UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

INSTITUTO DE INGENIERÍA Y TECNOLOGÍA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL Y MANUFACTURA,

"Desarrollo de un algoritmo para obtener la posición de una cámara con respecto a ArUcos"

Visión

Francesco García Luna

Alumnos

Martín Iván Martínez Vela-180000

Maximiliano García Carreón-180026

Edgar Irán García Ojeda-180032

Fernando Durán Romero-180079

Allan Daniel González Antonio-180083

Ciudad Juárez, Chihuahua. 6 Abril 2022

Índice general

1. Introducción. **1**
1.1. Antecedentes. 1
1.2. Planteamiento del problema. 1
1.3. Objetivo. 2
1.3.1. Objetivos especificos. 2

2. Marco Teórico. **3**
2.1. Marcador ArUco 3
2.2. OpenCV. 3
2.3. Matriz de rotación. 4
2.4. Matriz de transformación homogénea. 4
2.5. Ángulos de Euler. 4
2.6. Matriz de transformación inversa. 5

3. Metodologia. **6**
3.1. Diagrama de flujo. 7

4. Resultados **8**

5. Conclusion **10**

6. Referencias. **11**

Capítulo 1

Introducción.

En este presente documento se mostrarán los resultados del desarrollo de un algoritmo para apuntar al centro de un ArUco mediante un láser, así como la metodología empleada. El documento se encuentra seccionado por antecedentes, planteamiento del problema, objetivos, marco teórico, metodología, diagrama de flujo, resultados y conclusión.

1.1. Antecedentes.

La visión por computadora es aquel conjunto de herramientas y métodos que permiten obtener, procesar y analizar imágenes del mundo real con la finalidad de que puedan ser tratadas por un ordenador.

Una de las primeras aplicaciones de las imágenes digitales aprecio en la industria de la prensa. Dichas imágenes eran enviadas a través de un cable submarino entre Londres y Nueva york. De ahí los equipos se encargaban de codificar las imágenes para transmitirla y luego se reconstruían al otro extremo. En un inicio los trabajos para mejorar la calidad visual de las imágenes se concentraban principalmente en la selección de los métodos de impresión y la distribución de los niveles de intensidad.

Para el procesamiento de imágenes se ha desarrollado bibliotecas referentes a la visión artificial, la más utilizada y popular hasta la fecha es la biblioteca abierta conocida como OpenCV que significa open computer visión (visión artificial abierta). Es utilizada en la detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, estas son algunos ejemplos de sus aplicaciones. Si bien OpenCV está desarrollado en C++ también está disponible en el lenguaje de programación de Python.

1.2. Planteamiento del problema.

Conociendo las dimensiones de los cuadros que se encuentran en el tablero de ajedrez para calibrar la cámara, se obtienen los parámetros de esta. De esta manera, podemos aplicar un algoritmo para conocer la posición de la cámara con respecto al ArUco.

Sin embargo, en [2] se menciona que puede existir un problema de ambigüedad a la hora de estimar la posición utilizando un sólo ArUco. Este problema se presentó en nuestro caso, ya que variaba mucho la posición.

1.3. Objetivo.

Desarrollar un algoritmo para obtener la posición de una cámara con respecto a un par de ArUcos.

1.3.1. Objetivos especificos.

- Capturar fotografías de tablero de ajedrez con la cámara.
- Desarrollar algoritmo para calibrar la cámara.
- Obtener matriz de calibración de la cámara.
- Obtener posición y orientación del ArUco con respecto a la cámara.
- Obtener posición y orientación de la cámara con respecto a los ArUcos aplicando la matriz homogénea inversa.
- Generar gráfica para representar la posición y orientación de la cámara.
- Validar algoritmo.

Capítulo 2

Marco Teórico.

2.1. Marcador ArUco

Un marcador ArUco es un identificador único compuesto por un borde y una matriz binaria que se utiliza como un sistema de referencia, el cual permite conocer la posición y orientación utilizando la librería de OpenCV. La codificación binaria permite su identificación y la aplicación de técnicas de detección de errores.

El tamaño del marcador determina el tamaño de la matriz interna. Los distintos tipos de arucos contienen matrices de 4x4, 5x5, 6x6, 7x7 y funcionan con un diccionario que son un conjunto de marcadores para una aplicación específica para cada uno de los valores de las matrices.

2.2. OpenCV.

Biblioteca de código abierto para aplicaciones que utilicen visión artificial, es utilizada en varias aplicaciones como control de procesos, sistemas de seguridad con detección de movimiento, reconocimiento de objetos, robótica avanzada, etc.

Esta librería cuenta con una buena eficiencia computacional y un fuerte enfoque en aplicaciones de tiempo real. Opencv contiene también varias funciones utilizadas en distintas áreas de la visión por computadora.

Algunos ejemplos de sus aplicaciones son:

- Identificación de objetos
- Reconocimiento de rostros
- Interfaces de usuario
- Representación 3D.

2.3. Matriz de rotación.

Matriz que representa la rotación en grados del plano en sentido antihorario. La orientación de un punto puede expresarse con los tres ángulos de Euler que corresponden a la rotación respecto a los ejes x, y, z.

La rotación en cada eje puede expresarse como una matriz para cada ángulo, como se muestra a continuación:

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 & 0 \\ \sin(\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Matriz [1], [2] y [3]

2.4. Matriz de transformación homogénea.

Matriz 4x4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro.

Permite representar conjuntamente la posición y orientación y se facilita su uso mediante el algebra matricial, como se muestra a continuación:

$$T = \begin{bmatrix} R_{3X3} & P_{3X1} \\ f_{1X3} & 1X1 \end{bmatrix} = \begin{bmatrix} \textit{matriz de rotacion} & \text{vector de posicion} \\ \textit{transformacion de perspectiva} & \text{escalado} \end{bmatrix}$$

Matriz [4]

2.5. Ángulos de Euler.

Conjunto de tres coordenadas angulares útiles para especificar la orientación de un sistema de referencia de ejes ortogonales móvil respecto a otro sistema de referencia fijo.

En esta representación se realizan rotaciones alrededor de los ejes del sistema solidario al

cuerpo. Primero se rota un ángulo alrededor de Z, después un ángulo alrededor de Y y se rota un ángulo alrededor del eje X.

Estos ángulos se obtienen al realizar rotaciones en los ejes X, Y, Z.

2.6. Matriz de transformación inversa.

Teniendo dos elementos A y B y conociendo el vector de traslación que existe entre ellos podremos definir la matriz de transformación homogénea para obtener la posición de A con respecto a B:

$$T = \begin{bmatrix} R_B^A & T_r \\ 0 & 1 \end{bmatrix}$$

Matriz [5]

Pero para algunos casos particulares es necesario conocer la posición de B con respecto a A. Para este procedimiento se tiene que sacar la inversa de la matriz obtenida:

$$T_A^B = T_B^{A^{-1}}$$

Ecuación [1]

Esta inversa puede expresarse como una matriz de transformación homogénea:

$$T_A^B = \begin{bmatrix} R_A^B & T_{r_A}^B \\ 0 & 1 \end{bmatrix}$$

Matriz [6]

Como se sabe que la inversa de una matriz de rotación es su transpuesta y tomando como referencia el vector inicial de A a B nos queda el siguiente termino para la inversa de la matriz:

$$T_A^B = \begin{bmatrix} R_B^{A^T} & -R_B^A(T_{r_B}^A) \\ 0 & 1 \end{bmatrix}$$

Matriz [7]

Capítulo 3

Metodologia.

Para desarrollo de este proyecto tuvieron que realizar los siguientes pasos.

Primeramente, se calibró la cámara que fue utilizada, para esto de desarrolló un algoritmo de calibración en Python. El código utiliza fotos que son tomadas con la cámara, en las fotos se debe de apreciar un tablero con cuadros en diferentes posiciones y orientaciones, para tener una calibración que sea lo suficientemente precisa, se deben de tomar por lo menos 30 fotos y deben ser guardadas en la misma carpeta en donde se va a guardar el código de programación.

En el algoritmo de calibración primeramente se definen los cuadros que hay en los ejes X y Y, también se definen el tamaño de los cuadros, después se define el nombre de la carpeta de donde se van a tomar las imágenes que van a ser calibradas y la extensión con la que fueron guardadas las imágenes, posteriormente cada imagen se somete a un proceso en donde se convierte de RGB a espacio de grises, esto servirá para obtener las esquinas de los cuadros en las imágenes. Después con ayuda del comando de "cv2.calibrateCamera", se obtienen los parámetros que conformarán la matriz de calibración de la cámara y serán exportados a un archivo con extensión.p.

Una vez que fue calibrada la cámara y se obtuvo la matriz de calibración, se utilizará para desarrollar el algoritmo principal. Lo primero que realiza el algoritmo es una comunicación con la camara, para obtener imagen del medio en donde opera el sistema. Después se define el directorio de ArUcos para que puedan ser detectados por la cámara y se defina el centro de ellos. Después se obtiene la posición que tienen los ArUcos con respecto a la cámara, estas coordenadas serán utilizadas para obtener la rotación y traslación de los ArUcos, y después de obtener la matriz de transformación homogénea, esta matriz nos dará la posición que tiene el ArUco con respecto a la cámara en este caso cuando la cámara se encuentre en una posición fija el algoritmo es capaz de definir en que eje se desplaza el aruco o si sufre alguna rotación en algún ángulo, para poder hacer esto más gráfico y poder interpretar todo lo que está ocurriendo con la posición, en la última parte del código se grafica en 3D unos ejes que interpretan las rotación y traslación en casi tiempo real.

3.1. Diagrama de flujo.

En la Figura 1 se muestra el diagrama de flujo que se utilizó para desarrollar nuestro algoritmo:

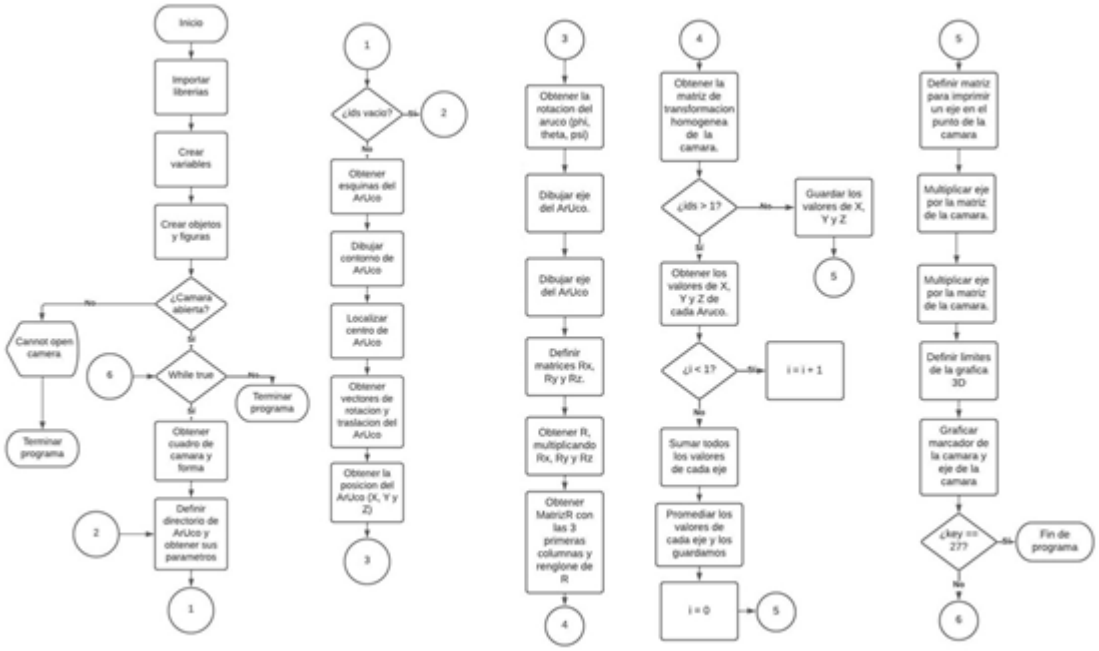

Fig. 1 Diagrama de flujo

Capítulo 4

Resultados

Como resultado obtuvimos un programa donde obtenemos la posición de la cámara a partir de la matriz de transformación homogénea del ArUco. Cuando colocamos un ArUco y la cámara lo identifica, se grafica en un entorno 3D la posición de la cámara con respecto al ArUco y cuando se detecta más de uno, promediamos la posición que arroja con cada ArUco para así obtener datos más precisos.

Aqui podemos ver como funciona con un solo ArUco

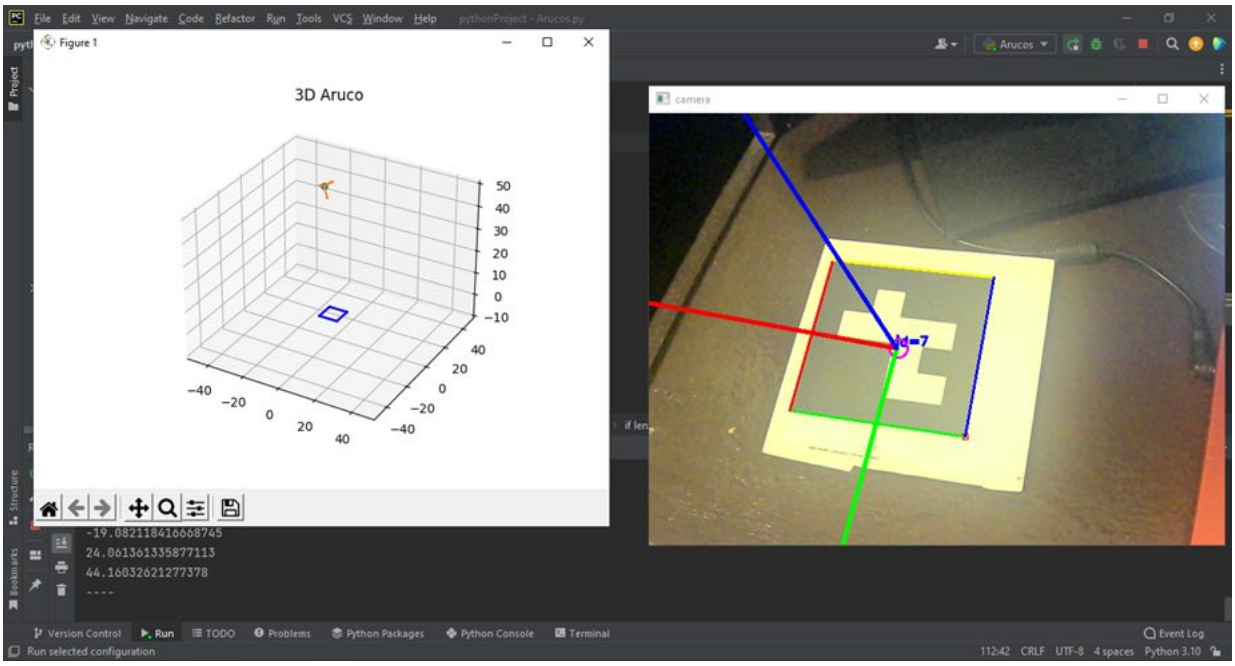

Fig. 2 Grafica con un ArUco.

Y aqui podmeos ver como funciona con los dos ArUcos, promediando la posicion.

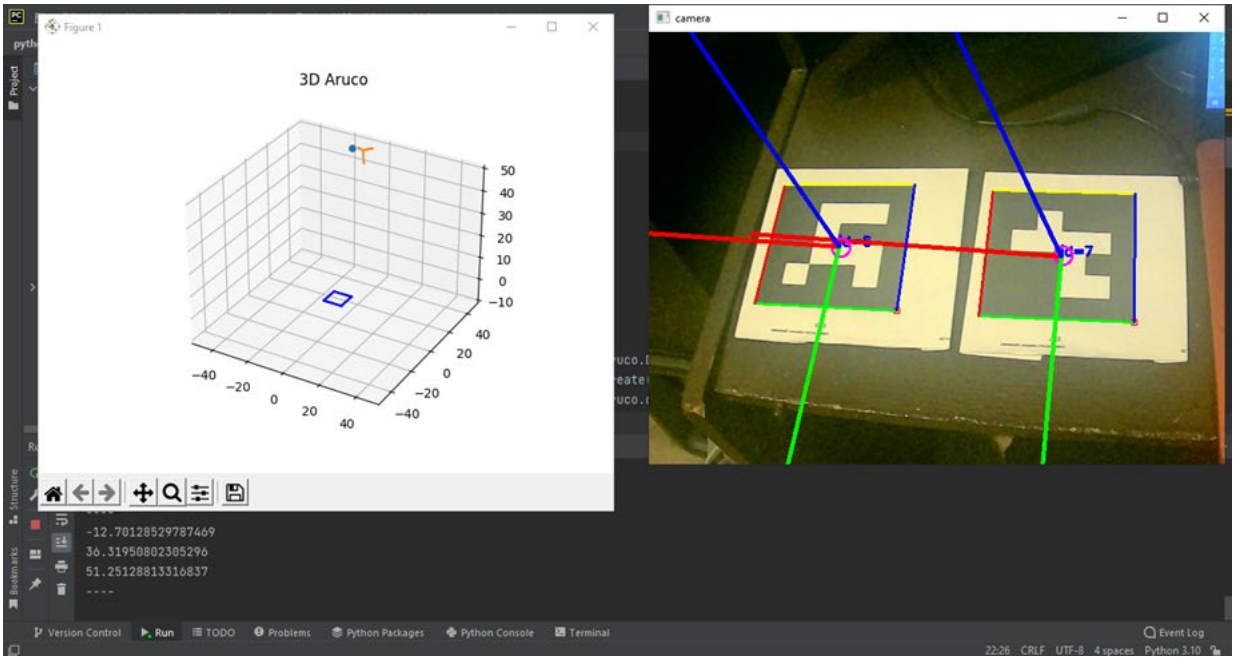

Fig. 3 Demostración promedio.

Capítulo 5

Conclusion

El sistema cumple con la tarea que fue definida, pero la precisión con la que trabaja el sistema depende de muchos factores, como la iluminación, la calidad de la cámara y la posición en donde se encuentra el ArUco. Es por ello que un trabajo a futuro, se buscaría de trabajar en los factores que afectan al sistema para que ofrezca una mejor precisión.

Capítulo 6

Referencias.

[1] About-OpenCV" . OpenCV (8 Oct 2021). [online], disponible: https://opencv.org/about/

[2] R. Jiménez, "Sistema de seguimiento de objetos usando OpenCv, ArUco y Filtro de Kalman extendido", tesis, Universidad de Sevilla, SE, España, 2018.

[3] J. N. A. Rivero, «Robot Dirigido Por Voz,» Universidad de las Américas Puebla, Puebla, 2007.

[4] E. A. Prieto y P. A. G. Quintero, «Diseño e implementación de un robot móvil autónomo con detección de color,» Universidad De San Buenaventura, Bogotá, 2006.

[5] G. Viera-Maza, «Procesamiento de imágenes usando OpenCV aplicado en Raspberry PI para la clasificación del cacao,» Universidad De Piura, Piura, 2017.

[6] USON, «USON,» 2010. [En línea]. Available: http://tesis.uson.mx/digital/tesis/docs/22170/Capitulo2.pdf. [Último acceso: 6 Abril 2022].

[7] OpenCV, «OpenCV,» 6 Abril 2022. [En línea]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial$_a ruco_d etection.html.[Últimoacceso : 6Abril2022]$.

[8] A. E. Benavides, «Sistema de control dinámico del manipulador de un robot industrial en empresas troqueladoras de 4 grados de libertad,» Universidad Rafael Belloso Chacín, Maracaibo, 2013.

[9] M. E. P. Chin, «Desarrollo de un sistema de visión computacional para el vuelo autónomo de un Vehículo Aéreo No Tripulado,» Universidad Autónoma de Yucatán, Mérida, 2019.