FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

67.61 - Fundamentos Matemáticos de la visión Robótica

Trabajo Práctico Final

Eliana Gamarra - 100016

Introducción

A partir del material didáctico brindado por la cátedra, se busca determinar la posición de los bloques ubicados sobre un tablero de trabajo para que un brazo robótico pueda tomar dichas piezas.

Por un lado tenemos nuestra terna de trabajo, donde se ubica el robot y la cámara. La cámara le indica al robot donde están ciertas piezas para que el robot las pueda tomar.

La aplicación de visión que vamos a programar tiene que detectar los objetos de las imágenes tomada por la cámara, determinar la coordenada de ese objeto en su imagen y después hacer una transformación de esos valores para saber cuales es la coordenada de ese objeto para la terna de trabajo desde la ubicación del robot.

Para eso se seguirán una serie de pasos desarrollados a continuación.

1. Calibración de parámetros intrínsecos.

1.1 Algoritmo de calibracion

Se tiene un conjunto de imágenes de un tablero de ajedrez de 28mm.
Mediante la utilización de OpenCV podremos utilizar la función calibrateCamara basada en el método de Zhang.

A partir del uso de los puntos 3D que se denominan puntos de objeto y los puntos de imagen 2D se denominan puntos de imagen, el método nombrado retorna:

- la matriz de parámetros intrínsecos,
- el vector de rotation,
- el vector de distorsión
- el vector de desplazamiento

Esto significa que tenemos toda la información sobre la cámara necesaria para determinar una relación precisa entre un punto 3D en el mundo real y su proyección 2D (píxel) en la imagen capturada por esa cámara calibrada.
La ecuación que relaciona el punto 3D (X_w, Y_w, Z_w) en coordenadas con su proyección (u, v) en las coordenadas de la imagen es la siguiente:

$$s\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = [K]\,[R_k|t_k]\begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Donde K es la matriz intrínseca que contiene los parámetros intrínsecos y la matriz extrínseca $[R_k \mid t_k]$ que es una combinación de matriz de rotación de 3×3 R y un 3×1 vector de traslación. K triangular superior de la siguiente manera:

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- f_x, f_y son las distancias focales x e y que suelen ser las mismas.
- c_x, c_y son las coordenadas x e y del centro óptico en el plano de la imagen. Usar el centro de la imagen suele ser una buena aproximación.
- γ es el sesgo entre los ejes y suele ser 0.

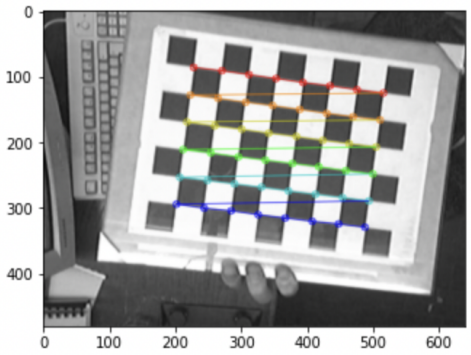
El skew, s, el cual tiene que ver con que el sensor no se encuentre perfectamente alineado al plano focal, pero en este caso será nulo.

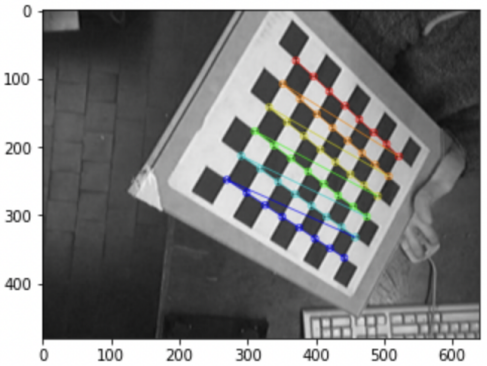
1.2 Desarrollo del algoritmo

En el archivo **1-calibracion** en la sección 1, se definen 2 funciones:

- table_size: crea la matriz de punto objetos a partir de las dimensiones del tablero de ajedrez (8,6) y luego los multiplica por la dimensiones del tablero, en este caso de 28mm
- calib donde:
    - Se define un criterio de corte para el proceso iterativo de refinamiento de esquinas y una variable epsilon para saber si iterar maxCount veces o si las esquinas se mueven menos de epsilon.
    - Luego se itera sobre el set de imágenes 1 ingresado donde se encuentran las imágenes del tablero de ajedrez en distintas posiciones. Se transforma cada una de estas imágenes a una escala de grises y con la función provista por openCV findChessboardCorners se encuentran los diferentes puntos del tablero

Un detalle a tener en cuenta al visualizar las imágenes es que no todas encontraron el mismo patrón en la calibración al estar rotados, se puede reconocer al ver el borde blanco lateral. Ejemplo: La primera imagen corresponde a img_cal_set1/img_cal3.png donde el borde blanco es mayor del lado derecho, esto es correcto. En cambio la segunda imagen corresponde a img_cal_set1/img_cal7.png y el borde blanco es mayor del lado izquierdo, por esta razón se decide quitar del análisis las imágenes donde el borde izquierdo banco es mayor debido a la rotación del tablero. Utilizando asi solo las imágenes: 1, 2, 3, 4, 12, 13, 14, 15





Finalmente utilizando la función de openCv calibrateCamara encontramos:

- la matriz de parámetros intrínsecos,
- el vector de rotation,
- el vector de distorsión
- el vector de desplazamiento

```
Camera Matrix =
[[811.31174395   0.         315.73355982]
 [  0.         811.74299018 243.60518088]
 [  0.           0.           1.        ]]

Distortion Coefficients =
```

```
[[0.04997142 0.         0.         0.         0.        ]]
```

2. Calibración extrínseca

En la sección 2. del archivo *1-calibración* utilizando el mismo método de la sección anterior para obtener las coordenadas del mundo real de los puntos 3D usando la imagen img_bloques/imgCalExtr.png de tablero de tamaño 28mm y de esta forma asignar una terna de trabajo (X, Y, Z)

Luego utilizando la función **solvePnP** se estima la posición del objeto utilizando: los punto objeto y punto imágenes obtenidos de la imagen imgCalExtr, y también utilizando la matriz intrínseca de la cámara y los coeficientes de distorsión obtenidos en la primera parte.
De esta forma se obtiene el vector de traslación y la matriz de rotación.

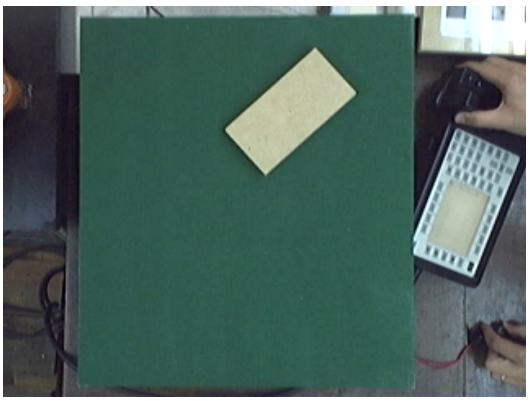
```
Rotation =
[[ 0.99919631 -0.00779203  0.03931953]
 [ 0.00771164  0.99996785  0.00219582]
 [-0.03933537 -0.00189083  0.99922428]]

Translation =
[[-108.62975346]
 [-150.3410905 ]
 [ 703.32230507]]
```

3. Algoritmo de búsqueda de bloques

Para este punto iniciamos con una serie de imágenes de bloques y se encontrarán a partir de un algoritmo los centros y vértices para luego terminar con su transformación a las dimensiones de milímetros orientadas a partir de la terna de trabajo del brazo robótico.

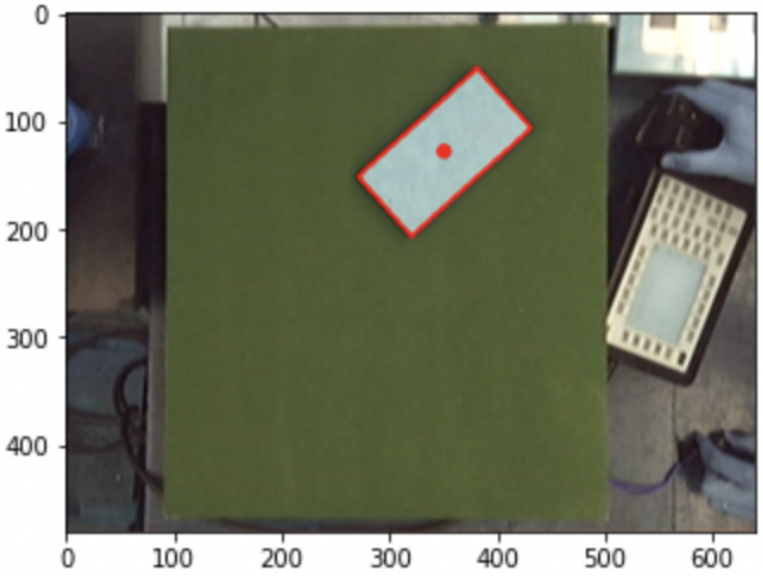
3.1 Búsqueda del bloque

1. Por cada una de las imágenes se realiza una transformación de los colores a la escala BGR de grises
2. Se realiza una binarización con Otzu.
3. Ya que las imágenes de los bloques estan tomadas desde el mismo lugar y cada uno de los bloques se encuentra sobre una mesa de color verde de la siguiente forma:



Para eliminar el ruido de todo lo que se encuentre fuera del tablero lo que se hizo fue hacer un recorte de los pixeles de un tamaño de los ejes x = [20, 500] y de y = [0, 500] tal que solo se encuentran bloques relativos a estas nuevas dimensiones, de esta forma se elimina el ruido externo

4. Se encuentran los contornos de los bloques a partir del offset (0,20) debido a la eliminación de bordes del paso anterior y se elimina toda irregularidad que tenga un área menor a 10000 px
5. Por cada imagen se guardan los datos de centros, vértices, área y orientación
6. Finalmente se dibuja en el bloque los bordes y un punto central solo de carácter ilustrativo.



3.2 Pieza en milímetros

Continuamos desarrollando en el archivo 2-bloques sección 2.3.

Teniendo ya calculados los parámetros intrínsecos y extrínsecos, los coeficientes de distorsión y las coordenadas de los centros de los bloques. Se resuelve la ecuación de a continuación que tiene como resultado el paso de coordenadas en unidades de píxel a unidades del mundo real, milímetros en este caso:

$$H^{-1} * F_{distorsion}^{-1}(K^{-1} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}) = \begin{bmatrix} x_w \\ y_w \\ 0 \\ 1 \end{bmatrix}$$

Dando como resultado, por cada uno de los bloques las coordenadas respecto a la terna de trabajo:

```
1. (139.76941914305286, 50.56909128616893)
2. (94.08573979683331, 74.50923474872253)
3. (193.75628376909057, 69.84356468798276)
4. (-8.323128550225952, 252.86829899641413)
5. (96.26989232029365, 246.32190707649045)
6. (190.19215561203148, 247.50135252134027)
7. (175.58224456531758, 172.83913444027306)
8. (78.6263922471499, 169.16438746579928)
9. (-33.44224960794508, 163.51443095901487)
10.   (-16.791129005006706, 64.81497096555786)
11.   (30.98532876228561, 51.07365585693049)
12.   (68.97470899793278, 51.463168428620484)
13.   (151.60632312171018, 48.00668443527267)
14.   (100.26337404921577, 134.36723040605926)
15.   (161.08403045402258, 213.82363630738715)
```

4. Método de validación del algoritmo

Una propuesta de un algoritmo de validación es la siguiente:

Corroborar la posición de los centros de cada bloque según la terna de trabajo y también los lados de cada bloque ambos medidos en píxeles ya que conozco el tamaño real de los bloques en mm y para estos valores realizar la conversión a mm utilizando el factor de conversión desde mm a pixeles.
Luego utilizando la imagen de calibración extrínseca, medir desde el origen de la terna en milímetros y ubicar el centro del bloque.

5. Algoritmo de medición de bloques

Continuando el análisis en el archivo 2-bloques sección 3.

Para cada uno de los bloques se debe hacer una transformación de las coordenadas de los vértices a valores en píxeles a milímetros utilizando el algoritmo del punto 3.

Luego teniendo el dato dado por enunciado que el tamaño real de los bloques es de altura 130 mm y ancho 65 mm se calcula el error relativo.

Finalmente con los valores obtenidos se grafica en el eje y el error relativo porcentual y en el eje x el número de bloque, la línea azul corresponde al error relativo del ancho y línea naranja error relativo de la altura.