

Calibración de Cámara

Callo Aguilar Alejandra Cristina
alejandra.callo@ucsp.edu.pe
UCSP

Resumen—

El presente trabajo trata sobre calibración de cámara para la detección de círculos desde archivos de video. La detección de los círculos está sujeta a las variaciones de brillos y contraste en los frames, los cambios de posición de los círculos y en la mayoría de casos el seguimiento del patrón sería una dificultada a superar

I. INTRODUCCIÓN

El presente trabajo será dividido en etapas, la primera será de DETECCIÓN, luego del SEGUIMIENTO DEL PATRÓN.

En la primera etapa se trabajara la eliminación de ruido y el mejor procesamiento de la imagen en las escenas de los diferentes videos.

En la segunda etapa se hará el seguimiento del patrón en tiempo real. Cada etapa se detallara más adelante.

II. DETECCIÓN DE CÍRCULOS

A. Configuración de PS3-eye camera para visual studio

Se configuro para la detección de tiempo real la cámara PS3

- Primero: se configura la librería LibUSB ya que se presentaron dificultades con la librería que nos dan para Windows al momento de importar al opencv.
- Segundo se añade el proyecto de la configuración con esta librería, este se adjunta como proyecto de visual.

B. Pre procesamiento de la escena

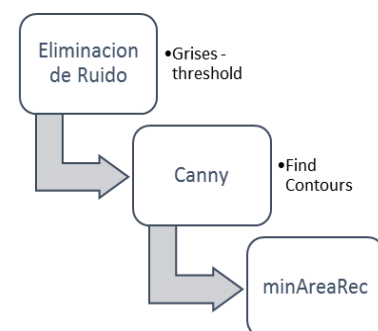


Para obtener la imagen más limpia dentro de la escena se ha de seguir una secuencia de pasos para poder analizarla:

- Primero se cambia la escena a una escala de grises
- Segundo se aplica un umbral en los frames para ir eliminando objetos, se puede usar la función: `cv2.threshold` o `cv2.adaptiveThreshold`.
- Si aún la imagen pierde algunos pixeles se aplica incremento de contraste que eliminara el ruido

C. Detección

- Para la detección de bordes se utiliza la función de *Canny* y *findContours*.
- La función *HoughCircles* () me permite capturar en un vector los círculos dentro de la escena. Este vector contiene la posición y el posible diámetro del círculo encontrado. Con este vector y el posible radio ya tenemos el centro de cada círculo dentro de la escena.

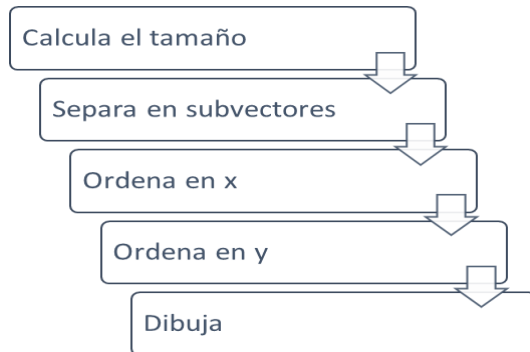


D. Segmentación

Para la segmentación se accede al vector que nos devolvió la función anterior y encuentro la mínima

área (cv: *minAreaRect*) que forman estos centros, luego delimito la escena a la parte donde se encuentran los círculos para encontrar el patrón.

III. ENCONTRAR PATRÓN



Como se obtienen los puntos de los centros de los círculos de manera desordenada es necesario ordenar esta información.

Se utiliza un `vector<vector<Point>>` `v_cir_sa` el cual agrupara los puntos. Sabemos que nuestro patrón tiene 20 círculos por lo que se organizan en 4 filas x 5 columnas.

Ordenaremos primero en base al valor de la posición en X (de manera ascendente) y cada subgrupo en su posición en Y.

IV. DIFICULTADES

básicamente me he encontrado con 3 dificultades:

- La primera: la configuración de la cámara PS3
- La segunda: aun dentro del código estoy usando Houghcircles a pesar de que el procesamiento es pesado, he implementado un código con un algoritmo basado en la distancia euclideana y el radio del círculo pero no he logrado culminarlo, opte por no usar ellipse ya que en las pruebas no conseguía un resultado aceptable
- En los videos que se adjuntan se verá que aun presenta un mal cálculo del patrón ya que la segmentación no está del todo terminada

La principal limitación que he tenido ha sido el tiempo más que de implementación, aunque hay algunas funciones que según la investigación realizada se podrían utilizar pero no he logrado comprenderlas por completo para poder utilizarla (como por ejemplo cuando el Rectángulo de la

segmentación rota aun me muestra errores de calculo).

V. RESULTADOS

En los siguientes gráficos se puede observar los resultados de la primera etapa de este trabajo

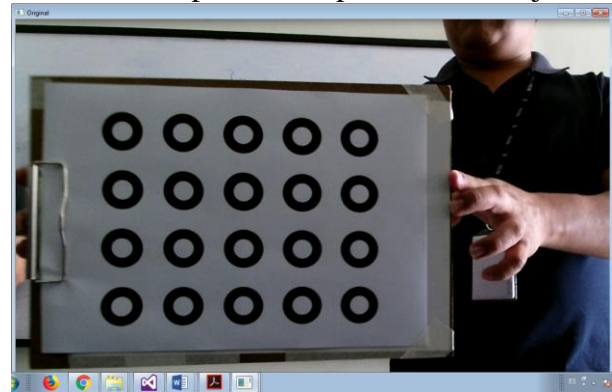


Imagen Original

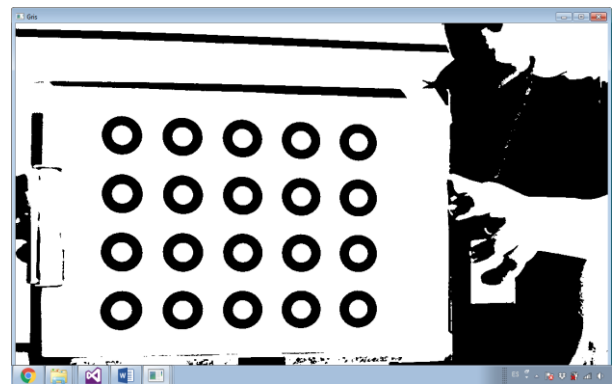
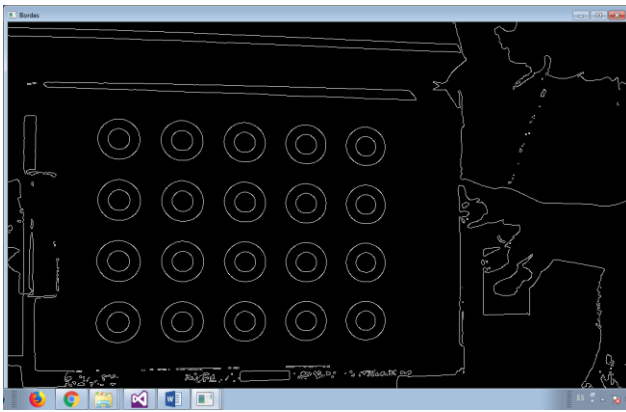


Imagen despues de la conversion a grises y aplicarle umbralizacion



Imagen al aumentar el contraste



Detección de Contornos algoritmo de Canny

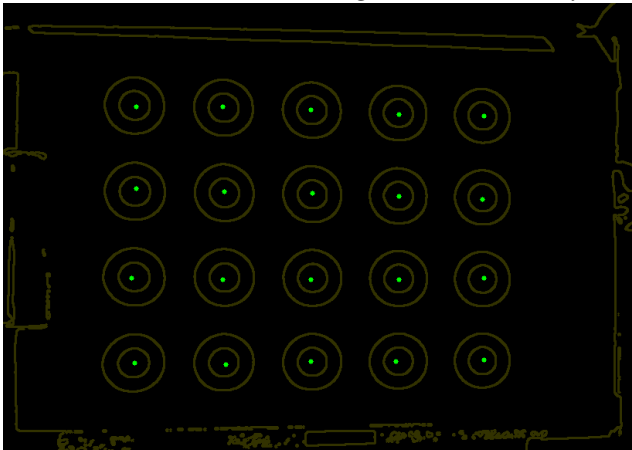
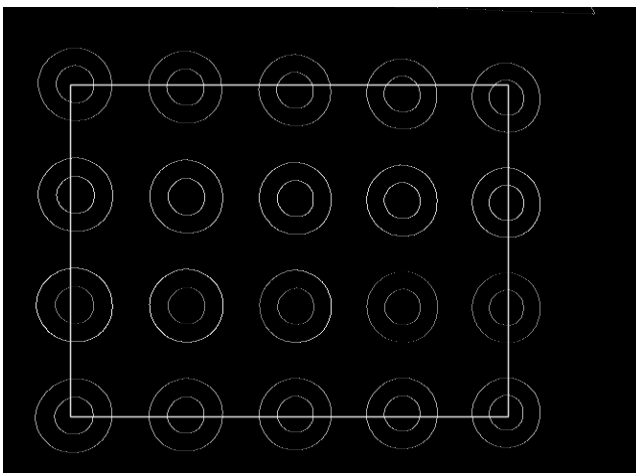
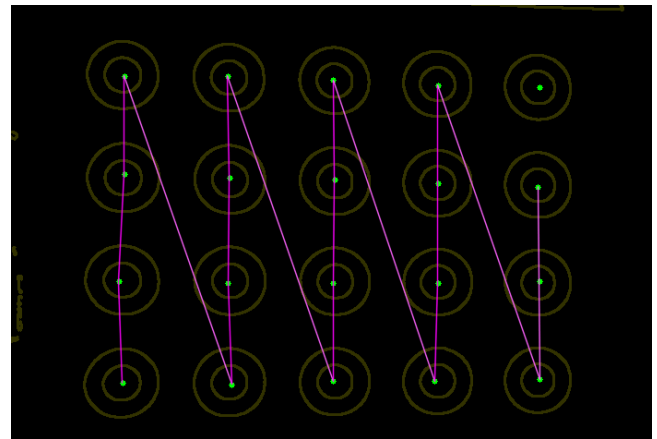


Imagen tras encontrar los centros de los círculos



Selección de la region de Interes



Patrón en zigzag

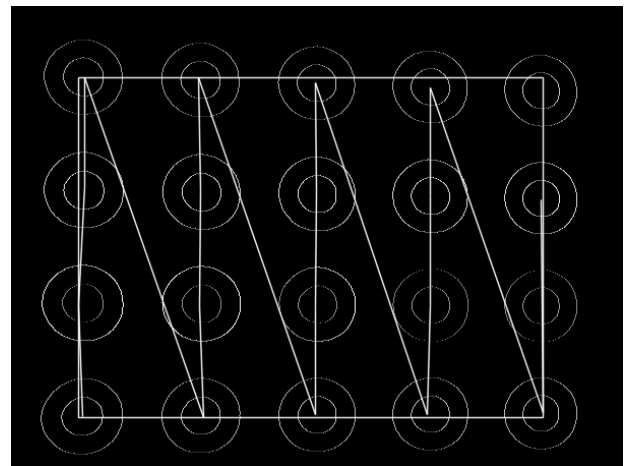
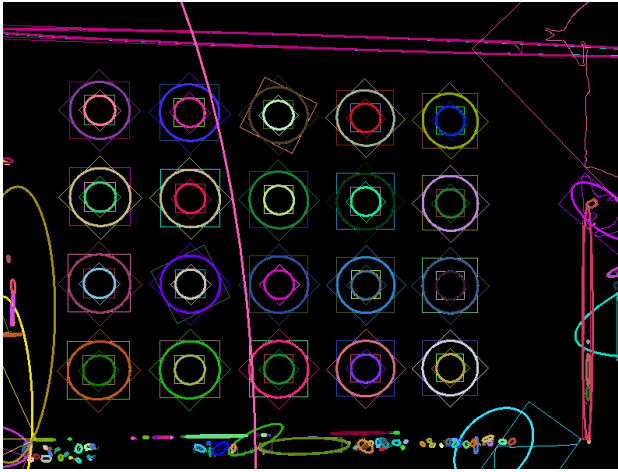


Imagen Como resultado Final

VI. AVANCES PARA LA SIGUIENTE SESIÓN

Culminar la búsqueda del Angulo de rotación para no perder el patrón.

Para ello probamos la función que utilizara como información el vector de círculos y sus posiciones `cv: RotatedRect box = cv: minAreaRect` y genera como resultado la siguiente imagen



VII. CONCLUSIONES

Hasta ahora el presente trabajo encuentra un patrón de círculos en una escena determinada, se agrupan técnicas de ordenamiento y se hace uso para el preprocesamiento diversas técnicas de opencv,

REFERENCIAS

[1] OpenCv.

“