imágenes: Primeramente se analizaron los diferentes softwares existentes en el mercado para el análisis y tratamiento de imagen. Luego se profundizó en el manejo de Matlab en particular en el Image analysis toolbox.

2. Definición de imágenes a utilizar: se definen diferentes imágenes para referencias la herramienta a utilizar y si son puntos discontinuos o trazo continuo.

3. Desarrollo del tratamiento y procesado de las imágenes: desarrollo de los scripts y funciones necesarias para el procesado de las imágenes.

4. Programación del robot mediante Robot Studio: desarrollo mediante la programación en RAPID de las acciones a ejecutar por el robot.

3. HERRAMIENTAS UTILIZADAS

3.1. Recursos utilizados

En la realización del trabajo se han utilizado los siguientes programas: Matlab concretamente el toolbox image Analysis toolbox, Connecting Sockets y Robot Studio de ABB.

3.2. Matlab

MATLAB (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

Es un software muy usado en universidades y centros de investigación y desarrollo.

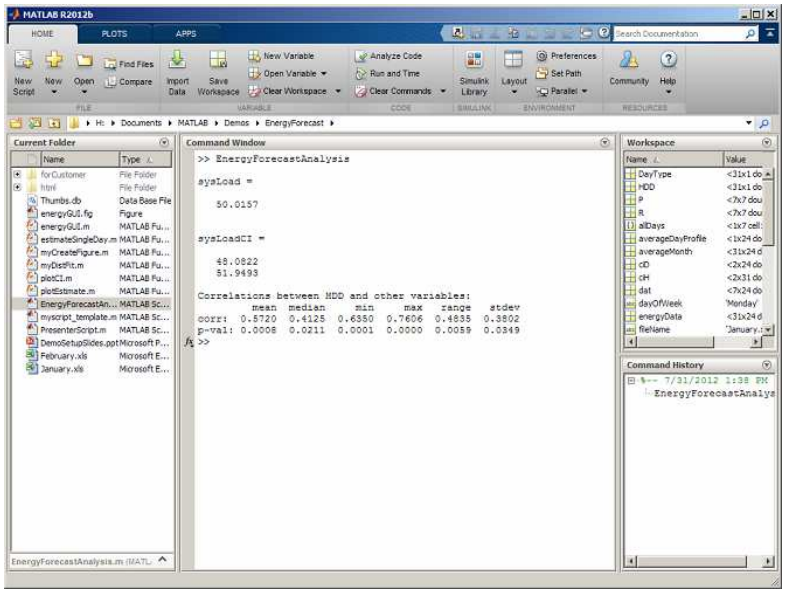


Figura 1 Zona de trabajo Matlab

3.2.1. Image Processing toolbox

Image Processing Toolbox proporciona un conjunto completo de algoritmos, funciones y aplicaciones de referencia estándar para el procesamiento, el análisis y la visualización de imágenes, así como para el desarrollo de algoritmos. Puede llevar a cabo análisis de imágenes, segmentación de imágenes, mejora de imágenes reducción de ruido, transformaciones geométricas y registro de imágenes.

Image Processing Toolbox soporta un conjunto diverso de tipos de imágenes, tales como las de alto rango dinámico o las de resolución de giga píxeles entre otras. Las características y aplicaciones de visualización permiten explorar imágenes y vídeos, examinar una región de píxeles, ajustar el color y el contraste, crear contornos o histogramas y manipular regiones de interés (ROI). Esta toolbox soporta flujos de trabajo para procesar y mostrar imágenes de gran tamaño, así como para navegar por ellas.

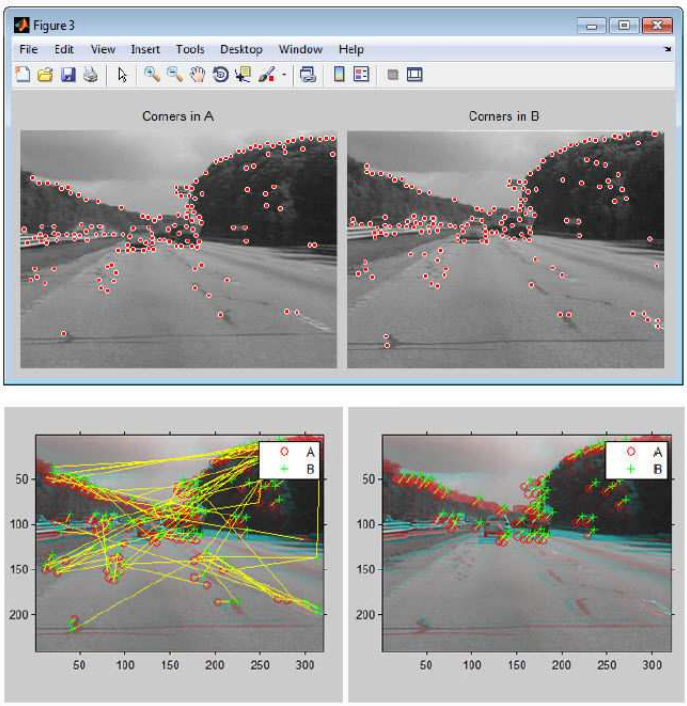


Figura 2 Image Processing toolbox

3.3. Connecting Sockets

Connecting Sockets (CSockets) es un programa que utiliza el winsock para hacer conexiones a un servidor y puerto especificado por el usuario e intercambiar comandos de texto. CSockets también pueden crear un socket de escucha en un puerto especificado por el usuario y acepta clientes. Además CSockets pueden comprobar un servidor y ver qué puertos están abiertos.

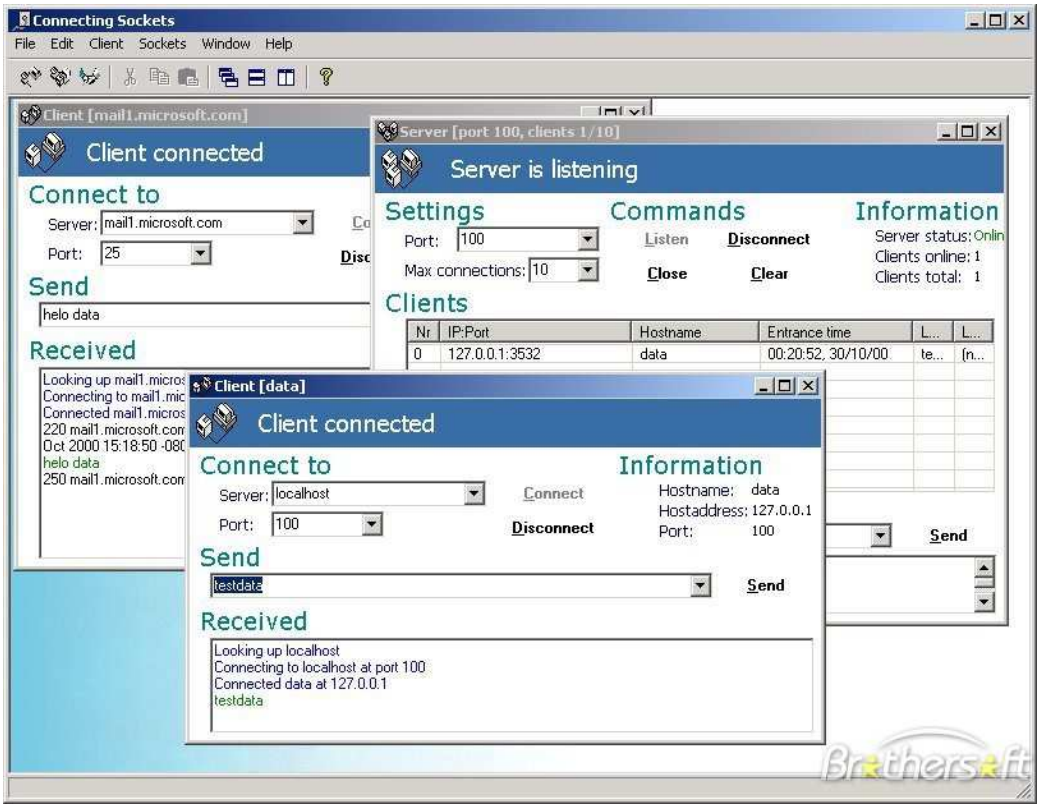


Figura 3 Connecting sockets

3.4. Robot Studio

El software de programación offline RobotStudio permite programar los robots en un PC sin necesidad de parar la producción. También se pueden preparar los programas de los robots anticipadamente, lo que implica un aumento de la productividad.

RobotStudio aporta herramientas que aumentan la rentabilidad del sistema de robots, pues permite realizar tareas tales como formación, programación y optimización de programas sin alterar la producción.

Se basa en el controlador virtual de ABB, una copia exacta del software real que emplean los robots en la producción. Ello permite ejecutar simulaciones muy realistas, utilizando programas de robots reales y archivos de configuración idénticos a los que se emplean en la fábrica.

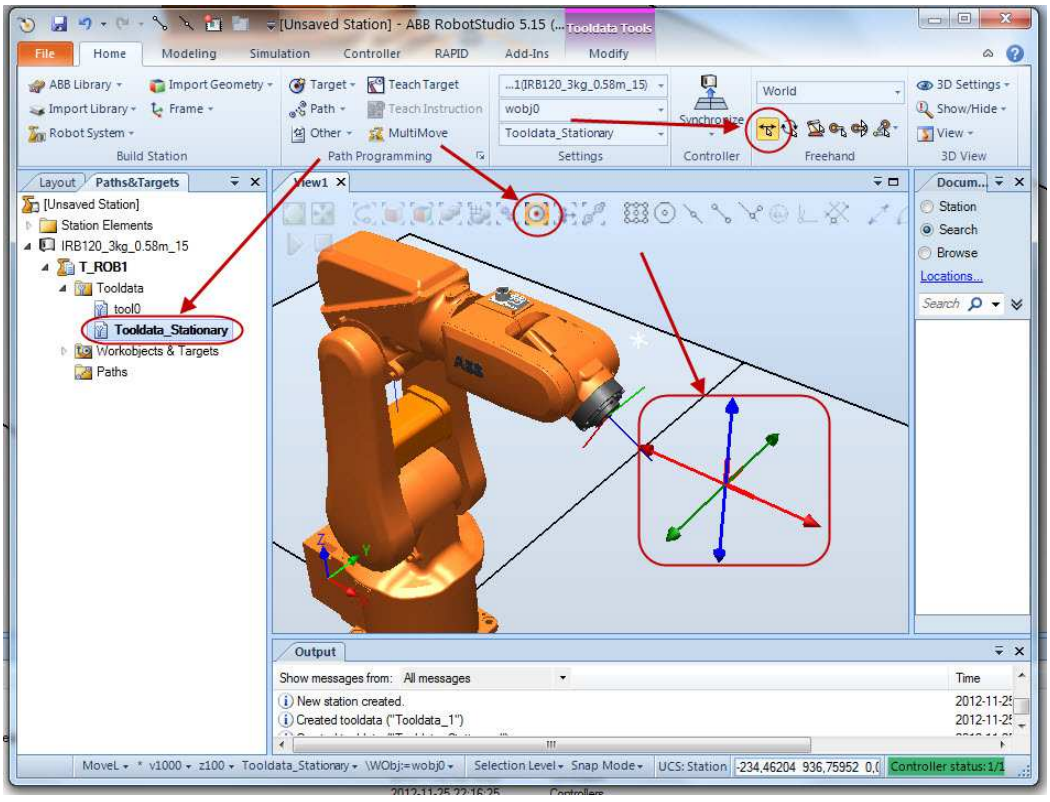


Figura 4 Robot Studio

3.5. Robot ABB IRB120

El robot ABB IRB120 es el más pequeño y se puede utilizar para muchas aplicaciones pesa solamente 25 Kg. y puede manipular hasta 3 Kg. (4 Kg. para la muñeca en posición vertical), 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.

Figura 5 Robot ABB IRB120

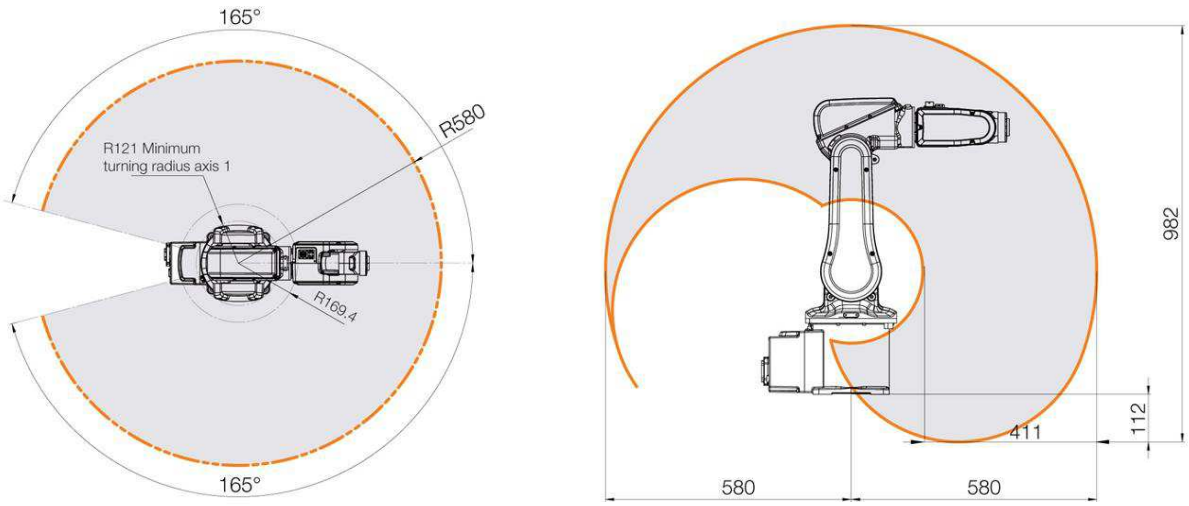


Figura 6 Rango de movimiento del robot ABB IRB120

4. ANALISIS Y TRATAMIENTO DE IMAGEN

4.1. Procesado digital de imágenes

Al conjunto de técnicas y procesos para descubrir o hacer resaltar información contenida en una imagen usando como herramienta principal una computadora se le conoce como procesamiento digital de imágenes (PDI). El PDI es un área de investigación muy específica en computación y está muy relacionada con el procesamiento digital de señales. Esta relación estriba en el hecho de que en esencia el PDI es una forma muy especial del procesamiento digital de señales en dos o tres dimensiones.

El interés en el estudio del PDI se basa en dos áreas de aplicación primordiales:

a) El mejoramiento de la calidad de la información contenida en una imagen con el fin de que esta información pueda ser interpretada por los humanos

b) El procesamiento de los datos contenidos en un escenario a través de una máquina de percepción autónoma.

4.2. Diseño y definición de imágenes

El primer paso a realizar es diseñar y definir que acciones queremos que el robot ejecute mediante el análisis y procesamiento de diferentes imágenes. Las funciones a realizar por el robot son:

- *Identificar la zona de trabajo*
- *Seleccionar la herramienta a utilizar*
- *Identificar si son puntos discontinuos o trazo continuo*
- *Realizar la trayectoria definida ya sean puntos sueltos o trazos continuos.*

Segundo paso es definir las imágenes que indicarán al robot las acciones a realizar.

A continuación se van a detallar las imágenes definidas y el procesamiento realizado para conseguir la información necesaria para el robot.

4.2.1. Zona de trabajo

La zona de trabajo donde el trabajador dibujará queda definida por tres o cuatros cuadrados negros de 20 Mm. en las esquinas de la mesa de trabajo.

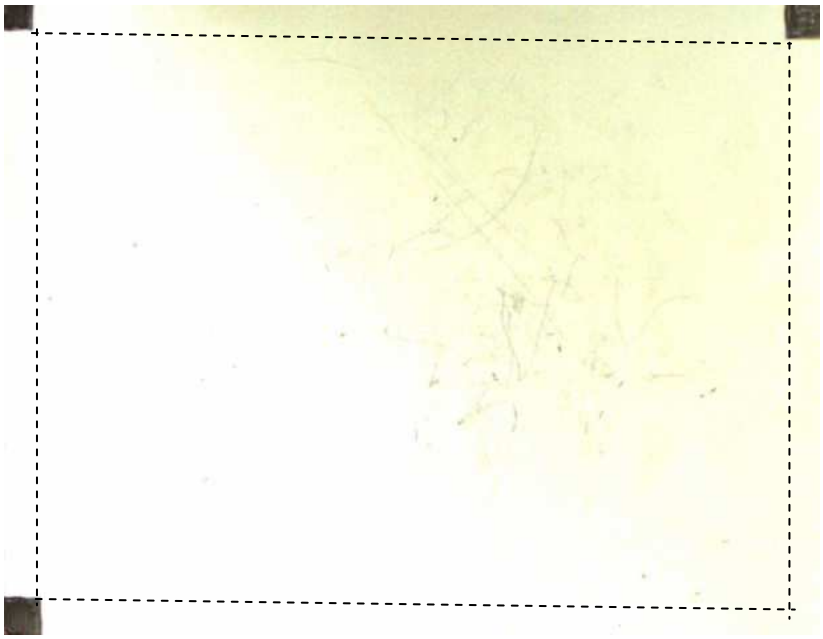
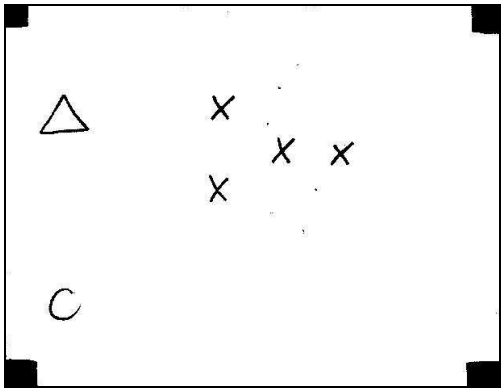


Figura 7 Zona de trabajo

Se ha desarrollado una función “DetectZonaVic” donde detectamos los rectángulos negros y luego se recorta el área de trabajo (líneas discontinuas de la figura 5) que es donde el trabajador realizará las imágenes.

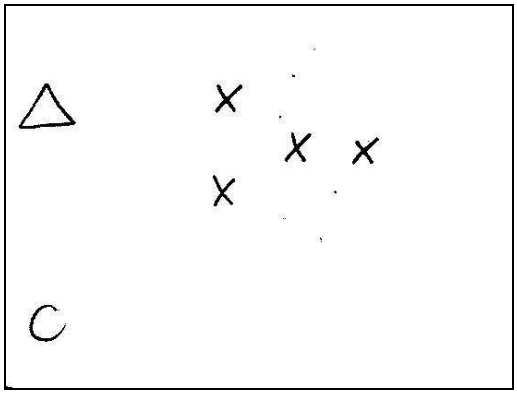
Se detalla a continuación el flujo de programa que realiza la función desarrollada para la detección de la zona de trabajo:

Paso 0. Captura de la imagen.
Se puede apreciar que hay cuatro cuadrados neg uno en cada esquina.



Paso 1. Detección de los cuadrados negros en las esquinas.
Etapas:
-conversión a escala de grises
CGris=rgb2gray(C);
-conversión a imgen binaria
Cbw=im2bw(CGris,level)
-dilatatación de la imagen
Dilatacion=imdilate(Climpio,SE1)
-detección de bordes
contornocanny=edge(Dilatacion,'canny');
-rellenado de los bordes detectados
rellenado=imfill(Dilatacion2,'holes');
-etiquetado de objetos y calculo de centros
[L,Ne]=bwlabel(rellenado2);
prop=regionprops(L,'Centroid');

Paso 2. Recorte de la zona de trabajo
Etapas:
-recorte de la imagen delimitada por los bordes de cuadrados negros
D=imcrop(Climpio,[x y width height]);



4.2.2. Selección de herramienta

Se han definido tres figuras geométricas para indicar al robot que herramienta debe utilizar. Las figuras seleccionadas son las siguientes:

- *Círculo: el robot deberá utilizar la herramienta ventosa*
- *Cuadrado: el robot deberá utilizar la herramienta boli.*
- *Triángulo: el robot deberá utilizar la herramienta rotulador.*

Mediante el procesamiento de imágenes se ha desarrollado una función para distinguir que figura geométrica se ha dibujado. Se ha definido también la zona donde hay que dibujar esta figura para que el procesamiento sea correcto. La zona definida es en el primer tercio de la zona de trabajo en sentido X y los 2/3 primeros en el sentido Y.

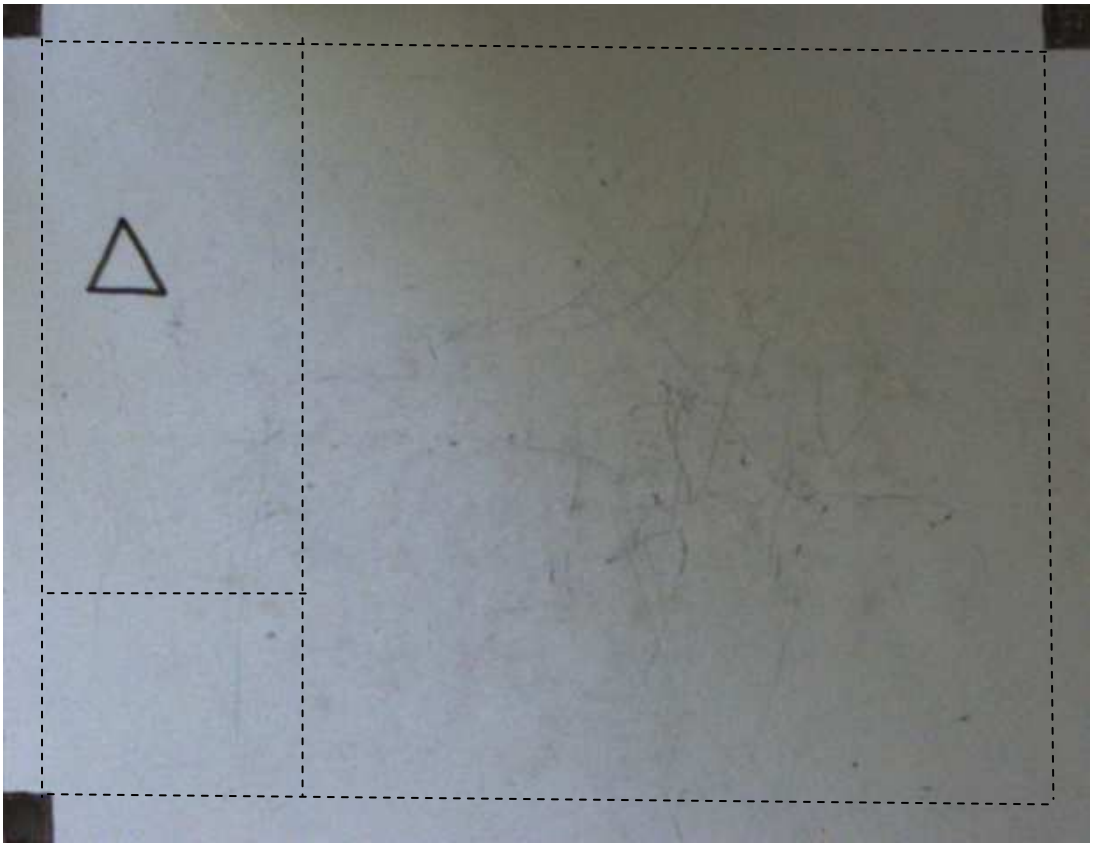
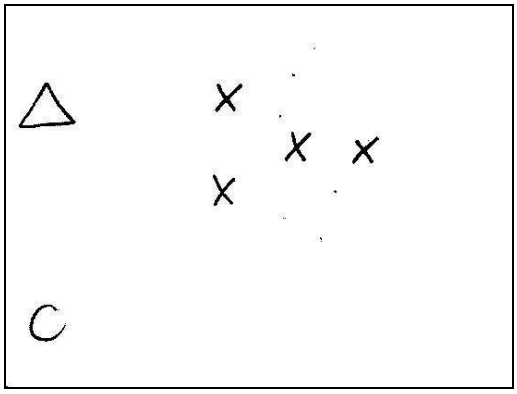


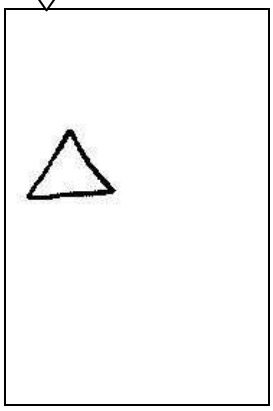
Figura 8 Zona definición herramienta

Se detalla a continuación el flujo de programa que realiza la función "Detectformasvic" desarrollada para la detección de la zona de trabajo:

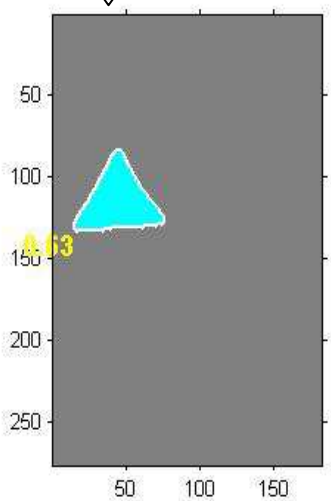
Paso 0. Imagen con zona de trabajo definida



Paso 1. Recorte de la zona definida para dibujar imagen del tipo de herramienta a utilizar.
Etapas:
-recorte de la zona definida
G=imcrop(D,[0 0 F(2)/3 F(1)*2/3]);



Paso 2. Detección de la figura geométrica diferenciand entre círculo, triángulo y cuadrado
Etapas:
-detección de bordes
contornocanny=edge(G,'canny');
-rellenado de los bordes detectados
rellenado=imfill(contornocanny,'holes');
-detección frontera del dibulo y visualización
[B,L] = bwboundaries(rellenado,'noholes');
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
-cálculo del área y perímetro de la figura
stats = regionprops(L,'Area','Perimeter');
-cálculo parámetro de redondez
metric = 4*pi*area/perimetro^2;
-clasificación entre círculo, cuadrado y triángulo en base a parámetro de redondez



4.2.3. Selección de modo

Con la selección de modo se indica mediante la función “Detectsubvic” al robot si lo que se va a dibujar son puntos o un trazo continuo. Para ello se ha definido dos imágenes:

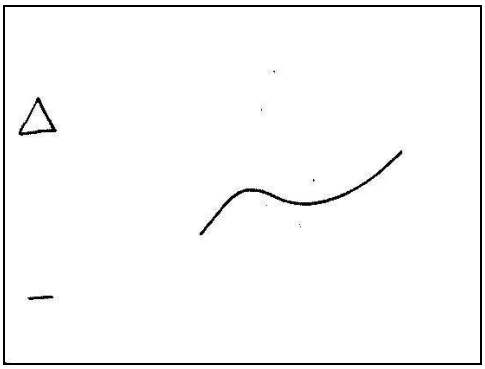
- *Círculo: significa que lo que se va a dibujar son puntos sueltos.*
- *Línea: indica que lo que se va a dibujar es un trazo continuo.*

La zona definida para indicar mediante imagen si son puntos sueltos o un trazo discontinuo es en el primer tercio en el sentido X y en el último tercio en el sentido Y de la zona de trabajo.

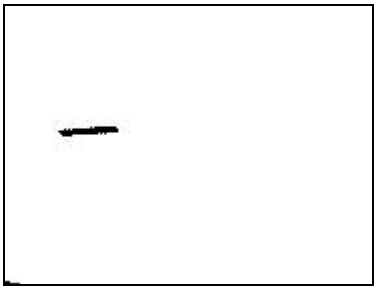
Figura 9 Zona definición modo

El flujo de programa de la función “Detectsubvic” queda de la siguiente manera:

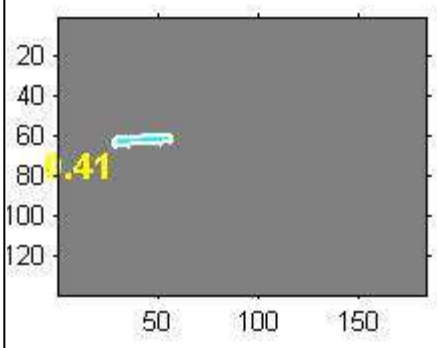
Paso 0. Imagen con zona de trabajo definida



Paso 1. Recorte de la zona definida para dibujar la imagen para indicar el modo.
Etapas:
-recorte de la zona definida
G=imcrop(D,[0 F(1)*2/3 F(2)/3 F(1)/3]);



Paso 2. Detección de la figura geométrica diferencian entre círculo y línea
Etapas:
-detección de bordes
contornocanny=edge(G,'canny');
-rellenado de los bordes detectados
rellenado=imfill(contornocanny,'holes');
-detección frontera del dibulo y visualización
[B,L] = bwboundaries(rellenado,'noholes');
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
-cálculo del área y perímetro de la figura
stats = regionprops(L,'Area','Perimeter');
-cálculo parámetro de redondez
metric = 4*pi*area/perimetro^2;
-clasificación de continuo o discontinuo en base al paráme de redondez.



4.2.4. Puntos discontinuos

La localización de los diferentes puntos se indicará con cruces dibujadas en la zona de trabajo.

Se ha desarrollado una función “DetectCruzvic” que es capaz de detectar el número de puntos dibujados, la localización de cada punto en píxeles y luego convertir los píxeles a mm.

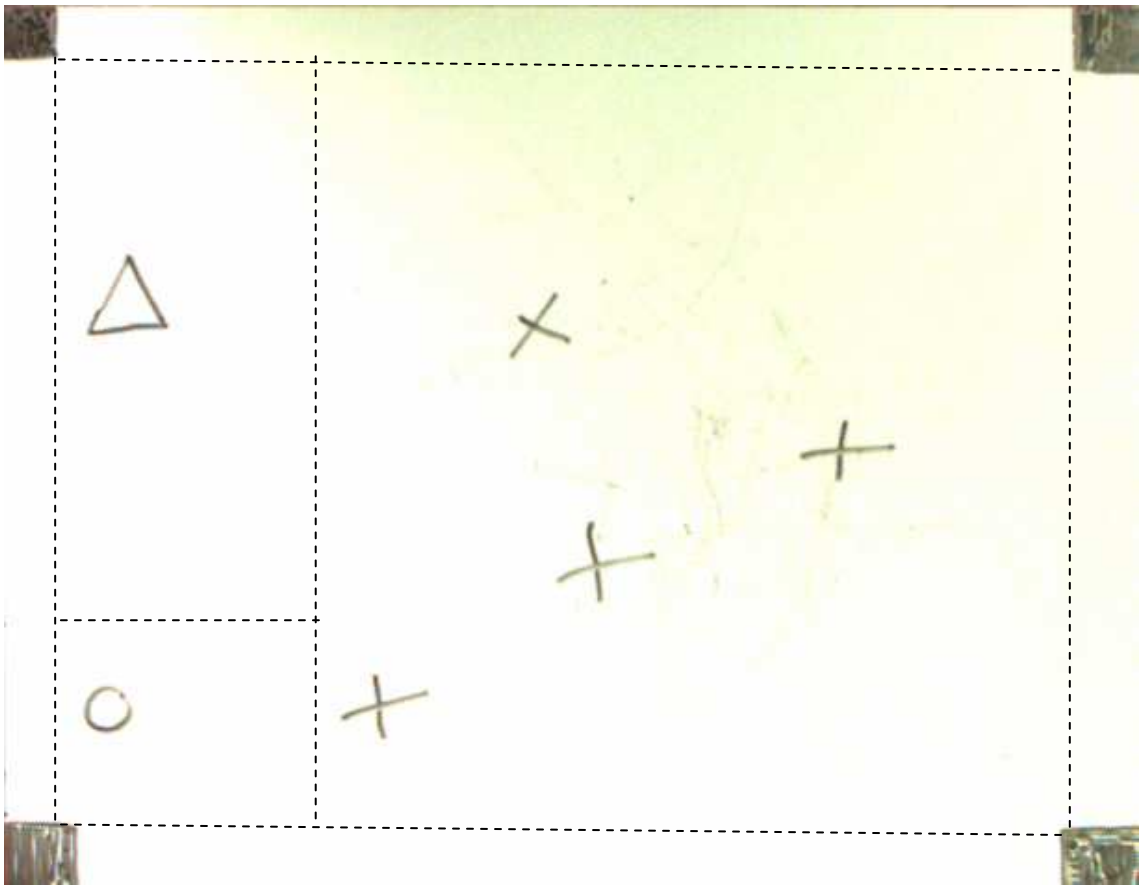
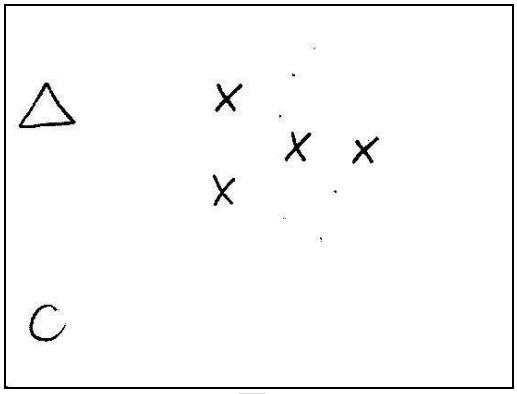


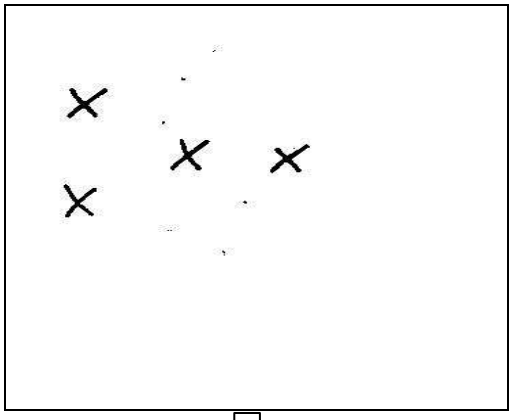
Figura 10 Zona definición puntos mediante cruces

El flujo de programa de la función “DetectCruzvic” queda de la siguiente manera:

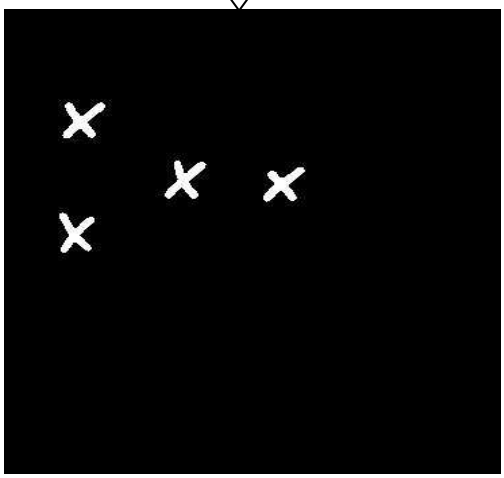
Paso 0. Imagen con zona de trabajo definida



Paso 1. Recorte de la zona definida para dibujar las cruces.
Etapas:
-recorte de la zona definida
L=imcrop(D,[H(2)/3 0 H(2) H(1)]);



Paso 2. Obtención del número de puntos y de su localización
Etapas:
-Erosión de la imagen
Erosion=imerode(L,SE);
-Dilatación de la imagen
Dilatacion=imdilate(Erosion,SE1);
-Detección de bordes
contornocanny=edge(Erosion,'canny');
-Rellenado de los bordes
rellenado=imfill(contornocanny,'holes)
-Etiquetado objetos y calculo centros
[L,Ne]=bwlabel(relimp);
-Calculo centros
prop=regionprops(L,'Centroid');
-Conversion a mm de los centros
for i=1:Ne
mm=prop(i,1).Centroid*0.7;
res(i,1).centro=mm;



4.2.5. Trazo continuo

En un trazo continuo hay una gran cantidad de puntos y esto requiere un mayor tiempo de procesamiento y complejidad a la hora de transmitir los datos.

Se ha desarrollado una función “DetectTrazovic” que es capaz de detectar todos los puntos contenidos en el trazo, la localización de cada punto en píxeles y luego convertir los píxeles a mm.

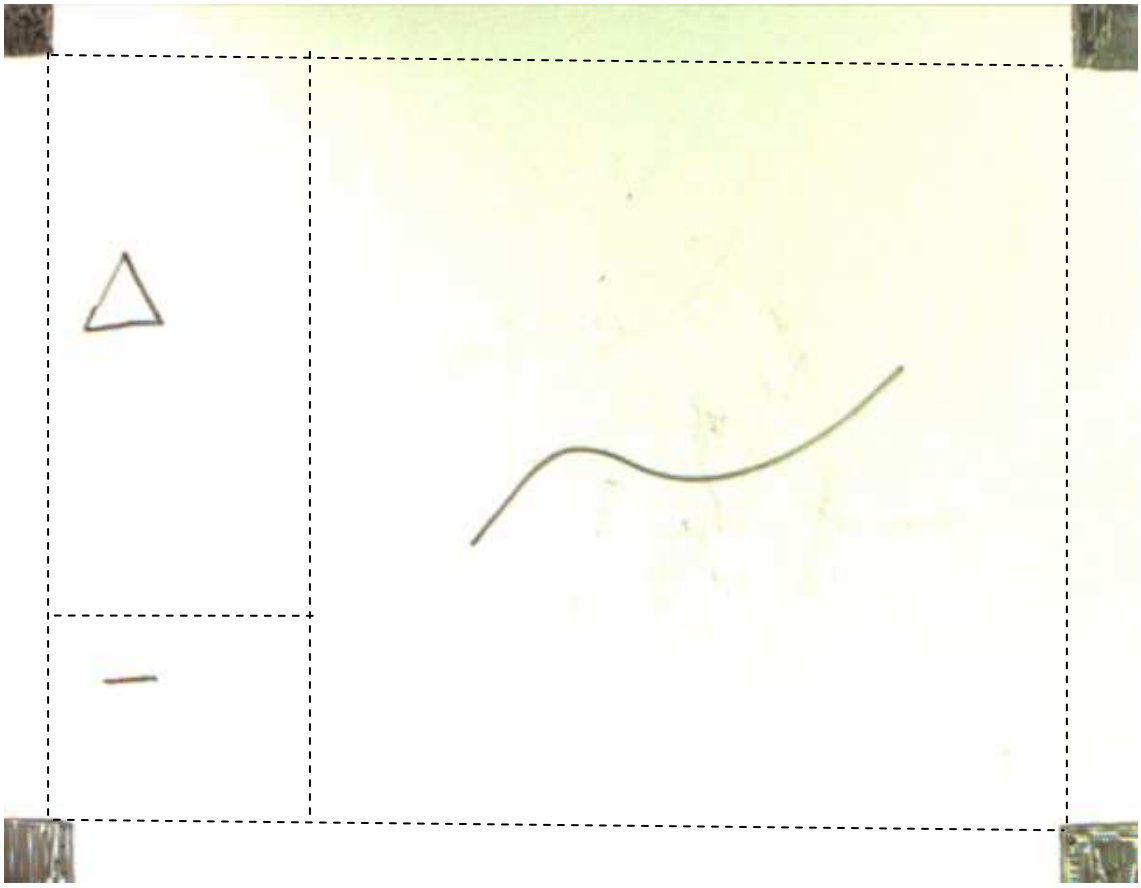
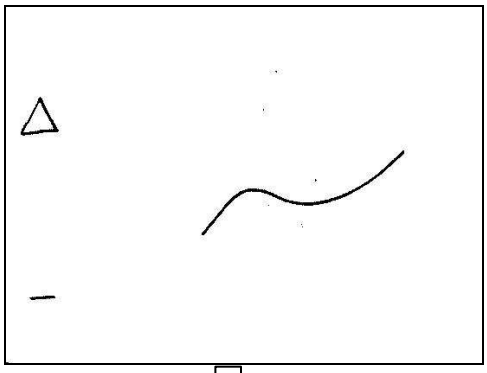


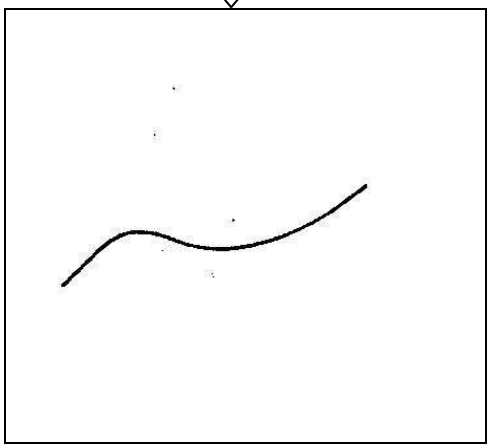
Figura 11 Zona de definición del trazo

El flujo de programa de la función “DetectTrazovic” queda de la siguiente manera:

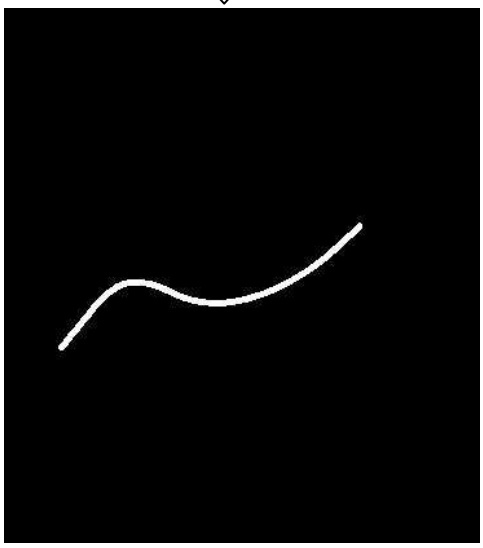
Paso 0. Imagen con zona de trabajo definida



Paso 1. Recorte de la zona definida para dibujar el trazo.
Etapas:
-recorte de la zona definida
L=imcrop(D,[H(2)/3 0 H(2) H(1)]);



Paso 2. Obtención de los puntos y su localizació
Etapas:
-Detección de bordes
contornocanny=edge(Erosion,'canny');
-Rellenado de los bordes
rellenado=imfill(contornocanny,'holes)
-Reducción de píxeles del trazo
BW2=bwmorph(rellenado,’thin’,’Inf’)
-Procesado de puntos
B = bwboundaries(rellenado,8,'noholes');
-Conversion a mm
Puntosmm=B{1,1}*0.7;



4.2.6. Comunicación con el robot

En este proyecto, se ha realizado un socket de comunicación, utilizando el protocolo TCP/IP para realizar la comunicación del brazo robótico con el software Matlab.

Los sockets (también llamados conectores) son un mecanismo de comunicación entre procesos que permiten la comunicación bidireccional tanto entre procesos que se ejecutan en una misma máquina como entre procesos lanzados en diferentes máquinas.

Un socket es un "canal de comunicación" entre dos programas que corren sobre ordenadores distintos o incluso en el mismo ordenador.

4.2.6.1. Protocolo TCP/IP

Las siglas TCP/IP se refieren a un conjunto de protocolos para comunicaciones de datos. Este conjunto toma su nombre de dos de sus protocolos más importantes, el protocolo TCP (Transmission Control Protocol) y el protocolo IP (Internet Protocol).

La popularidad del protocolo TCP/IP no se debe tanto a Internet como a una serie de características que responden a las necesidades actuales de transmisión de datos en todo el mundo, entre las cuales destacan las siguientes:

- *Los estándares del protocolo TCP/IP son abiertos y ampliamente soportados por todo tipo de sistemas, es decir, se puede disponer libremente de ellos y son desarrollados independientemente del hardware de los ordenadores o de los sistemas operativos.*
- *TCP/IP funciona prácticamente sobre cualquier tipo de medio, no importa si es una red Ethernet, una conexión ADSL o una fibra óptica.*
- *TCP/IP emplea un esquema de direccionamiento que asigna a cada equipo conectado una dirección única en toda la red, aunque la red sea tan extensa como Internet.*

La arquitectura de conexión utilizada es la arquitectura cliente-servidor. Las características de cada uno son las siguientes:

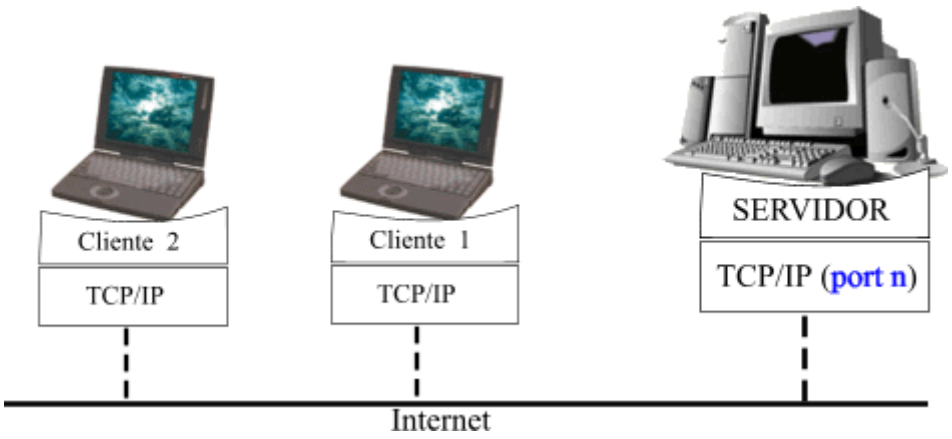
- *Servidor: es el programa que permanece pasivo a la espera de que alguien solicite conexión con él. Está ejecutándose y esperando a que otro quiera conectarse a él. Nunca da "el primer paso" en la conexión. Es el que "sirve" información al que se la pida.*
- *Cliente: es el programa que solicita la conexión para pedir datos al servidor. Es el programa que da el "primer paso" en la conexión. En el momento de ejecutarlo o cuando lo necesite, intenta conectarse al servidor. Es el que solicita información al servidor.*
- 

Figura 12 Arquitectura cliente-servidor

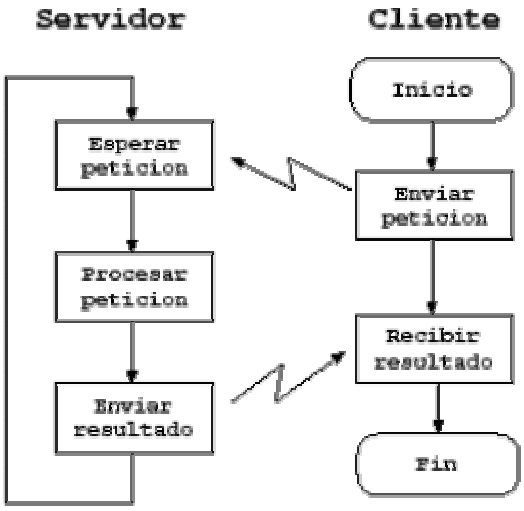


Figura 13 Flujograma cliente servidor

4.2.6.2. Script comunicación con el Robot

En este proyecto se ha definido como servidor el software de análisis de imagen Matlab y como cliente el brazo robótico IRB120.

Tal y como se ha comentado en el apartado anterior software de análisis se mantendrá en un bucle esperando la petición del brazo robótico.

Se ha desarrollado un script en Matlab para gestionar la comunicación con el robot. El script se ha denominado “comunicartodovic_jose”.m. El flujograma del programa es el siguiente:

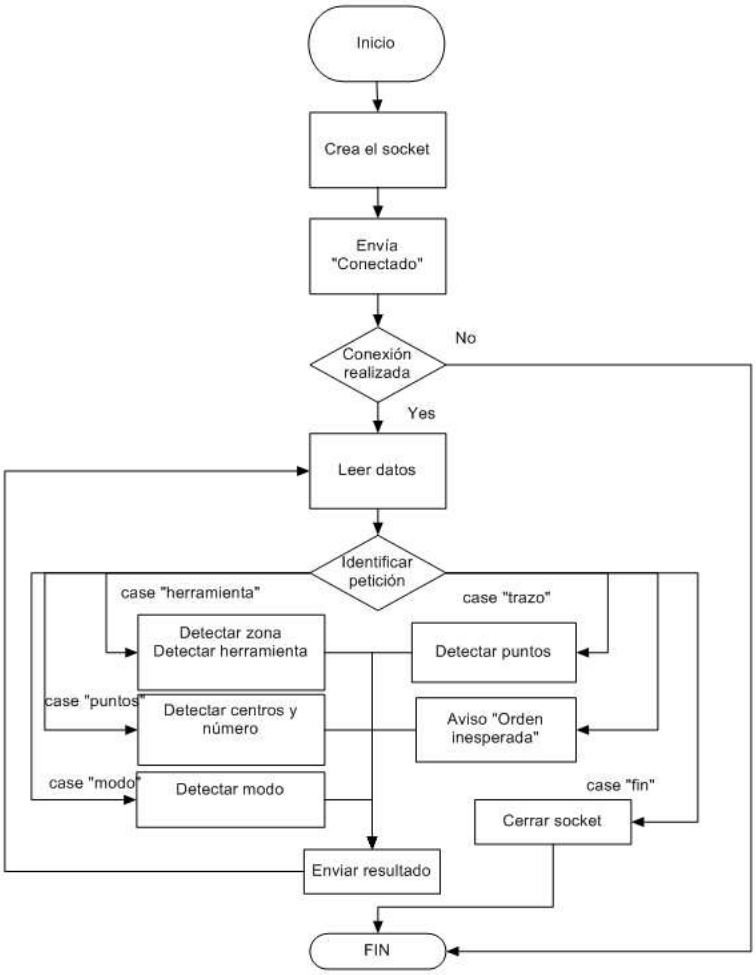


Figura 14 Flujograma comunicación Matlab

- Matlab hace la función de servidor. Gestiona las peticiones realizadas por el cliente, en nuestro caso el robot
- Crea el socket de conexión en una IP dada y un puerto dado.
- Cuando se conecta envía el string “Conectado” al cliente.
- Se mantiene en bucle leyendo datos y analizando la petición del robot mientras la conexión existe.
- La petición del cliente está definida por los siguientes strings:
  - “Herramienta”: el servidor debe analizar la imagen y responder si es un circulo, triangulo o un cuadrado.
  - “Modo”: el servido debe analizar la imagen y responder al robot si es un continuo o discontinuo
  - “Trazo”: el servidor debe analizar el trazo y identificar los puntos y su localización
  - “Puntos”: el servidor debe analizar el trazo y identificar los puntos y su localización
  - “Fin”: el servidor cierra el socket y finaliza conexión

El flujo del script “comunicartodovic_jose”.m detallado es el siguiente:

Paso 0. Crear conexión con robot en puerto 3100 y abrir conexión
Etapas:
-Borrar todo
clear all
-Crear conexión y abrir puerto
finprograma=1;
t = tcpip('127.0.0.1', 3100, 'NetworkRole', 'server');
fopen(t);
-Enviar “Conectado” al Robot.
fprintf(t,'Conectado');

Paso 1. Iniciar bucle, identificar petición cliente, ejecutar lo requerido transmitir los resultados.

Etapas:

```
-Inciar bucle
while finprograma ~= 0
-Leer datos
  data = fread(t);
 str=char(data)';
-Analizar peticion
        switch (str)

        case 'herramienta'
 [ D ]=DetectZonavic('4crucescirculoMODIFICADO.bmp');
           [ tipo ] = Detectformavic(D);
           fwrite(t,tipo);

        case 'modo'
           [ modo ] = Detectsubvic(D);
           fwrite(t,modo);

        case 'puntos'
        [ Centros, numero ] = DetectCruzvic(D);
           complet=[];
           pr=sprintf('?%1d',numero);
                 for i=1:numero
valorx=sprintf('x%2.0f',Centros(i,1).centro(1));
valory=sprintf('y%2.0f',Centros(i,1).centro(2));
                 result=strcat(valorx,valory);
                 complet=[complet result];
                 end;
           final=strcat(pr,complet);
           fwrite(t,final);

        case 'trazo'
           [ Pfinal,Pordenados] = DetectTrazovic(D);
              jl=size(Pfinal);
              completo=[];
                 for i=1:jl(1)
                 valorx=sprintf('x%2.0f',Pfinal(i,1));
                 valory=sprintf('y%2.0f',Pfinal(i,2));
                 result=strcat(valorx,valory);
                 completo=[completo result];
                 end;
            fwrite(t,completo);

        case 'fin'
           disp('zero');
           finprograma=0;
           fwrite(t,'Desconectado')
           fclose(t);
```

5. PROGRAMACIÓN DEL ROBOT

5.1. Lenguaje de programación RAPID

Es un lenguaje de programación de alto nivel diseñado por la compañía ABB para el control de los robots industriales. Una aplicación RAPID consta de un programa y una serie de módulos del sistema.

El programa consiste en una serie de instrucciones que describen el trabajo del robot. Hay instrucciones específicas para los diferentes comandos, tales como mover el robot, establecer una salida, etc.

Las instrucciones tienen por lo general una serie de argumentos asociados que definen que tendrá lugar en una instrucción específica.

Estos argumentos se pueden especificar en una de las siguientes maneras:

- *Como un valor numérico, por ejemplo, 7 o 3.1*
- *Como una referencia a los datos, por ejemplo, sev5*
- *Como una expresión, por ejemplo, 6 + 1 * sev5*
- *Como una llamada de función, por ejemplo, Abs (sev5)*
- *Como un valor de cadena, por ejemplo, "Arreglar parte C"*

Hay tres tipos de rutinas - procedimientos, funciones y rutinas TRAP.

- *Un procedimiento se utiliza como un subprograma.*
- *Una función devuelve un valor de un tipo específico y se usa como un argumento de una instrucción.*
- *Rutinas TRAP proporcionan una forma de responder a las interrupciones.*

Hay tres tipos de datos - constantes, variables y variables persistentes.

- *Una constante representa un valor estático y sólo se le puede asignar un nuevo valor manualmente.*
- *Una variable también se puede asignar un nuevo valor durante la ejecución del programa.*
- *Una persistente puede ser descrito como una variable "persistente". Cuando un programa es guardado el valor de inicialización refleja el valor actual de la persistente.*

5.2. Programa del robot

Se ha desarrollado la programación del robot en lenguaje RAPID. Se puede apreciar el flujograma en la siguiente figura.

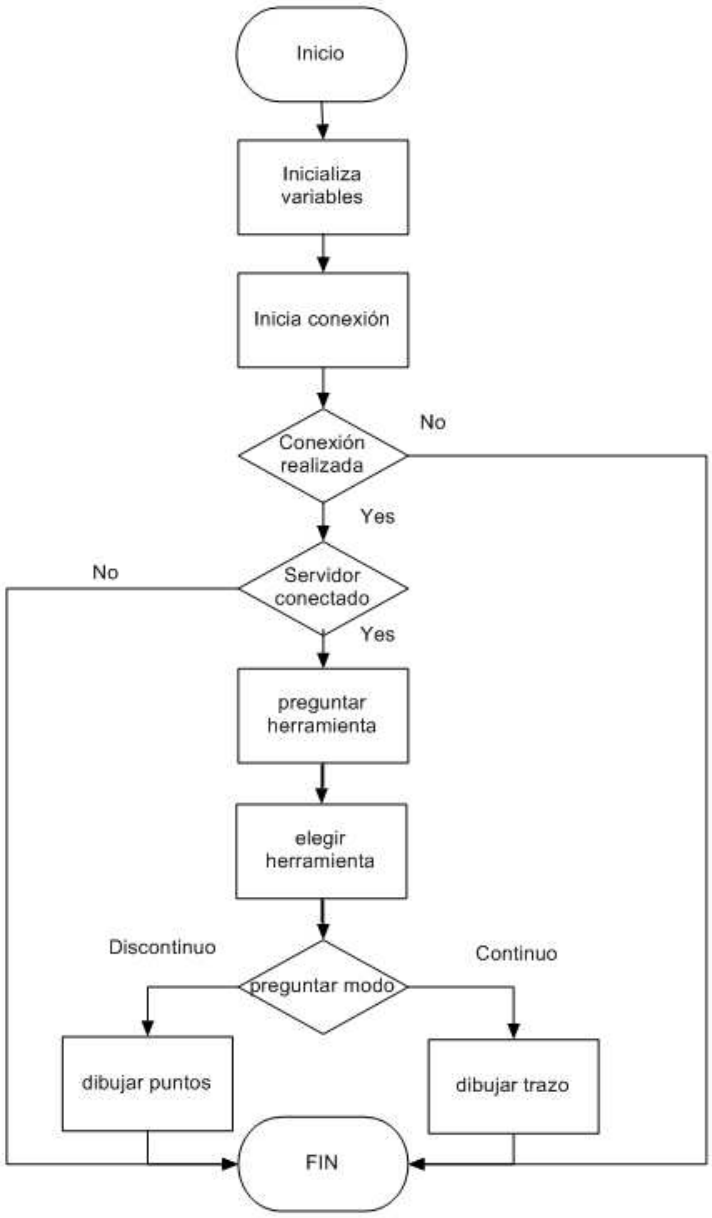


Figura 15 Flujograma programa Robot Studio

Ahora vamos a detallar el contenido de los módulos CalibData y Main.

5.2.1. Módulo CalibData

Definición de herramientas y objetos del sistema. Se han definido las herramientas ventosa, boli y herramienta. También se define la base de blanca qué es donde el robot dibujará en base a la información recibidad del servidor.	MODULE CalibData PERS tooldata TcpVentosa:=[TRUE,[[155.81,46.1,140.51],[0.930110719,-0.086693109,0.355360765,0.033122218]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]]; PERS tooldata TcpBoli:=[TRUE,[[-33.33,-123.6,129.57],[0.920732906,0.377323349,-0.091962974,-0.037693718]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]]; PERS tooldata TcpRetolador:=[TRUE,[[-109.97,109.1,153.66],[0.920231344,-0.232512569,-0.305228560.077121344]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]]; TASK PERS wobjdata WO_BaseBlanca:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]],[[132.182,84.511,30.1],[1,0,0,0]]]; ENDMODULE

5.2.2. Módulo Main

Definición de variables y constantes a utilizar durante la ejecución del programa Se han definido variables de tipo string, num y bool. También se define en este apartado el puerto a utlizar y la dirección IP.	!CONST robtarget Inicial:=[[200,100,50],[0,1,0,0],[0,-1,-2,1],[9E9,9E9,9E9,9E9,9E9,9E9]]; CONST robtarget Inicial:=[[100,132,50],[0,1,0,0],[0,-1,-2,1],[9E9,9E9,9E9,9E9,9E9,9E9]]; VAR robtarget Posicio0:=[[200,100,50],[0,1,0,0],[0,-1,-2,1],[9E9,9E9,9E9,9E9,9E9,9E9]]; VAR robtarget Posicio1:=[[200,100,50],[0,1,0,0],[0,-1,-2,1],[9E9,9E9,9E9,9E9,9E9,9E9]]; VAR robtarget Posicio2:=[[200,100,50],[0,1,0,0],[0,-1,-2,1],[9E9,9E9,9E9,9E9,9E9,9E9]]; VAR robtarget Posicio3:=[[200,100,50],[0,1,0,0],[0,-1,-2,1],[9E9,9E9,9E9,9E9,9E9,9E9]]; VAR robtarget Posicio4:=[[200,100,50],[0,1,0,0],[0,-1,-2,1],[9E9,9E9,9E9,9E9,9E9,9E9]]; VAR socketdev PC; VAR string PC_IP:="127.0.0.1"; !IP assignada per connexions locals, el teu propi ordinador. VAR num port:=3100; VAR string denviar; VAR string drecibido; VAR string msg_inicial:="Robot ABB connectado"; VAR bool recibir; VAR num cantpuntos;

Procedimiento iniciar conexión.

Se creo el socket para realizarla conexión.

Una parte del procedimiento define cómo hay que actuar en caso de errores.

```
PROC iniciar_connexion()
  SocketCreate PC;
  SocketConnect PC,PC_IP,port;
  SocketReceive PC\Str:=drecibido;
  ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
     RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
     RETURN;
  ELSE

  ENDIF
 ! SocketSend PC\Str:=msg_inicial;
ENDPROC
```

Procedimiento preguntar herramienta.

Cuando se ejecuta este procedimiento se envía al servidor el string “herramienta”. El servidor identifica el string y ejecuta el análisis de imagen adecuado enviando el resultado de este análisis al robot.

```
PROC preguntar_herramienta()
  denviar:="herramienta";
  SocketSend PC\Str:=denviar;
  SocketReceive PC\Str:=drecibido;
ENDPROC
```

Procedimiento preguntar modo.

Cuando se ejecuta este procedimiento se envía al servidor el string “modo”. El servidor identifica el string y ejecuta el análisis de imagen adecuado enviando el resultado de este análisis al robot.

```
PROC preguntar_modo()
  denviar:="modo";
  SocketSend PC\Str:=denviar;
  SocketReceive PC\Str:=drecibido;
ENDPROC
```

Procedimiento elegir herrramienta.

Cuando se ejecuta este procedimiento se identifica el string recibido del servidor y mediante la condición “IF” se ejecuta una acción u otra dependiendo del string.

Los string se pueden recibir son circulo, cuadrado y rectangulo.

Las herramientas se han definido de la siguiente manera:

-Círculo: ventosa

-Cuadrado: boli

-Triángulo: rotulador

```
PROC elegir_herramienta()

    IF  drecibido ="circulo" THEN
       tool:=[TRUE,[[155.81,46.1,140.51],[0.930110719,-
0.086693109,0.355360765,0.033122218]],[0.7,[14.3,0,78.3],[1,(
],0.012,0.016,0.014]];
    ELSEIF drecibido ="cuadrado" THEN
       tool:=[TRUE,[[-33.33,-
123.6,129.57],[0.920732906,0.377323349,-0.091962974,-
0.037693718]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]];
    ELSE
       tool:=[TRUE,[[-109.97,109.1,153.66],[0.920231344,-
0.232512569,-0.305228564,-
0.077121344]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]];
    ENDIF

ENDPROC
```

Procedimiento dibujar puntos.

En este procedimiento podemos diferenciar varias partes.

-Tratamiento de los datos recibidos del servidor en cuanto a número de puntos dibujados.

-Tratamiento de los datos recibidos del servidor en relación a las coordenadas X e Y de cada punto.

-Instrucción de movimiento del robot.

```
PROC dibujar_puntos ()
  !analiza datos con formato
?3x250y150x275y200x300y240
  long:=StrLen (drecibido);
  lugar:=StrFind(drecibido,1,"?");
  cadena:=StrPart(drecibido,lugar+1,1);
  ok:=Strtoval (cadena,cantpuntos);
  cadena:=StrPart(drecibido,3,long-2);

  FOR i FROM 1 TO cantpuntos DO
    !eje x
        longc:=StrLen (cadena);
        lugarx:=StrFind(cadena,1,"x");
        lugary:=StrFind(cadena,1,"y");
    cadenax:=StrPart(cadena,lugarx+1,lugary-2);
        ok:=Strtoval (cadenax,Valorx);
   cadena:=StrPart(cadena,lugary,longc-(lugary-1))

    !eje y
        longc:=StrLen (cadena);
        lugary:=StrFind(cadena,1,"y");
        lugarx:=StrFind(cadena,1,"x");
   cadenay:=StrPart(cadena,lugary+1,lugarx-2);
        ok:=Strtoval (cadenay,Valory);
  cadena:=StrPart(cadena,lugarx,longc-(lugarx-1));

    !movimiento
          SingArea \Wrist;
        ConfJ \Off;
        MoveJ
Offs(Inicial,Valory,Valorx,0),v100,z0,tool\WObj:=W
BaseBlanca;
        WaitTime \InPos,1;
        MoveJ Offs(Inicial,Valory,Valorx,-
40),v100,z0,tool\WObj:=WO_BaseBlanca;
        WaitTime \InPos,1;

  ENDFOR


ENDPROC
```

Procedimiento dibujar trazo.

En este procedimiento podemos diferenciar varias partes.

-Tratamiento de los datos recibidos del servidor en relación a las coordenadas X e Y de cada punto.

-Instrucción de movimiento del robot.

```
PROC dibujar_trazo ()
  !analiza datos con formato
x250y150x275y200x300y240x214y264
   n:=10;

  FOR i FROM 1 TO n DO

        long:=StrLen (drecibido);
        IF long > 0 THEN
     !eje x
           lugarx:=StrFind(drecibido,1,"x");
           lugary:=StrFind(drecibido,1,"y");
  cadenax:=StrPart(drecibido,lugarx+1,lugary-2);
           ok:=Strtoval (cadenax,Valorx);
drecibido:=StrPart(drecibido,lugary,long-(lugary-1)

      !eje y
           long:=StrLen(drecibido);
           lugary:=StrFind(drecibido,1,"y");
           lugarx:=StrFind(drecibido,1,"x");
   cadenay:=StrPart(drecibido,lugary+1,lugarx-2);
            ok:=Strtoval (cadenay,Valory);
drecibido:=StrPart(drecibido,lugarx,long-(lugarx-1))

     !movimiento
          ConfJ \Off;
  MoveJ
Offs(Inicial,Valorx,Valory,0),v100,z0,tool\WObj:=W
BaseBlanca;
          WaitTime \InPos,0;

    ELSE
MoveAbsJ
obj_inicial,v100,z100,TcpRetolador\WObj:=WO_Ba
Blanca;
           n:=i-1;

         ENDIF

  ENDFOR

ENDPROC
```

Procedimiento main.

Desde este procedimiento se llama a los distintos procedimientos nombrados anteriormente como el procedimiento de inicio de conexión, elegir herramienta, preguntar modo,……

Se ha creado un bucle que está activo mientras exista conexión con el servidor.

Dentro de este bucle se realizan una serie de acciones si se recibe del servidor el string “Conectado”.

El orden en que el cliente, en nuestro caso el robot, pide información del servidor es el siguiente:

-1º Pregunta qué herramienta debe utilizar.

-2º Pregunta qué tipo de puntos son: discontinuos o continuos.

-

```
PROC main()

  iniciar_connexion;
  MoveAbsJ
obj_inicial,v500,z100,TcpRetolador\WObj:=WO_Ba
Blanca;
  WHILE true DO

   !ejecutar;
    memb:=StrMemb(drecibido,2,"Co");
    IF memb THEN
    preguntar_herramienta;
     elegir_herramienta;
    MoveAbsJ
JointTarget_1,v100,z100,TcpBoli\WObj:=WO_Base
anca;
    WaitTime \InPos,2;
    preguntar_modo
       IF  drecibido ="discontinuo" THEN
            denviar:="puntos";
          SocketSend PC\Str:=denviar;
          SocketReceive PC\Str:=drecibido;
           dibujar_puntos;

      ELSEIF drecibido ="continuo" THEN
          denviar:="trazo";
          SocketSend PC\Str:=denviar;
          SocketReceive PC\Str:=drecibido;
          dibujar_trazo;

       ENDIF
     denviar:="fin";
     SocketSend PC\Str:=denviar;
     SocketReceive PC\Str:=drecibido;

    ELSE
        SocketClose PC;

    ENDIF

  ENDWHILE

ENDPROC
```

6. RESULTADOS Y CONCLUSIONES

6.1. Resultados y conclusiones

Revisando los objetivos planteados al principio de este proyecto podemos decir que en líneas generales se han alcanzado con resultados satisfactorios.

El Objetivo principal de este proyecto era establecer el protocolo de comunicación trabajador-robot mediante imágenes. El trabajador debería poder controlar el robot mediante dibujos realizados en la mesa de trabajo. Este objetivo se ha alcanzado ya que se ha conseguido que mediante imágenes el robot sea capaz de saber que herramienta utilizar e identificar la localización de los puntos que debe trazar en la mesa de trabajo.

Inicialmente fue necesario bastante tiempo para introducirnos en el mundo del procesamiento digital de imágenes y del software que hay en el mercado. Un mundo en el que se partía de cero pero que enseguida se observó que utilizando el procesamiento digital se puede transmitir una gran cantidad de información a través de imágenes. Esta manera de transmitir información aporta importantes mejoras que son de aplicación directa en el mundo de la industria.

A lo largo de este proyecto se ha apreciado el gran avance que sería para los sistemas productivos que la programación y aún más la comunicación entre el trabajador y robot fuera lo más cercana y simple posible.

Se puede decir que con este proyecto hemos descubierto las posibilidades que tiene el tratamiento de imágenes a la hora de comunicarse con máquinas. Algo realmente relevante a la hora de simplificar la programación de éstas. Ya que las fronteras entre el trabajador y el robot se reducen y la cooperación aumenta.

Podemos concluir que combinando el software de tratamiento de imágenes y los robots se puede reducir en gran medida el tiempo de programación de los robots. Esta reducción de complejidad en la programación conlleva a un aumento de productividad y demuestra una adaptación al cambio muy interesante en los medios productivos.

6.2. Avances futuros

Durante la realización del trabajo nos hemos centrado en el objetivo que se había definido que era el protocolo de comunicación trabajador-robot mediante imágenes pero se ha observado que hay muchas posibilidades de avanzar en este tema de la comunicación trabajador-robot.

A continuación se van a detallar las opciones que se podrían llevar a cabo para ampliar las posibilidades de comunicación trabajador-robot.

Una opción interesante sería desarrollar profundizar en el procesamiento digital de imagen de manera que el trabajador pudiera utilizar imágenes de colores e incluso utilizar vídeos con imágenes para enviar al robot la información necesaria para ejecutar las acciones correspondientes.

Otra opción para profundizar más en el procesamiento digital de imágenes sería que el trabajador a través de gestos con las manos pudiera comunicar al robot que es lo que tiene que hacer.

Finalmente, cabe destacar que durante la realización de este proyecto nos hemos dado cuenta de las infinitas posibilidades que nos puede aportar el procesamiento digital de imagen en lo que interacción con robots se refiere. Sin duda alguna un campo muy interesante para explorar.

:

7. BIBLIOGRAFIA

7.1. Libros y artículos

- *Barrientos, A. et al.“Fundamentos de Robótica". Mc.GrawHill. 2n Ed 2007*
- *Image Processing Toolbox User’s Guide R2014b*
- *Manual de referencia técnica. Instrucciones, funciones y tipos de datos de RAPID. Revisión:J. ABB Robotics Products*
- *Manual del operador. Robot Studio. Revisión:E.ABB Robotics Products*
- *Un curso de Matlab. Departamento de matemática Aplicada, Universidad de Zaragoza (2006).*
- *RAPID Reference Manual. Revisión:1.ABB Robotics Products*

7.2. Direcciones de Internet

- *Image Processing Toolbox Documentation-Mathworks España. Consulta: desde Septiembre a Diciembre 2014.*

http://es.mathworks.com/help/images/index.html;jsessionid=bc59399c4ec3d6114b046979a460

- *ABB IRB 120 - Robots (Robótica) - ABB en España. Consulta: Septiembre 2014*

http://www.abb.es/product/seitp327/26a64e26204118e5c12576a4004a8497.aspx

.

8. ANEXO

8.1. Código de análisis de imagen

A continuación detallamos el código realizado para el procesamiento de digital de las imágenes utilizando Matlab.

```
% 1 Crear conexión con robot con puerto 3100 y abrir conexión
clear all
finprograma=1;

%t = tcpip('10.1.107.15', 3100, 'NetworkRole', 'server');
t = tcpip('127.0.0.1', 3100, 'NetworkRole', 'server');
fopen(t);
fprintf(t,'Conectado');


while finprograma ~= 0
% 2 Leer datos

 data = fread(t);
 str=char(data)';

% 2 Analizar peticion

    switch (str)


        case 'herramienta'
            [ D ] = DetectZonavic('4crucescirculoMODIFICADO.bmp');
            [ tipo ] = Detectformavic(D);
            fwrite(t,tipo);

        case 'modo'
            [ modo ] = Detectsubvic(D);
            fwrite(t,modo);

        case 'puntos'
            [ Centros, numero ] = DetectCruzvic(D);
            complet=[];
            pr=sprintf('?%1d',numero);
                    for i=1:numero
                    valorx=sprintf('x%2.0f',Centros(i,1).centro(1));
                    valory=sprintf('y%2.0f',Centros(i,1).centro(2));
                    result=strcat(valorx,valory);
                    complet=[complet result];
                    %fprintf(t,'#');
                    end;
            final=strcat(pr,complet);
```

```
            fwrite(t,final);

        case 'trazo'
            [ Pfinal,Pordenados] = DetectTrazovic(D);
                jl=size(Pfinal);
                completo=[];
                    for i=1:jl(1)
                    valorx=sprintf('x%2.0f',Pfinal(i,1));
                    valory=sprintf('y%2.0f',Pfinal(i,2));
                    result=strcat(valorx,valory);
                    completo=[completo result];
                    %fprintf(t,'x%3d',Pordenados(i,1));
                    %fprintf(t,'y%3d',Pordenados(i,2));
                    end;
             fwrite(t,completo);

        case 'fin'
            disp('zero');
            finprograma=0;
            fwrite(t,'Desconectado')
            fclose(t);

        otherwise
            warning('Orden inesperada');
    end

%  fwrite(t,'Siguiente orden / "fin" para detener')

end;


function [ D ] = DetectZonavic(nombrefoto)


C=imread(nombrefoto);
CGris=rgb2gray(C);
level = 0.85;
Cbw=im2bw(CGris,level);
Climpio= bwmorph(Cbw, 'dilate', 2);
Climpio=bwareaopen(Cbw,100);
imtool(Climpio)
SE1=strel('square',20);
Dilatacion=imdilate(Climpio,SE1);
contornocanny=edge(Dilatacion,'canny');
SE2=strel('square',5);
Dilatacion2=imdilate(contornocanny,SE2);
rellenado=imfill(Dilatacion2,'holes');
rellenado2=bwareaopen(rellenado,30);
%etiquetado de objetos y calculo de centros
[L,Ne]=bwlabel(rellenado2);
prop=regionprops(L,'Centroid');
%Recorte zona de trabajo (distancia entre centros de cuadrados)
min=prop(1,1).Centroid;
```

```
x=2*min(1);
y=2*min(2);
w=prop(3,1).Centroid;
width=w(1)-x-(x/2);
h=prop(2,1).Centroid;
height=h(2)-y-(y/2);
D=imcrop(Climpio,[x y width height]);
imtool (D)


end

function [ tipo ] = Detectformavic( D )

F=size(D);
G=imcrop(D,[0 0 F(2)/3 F(1)*2/3]);
imtool (G)
contornocanny=edge(G,'canny');
rellenado=imfill(contornocanny,'holes');
rellenado=bwareaopen(rellenado,30);
 [B,L] = bwboundaries(rellenado,'noholes');
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
for k = 1:length(B)
boundary = B{k};
plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
stats = regionprops(L,'Area','Perimeter');
for k = 1:length(B)
boundary = B{k};
area = stats(k).Area;
perimetro = stats(k).Perimeter;
% compute the roundness metric
metric = 4*pi*area/perimetro^2;
% display the results
metric_string = sprintf('%2.2f',metric);
text(boundary(1,2)-35,boundary(1,1)+13,metric_string,'Color','y',...
       'FontSize',14,'FontWeight','bold');
end


function [ modo ] = Detectsubvic(D)

F=size(D);
G=imcrop(D,[0 F(1)*2/3 F(2)/3 F(1)/3]);
imtool (G)
contornocanny=edge(G,'canny');
rellenado=imfill(contornocanny,'holes');
rellenado=bwareaopen(rellenado,30);
 [B,L] = bwboundaries(rellenado,'noholes');
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
```

```
for k = 1:length(B)
boundary = B{k};
plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
stats = regionprops(L,'Area','Perimeter');
for k = 1:length(B)
boundary = B{k};
area = stats(k).Area;
perimetro = stats(k).Perimeter;
% compute the roundness metric
metric = 4*pi*area/perimetro^2;
% display the results
metric_string = sprintf('%2.2f',metric);
text(boundary(1,2)-35,boundary(1,1)+13,metric_string,'Color','y',...
       'FontSize',14,'FontWeight','bold');
end

if metric >=1/10
ab='discontinuo';
else ab='continuo';
end;
modo=ab


end


function [ Centros, numero ] = DetectCruzvic(D)


H=size(D);
L=imcrop(D,[H(2)/3 0 H(2) H(1)]);
imtool (L)
SE=strel('square',2);
Erosion=imerode(L,SE);
contornocanny=edge(Erosion,'canny');
rellenado=imfill(contornocanny,'holes');
rellenado=bwareaopen(rellenado,30);
relimp=bwareaopen(rellenado,50);
imtool(relimp)
 [L,Ne]=bwlabel(relimp);
prop=regionprops(L,'Centroid');
%conversion a mm de los centros
for i=1:Ne
mm=prop(i,1).Centroid*0.7;
res(i,1).centro=mm;
end
numero=Ne;
Centros=res;


end
```

```
function [ Pfinal,Pordenados] = DetectTrazovic(D)

H=size(D);
L=imcrop(D,[H(2)/3 0 H(2) H(1)]);
imtool (L);
Climpio=bwareaopen(L,30);
contornocanny=edge(Climpio,'canny');
rellenado=imfill(contornocanny,'holes');
rellenado=bwareaopen(rellenado,30);
BW2 = bwmorph(rellenado,'thin','Inf');
imtool (BW2)
%procesado de puntos
B = bwboundaries(BW2,8,'noholes');
%conversion a mm
Puntosmm=B{1,1}*0.7;
Redondear=round(Puntosmm);
Pordenados=unique(Redondear,'rows','stable');
fg=size(Pordenados);
Pfinal=Pordenados(1:10:fg(1),1:2);

end
```

8.2. Código programación robot

A continuación detalllamos el programa en lenguaje RAPID realizado para definir las acciones a relizar por el robot.

```
MODULE MainModule

!CONST robtarget Inicial:=[[200,100,50],[0,1,0,0],[0,-1,-
2,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Inicial:=[[100,132,50],[0,1,0,0],[0,-1,-
2,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR robtarget Posicio0:=[[200,100,50],[0,1,0,0],[0,-1,-
2,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR robtarget Posicio1:=[[200,100,50],[0,1,0,0],[0,-1,-
2,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR robtarget Posicio2:=[[200,100,50],[0,1,0,0],[0,-1,-
2,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR robtarget Posicio3:=[[200,100,50],[0,1,0,0],[0,-1,-
2,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR robtarget Posicio4:=[[200,100,50],[0,1,0,0],[0,-1,-
2,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR socketdev PC;
VAR string PC_IP:="127.0.0.1"; !IP assignada per connexions locals, el
teu propi ordinador.
VAR num port:=3100;
VAR string denviar;
```

```
VAR string drecibido;
VAR string msg_inicial:="Robot ABB connectado";
VAR bool recibir;
VAR num cantpuntos;
VAR num lugar;
VAR num lugarx;
VAR num lugary;
VAR string cadena;
VAR string cadenax;
VAR string cadenay;
VAR string prueba;
VAR num Valorx;
VAR num Valory;
PERS tooldata tool:=[TRUE,[[-109.97,109.1,153.66],[0.920231,-0.232513,-
0.305229,-0.0771213]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]];
VAR bool ok;
VAR num long;
VAR num longc;
VAR num n;
VAR bool memb;



PROC iniciar_connexion()
   SocketCreate PC;
   SocketConnect PC,PC_IP,port;
   SocketReceive PC\Str:=drecibido;
   ERROR
   IF ERRNO=ERR_SOCK_TIMEOUT THEN
       RETRY;
   ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
       RETURN;
   ELSE

   ENDIF
  ! SocketSend PC\Str:=msg_inicial;
ENDPROC



PROC preguntar_herramienta()
   denviar:="herramienta";
   SocketSend PC\Str:=denviar;
   SocketReceive PC\Str:=drecibido;
ENDPROC


PROC preguntar_modo()
  denviar:="modo";
  SocketSend PC\Str:=denviar;
  SocketReceive PC\Str:=drecibido;
ENDPROC
```

```
PROC elegir_herramienta()

     IF  drecibido ="circulo" THEN
        tool:=[TRUE,[[155.81,46.1,140.51],[0.930110719,-
0.086693109,0.355360765,0.033122218]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,
0.016,0.014]];
     ELSEIF drecibido ="cuadrado" THEN
         tool:=[TRUE,[[-33.33,-123.6,129.57],[0.920732906,0.377323349,-
0.091962974,-
0.037693718]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]];
     ELSE
         tool:=[TRUE,[[-109.97,109.1,153.66],[0.920231344,-0.232512569,-
0.305228564,-
0.077121344]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]];
     ENDIF

ENDPROC



PROC dibujar_puntos ()
   !analiza datos con formato ?3x250y150x275y200x300y240
   long:=StrLen (drecibido);
   lugar:=StrFind(drecibido,1,"?");
   cadena:=StrPart(drecibido,lugar+1,1);
   ok:=Strtoval (cadena,cantpuntos);
   cadena:=StrPart(drecibido,3,long-2);


   FOR i FROM 1 TO cantpuntos DO
       !eje x
            longc:=StrLen (cadena);
            lugarx:=StrFind(cadena,1,"x");
            lugary:=StrFind(cadena,1,"y");
            cadenax:=StrPart(cadena,lugarx+1,lugary-2);
            ok:=Strtoval (cadenax,Valorx);
            cadena:=StrPart(cadena,lugary,longc-(lugary-1));

       !eje y
            longc:=StrLen (cadena);
            lugary:=StrFind(cadena,1,"y");
            lugarx:=StrFind(cadena,1,"x");
            cadenay:=StrPart(cadena,lugary+1,lugarx-2);
            ok:=Strtoval (cadenay,Valory);
            cadena:=StrPart(cadena,lugarx,longc-(lugarx-1));

       !movimiento
            SingArea \Wrist;
            ConfJ \Off;
  MoveJ Offs(Inicial,Valory,Valorx,0),v100,z0,tool\WObj:=WO_BaseBlanca;
            WaitTime \InPos,1;
 MoveJ Offs(Inicial,Valory,Valorx,-40),v100,z0,tool\WObj:=WO_BaseBlanca;
            WaitTime \InPos,1;
```

```
   ENDFOR


ENDPROC

PROC dibujar_trazo ()
   !analiza datos con formato x250y150x275y200x300y240x214y264
    n:=10;

   FOR i FROM 1 TO n DO

            long:=StrLen (drecibido);
            IF long > 0 THEN
        !eje x
                lugarx:=StrFind(drecibido,1,"x");
                lugary:=StrFind(drecibido,1,"y");
                cadenax:=StrPart(drecibido,lugarx+1,lugary-2);
                ok:=Strtoval (cadenax,Valorx);
                drecibido:=StrPart(drecibido,lugary,long-(lugary-1));

        !eje y
                long:=StrLen(drecibido);
                lugary:=StrFind(drecibido,1,"y");
                lugarx:=StrFind(drecibido,1,"x");
                cadenay:=StrPart(drecibido,lugary+1,lugarx-2);
                ok:=Strtoval (cadenay,Valory);
                drecibido:=StrPart(drecibido,lugarx,long-(lugarx-1));

      !movimiento
             ConfJ \Off;
    MoveJ Offs(Inicial,Valorx,Valory,0),v100,z0,tool\WObj:=WO_BaseBlanca;
             WaitTime \InPos,0;

            ELSE

      MoveAbsJ obj_inicial,v100,z100,TcpRetolador\WObj:=WO_BaseBlanca;
                n:=i-1;

            ENDIF


   ENDFOR


ENDPROC
```

```
PROC main()

   iniciar_connexion;
   MoveAbsJ obj_inicial,v500,z100,TcpRetolador\WObj:=WO_BaseBlanca;

   WHILE true DO

     memb:=StrMemb(drecibido,2,"Co");
     IF memb THEN
     preguntar_herramienta;
     elegir_herramienta;
     MoveAbsJ JointTarget_1,v100,z100,TcpBoli\WObj:=WO_BaseBlanca;
     WaitTime \InPos,2;
     preguntar_modo;
              IF  drecibido ="discontinuo" THEN
            denviar:="puntos";
            SocketSend PC\Str:=denviar;
            SocketReceive PC\Str:=drecibido;
            dibujar_puntos;

        ELSEIF drecibido ="continuo" THEN
            denviar:="trazo";
            SocketSend PC\Str:=denviar;
            SocketReceive PC\Str:=drecibido;
            dibujar_trazo;

        ENDIF
       denviar:="fin";
       SocketSend PC\Str:=denviar;
       SocketReceive PC\Str:=drecibido;

     ELSE
           SocketClose PC;

     ENDIF


   ENDWHILE
   !Gestion de errores
   ERROR
   IF ERRNO=ERR_SOCK_TIMEOUT THEN
       RETRY;
   ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
       iniciar_connexion;
       RETRY;
   ELSE

   ENDIF

ENDPROC

ENDMODULE
```

```
MODULE CalibData
      PERS tooldata
TcpVentosa:=[TRUE,[[155.81,46.1,140.51],[0.930110719,-
0.086693109,0.355360765,0.033122218]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,
0.016,0.014]];
      PERS tooldata TcpBoli:=[TRUE,[[-33.33,-
123.6,129.57],[0.920732906,0.377323349,-0.091962974,-
0.037693718]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]];
      PERS tooldata TcpRetolador:=[TRUE,[[-
109.97,109.1,153.66],[0.920231344,-0.232512569,-0.305228564,-
0.077121344]],[0.7,[14.3,0,78.3],[1,0,0,0],0.012,0.016,0.014]];
      TASK PERS wobjdata
WO_BaseBlanca:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]],[[132.182,-
84.511,30.1],[1,0,0,0]]];
      TASK PERS wobjdata
WO_Esborrador:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]],[[246.182,-
190.528106985,92],[1,0,0,0]]];
      TASK PERS wobjdata PecaProves:=[FALSE,TRUE,"",[[226.181999999999,-
15.5109999999996,30.1],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
      TASK PERS wobjdata
CoordPecesCil:=[FALSE,TRUE,"",[[0,0,0],[1,0,0,0]],[[226.182,-
15.511,33.6],[1,0,0,0]]];
ENDMODULE
```