# Calibración de una cámara usando un padrón en forma de anillo

Alejandra Callo Aguilar[1] y Jose Jaita Aguilar[2]

***Abstract*— En este trabajo se propone un método para la calibración de cámara usando un padrón de anillos, posteriormente se compara los resultados usando otros padrones como el chessboard y el de symmetric circles. Una parte del pipeline es la detección de anillos, para lo cual se propone un enfoque simple, robusto y rápido para dicha tarea.**

## I. INTRODUCCIÓN

La calibración geométrica de la cámara, también conocida como resección de cámara , estima los parámetros de una lente y un sensor de imagen de una imagen o cámara de video. Puede usar estos parámetros para corregir la distorsión de la lente, medir el tamaño de un objeto en unidades mundiales o determinar la ubicación de la cámara en la escena. Estas tareas se usan en aplicaciones tales como visión artificial para detectar y medir objetos. También se usan en robótica, para sistemas de navegación y reconstrucción de escenas tridimensionales.

El trabajo basicamente consta de detectar los anillos, luego con los puntos de control (centro de los anillos ) se pasa a la parte de la calibracion usando OpenCV [1] para dicha tarea.

El enfoque que se propone para la deteccion es simple, debido a que no se usan heuristicas complejas, esto conlleva a que sea muy rápido y pueda correr en tiempo real.

## II. DETECCIÓN DEL PADRÓN

### *II-A. Detección de contornos*

Primero, para la detección de los anillos , la imagen de entrada se convierte a gray, luego se le aplica una binarazacion usando el metodo de Otsu[2], para esto usamos la funcion de OpenCV ”threshold”,con la particularidad que le deciamos que nos calcule un threshold adecuado usando el metodo de Otsu, se pudo haber elegido manualmente un threshold, pero debido a la variacion de iluminacion , esta etapa no hubiese sido robusta. Luego tenemos una imagen binaria a la cual le aplicamos la funcion ”findContours”, entonces tenemos todos los contornos de los anillos pero tambien de otros objetos.

### *II-B. Filtros*

Esta etapa consiste en remover todo el ruido de la images, es decir remover todos los contornos que no formen parte de los anillos, el primer filtro que se usa es en cuanto a la relacion de ancho y altura, debido a que es un círculo, la relacion de tales parámetros deberia estar cerca a 1, en la práctica se comprobo que dicha relacioón fue hasta un limite de 0.4, esto se debio la perspectiva en que se tomaba la imagen. El segundo filtro era el área que formaba el contorno detectado, para el video de Kinetic2 se tuvo areas que fueron desde 1000 hasta 9000 pixeles, el tercer filtro fue escoger solamente los contornos que seas padres, como se sabe el anillo tiene dos contornos, el circulo exterior y el interior, entonces al usar la funcion ”findContour”se obtenia la jeraquia de los contornos encontrados, entonces al tener dicha informacion se filtrada todo contorno que no sea padre. Como se menciona los parametros de relación y area de los contorno eran variables, entonces dichos parametros se actualizaban con cada imagen de entrada.

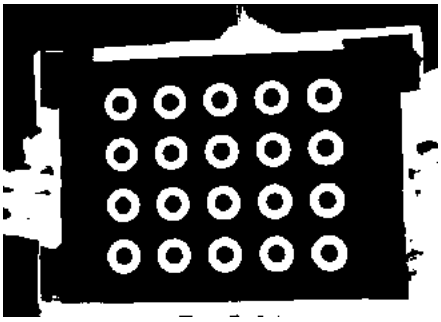

Fig. 1. Imagen binarizada usando Otsu

## III. REGIÓN DE INTERES (ROI)

La eficacia del metodo Otzu para hallar un threshold adecuado, depende mucho a que region de la imagen se aplique, lo ideal es que solamente se use la zona en donde se encuentra los anillos.

### *III-A. ROI*

Una vez detectado todos los anillos de define un roi, para esto simplemente hallabamos la ubicacion del anillo que se encuentra mas cerca al origen y el mas lejano, de este modo de definia el roi, pero debido a que dicho roi se aplicara en el siguiente frame, y los anillos podrían cambiar de posicion, entonces se le aplicaba un pequeño padding de 10 % del tamaño del roi, con tal ajuste era suficiente ya que entre frame y frame no hay mucha variacion ahora, si subiamos a porcentaje, se corre el riesgo de que el metodo de Otsu use pixeles irrelevante haciendo que no de un buen threshold. En la Fig 2. se mestra el roi de color celeste.

## IV. TRACKING

### *IV-A. Transformación*

Para la calibración es necesario hacer el tracking de los anillos detectados, cuando hallabamos los contornos en la

*Este trabajo es patrocinado por CONCYTEC

[1]A. Callo Aguilar, estudiante de la maestría en ciencias de la computación, Universidad Católica San Pablo, alejandra.callo@ucsp.edu.pe

[2]P. J. Jaita, estudiante de la maestría en ciencias de la computación, Universidad Católica San Pablo, jose.jaita@ucsp.edu.pe

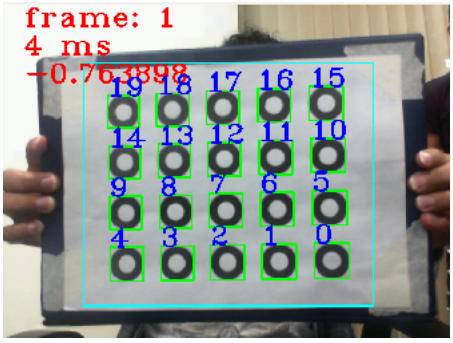


Fig. 2. Imagen con tracking

etapa anterior, si la imagen se encotraba sin ningun tipo de rotación, nosotros ya teniamos los anillos ordenados, la funcion ”findconoturs”devuelve los contorno con respecto al eje y. Los problemas comenzaban cuando se rataba la imagen entonces lo que se hizo fue una tranformacion de espacio, simplemente rotabamos la imagen y en esta nueva imagen encontrabamos los contornos, de esta forma teniamos el orden correcto de los contorno en la imagen original. Para que la rotación funsionase se debía conocer el angulo correcto, entonces nosotros actualizabamos constantemente dicho angulo, el cual era hallado usando la funcion de arcotangente entre la posicones del anillo 0 y el 1, obviamente el siguiente frame puede ser que cambie el angulo entonces no se trendra una ratación perfecta, ya que en cada frame se usa el angulo hallado con los anillos del frame anterior , entonces lo que se hace es llamar dos veces a la funcion de tranformacion, primera con el angulo pasado, luego hallabamos el nuevo angulo, de nuevo llamamos a la tranformación , entonces se tenia una transformacion perfecta.

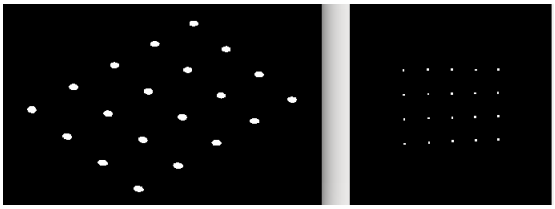

Fig. 3. Parte izquierda, la imagen original, lado derecho imagen rotada

En la Fig 3. vemos el resultados de transformar una imagen hacia otro espacio, en donde tenemos perfectamente ordenados lo anillos.

### *IV-B. Puntos de control*

La calibración depende exclusivamente de cuan preciso calculemos los centros de los anillos (puntos de control), una vez detectado el anillo volviamos a la imagen original para no perder informacion ya que , todos los metodos aplicados anteriorment se aplicaba al roi entero , el cual mete ruido , perdiendo detalles de los anillos ,por lo tanto para detectar el centro del anillo se trabaja solo en la region del contorno. Se aplica el metodo de Otsu para binarizar luego se encuentra los contornos, en este caso obteniamos solamente 2 contorno el del anillo exterior y el del interior, luego hallamos el centro de masa del objeto formado por cada contorno, en nuestros experimetos los centros de los dos círculo diferian por poco, menos de un pixel en promedio, entonces simplemente hallabamos el promedio de los dos, y este vendria hacer nuestro punto de control para nuestra calibración.

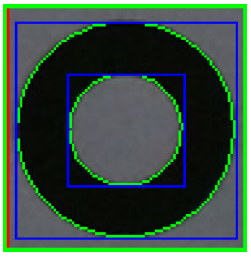

Fig. 4. Anillo segmentado

TABLE I
DIFERENCIAS DE CENTROS DE ANILLOS

| Anillo exterior | Anillo interior |
|---|---|
| 1206.02 - 656.312 | 1205.97 - 655.897 |
| 1084.8 - 621.017 | 1084.88 - 620.831 |
| 967.232 - 587.129 | 967.041 - 586.782 |
| 852.644 - 553.738 | 582.473 - 553.547 |

En la Fig 4. se muestra un anillo segementado con los bordes dectados, ahora se necesita el punto de control, en la tabla (I) se muestra la diferecia entre los centros de masa calculados para el circulo exterior y el interior. Por ejemplo [1206.02 - 656.321] es la ubicacion del circulo exterior y [1205.97 - 655.897] del circulo interior para el anillo de la fig 1. Los resultados anteriores fue para una imagen relativamente buena, en cambio cuando se calculo el centro de masa para otros frames en donde habia cierto blur debido al movimiento, la diferencia llego hacer de hasta 1 pixel.

## V. CALIBRACIÓN

Una vez hallado los 20 anillos pasamos a la calibración usando la función de OpenCV **calibratecamera**[1], a dicha funcion debiamos pasarle los puntos de control de la imagen (en nuestro caso los centros de los anillos) y las coordenadas de los objectos, en este caso sería la ubicacion de los puntos físicos, para esto se uso como referencia a la imagen mas no a la cámara, entonces para nuestros puntos fisicos nuestro eje de coordenadas es el tablero del padrón , y ya que cada punto de control esta distribuido uniformemente no era necesario medir distancia.

### *V-A. Parámatros intríncecos*

La preposion anterior era valida debido a que solo estamos hallando los parametros intrincecos, basicamente en la calibracion se halla la matriz de la cámara, la cual contiene las distancia focales fx y fy ademas de los centros opticos cx y cy expresados en pixeles. Pero la funcion de calibracion no

solo calculaba eso, sino que ademas nos dio los coeficientes de distorción radial.

## VI. REFINAMIENTO ITERATIVO DE LOS PUNTOS DE CONTROL

Nos basamos en el método propuesto por A. Datta[3], los pasos para obtener los puntos refinados se enumeran a continuación.

### *VI-A. Inicialización*

Lo primero que se hizo fue hacer una primera calibración, de modo que se obtenga tanto los parámetros intrinsecos como los extrínsecos, estos ultimos nos da el vector de rotación y traslación por cada imagen usada para la calibración (supongamos que usamos N imágenes).

### *VI-B. Un-Distorcionar y Un-proyectar*

Como ya calculamos los parámetros de la camara lo que hacemos es quitarle la distorción a los N frames usando el metodo de Opencv (undistord). Lo siguiente es obtener las imagenes en perspectiva fronto-paralela, cuando se hiso la primera calibracion, OpenCv nos pidió la ubicación de los puntos de control en el mundo real(puntos 3D), para esto nosotros definimos un eje de coordenadas relativo en donde el padron de anillos era el origen de coordenadas, con z = 0, y los puntos de control se encuentran a lo largo del eje XY, entonces para hacer la proyeccion de dichos puntos en la imagen necesitamos un matriz de homografia la cual nos permite pasar del sistema de coordenada del objeto al de la camara. Como se menciono anteriormente ya tenemos el vector de rotacion y translacion, el siguiente paso es hallar la matriz de homografia.

$$H = Hrotacion * Htraslacion \tag{1}$$

En (1) se tiene la matriz de homografía, la cual nos permitirá tener un vista fronto-paralela de padrón, de modo que se pueda calcular los puntos de control sin perder detalles. La homografia se descompone en Hrotacion y Htraslacion, actualmente la imagen esta a una distacial focal del centro optico ademas que se le aplico el desplazamiento del centro, entonces tenemos que regresar la imagen de modo que el centro coincida con la coordeanda (0,0) y para una distancia focal de 1, para esto lo multiplicamos por la inversa de la matriz de la camara(K), luego le aplicamos la matriz de rotacion inversa(es igual a la transpuesta), ahora ya tenemos el centro optico de la camara paralelo al padron objecto, en (2 R es la matriz de rotacion, previamente nosotros tenemos la rotacion pero en forma de vector, para convertirlo a matriz usamos Rodriguez de OpenCv), finalmente le aplicamos K para obtener la proyeccion del padron en la imagen, esto viene hacer Hrotacion, si bien la imagen esta fronto-paralela al padron , aun falta hacer un desplazmiento para tener un alineamiento perfecto entonces pasamos a calcular Hrotacion (3), en donde T es el vector de traslación.

$$Hrotacion = K * R^T * K^{-1} \tag{2}$$

$$Htraslacion = K * R^T * T \tag{3}$$

En la Fig. 5 se muestra la imagen un-proyectada, como se puede observar tiene una vista frontal.

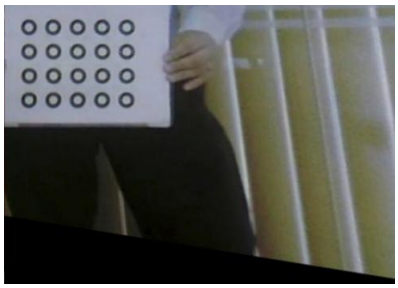

Fig. 5. Imagen un-proyectada

### *VI-C. Nuevos puntos de control*

Para tener el padron completo en la imagen hacemos una transformacion usando una homografía, se uso 4 puntos de control (especificamente las esquinas), entonces tenemos el padron en fronto-parelelo con z = 1, para esto usamos h1, que es la homografía para pasar a una vista canonica, en la Fig. 6 se muestra la imagen canonica del padrón, luego hallamos los puntos de control usando el algoritmo de detección adecuado dependiendo del padron, luego regresamos a la imagen obtenida de la un-proyección, y finalmente re-proyectamos (4) los puntos usando la inversa de H, en este punto ya tenemos el punto de control en la imagen original sin distorción usando Hregresar.

$$Hregresar = H^{-1} * h1^{-1} \tag{4}$$

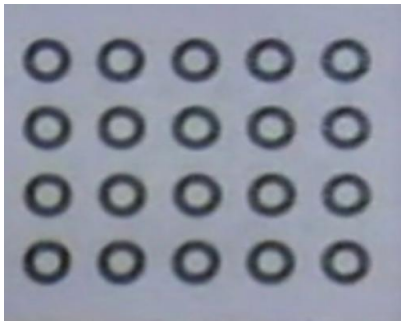

Fig. 6. Imagen canónica

### *VI-D. Distorcionar*

El punto de control actual esta en una imagen sin distorcion, pero como la imagen original estaba distorcionada, entonces es necesario distorcionarla, para esto usamos los coeficientes de distorción y la matriz de la camara.

### *VI-E. Nueva calibración*

Con los nuevos puntos de control para las N imágenes volvemos a calibrar pero previamente calulamos la media entre los puntos antiguos y los nuevos, esto se hace hasta que el rms converja. Para nuestros experimentos se uso 5 iteraciones, aunque en algunos videos de prueba la convergencia se alcanza en la segunda iteración. Para los videos

finales la itercion recien se alcanza en la quinta iteración. En Fig(7) se muestra la corrección de los puntos de control luego de hacer la re-proyección.

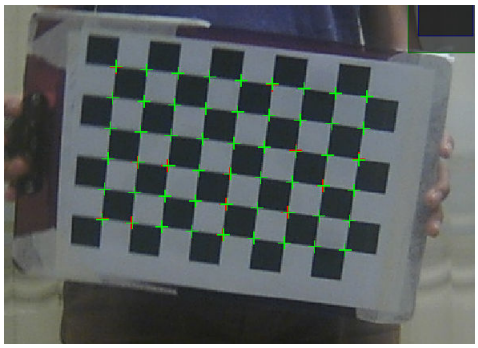

Fig. 7. Nuevos puntos de control

## VII. EXPERIMENTOS

Para nuestros experimentos usamos dos camaras (Playstation PS3 y lifecam), para la calibración de dichas camaras se uso 3 padrones:

- Chessboard: los puntos de control eran hallados usando la funciones de Opencv ”findChessboardCorners”.
- Symmetric circle: usamos la funcion de OpenCV ”findCirclesGrid”.
- Rings: para los anillos usamos nuestro metodo.

TABLE II
RMS

| Camera | Chessboard | Circle | Ring |
|---|---|---|---|
| LifeCam | 0.308884 | 0.28751 | **0.182048** |
| PS3 | 0.504785 | 0.41254 | **0.305087** |

TABLE III
DISTANCIA FOCAL FX, FY

| Camera | Chessboard | Circle | Ring |
|---|---|---|---|
| LifeCam | 597.6 587.5 | 580.4 585.2 | 582.2 580.1 |
| PS3 | 857.7 847.1 | 842.1 824.8 | 810.9 814.6 |

TABLE IV
CENTROS CX,CY

| Camera | Chessboard | Circle | Ring |
|---|---|---|---|
| LifeCam | 313.6 226.9 | 314.8 228.2 | 321.8 224.9 |
| PS3 | 338.4 262.9 | 325.7 249.5 | 316.8 241.6 |

En las tablas (II, III, IV, V) se muestran los resultados de los experimentos llevados a cabo, usando tres patrones , dos cámaras y 20 imagenes por cada video, como se puede ver los mejores resultados fueron obtenidos usando el padrón de anillos, en el caso de la colinearidad se muestra dos valores,

TABLE V
COLINEARIDAD

| Camera | Chessboard | Circle | Ring |
|---|---|---|---|
| LifeCam | 0.0879 - 0.0625 | 0.0825 - 0.0684 | 0.0966 - 0.0522 |
| PS3 | 0.124 - 0.0642 | 0.1025 - 0.0498 | 0.1102 - 0.0587 |

el primero es la colinearidad usando una simple calibracion y la segunda es usando el refinamiento.

Como se sabe un parámetro para saber cuan bien se calibro es el RMS, pero no solo es ese, sino que ademas de calcular dicho parámetro. se necesita ver como actua la calibracion en imagenes, en nuestros experimentos se probo muchas calibraciones, y cuando se aplicaba la correción usando su respectiva matriz de la camara y los factores de distorsión a veces se obtenia una imagen concava y aveces convexa. Ademas de tener una referencia gráfica, se aplico la colinearidad de modo que se tenga una metrica para compararar las calibraciones.

TABLE VI
SELECCIÓN ALEATORIA VS MANUAL

| razón | 20 | 50 | 80 | 150 | 300 |
|---|---|---|---|---|---|
| aleatorio | 0.572116 | 0.541251 | 0.562230 | 0.532517 | 0.525441 |
| manual | **0.305087** | - | - | - | - |

La selección de imágenes para la caibración se puede hacer de forma aleatoria y manualmente, en la tabla (VI) se muestra la comparacion , para la seleccion aleatoria se hizo correr el video y cada cierto numero de imagenes se grababa la imagen actual la cual posteriormente sería utilizada para la calibracion. En el resto de nuestros experimentos usamos imágenes seleccionadas manualmente ya que nos da mejores resultados debido a que se evita seleccionar imagenes repetidas o con poco rotacion ya sea a travez de eje X o Y, pero aparte de eso se hizo pruebas usando un filtrado de imagenes, en donde solo se usaban imagenes que tenga una rotacion entre 5 y 60 grados, en la Fig(8) se muestra la comparacion usando diferentes cantidades de imágenes, como se observa se tiene mejores resultados cuando las imágenes son filtradas.

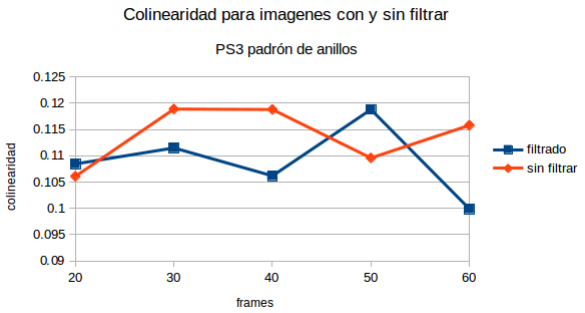


Fig. 8. Colinearidad para imágenes filtradas

En la Fig(9) ahora se observa como se comporta el rms con

respecto al filtrado de imagenes para la calibración usando diferentes cantidades de imagenes para la calibracion.

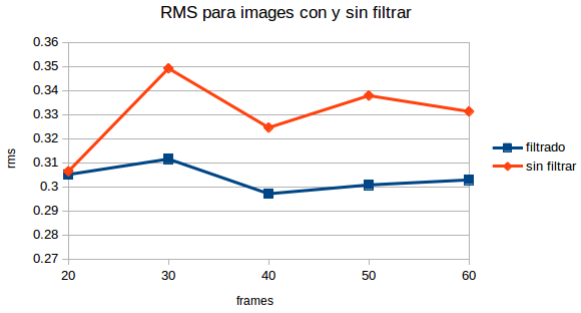


Fig. 9. RMS para imagenes filtradas

Como ya se mencionó se usaron 5 iteraciones para el refinamiento, en la Fig(10) se muestra en comportamiento del rms a traves de las 5 iteraciones, para los tres patrones usando tanto PS3 como LifeCam.

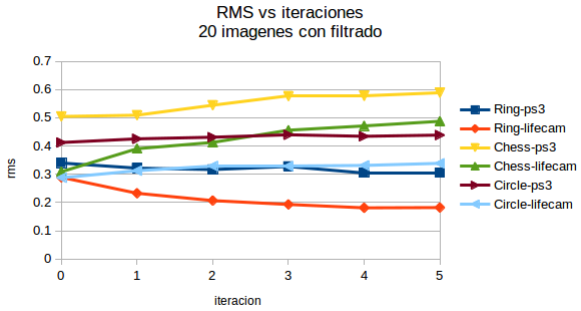


Fig. 10. RMS a travez del refinamiento

Ademas del RMS, en las Fig(11) y Fig(12) se muestra la distancia focal(fx,fy) y centro óptico(cx,cy) respectivamente a travez de las 5 iteraciones, para este experimento se uso el padrón de anillos con 20 imágenes y la PS3 ademas que para todos los expermientos se uso el filtrado de imagenes (6 a 60 grados) ya que nos da mejores resultados como se muestra en la Fig(8) y Fig(9)

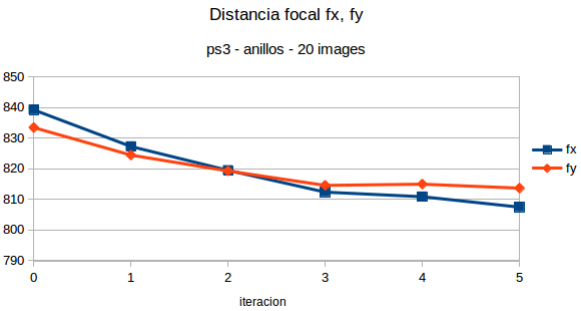


Fig. 11. RMS a traves del refinamiento

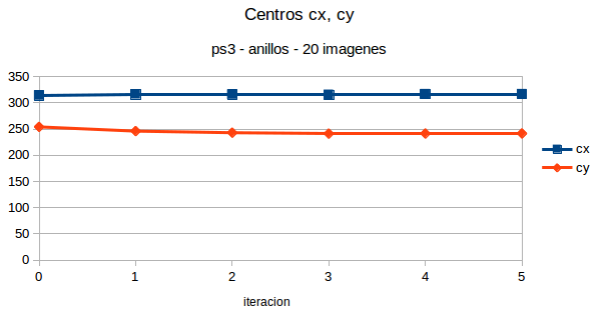


Fig. 12. RMS a traves del refinamiento

## VIII. RESULTADOS

En las Figs. (13, 14, 15) se muestra una imagen con distorsión, caibrada una sola vez y calibrada usando el refinamiento respectivamente, se uso el padrón de anillos y la cámara ps3. En las Figs (16, 17) usando el lifecam.

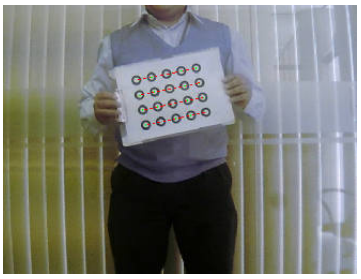

Fig. 13. Imagen original, ps3

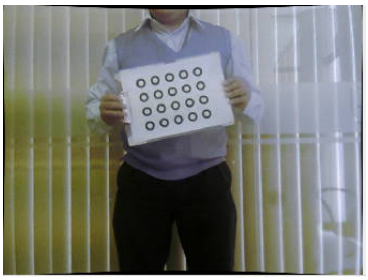

Fig. 14. Imagen calibrada, ps3, 1era iteración

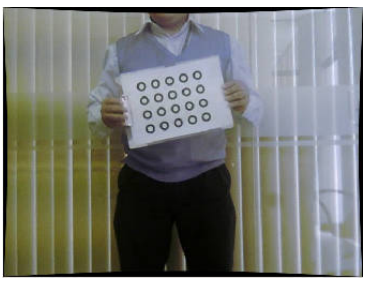

Fig. 15. Imagen calibrada, ps3, 5ta iteración

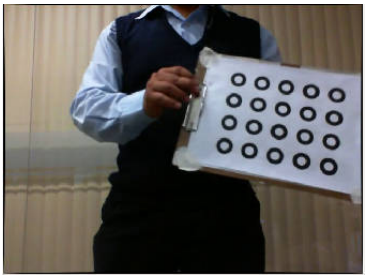

Fig. 16. Imagen calibrada, usando lifecam

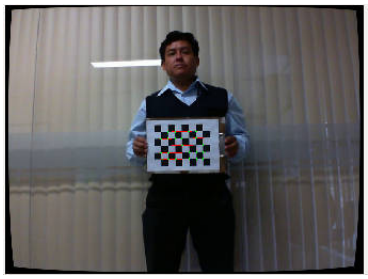

Fig. 17. Imagen calibrada, usando lifecam

## IX. CONCLUSIONES

El método propuesto para la detección de los anillos es robusto y rapido, corre en tiempo real, es mas puede correr a 60 fps, usando las imágenes de la cámara lifescan le tomaba entre 5 a 10ms hacer el procesamiento. El que los anillos tenga dos circulos ayuda mucho a la detección, facilmente se elimina falsos negativos. En cuanto a la calibración se escogio 20 imagenes por video, fue con el que nos dio los mejores resultados, la tabla 1 son los mejores resultados obtenidos despues de haber corrido varios experimentos con diferentes numero de imagenes. Tanto el padrón circular como el de anillos muestra una mejora con respecto al clasico chessboard, con esto se concluye que el padron en forma de anillo es el mejor para la calibración. El motivo que nosotros pensamos es porque los circulos son invariantes a la rotación, aunque con cierta perspectiva el circulo tiende a convertirse en elipse,pero aun asi se conserva el mismo punto de control, ya que lo que se halla es el centro, en cambio en el chessboard cuando hay rotacion mas un cambio de perpectiva se obtienen puntos de control diferentes. Se concluye que el parametro principal para que la calibracion sea buena, son los puntos de control calculados, estos deben ser lo mas preciso posible, tambien ayuda el numero de imagenes y cuantos puntos de control se utilizan por cada imagen, por ejemplo si hay un error en un punto y si los demas estan bien , entonces el error se reduce. Se comprobó que usar imagenes entre 5 y 60 grados da una mejor calibración, para los videos de prueba el refinamiento no funciono para el padron de ajedrez y el circular en cambio para el de anillos si hubo convergencia. Si bien en los videos de prueba tanto el padrón de ajedrez i criculas no funcionó en otros videos de prueba si hubo convergencia. La principal diferencia entre los videos es que en los de prueba el padron se encuentra muy alejado, otra causa que suponemos es porque algunos parámetros de la cámara no fueron configurados correctamente a la hora de grabar los videos.

## X. TRABAJOS FUTUROS

Para el siguiente trabajo se determinará porque en los padrones del ajedrez y el de circulos no funciona el refinamiento. Ademas una vez calibrada la cámara se puede hacer realidad aumentada, como inicio se dibujo un cubo en las imagenes de un video, esto se muestra en las Figs. (18) y (19) en diferentes posiciones.

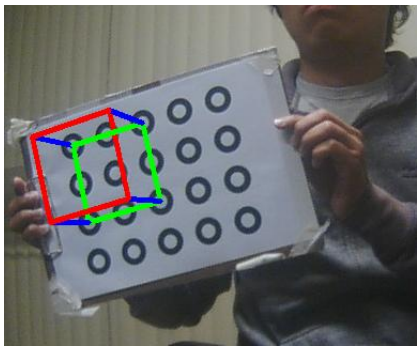

Fig. 18. Imagen con un cubo

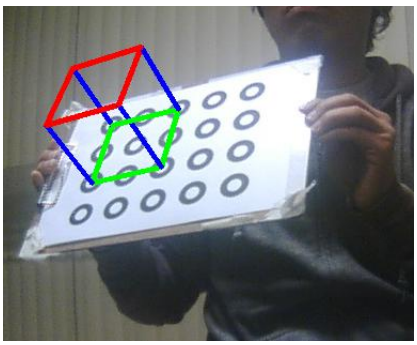

Fig. 19. Imagen con un cubo

## REFERENCES


[1] OpenCV, camera calibration: `http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`

[2] PoenCV, Otsu: `https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/threshold/threshold.html`

[3] A. Datta, J. S. Kim and T. Kanade, .Accurate camera calibration using iterative refinement of control points,"*2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, Kyoto, 2009, pp. 1201-1208.*