Instituto Balseiro

Introducción a la Robótica
Proyecto Final

Detección, manipulación y clasificación de piezas utilizando cámara y brazo robótico

Docentes
José Reyoso
Andrés Laudari

Alumnos
Enzo Schiavoni
Francisco Viñon
Luciano Gutierrez

Junio 2019

Índice

1. Resumen

En el presente informe se detallarán todas las actividades llevadas a cabo para cumplimentar con el proyecto final de la cátedra Introducción a la Robótica, perteneciente a la carrera Ingeniería Electromecánica con Orientación en Automatización Industrial de la Facultad de Ingeniería, UNLPam, dictada en el Instituto Balseiro.

Dicho proyecto consiste en la programación en lenguaje Python de un manipulador de 5 grados de libertad, el cual deberá tomar piezas ubicadas en la mesa de trabajo para luego clasificarlas según sea dispuesto. La identificación y localización de las piezas serán obtenidas mediante las imágenes captadas por una cámara, las cuales serán procesadas usando funciones de la biblioteca de visión artificial OpenCV.

Como punto de partida para el presente proyecto, se cuenta con los conocimientos adquiridos en prácticas anteriores relacionadas con el filtrado de imágenes, y, además, se tiene el cálculo de la cinemática inversa del robot que será utilizado.

Se detallarán aquí los pasos seguidos en relación al análisis de las imágenes (filtrado, manejo de posiciones dentro de éstas, transformaciones de ejes, escalado de magnitudes) y también sobre los comandos enviados al robot (cálculo y cárga de ángulos, verificación de errores).

2. Desarrollo

Se tiene como objetivo el reconocimiento de una pieza mediante las imágenes tomadas por una cámara (identificación de qué tipo de pieza es, su posición y orientación) y su posterior agarre y clasificación utilizando el robot de 5 grados de libertad Mitsubishi RM501.

Las piezas son de tres tipos diferentes: con forma de U, T o L. Todas tienen una altura de 50 [mm], son de tamaño similar y color negro, para lograr diferenciarlas del fondo.

El diagrama de flujo siguiente muestra cuáles son los pasos a seguir en el proceso:

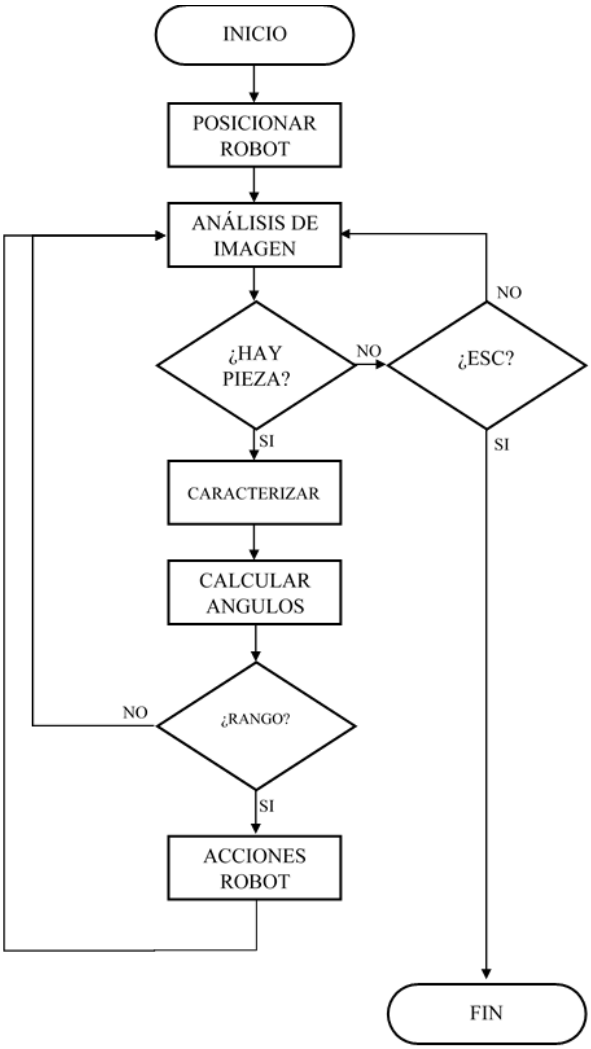

Figura 1 - Diagrama de flujo que describe el proceso de identificación, agarre y clasificación de las piezas.

El posicionamiento del robot consiste en dos movimientos: en primer lugar, es llevado hasta su posición *NEST*, según lo definido en el Manual de Usuario, en la cual todas las articulaciones llegan a hacer contacto con sus finales de carrera. Luego de esto, se comanda para que tome una posición *HOME*, la cual fue precargada en el driver y será donde esperará la orden de tomar la pieza que sea colocada en el área de trabajo.

El sistema estará continuamente analizando el área de trabajo, a la espera de una pieza (siempre que el área de trabajo haya sido correctamente identificada). En el momento en que detecta que algo fue colocado, se encarga de caracterizar qué tipo de pieza es, cuál es la posición a la cual debe comandarse la pinza y con qué orientación. En caso de no reconocer la pieza colocada, se mostrará un mensaje de error de identificación.

Luego, se evalúan los ángulos que deben ser comandados a cada articulación del robot, haciendo uso de la cinética inversa calculada para el robot. En caso de que se encuentre dentro del alcance, son efectivamente comandados al robot; de lo contrario, retornará a la posición de análisis de imagen, donde esperará que la pieza sea colocada en un lugar alcanzable.

Las acciones del robot consisten en ir hasta la posición donde fue detectada la pieza (con la pinza orientada de modo tal de poder tomarla), aproximarse a ésta lentamente, tomarla y luego depositarla en una posición preestablecida, dependiendo de qué tipo de pieza sea.

Para finalizar totalmente el programa, será necesario presionar la tecla ESC durante la ejecución del análisis de imagen.

2.1. Visión artificial y análisis de imágenes

Para la visión artificial, en cuanto al hardware, se utilizó una Webcam Genius colocada en un soporte de modo tal que pueda ser visualizada toda la zona de trabajo del robot. La programación de las tareas necesarias para el reconocimiento fue realizada en lenguaje Python, utilizando principalmente funciones de la biblioteca de visión artificial OpenCV.

2.1.1. Tratamiento de la imagen

En primer lugar, se pasa la imagen a escala de grises y se aplica un filtro de desenfoque gaussiano, el cual mezcla ligeramente los colores de los píxeles vecinos generando así un efecto de suavizado en la imagen. Esto será útil a la hora de realizar la detección de bordes, que será explicada más adelante.

Luego, se filtra la imagen de modo tal que quede en blanco y negro (seleccionando un valor umbral) con el fin de poder distinguir el fondo de los elementos de interés. Se intentó realizar filtros que se adapten al nivel de luz o utilizar los disponibles en la librería OpenCV, pero se descartó la opción dado que se obtenían peores resultados que con un valor umbral fijo.

2.1.2. Detección de contornos dentro de la imagen

Una vez obtenida la imagen filtrada, se procede a la detección de los contornos que se encuentren dentro de la misma utilizando la función *contours()* de OpenCV. Esta función retorna una lista con todos los contornos hallados, la cual es posteriormente ordenada teniendo en cuenta el área de los elementos, de menor a mayor. El objetivo de esto es diferenciar tres contornos de interés: dos puntos de diferente tamaño (los cuales serán la referencia para la identificación del área de trabajo, que será explicado más adelante) y el correspondiente a la pieza a identificar.

Estos contornos serán ampliamente utilizados a lo largo del proyecto, dado que indican la presencia o no de algún elemento dentro del área de trabajo.

2.1.3. Centroide de un contorno

Se desarrolló una función (denominada *CentroideObjeto()*) que será la encargada de encontrar las coordenadas del centroide de cualquier contorno que se le pase como argumento, respecto del sistema coordenado fijo a la cámara. Utiliza los momentos de inercia de primer orden del contorno, obtenidos mediante la función *Moments()* y calcula el punto de la siguiente forma:

$$C_x = \frac{M_{10}}{M_{00}} \qquad C_y = \frac{M_{01}}{M_{00}}$$

Será utilizada principalmente para encontrar la referencia del área de trabajo y también para la ubicación del centroide del objeto a identificar. Cabe destacar que todos los puntos devueltos por esta función son en unidades de píxeles. Posteriormente, deberán ser transformados a milímetros y en coordenadas respecto del sistema asociado al robot.

2.1.4. Escalado de dimensiones

Se implementó una función (denominada *Pixel2Milim()*) con el fin de transformar los píxeles medidos por la cámara a milímetros sobre la mesa de trabajo. Para ello se utiliza uno de los círculos de referencia (de diámetro real conocido, en [mm]) y se obtiene su diámetro en píxeles (utilizando la función *minEnclosingCircle()* de OpenCV y el contorno de la lista mencionada antes correspondiente a ese círculo). Este valor en pixeles es utilizado, junto con el diámetro en mm, para obtener escala que transforma cualquier dimensión de píxeles a mm, la cual será constante ya que la altura de la cámara se mantendrá fija.

2.1.5. Identificación del área de trabajo y transformaciones de coordenadas

La figura siguiente muestra los sistemas coordenados que deberán tenerse en cuenta para la identificación del área de trabajo, así como también las transformaciones que deberán implementarse para relacionarlos:

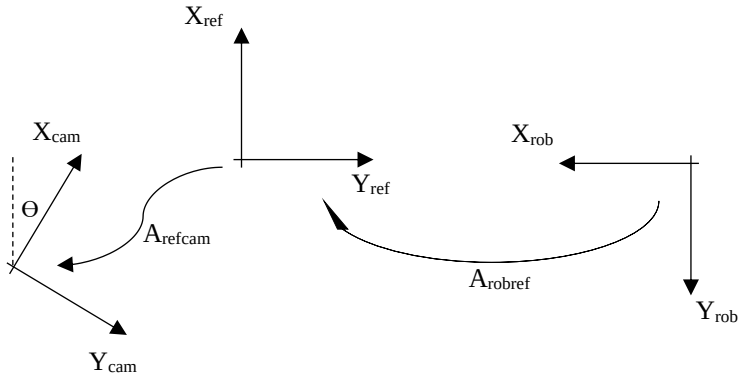

Figura 2 - Sistemas coordenados a tener en cuenta

Xref, Yref: Sistema de referencia fijo al área de trabajo.
Xcam, Ycam: Sistema de coordenadas de la cámara.
Xrob, Yrob: Sistema coordenado fijo a la base del robot

La identificación del área de trabajo se realizará mediante la detección de dos puntos de diferente tamaño sobre la mesa, denominados P1 y P2. Entre estos dos puntos, se generará el eje X_{ref}, fijo al área de trabajo; el eje Y_{ref} se ubicará como se muestra en la Figura 2, en igual sentido que los ejes fijos a la cámara cuando el ángulo de rotación de esta sea nulo.

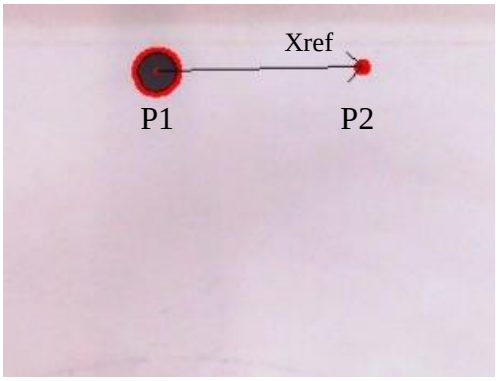

Figura 3 - Generación del eje Xref a partir de la detección de dos puntos de distinto tamaño sobre el área de trabajo

Considerando que la cámara puede ser rotada y trasladada respecto del sistema coordenado fijo al área de trabajo, se implementó la transformación homogénea que relaciona estos sistemas de referencia. Entonces, las coordenadas de un punto respecto del sistema de la cámara son transformadas al sistema de referencia de la mesa de la siguiente manera:

$$P_{Ref} = A_{RefCam} \cdot P_{Cam} = A_{CamRef}{}^{-1} \cdot P_{Cam} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & p1x \\ \sin\theta & \cos\theta & 0 & p1y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \cdot P_{Cam}$$

donde θ es el ángulo los ejes X de la cámara y del área de trabajo, calculado de la siguiente forma:

$$\theta = \operatorname{acos}\left(\frac{(p_{2x} - p_{1x})}{\sqrt{(p_{2x} - p_{1x})^2 + (p_{2y} - p_{1y})^2}} \right)$$

P1x y P1y y P2x y P2y corresponden a las coordenadas del punto 1 y 2 respecto del sistema coordenado fijo a la cámara.

Por otro lado, se obtuvo la transformación entre el sistema de referencia fijo a la mesa y el sistema fijo a la base del robot, la cual consiste en rotaciones sucesivas de 90º alrededor de los diferentes ejes:

$$P_{Rob} = A_{RobRef} \cdot P_{Ref} = \begin{pmatrix} 0 & -1 & 0 & Rxrob \\ -1 & 0 & 0 & Ryrob \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot P_{Ref}$$

Rxrob y Ryrob son las coordenadas del origen del sistema fijo al área de trabajo respecto del sistema fijo a la base del robot, las cuales corresponden al punto de referencia de mayor diámetro, tal como se explicó antes.

Finalmente, utilizando lo anterior, se obtienen las coordenadas de cualquier punto que detecte la cámara, respecto del sistema coordenado fijo a la base del robot, es decir:

$$P_{Rob} = A_{RobRef} \cdot A_{RefCam} \cdot P_{Cam}$$

Todas las transformaciones anteriormente mencionadas fueron implementadas en la función denominada *XYCamara2Robot()*. Dentro de esta función, también se realizan las conversiones pixel-mm necesarias, utilizando la función desarrollada para tal fin.

Cabe mencionar en este punto que todas las informaciones relacionadas tanto a la identificación del área de trabajo como a la detección de piezas serán mostradas en el extremo superior izquierdo de la imagen, permitiendo así la interacción con el usuario.

2.1.6. Identificación de la pieza

El reconocimiento y clasificación de las piezas fue realizado utilizando su área, medida en unidades de [px^2].

Con el fin de cargar la información sobre los elementos a identificar dentro del programa, se realizaron ensayos colocando las diferentes piezas que el robot deberá tomar en múltiples posiciones y orientaciones dentro del alcance de la cámara. Se registró el área detectada en cada caso, utilizando el programa desarrollado para tal fin, cuyo funcionamiento se explica en 2.1.8.

Luego, el valor de área promedio y los valores máximos y mínimos resultantes de los ensayos anteriores son cargados en la función *IdentificarObjeto()*, la cual realiza la comparación del área de la pieza presente en la mesa con los datos precargados, retornando T, U o L, según el resultado de la comparación.

2.1.7. Localización de la pieza

Para lograr tomar la pieza, es necesario conocer el punto de agarre y orientación respecto del sistema coordenado ubicado en la base del robot para que este la pueda manipular correctamente.

Se confeccionó un sistema coordenado solidario al objeto, con origen en el centroide y sus ejes paralelos a alguno de los lados de la pieza. Luego, se buscó la transformación de este sistema hacia el sistema de la cámara, y luego, mediante la función *XYCamara2Robot()* se logra transformar el punto al sistema ubicado en la base del robot.

Con este objetivo, en primer lugar, se calculó el centroide de la pieza con respecto a los ejes fijos a la cámara (*cx*, *cy*), utilizando la función *CentroideObjeto()* descripta anteriormente.

Mediante la función *minAreaRect()* de *OpenCv* se puede encontrar el rectángulo de área mínima que encierra la pieza. Con el centro de este rectángulo y el centroide de la pieza se confeccionó un vector solidario a la pieza (vector Cg-Crec). Este vector es paralelo a los lados de la pieza para algunos casos (U y T), pero no lo es para la L, por lo que en esta última se tuvo que buscar el ángulo para lograr un vector paralelo al lado.

En la imagen siguiente se puede observar el rectángulo que encierra la pieza y el punto rojo que indica el centroide de la pieza. El punto verde es el punto de agarre, más adelante se explicará cómo fue obtenido. El centro del rectángulo no se encuentra dibujado, pero se puede inferir su posición.

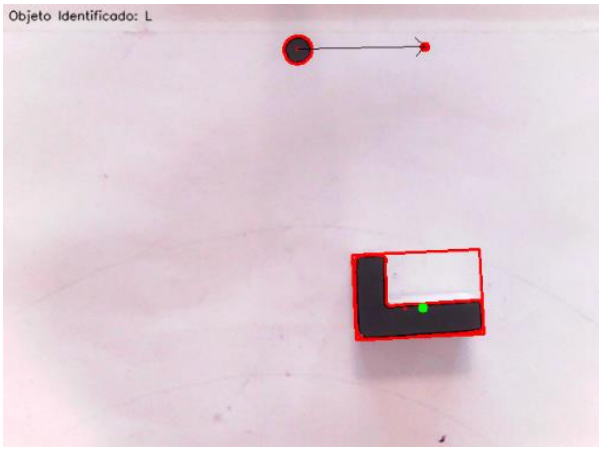

Figura 4. Pieza con rectángulo de área mínima para calcular su localización

Para armar la matriz de transformación de este sistema al sistema de la cámara ya se conoce la distancia que se deben trasladar los sistemas, los valores de *cx*, *cy*. Con la función *atan2()* de la librería *math* y las componentes del centroide y el centro de rectángulo se puede calcular el ángulo de rotación entre los ejes X de ambos sistemas denominado θ.

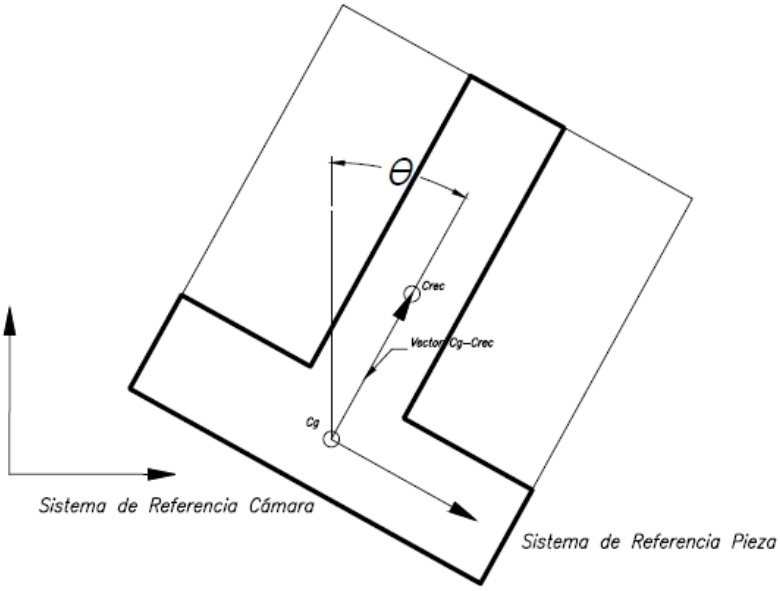

Figura 5 - Sistema coordenado solidario a la pieza y ángulo a rotar

Recordando que el vector Cg-Crec no es paralelo a ningún lado del objeto con forma de L, cuando se calcula el ángulo entre ejes se debe conocer la cara que apoya en la cancha (se tienen dos opciones distintas, a diferencia de las demás piezas). Dependiendo de esto, se debe restar o sumar 45[°]. Las siguientes figuras muestran la pieza en las dos situaciones posibles.

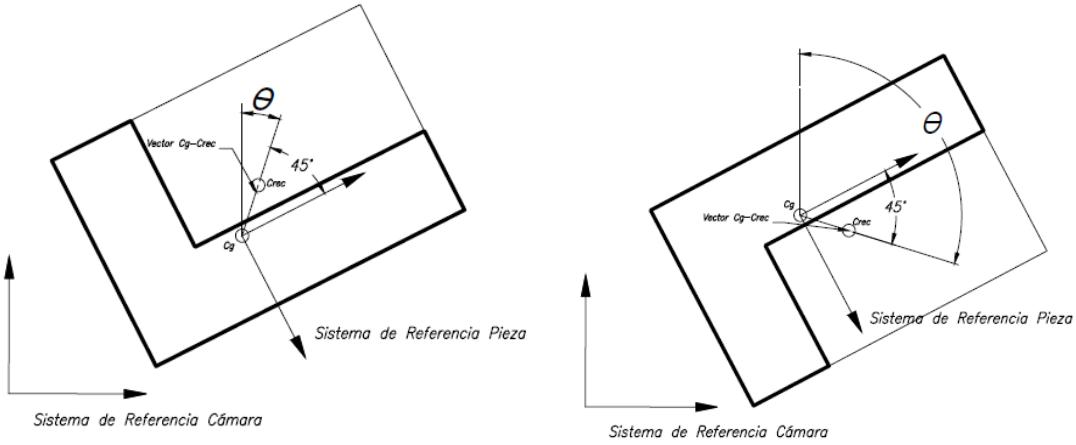

Figura 6 - Posibles posiciones de la pieza tipo L

Para solucionar este problema, se toma el valor de tita sin restar ángulo, y se toman 3 puntos respecto del sistema de referencia de la pieza. Estos 3 puntos se eligen de manera tal que, si la pieza se coloca de una manera o invertida, los pixeles de estos puntos cambien de negro a blanco. En la figura siguiente se muestran en azul los 3 puntos de prueba utilizados.

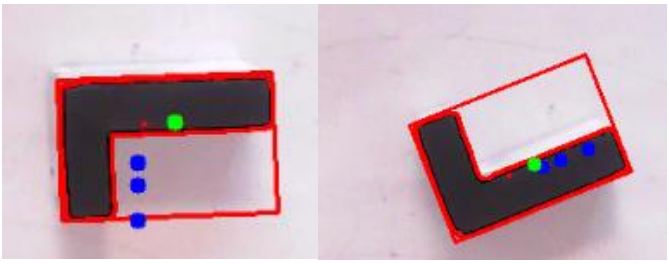

Figura 7 - Puntos de prueba para conocer la posición de la pieza tipo L

Una vez que se definió la matriz de transformación del sistema de referencia fijo a la pieza al sistema de la cámara, se pueden definir el punto de agarre de la pieza y la orientación de la pieza.

El punto de agarre se define eligiendo un punto que sea de fácil acceso para la pinza del robot. Se eligió cerca del centroide de la pieza para evitar que esta se mueva al levantarla. En las figuras anteriores, el punto de agarre se encuentra dibujado con color verde.

Se definieron los puntos de agarre para cada figura respecto del sistema referencia fijo a la pieza, luego se aplicaron las transformaciones necesarias y se obtuvo el mismo punto, pero respecto al sistema fijo al robot.

La orientación de la pieza ($\theta ROLL$), se define utilizando el ángulo de rotación θ definido anteriormente y el ángulo que forma el centroide de la pieza y el sistema de referencia del robot (Tita Pieza-Robot, θPR).

El ángulo $\theta ROLL$ es relativo al robot, esto quiere decir que, por su forma constructiva, el valor cero de este ángulo está sobre la línea que forman el origen del sistema de referencia del robot y el punto de agarre de la pieza (se usó el centroide de la pieza, para simplificar los cálculos).

Además de esto, para las piezas L y T, el eje de orientación es el eje X, se deseó que la pinza tome estas piezas sobre extremo más largo, para la U, el eje de orientación es el eje Y, por eso se debe agregar una rotación de 90 [°].

Por lo que tenemos:

$$\theta ROLL = -\theta - \theta PR \qquad (Para\ T\ y\ L)$$
$$\theta ROLL = -\theta - \theta PR + 90° \qquad (Para\ U)$$

Las figuras siguientes ayudan a comprender como se calcula $\theta ROLL$. Se identifica con un signo + el sentido de crecimiento de cada ángulo, para facilitar la comprensión.

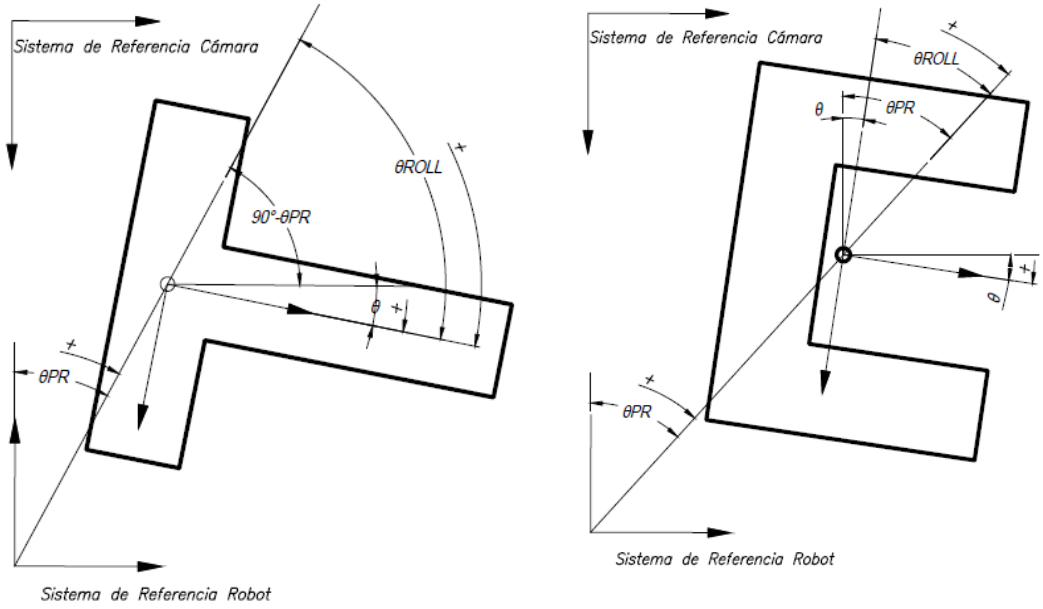

Figura 8 - Cálculo del ángulo de roll

Una vez definido el punto de agarre y la orientación, se pasan estos parámetros al robot y se comenzó a comandar el mismo.

2.1.8. Programa para la carga de piezas

Con el fin de hacer más dinámica la carga de los valores de las áreas que caracterizan a cada pieza, se diseñó un programa cuyo funcionamiento consiste en la toma de un número determinado de muestras del área de una pieza (definido por el usuario), colocando ésta en múltiples posiciones y orientaciones dentro del alcance de la cámara. La toma de varias muestras es necesaria para poder hacer estadística sobre la información brindada por el sensor, conocer sus variaciones y limitaciones.

El conjunto de datos recolectado es almacenado en un archivo .txt para disponibilidad del usuario y en pantalla se muestran los datos de interés: área promedio y mínima y máxima. Estos datos son los que posteriormente son cargados al programa principal, para realizar la clasificación de las piezas.

2.1.9. Programa para trabajar offline

Se implementó un programa que permitirá realizar todo el proceso (desde la identificación de la pieza hasta el comando al robot) de manera offline, prescindiendo del hardware. El objetivo es obtener los ángulos a comandar al robot, pero a partir de una imagen almacenada en la computadora (y no obtenida en tiempo real con la cámara).

Para ello, será necesario que la imagen posea los puntos P1 y P2 de referencia explicados antes, de modo tal de poder realizar la calibración y, a partir de ello, obtener todos los parámetros.

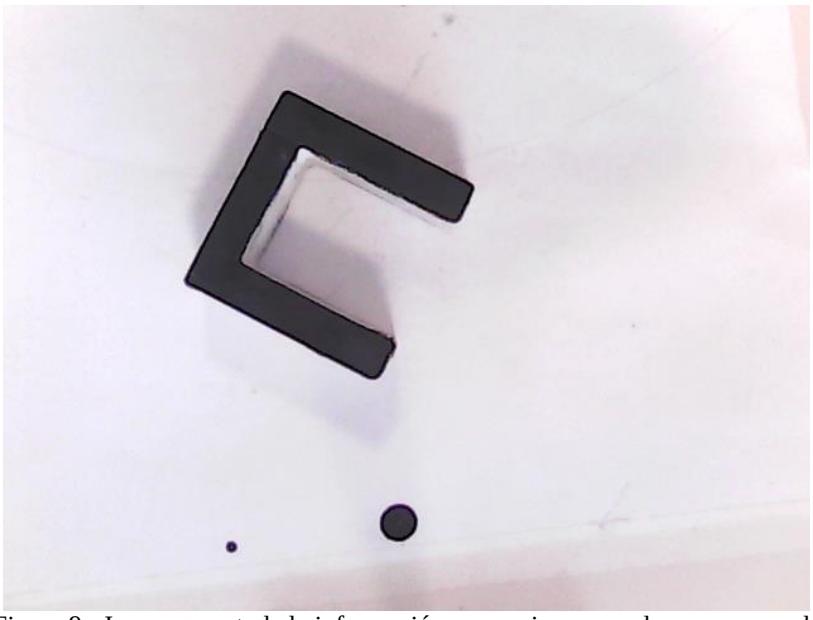

Figura 9 - Imagen con toda la información necesaria para poder ser procesada

2.2. Comando del manipulador

Con el punto de agarre, la orientación y el tipo de pieza, se procede a manejar el robot mediante comandos propios del mismo, enviados por comunicación serie.

Para cada movimiento se confeccionó un bloque de funciones donde cada una posee una utilidad bien definida.

La primer función se denominó *CalcularAngulos()*, la cual recibe la posición deseada y devuelve el ángulo que se debe mover cada articulación para llegar al punto deseado. Está conformada básicamente por las ecuaciones inversas del manipulador.

En caso de que ocurra un error en el cálculo de los ángulos se devuelve una señal de error.

La segunda función, *ErrorRango()*, verifica que los ángulos devueltos por la función anterior estén dentro del rango de trabajo del manipulador. De no ser así, retorna el vector de errores con un valor distinto de cero en la posición correspondiente al ángulo erróneo.

Si ninguna de las dos funciones anteriores arrojó error, se llama a la función *CargarAngulo()*, que compara el valor actual de los ángulos del manipulador con los

ángulos objetivo, y envía a mover el robot, mediante el comando *MI*, el valor de ángulo restante.

El comando *MI* propio del manipulador recibe un numero de pulsos equivalentes al ángulo que se pretende mover. Con la ayuda del manual de usuario, se conoce el rango en grados de cada articulación y el valor máximo de pulsos que puede moverse, con esta información se encontró la relación entre valor de ángulo y cantidad de pulsos.

Debido a la forma constructiva del manipulador, se debe mover las últimas dos articulaciones de forma separada. Por lo que, cuando es deseado mover ambas, primero se mueve una de ellas y luego, la siguiente.

La función *CargarAngulo()* además, devuelve un vector con los valores de desplazamiento que posee cada articulación luego de realizar el movimiento.

Para abrir o cerrar la pinza, se crearon dos funciones, *AbrirGarra(), CerrarGarra(),* en éstas se utilizan los comandos *GO* y *GC* propios del manipulador.

Mediante la función *Velocidad(),* se seteó la velocidad deseada del manipulador, utilizando para esto el comando *SP*.

Para clasificar las piezas se creó la función *DepositarPieza*(). En esta se definen 3 posiciones finales distintas, una para cada tipo de pieza. Luego, se utilizan los bloques de funciones mencionados anteriormente para realizar los movimientos necesarios.

A la hora de finalizar el programa, se implementó la función *GuardarRobot(),* que se encarga de mandar el robot al valor de NEST predeterminado. Se utilizó para esto, el comando NT.

3. Conclusiones

Se puede concluir que la ejecución de este proyecto ha sido un aprendizaje constante, tanto en lo académico como en lo grupal. Se adquirieron muchos nuevos conocimientos, se investigó sobre lo necesario y se solucionaron los problemas que se presentaron trabajando colectivamente.

Se aprendió un nuevo lenguaje de programación, ya que ninguno de los integrantes del grupo conocía Python. Se presentaron las dificultades propias de un proceso como este, pero pudieron ser superadas sin grandes inconvenientes.

Se tuvieron numerosas complicaciones con errores inesperados y no repetitivos del robot, los cuales se cree están relacionados con el transformador utilizado para alimentar el dispositivo, el cual no es lo suficientemente potente. Se realizó el cambio por uno apropiado y la frecuencia de estos errores se redujo notablemente.

Dado que el robot no posee comandos que informen su posición, todas las instrucciones debieron ser enviadas controlando tiempos. Esto hizo que los movimientos sean pausados y un poco tediosos, dado que, de no ser así, se corre el riesgo de incurrir en errores por envío de comandos cuando una maniobra no ha sido aún finalizada.

4. Bibliografía

[1] https://opencv.org/
[2] https://www.python.org/