Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Prototipo de un sistema de ojo de halcón

Tesis que presenta

Daybelis Jaramillo Olivares

para obtener el Grado de

Maestra en Ciencias

en Computación

Director de la Tesis

Dr. Luis Gerardo de la Fraga

Ciudad de México **Agosto 2018**

Resumen

En esta tesis se construye el hardware y software para desarrollar un prototipo de ojo de halcón. El hardware del sistema consiste en un modelo de madera con una iluminación controlada, una computadora en una sola tarjeta modelo Raspberry Pi 3 B y una cámara que es un módulo de la misma Raspberry Pi. El software de este sistema calcula por medio de procesamiento de imagen y técnicas de visión por computadora la orientación y posición de un marcador, que es un objeto conocido, que se mueve con la mano dentro del modelo de madera del ojo de halcón.

Primero se diseñó y construyó el modelo de madera y luego se desarrolló el software. La mayor parte de tiempo de desarrollo fue en la etapa de procesamiento de imagen para detectar de forma automática el marcador en las imágenes.

El marcador que se detecta consiste de puntos, y estos puntos se forman con los vértices de triángulos. Para obtener una precisión subpixel, se detectaron la posición de los vértices mediante la intersección de las aristas de los triángulos. Y cada arista se detecta a través de un ajuste lineal de los pixeles que lo componen.

De la parte de visión por computadora se detectó la pose del marcador por medio de la homografía entre los puntos del marcador y los puntos de la imagen que toma la Raspberry.

Abstract

In this thesis, the hardware and the software to develop an Hawk-eye prototype were built. The hardware of this system consists of a model made out of wood, with a controlled illumination, a computer, a single board computer (CSB), Raspberry Pi 3 B card and a camera which is a module of the Raspberry Pi. The software of this system calculates using image processing and computer vision techniques the orientation and position of a fiducial marker, which is a known bidimensional object. This marker is moved by the user's hand within the wooden model of the Hawk-eye prototype.

First of all, the wooden model was designed and built and then the software was developed. The biggest part of the development time was the stage of image processing to detect automatically the marker within the images.

The marker that is detected consists of a set of points, and these points are formed with the vertices of triangles. To obtain a subpixel precision, the position of the vertices was detected by the intersection of the edges of the triangles. And each edge is detected through a linear fitting of the set of pixels that belong to it.

Computer vision was used to estimate the pose of the marker, which was detected by the use of the homography between the points of the marker and the points of the image taken by the Raspberry.

Agradecimientos

Al CINVESTAV

A mi director de tesis el Dr.Luis Gerardo de la Fraga por todos los conocimientos que me inculcó desde el primer día que empecé mis estudios de posgrado en el CINVESTAV. Agradezco especialmente que me haya guiado en el desarrollo de esta tesis, por el conocimiento que me impartió de los temas que se incluyen en esta tesis y por instruirme con firmeza y paciencia. Gracias a su guía, la cual me motivo, he podido culminar mis estudios de maestría.

A la Dra.Sonia G. Mendoza Chapa y el Dr.José Guadalupe Rodríguez García agradezco por el tiempo invertido en la revisión de esta tesis.

Agradezco a los profesores del departamento de computación por todos los conocimientos que me impartieron, así como al personal administrativo por siempre estar presente haciendo mi estancia aquí una gran experiencia.

Al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, agradezco por abrir sus puertas para mí, darme la oportunidad de crecer profesionalmente. A lo largo de mi estancia en sus instalaciones, la cual duró dos años, se convirtió en un segundo hogar para mí. Gracias por todos los conocimientos adquiridos, las personas que conocí en sus instalaciones y todos aquellos buenos momentos los cuales llevaré siempre en mi memoria.

A mi familia y amigos

A mi mamá Nina Olivares Rodríguez y mi papá Renne Jaramillo Cázares, gracias por siempre estar a mi lado siendo mi apoyo en todo momento y ser parte fundamental en mi desarrollo tanto personal como profesional, lo cual me ha dado el impulso para continuar mis estudios en esta etapa.

A mis tíos Olga Olivares Rodríguez y Edgar Olivares Rodríguez quienes han sido como unos segundos padres para mí, siempre han estado apoyándome y aconsejándome de la manera en que un padre lo hace, gracias por todos aquellos sabios consejos que me han dado a lo largo de la vida. Y a mis tíos Nieves Olivares Rodríguez y Jorge Olivares Rodríguez, gracias por procurarme.

A mi abuela Eufrosina Rodríguez Román por siempre procurarme desde mi infancia, gracias por todo el apoyo que me ha dado.

A mi primo Pedro Olivares Herrera quien ha sido lo más cercano a un hermano de sangre para mí, siempre ha estado presente en mi vida desde la infancia, gracias por cuidarme y protegerme como una hermana.

A Adrián Josué Ramírez Díaz por acompañarme a lo largo de esta etapa desde el principio al fin, gracias por los momentos que compartimos, conservaré esos recuerdos por siempre. Gracias por el apoyo que en todo momento me brindaste, un pilar que no me dejó caer aún en los momentos difíciles.

A Jorge Salinas Lara por ser mi compañero y amigo por más de siete años. Nuestra amistad comenzó el primer día de nuestra carrera universitaria, jamás imaginé que llegaríamos a cursar nuestros estudios de posgrado juntos. Gracias por el apoyo, compañía y amistad que me has brindado todos estos años.

A mis compañeros de la generación quienes hicieron mi estancia en el CINVESTAV más amena.

A CONACYT

Al Consejo Nacional de Ciencia y Tecnología por la beca que me otorgó durante mis estudios de maestría durante mi estancia en esta institución.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Los sistemas de ojo de halcón ya se utilizan en la actualidad, por ejemplo en los partidos de tenis del circuito ATP [1] para la revisión de jugadas y tomar la decisión más justa. También son usados en los partidos de la copa de mundo de fútbol [2] y en la liga premier de Inglaterra, para verificar si el balón traspasa la línea de meta. El sistema pionero fue creado para su uso en el cricket [3].

En esta tesis se quiere desarrollar un prototipo de ojo de halcón para conocer su tecnología y desarrollar el software necesario para su funcionamiento. El sistema ojo de halcón necesita hacer el seguimiento de un objeto, aquí se propone reconocer un marcador que se usa en el área de realidad aumentada y robótica, como lo son los ARTags [4], AprilTags [5], BinARyD [6] y los usados en el ARToolkit [7]. Se usará aquí un nuevo tipo de marcador compuesto de triángulos y con los que se obtiene automáticamente un identificador y la correspondencia de puntos con el modelo del mismo marcador.

Para el procesamiento se usará computadoras en una sola tarjeta (CSTs). Esta computadora es una Raspberry Pi 3.Raspberry Pi 3 es una computadora de bajo costo que cuenta con un procesador Quad Core Broadcom BCM2837 de 64 bits y tiene una velocidad de procesamiento de 1.2 GHz [8]. Cada Raspberry tendrá conectada a través de su puerto CSI un módulo de la cámara versión 2, que se vende como aditamento para la Raspberry, tiene un sensor Sony IMX219 de 8 megapíxeles, es una cámara usada en dispositivos móviles la cual cuenta con soporte de 1080p30, 720p60 y modo de video VGA90 [9]. Este hardware es más fácil de utilizar porque ya viene soportado por el vendedor de la Raspberry Pi.

El marcador es un elemento único que facilita su identificación, se conoce su forma previamente a la captura y cuando es capturado se reconoce debido a un entrenamiento previo o algún algoritmo de identificación. El marcador nos aporta información para conocer la ubicación o pose (su posición y orientación) del objeto. Con técnicas de procesamiento de imagen se aislan los píxeles correspondientes al marcador y obtiene su ubicación en el área bidimensional con la cámara. Posteriormente utilizando métodos de visión se realiza la triangulación se obtiene la ubicación tridimensional de los vértices

del marcador.

Una vez que se obtengan las posiciones de los vértices se tienen que triangular las posiciones de estos para obtener las posiciones de las coordenadas tridimensionales de cada vértice. La pose, esto es la posición y orientación de cada marcador como lo ve cada cámara pueden obtenerse vía la homografía entre la imagen y el modelo del marcador o por un algoritmo de n puntos, que se conoce como problema PnP [10, 11].

El sistema de ojo de halcón resolverá la pose del marcador con respecto al área de trabajo, entonces se podría ver también como un dispositivo óptico para la captura de la pose de ese marcador. Finalmente, la pose del marcador se utilizará para

1.1. Planteamiento del problema

La visión por computadora ha realizado grandes procesos en los últimos años debido a las mejoras en tecnología de las cámaras, así como las nuevas capacidades de cómputo. La visión por computadora comprende varias áreas. Las que son de interés para la realización de este trabajo son la estimación de la pose y la reconstrucción tridimensional.

La estimación de pose es una herramienta fundamental en aplicaciones de visión por computadora como lo son la realidad aumentada y la reconstrucción tridimensional y consiste en determinar la transformación tridimensional requerida para obtener un punto en la escena de un sistema global de coordenadas y convertirlo al sistema de coordenadas de una cámara.

Como fue mencionado con anterioridad, en esta tesis se propone la realización de un prototipo con una computadora de bajo costo como lo es la Raspberry Pi 3 B. La capacidad de procesamiento de esta computadora no es de gran velocidad y su cámara integrada no son de gran resolución en comparación con las que se encuentran hoy en el mercado para dispositivos móviles. Sin embargo se pretende obtener eficiencia y precisión en la estimación de la pose en tiempo real con este equipo.

1.2. Objetivos general y particulares

El objetivo de este trabajo es construir un prototipo de sistema ojo de halcón como una interfaz para obtener orientación y posición de un objeto en tres dimensiones utilizando herramientas de costo bajo.

A continuación se listan los objetivos particulares:

- Delimitar el tamaño, resolución del área de visión y posicionamiento de la cámara.

- Diseñar el marcador que será usado para encontrar el objeto y conocer su pose.
- Programar la etapa de procesamiento de imágenes para obtener las posiciones de los vértices de los triángulos del marcador.
- Implementar el algoritmo para la obtención de la posición y orientación tridimensional.
- Desarrollar una interfaz gráfica OpenGL para realizar pruebas conceptuales del sistema.

1.3. Estructura del documento

La presente tesis esta compuesta por los siguientes capítulos:

En el capítulo 2 se exponen los conceptos teóricos que son base para la realización de este trabajo. En el capítulo 3 se describe de forma general el funcionamiento del prototipo del ojo de halcón. En el capítulo 4 se muestra una simulación numérica del prototipo y el funcionamiento del mismo. Finalmente, en el capítulo 5 se describen algunas conclusiones del desarrollo de este trabajo y el desarrollo futuro que se debería realizar para mejorar el prototipo.

Capítulo 2

Fundamentos teóricos

El presente capítulo pretende exponer los conceptos teóricos más relevantes empleados a lo largo de este trabajo, lo cual permitirá una mejor compresión de las bases para su realización. En la sección 2.1 se describen las etapas del procesamiento de imagen. En la sección 2.2 se exponen los conceptos de transformaciones geométricas las cuales permiten que un objeto realice cambios en su posición y orientación. Los conceptos de visión por computadora se presentan en la sección 2.3 en los cuales se tocan temas como el modelo de la cámara obscura, la calibración de la cámara y la triangulación.

2.1. Procesamiento de imagen

Se conoce como procesamiento de imagen al conjunto de técnicas que son aplicadas para realizar análisis y manipulación a las imágenes, cuyo objetivo principal es obtener una mejor calidad en la imagen o encontrar información relevante de su contenido [12, 13].

Las etapas para el reconocimiento de objetos en una imagen se realiza con técnicas de procesamiento de imagen y se describen en la figura 2.1 [12, 13].

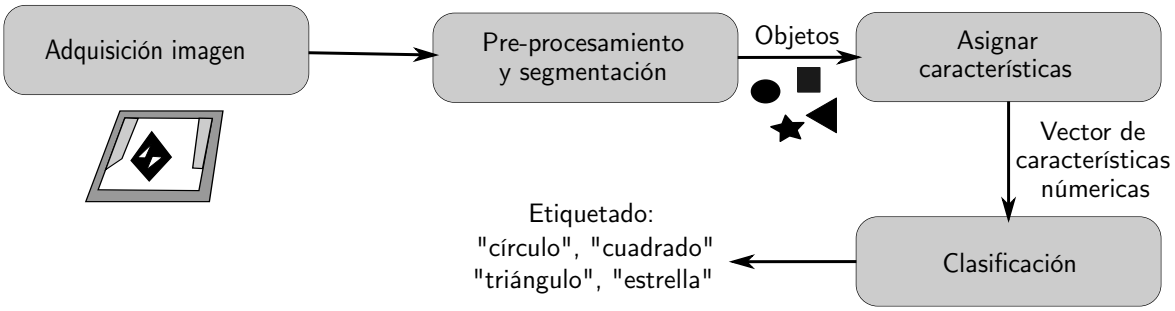


Figura 2.1: Etapas para el reconocimiento de objetos.

1. La adquisición de imagen consiste en capturar una escena del mundo real a través de cámaras

2. La etapa de pre-procesamiento se trata de aplicar técnicas de mejoramiento de la imagen, reducción de ruido y realzar sus características. Por otro lado la segmentación realiza la separación de los objetos que se encuentran presentes en la imagen.

3. Asignar características a los objetos es realizar una descripción de manera numérica los objetos segmentados, ya sea por color, textura, forma, área, entre otras características.

4. La clasificación es la parte principal en la cual asignamos una clase o categoría a cada objeto de la imagen basado en los rasgos del objeto.

5. El etiquetado es la última etapa del reconocimiento de objetos, consiste en asignar un nombre o etiqueta a los objetos que se encuentran en la imagen.

Con el fin de reconocer el marcador fiducial dentro de la imagen, se llevaron a cabo todas estas etapas como se describe a continuación:

La adquisición de imagen se realiza usando el módulo cámara de Raspberry Pi 3. La imagen es capturada a color con una resolución 480×640 píxeles.

El pre-procesamiento consiste en convertir la imagen captura en escala de grises usando el modelo RGB, posteriormente a esa imagen se le realizan técnicas de reducción de ruido con operaciones morfológicas y después se binariza la imagen, esto quiere decir convertir la imagen a blanco y negro, aplicando un umbral.

La segmentación de la imagen se lleva a cabo una vez que tenemos la imagen binarizada. La segmentación permite identificar todos los objetos que están presentes en la imagen y a éstos se les asigna una clase. La segmentación de objetos para el entrenamiento y obtención del clasificador, se realiza generando una nueva imagen por cada objeto que se encuentre en la imagen.

Un clasificador consiste en asignar a una clase o categoría disponibles a cada objeto de la imagen basado en sus características. La computadora realiza cálculos para poder hacer la clasificación de objetos, para que esto sea posible las características deben poder cuantificarse.

El proceso de clasificación siempre se compone de los siguientes pasos:

1. Se toman objetos muestra con clases conocidas. Se eligen y se calculan las características de los objetos muestra.

2. Estas características se usan para entrenar el clasificador.

3. Se extraen las mismas características de los objetos desconocidos que se desea clasificar.

4. El clasificador usa las características calculadas durante el entrenamiento para decidir en qué clase pertenecen los objetos a reconocer.

2.2. Transformaciones geométricas tridimensionales

Es fundamental representar la posición y orientación (pose) de los objetos en un entorno virtual. Las transformaciones geométricas son usadas con el propósito de realizar cambios sobre la pose de un objeto.

Un punto en el espacio se representa con tres valores de coordenadas. Matemáticamente se puede usar un vector para almacenar los valores del punto, pero un vector tiene un significado distinto porque representa una traslación o desplazamiento de un punto con respecto a otro punto de referencia. Por lo general, el punto de referencia es el origen del sistema de coordenadas.

Un objeto es identificado mediante un conjunto de puntos. De esta manera es posible definir la pose de un objeto en un espacio tridimensional, en lugar de localizar la posición de cada uno de los puntos que conforman el objeto.

2.2.1. Traslación

La traslación es la transformación geométrica isométrica la cual permite cambiar la posición de un objeto desplazándolo hacia un punto en el espacio. La operación de traslación se define como:

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \\ z' &= z + t_z \end{aligned} \tag{2.1}$$

La traslación también puede ser expresada en forma de matriz utilizando puntos tridimensionales homogéneos:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.2}$$

En la figura 2.4 se puede observar un ejemplo de esta transformación. El tetraedro es trasladado hacia la derecha y arriba de su posición original, como se puede apreciar éste mantiene su propia forma y tamaño.

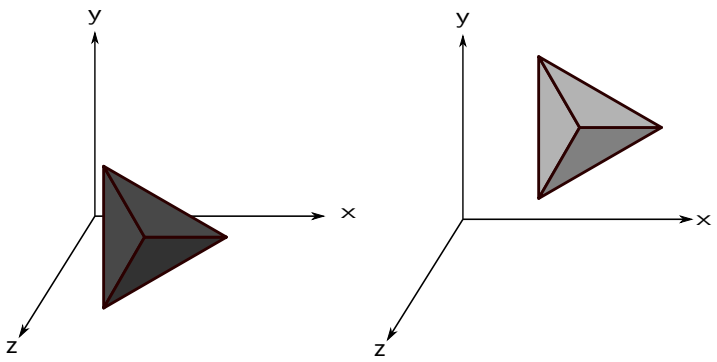


Figura 2.2: Transformación de traslación

2.2.2. Rotación

La rotación es la transformación geométrica isométrica que permite cambiar la orientación de los objetos en el espacio. La rotación puede expresarse por medio de una matriz de tamaño 3×3, que tiene tres grados de libertad en vez de nueve. Otra forma de expresar la rotación es por las rotaciones alrededor de los ejes principales x, y, z, que se definen a continuación:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\operatorname{sen}(\theta) \\ 0 & \operatorname{sen}(\theta) & \cos(\theta) \end{bmatrix} \tag{2.3}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \operatorname{sen}(\theta) \\ 0 & 1 & 0 \\ -\operatorname{sen}(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.4}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\operatorname{sen}(\theta) & 0 \\ \operatorname{sen}(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

En la figura 2.3 se puede observar como la figura está rotada sobre el eje x 45°, se puede observar como mantiene la misma forma y tamaño.

2.2.3. Rotación y traslación

Para realizar varios cambios en un objeto se utilizan un conjunto de transformaciones. Las transformaciones son representadas por operaciones matriciales, por lo que se tiene que tener especial atención al orden en que se realizan las operaciones. Es posible expresar la traslación y la rotación utilizando una matriz ampliada de transformación

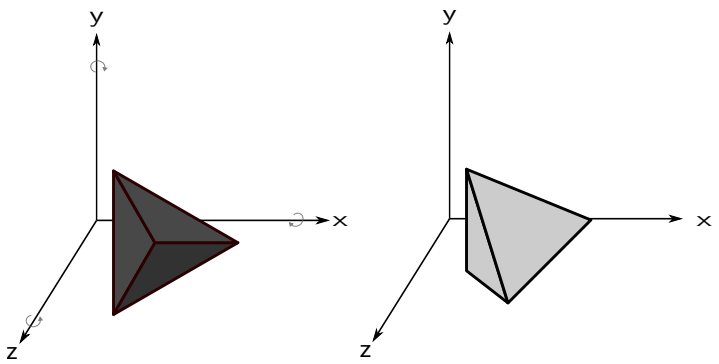


Figura 2.3: Transformación de rotación.

usando coordenadas homogéneas como se muestra a continuación [14, 15]:

$$T \cdot R = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Es importante tener en cuenta que al aplicar esta matriz de transformación, en primer lugar se aplica a los puntos del objeto la rotación y después la traslación.

2.2.4. Escalamiento

El escalamiento es la transformación geométrica isomórfica que permite cambiar el tamaño de un objeto en el espacio. Dado que se aplica una multiplicación sobre todos los puntos que conforman un objeto, la proporción de éste cambia. El escalamiento de un punto tridimensional se define como:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} e_x & 0 & 0 & 0 \\ 0 & e_y & 0 & 0 \\ 0 & 0 & e_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.7}$$

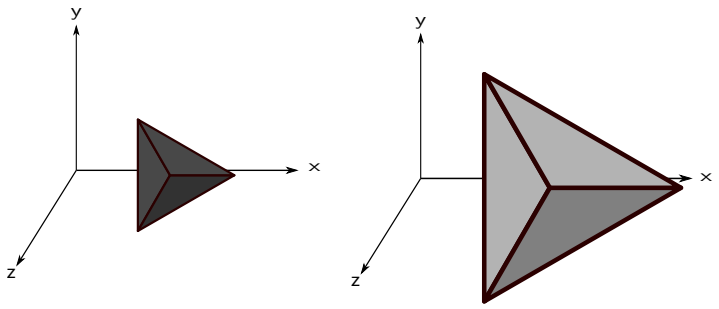


Figura 2.4: Transformación de escalamiento.

2.3. Conceptos de Visión por Computadora

Para poder llevar a cabo la recuperación de la pose de un objeto de manera tridimensional a partir de imágenes bidimensionales es necesario comprender la manera en que son formadas las imágenes en base a capturas de entornos tridimensionales. Al realizar la captura de un espacio tridimensional $\mathbb{R}^3$ en una imagen bidimensional $\mathbb{R}^2$ se realiza un proceso proyectivo, en el cual se pierde una dimensión [14]. En visión por computadora se estudia como recuperar esa dimensión perdida a partir de imágenes bidimensionales.

2.3.1. Pose de un objeto

La pose de un objeto se refiere a la posición y orientación que tiene éste respecto a un sistema de coordenadas. La pose se define por las coordenadas y los ángulos de rotación respecto a los tres ejes x, y, z, por lo que la pose de un objeto en el espacio tridimensional tiene seis grados de libertad.

2.3.2. Modelo de la cámara obscura

Formalmente, el modelo de la cámara obscura [14] define la relación entre la ubicación de un punto 3D $\mathbf{P} \in \mathbb{R}^3$ en una escena del mundo y la transformación de proyección de ese punto tridimensional como es visto por la cámara de forma bidimensional en un punto $\mathbf{p} \in \mathbb{R}^2$. En la figura 2.5 se muestra este relación.

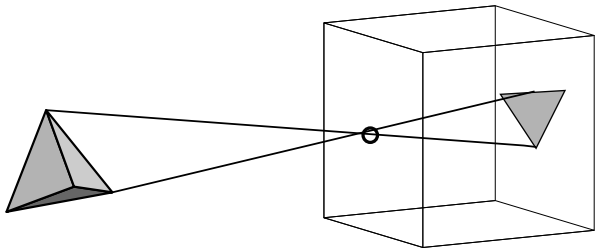


Figura 2.5: Modelo de la cámara obscura.

Dado un punto tridimensional $\mathbf{P}$ cuyas coordenadas homogéneas son $\mathbf{P} = [x, y, z, 1]^T$, y su correspondiente punto en la imagen $\mathbf{p} = [u, v, 1]^T$, representado también en coordenadas homogéneas.

La relación existente entre un punto de la imagen $\mathbf{p}$ y el punto en el mundo $\mathbf{P}$ es:

$$\lambda \mathbf{p} = A\mathbf{P} \tag{2.8}$$

donde A es una matriz de tamaño 3×4 que mantiene la información necesaria para rotar, trasladar y proyectar el punto tridimensional $\mathbf{P}$ a un punto $\mathbf{p}$ sobre el plano que

forma la imagen y λ es un factor de escala.

Por otro lado, A esta definida como:

$$A = KM \tag{2.9}$$

donde M es una matriz aumentada compuesta por los parámetros extrínsecos $[R|\mathbf{t}]$. R es una matriz de rotación de tamaño 3×3 y $\mathbf{t}$ es un vector de traslación de tamaño 3. K es una matriz de tamaño 3×3 que se conoce como matriz de parámetros intrínsecos y calcula la transformación de proyección en perspectiva.

Al realizar la sustitución de A en la ecuación (2.8) se obtiene:

$$\lambda \mathbf{p} = K[R|\mathbf{t}]\mathbf{P}, \tag{2.10}$$

Desarrollando la ecuación (2.10) se obtiene:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.11}$$

El punto 3D $\mathbf{P}$ primero es rotado por R y luego trasladado según el vector $\mathbf{t}$. f en la matriz K es la distancia focal, $(u0, v0)$ es el punto principal de la cámara.

La ecuación (2.10), puede cambiarse a otra forma más intuitiva al sacar R de la matriz aumentada $[R|\mathbf{t}]$, con lo que se obtiene:

$$\lambda \mathbf{p} = KR[I| - \mathbf{c}]\mathbf{P}, \tag{2.12}$$

donde $\mathbf{c} = -R^T\mathbf{t}$ y es la posición de la cámara. R se obtiene a partir de $\mathbf{c}$, un punto a donde ve la cámara sobre el objeto $\mathbf{o}$, y un vector que indica la dirección de arriba para la cámara $\mathbf{a}$, aplicando las siguientes fórmulas:

$$\mathbf{z_c} = \frac{\mathbf{c} - \mathbf{o}}{\| \mathbf{c} - \mathbf{o} \|} \tag{2.13}$$

$$\mathbf{x_c} = \frac{\mathbf{a} \times \mathbf{z_c}}{\| \mathbf{a} \times \mathbf{z_c} \|} \tag{2.14}$$

$$\mathbf{y_c} = \mathbf{z_c} \times \mathbf{x_c} \tag{2.15}$$

$$R = \begin{bmatrix} \mathbf{x_c^T} \\ \mathbf{y_c^T} \\ \mathbf{z_c^T} \end{bmatrix} \tag{2.16}$$

La matriz A, como se puede ver en la ecuación (2.9), es una combinación de los parámetros intrínsecos y extrínsecos de la cámara. K se calcula con un método de calibración de la cámara y una vez conocida K se puede obtener R y $\mathbf{t}$.

Al realizar los cálculos se obtiene la posición y orientación mediante la traslación ($\mathbf{t}$) y la rotación (R) del marcador en cada imagen respecto al sistema global de coordenadas.

2.3.3. Calibración de la cámara

El proceso de calibración de la cámara consiste en obtener los parámetros intrínsecos y extrínsecos. Las técnicas de calibración se basan en el uso de un conjunto de puntos formando un patrón conocido.

Conociendo el modelo de un patrón es posible realizar la calibración de la cámara.

Homografía

La homografía también conocida como transformación de perspectiva [16] determina la correspondencia entre dos figuras geométricas planas, de forma que a cada uno de los puntos de una de ellas le corresponden, respectivamente, a un punto de la otra figura como se observa en la figura (2.6).La homografía esta representada por una matriz 3 × 3 el cual usa coordenadas homogéneas.

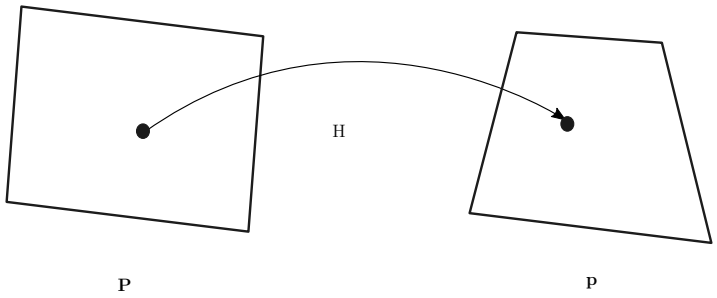


Figura 2.6: Homografía

Del modelo de la cámara obscura de la ecuación (2.10). La matriz aumentada esta compuesta por los parámetros extrínsecos $[R|\mathbf{t}]$. En este caso se esta trabajando con puntos en un plano que tiene coordenadas en x y y, por lo tanto se asume que $z = 0$.

$$\lambda \mathbf{p} = K[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]\mathbf{P}. \tag{2.17}$$

El punto $\mathbf{P}$ es ahora un punto bidimensional, pero se sigue representando así para indicar que es un punto de la escena o del objeto tridimensional. Entonces $\mathbf{p}$ y $\mathbf{P}$ son puntos sobre planos y la transformación que se aplica a ellos es la homografía H. Según

la ecuación (2.17) H esta compuesta por $K[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]$, esta tiene ocho grados de libertad ya que es invariante a un factor de escala λ. Entonces H también representa:

$$\lambda \mathbf{p} = H\mathbf{P}, \tag{2.18}$$

Para obtener la homografía se necesita tener la correspondencia de puntos entre dos imágenes. La correspondencia de puntos nos da la equivalencia entre $\mathbf{p}_i$ y $\mathbf{P}_i$, para n correspondencias y $i = 1, 2, \ldots, n$. Al expandir los dos lados de la ecuación (2.18) se obtiene:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.19}$$

Aquí son tres ecuaciones que se hacen dos al eliminar el factor de escala λ, resultando:

$$\begin{aligned} (h_{31}x + h_{32}y + h_{33})u &= h_{11}x + h_{12}y + h_{13} \\ (h_{31}x + h_{32}y + h_{33})v &= h_{21}x + h_{22}y + h_{23} \end{aligned} \tag{2.20}$$

Se puede fijar el factor de escala haciendo $h_{33} = 1$ y se acomodan las ecuaciones de forma que las incógnitas son el resto de los elementos de la homografía, en un vector $\mathbf{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}]$. Se forman el sistema de ecuaciones:

$$A\mathbf{h} = \mathbf{b} \tag{2.21}$$

La matriz A y el vector $\mathbf{b}$ se obtienen de expander las ecuaciones (2.20):

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xu & -yu \\ 0 & 0 & 0 & x & y & 1 & -xv & -yv \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.22}$$

Como se tienen ocho incógnitas se necesitan ocho ecuaciones y como se generan dos ecuaciones por cada correspondencia de puntos, entonces se necesitan al menos cuatro correspondencias de puntos para poder calcular una homografía.

Aunque directamente se puede aplicar este procedimiento, es absolutamente necesario poner las coordenadas de los puntos en ambas correspondencias a media cero y desviación estándar igual a uno. El cálculo directo de la ecuación (2.21) presente

inestabilidades numéricas, por las distintos valores y magnitudes de K, R y $\mathbf{t}$. Si se representan los puntos transformados como:

$$\mathbf{p}' = T_1\mathbf{p}, \qquad \mathbf{P}' = T_2\mathbf{P}, \tag{2.23}$$

Aquí se calcula la homografía entre las correspondencias normalizadas $\mathbf{p}'$ y $\mathbf{P}'$:

$$\lambda\mathbf{p}' = H'\mathbf{P}', \tag{2.24}$$

y al sustituir las expresiones anteriores se obtiene:

$$\begin{aligned} \lambda T_1\mathbf{p} &= H'T_2\mathbf{P}, \\ \lambda\mathbf{p} &= T_1^{-1}H'T_2\mathbf{P}, \end{aligned}$$

y finalmente la homografía $H = T_1^{-1}H'T_2$

Si se tienen más de cuatro correspondencias de puntos es necesario resolver un sistema sobredeterminado de ecuaciones que forman una matriz de tamaño $2n \times 8$, en donde n es el número de correspondencias de puntos entre el modelo y la imagen. Es mejor aplicar aquí la descomposición QR.

$$\begin{aligned} QR\mathbf{h} &= \mathbf{b}, \text{ aplicando ecuaciones normales:} \\ (QR)^T(QR)\mathbf{h} &= (QR)^T\mathbf{b}, \\ R^TQ^TQR\mathbf{h} &= R^TQ^T\mathbf{b}, \\ R^TR\mathbf{h} &= R^TQ^T\mathbf{b}, \\ R\mathbf{h} &= Q^T\mathbf{b}. \end{aligned}$$

La expresión al final resuelve $\mathbf{h}$.

2.3.4. Triangulación

Del modelo de la cámara obscura en (2.10), se tiene la pose i para cada imagen i, $i = \{1, 2, 3\}$, con lo que se tiene:

$$\lambda\mathbf{p}_i = K_i[R_i|\mathbf{t}_i]\mathbf{P}, \tag{2.25}$$

Al multiplicar los dos lados de la ecuación (2.25) por K_i^{-1} se tiene:

$$\lambda\hat{\mathbf{p}}_i = \lambda K_i^{-1}\mathbf{p}_i = [R_i|\mathbf{t}_i]\mathbf{P} = M\mathbf{P} \tag{2.26}$$

Expandiendo la expresión anterior se obtiene:

$$\lambda \begin{bmatrix} \hat{u}_i \\ \hat{v}_i \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.27}$$

Al resolver (2.27) se tiene la siguiente expresión:

$$\begin{aligned}\lambda\hat{u}_i &= m_{11}x + m_{12}y + m_{13}z + m_{14}\\ \lambda\hat{v}_i &= m_{21}x + m_{22}y + m_{23}z + m_{24}\\ \lambda &= m_{31}x + m_{32}y + m_{33}z + m_{34}\end{aligned}$$

Al eliminar el valor de λ, de la tercera expresión en las primeras dos se obtiene:

$$\begin{bmatrix}\hat{u}_i m_{31} - m_{11} & \hat{u}_i m_{32} - m_{12} & \hat{u}_i m_{33} - m_{13}\\ \hat{v}_i m_{31} - m_{21} & \hat{v}_i m_{32} - m_{12} & \hat{v}_i m_{33} - m_{23}\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix} = \begin{bmatrix}m_{14} - \hat{u}m_{34}\\ m_{24} - \hat{v}m_{34}\end{bmatrix} \tag{2.28}$$

La ecuación (2.28) representa dos ecuaciones por cada punto tridimensional visto en dos imágenes a las que se les conoce la pose.

Para obtener el punto tridimensional por este método se necesitaría un mismo punto visto en tres imágenes, se tendría un sistema de cuatro ecuaciones y tres incógnitas (las coordenadas del punto tridimensional). Este es el método más sencillo de triangulación.

Capítulo 3

Propuesta de solución

En el presente capítulo se describe el funcionamiento del prototipo del ojo de halcón. En la sección 3.1 se presenta la estructura en donde se coloca la cámara y dentro de la cual se mueve el . En la sección 3.2 se presenta los dispositivos usados para el sistema. En la sección 3.3 se presenta el clasificador propuesto para poder reconocer el marcador a partir de las imágenes tomadas por la cámara. En la sección 3.4 se describe el marcador utilizado, se presenta el procedimiento propuesto para llevar a cabo la obtención de los puntos del marcador, la correspondencia de puntos con la imagen y su pose. En la sección 3.5 se describe la manera en que se lleva a cabo el procesamiento de las imágenes en la Raspberry. En la sección 3.6 se muestra la obtención del foco.

3.1. Estructura

Se propone una estructura para delimitar el área de visión y en donde se coloquen los dispositivos que conforman el sistema como son la cámara, la raspberry y un foco para proporcionar iluminación. La estructura esta hecha de madera, debido al precio y durabilidad de este material. El plano de la estructura propuesta se muestra en la figura 3.1. Se compone de una base cuadrada de 40 cm y una altura de 60 cm.

La estructura también se compone de iluminación. Un foco empotrable de luz LED fría de 3 vatios con difusor se encarga de proporcionar la iluminación, éste se ubica en el centro de la tapa de la estructura. Las características que posee este foco son importantes para la adquisición de las imágenes. El foco se puede empotrar en la tapa de la estructura, por lo que el espacio del área de visión no se ve afectado por incluir iluminación a la estructura, otra característica importante es la presencia del difusor, el cual difunde el haz de luz de manera uniforme y evita la presencia de reflejos para que la luz no se localice en un sólo lugar y distorsione la imagen.

La cámara es colocada en la tapa en uno de los lados del cuadrado en la parte de en medio, lo cual proporciona un campo de visión de todo la estructura. El área de visión de la cámara al fondo de la estructura es de 50 cm. La Raspberry se coloca en la parte

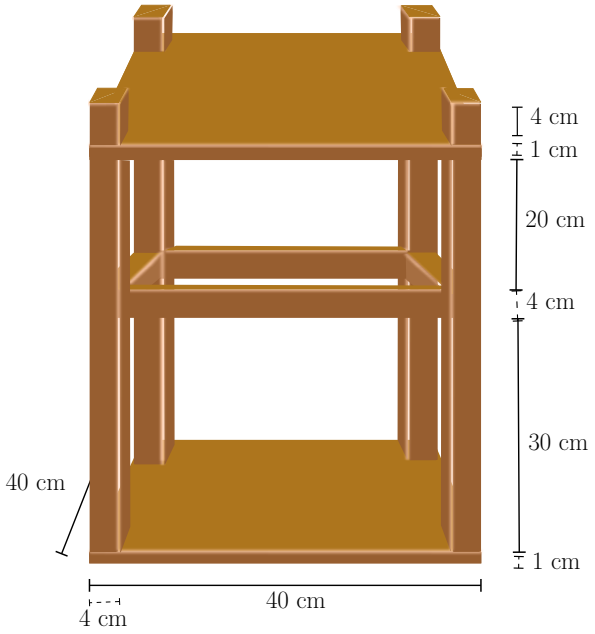


Figura 3.1: Plano de la estructura del sistema.

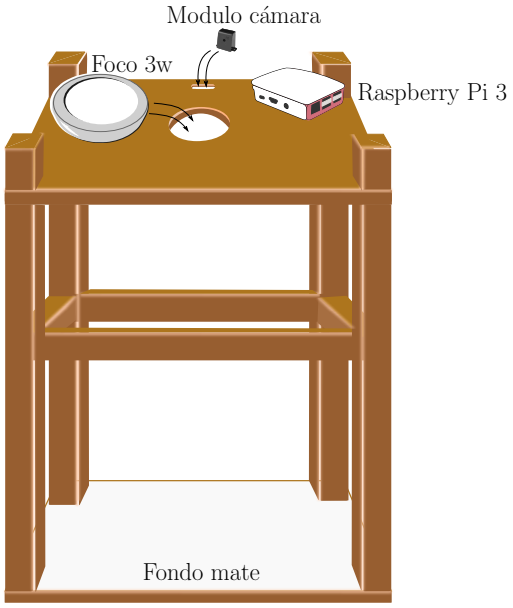


Figura 3.2: Elementos del sistema.

superior de la tapa, conectada a la cámara por medio del puerto CSI.

El fondo de la estructura se pinta de un color blanco mate al realizar esto se facilita la eliminación de sombras y de reflejos a causa de la iluminación presente.

En la figura 3.2 se pueden ver todos estos elementos en conjunto.

3.2. Dispositivos del sistema

Se propone el uso de una computadora en una sola tarjeta (CST) modelo Raspberry Pi 3 modelo B, debido a que son dispositivos de bajo costo.

Un sistema de computadora de una sola tarjeta se refiere a que todos los elementos que componen a una computadora como lo son el procesador, dispositivos de entrada y salida,la RAM y dispositivos de red están integrados en una tarjeta. Las diferencias que presentan las CST de sistemas de computadoras tradicionales es que no son modulares, su hardware no puede ser actualizado ni remplazado, son portables, tienen un menor tamaño y su costo es menor. Las CSTs se usan como computadoras de bajo costo en entornos académicos y de investigación. El uso de computadoras de una sola tarjeta en sistemas integrados es muy frecuente, y muchas personas y organizaciones han desarrollado y lanzado productos totalmente funcionales basados en CSTs.

Raspberry Pi es una serie de computadoras en una sola tarjeta, cabe en la palma

de una mano, de bajo costo (alrededor de $1,000 pesos) creado con la intención de promover el uso de la computación. Se propone el uso de una computadora de la serie Raspberry Pi, la computadora en una sola tarjeta Raspberry Pi 3 modelo B [8], en la figura 3.3 se pueden apreciar los componentes y su tamaño. Esta fue lanzada en febrero del 2016, es de las computadoras de las serie Pi que cuenta con mayor procesamiento, lo cual es necesario para aplicaciones de visión por computadora debido a que se manejan en tiempo real.

Figura 3.3: Raspberry Pi 3 modelo B.

El precio de esta CTS está alrededor $45.57 dolares [17], el cual es de menor costo considerando el precio de una computadora portátil o de escritorio lo cual lo hace accesible a una mayor cantidad de personas, otra ventaja es el tamaño el cual es pequeño permitiendo que sea un dispositivo portable. Una desventaja que tiene es la cantidad de recursos que posee no equipará a una computadora no cuenta con mucha capacidad de almacenamiento y su capacidad de procesamiento es menor. En la tabla 3.2 se muestran las especificaciones de la Raspberry Pi 3 modelo B.

SoC broadcom	BCM2837
CPU	Quad Core 1.2GHz 64bit
Memoria	1 GB RAM
Puertos	40 pines GPIO
	4 puertos USB
	Puerto CSI (cámara)
	Puerto DSI (pantalla touch)
	Puerto MicroSD
Salida de vídeo	HDMI
Almacenamiento	Micro SD
Red	10/100 Mbps Ethernet, Bluetoothy WiFi
Fuente de poder	5V MicroUSB
Tamaño	85.60mm x 53.98 mm

Cuadro 3.1: Especificaciones Raspberry Pi 3 Modelo B.

La adquisición de fotos se realizó con ayuda del módulo cámara versión 2 de Raspberry el cual se conecta a través del puerto CSI de la Raspberry. Esta cámara se vende como aditamento para la Raspberry. Como especificaciones de esta cámara cuenta con un sensor Sony IMX219 de 8 megapíxeles que es una cámara usada en dispositivos móviles la cual cuenta con soporte de 1080p30, 720p60 y modo de video VGA90 [9]. Este hardware es más fácil de utilizar porque ya viene soportado por el vendedor de la Raspberry Pi, además de tener un pequeño tamaño en comparación de una webcam. El costo aproximado es de $32.55 dolares [17].

Figura 3.4: Módulo cámara Raspberry Pi.

3.3. Clasificador

Se adquirieron 245 imágenes del marcador, se realizó la segmentación de los objetos presentes y de manera visual se identificaron seis clases, estas clases tienen las características siguientes:

1. Barra Superior: Está siempre presente en la imagen una barra en la parte superior, debido a la estructura en donde se coloca la cámara.

2. Puntos: Son todos aquellos objetos que presentan áreas muy pequeñas dentro de la imagen, son irregularidades de la imagen que no se lograron eliminar con la eliminación de ruido o son sombras diminutas.

3. Barras: Al igual que la barra superior debido a la estructura siempre están presentes dos barras en la parte inferior de la imagen, una de lado izquierdo y otra del lado derecho.

4. Mano: En la imagen siempre se ve presente la mano del usuario ya que sostiene al marcador fiducial.

5. Marcador: Es el objeto de interés, esta conformado por un cuadrado con dos triángulos interiores.

6. Sombras: Esta clase se compone de todos aquellos objetos formados debido a la iluminación, son las sombras creadas por los otros objetos presentes en la imagen. Estos objetos no tienen características bien definidas ya que son objetos de distintos tamaños y que no presentan una forma definida.

Un ejemplo de las clases que fueron identificadas se muestran en la figura 3.5.

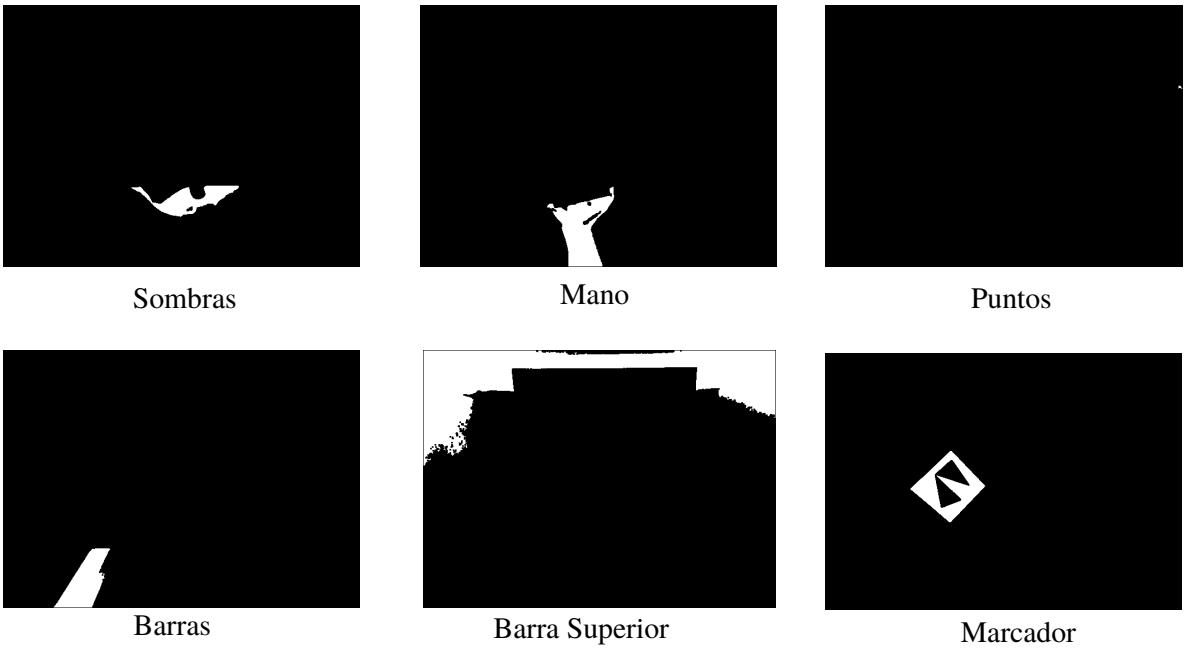


Figura 3.5: Clases identificadas en la imagen.

A fin de identificar el marcador fiducial se creo un clasificador usando las muestras de las seis clases antes mencionadas barra superior, puntos, barras, mano, marcador y sombras. Con el propósito de identificar las características a elegir de los objetos se obtuvieron las características para realizar el entrenamiento del clasificador con el 60 % de las imágenes capturadas y se eligieron las características. Las características elegidas son el área y el primer momento de Hu del objeto y el primer y tercer momentos de Hu del perímetro.

En el diagrama 3.6 se observa los pasos por los que está compuesto el clasificador para encontrar el marcador dentro de la captura de una imagen. Se puede observar como se utilizan todas las características antes mencionadas y como el clasificador diferencia los objetos por medio de las características numéricas. Este clasificador se obtuvo por medio de las imágenes de entrenamiento.

El clasificador completo esta formado por cinco etapas, a continuación se describe de manera breve el clasificador:

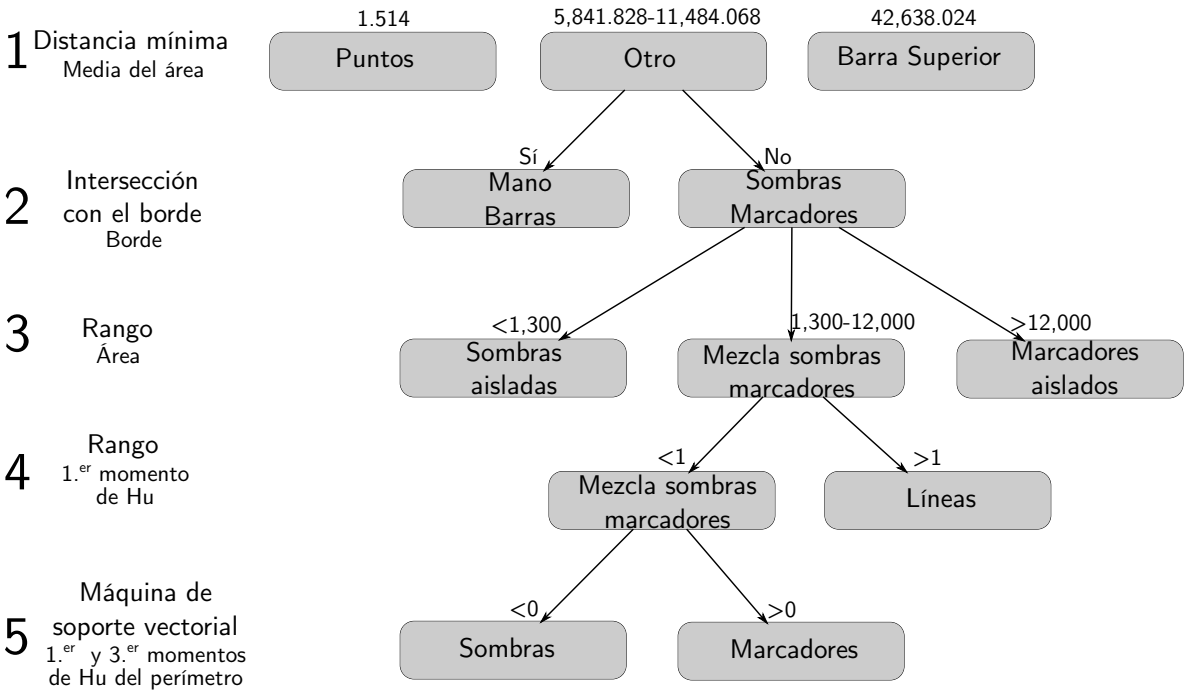


Figura 3.6: Diagrama clasificador.

1. Un clasificador de mínima distancia usando como característica el área. Se utiliza la media obtenida en el entrenamiento de las clases. Esta clasificación resulta en tres clases.

   - La clase *puntos* presenta objetos con un área muy pequeña.
   - La clase *otro* que esta conformada por las subclases *mano, marcador, sombras y barras* que presentan objetos con áreas similares.
   - La clase *barra superior* contiene objetos con el área muy grande.

2. La clase *otro* se debe separar en subclases basándose en la intersección del perímetro del objeto con el borde de la imagen:

   - La subclases *mano y barras* siempre tocan el borde de la imagen.
   - La subclases *marcador y sombras* no tocan el borde de la imagen.

3. Las subclases marcador y sombra se separan a su vez en tres subclases conforme el área de los objetos:

   - Si el área del objeto esta en un rango de 0 a 1,300 pertenece a la subclase *sombras aislados*
   - Si el área del objeto esta en un rango de 1,300 a 12,000 pertenece a la subclase *mezcla de marcadores y sombras.*

- Si el área del objeto esta en un rango mayor a 12,000 pertenece a la subclase *marcadores aislados*. Si este es el caso, se ha encontrado el marcador.

4. La subclase *mezcla de marcadores y sombras* se divide en dos subclases por la característica del primer momento de Hu.
   - Si el primer momento de Hu es mayor a 1, el objeto es clasificado en *líneas*.
   - Si el primer momento de Hu es menor a 1, es una *mezcla de sombras y marcadores*.
5. Finalmente la subclase *mezcla de marcadores y sombras* se separa en las clases *marcadores* y *sombras* usando una máquina de soporte vectorial lineal.

Con el 40 % de imágenes que no fueron utilizadas para realizar el entrenamiento, se realizan pruebas de funcionamiento del clasificador. Los valores de las características que tienen estas imágenes no se encuentran en la realización del entrenamiento, así que si las imágenes son clasificadas de manera correcta se puede conocer si el clasificador tiene un buen funcionamiento.

3.3.1. Clasificador de distancia mínima

El reconocimiento de objetos se hace por medio de las características o rasgos numéricos que presenten los objetos contenido en las imágenes. Se pude hacer la analogía que entre menor distancia haya de los valores de las características existe una mayor semejanza entre los objetos. La primera etapa del clasificador es un clasificador de distancia mínima usando la característica del área de los objetos. Siguiendo al analogía de distancia y semejanza se puede deducir que si el área de un objeto es cercana a otra tiene áreas parecidas, lo que se puede deducir en que son objetos parecidos, por el contrario la similitud del objeto se reduce a medida que se alejan las áreas de los objetos.

La clasificación por distancia mínima consiste en:

1. Medir la distancia entre el patrón que se desea reconocer y las medias obtenidas con el conjunto de entrenamiento.
2. Comparar e identificar la menor distancia y la etiqueta de la clase correspondiente.
3. Asignar la clase al objeto que se desea reconocer.

Las medias obtenidas con el conjunto de entrenamiento son las siguientes:

En la figura 3.7 realiza la gráfica correspondiente a la media de las áreas obtenidas, estos valores se muestran la tabla 3.2.

- La clase *puntos* presenta objetos con un área muy pequeña.

Clase		Media áreas
Barra Superior		42638.024
Otro	Barras	5841.828
	Mano	11484.068
	Sombras	636.158301
	Marcador	9140.716
Puntos		1.514

Cuadro 3.2: Valores de medias calculadas en la etapa de entrenamiento.

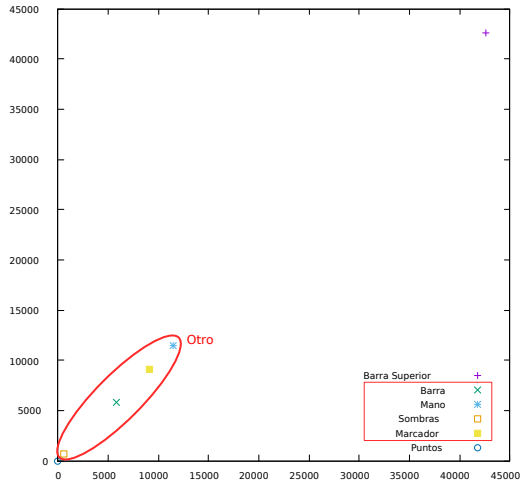


Figura 3.7: Medias de las áreas.

- Las clases *mano, marcador, sombras y barras* presentan objetos con áreas similares, lo que hace que no se pueda obtener una clasificación exacta de éstas usando este método de clasificación, por lo cual se decidió convertir estas clases en subclases y crear una sola clase llamada *otro*.
- La clase *barra superior* contiene objetos con el área muy grande.

Se utiliza la distancia euclidiana con el fin de medir la distancia que hay entre las medias obtenidas y el objeto a reconocer. La distancia euclidiana (ED) es una técnica de clasificación basado en distancia para la clasificación de imágenes. En los sistemas biométricos se usa esta técnica de clasificación para determinar la precisión de sistemas biometricos, como en [18, 19] que buscan encontrar la autenticidad de firmas.

Distancia euclidiana

La distancia euclidiana [20] es la función que mide la distancia entre dos puntos en un espacio n-dimensional. Formalmente, la distancia euclidiana se puede expresar

matemáticamente como:

$$d_E(p_1, p_2) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_3)^2 + (a_n - b_n)^2} \tag{3.1}$$

donde a y b son dos puntos en un espacio n-dimensional.

En este caso representaremos los datos en un espacio bidimensional (x,y) y el objetivo es calcular la distancia entre los valores del punto 1 (p_1) y el punto 2 (p_2). En la siguiente figura 3.8 se muestra la representación de como se obtiene la distancia euclidiana de dos puntos en un espacio bidimensional.

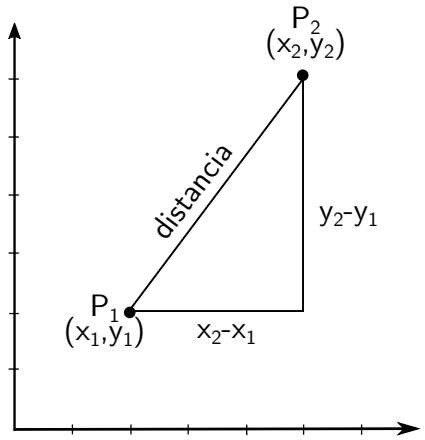


Figura 3.8: Distancia euclidiana.

De la ecuación (3.1) de distancia euclidiana n-dimensional, podemos deducir la siguiente fórmula para calcular la distancia euclidiana bidimensional.

$$d_E(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.2}$$

A fin de facilitar el procesamiento del algoritmo en la computadora se calcula la distancia euclidiana al cuadrado. Despejamos la ecuación (3.2) para obtener la distancia al cuadrado.

$$d_E(p_1, p_2)^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 \tag{3.3}$$

En este caso x y y tienen el mismo valor, entonces podemos decir que la ecuación es equivalente a:

$$d_E(p_1, p_2)^2 = 2(x_2 - x_1)^2 \tag{3.4}$$

Para realizar la menor cantidad de operaciones reducimos la ecuación y quitamos la exponencial, por lo que al final tenemos:

$$\frac{d_E(p1, p2)^2}{2} = (x_2 - x_1) * (x_2 - x_1) \tag{3.5}$$

Se calcula la distancia con las medias de las seis clases y se comparan con el fin de encontrar la mínima distancia. Se etiqueta al objeto con la clase que tenga la distancia mínima.

En el caso de que el resultado de la mínima distancia sea (*Barra Superior*) o (*Puntos*) las imágenes son descartadas, en caso de que la distancia mínima corresponda a alguna de las otras clases será identificada como la clase (*Otro*) y se continuará con el proceso de clasificación de estas imágenes.

3.3.2. Intersección con el borde

El proceso de clasificación continua con revisar la intersección con el borde. Se revisa el borde de la imagen con el fin de buscar si un objeto lo interseca. Esta es una característica binaria por lo que se tienen las dos siguientes posibilidades:

- Tiene intersección con el borde: Las subclases *mano y barras* siempre tocan el borde de la imagen.
- No tiene intersección con el borde: Las subclases *marcador y sombras* no tocan el borde de la imagen.

En la figura 3.9 se muestran varias imágenes de ejemplo que intersecan el borde.

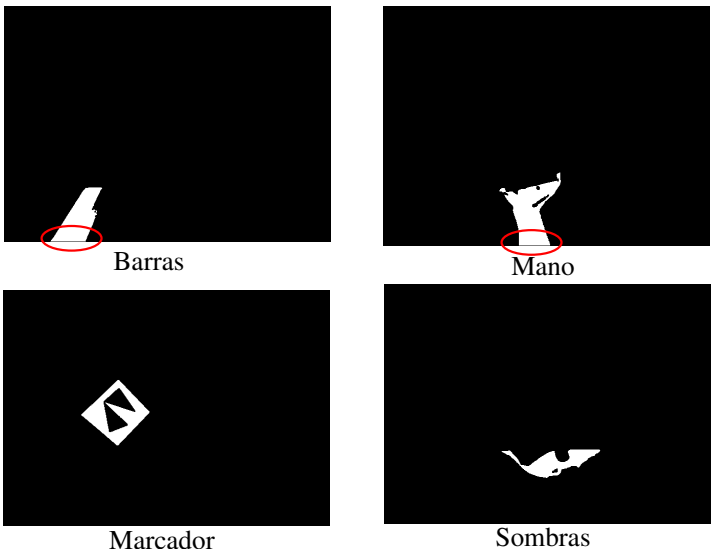


Figura 3.9: Intersección con el borde.

Con base en esta característica se descartan las subclases *mano y barras*, debido a que éstos objetos siempre tocan el borde y se continua con el proceso de clasificación de las imágenes.

3.3.3. Clasificación por rango para el área de los objetos

Hasta este momento de la clasificación se tienen dos subclases: marcador y sombras.

Se obtiene el área de los objetos que no han sido clasificados, para poder realizar la separación de las sombras con el marcador.

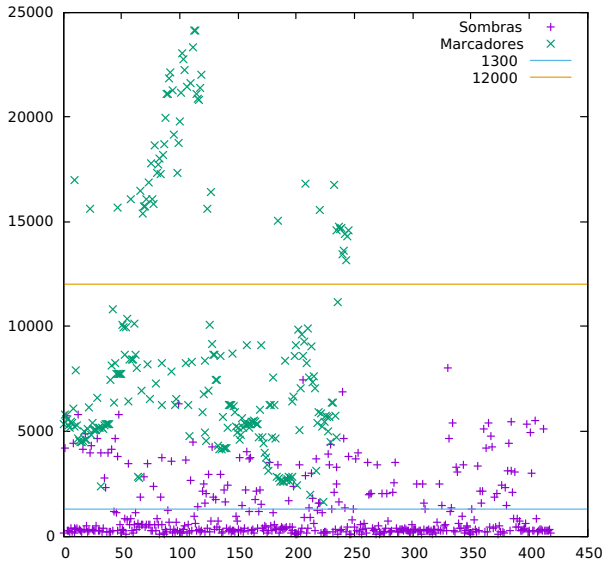


Figura 3.10: Clasificación en base al área.

En la figura 3.10 se muestra la separación que se realiza de acuerdo al área de los objetos. Antes de la hacer esta clasificación contamos con las subclases marcador y sombra, pero después de la realización de esta clasificación esas subclases se separan a su vez en tres subclases: *marcadores aislados, sombras aisladas, una mezcla de marcadores y sombras.*

Los objetos que tenemos pueden se clasifican de la siguiente manera:

- *Sombras aisladas*: Objetos con área menor a 1,300, compuesto por sombras únicamente.

- *Mezcla de sombras y marcadores*: Objetos que presenten una área entre 1,300 y 12,000, en esta subclase se encuentran revueltos tanto sombras como marcadores. Se deberá continuar con el proceso de clasificación en esta clase.

- *Marcadores aislados*: Aquellos objetos que tengan un área mayor a 12,000, esta subclase se compone de marcadores. Estos marcadores tienen la característica de estar a una menor distancia por lo cuál su área es mayor, es importante aclarar que no deben de tener intersección con el borde.

3.3.4. Clasificación por rango usando el primer momento de Hu

En esta cuarta etapa de la clasificación se usa como característica el primer momento de Hu de los objetos. Se calculo el primer momento de Hu de los objetos y se clasifica de acuerdo al rango de este valor.

En 1961 Hu propuso momentos basados en métodos algebraicos invariantes en [21]. Usando momentos geométricos invariantes, el derivo un conjunto de momentos invariantes los cuales tenían las propiedades de ser invariantes en imágenes a pesar de utilizar las transformaciones de rotación, traslación y escalamiento. Estos momentos son conocidos en la actualidad como momentos de Hu, los cuales son un conjunto de siete momentos invariantes que cuantifican la forma de un objeto.

Los momentos geométricos de orden $p+q$ se calculan con la ecuación (3.6).

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y). \tag{3.6}$$

Los momentos centralizados se obtienen mediante la ecuación (3.7).

$$\mu_{pq} = \sum_x \sum_y (x-\overline{x})^p y (y-\overline{y})^q f(x,y) \tag{3.7}$$

en donde

$$\overline{x} = \frac{m_{10}}{m_{00}}, \quad \overline{y} = \frac{m_{01}}{m_{00}}$$

Los momentos normalizados se calculan con la expresión 3.8:,

$$\eta_{pq} = \frac{\mu_{pq}}{m_{00}^{\gamma}}, \quad \gamma = \frac{p+q}{2} + 1, \quad \gamma > 2 \tag{3.8}$$

El primer momento de Hu se calcula con la ecuación (3.9).

$$\phi_1 = \eta_{20} + \eta_{02} \tag{3.9}$$

en donde η_{pq} corresponde a los momentos centralizados normalizados. Los momentos centrales normalizados utilizados para calcular el primer momento de Hu son los siguientes:

$$\eta_{20} = \frac{m_{20} - \frac{m_{10}^2}{m_{00}}}{m_{00}^2},$$
$$\eta_{02} = \frac{m_{02} - \frac{m_{01}^2}{m_{00}}}{m_{00}^2},$$

Los objetos cuyos cálculo del primer momento de Hu es un valor mayor a uno son líneas diagonales, la mayoría ocasionadas por la presencia de sombras. Existe también la posibilidad que se vea la presencia de un marcador sin forma cuadrada debido a la unión del cuadrado y los triángulos internos, en este caso este marcador también es descartado, debido a que no está bien formado el clasificador no podría identificarlo como un marcador. En la figura 3.11, se muestran ejemplos de este caso en el cual el momento de Hu de los objetos es mayor a uno.

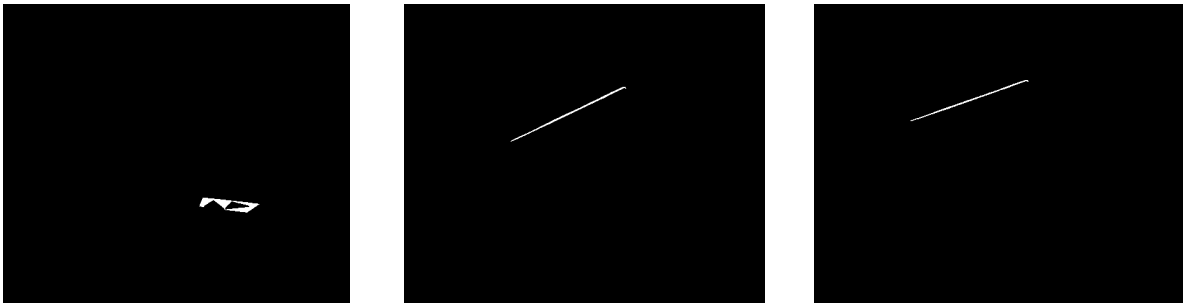


Figura 3.11: Objetos cuyo cálculo del primer momento de Hu es mayor a 1.

Por otro lado, si el valor calculado es menor de uno se continua con el proceso de clasificación, ya que los objetos presentes en la clase sigue conteniendo una mezcla de sombras y de marcadores.

3.3.5. Máquina de soporte vectorial

En esta etapa se usa como característica el primer y tercer momento de Hu del perímetro de los objetos y con ayuda de una máquina de soporte vectorial se clasifica los objetos que aún no hayan sido descartados. Esta etapa es la última del clasificador, en este punto sólo hay dos posibles opciones de los objetos que resta por clasificar, que sean sombras o que sean marcadores. Esta máquina vectorial ayuda con esta clasificación.

Las máquinas de soporte vectorial son una de las formas más usadas para clasificación. Es una máquina de aprendizaje para problemas de clasificación binaria. La máquina asigna vectores de entrada de forma no lineal a un espacio de características de muy alta dimensión. Se construyó un vector para tener una decisión lineal [22].

El hiperplano que permite separar dos clases no es único, es decir, existen infinitos hiperplanos separables, representados por todos aquellos hiperplanos que son capaces de cumplir las restricciones de separar las dos clases como se observa en la 3.12.

A pesar de que existen diferentes hiperplanos que realizan esta separación para su clasificación, existe un hiperplano óptimo el cual se obtiene por medio de métodos de

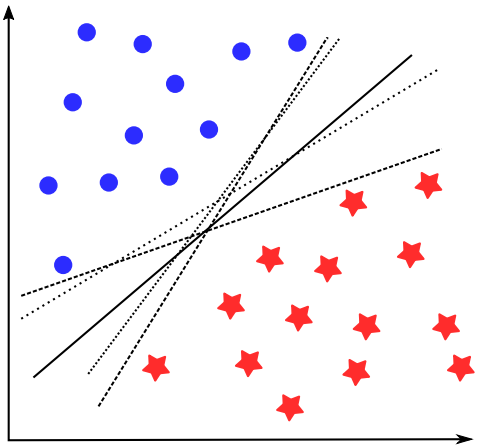


Figura 3.12: Hiperplanos de separación en un espacio bidimensional de un conjunto de ejemplos separables en dos clases.

optimización. Un hiperplano es óptimo cuando equidista al elemento más cercano de cada clase.

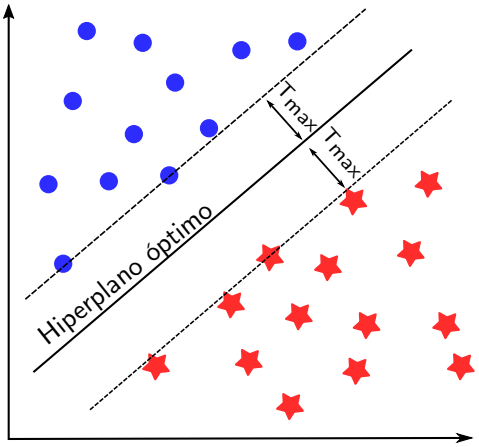


Figura 3.13: Hiperplanos de separación óptimo.

La máquina vectorial se realizó con ayuda de la biblioteca LIBSVM disponible en [1]. LIBSVM es un software integrado para clasificación de vectores de soporte, estimación de distribución y además soporta clasificación multiclase [23].

Se obtuvieron con ayuda de la biblioteca LIBSVM los valores correspondientes para la separación de las dos clases, y se programó con esos valores la máquina de soporte vectorial con un kernel lineal para obtener un clasificador binario.

El primer y tercer momentos de Hu fueron las características que se ocuparon para

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm

separar las clases *marcadores* y *sombras*. Se obtiene el perímetro de los objetos y se calculan los momentos de Hu, en la subsección anterior 3.3.4 se muestran las ecuaciones para obtener el cálculo de estos. El primer momento de Hu se calcula con la ecuación en (3.9) y el tercer momento de Hu se calcula con la ecuación (3.10).

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{3.10}$$

en donde los momentos centrales normalizados se calculan a partir de la ecuación (3.8) vista en la cuarta etapa de clasificación.

$$\eta_{30} = \frac{m_{30} - 3\frac{m_{10}m_{20}}{m_{00}} + 2\frac{m_{10}^3}{m_{00}^2}}{m_{00}^{5/2}},$$

$$\eta_{12} = \frac{m_{12} - 2\frac{m_{01}m_{11}}{m_{00}} - \frac{m_{10}m_{02}}{m_{00}} + 2\frac{m_{01}^2 m_{10}}{m_{00}^2}}{m_{00}^{5/2}},$$

$$\eta_{21} = \frac{m_{21} - 2\frac{m_{10}m_{11}}{m_{00}} - \frac{m_{01}m_{20}}{m_{00}} + 2\frac{m_{10}^2 m_{01}}{m_{00}^2}}{m_{00}^{5/2}},$$

$$\eta_{03} = \frac{m_{03} - 3\frac{m_{01}m_{02}}{m_{00}} + 2\frac{m_{01}^3}{m_{00}^2}}{m_{00}^{5/2}}$$

Para realizar esta clasificación a los objetos se debe obtener el perímetro de los objetos. Para obtener el perímetro de los objetos se realiza la dilatación del fondo y se obtiene el contorno del objeto. Una vez que se tiene el perímetro de los objetos se obtienen el primer y tercer momento de Hu y con estos valores se realiza la clasificación por medio de la máquina de soporte vectorial.

3.4. Marcador

El marcador es un elemento único que facilita su identificación, se conoce su forma previamente a la captura y cuando es capturado se reconoce debido a un entrenamiento previo o algún algoritmo de identificación. El marcador se usa en el área de realidad aumentada y robótica, como lo son los ARTags [4], AprilTags [5], BinARyD [6] y los usados en el ARToolkit [7]. En la siguiente figura 3.14 se muestra algunos ejemplos de estos marcadores:

El marcador nos aporta información para conocer la pose del objeto. Con técnicas de procesamiento de imagen se aislan los pixeles correspondientes al marcador y obtiene

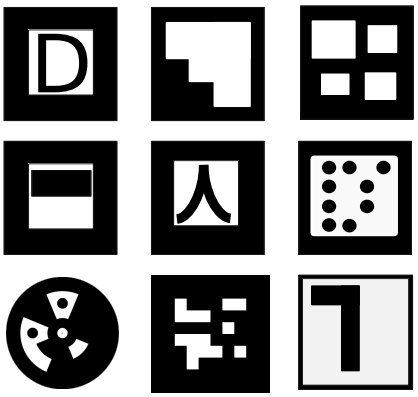


Figura 3.14: Ejemplo de marcadores usados en el área de realidad aumentada y robótica.

su ubicación en el área bidimensional.

3.4.1. Marcador utilizado

El marcador utilizado es un marcador fiducial. Se propone aquí el uso de un tipo de marcador compuesto de triángulos y con los que se obtiene automáticamente un identificador y la correspondencia de puntos con el modelo del mismo marcador.

El marcador propuesto se puede ver en la figura 3.15, el cual es un marcador cuadrado negro que tiene un borde blanco el cual presenta un gran contraste, lo cual permite una mejor separación de objetos. Tiene en su interior seis puntos que conforman dos triángulos, los cuales contienen información para calibrar la cámara y encontrar la pose del marcador. Este marcador fue impreso de tamaño de 10 cm $\times$ 10 cm, y colocado sobre una superficie plana rígida, para evitar deformaciones de la imagen.

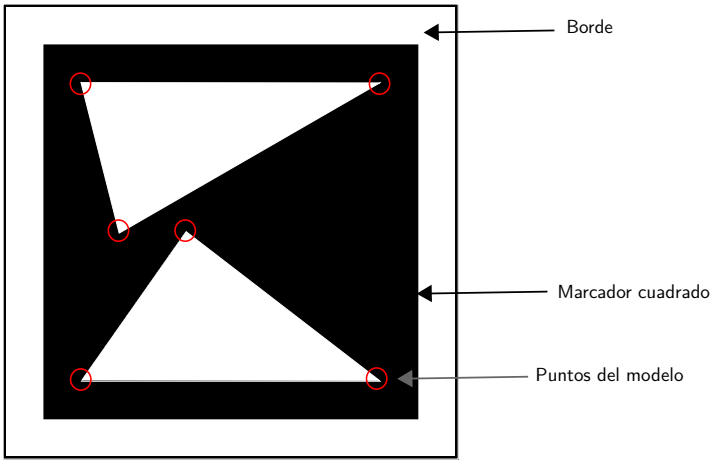


Figura 3.15: Marcador utilizado.

Los seis puntos que se utilizaron para el marcador son $(0, 200)$, $(0, 0)$, $(25, 100)$,

$(70, 100)$, $(200, 0)$, $(200, 200)$. Esta secuencia de puntos se escogió porque resulta en una *perturbación máxima* [24] de 10.61, este valor significa que dos puntos tienen que moverse 10.61 unidades (con respecto al valor de la secuencia de los puntos) para cambiar su *tipo de orden* [24, 25]. Entonces dos puntos pueden moverse $10.61/200 \approx 0.053$, esto es, 5.3 % con respecto al tamaño de todo el marcador que es de 200×200. El tipo de orden es una medida combinacional de la orientación de los puntos. El tipo de orden no cambia si los puntos, que están sobre el plano, se trasladan, rotan o se les aplica una proyección en perspectiva [24].

Con la secuencia de puntos del marcador, en ese mismo orden que se escribieron en el párrafo anterior, su matriz lambda [24, 25] es: 432100124313231222223012441320. Esta matriz lambda significa cuántos puntos están a la izquierda de la línea dirigida del punto i al j. No hay más de dos puntos colineales en la secuencia de puntos usada para el marcador, por lo tanto para cualquier entrada en la matriz lambda se cumple $\lambda(j, i) = n - 2 - \lambda(i, j)$, donde n es el número de puntos usado, por lo que se podría usar solo la matriz superior triangular, sin los valores de la diagonal principal, para representar una matriz lambda. Nótese que $\lambda(i, i)$ no está definida, esto es, no se puede trazar una línea usando un mismo punto.

La perturbación máxima y la matriz lambda se calcularon con los programas disponibles en [2].

Si los puntos están guardados en el archivo `puntos.txt`, su perturbación máxima se calculó como:

```
$ ./Scripts/ComputesMaximalPerturbation/computesMP puntos.txt
10.606602
```

Y su matriz lambda se calculó como:

```
$ ./Scripts/ComputesLambdaMatrix/computesLambdaMatrix puntos.txt 1
Labels: 0 1 2 3 4 5
432100124313231222223012441320
```

Como son seis puntos solo se tiene un dígito para codificar cada entrada en la matriz lambda, y es fácil decodificar la secuencia anterior en la matriz en la expresión (3.11). Los puntos se pueden visualizar en la figura 3.16 y están etiquetados en la forma en que se calculó la matriz (3.11); con la ayuda de esta figura se pueden verificar cada entrada en la matriz (3.11).

[2] http://cs.cinvestav.mx/~fraga/OTT/Data.zip

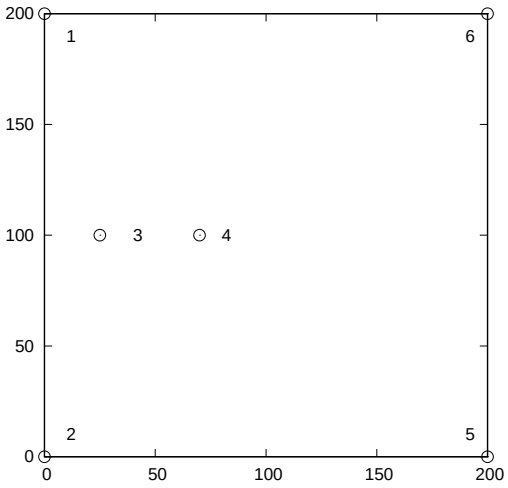


Figura 3.16: Puntos usados en el marcador. Los puntos están etiquetados para obtener la matriz en la expresión (3.11).

$$\begin{bmatrix} - & 4 & 3 & 2 & 1 & 0 \\ 0 & - & 1 & 2 & 4 & 3 \\ 1 & 3 & - & 2 & 3 & 1 \\ 2 & 2 & 2 & - & 2 & 2 \\ 3 & 0 & 1 & 2 & - & 4 \\ 4 & 1 & 3 & 2 & 0 & - \end{bmatrix} \tag{3.11}$$

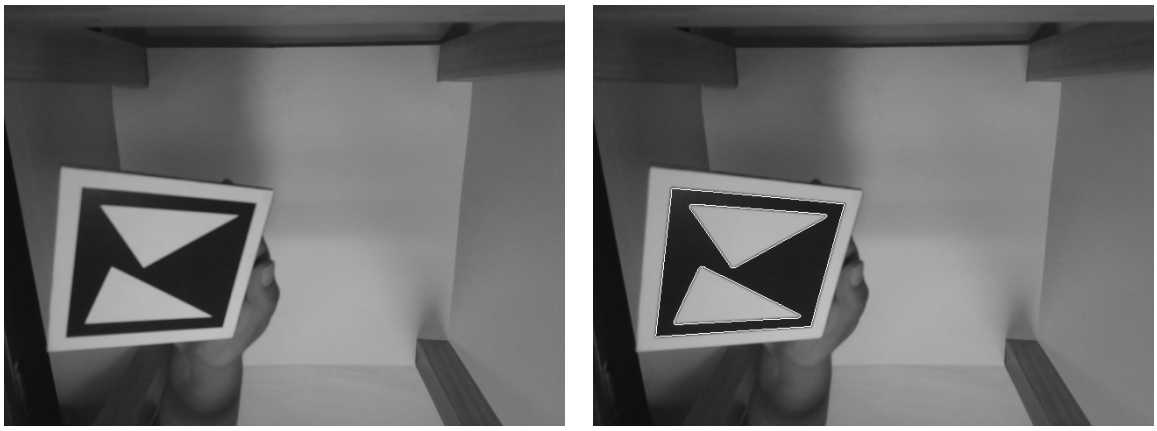
En la matriz en la expresión (3.11), una entrada con "$-$"significa que no está definido su valor.

3.4.2. Emparejamiento de puntos usando el tipo de orden

Para calibrar la cámara y también para encontrar la pose del marcador es necesario emparejar los puntos del modelo (este modelo es el marcador mostrado en la figura 3.16) y los puntos en la imagen. Este proceso se describirá aquí con un ejemplo.

La cámara toma la imagen mostrada en la figura 3.17(a). En este imagen se detecta el marcador y los dos triángulos que se dibujan en la figura 3.17(b). La detección del marcador se describió en la sección XX y la detección de los triángulos en la sección YY. Los vértices de los dos triángulos detectados son:

```
112.227866 226.328156
156.328855 301.611061
271.901517 238.626553
122.599231 296.314231
```



a) Imagen original

b) Imagen con el marcador y los triángulos detectados

Figura 3.17: Una imagen con el marcador

```
 89.391626 364.155310
242.480008 355.930387
```

Como se nota, estos vértices están calculados son una precisión subpixel.

En la imagen, el sentido del eje y es hacia abajo, para situar el centro del sistema de coordenadas en la esquina inferior izquierda de la imagen y el sentido del eje y hacia arriba se hace la transformación

$$y' = 479 - y, \tag{3.12}$$

y las coordenadas x no sufren cambios. Realizando esta transformación los puntos de los triángulo quedan como:

```
112.227866 252.671844
156.328855 177.388939
271.901517 240.373447
122.599231 182.685769
 89.391626 114.844690
242.480008 123.069613
```

Transformamos estos puntos a coordenadas enteras (al pixel más cercano) y quedarían:

```
112 253
156 177
272 240
123 183
 89 115
242 123
```

El cambio de los valores de los puntos a valores enteros no cambia su tipo de orden. El marcador tiene una máxima perturbación mucho más grande que el redondeo al pixel más cercano.

Si calculamos la matriz lambda de este conjunto de números (los guardamos en el archivo `pimagen.txt`) resulta en:

```
./Scripts/ComputesLambdaMatrix/computesLambdaMatrix pimagen.txt 1
Labels: 0 4 3 1 5 2
432100124313231222223012441320
```

Note que la matriz lambda es la misma que la obtenida del marcador (en la sección anterior).

Para poner los puntos en la misma secuencia de entrada de los puntos del marcador, primero tenemos que los puntos se enumeran como se leen, y segundo hay que ordenarlos en la secuencia 0 4 3 1 5 2 (con respecto al orden en que se leen). Para claridad, repetimos la secuencia de puntos y escribimos su orden:

```
0 112 253
1 156 177
2 272 240
3 123 183
4  89 115
5 242 123
```

La secuencia 0 4 3 1 5 2 se obtiene como:

```
0 112 253
4  89 115
3 123 183
1 156 177
2 272 240
5 242 123
```

Si calculamos de nuevo la matriz lambda a este conjunto de puntos (están guardados ahora en el archivo `pimagen2.txt`) obtenemos:

```
./Scripts/ComputesLambdaMatrix/computesLambdaMatrix pimagen2.txt 1
Labels: 0 1 2 3 5 4
432100124313231222223012441320
```

Y la secuencia está igual que los puntos del marcador. Esto significa que hay una correspondencia uno a uno de los puntos del marcador con los de la imagen, en el orden en que se leen. Esto se puede garantizar siempre y cuando solo exista una sola matriz lambda [24].

Los puntos en el archivo `pimagen2.txt` ya están empatados con los puntos del modelo del marcador $(0, 200)$, $(0, 0)$, $(25, 100)$, $(70, 100)$, $(200, 0)$, $(200, 200)$ (en exactamente este orden) y pueden usarse para calcular la homografía, calibrar la cámara y encontrar la pose del marcador como se ve en la figura 3.17. O dicho de otra forma, a partir de estos puntos podemos encontrar la matriz de rotación y el vector de traslación que aplicados al marcador producen la vista del marcador en la figura 3.17.

El procedimiento para obtener la homografía y la pose se explicó en la sección 2.3.3, en la página 12.

3.5. Procesamiento de imagen en la Raspberry

La manera en la que se hace la segmentación de los objetos en el procesamiento de imagen en la raspberry es diferente a la que se realiza en la etapa de entrenamiento. La manera en la que se realiza el clasificador en la etapa de entrenamiento es separar en diferentes imágenes cada uno de los objetos presentes en la imagen. Al separar los objetos en imágenes diferentes su identificación es más fácil para su clasificación de manera visual y la extracción de características para la etapa de entrenamiento.

El proceso de extracción de imágenes antes mencionado ocupa muchos recursos y tiempo de procesamiento lo cual en dispositivos de bajos recursos como lo es la computadora Raspberry Pi 3 es un gran inconveniente, por lo que se plantea el procesamiento y clasificación de la imagen con todos los objetos en una sola imagen.

Al poner una etiqueta numérica diferente a cada objeto dentro de la imagen se pueden identificar los distintos objetos que conforman la imagen. Al estar trabajando en una escala de grises, se tiene tonos en el rango de 0 a 255 para usar como identificador. La imagen se procesa empezando por la esquina superior izquierda y se recorre la imagen hacia la derecha y hacia abajo, el etiquetado se hace conforme se encuentran nuevos objetos empezando por el tono 1 hasta encontrar el marcador. Al tener una imagen binarizada los objetos presentes están en color negro identificados con el tono 0 y el fondo es blanco identificado con el tono 255.

En la figura 3.18 se puede observar un ejemplo de la forma en que se realiza esta identificación. Todo aquello que no tiene un número identificador en la figura corresponde al fondo blanco por lo cual tiene asignado el número 255. En este ejemplo se observa que debido a que el etiquetado en escala de grises, la identificación de las etiquetas de manera visual no es fácil. Se etiqueta cada objeto, incluyendo puntos que son imperceptibles a la vista, en ese ejemplo en específico el marcador está identificado con el 65. Las objetos siguientes ya no fueron etiquetados debido a que el marcador ya fue encontrado (el resto de los objetos no etiquetados tienen pixeles iguales a cero, que significa que no fueron etiquetados).

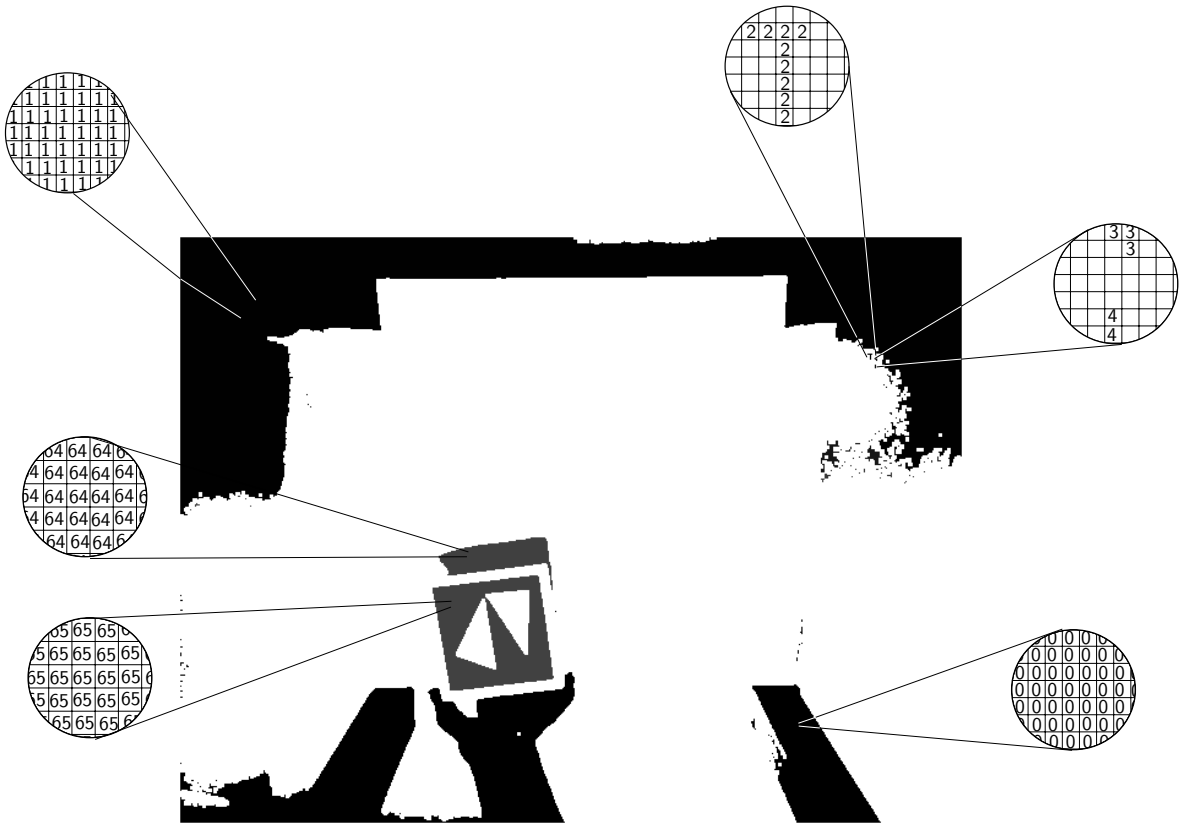


Figura 3.18: Segmentación de objetos en una imagen.

En la imagen existen áreas extensas en donde no se encuentra ningún objeto, por lo cual el realizar procesamientos en esas áreas es pérdida de tiempo en cálculos innecesarios. De esta forma al hacer la segmentación un objeto, se encuentran los puntos de la caja que encierran al objeto, y el clasificación del objeto se realiza únicamente dentro de esa caja. En caso de que dos objetos coexistan en el área de la caja, el algoritmo es capaz de diferenciar los objetos debido al etiquetado por tono del objeto.

Una vez encontrado el marcador por medio del clasificador se procede a la búsqueda de los puntos internos del marcador, los cuales conforman los dos triángulos internos, como se explicó en la subsección 3.4.1 estos puntos son usados para calibrar la cámara y encontrar la pose del marcador.

Finalmente con la pose del marcador, encontrada por medio de los puntos del modelo del marcador, se moverá un objeto tridimensional mostrado en la computadora. El objeto tridimensional en la computadora se moverá de acuerdo a la rotación y traslación calculadas por la pose del marcador.

3.6. Calibración en la imagen

Se usan patrones para determinar los parámetros intrínsecos y extrínsecos de la cámara. Estos patrones por lo general suelen estar sobre un entorno plano como lo son los tableros de ajedrez. Existen muchos trabajos [26, 27, 28] que utilizan este tipo de patrones planos. Para obtener los parámetros se necesita conocer el patrón y los puntos que conforman el modelo del patrón de distintas imágenes.

El patrón que se usa para la obtención del parámetro del foco de la cámara se observa en la figura 3.19. Este consiste en un patrón plano similar al tablero de ajedrez. Este patrón es de tamaño 11 cm ×11 cm, el cual contiene 25 cuadros de tamaño 1 cm × 1 cm, cada cuadro es separado por 1 cm en todos sus lados.

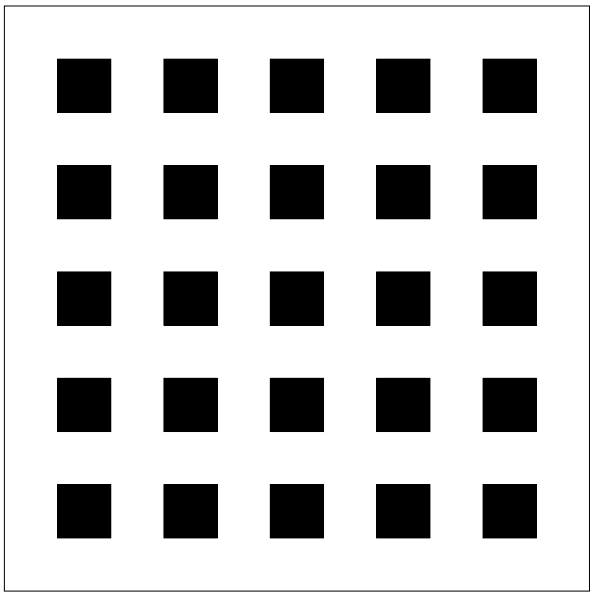


Figura 3.19: Marcador plano tipo tablero de ajedrez.

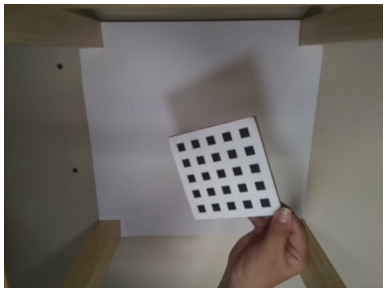
En la figura 3.20 se muestran capturas realizadas con la cámara de la Raspberry con una resolución de 640×480. Las capturas fueron tomadas de diferentes alturas con la finalidad obtener el foco de la cámara y ver si este cambiaba conforme a la altura.

Se creo el modelo $\mathbf{P}$ con base en el marcador y por cada imagen se obtuvo $\mathbf{p}$ de con los puntos de las esquinas de todos los cuadrados de la imagen siguiendo el mismo patrón indicado por el modelo $\mathbf{P}$.

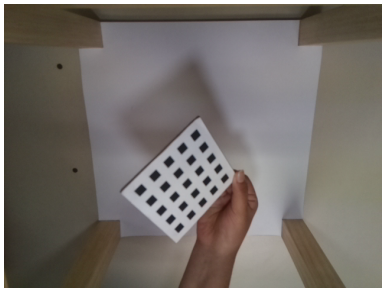
Los resultados obtenidos fueron los siguientes:

a) Marcador con la menor distancia de la cámara.

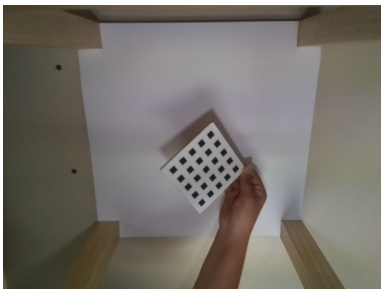
- Distancia aproximada de 20 cm de la cámara al marcador.



a) Marcador con la menor distancia de la cámara.



b) Marcador con una distancia media de la cámara.



c) Marcador con una distancia más lejana de la cámara.

Figura 3.20: Capturas realizadas con el modulo de cámara de la raspberry

- Foco: 529.14616895.
- Error: 0.890745338504.

b) Marcador con una distancia media de la cámara.

- Distancia aproximada de 30 cm de la cámara al marcador.
- Foco: 465.804880325.
- Error: 0.861189358064.

c) Marcador con una distancia más lejana de la cámara.

- Distancia aproximada es un poco menor de 50 cm de la cámara al marcador.
- Foco: 578.615370973.
- Error: 0.626918346033.

Al observar los resultados podemos notar que sí existe variación entre los focos obtenidos. La diferencia que existe entre el mayor foco obtenido (c) y el menor (b) es de 112.810490648, el cual es un cambio considerable en el foco.

Debido al cambio que se obtuvo de los focos, una posible solución para fijar el foco es la diferencia entre el mayor foco obtenido y el menor, el cual nos da un promedio de 522.210125649. Otra opción posible es obtener la media de estos valores, con los datos obtenidos la media es 524.522140083.

Ambas opciones nos dan un foco parecido al obtenido por (a) de 529.14616895 cuando el patrón se encuentra a menor distancia. Se opto por elegir la media de los focos y se hizo su redondeo a 525.

Capítulo 4

Simulación y Resultados

En esta sección se muestra los resultados tanto de una simulación del sistema como del sistema ya funcionando.

Para la simulación se tomaron los valores para la cámara tal como se midieron en el sistema físico. También fue muy importante poner los valores de los vértices de los triángulos del marcador en unidades de milímetros, así todas las medidas quedan expresadas en milímetros. La simulación además arroja que es necesario obtener la posición de los vértices de los triángulos a nivel subpixel para que se pueda obtener la pose del marcador de forma funcional (con poco ruido).

4.1. Simulación

Las posiciones de los vértices del modelo del marcador usado en milímetros son: (-33.5, 33.5), (-33.5, -33.5), (-25.125, 0.0), (-10.05, 0.0), (33.5, -33.5), (33.5, 33.5). Son los mismos vértices usados en las figuras 3.15 y 3.16. Estos valores están referidos al centro del marcador.

La cámara se situó en (0.0, 200.0, 540.0) mm, viendo hacia el centro del marcador, que es el punto (0,0,0). La orientación hacia arriba de la cámara es el eje de las y, esto es el vector (0,1,0).

El foco de la cámara se tomó al promedio de las calibraciones realizadas en la sección 3.6, $f = (529 + 466 + 579)/3 = 525$. El punto principal es el punto medio de las imágenes (320, 240).

Se simuló el marcador moviéndose en un círculo de radio 100 mm y a varias distancias en z. En la figura 4.1 se muestra el movimiento del marcador cada 5°, lo que hace un total de 72 pares de triángulos que son los que se muestran. En la figura se puede apreciar también la base de la estructura, tal como la cámara la ve. El código de está simulación se encuentra en el apéndice B, en la página 61.

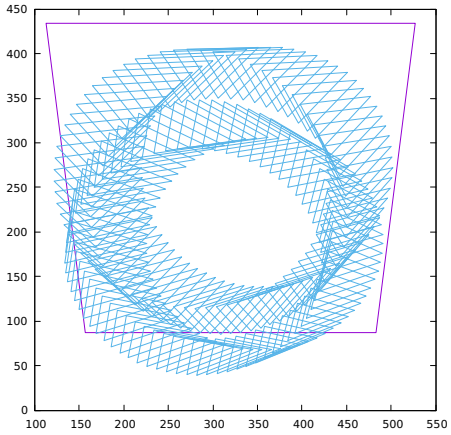


Figura 4.1: Simulación del movimiento del marcador rotado a 30° y traslado en forma circular cada 5° con un radio de 100 mm.

Lo que se quiere es que el movimiento del marcador dentro del aparato, entonces esta transformación que se quiere queda como:

$$\mathbf{P}_2 = T_1(\mathbf{t})R_1\mathbf{P}_1 \tag{4.1}$$

Con los datos arriba expuestos el modelo de la cámara oscura queda como:

$$\lambda\mathbf{p}_3 = KR_c[I|-c]\mathbf{P}_2, \tag{4.2}$$

donde R_c es la rotación inducida por la posición de la cámara dentro del aparato, $-c$ es la posición de la cámara. Considerando sólo la transformación geométrica que sufren los puntos $\mathbf{P}_2$ queda como:

$$\mathbf{P}_3 = R_cT_2(-c)\mathbf{P}_2, \tag{4.3}$$

esto es, los puntos del marcador se trasladan una distancia igual a $-c$ y luego se rotan por R_c.

Lo que se obtiene a partir de la homografía es la transformación que incluye las transformaciones en (4.3) y en (4.1), esto es:

$$\mathbf{P}_3 = R_cT_2(-c)T_1(\mathbf{t})R_1\mathbf{P}_1 = M\mathbf{P}_1. \tag{4.4}$$

Entonces para obtener $\mathbf{t}$ y R_1 a partir de M, se hace:

$$\begin{aligned} R_cT_2(-c)T_1(\mathbf{t})R_1 &= M, \\ T_2(-c)T_1(\mathbf{t})R_1 &= R_c^TM, \\ T_1(\mathbf{t})R_1 &= T_2(c)R_c^TM. \end{aligned} \tag{4.5}$$

Entonces a la matriz obtenida por la homografía hay que quitarle la rotación inducida por la posición de la cámara y la traslación de la cámara y se obtiene la rotación

y traslación que sufre el marcador. Y esta transformación es la que se quiere utilizar para mover o controlar otro cuerpo tridimensional.

En las gráficas 4.2, 4.3 y 4.4 se muestran los errores obtenidos en $\mathbf{t}$ de las expresiones (4.1), (4.5) y en los ángulos 30°, 45° y 60°, respectivamente, en los que se rotó el marcador en el eje y. Esta simulación se realizó redondeando las posiciones de los vértices de los triángulos del marcador al pixel más cercano. Y como se nota en las gráficas el error es muy grande para que sea útil la medición.

Si la posición de los pixeles se trunca a un sólo decimal, se obtienen los errores mostrados en las gráficas de las figuras 4.5, 4.6 y 4.7.

Si se trunca las posiciones de los pixeles a dos decimales, se obtienen los resultados mostrados en las gráficas de las figuras 4.8, 4.9 y 4.10. Con esta precisión los errores son despreciables y las mediciones obtenidas de $\mathbf{t}$ y R_1 son útiles. Como las posiciones de los vértices se obtienen a partir de la intersección de las aristas que forman cada triángulo, es plausible tener esta alta precisión.

Los valores de las gráficas 4.2-4.10 se incluyen en el apéndice A.

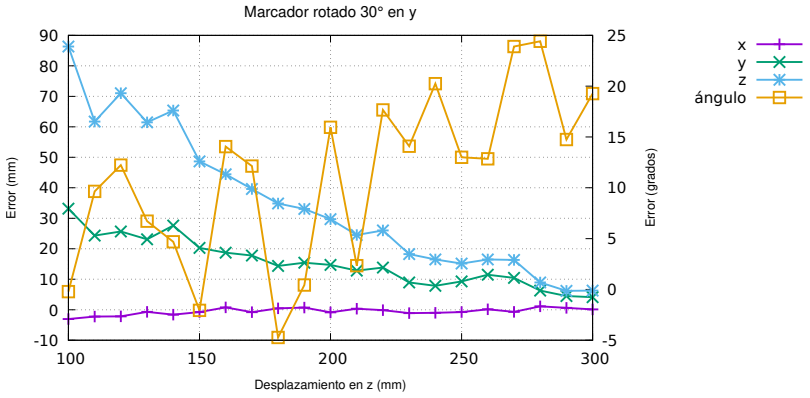


Figura 4.2: Simulación 1 ajustando los valores de los vértices de los triángulos al entero más cercano y el marcador rotado 30° en el eje y

4.2. Resultados

En la figura 4.11 se muestra el aparato funcionando. El marcador se posicionó encima de una lata de spray para evitar meter el brazo. En la figura 4.12 se muestra la pantalla del algunos datos del sistema en funcionamiento. Finalmente en la figura 4.13 se muestra la imagen que se obtiene de la Raspberry y un objeto virtual.

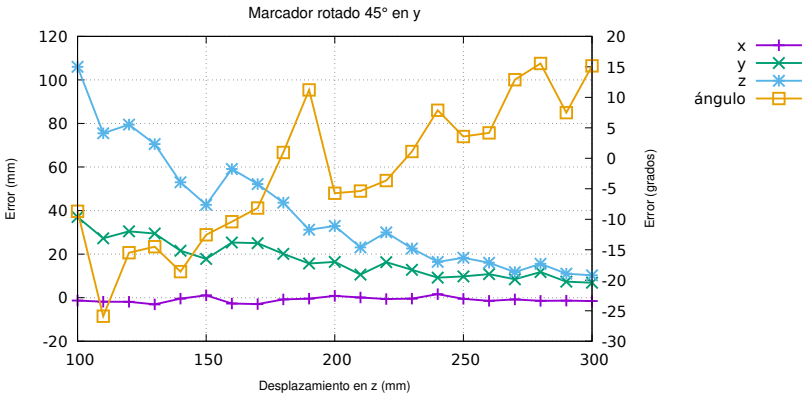


Figura 4.3: Simulación 1 ajustando los valores de los vértices de los triángulos al entero más cercano y el marcador rotado 45° en el eje y

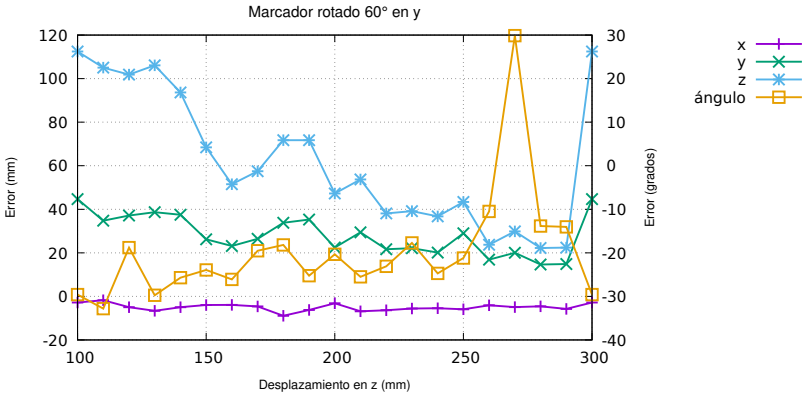


Figura 4.4: Simulación 1 ajustando los valores de los vértices de los triángulos al entero más cercano y el marcador rotado 60° en el eje y

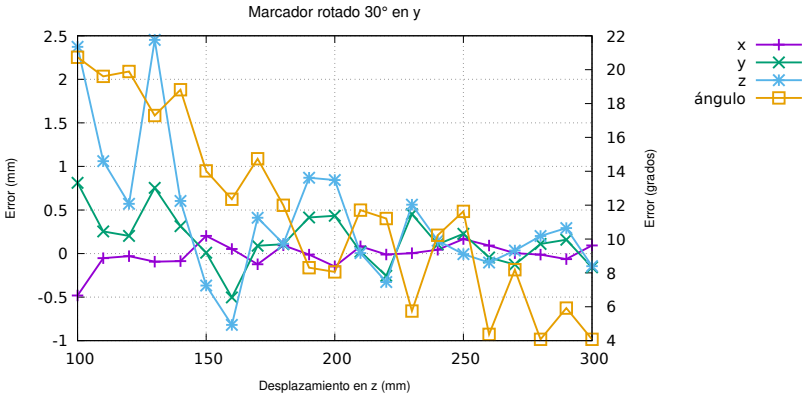


Figura 4.5: Simulación 2 ajustando los valores de los vértices de los triángulos con una cifra decimal y el marcador rotado 30° en el eje y

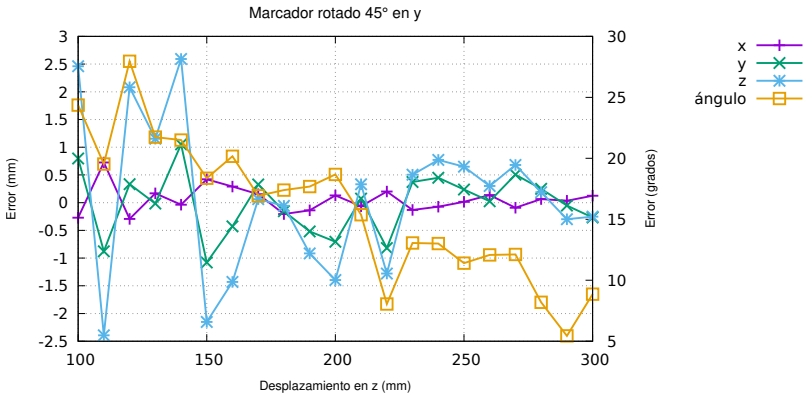


Figura 4.6: Simulación 2 ajustando los valores de los vértices de los triángulos con una cifra decimal y el marcador rotado 45° en el eje y

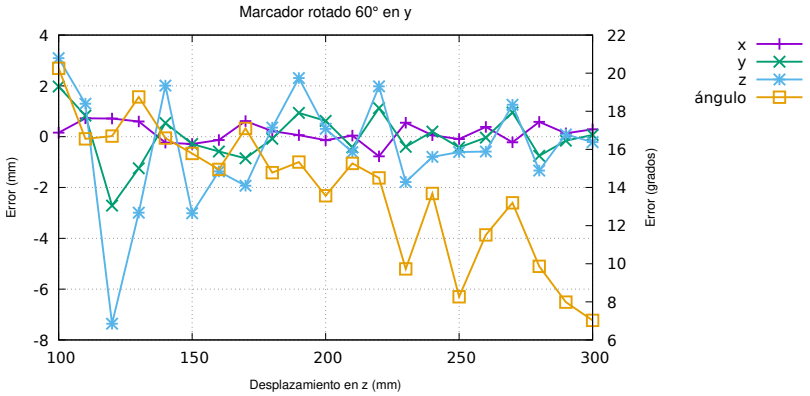


Figura 4.7: Simulación 2 ajustando los valores de los vértices de los triángulos con una cifra decimal y el marcador rotado 60° en el eje y

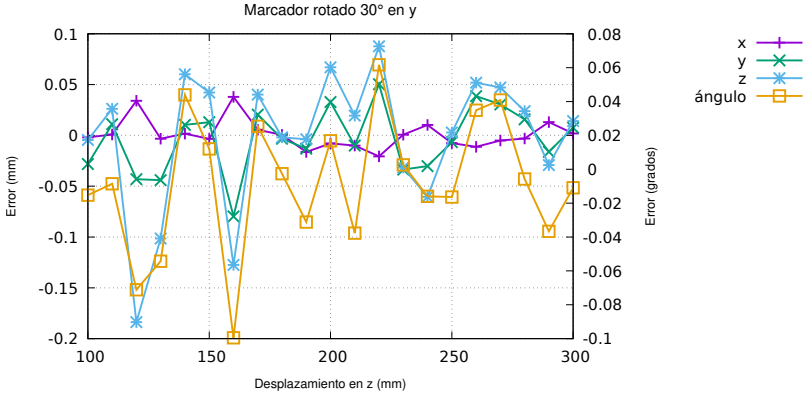


Figura 4.8: Simulación 3 ajustando los valores de los vértices de los triángulos con dos cifras decimales y el marcador rotado 30° en el eje y

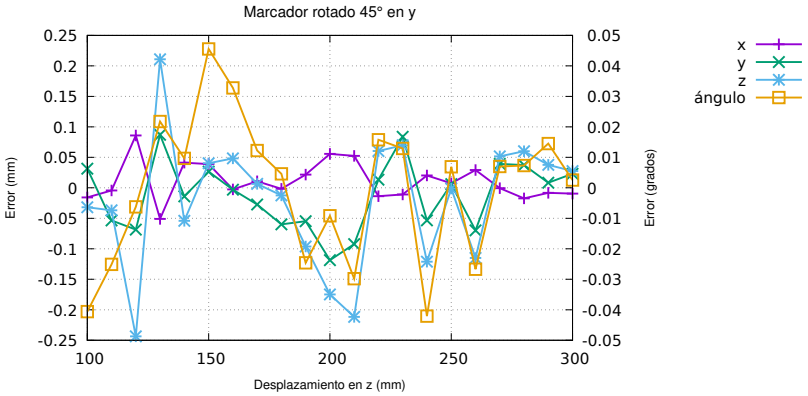


Figura 4.9: Simulación 3 ajustando los valores de los vértices de los triángulos con dos cifras decimales y el marcador rotado 45° en el eje y

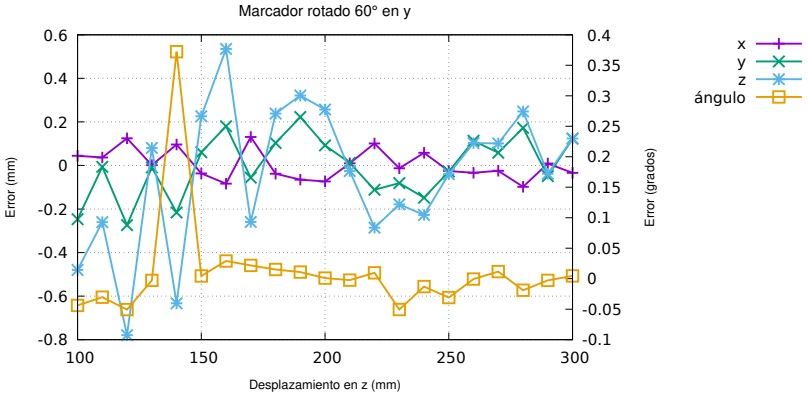


Figura 4.10: Simulación 3 ajustando los valores de los vértices de los triángulos con dos cifras decimales y el marcador rotado 60° en el eje y

Figura 4.11: El aparato en funcionamiento.

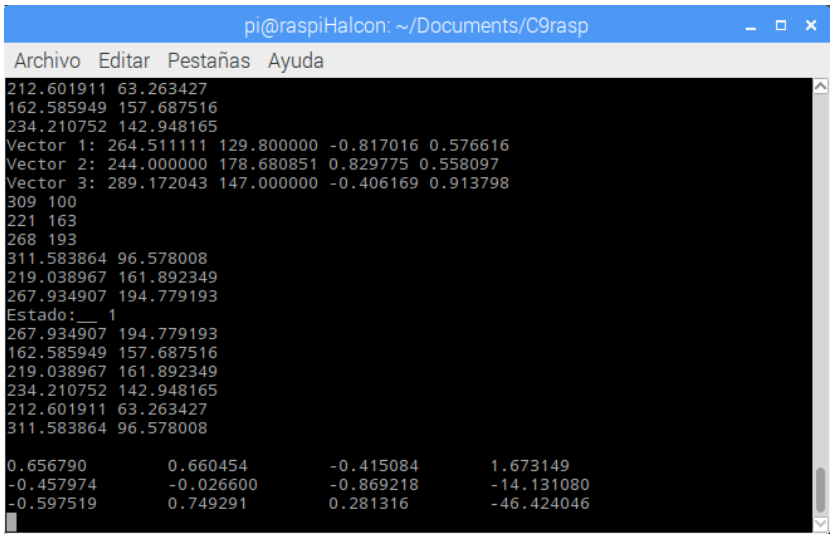


Figura 4.12: Salida de pantalla de algunos datos del programa en funcionamiento.

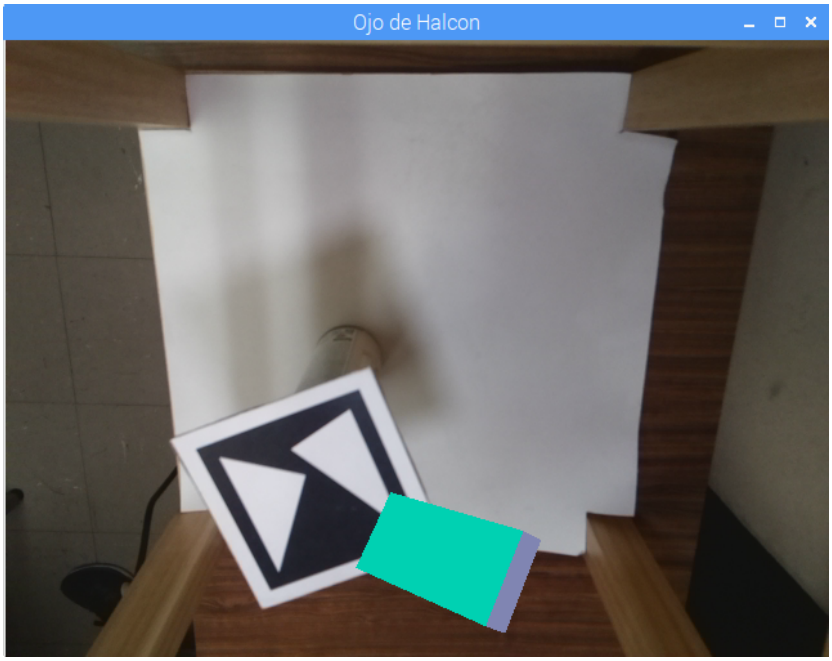


Figura 4.13: Pantalla del programa en OpenGL que muestra la imagen obtenida en la Raspberry y un objeto virtual sobre de ella.

Capítulo 5

Conclusiones

Se realizó un dispositivo para capturar la pose de un marcador y usar esa pose como entrada para desplazar o controlar algún otro dispositivo tridimensional. Por supuesto, se realizó todo el software que captura la pose en una computadora en una sola tarjeta modelo Raspberry Pi B.

El aparato tiene unas dimensiones de 40 cm $\times$ 40 cm $\times$ 60 cm (en x,y,z, respectivamente). Se uso un foco de 3 vatios con difusor para tener un ambiente controlado y la que la iluminación externa no afecté el funcionamiento.

Se usó la computadora en una sola tarjeta Raspberry Pi B por las prestaciones que tiene su cámara. Está cámara es un aditamento y los módulos para su funcionamiento ya vienen incluidos en la distribución de GNU/Linux que es de licencia libre. La alternativa era usar una cámara genérica a través del puerto USB, pero esta solución complicaba demasiado el desarrollo y el tamaño del software, que es más grande para funcionar con cámaras genéricas.

Se usó un marcador basado en tipo de orden que simplifica el emparejamiento de los puntos de la imagen del marcador que se obtiene con los del modelo del marcador, como se explicó en la sección 3.4.1, en la página 32 de esta tesis.

El marcador viene indicado por puntos, en la imagen se usaron vértices de triángulos para indicar la posición de esos puntos. Las posiciones de los puntos se detectaron por la intersección de las aristas de los triángulos.

En las simulaciones que se hizo del sistema se encontró que la posición de los vértices debe encontrarse con precisión subpixel para que la pose del marcador sea funcional, esto es, sin ruido.

En el software desarrollado se usó la aproximación más sencilla a través de la homografía para conocer la pose del marcador. Para esto también se tuvo que calibrar la cámara.

Aunque se tiene ya un sistema completo para capturar la pose del marcador en las pruebas hechas se detectó que hacen falta muchas mejoras que se deja como trabajo a futuro y que son descritas en la siguiente sección.

5.1. Trabajo a futuro

De las mejoras que pueden hacerse se visualizaron las siguientes:

1. Incorporar una umbralización adaptiva en el procesamiento de imagen. Aunque se estudió esta posibilidad no se incorporó en el software actual.

2. Usar un mejor algoritmo para la obtención de la pose del marcador. Actualmente el algoritmo IPP (infinitesimal plane based pose estimation [29]), que ya se probó y da mejores aproximaciones que el basado en la homografía. Este software esta disponible públicamente pero depende de OpenCV, y se deja como trabajo a futuro incorporar este software a este desarrollo.

3. Considerar si un sistema de ojo de halcón basado en triangulación es más preciso que el propuesto en esta tesis. Un sistema basado en triangulación tiene las desventajas de que requeriría el uso de al menos tres computadoras Raspberry, para que tome tres imágenes independientes y deben de estar sincronizadas de alguna forma en red.

4. Agregar al software que puedan usarse con muchos marcadores de distintos tipos, todos basados en tipo de orden y que funcionen también cuando ocurra oclusión de alguna parte de los marcadores.

5. Aún queda como trabajo de investigación cual es el efecto de dejar el foco de la cámara fijo. Con la información que se tiene de la cámara de la Raspberry esta realiza un autofoco de manera automática y se debe saber cual es el efecto de esta característica en el sistema propuesto.

Apéndice A

Valores Obtenidos en la Simulación

	$\theta = 30°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	ángulo	$\sigma_{\text{ángulo}}$
100	-3.05	16.43	33.16	31.63	86.32	66.02	-0.22	65.05
110	-2.24	15.44	24.33	30.18	61.74	63.89	9.65	61.62
120	-2.16	15.55	25.66	28.30	71.03	58.71	12.24	59.98
130	-0.71	14.68	23.09	25.98	61.46	54.17	6.72	61.63
140	-1.61	15.07	27.61	26.99	65.35	50.55	4.68	58.51
150	-0.77	12.63	20.28	26.19	48.66	49.97	-2.06	61.45
160	0.78	11.73	18.73	24.85	44.44	47.04	14.05	52.12
170	-0.79	10.83	17.77	25.50	39.62	46.94	12.14	59.43
180	0.48	10.72	14.37	19.69	34.84	41.07	-4.73	58.53
190	0.73	11.07	15.44	21.82	33.06	39.23	0.43	60.02
200	-0.85	9.84	14.70	20.75	29.72	35.16	15.95	50.20
210	0.34	9.17	14.70	20.75	24.52	31.19	2.34	53.19
220	-0.13	8.48	13.82	21.85	26.02	34.01	17.66	50.16
230	-1.12	9.51	8.96	15.01	18.26	28.56	14.09	48.60
240	-1.02	8.26	7.86	17.14	16.51	27.26	20.25	47.23
250	-0.75	8.43	9.28	18.19	15.15	28.09	13.00	45.20
260	0.16	6.96	11.48	19.98	16.51	25.75	12.85	49.12
270	-0.69	7.98	10.45	15.53	16.35	22.36	23.90	39.75
280	1.11	5.65	6.30	16.94	8.93	20.96	24.41	36.19
290	0.63	6.15	4.51	17.16	6.19	20.38	14.75	42.39
300	0.11	6.10	4.14	14.35	6.28	17.42	19.27	39.26

Tabla 1. Datos obtenidos en la simulación con el marcador rotado en el eje y 30° truncando las posiciones de los puntos al entero más cercano. La simulación se realizó como se describió en el capítulo 4. Note la gran variabilidad de los datos, lo que indica que no es suficiente esta precisión.

	$\theta = 45°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	$\bar{\text{ángulo}}$	$\sigma_{\text{ángulo}}$
100	-1.32	20.91	37.04	36.13	106.02	83.44	-8.69	60.14
110	-1.84	19.98	27.27	36.32	75.49	83.75	-25.89	70.70
120	-1.90	19.07	30.48	35.97	79.48	75.81	-15.48	64.40
130	-3.08	19.23	29.47	40.30	70.54	82.68	-14.50	64.76
140	-0.45	16.97	21.56	35.27	53.02	76.54	-18.58	59.38
150	1.09	15.09	17.79	35.62	42.68	76.01	-12.53	56.14
160	-2.67	16.70	25.33	34.76	59.09	66.73	-10.40	60.51
170	-2.96	15.13	25.05	32.61	52.17	60.95	-8.16	59.15
180	-0.80	15.34	20.10	28.60	43.63	56.84	0.96	52.03
190	-0.44	13.48	15.69	31.65	31.20	58.10	11.21	43.91
200	0.85	11.81	16.44	24.31	32.91	44.86	-5.73	47.94
210	0.08	10.34	10.58	20.96	23.12	40.52	-5.37	51.62
220	-0.63	13.02	16.31	26.50	29.97	46.87	-3.64	52.26
230	-0.44	12.24	12.79	27.93	22.59	45.61	1.11	48.66
240	1.62	9.50	9.19	22.24	16.44	36.37	7.88	45.41
250	-0.52	9.77	9.78	20.34	18.31	33.91	3.57	46.93
260	-1.47	10.21	10.82	23.00	16.05	35.38	4.16	38.65
270	-0.80	9.05	8.45	21.33	11.76	31.80	12.90	43.63
280	-1.46	10.91	11.79	23.35	15.55	36.48	15.55	36.48
290	-1.35	9.47	7.34	18.35	11.06	27.30	7.49	40.34
300	-1.55	9.16	6.89	18.73	10.24	25.50	15.17	30.90

Tabla 2. Datos obtenidos en la simulación con el marcador rotado en el eje y 45° truncando las posiciones de los puntos al entero más cercano. La simulación se realizó como se describió en el capítulo 4. Note la gran variabilidad de los datos, lo que indica que no es suficiente esta precisión.

	$\theta = 60°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	$\bar{\text{ángulo}}$	$\sigma_{\text{ángulo}}$
100	-2.80	22.27	44.68	47.58	112.47	103.33	-29.59	70.40
110	-1.78	24.89	34.78	41.45	105.07	96.46	-32.82	70.48
120	-4.97	24.74	37.14	46.81	101.83	101.04	-18.80	66.75
130	-6.64	24.02	38.69	41.15	106.09	91.36	-29.74	68.29
140	-4.94	21.70	37.50	41.44	93.63	85.19	-25.71	68.31
150	-3.94	23.08	26.22	43.76	68.45	96.92	-23.92	62.45
160	-3.89	19.53	23.17	44.97	51.51	90.29	-26.10	61.97
170	-4.60	21.65	26.42	40.68	57.48	90.40	-19.52	60.82
180	-8.92	24.00	33.82	44.85	71.74	91.89	-18.20	62.21
190	-6.23	20.82	35.28	37.63	71.73	74.76	-25.25	67.03
200	-3.18	17.86	22.42	38.76	47.28	70.18	-20.34	63.54
210	-6.84	19.33	29.41	37.80	53.73	71.30	-25.52	64.77
220	-6.35	18.72	21.67	37.94	38.08	71.27	-23.11	62.03
230	-5.59	17.30	22.16	36.24	39.13	60.62	-17.74	62.52
240	-5.45	16.74	20.12	34.35	36.74	58.68	-24.71	62.85
250	-5.91	17.55	28.98	36.87	43.35	58.16	-21.18	61.37
260	-4.09	13.40	16.86	31.86	23.73	50.34	-10.50	51.75
270	-4.93	16.27	20.05	36.24	29.87	51.83	29.87	51.83
280	-4.56	15.67	14.63	35.38	22.23	51.37	-13.84	55.59
290	-5.76	16.17	14.87	32.72	22.39	47.60	-14.05	55.96
300	-2.80	22.27	44.68	47.58	112.47	103.33	-29.59	70.40

Tabla 3. Datos obtenidos en la simulación con el marcador rotado en el eje y 60° truncando las posiciones de los puntos al entero más cercano. La simulación se realizó como se describió en el capítulo 4. Note la gran variabilidad de los datos, lo que indica que no es suficiente esta precisión.

	$\theta = 30°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	$\bar{\text{ángulo}}$	$\sigma_{\text{ángulo}}$
100	-0.48	1.68	0.81	4.12	2.37	10.15	20.73	28.80
110	-0.05	1.61	0.25	3.85	1.06	9.22	19.60	28.66
120	-0.03	1.22	0.20	3.24	0.57	7.54	19.89	28.45
130	-0.09	1.40	0.75	3.26	2.45	7.57	17.29	26.71
140	-0.09	1.34	0.31	2.85	0.60	6.97	18.81	28.37
150	0.20	1.54	0.01	3.11	-0.37	7.12	14.02	25.62
160	0.05	1.29	-0.50	2.92	-0.82	6.17	12.35	25.62
170	-0.12	1.33	0.09	2.89	0.41	6.20	14.74	26.25
180	0.09	1.02	0.11	2.27	0.11	4.93	12.00	24.71
190	-0.01	0.83	0.41	1.90	0.87	3.75	8.31	19.67
200	-0.15	0.88	0.43	1.98	0.84	3.84	8.06	20.97
210	0.08	1.02	0.02	2.12	0.01	4.14	11.70	23.80
220	-0.01	1.14	-0.26	2.46	-0.33	4.38	11.21	24.03
230	0.00	0.79	0.45	1.81	0.56	3.17	5.75	17.96
240	0.04	1.00	0.11	2.06	0.15	3.55	10.22	22.34
250	0.17	0.96	0.23	2.05	-0.01	3.41	11.64	23.79
260	0.09	0.75	-0.05	1.79	-0.10	2.67	4.37	15.33
270	0.00	0.87	-0.13	1.92	0.03	2.89	8.19	20.86
280	-0.01	0.61	0.11	1.34	0.20	1.95	4.08	15.34
290	-0.06	0.70	0.16	1.62	0.29	2.14	5.92	17.84
300	0.09	0.67	-0.16	1.44	-0.15	1.98	4.08	15.32

Tabla 4. Datos obtenidos en la simulación con el marcador rotado en el eje y 30° truncando las posiciones de los puntos a un valor con un decimal. La simulación se realizó como se describió en el capítulo 4.

	$\theta = 45°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	$\bar{\text{ángulo}}$	$\sigma_{\text{ángulo}}$
100	-0.27	3.19	0.80	8.03	2.46	19.94	24.36	22.69
110	0.73	3.48	-0.88	6.99	-2.39	16.91	19.53	22.93
120	-0.29	2.95	0.33	6.78	2.08	16.30	27.95	22.05
130	0.17	2.49	-0.02	5.93	1.15	13.91	21.74	22.64
140	-0.04	2.87	1.05	6.92	2.59	14.97	21.49	22.30
150	0.42	2.47	-1.08	5.66	-2.15	12.45	18.36	22.55
160	0.29	2.61	-0.43	5.62	-1.43	12.41	20.17	22.33
170	0.15	1.83	0.33	4.15	0.06	9.05	16.93	21.79
180	-0.21	2.21	-0.15	4.64	-0.06	10.23	17.39	22.03
190	-0.14	1.85	-0.52	3.92	-0.92	8.44	17.68	22.49
200	0.13	2.45	-0.71	4.85	-1.40	9.93	18.68	22.30
210	-0.06	1.83	0.07	4.25	0.33	8.01	15.38	21.58
220	0.20	1.38	-0.82	3.86	-1.28	6.23	8.05	17.42
230	-0.13	1.66	0.37	4.58	0.50	7.15	13.06	20.57
240	-0.07	1.56	0.45	3.18	0.77	5.89	13.01	20.60
250	0.02	1.50	0.24	3.24	0.65	5.51	11.41	19.45
260	0.14	1.30	0.03	3.11	0.30	4.84	12.08	19.76
270	-0.09	1.59	0.51	3.43	0.68	5.11	12.12	19.70
280	0.07	1.56	0.25	3.48	0.19	4.93	8.19	17.33
290	0.03	1.04	-0.05	2.50	-0.30	3.37	5.43	14.96
300	0.13	1.14	-0.27	2.40	-0.25	3.40	8.86	17.79

Tabla 5. Datos obtenidos en la simulación con el marcador rotado en el eje y 45° truncando las posiciones de los puntos a un valor con un decimal. La simulación se realizó como se describió en el capítulo 4.

	$\theta = 60°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	$\bar{\text{ángulo}}$	$\sigma_{\text{ángulo}}$
100	0.16	5.32	1.97	11.91	3.09	32.51	20.26	13.77
110	0.72	6.01	0.83	12.37	1.30	33.65	16.55	14.49
120	0.72	5.60	-2.71	10.99	-7.36	29.33	16.70	15.02
130	0.60	5.39	-1.24	11.41	-2.99	27.75	18.75	14.67
140	-0.24	5.63	0.54	12.52	2.01	28.82	16.59	14.46
150	-0.29	5.19	-0.29	5.19	-3.01	26.43	15.78	14.77
160	-0.13	3.49	-0.58	8.74	-1.37	18.05	14.95	15.12
170	0.62	5.11	-0.85	10.59	-1.93	23.01	17.10	14.88
180	0.22	4.18	-0.08	9.61	0.36	19.51	14.78	14.86
190	0.06	4.98	0.94	10.41	2.31	21.37	15.34	14.73
200	-0.14	3.89	0.62	9.00	0.31	17.61	13.56	14.48
210	0.04	3.77	-0.42	8.05	-0.61	15.88	15.27	14.77
220	-0.77	4.06	1.13	8.67	1.98	16.04	14.51	14.73
230	0.55	4.26	-0.40	7.63	-1.78	15.43	9.73	14.02
240	0.06	3.41	0.20	7.59	-0.79	13.07	13.69	14.67
250	-0.09	3.85	-0.42	7.87	-0.60	13.14	8.27	13.15
260	0.38	3.11	-0.04	6.53	-0.58	10.57	11.51	14.37
270	-0.22	3.28	0.96	7.02	1.26	10.96	13.19	14.74
280	0.58	3.28	-0.75	5.91	-1.33	9.66	9.86	13.88
290	0.13	3.65	-0.16	7.54	0.09	10.66	7.98	12.85
300	0.29	3.25	0.07	5.83	-0.21	8.77	7.02	12.36

Tabla 6. Datos obtenidos en la simulación con el marcador rotado en el eje y 60° truncando las posiciones de los puntos a un valor con un decimal. La simulación se realizó como se describió en el capítulo 4.

	$\theta = 30°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	ángulo	$\sigma_{\text{ángulo}}$
100	0.00	0.20	-0.03	0.46	0.00	1.13	-0.02	0.57
110	0.00	0.19	0.01	0.40	0.03	0.99	-0.01	0.45
120	0.03	0.16	-0.04	0.35	-0.18	0.87	-0.07	0.42
130	0.00	0.19	-0.04	0.40	-0.10	0.94	-0.05	0.47
140	0.00	0.13	0.01	0.28	0.06	0.68	0.04	0.36
150	0.00	0.16	0.01	0.34	0.04	0.76	0.01	0.39
160	0.04	0.15	-0.08	0.28	-0.13	0.66	-0.10	0.39
170	0.01	0.14	0.02	0.34	0.04	0.67	0.03	0.39
180	0.00	0.11	0.00	0.25	0.00	0.53	0.00	0.31
190	-0.02	0.11	-0.01	0.24	0.00	0.49	-0.03	0.32
200	-0.01	0.13	0.03	0.29	0.07	0.53	0.02	0.33
210	-0.01	0.12	-0.01	0.12	0.02	0.47	-0.04	0.28
220	-0.02	0.09	0.05	0.22	0.09	0.38	0.06	0.27
230	0.00	0.08	-0.03	0.18	-0.03	0.30	0.00	0.21
240	0.01	0.08	-0.03	0.19	-0.06	0.31	-0.02	0.23
250	-0.01	0.12	-0.01	0.22	0.00	0.38	-0.02	0.24
260	-0.01	0.06	0.04	0.16	0.05	0.23	0.03	0.18
270	-0.01	0.06	0.03	0.14	0.05	0.20	0.04	0.19
280	0.00	0.07	0.02	0.15	0.02	0.22	-0.01	0.19
290	0.01	0.08	-0.02	0.15	-0.03	0.22	-0.04	0.17
300	0.00	0.06	0.01	0.12	0.01	0.16	-0.01	0.14

Tabla 7. Datos obtenidos en la simulación con el marcador rotado en el eje y 30° truncando las posiciones de los puntos a un valor con dos decimales. La simulación se realizó como se describió en el capítulo 4. Aquí los errores ya son pequeños para poder usar los valores obtenidos.

	$\theta = 45°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	$\bar{\text{ángulo}}$	$\sigma_{\text{ángulo}}$
100	-0.02	0.29	0.03	0.73	-0.03	1.77	-0.04	0.42
110	0.00	0.38	-0.05	0.79	-0.04	1.98	-0.03	0.43
120	0.09	0.27	-0.07	0.66	-0.24	1.61	-0.01	0.38
130	-0.05	0.28	0.09	0.60	0.21	1.45	0.02	0.37
140	0.04	0.27	-0.01	0.66	-0.05	1.47	0.01	0.35
150	0.04	0.22	0.03	0.52	0.04	1.20	0.05	0.28
160	0.00	0.23	0.00	0.58	0.05	1.18	0.03	0.29
170	0.01	0.22	-0.03	0.53	0.01	1.09	0.01	0.30
180	0.00	0.22	-0.06	0.49	-0.01	1.03	0.00	0.27
190	0.02	0.22	-0.05	0.43	-0.10	0.92	-0.02	0.26
200	0.06	0.27	-0.12	0.55	-0.17	1.09	-0.01	0.27
210	0.05	0.20	-0.09	0.42	-0.21	0.84	-0.03	0.24
220	-0.01	0.19	0.01	0.39	0.06	0.74	0.02	0.21
230	-0.01	0.16	0.08	0.35	0.07	0.62	0.01	0.19
240	0.02	0.15	-0.05	0.32	-0.12	0.56	-0.04	0.19
250	0.01	0.14	0.01	0.32	0.00	0.52	0.01	0.18
260	0.03	0.14	-0.07	0.28	-0.11	0.47	-0.03	0.16
270	0.00	0.12	0.04	0.28	0.05	0.44	0.01	0.17
280	-0.02	0.12	0.04	0.25	0.06	0.39	0.01	0.14
290	-0.01	0.15	0.01	0.37	0.04	0.48	0.01	0.16
300	-0.01	0.10	0.02	0.22	0.03	0.30	0.00	0.11

Tabla 8. Datos obtenidos en la simulación con el marcador rotado en el eje y 45° truncando las posiciones de los puntos a un valor con dos decimales. La simulación se realizó como se describió en el capítulo 4. Aquí los errores ya son pequeños para poder usar los valores obtenidos.

	$\theta = 60°$							
z	$\bar{x}$	σ_x	$\bar{y}$	σ_y	$\bar{z}$	σ_z	$\bar{\text{ángulo}}$	$\sigma_{\text{ángulo}}$
100	0.04	0.49	-0.25	1.10	-0.48	2.90	-0.04	0.28
110	0.04	0.49	-0.01	1.05	-0.26	2.79	-0.03	0.28
120	0.12	0.60	-0.27	1.24	-0.78	3.13	-0.05	0.28
130	0.00	0.60	-0.01	1.21	0.08	3.14	0.00	0.29
140	0.10	0.59	-0.22	1.28	-0.63	2.98	0.37	3.53
150	-0.04	0.55	0.06	1.24	0.23	2.81	0.00	0.29
160	-0.08	0.46	0.18	1.00	0.54	2.24	0.03	0.25
170	0.13	0.48	-0.06	1.10	-0.26	2.31	0.02	0.24
180	-0.04	0.46	0.10	1.09	0.24	2.20	0.02	0.27
190	-0.07	0.45	0.22	1.01	0.32	2.06	0.01	0.22
200	-0.07	0.40	0.09	0.99	0.26	1.79	0.00	0.23
210	0.01	0.46	0.01	0.97	-0.03	1.86	0.00	0.21
220	0.10	0.39	-0.11	0.79	-0.29	1.50	0.01	0.17
230	-0.01	0.45	-0.08	0.96	-0.18	1.69	-0.05	0.19
240	0.06	0.36	-0.15	0.74	-0.23	1.32	-0.01	0.15
250	-0.03	0.39	-0.03	0.84	-0.04	1.36	-0.03	0.14
260	-0.03	0.37	0.11	0.72	0.10	1.19	0.00	0.13
270	-0.02	0.36	0.06	0.77	0.10	1.17	0.01	0.12
280	-0.10	0.30	0.17	0.67	0.25	0.98	-0.02	0.11
290	0.01	0.36	-0.05	0.73	-0.04	1.10	0.00	0.12
300	-0.03	0.30	0.12	0.62	0.12	0.86	0.00	0.12

Tabla 9. Datos obtenidos en la simulación con el marcador rotado en el eje y 60° truncando las posiciones de los puntos a un valor con dos decimales. La simulación se realizó como se describió en el capítulo 4. Aquí los errores ya son pequeños para poder usar los valores obtenidos.

Apéndice B

Código de la simulación

La simulación se realizó con dos scripts de python y un programa en C. El programa en C calcula la homografía y se realiza con la biblioteca ligera en [30].

Los scripts de python, uno es el módulo `marker.py` y el programa principal está en el script `simulacion.py`, se encuentran a continuación.

Listing B.1: Script marker.py

```
import sys
import numpy
import math

class Marker :
  # La matriz de la cámara
  K = numpy.matrix( [ [525.0,    0.0, -320.0],
                      [    0.0, 525.0, -240.0],
                      [    0.0,    0.0,   -1.0] ] )

  def __init__( self ) :
    self.v = numpy.zeros( (6,3) )
    self.img = numpy.zeros( (6,2) )


  def create( self, vpts ) :
    n = len( vpts )
    i = 0
    while i < 6 :
      self.v[i][0] = vpts[i][0]
      self.v[i][1] = vpts[i][1]
      self.v[i][2] = 0.0
      i += 1
```

```
  def draw2D( self ) :
    i = 0
    while i < 6 :
      print self.img[i][0], self.img[i][1]
      i += 1

  #
  def pts2DtoFile( self, nombre ) :
    try:
      arch = open( nombre, "w" )
    except:
      print "No puedo abrir el archivo", nombre
      sys.exit(1)

    i = 0
    while i < 6 :
      arch.write( '%f %f\n' %( self.img[i][0], self.img[i][1] ) )
      # Con una cifra decimal significativa:
      #arch.write( '%3.1lf %3.1lf\n' %( self.img[i][0], self.img[i][1] ) )
      # Con dos cifras decimales significativas:
      #arch.write( '%3.2lf %3.2lf\n' %( self.img[i][0], self.img[i][1] ) )
      i += 1

    arch.close( )

  def drawTriangles( self ) :
    print self.img[0][0], self.img[0][1]
    print self.img[3][0], self.img[3][1]
    print self.img[5][0], self.img[5][1]
    print self.img[0][0], self.img[0][1]
    print
    print self.img[1][0], self.img[1][1]
    print self.img[2][0], self.img[2][1]
    print self.img[4][0], self.img[4][1]
    print self.img[1][0], self.img[1][1]

  def calculatesR( self, vo, vc, va ) :
    self.vc = vc - vo
    k = numpy.linalg.norm( self.vc )
    vz = self.vc / k

    v3 = numpy.cross( va, vz )
    vx = v3/numpy.linalg.norm(v3)
```

```
    vy = numpy.cross( vz, vx )

    self.R = numpy.matrix ( [ vx, vy, vz ] )
    # print self.R
    alfa1 = math.atan2( self.R[2,1], -self.R[2,0] )
    self.theta1 = math.degrees( alfa1 )

    beta1 = math.acos( self.R[2,2] )
    self.theta2 = math.degrees( beta1 )

    gama1 = math.atan2( self.R[1,2], self.R[0,2] )
    self.theta3 = math.degrees( gama1 )


  def proyect( self, theta, x, y, z ) :
    # Rota el marcador theta grados en y
    # y lo translada a x, y, z

    vp = numpy.matrix( [ [0.0], [0.0], [0.0]] )
    vp2 = numpy.matrix( [ [0.0], [0.0], [0.0]] )
    c = math.cos( theta )
    s = math.sin( theta )
    Rp = numpy.matrix( [ [c, 0.0, s], [0.0, 1.0, 0.0], [-s,0.0, c]] )

    i = 0
    while i<6 :
      vp[0] = self.v[i][0]
      vp[1] = self.v[i][1]
      vp[2] = self.v[i][2]
      vp2 = Rp * vp
      #
      # Aquí se calcula vp = T(-c) vr
      #
      vp[0] = vp2[0] - self.vc[0] + x
      vp[1] = vp2[1] - self.vc[1] + y
      vp[2] = vp2[2] - self.vc[2] + z

      v3 = self.R * vp
      v2 = Marker.K * v3

      self.img[i][0] = v2[0]/v2[2]
      self.img[i][1] = v2[1]/v2[2]
```

```
        i += 1
```

Listing B.2: Script simulacion.py

```
# encoding: utf-8
import numpy
import sys
import marker
import math
import subprocess

vo = numpy.array( [0.0, 0.0, 0.0] )
vc = numpy.array( [0.0, 200.0, 540.0] )
va = numpy.array( [0.0, 1.0, 0.0] )

# Los puntos del marcador en mm
lver = [ [-33.5, 33.5], [-33.5,-33.5],[-25.125,0.0], \
         [-10.05,0.0],[33.5,-33.5],[33.5, 33.5] ]

fiducial = marker.Marker( )

fiducial.create( lver )

# Se pone la cámara
fiducial.calculatesR( vo, vc, va )

i = 0
angtheta = 30.0
theta = angtheta * math.pi/180.0
angulo = 5.0*math.pi/180.0
errorx  = errory  = errorz  = errora  = 0.0
errorx2 = errory2 = errorz2 = errora2 = 0.0
while i < 72 :
  ang = angulo*i
  x = 100.0 * math.cos( ang )
  y = 100.0 * math.sin( ang ) + 50.0
  z = 200.0  # 100 <= z <= 300

  fiducial.proyect( theta, x, y, z )

  print "# Image", str(i)
  print "# (xyz) = ", x, y, z
  ang *= 180.0/math.pi

  fiducial.pts2DtoFile( "tmp" )
```

```
    # homografía es un programa en C.
    salida = subprocess.check_output(["../L3/homografia", "tmp"])
    datos = salida.split( )
    ex = float(datos[0]) - x
    ey = float(datos[1]) - y
    ez = float(datos[2]) - z
    ea = float(datos[3]) - angtheta

    errorx += ex
    errory += ey
    errorz += ez
    errora += ea

    errorx2 += ex*ex
    errory2 += ey*ey
    errorz2 += ez*ez
    errora2 += ea*ea

    i += 1

mx = errorx / 72.0
my = errory / 72.0
mz = errorz / 72.0
ma = errora / 72.0

sx = math.sqrt( errorx2 / 72.0 - mx*mx )
sy = math.sqrt( errory2 / 72.0 - my*my )
sz = math.sqrt( errorz2 / 72.0 - mz*mz )
sa = math.sqrt( errora2 / 72.0 - ma*ma )

print mx, sx
print my, sy
print mz, sz
print ma, sa
```

Bibliografía

[1] ATP World Tour. Ojo de halcón en directo en las next gen atp finals, 2017. http://www.atpworldtour.com/es/news/next-gen-atp-finals-2017-hawk-eye-live.

[2] Fifa.com. Hawk-eye, elegido proveedor de la tecnología de var, 2017. http://es.fifa.com/about-fifa/news/y=2017/m=5/news=hawk-eye-elegido-proveedor-de-la-tecnologia-de-var-2884454.html.

[3] P. Hawkins and D. Sherry. Video processor systems for ball tracking in ball games. (W0 0141884), June 2001.

[4] M. Fiala. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1317–1324, July 2010.

[5] M. Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596, Washington, DC, USA, June 2005. IEEE Computer Society.

[6] D. Flohr and J. Fischer. A lightweight id-based extension for marker tracking systems. In *Eurographics Sysmposium on Virtual Environments [EGVE]*, pages 59–64, 01 2007.

[7] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999. Proceedings. 2nd IEEE and ACM International Workshop on*, IWAR '99, pages 85–94, Washington, DC, USA, 1999. IEEE Computer Society.

[8] Raspberry Pi 3 model B specifications, 2017. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[9] Sony Semiconductor Solutions Corporation. Imx219pq specifications, 2016. http://www.sony-semicon.co.jp/products_en/new_pro/april_2014/imx219_e.html.

[10] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisiting the PnP problem: A fast, general and optimal solution. In *2013 IEEE International Conference on Computer Vision*, pages 2344–2351, December 2013.

[11] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011. DOI: 10.1109/CVPR.2011.5995464.

[12] C. Gonzalez Rafael and E. Woods Richard. *Digital Image Processing*. Addison-Wesley, 1992.

[13] Kenneth R. Castleman. *Digital Image Processing*. Prentice-Hall, New Jearsey, 1996.

[14] A. Zisserman R. Hartley. *Multiple view geometry in computer vision*. Cambridge University Press, NY, New York, 1st edition, 2000.

[15] Emanuele Trucco and Alessandro Verri. *Introductory techniques for 3-D computer vision*. Prentice Hall, New Jearsey, 1998.

[16] R. Szeliski. *Computer vision algorithms and applications*. Springer, 2011.

[17] Element 14. Newark, element14, 2018. http://mexico.newark.com/.

[18] P. M. Mhatre and M. Maniroja. Offline signature verification based on statistical features. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ICWET '11, pages 59–62, New York, NY, USA, 2011. ACM.

[19] Yaseen Moolla, Serestina Viriri, Jules R. Tapamo, and Fulufhelo V. Nelwamondo. Local directional pattern based signature verification using weighted fractional distance classification. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, SAICSIT '12, pages 79–83, New York, NY, USA, 2012. ACM.

[20] James M. Kang. *Distance Metrics*, pages 483–484. Springer International Publishing, Cham, 2017.

[21] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, February 1962.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[23] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[24] H. Cruz-Hernández and LG de la Fraga. A fiducial tag invariant to rotation, translation, and perspective transformations. *Pattern Recognition*, 81:213–223, 2018.

[25] LG de la Fraga and H. Cruz-Hernandez. Point set matching with order type. In *Mexican Conference on Pattern Recognition*, pages 229–237. Springer, 2018.

[26] Yujie Wang, Xiaoping Hu, Junxiang Lian, Lilian Zhang, Xiaofeng He, and Chen Fan. Geometric calibration algorithm of polarization camera using planar patterns. *Journal of Shanghai Jiaotong University (Science)*, 22(1):55–59, Feb 2017.

[27] Klaus Strobl and Gerd Hirzinger. More accurate pinhole camera calibration with imperfect planar target. 11 2011.

[28] Seung-Hae Baek, Pathum Rathnayaka, and Soon-Yong Park. Calibration of a stereo radiation detection camera using planar homography. *Journal of Sensors*, 2016.

[29] T. Collins and A. Bartoli. Infinitesimal plane-based pose estimation. *International Journal on Computer Vision*, 109, 2014.

[30] L.G. de la Fraga, N. A. García-Morales, D. Jaramillo-Olivares, and A. J. Ramírez-Díaz. A lightweight library for augmented reality applications. In *Mexican Conference on Pattern Recognition*, pages 221–228. Springer, 2018.