BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCIÓN EN AUTOMATIZACIÓN

"Sensor de distancia/orientación basado en 4 puntos"**

T E S I S

Presentada para obtener el título de:
Maestro en Ciencias de la Electrónica

Presenta:

Lic. Luis Ernesto Valencia Segura*

Directores:

Dr. Amparo Palomino Merino
Dr. Gibran Etcheverry Doger
Dr. Juan Antonio Escareño Castro

Puebla, México **Diciembre2016**

***BECARIO CONACYT**
**** TRABAJO FINANCIADO POR EL PROYECTO VIEP**

BUAP

Agradecimientos

Este trabajo se pudo llevar a cabo con éxito gracias al el apoyo brindado, por lo cual quiero expresar mis más sinceros agradecimientos.

A mi esposa Alicia Susana Rosales Alvarado, gracias a confianza y el apoyo que me brinda he podido superar esta etapa de mi vida.

A mis padres Sara Bertha Segura Roldan y Alejandro Alfonso Valencia García, los que siempre han estado conmigo en las buenas y en las malas, enseñándome que siempre hay que realizar un buen trabajo, de una manera honesta y responsable, gracias por su apoyo incondicional. También a mis hermanos que siempre están a mi lado ayudándome cuando más los necesito.

A la Benemérita Universidad Autónoma de Puebla (BUAP), por acogerme como un hijo más de esta honorable institución. Al Consejo Nacional de Ciencia y Tecnología (CONACyT), por su apoyo para la realización de este posgrado. También a la Vicerrectoría de Investigación y Estudios de Posgrado (VIEP) por el gran apoyo otorgado. A la Facultad de Ciencias de la Electrónica (FCE) por la experiencia y conocimientos brindado durante mi formación académica.

Un reconocimiento especial a la Dra. Amparo Palomino Merino, al Dr. Gibran. Etcheverry Doger y al Dr. Juan Antonio Escareño Castro por darme la confianza y el apoyo para desarrollar este tema de tesis. También tuve la oportunidad de conocer su calidad de seres humanos.

A mi jurado de tesis: Dr. Sergio Vergara Limon, Dr. Jaime Cid Monjaraz y la Dra. María Monserrat Morín Castillo por sus orientaciones y disponibilidad en la estructuración final de este trabajo.

A mis compañeros de generación: Ricardo De Gante, Alfredo López, Juan Carlos Gutiérrez, Daniel Gómez, Ricardo Torres, Miguel Ángel Ortega, Osvaldo Tepal, Froylan Bello y Araceli Luna, por todos los momentos compartidos y vividos, sobre todo el sorprendente apoyo que hemos tenido siempre apoyándonos mutuamente, creando un fuerte lazo de amistad.

Índice

Índice de figuras

Resumen

En el presente trabajo se utiliza la teoría de la geometría proyectiva, con el objetivo de que con una imagen obtenida de 4 puntos cooplanares, en el mundo físico, se calculen los seis grados de libertad del plano que contiene estos puntos, así, será posible saber tanto la orientación como la traslación de los puntos con respecto del plano de la cámara con la que se capturó la imagen.

Para llevar a cabo esta tarea se estudia la teoría de la geometría proyectiva, de la cual se desprende el algoritmo que calculara los parámetros de rotación y traslación del plano con base en una imagen. Una vez obtenido este algoritmo se implementará en software y hardware para verificar su validez.

Introducción

Planteamiento del problema

En aplicaciones de localización y orientación de los sistemas basados en robots, ya sean terrestres, aéreos o submarinos el problema de detectar la posición del vehículo, respecto de su objetivo, ha sido abordado ampliamente utilizando diferentes métodos (sistemas basados en central inercial, sensores IR, acelerómetros, visión, entre otros). La selección adecuada debe estar en función de la aplicación, sin embargo, el problema de la auto localización requiere de varios sensores, por ejemplo, un acelerómetro nos proporciona las aceleraciones en los tres ejes (x, y, z), por consiguiente, para obtener la posición se tiene que integrar dos veces, lo cual genera un error que con el pasar del tiempo aumenta cada vez más, haciéndose necesario utilizar métodos para eliminar el error como seria implementar un filtro Kalman. También se tiene el inconveniente de que el sistema por sí sólo no determina la posición inicial con respecto a su sistema de referencia. El problema de la ubicación del objeto, también se podría abordar un sensor GPS (Global Positioning System) el inconveniente aquí es que el sistema GPS tiene un error medido en metros, lo cual para vuelos en interiores no es adecuado, además de que para detectar o medir las orientaciones sería necesario hacer uso de algún otro sensor. Por otro lado, se podrían utilizar un conjunto de sensores para auto localizar el sistema, pero se tiene el inconveniente de que el hardware se complicaría al igual que el software o firmware.

Justificación

La visión por computadora puede resolver el problema de la autolocaización abordándolo desde distintos puntos de vista; sin embargo, el uso de un sensor es determinante, cuando el objetivo es el vuelo en cooperación, llamados también drones interactivos (conjunto de vehículos aéreos volando simultáneamente y comunicándose entre ellos), dado que se requiere de precisión y rapidez al mismo tiempo.

Propuesta de solución de tesis

Para este trabajo se implementó un método para la localizar un objeto en coordenadas físicas usando un sistema monocular. Este sistema captura imágenes de cuatro puntos pertenecientes a un plano y usando la teoría de la geometría proyectiva, la posición del plano es recuperada desde una imagen en 2D. Con este método obtenemos los 6 grados de libertad del plano en coordenadas físicas con respecto a las coordenadas de la cámara (tres de traslación y 3 de rotación).

Para poder obtener los parámetros de rotación y traslación de un objeto a partir de una imagen que contiene 4 puntos, es necesario obtener la matriz que nos relacione los puntos en el plano físico con los puntos en la imagen. Ya que esta matriz contiene toda la información de las relaciones entre los puntos de ambos planos (plano de la imagen y plano en el mundo físico), para determinar los parámetros que nos interesan, será necesario interpretar los datos de esta matriz.

Para resolver satisfactoriamente los objetivos planteados en esta tesis se propone un sistema como se presenta en la figura 1.

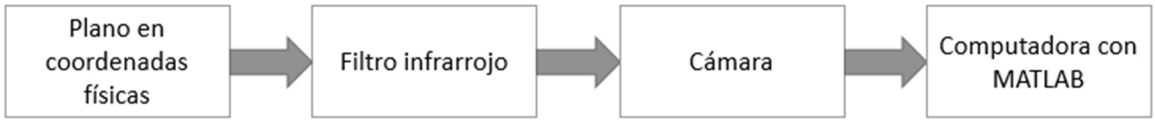


Figura 1. Diagrama a bloques del sistema.

Objetivos

Para llevar a cabo el proyecto de tesis, se fijaron los siguientes objetivos.

Objetivo General:

Desarrollar un sensor de estimación/medición de la altura y orientación de un plano con 4 puntos a través de una cámara infrarroja.

Objetivos específicos:

1) Estudiar los conceptos básicos de visión por computadora, geometría proyectiva, modelo de la cámara y sensores infrarrojos.
2) Estudia y calibrar la cámara infrarroja que permita obtener las ecuaciones de proyección.
3) Realizar la interface electrónica con microcontrolador Arduino que permita obtener los datos de la cámara.
4) Desarrollar la interfaz gráfica utilizando MATLAB/Simulink.
5) Estudio de los métodos de obtención de las funciones (lineales o no lineales) de descripción de la geometría proyectiva.
6) Pruebas experimentales.
7) Publicación de Resultados.
8) Escritura de la Tesis.

1 Antecedentes

1.1 Historia

La historia del PDI (Procesamiento Digital de Imágenes) se remonta a la década de los 60 y está directamente ligada con el desarrollo y evolución de las computadoras. Su progreso ha ido de la mano con el desarrollo de las tecnologías de hardware, ya que requiere un alto poder y recursos computacionales para almacenar y procesar las imágenes. De igual manera el desarrollo de los lenguajes de programación y los sistemas operativos han hecho posible el crecimiento continuo de aplicaciones relacionadas al procesamiento de imágenes, tales como: imágenes médicas, satelitales, astronómicas, geográficas, arqueológicas, biológicas, aplicaciones industriales, entre otras.

1.2 Estado del Arte

En el año 2009 el documento (Fernando Caballero, 2009) detalla la implementación de un sistema de seguimiento de cabeza adecuada para su uso en estaciones de teleoperación o centros de control, teniendo en cuenta las limitaciones y restricciones que suelen asociarse a esos ambientes. El documento analiza y justifica la selección de los diferentes métodos y sensores para construir el sistema de seguimiento de la cabeza, detallando también las etapas de procesamiento del sistema en funcionamiento. Un prototipo para validar el enfoque propuesto también se presenta junto con varias pruebas en un entorno real, con resultados prometedores [1].

En el año 2009 el documento (Guillermo Heredia, 2009) se presenta un método para aumentar la fiabilidad de vehículo aéreo no tripulado (UAV) sensor de detección de fallas e Identificación (IED) en un contexto multi-UAV. El Sistema de Posicionamiento Global Diferncial (DGPS) y sensores inerciales se utilizan para la IED sensor en cada UAV. El método utiliza estimaciones de posición adicionales que aumentan el sistema IED UAV individual. Estas estimaciones adicionales se obtienen utilizando imágenes de la misma escena plana tomado de dos UAV diferentes. Puesto que la precisión y el nivel de ruido de la estimación depende de varios factores, replanificación dinámica del equipo multi-UAV se puede utilizar para obtener una mejor estimación en caso de defectos causados por errores de crecimiento lento de estimación de la posición absoluta de que no se pueden detectar mediante el uso de IED locales en los vehículos aéreos no tripulados. También se presentan los resultados experimentales con los datos de dos UAV reales [2].

En 1995 en el artículo (Quan, 1995) explica que hay tres proyecciones invariantes de un conjunto de seis puntos en una posición en general en el espacio. El documento se concentra en el cálculo de las distancias constantes de un conjunto de seis puntos en el espacio desde tres imágenes tomadas con cámaras no calibradas, asumiendo que las correspondencias entre los puntos de las imágenes son conocidos [3].

En el año 2000 en el artículo (Dano, 2000) explica el desarrollo de un algoritmo el cual calcula la estructura proyectiva de un conjunto de seis puntos vistos en una secuencia de imágenes. El método está basado en la dualidad entre cámaras y los puntos señalados primero

por Carlsson y Weinshall. Esta implementación evita la debilidad inherente en implementaciones previas a este método. Este método calcula la matriz fundamental minimizando la función a una aproximación de primer orden [4].

En el año 2004 en el artículo (Ciwi, 2004), se enfoca en la auto localización basada en visión de un robot para interiores que trabaja con una sola cámara, basado en el principio de la geometría proyectiva y la imagen virtual extendida al plano, la posición transversa y la orientación del robot es calculado a través de la considerar al plano de la imagen y al plano de la planta como un conjunto. Después de un sencillo proceso off-line sobre el punto de referencia, el robot de servicio puedo funcionar bien en un edificio [5].

En 2009 en el trabajo (Ren Yuan, 2009) habla sobre la reconstrucción tridimensional que es una técnica para recobrar información en 3D de una escena de imágenes en 2D. El enfoque en la reconstrucción, en general cae en dos grupos de acuerdo al número de imágenes requeridas, el enfoque monocular y el enfoque binocular. El enfoque monocular de la reconstrucción en 3D se forma principalmente desde X, la investigación es escogida desde los contornos ya que estos son relativamente estables y esenciales en comparación con otros métodos tales como iluminación colores y texturas que pueden ser modificados en gran medida por las codificaciones hechas a las imágenes. El algoritmo de contorno presentado aquí emplea el proceso inverso de formación de la imagen el cual es considerado como simples proyecciones en perspectiva de círculos y rectángulos para reconstrucciones. Por un lado los métodos estrictos de la geometría proyectiva y el álgebra se utilizan para el cálculo de las orientaciones y profundidades relativas de que ubican nuestros planos con formas de interés con respecto a la observación de un sistema de coordenadas; Por otro lado se utiliza un poco de conocimiento previo para obtener profundidades absolutas y longitudes [6]**.**

1.3 Estrategias generales.

En visión por computadora existen dos estrategias generales para obtener información de un espacio 3D con base en imágenes en 2D: el primero utiliza dos imágenes de la misma escena con algunas diferencias generadas por la distancia entre cámaras (visión estéreo) y con ayuda de la geometría epipolar, se recupera la información en 3D de la escena. La segunda estrategia utiliza una sola imagen (visión monocular) para recuperar la información 3D perdida por el proceso de digitalización de una escena. En esta escena es necesario tener información previa de los componentes que la forman, para posteriormente, mediante la geometría proyectiva se pueda recuperar la información de pedida.

1.3.1 Visión estéreo

A pesar de la riqueza de información contenida en una imagen, la información de la profundidad de un punto no es directamente accesible en una imagen. Por otro lado, con al menos dos imágenes de la misma escena, la profundidad puede ser medida a través de triangulación. La información de dos imágenes de una escena, tomada desde diferentes puntos de vista, es combinada para determinar la estructura en 3D del mundo9 (figura 1.1).

Para esta tarea se utiliza la geometría epipolar y las restricciones epipolares que pueden ser representadas algebraicamente por una matriz de 3×3 llamada matriz esencial [7].

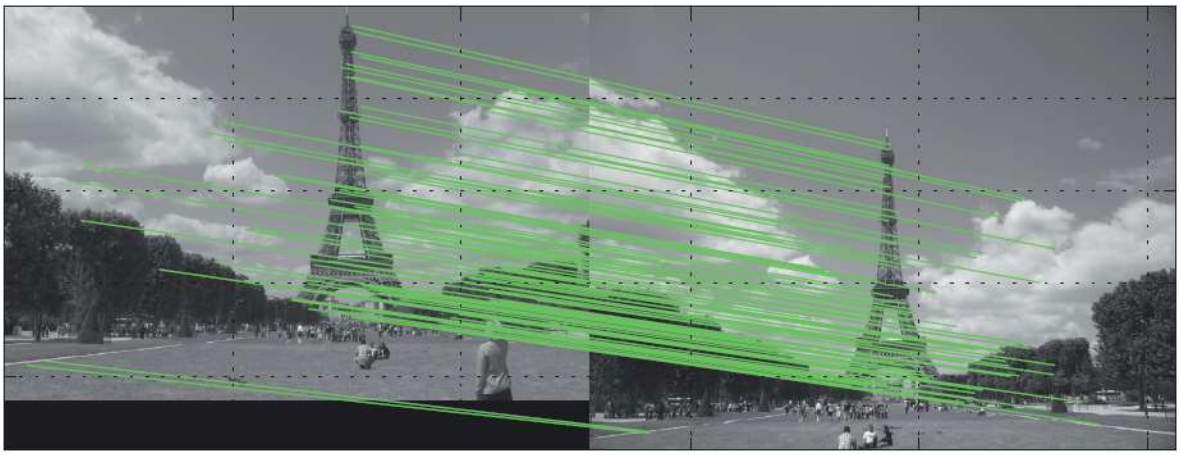


Figura 1.1. Conjunto de líneas que conectan dos puntos que concuerdan en dos dos imágenes tomadas de la misma escena con diferente ángulo de visión.

1.3.1.1 Geometría epipolar

Si consideramos la imagen p y p' de un punto P observado por dos cámaras con centro focal O y O'. Estos cinco puntos pertenecen al plano epipolar definido por las dos intersecciones de las líneas OP y O'P (figura 1.2). En particular, el punto p' permanece en la línea l' donde el plano epipolar y el plano de la imagen Π' de la segunda cámara se intersectan. La línea l' es la línea epipolar asociada con el punto p, y este pasa a través del punto e' donde la línea base que une os centros focales O y O' intersecta Π'. De igual manera, el punto p permanece en la línea epipolar l asociada con el puto p', y esta línea pasa través de la intersección e de la línea base con el plano Π [8].

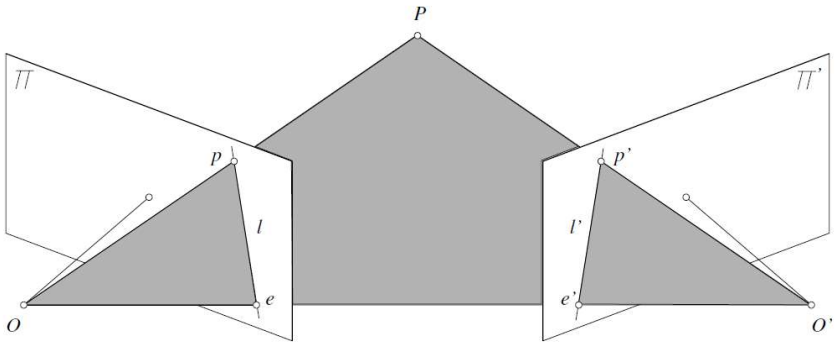


Figura 1.2. Geometría epipolar: el punto P, el centro focal O y O' de las dos cámaras, y las dos imágenes p y p' de P todas pertenecientes al mismo plano.

Los puntos e y e' son llamados los puntos epipolares de las cámaras. El punto epipolar e' es la imagen (virtual) del centro óptico O de la primera cámara en la imagen observada por la segunda cámara, y viceversa. Si p y p' son imágenes del mismo punto, entonces p' debe permanecer la línea epipolar asociada con p. Esta limitación juega un fundamental rol en la visión con dos cámaras y el análisis de movimiento. La parte más complicada del

análisis de datos de la visión estéreo es establecer las correspondencias entre dos imágenes, por ejemplo, decidir cuales puntos en la imagen derecha concuerdan con la imagen izquierda. La restricción epipolar limita en gran medida la búsqueda de estas correspondencias, ya que nosotros asumimos que las cámaras estén correctamente calibradas, las coordenadas del punto p determinan completamente el rayo que une O y p, y por lo tanto el plano epipolar definido por OO'p y la línea epipolar. La búsqueda de las concordancias de los puntos puede ser restringida a esta línea en lugar de toda la imagen (figura 1.3).

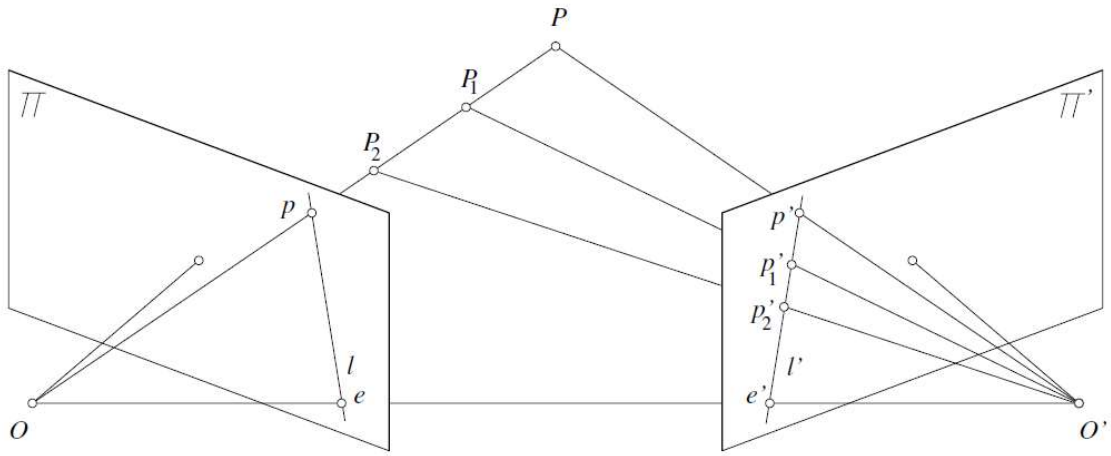


Figura 1.3. Restricción epipolar: dada una correcta calibración estéreo, el conjunto de posibles coincidencias para el punto p está restringido a permanecer en la línea epipolar asociada l'.

1.3.1.2 Visión monocular

El segundo procedimiento consiste en que lo que se encuentra en la imagen en dos dimensiones tenga unas claves (invariantes geométricas) que nos permitan reconstruir la realidad tridimensional. Esta es la idea de los distintos tipos de mapas y de la Geometría Proyectiva en general.

1.3.1.3 Geometría proyectiva

Cuando vemos una pintura, vemos cuadrados que no son cuadrados, o círculos que no son círculos. La transformación que mapea estos objetos planos dentro de la pintura es un ejemplo de transformación proyectiva. Como se puede ver en la figura 1.4 la forma del circulo parece una elipse, por lo tanto, se puede ver que las propiedades geométricas no se conservan, ya que, las longitudes de los radios perpendiculares de un circulo tienen diferentes tamaños, los ángulos tampoco se conservan al igual que las relaciones entre longitudes y ángulos. Sin embargo, las líneas rectas son lo único que permanece en una imagen bajo una transformación proyectiva y este es el requisito más general, y puede definir una transformación proyectiva de un plano [9]. Con esta información y con información a priori de la escena es posible recuperar la información 3D desde una sola imagen.

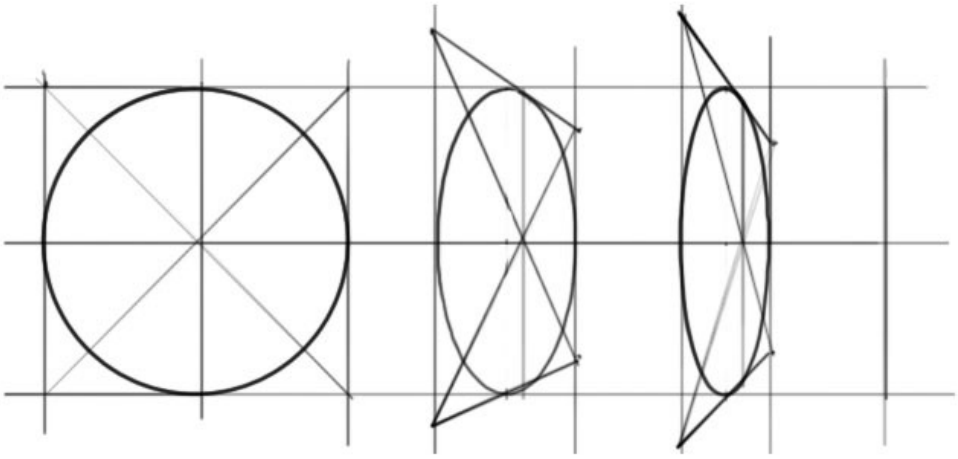


Figura 1.4. Imagen de un circulo bajo diferentes transformaciones proyectivas.

2 Marco teórico

2.1 Geometría proyectiva

2.1.1 Propiedades del plano proyectivo

El plano proyectivo es un concepto matemático destinado a modelar las propiedades geométricas de una secuencia de uno o más proyecciones en perspectiva. En el modelo de plano proyectivo, las transformaciones son representadas por mapeos del plano en sí mismo. Así una transformación puede ser vista como un reordenamiento de los puntos del plano proyectivo llamado coalineación [9]. El comportamiento de las estructuras geométricas, como las líneas, bajo coalineaciones es el principal foco de la teoría del plano proyectivo.

El efecto de una coalineación en las propiedades geométricas difiere de las transformaciones Euclidianas principalmente en dos aspectos:

i. **Distancia -** En el plano Euclidiano la distancia queda por la ecuación 2.2

$$\sqrt{(x - x_0)^2 + (y - y_0)^2} \tag{2.2}$$

Entre dos puntos, **P** y $\mathbf{P_0}$ no es afectada por transformaciones Euclidianas, i.e. traslación y rotación. Bajo perspectiva, la distancia entre dos puntos puede ser transformada por cualquier valor.

ii. **Líneas paralelas** - la imagen de líneas paralelas puede ser dos líneas intersectadas cuando se ve bajo perspectiva.

El plano proyectivo puede ser definido como una generalización del plano Euclidiano en el que algunas propiedades son removidas. Esta generalización procede en dos pasos. Primero, la noción de distancia es descartada, formando una estructura llamada el plano *afín*. La principal propiedad que caracteriza el plano afín es que el paralelismo es una invariante de las transformaciones afines. Bajo transformaciones afines, las coordenadas se someten a una escala anisotrópica, e. g. un cuadrado es transformado en un paralelogramo arbitrario. El plano afín es empleado a menudo en aplicaciones de visión como una aplicación razonable a la formación de imagen en perspectiva.

El modelo remueve el concepto de líneas paralelas. Todo par de líneas intersecta en algún punto único bajo la visualización perspectiva. Con el fin de tener en cuenta el caso en el que dos líneas son en realidad paralelas, i. e. que se unirán en el infinito, se introduce la noción de un punto ideal. Como podremos ver, la orientación de cada línea paralela, define un punto ideal diferente. Por ejemplo, la línea de horizonte en una imagen representa una línea hacia el infinito la cual ha sido proyectada en una línea de puntos finitos por la transformación perspectiva. En el modelo de plano proyectivo, los puntos ideales no se distinguen lo cual lleva a una considerable simplificación en el análisis de los efectos de la perspectiva.

Un plano afín con una línea de puntos unidos a partir de los cuales no se distingue es llamado el *plano proyectivo.* Los axiomas de incidencia para el plano proyectivo son como sigue:

A1. Dos puntos distintos determinan una línea única.

A2. Dos líneas distintas determinan un único punto.

Note que el axioma A2 no podría sostenerse en el plano afín dado que las líneas paralelas no se conocen, no se unen.

El aspecto más importante de estos axiomas es que son idénticos excepto en las palabras punto y línea. En efecto, nosotros podemos intercambiar las palabras e intercambias los axiomas A1 y A2. Así, se dice que en el plano proyectivo las líneas y puntos son duales. Cualquier teorema (propiedad) aplicado a líneas también aplica a puntos y viceversa. En una afirmación que involucra tanto a puntos y líneas, las dos palabras se pueden cambiar sin afectar a la verdad de la declaración. Encontraremos este concepto de dualidad muy importante en la aplicación de geometría proyectiva. Una vez que un resultado se ha elaborado para los puntos, se obtiene gratis un resultado similar para las líneas.

2.1.2 Modelos para el plano proyectivo

Como hemos visto, el plano Euclidiano familiar no puede ser empleado para modelar las propiedades de transformaciones proyectivas. Se necesitan nuevos modelos para proporcionar la comprensión sobre el comportamiento de las relaciones geométricas bajo la proyección.

Quizás el modelo más útil es proporcionado por un grupo de rayos en el espacio tridimensional, $\mathbb{R}^3$. Como se muestra en la figura 2.1, todos los rayos emanan de un origen común. Cada rayo representa un punto proyectivo. Sólo la dirección de un rayo es importante en el modelo. Suponga que un plano arbitrario π, que no pasa por el origen, es construido en $\mathbb{R}^3$. Los rayos que cruzan el plano corresponden a puntos en el plano afín. Los rayos que son paralelos a los puntos ideales del modelo de plano. El grupo de todos los rayos paralelos a π y que pasa a través del origen es un parámetro familiar y es matemáticamente equivalente a una línea, la línea ideal. Dado que el plano es arbitrario, cualquier punto a lo largo del rayo es equivalente a cualquier otro punto. Tampoco hay una distinción real entre los rayos en tanto que afines o puntos ideales, dado que π es arbitrario.

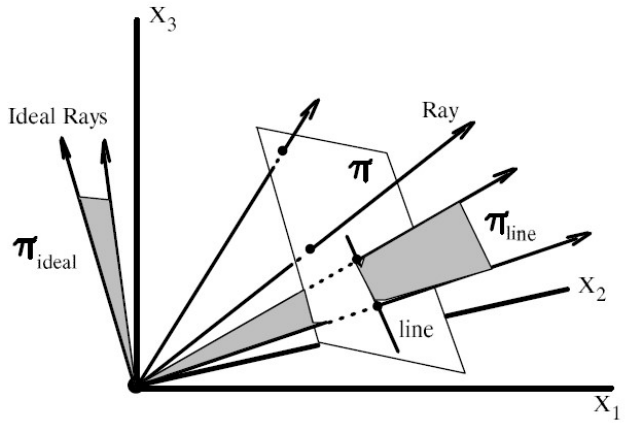


Figura 2.1. Un modelo del plano proyectivo puede ser construido por rayos en el espacio 3D. Los rayos corresponden a putos en el plano proyectivo. Dos rayos atravesando el origen definen un plano único atravesando el origen. Cualquier plano atravesando el origen corresponde a una línea proyectiva.

El plano a través del origen definido por cualquiera de dos rayos distintos es un modelo para la línea proyectiva. Para ejemplificar esto, considere cada uno de los dos axiomas de incidencia, A1 y A2. Primero, dos rayos siempre definen un único plano a través del origen, el cual satisface A1. Segundo, un rayo único existe siempre que sea la intersección de esos dos planos, proporcionando así un modelo para el axioma A2. El plano afín, π, intersecta el plano definido por dos rayos, π_{linea},. De nuevo, este plano es solo un modelo para una línea proyectiva y no se distingue de otros planos.

El modelo del rayo también ilustra el comportamiento de una perspectiva del mapeo entre planos. En este caso dos planos son introducidos, Π y π. El origen de los rayos puede ser considerado como un centro de proyección y un rayo intersecta ambos planos en los puntos correspondientes, transformados respectivamente. La misma relación puede conseguirse al considerar a Π y π como un plano único el cual se transforma de un plano al otro al transformar el sistema de coordenadas del espacio de rayos. Las intersecciones de los rayos con π y Π son modelos de la transformación proyectiva de puntos. Desde este modelo para una transformación perspectiva, es fácil ver que los puntos ideales pueden ser mapeados en puntos finitos y viceversa.

Es posible modelar la relación arbitraria entre π y Π al rotar y escalar los rayos en $\mathbb{R}^3$. Todas las posiciones y orientaciones de π con respecto a los rayos puede lograrse por una rotación y escalamiento de $\mathbb{R}^3$. La posición de las intersecciones con π a lo largo de un rayo no es importante así que la transformación de $\mathbb{R}^3$ puede involucrar ambas, la rotación y el escalamiento anisotrópico en general. La rotación y el escalamiento de $\mathbb{R}^3$ están representados por una matriz general de multiplicación 3×3, como se discute abajo. Se sigue que cualquier composición de las proyecciones proyectivas también corresponde a algunas rotaciones combinadas y escalamiento del espacio de los rayos.

Este modelo de rayos representa todas las propiedades geométricas del plano proyectivo por la interpretación de rayos como puntos y planos a través del origen como líneas. Una teoría analítica puede ser establecida al introducir las coordenadas del espacio tridimensional de los rayos.

2.1.3 Coordenadas homogéneas

Esta geometría analítica del plano proyectivo es una consecuencia directa de las propiedades algebraicas de las coordenadas del espacio tridimensional de los rayos. De acuerdo al modelo recientemente desarrollado, un punto en el plano proyectivo es representado por tres coordenadas cartesianas, $\boldsymbol{p} = (x_1, x_2, x_3)^t$, el cual representa un rayo a través del origen en el espacio tridimensional. $(x_1, x_2, x_3)^t$ son llamadas coordenadas homogéneas porque las expresiones algebraicas representan formas, tales como las cónicas, se convierten en ecuaciones homogéneas cunado se expresan como polinomios en $(x_1, x_2, x_3)^t$[9]. Solo la dirección de los rayos es importante, así todos los puntos de la forma $\lambda\mathbf{p} = (\lambda x_1, \lambda x_2, \lambda x_3)^{\boldsymbol{t}}$ son equivalentes. A la inversa, todas las propiedades proyectivas de un punto deben mantenerse independientes del valor de λ. Claramente, la dirección de un rayo de longitud cero no está definida. Las coordenadas homogéneas correspondientes (0, 0, 0) no tienen significado, y es indefina en el plano proyectivo.

Una relación en el plano a las coordenadas Cartesianas convencionales, (x, y), puede ser establecida al construir un plano especial, π_e, el cual es perpendicular al eje x_3- y en unidad de distancia a lo largo de x_3. La intersección de los rayos $\mathbf{p}$ es el punto, $\mathbf{p_e} = (x, y, 1)$, donde el par (x, y) corresponde al estándar de las coordenadas Cartesianas de $\mathbf{p}$. Como discutimos arriba, un rayo paralelo a π_e es llamado un punto ideal. Cualquier punto ideal por consiguiente tiene $x_3 = 0$. La condición $x_3 = 0$ define una línea llamada la línea ideal.

No es necesario identificar una unidad de distancia a lo largo de x_3 para la localización de π_e. En cambio, las coordenadas cartesianas correspondientes a un punto proyectivo son definidas por,

$$\mathbf{p_e} = (\frac{x_1}{x_3}, \frac{x_2}{x_3}, 1)^t = (x, y, 1,)^t \tag{2.3}$$

De este modo la posición del plano no afecta el valor de las coordenadas Cartesianas. No es esencial que el plano sea perpendicular a x_3 pero este es el estándar convencional así que los puntos ideales se denotan por x_3=0.

2.1.4 La línea proyectiva

La representación de coordenadas de una línea en el plano proyectivo se deriva de la representación analítica de un plano general que atraviesa el origen del espacio de rayos. La ecuación de este plano es dada por,

$$u_1x_1 + u_2x_2 + u_3x_3 = 0 \tag{2.4}$$

El coeficiente del plano $u = (u_1, u_2, u_3)^t$ corresponde a las coordenadas homogéneas para la línea proyectiva. De nuevo, $\lambda\mathbf{u}$ es la misma línea que **u.** Tome en cuenta que la ecuación es homogénea, a partir de que el grado de cada término es el mismo. El caso de $u_3 = 0$ corresponde a una línea que atraviesa el origen. La línea ideal es $\mathbf{u} = (0, 0, 1)^t$ lo cual tiene la ecuación, $\boldsymbol{x_3} = 0$. La ecuación proyectiva de una línea puede ser representada en varios vectores y la matriz de los vectores.

$$\mathbf{u} \cdot \mathbf{p} = \mathbf{u}^t\mathbf{p} = \mathbf{p}^t\mathbf{u} = 0 \tag{2.5}$$

La dualidad de puntos y líneas es indicada por la forma simétrica de estas ecuaciones. Esto es, el rol de **u** y de **p** puede ser intercambiado son afectar la forma de la ecuación. La proyección homogénea de la línea puede estar relacionada a la ecuación de la línea Cartesiana estándar. En las coordenadas Cartesianas la ecuación de una línea es:

$$n_x x + n_y y - d = 0 \tag{2.6}$$

Donde $\mathbf{n} = (n_x, n_y)^t$ es la normal a la línea y d es la distancia desde el origen a la línea en la dirección perpendicular a la línea. Nosotros podemos comparar esta expresión a la ecuación de la línea homogénea y determinar la relación entre los parámetros de la línea Cartesiana y los coeficientes de las líneas homogéneas. Los componentes de la línea normal son,

$$n_x = -d\frac{u_1}{u_3} \qquad\qquad n_y = -d\frac{u_2}{u_3} \tag{2.7}$$

Mostrando que la normal a la línea Cartesiana es justamente la proyección en el plano xy de la normal, (u_1, u_2, u_3), del plano correspondiente en el espacio de los rayos.

El concepto de punto de fuga discutido antes puede ser relacionado a las coordenadas homogéneas como sigue. Cualquier punto de fuga, $\mathbf{p^v}$, corresponde a un punto ideal $x_3 = 0$. Así, $\mathbf{p^v}$ debe ser de la forma $\mathbf{p^v} = (p_1^v, p_2^v, 0)$. La ecuación de una línea incidental al punto de fuga es dada por

$$u_1 p_1^v + u_2 p_2^v = 0 \tag{2.8}$$

Así, para toda línea tal

$$\frac{u_1}{u_2} = -\frac{p_2^v}{p_1^v} \tag{2.9}$$

Y de la ecuación 2.9

$$\frac{n_x}{n_y} = -\frac{p_2^v}{p_1^v} \tag{2.10}$$

Así $(p_1^v, p_2^v, 0)$ debe corresponder a la dirección de la línea, i. e.

$$(n_x, n_y) \cdot (p_1^v, p_2^v) = n_x p_1^v + n_y p_2^v = 0 \tag{2.11}$$

Se sigue que cualquier línea incidente (en el plano) con el punto de fuga tendrá la misma dirección, a partir de que el radio n_x/n_y se arregla. Cuando este grupo de líneas paralelas es mapeado proyectivamente las líneas ya no son necesariamente paralelas. Como sea, todas ellas seguirán siendo incidentes con la proyección de $\mathbf{p}^v$. Esta concepción muestra que el concepto de paralelismo no tiene importancia en la geometría proyectiva a partir de que no existe nada que identifique a $\mathbf{p}^v$ como un punto ideal. Con el fin de definir el concepto de paralelismo, es necesario distinguir un grupo de puntos ideales de otros puntos en el plano proyectivo. Las líneas son paralelas si se cruzan en un punto ideal. Aumentar el plano proyectivo con la estructura adicional de los puntos ideales resulta en un plano afín.

Es necesario especificar una línea de puntos ideales en el plano antes de que el paralelismo pueda ser definido.

2.1.5 Transformaciones proyectivas

Como se muestro, una transformación proyectiva entre dos planos proyectivos puede ser representada por una transformación lineal general del espacio de rayos, $(x_1, x_2, x_3) = \mathrm{H}(X_1, X_2, X_3)^t$. Puede ser demostrado que todas las propiedades de una transformación proyectiva general son representadas por la matriz de transformación. A la inversa, cualquier transformación (no lineal) de coordenadas homogéneas define una transformación proyectiva del plano proyectivo. Esta forma de transformación proyectiva es una beneficiosa clave de introducción de coordenadas homogéneas, ya que muchos resultados importantes pueden ser obtenidos directamente por manipular matrices simples y expresiones vectoriales.

Ya que el plano proyectivo tiene tres coordenadas homogéneas la transformación es representada por una matriz 3×3 con 8 parámetros esenciales. La escala en general de la matriz no es importante ya que los puntos proyectivos son equivalentes hasta el multiplicador λ. El grupo de distintas transformaciones proyectivas es un subespacio de ocho dimensiones del espacio de nueve dimensiones definido por los elementos de la matriz.

La transformación general proyectiva de un plano proyectivo, **Π** a otro, **π**, es representada como

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} h_{11}\ h_{12}\ h_{13} \\ h_{21}\ h_{22}\ h_{23} \\ h_{31}\ h_{32}\ h_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \tag{2.12}$$

o

$$\mathbf{x} = \mathbf{HX}$$

Si la transformación es representada en coordenadas cartesianas la naturaleza no lineal de la transformación proyectiva en espacio Euclidiano o afín es aparente.

$$x = \frac{x_1}{x_3} = \frac{h_{11}X + h_{12}Y + h_{13}}{h_{31}X + h_{32}Y + h_{33}} \tag{2.13}$$

$$y = \frac{x_2}{x_3} = \frac{h_{21}X+h_{22}Y+h_{23}}{h_{31}X+h_{32}Y+h_{33}} \tag{2.14}$$

De nuevo, la escala general del factor T no afecta las coordenadas Cartesianas ya que el mismo factor aparece en el numerador y el denominador de cada expresión. Esta forma racional lineal representa el efecto en coordenadas Cartesianas de una transformación proyectivas y vale (cuenta) para una secuencia de proyecciones centrales entre dos planos en el espacio. Consecuentemente, es de mayor importancia en aplicaciones de visión.

2.1.6 Transformación proyectiva de líneas.

Ya que las líneas y los puntos son duales en los planos proyectivos, la transformación de las coordenadas de las líneas es también una transformación lineal. Considere la ecuación para un punto de incidencia con una línea definida anteriormente,

$$U_1X_1 + U_2X_2 + U_3X_3 = 0 \tag{2.15}$$

$$\mathrm{U}^{\mathrm{t}}\mathbf{P} = 0$$

Sí un punto, $\mathbf{P}$, se transforma como $\mathbf{p} = \mathrm{H}\mathbf{P}$, entonces $\mathbf{P} = \mathrm{H}^{-1}\mathbf{p}$. Sustituyendo esta información inversa en la línea la ecuación resulta en

$$\mathrm{U}^{\mathrm{t}}\mathrm{H}^{-1}\mathbf{p} = 0 \tag{2.16}$$

Se ve que la colinearidad de puntos se preserva bajo una transformación proyectiva ya que la forma general del producto escalar no es afectada por la transformación homogénea lineal. Más allá, la línea de la ecuación trasformada es $\mathbf{u}^{\mathrm{t}}\mathbf{p}$, y las coordenadas de la línea transformada, **u**, deben ser

$$\mathbf{u} = [\mathrm{H}^{-1}]^{\boldsymbol{t}}\,\mathbf{U} \tag{2.17}$$

Lo cual a menudo se representa de forma abreviada

$$\mathbf{u} = \mathrm{H}^{-1}\mathbf{U} \tag{2.18}$$

Así, las líneas en el plano proyectivo se transforman linealmente, tal como puntos, pero la correspondiente matriz de transformación es la transpuesta de la inversa de la matriz que define el punto de transformación.

Este mapeo de puntos en puntos y líneas en líneas es llamado una coalineación. El término coalineación implica que ninguna transformación proyectiva del plano preserva la colinearidad de un grupo de puntos. Así la forma de la transformación de puntos y líneas es la misma, es natural también considerar la posibilidad de una transformación del plano proyectivo que torne puntos en líneas y líneas en puntos. Así una transformación es llamada una correlación. Nosotros encontraremos el concepto de la correlación útil en la interpretación de propiedades proyectivas de curvas algebraicas. Por ejemplo, un cono define una correlación entre polos y líneas polares correspondientes.

2.1.7 **Cuatro puntos definen una transformación proyectiva**

La matriz de transformación proyectiva, H, requiere ocho parámetros independientes para definir un mapeo único. Así cada punto en el plano proporciona dos ecuaciones de coordenadas Cartesianas, es necesario encontrar cuatro correspondencias de puntos entre dos planos proyectivamente transformados únicamente para definir la matriz de transformación. La escala general de T es arbitraria, así nosotros podemos escoger $t_{33} = 1$. Dejando a los cuatro puntos correspondientes ser representados por, $(\lambda_i x_i, \lambda_i y_i, \lambda_i)^t = \mathrm{H}(X_i, Y_i, 1)^t$. El sistema lineal resultante de la ecuación es,

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -x_3Y_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y_3X_3 & -y_3Y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -yX_4 & -yY_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} \tag{2.19}$$

La existencia de este sistema lineal asegura, en principio, la unicidad de **H**, dados cuatro puntos correspondientes, dando la condición de que ninguno de los tres puntos son colineales.

Estos resultados guían inmediatamente a un sistema de coordenadas proyectivas canónico basado en cuatro puntos, donde las propiedades de las figuras geométricas pueden ser invariablemente representadas. Una opción obvia es seleccionar el marco canónico para que sea un único cuadro donde los puntos transformados de referencia tienen coordenadas $\{(0,0,1)^t,\ (1,0,1)^t, (0,1,1)^t, (1,1,1)^t\}$. Cualquier referencia cuadrilateral puede ser transformada proyectivamente en este cuadrado de unidad y entonces, las relaciones geométricas de otros puntos y líneas pueden ser invariantemente representadas en el marco canónico. Esta aproximación se toma en representación de planos de formas curvas para proporcionar una firma invariante para la curva de reconocimiento. Otro marco canónico de coordenadas se introduce en la geometría proyectiva clásica el cual simplifica el análisis de algunas configuraciones geométricas y en particular es útil en el análisis de las propiedades proyectivas de los conos. El sistema de coordenadas basado en un triángulo de referencia, se muestra en la figura 2.2.

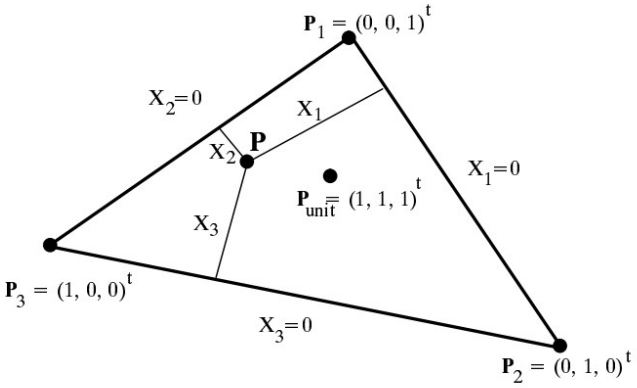


Figura 2.2. El espacio de coordenadas proyectivas estándar definida por cuatro puntos, y llamada el triángulo de referencia. Las coordenadas homogéneas son definidas como las distancias perpendiculares desde un punto hasta las líneas perimetrales del triángulo. La distancia escalada es determinada por el punto unit.

Tres de los puntos son empleados para definir un triángulo de referencia. Las coordenadas estándar son asignadas a los puntos como se muestra. El cuarto punto es coordenadas asignadas, $\mathbf{p}_{unit} = (1,1,1)^t$ y es llamado el punto único. Una vez que los cuatro puntos han sido especificados, las coordenadas homogéneas de la representación del punto Cartesiano, $\mathbf{p} = (x, y)^t$, son definidas por,

$$x_1 = \lambda\,(U_{i1}x + U_{i2}y + U_{i3}) \tag{2.20}$$

Donde $\boldsymbol{u_i}$ son las líneas de coordenadas de los bordes o lados del triángulo de referencia. La ecuación 2.20 representa la ecuación de la línea para un punto tendido sobre uno de los lados del triángulo. El valor de esta expresión es cero si el punto $(x, y, 1)^t$ yace en la línea. De otra manera el valor es proporcional a la distancia perpendicular del punto a la línea. El valor de cada coordenada homogénea x_1, es la distancia Euclidiana del punto dado del lado correspondiente del triángulo de referencia. Por ejemplo, x_1 es la distancia de la línea, $\overline{P_1P_2}$, i.e. $x_1 = 0$. El factor de escala arbitrario de la distancia λ está determinado al dejar $\mathbf{p} = \mathbf{p}_{unit}$.

2.2 Transformaciones

Como ya se mencionó una transformación proyectiva planar es una transformación lineal en vectores homogéneos representados por una matriz de 3×3 no singular.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \tag{2.21}$$

O en su forma reducida $x = HX$

Se puede notar que la matriz H en la ecuación 2.21 puede ser multiplicada por cualquier número diferente de cero (factor de escalamiento) sin alterar la transformación. Por lo tanto, decimaos que la matriz H es una matriz homogénea, ya que como en la

representación homogénea de puntos, solo las relaciones de los elementos de la matriz son significantes. Para la matriz H existen 8 relaciones entre los nueve elementos que la componen, y esto no da como resultado que una transformación proyectiva tiene 8 grados de libertad.

Una transformación proyectiva proyecta cada figura en otra figura proyectivamente equivalente, dejando todas sus propiedades proyectivas invariantes. En el modelo de rayo de la figura 2.3 una trasformación proyectiva es simplemente una trasformación lineal en $\mathbb{R}^3$.

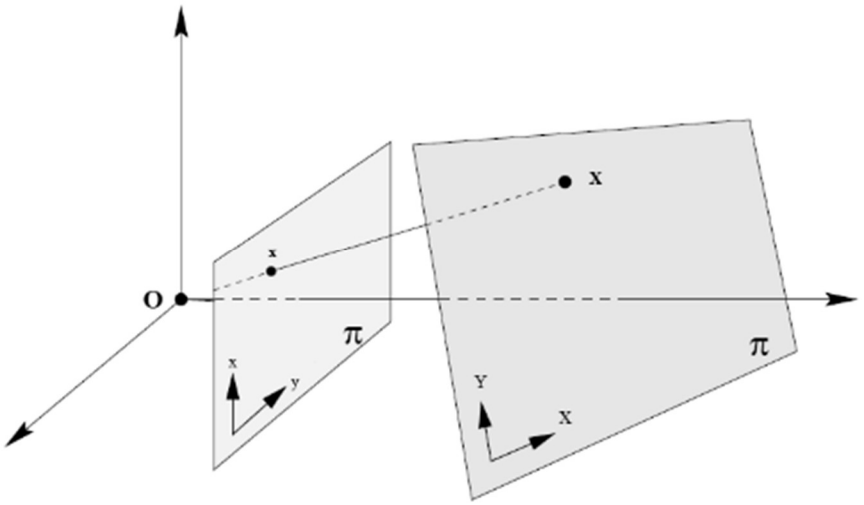


Figura 2.3. Mapeo de un punto en u plano a otro plano mediante una trasformación proyectiva.

2.2.1 Mapeo entre planos

La proyección entre rayos atreves de un punto común (centro de la proyección) define un mapeo desde un plano hasta otro. También este mapeo punto a punto preserva las líneas que perteneces a un plano hasta el otro plano. Si un sistema de coordenadas está definido en cada plano y los puntos están representados en coordenadas homogéneas, entonces el mapeo de proyección central puede ser expresado por $x = Hx$. De hecho, si los dos sistemas de coordenadas están definidos en dos planos, ambos en el espacio euclidiano (coordenadas rectangulares) entonces el mapeo definido por la proyección central está más restringido que una trasformación proyectiva arbitraria. Esta trasformación se prefiere llamar una trasformación perspectiva que una proyección completa.

Para ilustrar el efecto que tiene este tipo de transformación en perspectiva se presenta la figura 2.4, donde se puede ver como la trasformación en perspectiva elimina la distorsión de la perspectiva de la imagen original. Para calcular la transformación proyectiva se inicia seleccionando una sección de la imagen correspondiente a una sección plana del mundo real. La imagen 2D (figura 2.4 a) y las coordenadas mundiales son seleccionado como se muestra en la figura 2.3. Se escogen un par de puntos coincidentes X y x en el plano del mundo y en el plano de la imagen respectivamente tal que en coordenados no homogéneas sean (X, Y) y (x, y). Utilizamos coordenadas no homogéneas aquí en lugar de las coordenadas homogéneas de los puntos, porque son estas coordenadas no homogéneas que se miden directamente desde

la imagen y desde el plano del mundo. Con las ecuaciones 2.13 y 2.14 se pueden pasar las coordenadas no homogéneas a coordenadas homogéneas, lo que nos da como resultado las siguientes expresiones.

$$x = \frac{x_1}{x_3} = \frac{h_{11}X+h_{12}Y+h_{13}}{h_{31}X+h_{32}Y+h_{33}} \qquad y = \frac{x_2}{x_3} = \frac{h_{21}X+h_{22}Y+h_{23}}{h_{31}X+h_{32}Y+h_{33}} \tag{2.22}$$

Cada punto correspondiente genera dos ecuaciones para los elementos de H, que después de multiplicarlos nos quedan como:

$$x(h_{31}X + h_{32}Y + h_{33}) = h_{11}X + h_{12}Y + h_{13} \tag{2.23}$$

y

$$y(h_{31}X + h_{32}Y + h_{33}) = h_{21}X + h_{22}Y + h_{23} \tag{2.24}$$

Estas ecuaciones son lineales. Las correspondencias de cuatro puntos conducen a ocho ecuaciones lineales de este tipo en las entradas de H, que son suficientes para resolver H hasta un factor multiplicativo insignificante. La única restricción es que los cuatro puntos deben estar en "posición general", lo que significa que no hay tres puntos colineales. La inversa de la transformación H calculada anteriormente se aplica entonces a toda la imagen para deshacer el efecto de la distorsión de perspectiva en el plano seleccionado (figura 2.4b).

a b

Figura 2.4. Removiendo la distorsión causada por la perspectiva a) imagen original b) imagen obtenida después aplicar la transformación proyectiva.

2.3 Invariantes

Una forma de describir las transformaciones “algebraicas”, como sería una matriz que actúa sobre coordenadas de un punto, es describir la transformación en términos de esos elementos o cantidades que son preservados o son invariantes. Un invariante (escalar) de una configuración geométrica es una función de la configuración cuyo valor no cambia por una transformación particular [8]. Por ejemplo, la separación de dos puntos no se modifica por

una transformación euclidiana (traslación y rotación), pero no por una transformación semejante (por ejemplo, traslación, rotación y escalamiento isotrópico). La distancia así es un invariante euclidiana, pero no semejante. El ángulo entre dos líneas es una invariante para ambas transformaciones.

2.3.1 Clase I: Isométrica

Las transformaciones isométricas son transformaciones del plano en $\mathbb{R}^2$ que preserva las distancias euclidianas. Una transformación isométrica es representada en la forma de la ecuación 2.25.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} \epsilon\cos(\theta) & -sen(\theta) & t_x \\ \epsilon\, sen(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.25}$$

Donde $\epsilon = \pm 1$. Si $\epsilon = 1$ entonces la isométrica preserva la orientación y es una transformación euclidiana (una combinación de traslación y rotación). Si $\epsilon = -1$ entonces la isométrica invierte la orientación.

El modelo de transformación euclidiana puede ser escrito más conciso en forma de bloques como se muestra en la siguiente ecuación.

$$x = \boldsymbol{H_E} X = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} X \tag{2.26}$$

donde **R** es una matriz de rotación de 2×2 (una matriz ortogonal tal que $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$), **t** es un vector de dimensión dos de traslación y **0** un vector de dimensión 2 con sus componentes en cero. Una transformación euclidiana es también conocida como un desplazamiento.

Invariantes. Los invariantes son muy familiares, por ejemplo: longitud (la distancia entre dos puntos), ángulo (el ángulo entre dos líneas) y área.

Grupos y orientación. Una isometría preserva la orientación si la matriz **R** tiene determinante 1. Las isométricas que preservan la orientación forman un grupo, las de orientación inversa no lo hacen. Esta distinción se aplica también en el caso de la semejanza y las transformaciones afines.

2.3.2 Clase II: Transformación de similitud

Una transformación de similitud es una isométrica compuesta de un escalamiento isotrópico. En el caso de una transformación euclidiana compuesta con escalamiento (no reflexión) la semejanza tiene una representación matricial como lo muestra la siguiente ecuación.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} s\,cos\theta & -s\,sen\theta & t_x \\ s\,sen\theta & scos\theta & {}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.27}$$

La ecuación 2.27 puede ser escrita en forma de bloques y queda de la siguiente manera.

$$\boldsymbol{x} = \boldsymbol{H_s X} = \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ 0^T & 1 \end{bmatrix} \boldsymbol{X} \tag{2.28}$$

donde el escalar *s* representa el escalamiento isotrópico. Una transformación semejante preserva la "forma".

Invariantes. Los invariantes pueden ser construidos desde los invariantes euclidianos agregando el grado adicional de libertad de escalado. Los ángulos entre las líneas no son afectados por la rotación, traslación o el escalamiento isotrópico, y son por lo tanto invariantes de similitud. En particular las líneas paralelas son mapeadas en líneas paralelas. La distancia entre dos puntos no es un invariante de similitud, pero la relación de dos distancias si es un invariante, porque el escalamiento es en todas las longitudes. Similarmente la relación de áreas es un invariante.

2.3.3 Clase III: Transformación afín

Una transformación afín es una transformación lineal no singular seguida por una traslación. Su representación matricial es la siguiente.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.29}$$

O en forma de bloque.

$$\boldsymbol{x} = \boldsymbol{H_A X} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ 0^T & 1 \end{bmatrix} \boldsymbol{X} \tag{2.30}$$

Siendo **A** una matriz no singular. Una transformación planar afín tiene seis grados de libertad correspondientes a los seis elementos de la matriz. La transformación se puede calcular a partir de tres correspondencias de puntos.

Una manera útil de entender los efectos geométricos de la componente lineal **A** de una trasformación afín es como la composición de dos transformaciones fundamentales, cruzando rotación y escalamiento no isotrópico. La matriz afín **A** puede ser descompuesta como:

$$\boldsymbol{A} = \boldsymbol{R}(\theta)\boldsymbol{R}(-\phi)\boldsymbol{D}\boldsymbol{R}(\phi) \tag{2.31}$$

Donde $\boldsymbol{R}(\theta)$ y $\boldsymbol{R}(\phi)$ son rotaciones por θ y ϕ respectivamente, y **D** es la matriz diagonal:

$$\boldsymbol{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{2.32}$$

La matriz afín A es la concatenación de una rotación (por ϕ); un escalamiento de λ_1 y λ_2 respectivamente en las direcciones *x* y *y*; una rotación de regreso (por $-\phi$); y finalmente otra rotación (por θ). Lo único nuevo geométricamente, comparado a la semejanza, es el

escalamiento no isotrópico. Los grados de libertad aumentan en dos con respecto a la transformación semejante. Estos son, el ángulo ϕ que especifica la dirección de desplazamiento y la relación de los parámetros de escala $\lambda_1 : \lambda_2$. Esencialmente la transformación afín esta escala en direcciones ortogonales, orientadas en un ángulo particular como se puede ver en la figura 2.5.

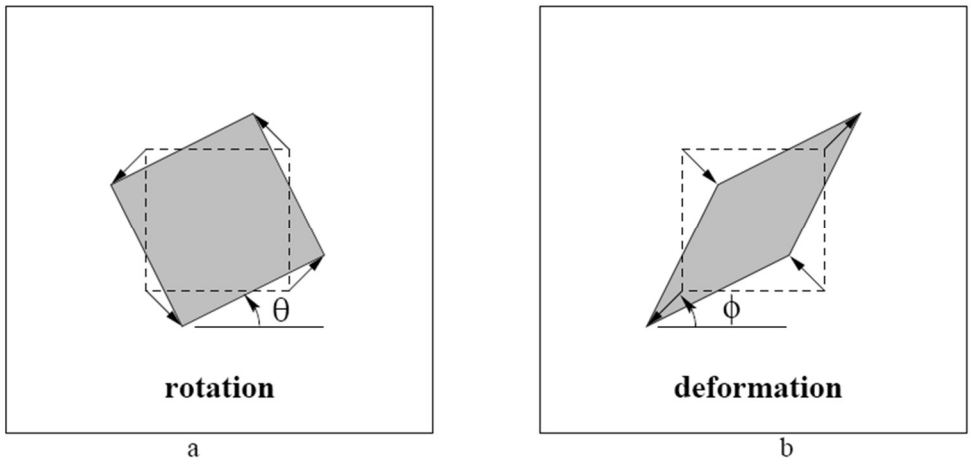


Figura 2.5. Distorsión de una transformación afín.

Invariantes. Ya que una transformación afín incluye escalamiento no isotrópico, las invariantes de similitud como la relación de distancias y ángulos entre líneas no son preservadas bojo una afinidad. Tres importantes invariantes son:

i. **Líneas paralelas.** Considerando dos líneas paralelas las cuales se intersectan en un punto $(x_1, x_2, 0)^T$ en el infinito. Bajo una transformación afín este punto es mapeado a otro punto en el infinito. Consecuentemente, las líneas paralelas son mapeadas a líneas que se intersectan en el infinito, y por lo tanto son paralelas después de la transformación.

ii. **Relación de longitudes de segmentos de línea paralelas.** La escala de longitud de un segmento de línea depende únicamente del ángulo entre la dirección de la línea y las direcciones de escala. Supongamos que la recta está en el ángulo α con respecto al eje x de la dirección de escalamiento ortogonal, entonces la magnitud de escala es $\sqrt{\lambda_1^2 \cos^2 \alpha + \lambda_2^2 sen^2 \alpha}$. Este escalamiento es común a todas las líneas con la misma dirección.

iii. **Relación de áreas.** Esta invariante puede ser deducida directamente desde la descomposición de la ecuación 2.28. Las rotaciones y traslaciones no afectan el área, solo los escalamientos por λ_1 y λ_2 son importantes. El efecto que el área sea escalada por $\lambda_1 \lambda_2$ el cual es igual al $\det(\boldsymbol{A})$. Así el área de cualquier forma es escalada por el $\det(\boldsymbol{A})$, y, por lo tanto, el escalamiento se anula para la relación entre áreas.

Una afinidad preserva la orientación o reversa de acuerdo a si el $\det(\boldsymbol{A})$ es positivo o negativo, respectivamente. Ya que el $\det(\boldsymbol{A}) = \lambda_1\lambda_2$ depende solo del signo del escalamiento.

2.3.4 2.4.4 Clase IV: Transformación proyectiva

La transformación proyectiva se definió en la sección 2.3, esta es una transformación general lineal de coordenadas homogéneas. Esta generaliza una transformación afín, que es la composición de una transformación lineal general no singular de coordenadas no homogéneas y una traslación. La forma en bloques de la matriz de transformación proyectiva es la siguiente.

$$\boldsymbol{x} = \boldsymbol{H_P}\,\boldsymbol{X} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{v^T} & v \end{bmatrix} \boldsymbol{X} \tag{2.33}$$

Donde el vector $\boldsymbol{v} = (v_1, v_2)^T$. La matriz tiene nueve elementos con sus relaciones significativas, por lo que la transformación se especifica por ocho parámetros. Note que no siempre es posible escalar la matriz ya que la unidad *v* puede ser cero. Una transformación proyectiva entre dos planos puede ser calculada desde la correspondencia de cuatro puntos, con tres puntos no colineales.

Invariantes. El invariante más fundamental en la transformación proyectiva es la relación de cruce de cuatro puntos coloniales: una relación de longitudes en una línea es un invariante de una transformación afín, pero no lo es en una transformación proyectiva. Sin embargo, una relación de relaciones o relación de cruz de longitudes en una línea es una invariante proyectiva.

2.3.5 Relación de cruce

La relación de cruce es la relación está dada por 4 puntos sobre una línea y esta relación es preservada después de una transformación proyectiva. Existen varios resultados en la geometría proyectiva los cuales resultan de una interpretación en términos de la relación de cruce. La relación de cruce se explica con base en la figura 2.6. Como ya se mencionó, la relación de distancias no es preservada bajo una transformación proyectiva, sin embargo, la relación de relaciones de distancias es invariante. La relación de cruce está definida por la siguiente ecuación

$$Cr(\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3, \boldsymbol{P}_4) = \frac{(X^3 - X^1)}{(X^3 - X^2)} \frac{(X^4 - X^2)}{(X^4 - X^1)} \tag{2.34}$$

Donde X^1, X^2, X^3, X^4 son la correspondiente posición de cada punto a lo largo de la línea, por ejemplo, $(X^3 - X^1)$ es la distancia entre $\boldsymbol{P}_3$ y $\boldsymbol{P}_1$.

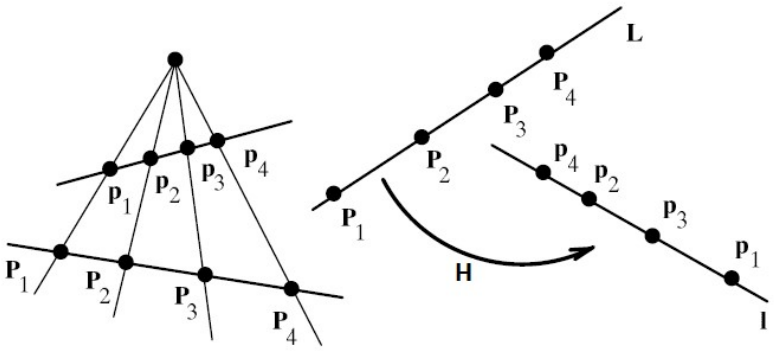


Figura 2.6. Un modelo del plano proyectivo puede ser construido por rayos en el espacio 3D. Los rayos corresponden a putos en el plano proyectivo. Dos rayos atravesando el origen definen un plano único atravesando el origen. Cualquier plano atravesando el origen corresponde a una línea proyectiva.

2.3.6 Descomposición de una trasformación proyectiva

Una transformación proyectiva puede ser descompuesta en una cadena de transformaciones, donde cada matriz en la cadena representa una transformación más alta en la jerarquía que la anterior.

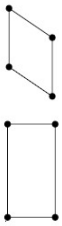
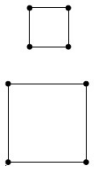
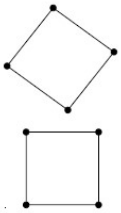
$$\boldsymbol{H} = \boldsymbol{H}_S\boldsymbol{H}_A\boldsymbol{H}_P = \begin{bmatrix} s\boldsymbol{R} & t \\ \boldsymbol{0}^T & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{V}^T & v \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{V}^T & v \end{bmatrix} \tag{2.35}$$

Con una matriz no singular $\boldsymbol{A}$ dada por $\boldsymbol{A} = s\boldsymbol{R}\boldsymbol{K} + \boldsymbol{t}\boldsymbol{V}^T$ y $\boldsymbol{K}$ una matriz triangular superior normalizada como el $\det(\boldsymbol{K}) = 1$. Esta descomposición es válida siempre que $v = 0$, y es única si s se elige positivo.

La tabla 1 resume los grupos de transformaciones y sus invariantes. Las transformaciones de más abajo en la tabla son especializaciones de las de arriba.

Tabla 2.1. Propiedades geométricas invariantes a las transformaciones planares que ocurren comúnmente.

Grupo	Matriz	Distorsión	Invariantes
Proyectiva 8 gdl	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrencia, colinealidad, orden de contacto: intersección (1 punto de contacto); Tangencia (2 puntos de contacto); Inflexiones (3 puntos de contacto con la línea); Discontinuidades tangentes y cúspides. Relación cruzada (relación de relación de longitudes).

Afín 6 gdl	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Paralelismo, relación de áreas, relación de longitudes en líneas colineales o paralelas (por ejemplo, puntos medios), combinaciones lineales de vectores (por ejemplo, centroides). La línea en el infinito.
Similitud 3 gdl	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Relación de longitudes y ángulos.
Euclidiano 3 gdl	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Relación de longitud y área

3 Cámara e imagen

En este capítulo se describirán conceptos de la formación de la imagen, el modelo y parámetros que describen la cámara y algunos métodos matemáticos de Visión por computadora.

3.1 Cámara

La cámara con modelo pinhole produce una imagen muy tenue ya que su potencia radiante es la luminancia de la escena en unidades de Wm^{-2} multiplicado por el tamaño del agujero de alfiler. La clave para obtener imágenes más brillantes es recoger la luz sobre un área más grande usando una lente o un espejo curvo. Las lentes convexas pueden formar una imagen como un agujero de alfiler, pero el diámetro más grande de la lente permite que pase más luz, lo que conduce a imágenes mucho más brillantes.

Los aspectos elementales de la formación de la imagen con una lente delgada se muestran en la figura 3.1. El eje z positivo es el eje óptico de la cámara. La coordenada z del objeto y su imagen están relacionadas por la ley del objetivo [7].

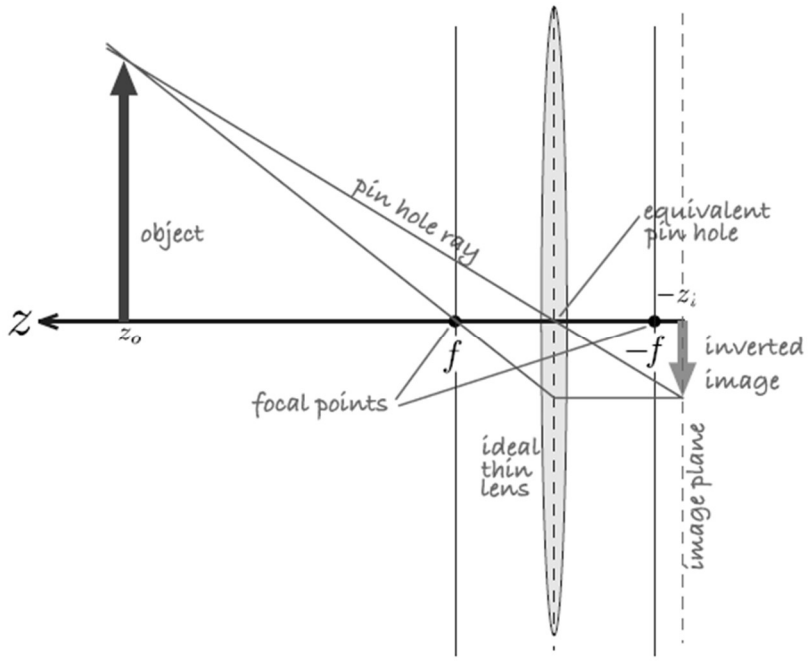


Figura 3.1. Geometría de formación de imagen para una lente convexa delgada que se muestra en la sección transversal bidimensional. Distancia de f en cada lado de la lente. Por convención, el eje óptico de la cámara es el eje z.

En la visión por computadora es común utilizar el modelo de imagen en perspectiva central mostrado en la figura 3.2. Los rayos convergen en el origen de los ejes x y y de la cámara {C} y una imagen no invertida se proyecta sobre el plano de la imagen situado en $z = f$. Usando triángulos similares podemos mostrar que un punto en las coordenadas mundiales $P = (X, Y, Z)$ es proyectado al plano de imagen $p = (x, y)$ por las ecuaciones siguientes.

$$x = f\frac{X}{Z} \ , y = f\frac{Y}{Z} \tag{3.1}$$

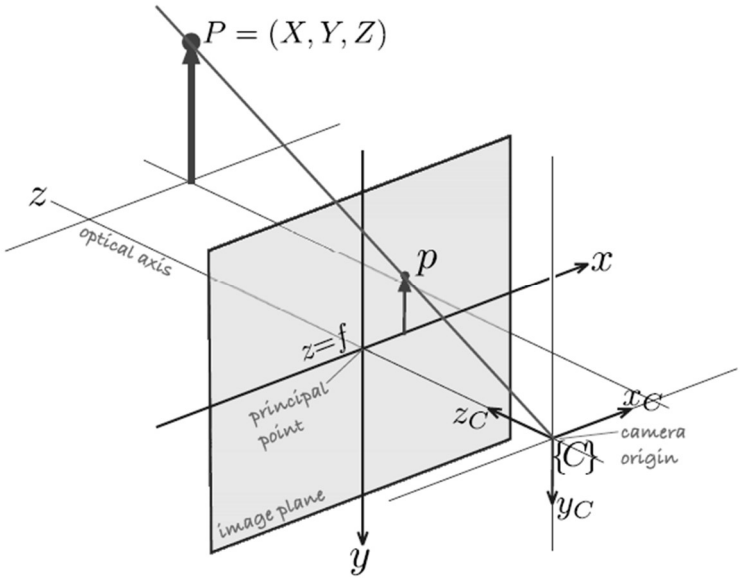


Figura 3.2. El modelo de proyección central. El plano de la imagen es f que se encuentra delante del origen de la cámara y sobre el cual se forma una imagen no invertida. El marco de coordenadas de la cámara es diestro con el eje z que define el centro del campo de visión

Las ecuaciones 3.1 son una transformación proyectiva, desde coordenadas del mundo al plano de la imagen y tiene las siguientes características:

1. Realiza un mapeo desde el espacio tridimensional al plano de la imagen bidimensional: $\mathbb{R}^3 \rightarrow \mathbb{R}^2$.
2. Las líneas rectas en el mundo se proyectan a líneas rectas en el plano de la imagen.
3. Las líneas paralelas en el mundo se proyectan a líneas que se cruzan en un punto en el infinito como se muestra en la figura 3.3. La excepción son las líneas en el plano paralelo al plano de la imagen que no convergen.

Figura 3.3 Las líneas paralelas en el mundo real convergen en el punto en el infinito en el plano bidimensional.

4. Las cónicas en el mundo se proyectan a cónicas en el plano de la imagen [7]. Por ejemplo, un círculo se proyecta como un círculo o una elipse como se muestra en la figura 3.4.

Figura 3.4. Los círculos se proyectan en elipses.

5. El mapeo no es uno a uno y no existe una única inversa. Es decir, dado (x, y) no podemos determinar de manera única (X, Y, Z). Todo lo que se puede decir es que el punto del mundo se encuentra en algún lugar a lo largo del rayo de proyección OP mostrado en la figura 3.2.
6. La transformación es proyectiva - no conserva la forma ya que los ángulos internos no se conservan.

Se puede escribir los puntos del plano de la imagen en coordenadas homogéneas en forma $\tilde{\boldsymbol{p}} = (X, Y, Z)$ donde

$$x = \frac{fX}{z}, y = \frac{fY}{z}, z = Z \tag{3.2}$$

O en forma compacta:

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{3.3}$$

Donde las coordenadas no homogéneas del plano de la imagen son:

$$x = \frac{X}{Z}, \quad y = \frac{Y}{Z} \tag{3.4}$$

A menudo se hace referencia a ellas como las coordenadas del plano de imagen. Para el caso en el que f = 1, las coordenadas se denominan coordenadas de plano de imagen normalizadas o canónicas. Si también escribimos las coordenadas mundiales en forma de coordenadas homogéneas tal que ${}^{C}\widetilde{\boldsymbol{P}} = (X, Y, Z, 1)^{T}$ entonces la trasformación proyectiva puede ser escrita en forma linear como:

$$\widetilde{\boldsymbol{p}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^{C}\widetilde{\boldsymbol{P}} \tag{3.5}$$

O

$$\widetilde{\boldsymbol{p}} = \boldsymbol{C}\, {}^{C}\widetilde{\boldsymbol{P}} \tag{3.6}$$

Donde $\boldsymbol{C}$ es una matriz de 3×4 conocida como la matriz de la cámara. Nótese que sea escrito ${}^{C}\widetilde{\boldsymbol{P}}$ para remarcar el hecho de que esta es la coordenada del punto respecto a sistema de coordenadas de la cámara $\{\mathrm{C}\}$. La tilde indica que son cantidades homogéneas. La tercera columna de $\boldsymbol{C}$ es un vector paralelo la je óptico de la cámara en el sistema de coordenadas mundiales. La matriz de la cámara puede ser factorizada de la siguiente manera.

$$\widetilde{\boldsymbol{p}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^{C}\widetilde{\boldsymbol{P}} \tag{3.7}$$

Donde la segunda matriz es la matriz de proyección.

3.1.1 Parámetros intrínsecos de la cámara

Se pueden asociar dos diferentes planos de imagen en una cámara: el primero es un plano normalizado a una unidad de distancia del pinole (distancia focal). Se define a este plano su propio sistema de coordenadas con un origen situado en el punto $\widetilde{\boldsymbol{C}}$ donde el eje óptico lo atraviesa (figura 3.5). La ecuación de proyección puede ser escrita en su sistema de coordenadas normalizadas de la siguiente manera:

$$\begin{cases} \tilde{u} = \frac{X}{Z} \\ \tilde{v} = \frac{Y}{Z} \end{cases} \Leftrightarrow \widetilde{\boldsymbol{p}} = \frac{1}{Z}(\boldsymbol{I}\ \boldsymbol{0}) \begin{pmatrix} \boldsymbol{P} \\ 1 \end{pmatrix} \tag{3.8}$$

Donde $\widetilde{\boldsymbol{p}} = (\tilde{u}, \tilde{v}, 1)^{T}$ es el vector de coordenadas homogéneas de la proyección de $\widetilde{\boldsymbol{p}}$ del puto $\boldsymbol{P}$ dentro del plano de la imagen normalizada.

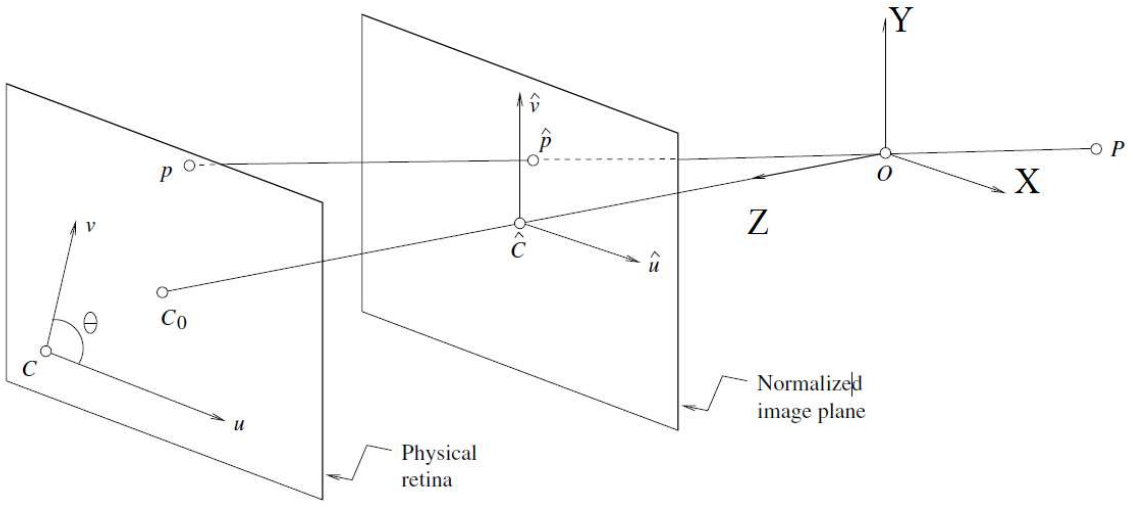


Figura 3.5. Sistema de coordenadas de la imagen física y la imagen normalizada.

La retina física de la cámara es en general diferente de la figura 16, está localizada a una distancia $f \neq 1$ desde el pinhole, y las coordenadas de la imagen (u, v) del punto p de la imagen están usualmente expresadas en unidades de pixel (en lugar de metros). También los pixeles son normalmente rectangulares en lugar de cuadrados, por lo tanto, la cámara tiene dos parámetros de escalamiento adicionales *k* y *l*, donde

$$\begin{cases} u = kf\frac{X}{Z}\,, \\ v = lf\frac{Y}{Z}\,. \end{cases} \tag{3.9}$$

f es una distancia expresada en metros, y un pixel tendrá dimensiones de $\frac{1}{k} \times \frac{1}{l}$, donde k y l están expresados en $pixeles \times m^{-1}$. Los parámetros *k*, *l* y *f* no son independientes y pueden ser reemplazados por las variables $\alpha = kf$ y $\beta = lf$ expresadas en unidades de píxeles.

En general, el origen real del sistema de coordenadas de la cámara está en una esquina C de la retina (por ejemplo, en el caso representado en la figura 16, la esquina inferior izquierda o, a veces, la esquina superior izquierda cuando las coordenadas de la imagen son los índices de fila y columna de un píxel) y no en su centro, y el centro de la matriz CCD usualmente no coincide con el punto principal C_0. Esto suma dos parámetros u_0 y v_0 que definen la posición (en unidades de píxeles) de C_0 en el sistema de coordenadas de la retina. Así, la ecuación 3.9 se sustituye por

$$\begin{cases} u = \alpha\frac{X}{Z} + u_0\,, \\ v = \beta\frac{Y}{Z} + v_0\,. \end{cases} \tag{3.10}$$

Por último, el sistema de coordenadas de la cámara también puede estar sesgado, debido a algún error de fabricación, por lo que el ángulo θ entre los dos ejes de imagen no es igual a 90 grados. Por lo tanto, las ecuaciones nos quedan de la siguiente manera:

$$\begin{cases} u = \alpha \frac{X}{Z} - \alpha \, cot\theta \frac{Y}{Z} + u_0 \, , \\ v = \frac{\beta}{sen\,\theta} \frac{Y}{Z} + v_0 \, . \end{cases} \tag{3.11}$$

Utilizando las ecuaciones 3.8 y 3.11 podemos escribir el cambio de coordenadas entre la imagen física y la imagen normalizada como una transformación afín:

$$\boldsymbol{p} = \boldsymbol{K}\tilde{\boldsymbol{p}}, \qquad \text{donde} \qquad \boldsymbol{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \qquad \text{y} \qquad \boldsymbol{K} = \begin{bmatrix} \alpha & -\alpha \, cot\theta & u_0 \\ 0 & \frac{\beta}{sen\,\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Poniendo todo esto junto, se obtiene:

$$\boldsymbol{p} = \frac{1}{Z}\boldsymbol{M}\boldsymbol{P}, \qquad \text{donde} \qquad M = [\boldsymbol{K}\ 0]. \tag{3.12}$$

$\boldsymbol{p}$ denota el vector de coordenadas homogéneas en el sistema de coordenadas de la cámara. Con las coordenadas homogéneas se puede representar el mapeo de proyectivo por la matriz $\boldsymbol{M}$ de 3×4.

3.1.2 Parámetros extrínsecos

Cuando diferentes sistemas de coordenadas son considerados al mismo tiempo, es conveniente denotar por ${}^{F}P$ la coordenada vector del punto **P** en el sistema de coordenadas $\boldsymbol{F}$ [10].

$${}^{F}P = {}^{F}\overrightarrow{OP} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \Leftrightarrow \quad \overrightarrow{OP} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$$

También vamos a decir que dos sistemas de coordenadas están separados por pura traslación cuando $\overrightarrow{O_B P} = \overrightarrow{O_B O_A} + \overrightarrow{O_A P}$

$$\text{Así, } {}^{B}P = {}^{A}P + {}^{B}O_A \, .$$

En cambio, cuando las coordenadas solo están puramente rotadas se aplicará:

$${}^{B}_{A}R = \begin{bmatrix} \cos\theta & sen\,\theta & 0 \\ -sen\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Así para representar la rotación de un punto se escribirá ${}^{B}P = {}^{B}_{R}R\,{}^{A}P$. Por lo tanto cunado un punto es trasladado y rotado se tendría

$${}^{B}P = {}^{B}_{R}R\,{}^{A}P + {}^{B}O_A$$

Y en particular se escribirán el cambio de coordenadas con formato en coordenadas homogéneas como:

$$\begin{pmatrix} {}^{B}P \\ 1 \end{pmatrix} = {}^{B}_{A}H \begin{pmatrix} {}^{A}P \\ 1 \end{pmatrix}, \qquad \text{donde} \qquad {}^{B}_{A}H = \begin{bmatrix} {}^{B}_{R}R & {}^{B}O_{A} \\ 0^{T} & 1 \end{bmatrix}.$$

Se considerará el caso cuando la cámara con sistema de coordenadas (**C**) es distinta del sistema de coordenadas mundiales (**W**). Nótese que:

$$ {}^{C}P = \begin{pmatrix} {}^{C}_{W}R & {}^{C}O_{W} \end{pmatrix} \begin{pmatrix} {}^{W}P \\ 1 \end{pmatrix} \tag{3.13}$$

Y sustituyendo en la ecuación 3.12.

$$\boldsymbol{p} = \frac{1}{Z}\boldsymbol{MP}, \qquad \text{donde} \qquad \boldsymbol{M} = \boldsymbol{K}(\boldsymbol{R}\ \boldsymbol{t}), \tag{3.14}$$

Donde $\boldsymbol{R} = {}^{C}_{W}R$ es una matriz de rotación, $\boldsymbol{t} = {}^{C}O_{W}$ es un vector de traslación, y $\boldsymbol{P}$ denota el vector en coordenadas homogéneas de $\boldsymbol{P}$ en el sistema de coordenadas $\boldsymbol{W}$.

En general se escribe la ecuación de proyección como $Z\boldsymbol{p} = \boldsymbol{MP}$, o $Z\boldsymbol{p} = \boldsymbol{MP}$ con la convención de que un vector homogéneo solo esta escalado, y el sistema de coordenadas actual del punto de la imagen $\boldsymbol{p}$ está definido como $\frac{u}{w}$ y $\frac{v}{w}$ si $\boldsymbol{p} = (u, v, w)^{T}$. En este contexto, la matriz $\boldsymbol{M}$ esta igual definida a escala, con 11 parámetros libres. Nótese que hay 5 parámetros intrínsecos (α, β, u_0, v_0 y θ) y seis parámetros extrínsecos (los tres angulos que definen $\boldsymbol{R}$ y tres coordenadas de $\boldsymbol{t}$), los cuales coinciden con el número de coeficientes independientes de $\boldsymbol{M}$.

La matriz $\boldsymbol{M}$ puede ser escrita explícitamente como una función de los parámetros intrínsecos y extrínsecos de la cámara. De la siguiente forma:

$$\boldsymbol{M} = \begin{bmatrix} \alpha\, \boldsymbol{r}_1^{T} - \alpha\, cot\theta\ \ \boldsymbol{r}_2^{T} + u_0 \boldsymbol{r}_3^{T} & \alpha\, t_X - \alpha\, cot\theta\ \ t_Y + u_0 t_Z \\ \frac{\beta}{sen\,\theta}\boldsymbol{r}_2^{T} + v_0\, \boldsymbol{r}_3^{T} & \frac{\beta}{sen\,\theta} t_Y + v_0\, t_Z \\ \boldsymbol{r}_3^{T} & t_Z \end{bmatrix} \tag{3.15}$$

Donde $\boldsymbol{r}_1^{T}, \boldsymbol{r}_2^{T}$ y $\boldsymbol{r}_3^{T}$ denotan las tres columnas de la matriz $\boldsymbol{R}$ y t_X , t_Y y t_Z son las coordenadas del vector $\boldsymbol{t}$ en el sistema de coordenadas de la cámara. Si $\boldsymbol{R}$ representa el producto de tres rotaciones elementales, el vector $\boldsymbol{r}_i$ (i=1,2,3) puede ser escrito explícitamente en términos de sus correspondientes ángulos.

Cabe señalar que la matriz $\boldsymbol{M}$ determina el vector coordenadas del centro óptico de la cámara en el sistema de coordenadas del mundiales, y, por lo tanto $\boldsymbol{M}\begin{pmatrix} \boldsymbol{C} \\ 1 \end{pmatrix} = 0$.

3.2 Definición de la imagen

Una imagen puede definirse como una función bidimensional $f(x, y)$ donde x y y son coordenadas en el plano y la amplitud f es llamada intensidad o nivel de gris en ese punto [11]. Cuando (x, y) y f son todos finitos (cantidades discretas) llamamos a la función como imagen digital. Es decir, una imagen digital estará compuesta por un número finito de

elementos llamados píxeles, cada uno de los cuales con un valor y una posición particular como se muestra en la figura 3.6.

El valor es relativo a alguna propiedad del punto que representa, como por ejemplo su brillo o su matiz.

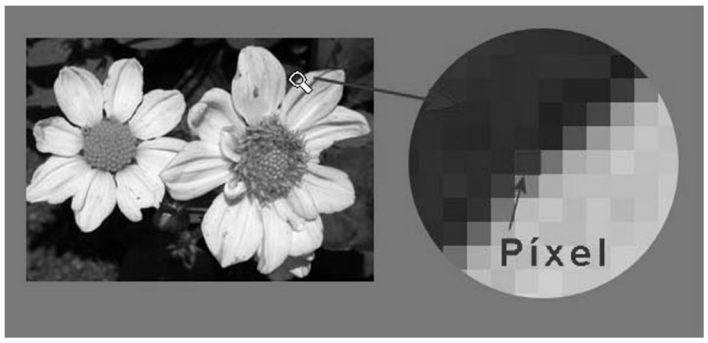


Figura 3.6. Composición de una imagen por pixeles.

3.2.1 Pixel

El término **pixel** (abreviación de Picture element o elemento de imagen), se trata de la unidad mínima de información de una imagen, la cual aparece como un punto en la pantalla o en una impresora [11]. En realidad, cada píxel se compone de tres registros de color, mediante la combinación de cierta cantidad de rojo, verde y azul, el píxel adopta un color particular.
Las imágenes bidimensionales son el resultado de una proyección en perspectiva de escenas tridimensionales. Cuando se obtiene una imagen bidimensional del mundo tridimensional desaparece gran cantidad de información.

3.2.2 Relaciones entre pixeles

Un pixel p con coordenadas (x, y) tiene cuatro vecinos, dos horizontales y dos verticales, cuyas coordenadas son: $(x+1, y)$, $(x-1, y)$, $(x, y-1)$, $(x, y+1)$. A este conjunto de píxeles se llama vecindad 4 de p y se denota por N4(p), ver la figura 3.7. Nótese que para cada uno de estos pixeles hay una distancia de 1 (uno) desde p y que en los bordes de la imagen algunos de estos pixeles quedarán fuera de la imagen.

	(x-1, y)	
(x, y-1)	(x, y)	(x, y+1)
	(x+1, y)	

(x-1, y-1)		(x-1, y+1)
	(x, y)	
(x+1, y-1)		(x+1, y+1)

Figura 3.7. Vecindad $N_4(p)$ y Vecindad $N_D(p)$.

Existen también 4 vecinos diagonales de p con coordenadas: $(x+1, y+1)$, $(x+1, y-1)$, $(x-1, y-1)$, $(x-1, y+1)$ y se les denota por $N_D(p)$, figura 3.7. Conjuntamente, $N_4(p)$ y $N_D(p)$ forman la vecindad 8 de p denotada por $N_8(p)$.

3.2.3 Conectividad

La conectividad es un concepto importante utilizado para establecer los límites de objetos en regiones dentro de una imagen digital. Para determinar si dos píxeles están conectados se determina si son adyacentes en algún sentido, sea $N_D(p)$ o $N_4(p)$ por ejemplo, y si sus niveles de gris satisfacen algún criterio de similitud (si son iguales o parecidos). Por ejemplo, en una imagen binaria con valores de 1 y 0, dos pixeles pueden ser vecinos $N_4(p)$, pero se dice que están conectados sólo cuando tienen el mismo valor.
En la figura 3.8 se observa la conectividad de pixeles en una imagen binaria. El pixel 6 está conectado con el 2 y 8. El pixel 3 está conectado con el 5.

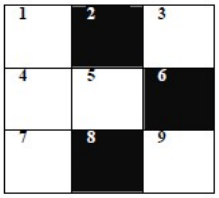


Figura 3.8. Conectividad de Pixeles.

3.2.4 Distancia

La distancia o transformada de distancia proporciona una medición de la separación existente entre dos puntos dentro de una imagen [12]. Dados dos píxeles p y q con coordenadas (x, y) y (s, t), respectivamente, se puede definir una función de distancia D si se cumple:

$$D(p,q) \geq 0 \tag{3.16}$$
$$D(p,q) = 0, si\ p = q \tag{3.17}$$
$$D(p,q) = D(q,p) \tag{3.18}$$

Las funciones de distancia comúnmente usadas son: distancia euclidiana y distancia tablero de ajedrez.
Distancia euclidiana entre p y q:

$$\mathrm{D_E(p,q)} = \sqrt{(\mathrm{x-s})^2 + (\mathrm{y-t})^2} \tag{3.19}$$

En la figura de abajo se muestra la distancia euclidiana para una imagen de 5 por 5.

$$\begin{array}{ccccc} \sqrt{8} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{8} \\ \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} \\ 2 & 1 & 0 & 1 & 2 \\ \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} \\ \sqrt{8} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{8} \end{array}$$

Distancia tablero de ajedrez: en donde se observa que los 4-vecinos están a una distancia unitaria del píxel central; si se desea que los 8-vecinos estén a la misma distancia se toma:

$$D(p,q) = Max(x-s\,, y-t) \tag{3.20}$$

En la figura de abajo se muestra la distancia tablero de ajedrez para una imagen de 5 por 5.

$$\begin{matrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{matrix}$$

3.2.5 Ruido en imágenes

Todas las imágenes tienen cierta cantidad de ruido, la cual se puede deber a la cámara, escáner o al medio de transmisión de la señal. Generalmente el ruido se manifiesta como píxeles aislados que toman un nivel de gris diferente al de sus vecinos. Los algoritmos de filtrado permiten eliminar o disminuir este ruido.

Ruido dependiente de la señal: Este tipo de ruido viene producido generalmente en los propios sensores fotoeléctricos (CCD por ejemplo) o también debido al grano de la película de una fotografía o cinta de vídeo. Es un ruido dependiente de la señal. La función generadora de este tipo de ruido viene determinada por el producto de la imagen original con una función ruido. Esta función ruido intenta simular el ruido dependiente de la señal mediante una función normal-logarítmica y el ruido térmico, que se produce en los sensores fotoeléctricos, mediante la suma de una función de ruido Gaussiano. Resumiendo, en la literatura especializada se define la siguiente transformación:

$$g(i,j) = c2 \cdot (h(i,j) ** f(i,j))\gamma + (c2 \cdot (h(i,j) ** f(i,j))\gamma 1/2 \cdot n(i,j) + nt(i,j) \tag{3.21}$$

siendo $f(i,j)$ la imagen original, $n(i,j)$ el ruido dependiente de la señal, $nt(i,j)$ el ruido térmico y h(i,j) la función de transferencia del sistema óptico. El símbolo ** denota una convolución de dos dimensiones.

Ruido salt-pepper o ruido impulsivo: Está formado por puntos blancos o/y negros que se distribuyen de manera aleatoria por la imagen, este tipo de ruido es independiente depende de la señal. Es un ruido que aparece muchas veces producido por interferencias atmosféricas, o por acciones hechas por el humano (motores de coches, etc.). Se puede modelar de la siguiente manera:

$$g(i,j) = \begin{cases} z(i,j) & con\ probabilidad\ p \\ f(i,j) & con\ probabildad\ 1-p \end{cases} \tag{3.22}$$

siendo $z(i,j)$ un valor fijo (0 o 255) o variable, y $f(i,j)$ la imagen original que va a recibir el ruido.

3.2.6 Imágenes binarias

Las imágenes digitales generalmente están compuestas por un amplio rango de valores de intensidad. A estas imágenes se las ha denominado imágenes de nivel de gris. Aunque el rango de niveles de gris utilizado para representar la imagen tradicionalmente viene siendo de 256 valores (8 bits por píxel), este es variable y depende de la aplicación. La tendencia natural, debido al aumento de la potencia computacional y de la calidad de los sensores es a aumentar este rango para dotar de mayor fidelidad a la imagen, y no parece que

esté muy lejos el día en que los 16 bits por píxel (65536 valores) se conviertan en la opción estándar. No obstante, la mayor parte de las aplicaciones no precisan de tantos niveles de gris sino más bien lo contrario; pueden utilizar pocos niveles debido a que trabajan con escenas de muy alto contraste. Tanto es así que en muchas aplicaciones industriales se llega al extremo de utilizar únicamente dos niveles de gris. De esta forma, se obtiene lo que se conoce como imagen binaria.

Trabajar con imágenes binarias resulta muy interesante por dos motivos:

- En primer lugar, porque se reduce al mínimo los datos necesarios para representar la imagen y ello permite un máximo aprovechamiento de la potencia computacional.
- En segundo lugar, porque las propiedades geométricas y topológicas de los objetos presentes en la imagen, en las que se basan un gran número de aplicaciones industriales, puede obtenerse rápida y fácilmente a partir de las imágenes binarias.

Desde el punto de vista computacional, las imágenes binarias se procesan mucho más rápidamente. Por este motivo, las primeras aplicaciones de visión artificial en línea emplearon este tipo de imágenes casi exclusivamente. Incluso hoy en día, que se dispone de potentes procesadores para hacer frente a imágenes en niveles de gris, siguen siendo aún más numerosas las aplicaciones que trabajan sobre imágenes binarias por su simplicidad y robustez.

Las imágenes binarias siempre se obtienen a partir de imágenes de niveles de gris. En la actualidad no existen cámaras comerciales que proporcionen imágenes binarias. El proceso de conversión de una imagen de nivel de gris a una imagen formada solo por dos valores o etiquetas (0 para el negro y 1 para el blanco) se conoce como binarización (figura 3.9).

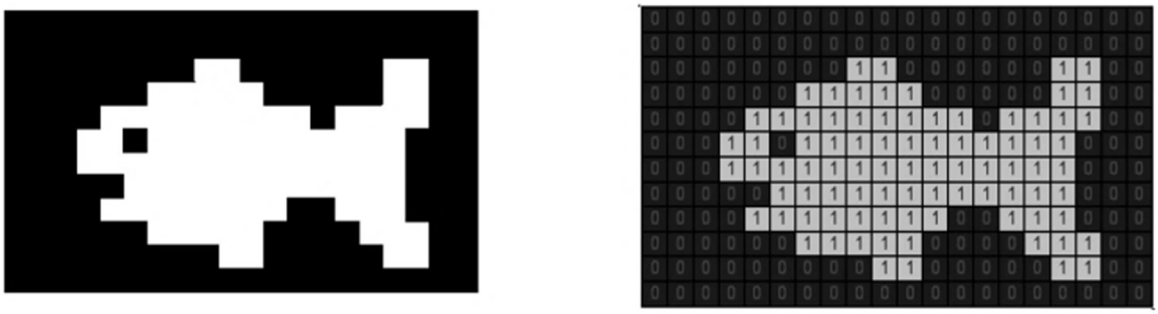


Figura 3.9. Las imágenes binarias solo están representadas por dos niveles de gris: el negro y el blanco. Cada píxel está etiquetado con 0 o 1 respectivamente.

En general, la binarización consiste en, a partir de un nivel de gris predeterminado denominado umbral de binarización, etiquetar como 0 todos los pixeles con niveles de gris inferior a ese umbral y como 1 los que tengan un nivel de gris igual o superior.

Si la imagen está bien contrastada, la pérdida de información es mínima. Muchas veces esta sencilla operación permite separar los objetos del fondo. Hay que insistir en que, para ello, es fundamental que la imagen de nivel de gris tenga un alto contraste, es decir, que los dos

grupos de pixeles correspondiente a objetos y al fondo, posean niveles de gris bien diferenciados.

3.2.7 Dilatación y erosión de imágenes

La erosión y la dilación son las operaciones morfológicas más usadas en procesamiento de imágenes.

La función de dilación es tomar cada píxel del objeto (con valor “1”) y poner al valor “1” todos aquellos píxeles pertenecientes al fondo (background) que tienen una conectividad C (C=4, C=8, etc) con el píxel del objeto. En pocas palabras, poner a “1” los píxeles del fondo vecinos a los píxeles del objeto.

En nuestro caso la dilatación se usa para agrandar los puntos encontrados y eliminar partes de los puntos que estén separadas del punto principal, formando un solo punto, así evitando errores en la posterior cuantificación de puntos.

La erosión es cada pixel del objeto que tiene una conectividad C con los pixeles del fondo y ponerlo al valor “0”. En otras palabras, poner a “0” los pixeles del objeto vecinos a los píxeles del fondo. Esto se hace después de la dilación para regresar a su tamaño normal los puntos y de esa forma tener solo el número correcto de puntos. Como se ve en la figura 3.10.

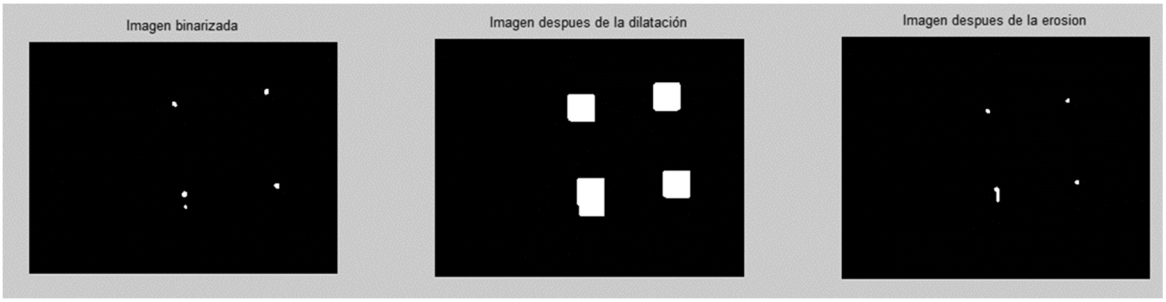


Figura 3.10. Izquierda: Imagen binaria original. Centro: Puntos dilatados. Derecha: Puntos erosionados.

3.2.8 Etiquetado

Es una técnica que asigna una etiqueta a cada componente conexo para obtener al final una región [13]. Básicamente el algoritmo recorre una imagen binaria de izquierda a derecha y de arriba hacia abajo en busca de píxeles con valor “1” que se encuentren conectados entre sí en un rango que puede ser C=8, C=4 etc. y les asigna una etiqueta construyendo de esta manera regiones (figuras 3.11 y 3.12). Esta técnica, no es tan susceptible a tener errores si los caracteres están rodeados o de líneas que cubran o rodeen los caracteres. Además, no requiere que la imagen se encuentre alineada con respecto al eje para ser efectiva lo cual es de gran utilidad debido a que no siempre se pueden tener correctamente alienado el dispositivo móvil con la imagen como seria con un escáner de sobremesa o plano. Al final del etiquetado puede suceder que en una misma región se etiquete con diferentes números (figura 3.12) para lo cual se tiene que utilizar una técnica de eliminación de colisiones [13] y el resultado se puede ver en la figura 3.13.

3.2.9 Centroides

Ya que se tienen los puntos etiquetados es necesario encontrar sus coordenadas en la imagen, para esto se determinan los centroides de los puntos u objetos en la imagen.

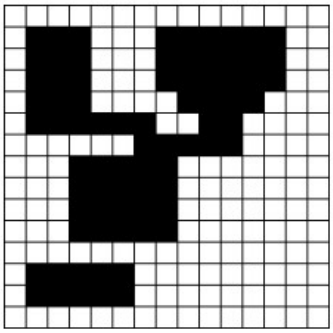


Figura 3.11. Imagen binaria.

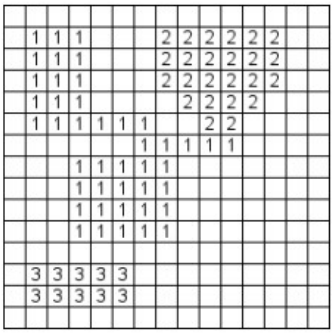


Figura 3.12. Etiquetado de la imagen binaria

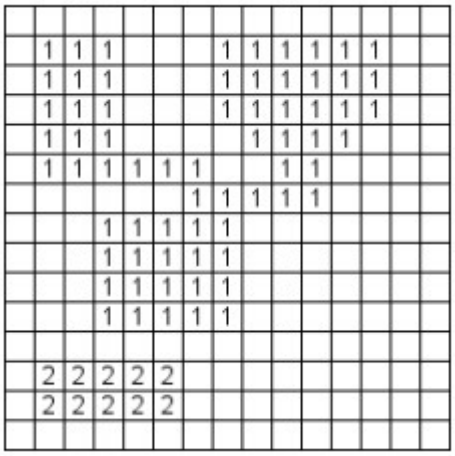


Figura 3.13. Etiquetado después de la eliminación de colisiones.

El centroide o centro de masa de un objeto es el punto en donde por su geometría se encuentra concentrada la masa del objeto [13]. El centroide $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y})$ de un objeto binario se calcula como el punto medio aritmético de las coordenadas en la dirección y y x, tal que

$$\bar{x} = \frac{1}{Area(O)} \sum_{(x,y)\epsilon O} x \qquad \bar{y} = \frac{1}{Area(O)} \sum_{(x,y)\epsilon O} y \tag{3.23}$$

4 Sistema experimental

En este capítulo se explicará el método que se utilizó para calcular la orientación y la posición a partir de la geometría proyectiva; primero se explicara la parte teórica del sistema y posteriormente se explicara tanto el hardware como el software que se implementó.

4.1 Desarrollo teórico del sistema.

En esta sección se va a calcular la posición y orientación relativas (matriz de rotación $\boldsymbol{R}$ y vector de traslación $\boldsymbol{t}$) entre el plano que se encuentra en el mundo físico y el plano de la cámara. Sabiendo las distancias relativas entre cuatro puntos que pertenecen al plano físico se calculara la matriz $\boldsymbol{P}$ que nos relacionara los cuatro puntos en 3D del plano físico con cuatro puntos que estarán en la imagen 2D.

4.1.1 Cálculo de la matriz H de transformación 2D a 2D

Para el cálculo de la matriz de transformación proyectiva H (o modelo de la cámara) que nos relaciona el plano de la imagen con el plano que está en el mundo físico, es decir

$$\boldsymbol{x}_{uv} = \boldsymbol{H}\boldsymbol{X}_{xy} \tag{4.1}$$

Donde $\boldsymbol{x}$ representa un punto en las coordenadas del plano de la imagen y $\boldsymbol{X}$ representa el mismo punto, pero en coordenadas del plano en el mundo físico. La ecuación 4.1 en su forma explícita sería:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Donde u y v son las coordenadas homogéneas del punto $\boldsymbol{x}$ en el plano de la imagen y $X\ Y$ son las coordenadas homogéneas del punto $\boldsymbol{X}$ en el plano en el mundo físico.

Primero se escoge punto arbitrario y se desarrollan las ecuaciones de 4.1 para u_1 y v_1, lo cual nos da las siguientes ecuaciones (4.2):

$$\begin{aligned} h_{11}X_1 + h_{12}Y_2 + h_{13} &= u_1 \\ h_{21}X_1 + h_{22}Y_2 + h_{23} &= v_1 \\ h_{31}X_1 + h_{32}Y_2 + h_{33} &= 1 \end{aligned} \tag{4.2}$$

Ya que $h_{31}X_1 + h_{32}Y_2 + h_{33} = 1$, podemos multiplicar los dos lados de las dos primeras ecuaciones por uno, dándonos como resultado las siguientes ecuaciones:

$$\begin{aligned} h_{11}X_1 + h_{12}Y_2 + h_{13} &= u_1h_{31}X_1 + u_1h_{32}Y_2 + u_1h_{33} \\ h_{21}X_1 + h_{22}Y_2 + h_{23} &= v_1h_{31}X_1 + v_1h_{32}Y_2 + v_1h_{33} \end{aligned} \tag{4.3}$$

Reacomodando los términos de las ecuaciones 4.3, de tal forma que los parámetros h_{ij} queden del lado izquierdo, y solo para fines de aclaración, se colocaran todos los parámetros h_{ij} en las dos ecuaciones, quedando las siguientes ecuaciones:

$$h_{11}X_1 + h_{12}Y_2 + h_{13} + h_{21}0 + h_{22}0 + h_{23}0 - u_1h_{31}X_1 - u_1h_{32}Y_2 - u_1h_{33} = 0$$
$$h_{11}0 + h_{12}0 + h_{13}0 + h_{21}X_1 + h_{22}Y_2 + h_{23} - v_1h_{31}X_1 - v_1h_{32}Y_2 - v_1h_{33} = 0 \quad (4.4)$$

Haciendo este procedimiento para cuatro puntos se obtienen 8 ecuaciones de las cuales se conocen los valores de las coordenadas en el plano físico y los valores en el plano de la imagen y resolviendo el sistema (4.5) para los paramentos h_{ij} se puede calcular los valores de la matriz de proyección.

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -u_2 2X_2 & -u_2Y_2 & -u_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -v_2X_2 & -v_2Y_2 & -v_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -u_3 2X_3 & -u_3Y_3 & -u_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -v_3X_3 & -v_3Y_3 & -v_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -u_4X_4 & -u_4Y_4 & -u_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -v_4X_4 & -v_4Y_4 & -v_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \quad (4.5)$$

4.1.2 Cálculo de los valores extrínsecos de la cámara

Una vez obtenida la matriz de transformación $\boldsymbol{H}$, el paso siguiente es obtener la matriz de rotación y vector de traslación (valores extrínsecos) para obtener los valores de la posición y rotación del plano [8].

Se sabe que:

$$\boldsymbol{K^{-1}H} = \lambda \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

Que es igual a:

$$\boldsymbol{K^{-1}H} = \lambda[\boldsymbol{R}\ \boldsymbol{t}] \quad (4.6)$$

donde $\boldsymbol{K}$ es la matriz de parámetros intrínsecos de la cámara (esta se obtiene cuando se calibra la cámara), $\boldsymbol{R}$ es la matriz de rotación, $\boldsymbol{t}$ es el vector de traslación y λ es una constante [5]. Si agrupamos los elementos de la matriz como vectores la ecuación 4.6 se puede expresar como.

$$\boldsymbol{K^{-1}H} = \lambda[\ \boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{t}\ ] \quad (4.7)$$

Donde $\boldsymbol{r}_1$ y $\boldsymbol{r}_2$ son vectores en $\mathbb{R}^3$ y son los vectores ortonormales del plano físico y $\boldsymbol{t}$ es el vector de traslación en $\mathbb{R}^3$. Sabiendo esto tenemos que:

$$\boldsymbol{K^{-1}}[\boldsymbol{h}_1\ \boldsymbol{h}_2\ \boldsymbol{h}_3\ ] = [\lambda \boldsymbol{r}_1\ \lambda \boldsymbol{r}_2\ \lambda \boldsymbol{t}] \quad (4.8)$$

Donde de igual forma se reagrupa la matriz $\boldsymbol{H}$ en vectores columna en $\mathbb{R}^3$. Siendo $h_1 = (h_{11}\ h_{21}\ h_{31})^T$, $h_2 = (h_{12}\ h_{22}\ h_{32})^T$ y $h_3 = (h_{13}\ h_{23}\ h_{33})^T$.

Igualando los vectores se tiene que $\boldsymbol{K^{-1}h_1 = \lambda r_1}$ y $\boldsymbol{K^{-1}h_2 = \lambda r_2}$. Si se saca la norma de los vectores se tiene que $\|r_1\| = \|r_2\| = 1$ y por lo tanto se puede escribir:

$$\|k^{-1}h_1\| = \|k^{-1}h_2\| = \lambda \tag{4.9}$$

Utilizando la ecuación 4.9 para encontrar el valor de λ nos queda expresión podemos encontrar el valor de la matriz de valores extrínsecos (matriz de proyección).

$$\boldsymbol{[R\ t] = \frac{K^{-1}H}{\lambda}} \tag{4.10}$$

4.1.3 Cálculo de la matriz de proyección de 3D a 2D

Para realizar el cálculo de esta matriz y ya que $\boldsymbol{r_1}$y $\boldsymbol{r_2}$ son vectores ortogonales al plano que se encuentra en el mundo físico, sólo se necesita obtener el vector perpendicular a estos. Utilizando el producto vectorial se tiene que:

$$\boldsymbol{r_3 = r_1 \times r_2} \tag{4.11}$$

Utilizando esta información se puede escribir la matriz de proyección de la cámara que nos relaciona los puntos en el espacio 3D con el espacio 2D y por lo tanto nos queda:

$$\boldsymbol{P = [r_1\ r_2\ r_3\ t]} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{4.12}$$

Esta matriz contiene toda la información de las rotaciones y traslaciones del plano físico con respecto al plano de la cámara.

Por último, la matriz $\boldsymbol{P}$ se multiplica por la matriz de parámetros intrínsecos de la cámara; esto nos dará como resultado una matriz de proyección de la cámara de la forma:

$$\boldsymbol{P_K = KP} \tag{13}$$

4.1.4 Recuperación de los datos del plano físico

Una vez obtenida la matriz $\boldsymbol{P_K}$ de la ecuación (12) es necesario obtener los datos de traslación y rotación del plano físico.

Los valores de la traslación se obtienen directamente del vector $\boldsymbol{t}$ esto quiere decir que se cumple que:

$$\boldsymbol{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{4.14}$$

Por otro lado, para recuperar los valores de rotación del plano físico se utilizan las matrices de rotación euclidiana tal que una rotación general está formada por tres matrices de rotación con respecto de los ejes x,y,z:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & sen(\theta_x) \\ 0 & -sen(\theta_x) & \cos(\theta_x) \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & -sen(\theta_y) \\ 0 & 1 & 0 \\ sen(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & sen(\theta_Z) & 0 \\ -sen(\theta_Z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\theta_z)\cos(\theta_y) & \cos(\theta_y)\,sen(\theta_Z) & -sen(\theta_y) \\ cos(\theta_z)sen(\theta_y)sen(\theta_x) - co\ (\theta_x)sen(\theta_z) & sen(\theta_z)sen(\theta_y)sen(\theta_x) + cos(\theta_z)cos(\theta_x) & cos(\theta_y)sen(\theta_x) \\ sen(\theta_z)sen(\theta_x) + cos(\theta_z)cos(\theta_x)sen(\theta_y) & cos(\theta_x)sen(\theta_z)sen(\theta_y) - cos(\theta_z)sen(\theta_x) & cos(\theta_y)cos(\theta_x) \end{bmatrix} \quad (4.15)$$

Donde los vectores $\boldsymbol{r}_i$ con $i = 1,2,3$ de la matriz de rotación (4.15) quedan definidas de la siguiente manera.

$$r_1 = \begin{pmatrix} \cos(\theta_z)\cos(\theta_y) \\ cos(\theta_z)sen(\theta_y)sen(\theta_x) - cos(\theta_x)sen(\theta_z) \\ sen(\theta_z)sen(\theta_x) + cos(\theta_z)cos(\theta_x)sen(\theta_y) \end{pmatrix} \quad (4.16)$$

$$r_2 = \begin{pmatrix} \cos(\theta_y)\,sen(\theta_Z) \\ sen(\theta_z)sen(\theta_y)sen(\theta_x) + co\ (\theta_z)cos(\theta_x) \\ -cos(\theta_z)sen(\theta_x) + cos(\theta_x)sen(\theta_z)sen(\theta_y) \end{pmatrix} \quad (4.17)$$

$$r_3 = \begin{pmatrix} -sen(\theta_y) \\ cos(\theta_y)sen(\theta_x) \\ cos(\theta_y)cos(\theta_x) \end{pmatrix} \quad (4.18)$$

Siendo θ_x, θ_y y θ_z los ángulos de rotación del plano físico con respecto de los ejes x y y z respectivamente.

Utilizando álgebra para despejar los ángulos de rotación, tenemos que, de la ecuación (4.18)

$$\theta_y = -sen^{-1}(p_{13})\ ,\ \theta_x = \text{sen}^{-1}\left(\frac{p_{33}}{\cos(\theta_y)}\right),\ \theta_z = \cos^{-1}\left(\frac{p_{11}}{\cos(\theta_y)}\right) \quad (4.19)$$

Donde p_{ij} son los elementos de la matriz $\boldsymbol{P}$. Estas ecuaciones (4.19) nos regresan la información de la orientación del plano físico con respecto del plano de la cámara.

4.2 Implementación del Hardware

Lo anteriormente descrito se ha llevado a su implementación con el fin de comprobar de forma experimental la obtención de la información. El diagrama a bloques en la figura 4.1 muestra como está constituido el sistema completo.

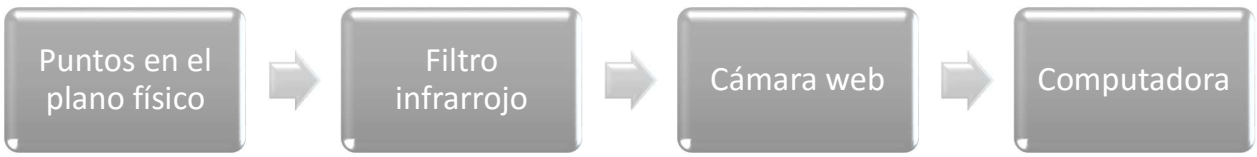


Figura 4.1. Diagrama a bloques del sistema.

Para generar los puntos en el espacio 3D (plano físico) se construyó un dispositivo experimental, con base en un marco de material ABS el cual tiene 4 led´s infrarrojos ubicados en las esquinas del cuadrado y estos leds están separados por una distancia de 10 cm como se muestra en la figura 4.2. De esta manera generamos cuatro puntos sobre un plano, el cual permitirá adquirir una imagen que pueda ser rotada-desplazada en los ejes y detectar las variables buscadas.

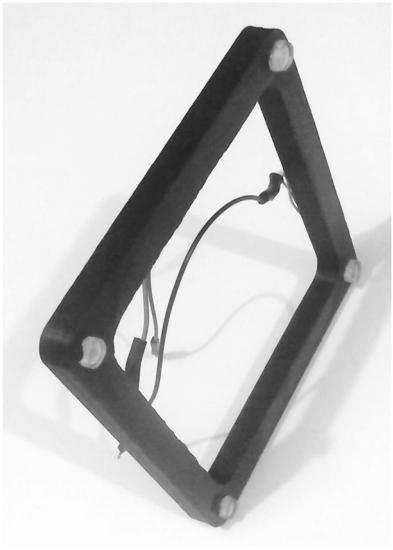


Figura 4.2. Disposición de los led's infrarrojos en un marco de plástico

El siguiente elemento que se utiliza es un filtro de luz infrarroja (figura 4.3) para discriminar toda la luz que llega del ambiente a la cámara web y únicamente dejar pasar la luz infrarroja proveniente de los 4 puntos (led's infrarrojos). Se escogió utilizar el filtro ya que con esto se evita hacer demasiado procesamiento de imagen. Con esto solo es necesario encontrar los valores de los centroides lo cual puede llegar a hacerse con microcontroladores.

Figura 4.3. Filtro de luz infrarroja.

En las pruebas para detección de los cuatro puntos con la cámara figura 4.4 se puede ver el dispositivo experimental con los 4 led's infrarrojos tomados con la cámara sin el filtro infrarrojo, en contraste con la figura 4.5 en la que se puede observar el mismo dispositivo experimental con los led's pero esta vez con el filtro infrarrojo.

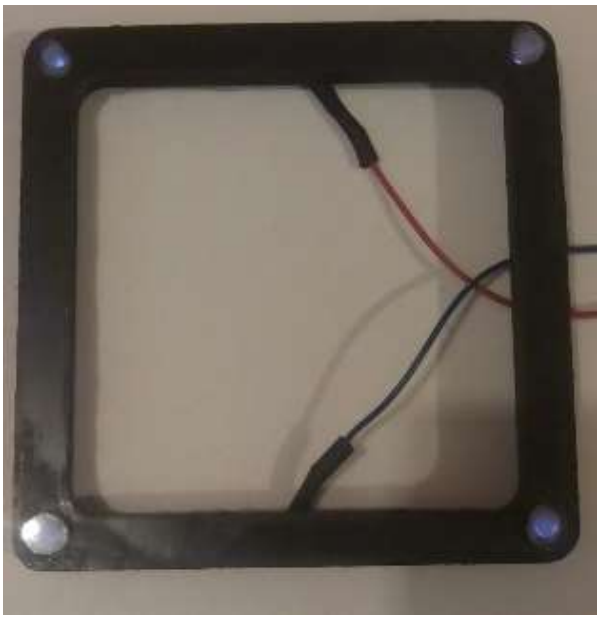


Figura 4.4. Imagen obtenida desde la cámara sin filtro infrarrojo.

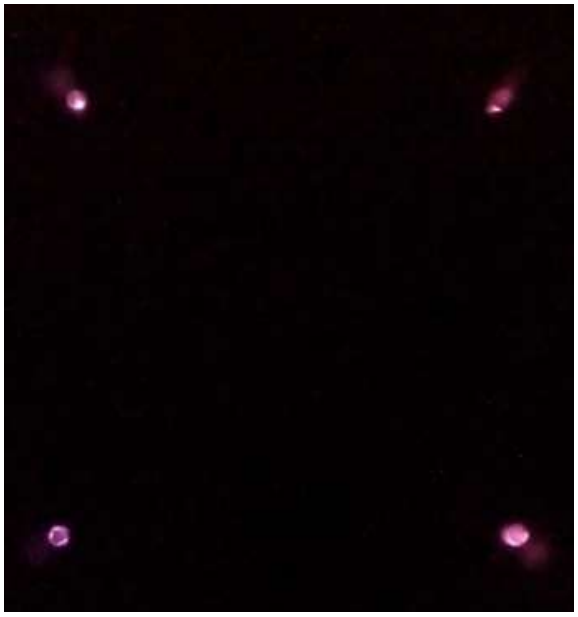


Figura 4.5. Imagen obtenida desde la cámara con filtro infrarrojo.

La cámara que se utiliza para las pruebas es la cámara Web Perfect Choice PC-320449 [14]que se muestra en la figura 4.6. Aunque esta cámara no posee las mejores características técnicas, consideramos que es suficiente como una primera aproximación para obtener resultados experimentales que permitan comprobar la teoría.

Figura 4.6. Cámara utilizada para las pruebas.

4.3 Calibración de la cámara

Para la calibración de la cámara se hizo uso de la herramienta **Camera Calibration Toolbox de Matlab** [15]. Con el patrón de rejilla que se muestra en la figura 4.7, se toman 20 imágenes (pueden ser menos) alejando y acerando la rejilla de forma aleatoria, esto para mejorar el desempeño del Toolbox. En la figura 4.8 se muestran 4 imágenes que se tomaron a la rejilla con la cámara.

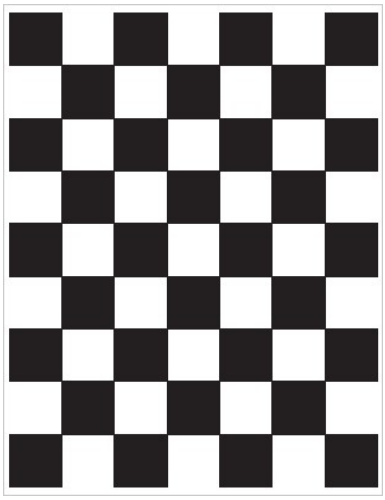


Figura 4.7. Patrón de rejilla para calibración de cámara

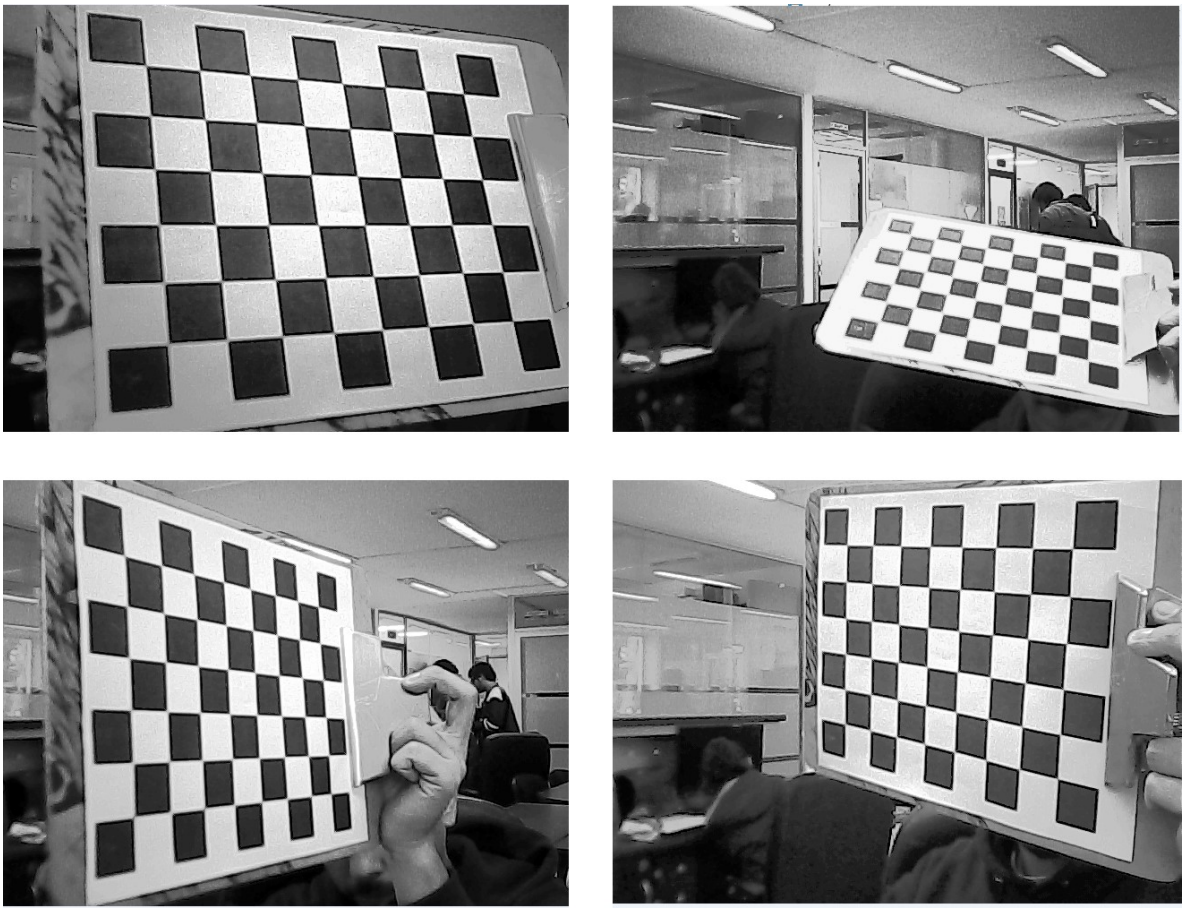


Figura 4.8. Imágenes tomadas de la rejilla desde diferentes posiciones.

Enseguida dentro de Matlab se ejecuta el programa de calibración, el cual desplegara una ventana con varias opciones como se muestra en la figura 4.9.

Figura 4.9. Herramienta para calibración de la cámara.

Se da click en el botón “Image names”, se troduce el nombre base de las imágenes de calibración (**imagen**) y el formato de imagen (**TIF**). Todas las imágenes se cargan en la memoria. La ventana de comandos de Matlab desplegara la siguiente información y se desplegara una ventana (figura 4.10) con todas las imágenes cargadas.

```
Basename camera calibration images (without number nor suffix): Imagen
Image format: ([]='r'='ras', 'b'='bmp', 't'='tif', 'p'='pgm', 'j'='jpg', 'm'='ppm') t
Loading image 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...
done
```

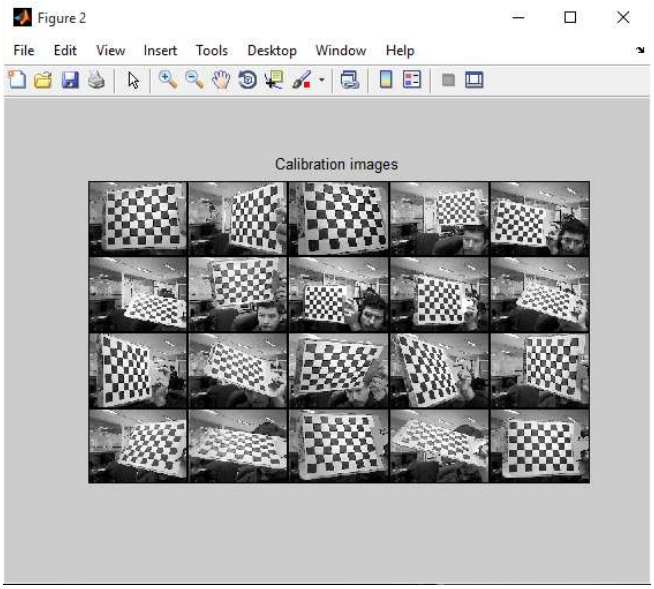


Figura 4.10. Imágenes cardadas en el Camera Calibration Toolbox de Matlab

Posteriormente se extraen las esquinas de la rejilla en todas las imágenes dando click en el botón “Extract grid corners”, se nos pedirá ingresar el número de imágenes a las cuales se les extraerán las esquinas, también se tendrá que ingresar la ventana o tamaño en pixeles del buscador de esquinas, el cual por defecto tiene el valor de 5, en este caso de dejo con el valor por default. Se selecciona el buscador de esquinas automático, enseguida se desplegará una ventana en la cual es necesario seleccionar cuatro esquinas de la rejilla con el mouse, cuando se selecciona la última esquina el software reconocerá automáticamente todas las esquinas, esto se puede ver en la figura 4.11. Se introduce los tamaños **dX** y **dY** en X e Y de cada cuadrado de la cuadrícula en este caso es de 7.8mm.

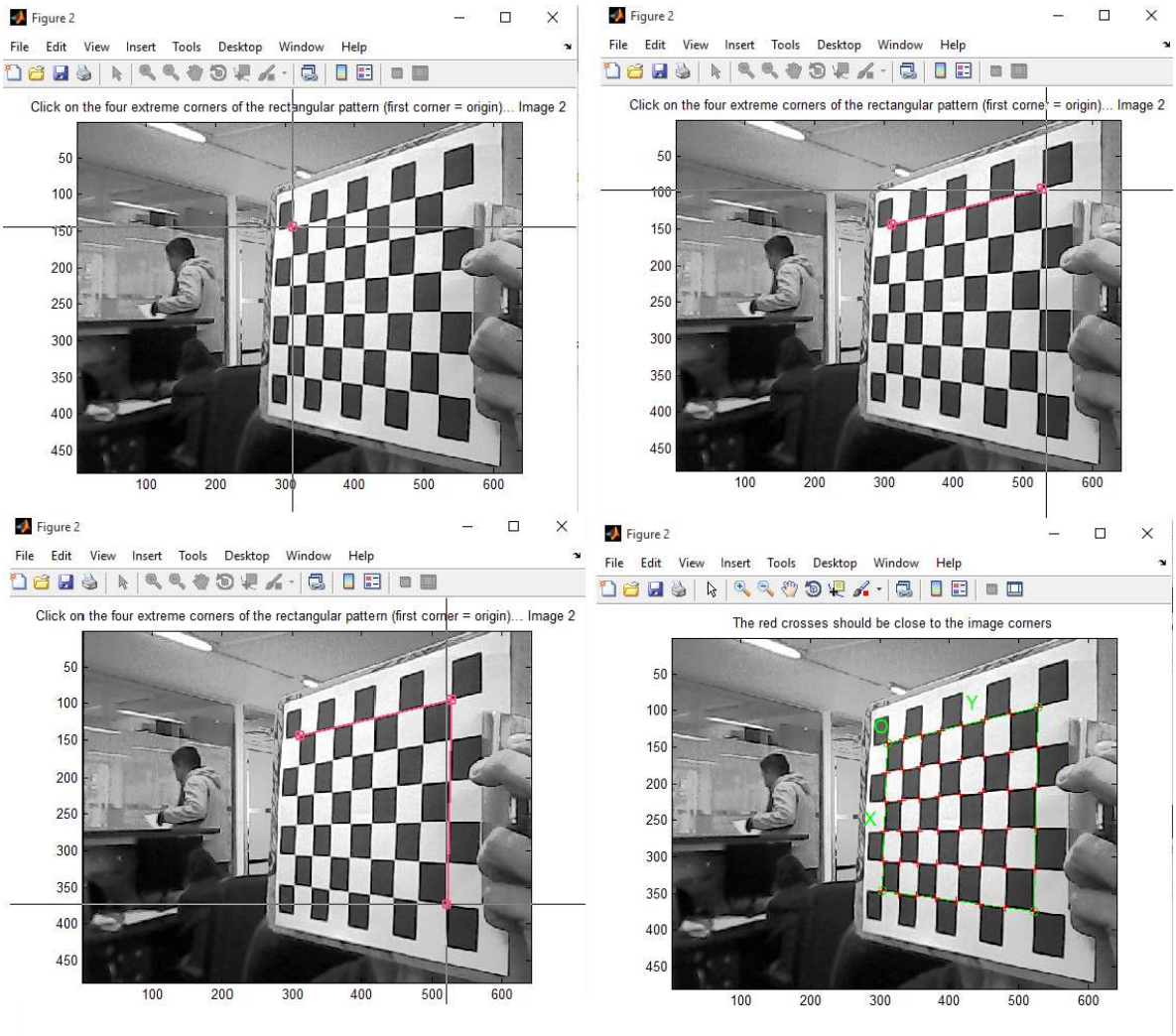


Figura 4.11. Selección de las esquinas.

Una vez realizado este proceso se tiene que realizar para todas las imágenes cargadas. Al final Matlab desplegara el siguiente mensaje:

```
Processing image 20...
Using (wintx,winty)=(5,5) - Window size = 11x11      (Note: To reset the window size, run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first clicked corner is the origin)...
Size of each square along the X direction: dX=27.8mm
Size of each square along the Y direction: dY=27.8mm    (Note: To reset the size of the squares, clear the variables dX and dY)
If the guessed grid corners (red crosses on the image) are not close to the actual corners,
it is necessary to enter an initial guess for the radial distortion factor kc (useful for subpixel detection)
Need of an initial guess for distortion? ([]=no, other=yes)
Corner extraction...
done
```

Una vez realizados los pasos anteriores se procedió a hacer la calibración haciendo click en el botón “Calibration” el Camera Calibration Toolbox nos da como resultado los parámetros siguientes:

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated (DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point, set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
    Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .
Initialization of the principal point at the center of the image.
Initialization of the intrinsic parameters using the vanishing points of planar patterns.

Initialization of the intrinsic parameters - Number of images: 20


Calibration parameters after initialization:

Focal Length:         fc = [ 844.91562   844.91562 ]
Principal point:      cc = [ 319.50000   239.50000 ]
Skew:           alpha_c = [ 0.00000 ]   => angle of pixel = 90.00000 degrees
Distortion:           kc = [ 0.00000   0.00000   0.00000   0.00000   0.00000 ]

Main calibration optimization procedure - Number of images: 20
Gradient                                                descent                                                iterations:
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...22...23...24...25...26...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:         fc = [ 858.53407   860.30193] +/- [ 6.05469   5.76163]
Principal point:      cc = [ 339.75465   206.77321] +/- [ 13.41846   10.27348]
Skew:           alpha_c = [ 0.00042] +/- [ 0.00019]   => angle of pixel axes = 89.97622+/- 0.01092 degrees
Distortion:           kc = [ 0.03308   0.17347   -0.01256   0.00155 0.00000] +/- [ 0.05211   0.46473   0.00484
0.00616 0.00000]
Pixel error:         err = [ 0.47038   0.40408]

Note: The numerical errors are approximately three times the standard deviations (for reference).


Recommendation: Some distortion coefficients are found equal to zero (within their uncertainties).
          To reject them from the optimization set est_dist=[1;0;1;1;0] and run Calibration
```

Con estos datos podemos obtener la matriz de parámetros intrínsecos, sabiendo que:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{f_c(1)} & \boldsymbol{alpha\_c * f_c(1)} & \boldsymbol{cc(1)} \\ 0 & \boldsymbol{f_c(2)} & \boldsymbol{cc(2)} \\ 0 & 0 & 1 \end{bmatrix}$$

Por último, para para visualizar el efecto de las distorsiones de la imagen se ejecuta el script “visualize_distortions” que viene con el Camera Calibration Toolbox y nos despliega la figura 4.12, cabe aclarar que la figura 4.12 es la distorsión particular de la cámara con la que se realizó la calibracion:

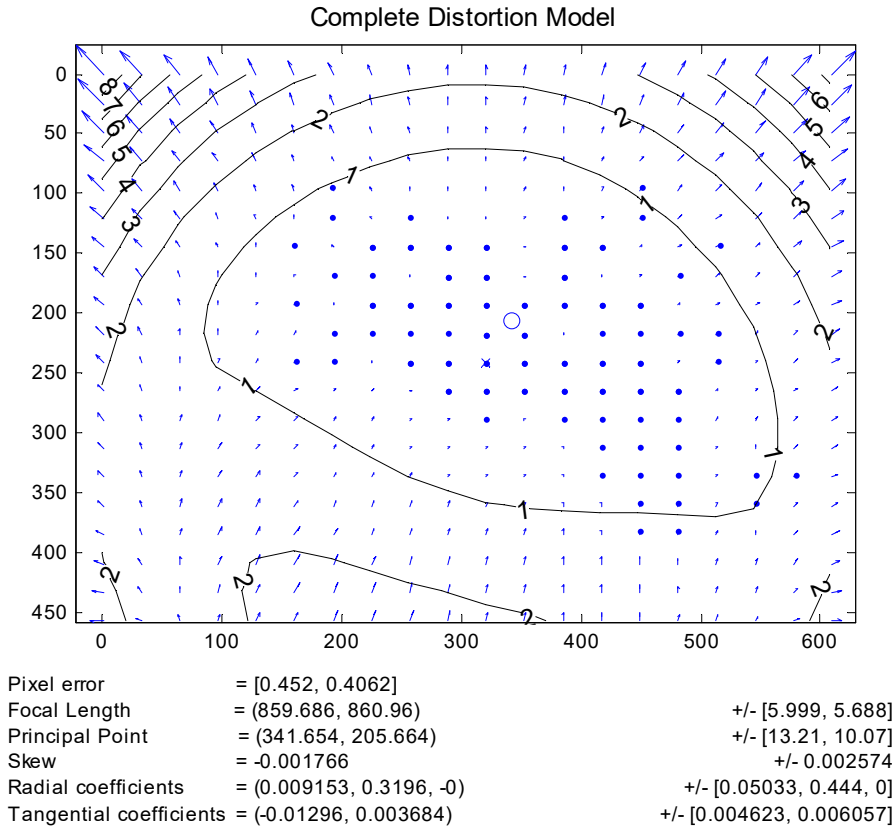


Figura 4.12. Distorsión de la lente en la imagen.

4.4 Implementación del software

En la figura 4.13 se presenta el diagrama de flujo del software que se implementó en el lenguaje de programación de Matlab[16] para la detección del plano y el cálculo de posición y orientación.

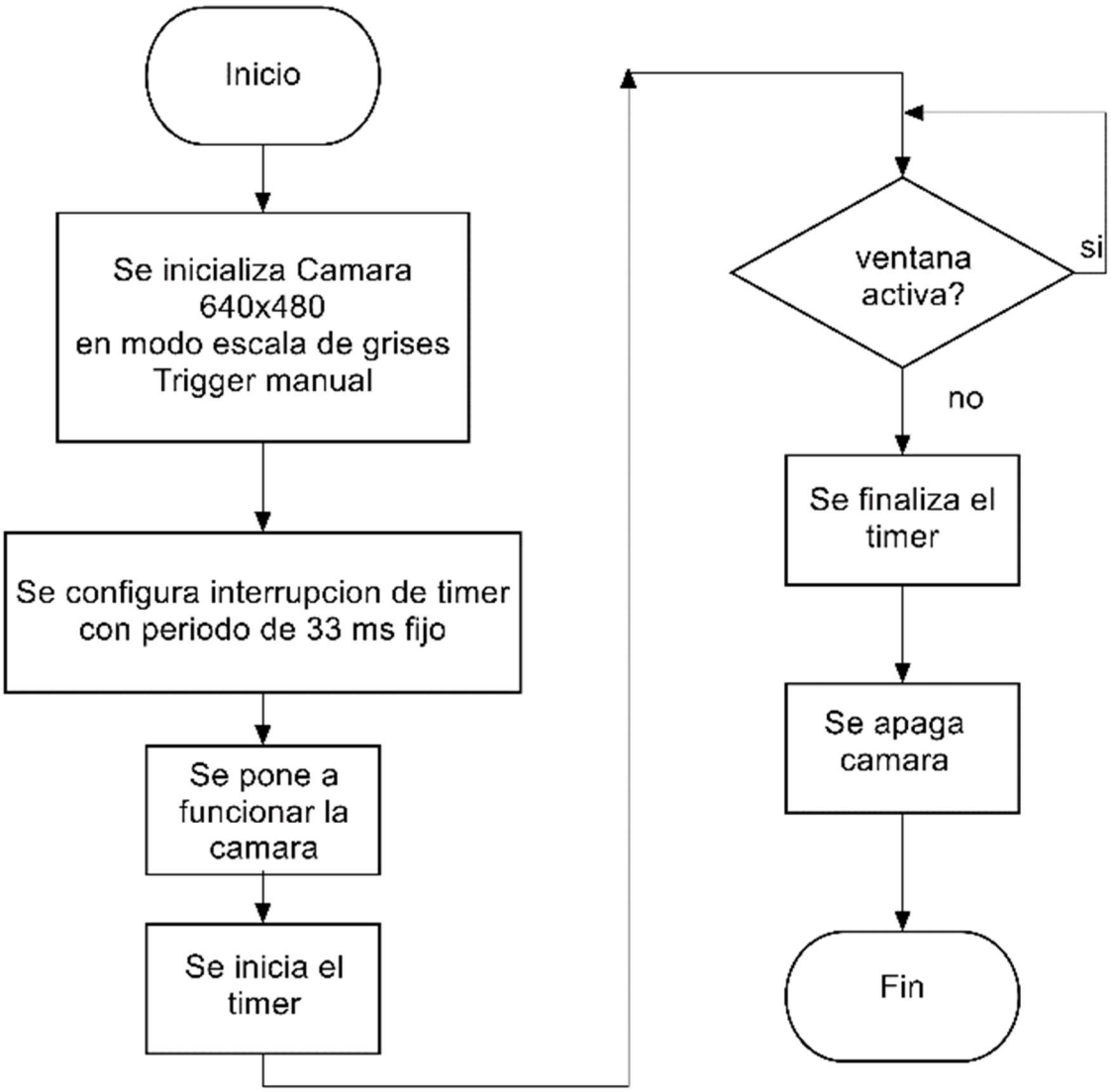


Figura 4.13. Diagrama de flujo del programa principal.

El primer paso es la configuración de la cámara y adquisición de las imágenes para lo cual se utilizó la herramienta Image Acquisition Toolbox de MatLab. Con esta herramienta se configuró la cámara para que capture las imágenes con una resolución de 640x480 pixeles, también se configuró para que tome las imágenes en escala de grises y que se capture la imagen en forma manual. Posteriormente se configura un timer para mandar una interrupción cada 33 ms ya que la cámara toma imágenes a 30 cuadros por segundo y esta velocidad es fijada por el fabricante.

Para la configuración de la cámara se requiere tener instalado Image Acquisition Toolbox de MatLab, y es necesario ver la información de los adaptadores disponibles (Camaras web), para esta tarea se utilizó el comando *imaqhwinfo*. El cual despliega la siguiente información.

```
>> imaqhwinfo
ans =
    InstalledAdaptors: {'winvideo'}
```

```
        MATLABVersion: '8.3 (R2014a)'
          ToolboxName: 'Image Acquisition Toolbox'
       ToolboxVersion: '4.7 (R2014a)'
```

Con esta información se puede obtener las propiedades de un adaptador especifico utilizando el mismo comando, pero agregando como argumento el nombre del controlador de video de la cámara.

```
imaqhwinfo('winvideo')
ans =
       AdaptorDllName:
'C:\MATLAB\SupportPackages\R2014a\genericvideo\adaptor\win64\mwwinv...'
    AdaptorDllVersion: '4.7 (R2014a)'
          AdaptorName: 'winvideo'
            DeviceIDs: {[1]}
           DeviceInfo: [1x1 struct]
```

Posteriormente es necesario ver la información de la cámara, como por ejemplo el modo de color y la resolución, de la cámara que estamos usando; debemos indicar el identificador del dispositivo.

```
>> imaqhwinfo('winvideo',1)
ans =
             DefaultFormat: 'YUY2_160x120'
       DeviceFileSupported: 0
                DeviceName: 'Amcap'
                  DeviceID: 1
     VideoInputConstructor: 'videoinput('winvideo', 1)'
    VideoDeviceConstructor: 'imaq.VideoDevice('winvideo', 1)'
          SupportedFormats: {1x5 cell}
```

Para inicializar la cámara debemos indicar el adaptador (winvideo), el identificador del dispositivo (1) y la resolución junto con el formato con el que se desea trabajar (YUYV_160x120). La siguiente función genera objeto con el que se inicializa la cámara.

```
winvi = videoinput('winvideo',1,'YUYV_160x120');
```

Una vez que se creó el objeto, este se utiliza para iniciar la cámara y posteriormente poder adquirir imágenes. Para esta tarea se utiliza la función start.

```
start(winvi);
```

Si se necesita previsualizar la imagen, se puede utilizar el siguiente comando.

```
preview(winvi);
```

Para pedir la imagen a la cámara se utiliza la siguiente función.

```
img = getdata(winvid);
```

Para la configuración de la cámara fue necesario configurarla de forma específica, esto es como ya se mencionó, que las imágenes estuvieran en escala de grises, tuvieran una resolución de 640X480 pixeles y que se tomara una imagen cada 33ms. Para dar el formato y la resolución a la cámara se utilizó el comando videoimput, sin embargo, para obtener las imágenes en escala de grises se utiliza el comando set, poniendo la propiedad ReturnedColorSpace en grayscale como se muestra a continuación.

```
set(winvid,'ReturnedColorSpace','grayscale');
```

De esta forma la imagen automáticamente se guarda en formato de escalas de grises. Con esto se evita obtener una imagen a color y tener que pasarla a escala de grises consumiendo tiempo de procesado del software.

Como segundo paso se tiene que configurar que la cámara para que tome imágenes después de una llamada del programa (trigger), de esta forma desde el software se podrá controlar cuando se toma la imagen. Con la función triggerconfig se logra esto, y se tiene que dar como argumento el objeto a controlar en este caso winvid también se le tiene que agregar la sentencia "Manual", esto para que se active por software y no por otro evento diferente, como sería un botón o por eventos automáticos de la cámara. También es necesario indicarle a la cámara cuantas veces se tiene que repetir el trigger, con ayuda del comando set se configura que el trigger, ya que si TriggerRepeat está ajustado a su valor por defecto de 0, entonces el disparo se produce una vez, si TriggerRepeat se establece en un valor entero positivo, entonces el gatillo se repite el número de veces especificado y si TriggerRepeat se establece en inf el trigger se repite continuamente y se puede detener el objeto winvid únicamente mediante la emisión de la función stop.

```
triggerconfig(winvid,'Manual');
set(winvid,'TriggerRepeat',Inf);
```

Posteriormente se tiene que configurar cuantas imágenes se obtienen al activar el trigger, ya que por defecto el número de imágenes tomadas es de 10 y solo es necesario obtener una imagen. La propiedad FramesPerTrigger es la que se encarga de esto. La función debe de escribirse de la siguiente manera:

```
set(winvid,'FramesPerTrigger',1);
```

Una vez que está configurada la cámara como es requerida se tiene que generar una interrupción de 33ms para obtener la imagen a 30 cuados por segundo. Para esto se utiliza la función timer que ayuda a generar una interrupción que depende del tiempo especificado. La forma de hacer esto es como se muestra en el siguiente pedazo de código:

```
TimerData=timer('TimerFcn',{@FrameRateDisplay,winvid},'Period',1/30,...
                'ExecutionMode','fixedRate');
```

Donde a la propiedad TimerFcn se le tiene que especificar 1) el nombre del archivo donde se encuentra el código a ser ejecutado una vez que se inicia la interrupción 2) el objeto o variables externas si es que la hay para que sean reconocidos por el programa a ejecutarse, en este caso FrameRateDisplay es el nombre del fichero el cual tiene contiene el código a ejecutarse durante la interrupción y winvid es la variable objeto que contiene la información de configuración de la cámara. El valor del periodo de tiempo que se ejecutara se define con la propiedad Period. La propiedad ExecutionMode se tiene que poner en fixedRate para configurar que el timer se reinicie inmediatamente después de que se hizo la llamada a la función que atenderá la interrupción. Esto garantiza que la interrupción sea a 33ms. Con esto se logra generar la interrupción y será guardada en la variable TimerData.

Para iniciar tanto la configuración de la cámara como iniciar la interrupción se utiliza la función start, como se indica a continuación.

```
start(winvid);
start(TimerData);
```

Una vez que se termina el programa es necesario terminar la interrupción y liberar de la configuración que le asignamos a la cámara con la ayuda de la función stop, también se tiene que limpiar de la memoria las variables que se crearon para la configuración. Esto se hace hasta que se quiere terminar la ejecución del programa.

```
stop(TimerData);
delete(TimerData);
stop(winvid);
delete(winvid);
```

En la figura 4.14 se muestra el diagrama de flujo de la subrutina que se ejecuta cada vez que la interrupción del timer es activada. Como se muestra en el diagrama de flujo de la figura 4.14 cuando se entra a la subrutina lo primero que se hace es indicarle a la cámara que tome una imagen o frame, esto se logra con la función trigger.

```
trigger(winvid);
```

Una vez que se utilizó la función trigger la imagen queda guardada en la memoria de la cámara y es necesario obtener la imagen, para esta tarea se utiliza la función getdata de la siguente forma:

```
frame=getdata(winvid,1,'uint8');
```

Los argumentos para la función getdada que se pueden ver arriba son: primero la variable objeto que define a la cámara despues el número uno que indica que solo va a recibir una imagen y por último la palabra unit8 que indica el formato con el cual se guardará la información de la imagen (información de cada pixel), este será en un formato de 8 bits enteros positivos, esto quiere decir que el valor en escala de grises que pueden obtener los pixeles van de 0 a 255. La palabra frame indica la variable donde se guardará la imagen.

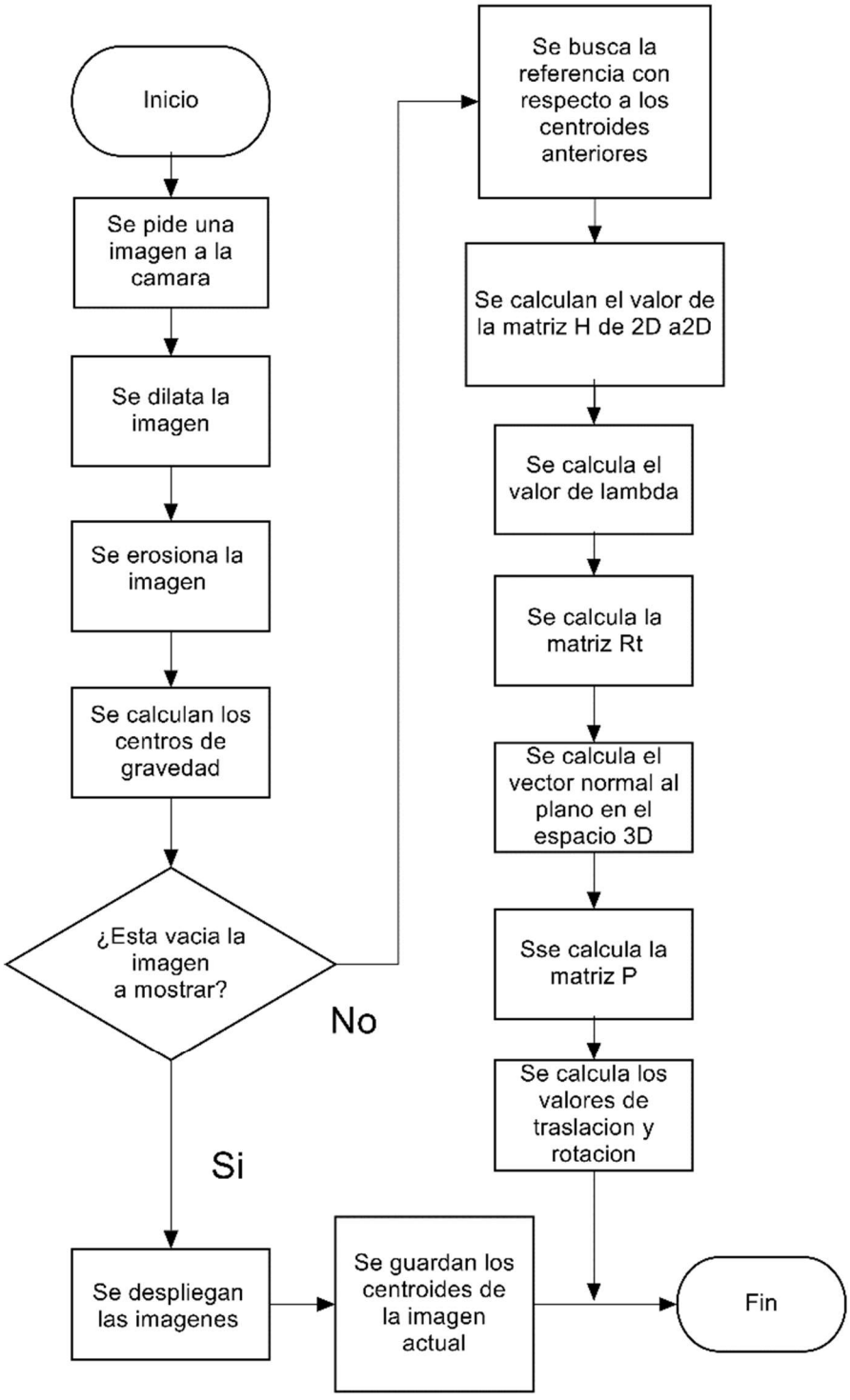


Figura 4.14 Diagrama de flujo del proceso para ejecutar la interrupción.

Una vez que el programa adquiere la imagen se procede a realizar un preprocesamiento, con el fin de eliminar ruido, de tal manera que en el procesamiento posterior se obtengan mejores resultados. Este preprocesamiento consiste en una binarización, dilatación, erosión, etiquetado y cálculo de centroides.

Para la binarización solo se hace la comparación del valor de cada pixel con un umbral como ya se mencionó, esto se logra con una línea de código como se muestra a continuación.

```
frameBin=frame>=100;
```

Aquí se indica que los valores que son mayores o iguales a 100 se pondrán en uno y los que son menores se pondrán en cero. En la figura 4.15 se puede ver el resultado de la binarización.

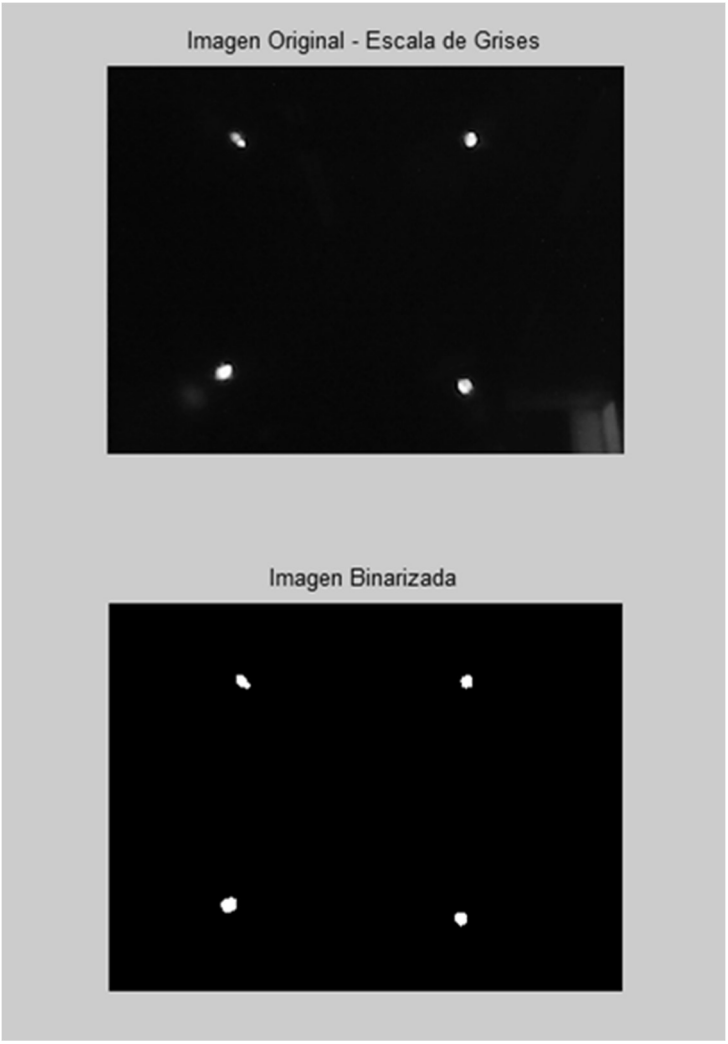


Figura 4.15. Comparación de imagen en escala de grises e imagen binarizada.

Después del binarizado se dilato y posteriormente se erosiono la imagen esto permite que se eliminen ruidos, por ejemplo, en la figura 4.16 se puede ver que en la imagen de la izquierda hay cinco puntos el quinto punto (el punto que se encuentra más abajo) es el reflejo de la luz sobre la mesa en la que se encuentra el marco con 4 led´s. Como se puede apreciar al hacer la dilatación el punto más abajo se une con el punto de arriba generando un solo punto (imagen del centro), y la última imagen es el resultado de la erosión, en esta se puede ver como ahora queda un solo punto lo cual elimina el punto extra que en realidad no existía.

Para hacer la dilatación y la erosión se utilizan 3 funciones la cuales son mostradas a continuacion:

```
se1 = strel('disk',10);
imclose(frameBin,se1);
imopen(frameBin,se1);
```

La primera función strel genera una imagen binaria la cual tiene una forma específica que depende del primer argumento, en este caso sería disk el cual genera un circulo con radio de 10 pixeles, este radio está dado por el segundo argumento de la función. La imagen tiene una dimensión de 10X10 pixeles.

Las otras dos funciones imclose e imopen son las funciones que hacen la erosión y la dilatación respectivamente, estas funciones utilizan como argumentos la imagen a dilatar o erosionar y la imagen generada por la función strel.

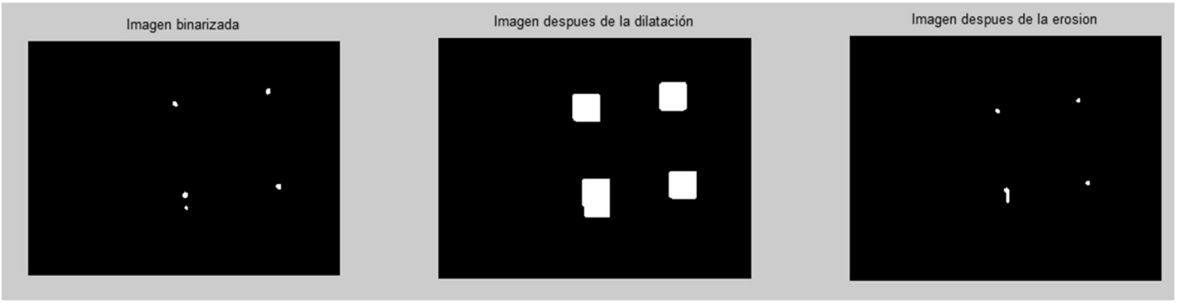


Figura 4.16. Izquierda: Imagen binaria original (5 puntos). Centro: Puntos dilatados. Derecha: Puntos erosionados (4 puntos).

Para el cálculo del etiquetado y los centroides se utilizó la función regionprops, la cual permite medir un conjunto de propiedades definidas en la tabla 2. Para cada uno de los objetos etiquetados. La sintaxis general para esta función es la siguiente:

```
stats=regionprops(Imagen_Binaria,propiedades);
```

Tabla 4.1. Propiedades que pueden ser calculadas por la función regionprops.

Propiedad	Descripción
'Area'	Encuentra el área del objeto etiquetado.
'BuondigBox'	Encuentra el cuadrado mínimo que engloba al objeto.
'Centroid'	Encuentra el centroide del objeto.
'ConvexHull'	Encuentra el polígono mínimo que encierra al objeto
'ConvexArea'	Encuentra el área que encierra el polígono determinado por ConvexHull
'EulerNumber'	Encuentra el número de Euler del objeto.
'MajorAxisLength'	Encuentra el tamaño del eje que longitudinalmente es más grande en el objeto.
'MinorAxisLength'	Encuentra el tamaño del eje que longitudinalmente es más pequeño en el objeto.

‘Orientation’	Encuentra la orientación del eje longitudinal máximo con respecto a la horizontal.
‘Perimeter’	Calcula el perímetro del objeto etiquetado.

La función regionprops regresa como resultado un arreglo de estructuras llamado stats. Una estructura es una agrupación de datos de tipo diferente bajo un mismo nombre. Estos datos se llaman campos (fields) son accedidos mediante el formato:

Dato=Nombre_de_la_estructura.field(1)

Donde a la variable Dato se le asigna el valor del índice 1 del campo field de la estructura Nombre_de_la _estructura. Por lo tanto se tiene que para sacar los centroides se puede hacer lo siguiente:

```
centroides=cat(1,Im_label.Centroid);
```

La función cat en este caso sirve para concatenar todos los valores de los centroides en un solo arreglo y sea más fácil su manipulación.

El inconveniente de llevar acabo el etiquetado, con la función regionprops, es que para encontrar los objetos, el algoritmo de etiquetado hace un barrido de izquierda a derecha y de arriba abajo, etiquetando como uno al primer objeto que se encuentra. Esto hace que perdamos el punto de referencia que escogimos para poder hacer los cálculos. Como la rotación de los puntos es aleatoria es necesario marcar un punto y seguirlo para tener la certeza de que los cálculos son correctos.

Para resolver este inconveniente se propuso el siguiente diagrama de flujo mostrado en la figura 4.17.

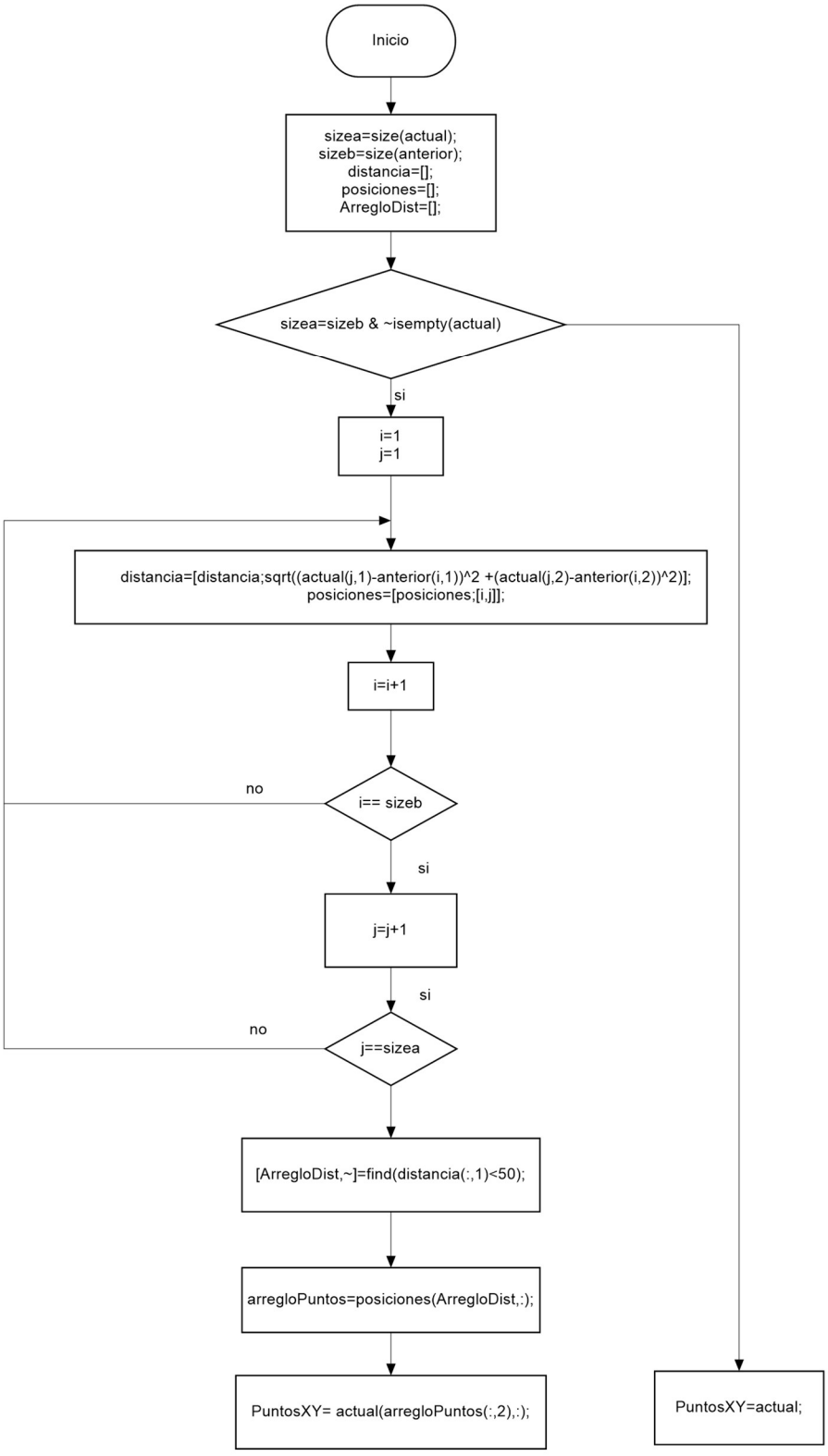


Figura 4.17. Diagrama de flujo para seguimiento de un punto

El algoritmo guarda los centroides de la imagen anterior y las compara con los centroides de la imagen actual (figura 4.18). Calculando las distancias Euclidianas entre los centroides se determinó las distancias más cortas entre los puntos, y se pudo relacionar la posición de los centroides actuales con respecto a los centroides anteriores. Esto se puede ver gráficamente en figura 4.19 que es la superposición de las dos imágenes, en esta figura se ven cuatro líneas que parten de un punto, la línea más corta en este caso la de color rojo es la distancia más corta, y por lo tanto, el punto del que parten todas las líneas y el punto en el que termina la línea roja es el mismo punto en diferentes imágenes.

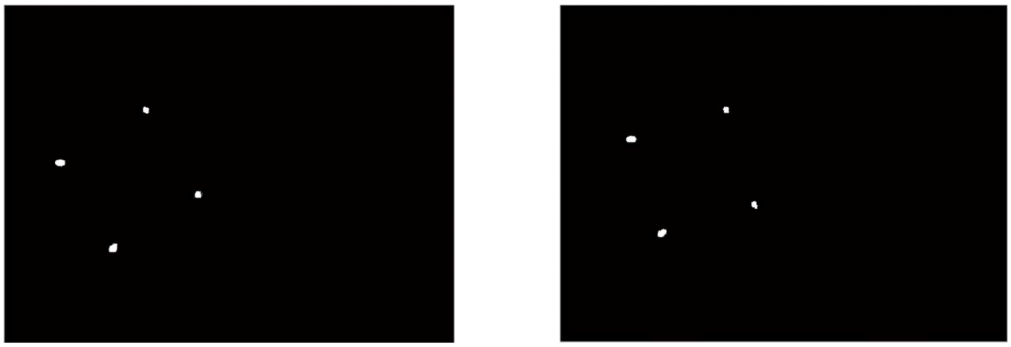


Figura 4.18. Izquierda: Imagen actual. Derecha: Imagen anterior.

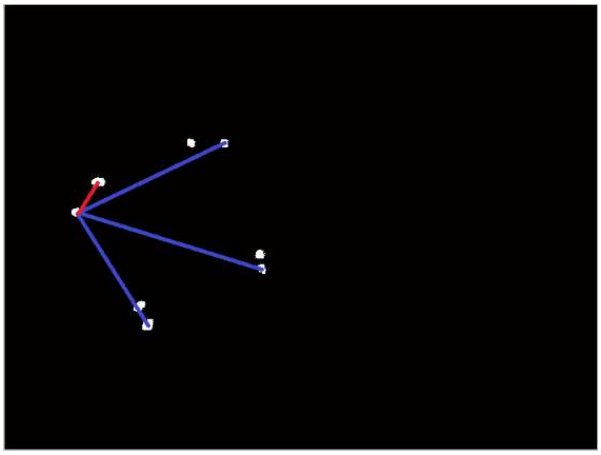


Figura 4.19. Imágenes anterior y actual sobrepuestas, las líneas representan las distancias Euclidianas entre un centroide de la imagen actual y los centroides de la imagen anterior.

Una vez que se tienen los centroides se puede empezar a calcular la matriz de proyección de 2D a 2D , para esto se toman los valores de los centroides como los valores u y v que vienen a ser las coordenadas en la imagen de los puntos que están proyectados en la imagen. De esta forma se tiene que:

```
Pc1=centroides(1,:);
Pc2=centroides(2,:);
Pc3=centroides(3,:);
Pc4=centroides(4,:);
```

Donde Pc1, Pc2, Pc3 y Pc4 son las coordenadas de cada uno de los puntos en forma euclidiana (u, v). Por otro lado, los puntos en el plano físico se escogen de tal forma que coincidan con coordenadas en un plano, esto se puede ver en la figura 4.20 donde se muestra

las coordenadas de los cuatro puntos (led´s). Estas coordenadas están en milímetros lo cual haría que todos los resultados estén en milímetros. Estos valores de coordenadas son constantes, ya que el que se mueve es el plano, pero, los puntos con respecto al plano no se mueven por lo tanto podemos hacer lo siguiente:

```
Pm1=[0 0 1];
Pm2=[100 0 1];
Pm3=[100 100 1];
Pm4=[0 100 1]
```

Donde Pm1, Pm2, Pm3 y Pm4 son las coordenadas homogéneas del plano físico.

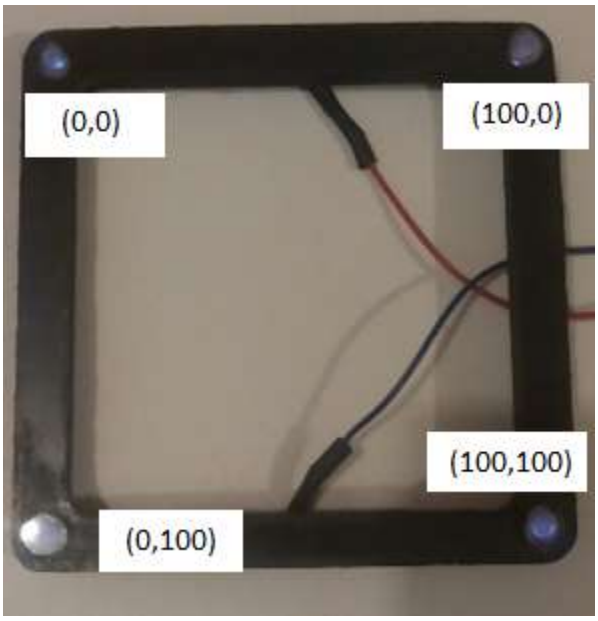


Figura 4.20. Coordenadas (X,Y) del plano físico.

Con estos valores es posible armar la matriz 4.5 que es la matriz que nos permite calcular los valores de la matriz ***H***. Por lo tanto, se tiene que:

```
A(1,[1:3]) = Pm1;
A(2,[4:6]) = Pm1;
A(1,7) = -Pc1(1)*Pm1(1);
A(1,8) = -Pc1(1)*Pm1(2);
A(2,7) = -Pc1(2)*Pm1(1);
A(2,8) = -Pc1(2)*Pm1(2);
A([1:2],9) = -Pc1';

A(3,[1:3]) = Pm2;
A(4,[4:6]) = Pm2;
A(3,7) = -Pc2(1)*Pm2(1);
A(3,8) = -Pc2(1)*Pm2(2);
A(4,7) = -Pc2(2)*Pm2(1);
A(4,8) = -Pc2(2)*Pm2(2);
A([3:4],9) = -Pc2';

A(5,[1:3]) = Pm3;
```

```
A(6,[4:6]) = Pm3;
A(5,7) = -Pc3(1)*Pm3(1);
A(5,8) = -Pc3(1)*Pm3(2);
A(6,7) = -Pc3(2)*Pm3(1);
A(6,8) = -Pc3(2)*Pm3(2);
A([5:6],9) = -Pc3';

A(7,[1:3]) = Pm4;
A(8,[4:6]) = Pm4;
A(7,7) = -Pc4(1)*Pm4(1);
A(7,8) = -Pc4(1)*Pm4(2);
A(8,7) = -Pc4(2)*Pm4(1);
A(8,8) = -Pc4(2)*Pm4(2);
A([7:8],9) = -Pc4';
```

Ya que se tiene la matriz 4.5 es necesario resolverla para encontrar los parámetros h_{ij} de la matriz $\boldsymbol{H}$. De esta forma el problema se resume en resolver un sistema matricial $AX = 0$, que puede ser visto como un problema de eigenvalores para $B = A^T A$ [17]. Donde debemos buscar el eigenvector relacionado con el menor eigenvalor obtenido para B, para calcular los valores de los elementos de $\boldsymbol{H}$. Esto se logra multiplicando la matriz A por su transpuesta y posteriormente se encuentran sus eigenvectores que se guardan en la matriz $\boldsymbol{V}$.

```
B = A'*A;
[V,D] = eig(B);
```

Se reacomodan los valores de la matriz $\boldsymbol{V}$para dar como resultado la matriz $\boldsymbol{H}$.

```
H(1,:) = V([1:3],1);
H(2,:) = V([4:6],1);
H(3,:) = V([7:9],1);
```

El siguiente paso es encontrar el valor de λ, con la ecuación 9.4 se puede escribir:

```
lambda = norm(K\h1);
```

donde:

```
h1 = H(:,1);
```

Ya con esta información se puede encontrar la matriz de proyección 2D, que quedaría de la siguiente forma (la variable que contiene la matriz seria KRT):

```
RT = K\H ./ lambda;
```

Una vez que se tiene la matriz de proyección en 2D es posible encontrar el vector $\boldsymbol{r}_3$para esto se toma los vectores $\boldsymbol{r}_1$ y $\boldsymbol{r}_2$ y se calcula el producto vectorial con estos dos vectores.

```
r1 = RT(:,1);
r2 = RT(:,2);
r3 = cross(r1,r2);
```

Con todos estos datos es posible encontrar la matriz de proyección de 3D a 2D la cual queda definida por

```
t = RT(:,3);
P = [r1 r2 r3 t];
```

Por último, se tiene que extraer la información, como ya se mostró, el vector de traslación queda definido por $\boldsymbol{t} = (x, y, z)^T$,mientras que los valores de los ángulos quedan definidos por:

```
thetaY=-(180/pi)*asin(r3(1));
thetaX=(180/pi)*asin(r3(2)/cos(asin(r3(1))));
thetaZ=(180/pi)*acos(r1(1)/cos(asin(r3(1))));
```

Para comprobar que la matriz de transformación funciona correctamente se proyectaron 8 puntos en una imgen de tal forma que se construyera un cubo. Estos puntos quedan definidos de la siguiente manera:

```
p1 = [0,0,0,1];
p2 = [100,0,0,1];
p3 = [100,100,0,1];
p4 = [0,100,0,1];
```

Estos puntos (homogéneos) se encuentran sobre el plano esto quiere decir que el valor de la componente $Z = 0$. Pero para que se forme un cubo es necesario generar otros cuatro puntos que sean coplanares y que el plano al que pertenecen sea paralelo al plano $Z = 0$. Por esta razón los puntos nos quedarían definidos como:

```
p5 = [0,0,-100,1];
p6 = [100,0,-100,1];
p7 = [100,100,-100,1];
p8 = [0,100,-100,1];
```

Para proyectar los puntos será necesario multiplicarlos por la matriz de transformación de 3D a 2D multiplicada por la matriz de parámetros intrínsecos $\boldsymbol{K}$. Posteriormente se tienen que normalizar de tla manera que:

```
Pk = K * [r1 r2 r3 t];
Pp1 = Pk*p1'; %Proyectamos
Pp1 = Pp1 ./ Pp1(3,1); %Normalizamos
Pp2 = Pk*p2'; %Proyectamos
Pp2 = Pp2 ./ Pp2(3,1); %Normalizamos
Pp2 = Pk*p2'; %Proyectamos
Pp2 = Pp2 ./ Pp2(3,1); %Normalizamos
Pp3 = Pk*p3'; %Proyectamos
Pp3 = Pp3 ./ Pp3(3,1); %Normalizamos
Pp4 = Pk*p4'; %Proyectamos
Pp4 = Pp4 ./ Pp4(3,1); %Normalizamos
Pp5 = Pk*p5'; %Proyectamos
Pp5 = Pp5 ./ Pp5(3,1); %Normalizamos
Pp6 = Pk*p6'; %Proyectamos
Pp6 = Pp6 ./ Pp6(3,1); %Normalizamos
Pp7 = Pk*p7'; %Proyectamos
Pp7 = Pp7 ./ Pp7(3,1); %Normalizamos
Pp8 = Pk*p8'; %Proyectamos
Pp8 = Pp8 ./ Pp8(3,1); %Normalizamos
```

Con esto se dibujará un cubo como se muestra en la figura 4.21

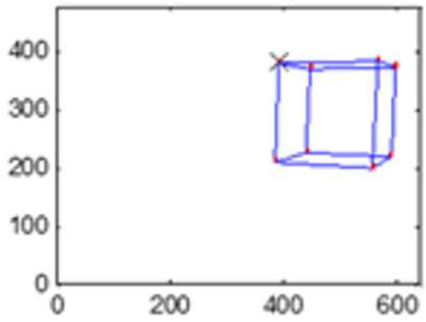


Figura 4.21. Proyección de 8 puntos que forman un cubo.

4.5 Hardware

Una vez que se probó la teoría con Matlab se implementó en hardware una cámara de bajo costo y esta se controló con un Arduino Due. Con este hardware se logró hacer el preprocesamiento de la imagen hasta sacar los centroides, una vez que se obtuvieron los centroides estos eran enviados a través de la interfaz USB del Arduino.

Para esto se escogió el Arduino Due ya que este tiene los niveles de voltaje para hacer la interfaz de la cámara. También se eligió el Arduino Due que posee ya que la cámara no posee un registro FIFO (Frist In Frist Out) y es necesario contar con suficiente memoria ram.

4.5.1 Tarjeta Arduino

Por todo lo anterior se utilizará la tarjeta Arduino Due [18] ya que funciona con niveles de voltajes de 0 a 3.3 volts, lo cual es necesario ya que la cámara funciona con los mismos niveles de voltaje, ayudando esto a no caer en la necesidad de utilizar adaptadores de señal evitando ruidos innecesarios.

La tarjeta Arduino se muestra en la figura 4.22:

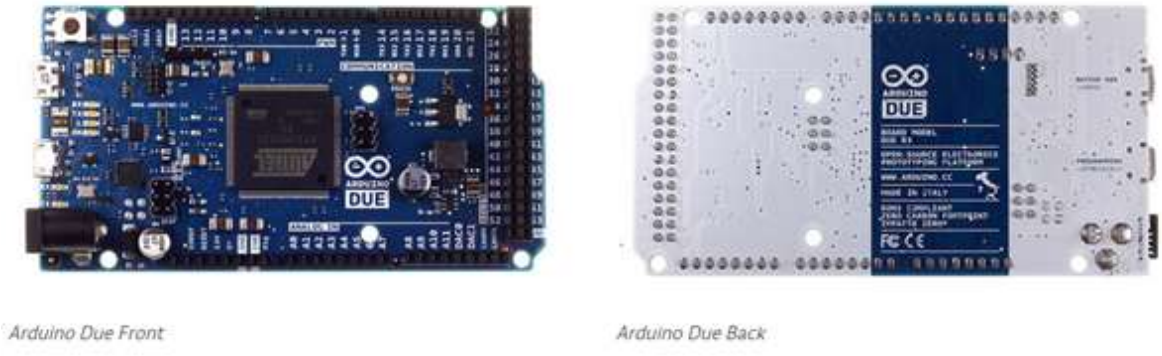


Figura 4.22. Vista de frente y posterior del Arduino Due

4.5.2 4.5.2 Descripción general del Arduino Due.

El Arduino Due es una tarjeta basada en el microcontrolador de Atmel SAM3X8E ARM Cortex-M3 CPU. Esta es la primera tarjeta de Arduino basada en un microcontrolador ARM de 32 bits (ARM es el mnemónico de Advanced RISC Machine).

Tiene 54 pines de entrada/salida digitales (de los cuales 12 pueden emplearse como salidas de PWM), 12 entradas analógicas, 4 UARTs (puerto serial por hardware), un reloj de 84 MHz, un USB On-The-Go (USB OTG), 2 DAC (convertidor digital analógico) 2 TWI (Two Wire Interface), un conector de alimentación, un conector SPI, un conector de JTAG (Join Test Accion Group), un botón de reset y un botón de borrado.

El Arduino DUE continua con el estándar de pines que el Arduino One, siendo la única diferencia que este último trabaja a 5V.

4.5.3 Los beneficios del ARM Core

El Arduino Due tiene un núcleo ARM a 32 bits que puede superar a las típicas tarjetas basadas en microcontroladores de 8 bits. Las diferencias más significativas son:

- Un núcleo de 32 bits que te permite operaciones de datos de palabras de 4 bytes con un único reloj de CPU.
- El reloj de CPU a 84 MHz.
- 96 KBytes de SRAM.
- 512 Bytes de memoria Flash para programa.
- Un controlador DMA.

A continuación, en la tabla 4.2 se presenta una lista del mapeo de pines. Y en la tabla 4.3 se presentan sus características generales.

Tabla 4.2. Listado de pines de la tarjera Arduino Due.

Due Pin Number	SAM3X Pin Name	Mapped Pin Name	Max Output Current (mA)	Max Current Sink (mA)
0	PA8	RX0	3	6
1	PA9	TX0	15	9
2	PB25	Digital Pin 2	3	6
3	PC28	Digital Pin 3	15	9
4	connected to both PA29 andPC26	Digital Pin 4	15	9
5	PC25	Digital Pin 5	15	9
6	PC24	Digital Pin 6	15	9
7	PC23	Digital Pin 7	15	9
8	PC22	Digital Pin 8	15	9
9	PC21	Digital Pin 9	15	9
10	connected to both PA28 andPC29	Digital Pin 10	15	9

11	PD7	Digital Pin 11	15	9
12	PD8	Digital Pin 12	15	9
13	PB27	Digital Pin 13 / Amber LED "L"	3	6
14	PD4	TX3	15	9
15	PD5	RX3	15	9
16	PA13	TX2	3	6
17	PA12	RX2	3	6
18	PA11	TX1	3	6
19	PA10	RX1	3	6
20	PB12	SDA	3	6
21	PB13	SCL	3	6
22	PB26	Digital Pin 22	3	6
23	PA14	Digital Pin 23	15	9
24	PA15	Digital Pin 24	15	9
25	PD0	Digital Pin 25	15	9
26	PD1	Digital pin 26	15	9
27	PD2	Digital Pin 27	15	9
28	PD3	Digital Pin 28	15	9
29	PD6	Digital Pin 29	15	9
30	PD9	Digital Pin 30	15	9
31	PA7	Digital Pin 31	15	9
32	PD10	Digital Pin 32	15	9
33	PC1	Digital Pin 33	15	9
34	PC2	Digital Pin 34	15	9
35	PC3	Digital Pin 35	15	9
36	PC4	Digital Pin 36	15	9
37	PC5	Digital Pin 37	15	9
38	PC6	Digital Pin 38	15	9
39	PC7	Digital Pin 39	15	9
40	PC8	Digital Pin 40	15	9
41	PC9	Digital Pin 41	15	9
42	PA19	Digital Pin 42	15	9
43	PA20	Digital Pin 43	3	6
44	PC19	Digital Pin 44	15	9
45	PC18	Digital Pin 45	15	9
46	PC17	Digital Pin 46	15	9
47	PC16	Digital Pin 47	15	9
48	PC15	Digital Pin 48	15	9
49	PC14	Digital Pin 49	15	9
50	PC13	Digital Pin 50	15	9
51	PC12	Digital Pin 51	15	9
52	PB21	Digital Pin 52	3	6
53	PB14	Digital Pin 53	15	9
54	PA16	Analog In 0	3	6

55	PA24	Analog In 1	3	6
56	PA23	Analog In 2	3	6
57	PA22	Analog In 3	3	6
58	PA6	Analog In 4	3	6
59	PA4	Analog In 5	3	6
60	PA3	Analog In 6	3	6
61	PA2	Analog In 7	3	6
62	PB17	Analog In 8	3	6
63	PB18	Analog In 9	3	6
64	PB19	Analog In 10	3	6
65	PB20	Analog In 11	3	6
66	PB15	DAC0	3	6
67	PB16	DAC1	3	6
68	PA1	CANRX	3	6
69	PA0	CANTX	15	9
70	PA17	SDA1	3	6
71	PA18	SCL2	15	9
72	PC30	LED "RX"	15	9
73	PA21	LED "TX"	3	6
74	PA25	(MISO)	15	9
75	PA26	(MOSI)	15	9
76	PA27	(SCLK)	15	9
77	PA28	(NPCS0)	15	9
78	PB23	(no conectado)	15	9
USB	PB11	ID	15	9
USB	PB10	VBOF	15	9

Tabla 4.3. Características generales de la tarjeta Arduino Due.

Microcontrolador	AT91SAM3X8E
Voltaje de operación	3.3V
Voltaje de alimentación recomendado	7-12V
Límites de voltajes de alimentación	6-16V
Pines digitales de entrada/salida	54 (of which 12 provide PWM output)
Entradas analógicas	12
Salidas analógicas	2 (DAC)
Salida máxima de corriente en todos los pines	130 mA
Corriente de DC para pines de 3.3 V	800 mA
Corriente de DC para pines de 5 V	800 mA
Memoria Flash	512 KB disponibles para aplicaciones de usuario
SRAM	96 KB (dos bancos: 64KB y 32KB)
Frecuencia de reloj	84 MHz
Largo	101.52 mm
Ancho	53.3 mm

Peso	36 g

Fue necesario descargar el programa Atmel Studio 7 [] para poder llevar a cabo la programación de la tarjeta.

Se considera necesario aclarar en este punto que la memoria de esta tarjeta es la SAM3X, misma que posee 512 KB de memoria flash para código de almacenamiento. El acrónimo SAM (*Software Acquisition Management* en idioma inglés) significa software de gestión de adquisición en español.

Subir programas a la SAM3X resulta diferente que los microcontroladores AVR hallados en otras tarjetas Arduino debido a que la memoria flash necesita ser borrada antes de ser reprogramada.

Las subidas al chip se administran por ROM en la SAM3X, la cual corre únicamente cuando la memoria flas está vacía.

4.5.4 Cámara

La cámara que se utilizó fue la OV7670 [19] ya que esta es totalmente controlable por un microcontrolador. Sus características son proporcionadas por el fabricante, cabe aclarar que esta cámara no posee registro FIFO, por lo cual es necesario implementar uno con el microcontrolador. En la figura 4.23 se muestra una imagen de la cámara.

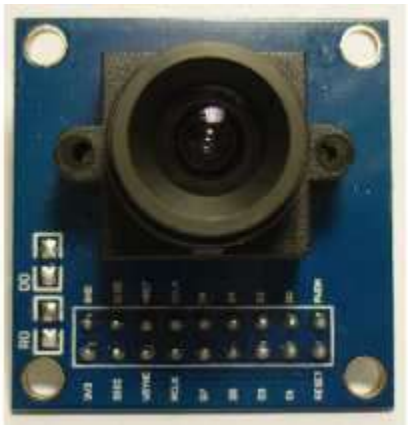


Figura 4.23. Cámara OV7670.

Este módulo posee un sensor de imagen CMOS VGA OV7670, capaz de trabajar a un máximo de 30 fps (cuadros por segundo) a una resolución de 640x480 pixeles (0.3MPx). Es un SoC (sistema en chip) por lo que es capaz de realizar procesamiento de imágenes, como: control de exposición, gamma, balance de blancos, saturación de color, control de tono (hue). Estos parámetros son configurables mediante la interface SCCB (Bus de Control de Cámara Serial). El sensor incluye filtros propios de eliminación de ruido eléctrico, fixed pattern noise (FPN), smearing, blooming, etc.

El diagrama de conexión de modulo se puede ver en la figura 2.24

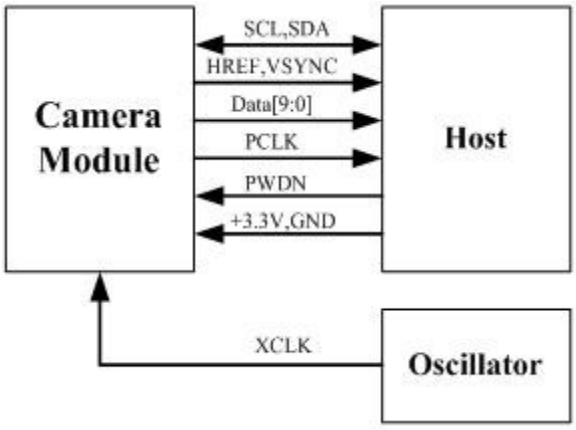


Figura 4.24. diagrama de conexión del módulo OV7670.

El listado de los pines de este módulo se puede ver en la tabla 5.

Tabla 4.4. Listado de pines del módulo OV767

Pin No.	PIN NAME	TYPE	DESCRIPTION
1	VCC	Alimentación	3.3v Power supply
2	GND	Tierra	Power ground
3	SIO_C	Input	Two-Wire Serial Interface Clock
4	SIO_D	Bi-directional	Two-Wire Serial Interface Data I/O
5	VSYNC	Output	Active High: Frame Valid; indicates active frame
6	HREF	Output	Active High: Line/Data Valid; indicates active pixels
7	PCLK	Output	Pixel Clock output from sensor
8	XCLK	Input	Master Clock into Sensor
9	D7	Output	Pixel Data Output 9 (MSB)
10	D6	Output	Pixel Data Output 8
11	D5	Output	Pixel Data Output 7
12	D4	Output	Pixel Data Output 6
13	D3	Output	Pixel Data Output 5
14	D2	Output	Pixel Data Output 4
15	D1	Output	Pixel Data Output 3
16	D0	Output	Pixel Data Output 2 (LSB)
17	RESET	Input	Reset
18	PWDN	Input	Power Down

Tabla 4.5. Características dela cámara

Voltaje de Operación	3.3V DC
Consumo de energía	60mW/15fpsVGAYUV

Corriente Sleep	<20µA
Cámara	OV7670
Transmisión de datos en paralelo	(8 bits)
Lente óptico	1/6”
Ángulo de visión (FOV)	25º
Resolución	640x480 VGA
Sensibilidad	1.3V / (Lux-sec)
Relación Señal-Ruido (SNR)	46 dB
Rango Dinámico	52 dB
Modo de vista	Progresivo
Lente de alta calidad	F1.8/6mm
Máximo refresco de cuadro	30 fps VGA
Exposición electrónica	1 a 510 filas
Cobertura de pixel	3.6µm x 3.6µm
Formatos de Salida	Raw RGB (8 digit), RGB (GRB 4:2:2, RGB 565/555/444), YUV (4:2:2) y YCbCr (4:2:2)
Necesita una frecuencia	de 8MHz

Las principales funciones de la cámara OV7670 son:

- Alta sensibilidad en ambientes de poca iluminación.
- Bajo voltaje, adecuado para aplicaciones portátiles.
- Formatos de imagen: VGA, CIF (hasta CIF 40x30).
- VarioPixel, usado para sub-samplig.
- Funciones automáticas: Control automático de exposición (AEC), Control Automático de ganancia (AGC), Balance automático de blancos (AWB), Filtro pasa banda automatico (ABF), Calibración automática de nivel de negro (ABLC).

- Parámetros configurables: Saturación de color, Tono (hue), gamma, sharpness (mejora de bordes) y anto-blooming.
- ISP incluye reducción de ruido y corrección de defectos.
- Compatible con luz LED y strobe.
- Soporta escalamiento.
- Corrección de sombra de lente.
- Auto detección de Flicker (50/60 Hz)
- Autoajustes de nivel de saturación (UV)

4.5.5 Comunicación con la cámara

La cámara OV7670 posee su propia interfaz de comunicación serial la cual el fabricante denomina como Serial Camera Control Bus (SCCB) [20].

En la figura 4.25 se puede ver la implantación que nos permite conectar un dispositivo SCCB maestro con un dispositivo esclavo.

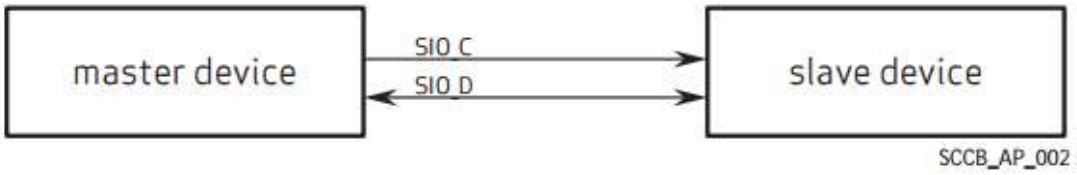


Figura 4.25. Diagrama a bloques para la comunicación con dos cables.

La implementación con dos cables requiere uno de los dos métodos de control en orden para facilitar la comunicación SCCB.

1. En primera instancia, el dispositivo maestro debe ser capaz de soportar y mantener la línea de datos del bus en modo tri-estado.
2. El método alternativo será cuando el maestro no pueda mantener una condición de tri-estado de la línea de datos este método debe manejar la línea o en alto o en bajo y notar la transición para afirmar las comunicaciones con el dispositivo esclavo.

La señal SIO_C es una señal de control activa alta y unidireccional que debe ser manejada por el dispositivo maestro. Esta indica cada bit trasmitido. El maestro debe llevar a un uno lógico cuando el bus este desocupado. Una trasmisión de datos empieza cuando SIO_C es llevado a un cero lógico. Un uno lógico de SIO_C durante la trasmisión de datos indica solo un bit trasmitido. Así, SIO_D pude cambiar solo cuando SIO_C es llevado a cero.

La señal SIO_D es una señal de datos bidireccional que puede ser manejado ya sea por el dispositivo maestro o el dispositivo esclavo. Esta señal permanece flotando o en tri-estado cuando se encuentra desocupada. El manejo de esta señal es responsabilidad tanto del dispositivo maestro como del esclavo, en este orden para evitar propagaciones desconocidas.

Existe otra señal la cual está definida como SCCB_E que es la habilitación del dispositivo esclavo, sin embargo, en el módulo OV7670 esta señal siempre está activa y por lo tanto el modulo siempre está habilitado, esto implica que solo se puede conectar un dispositivo al bus.

En la figura 4.26 se muestra el diagrama de tiempo para una trasmisión de datos. Para que se lleve a cabo un inicio de trasmisión el dispositivo maestro debe llevar a un uno lógico, posteriormente se tiene que llevar a un cero lógico la señal SIO_D. Después se pone a cero lógico la señal SIO_C. Con esto se genera la condición de trasmisión de datos. Esto se puede ver en la figura 4.27.

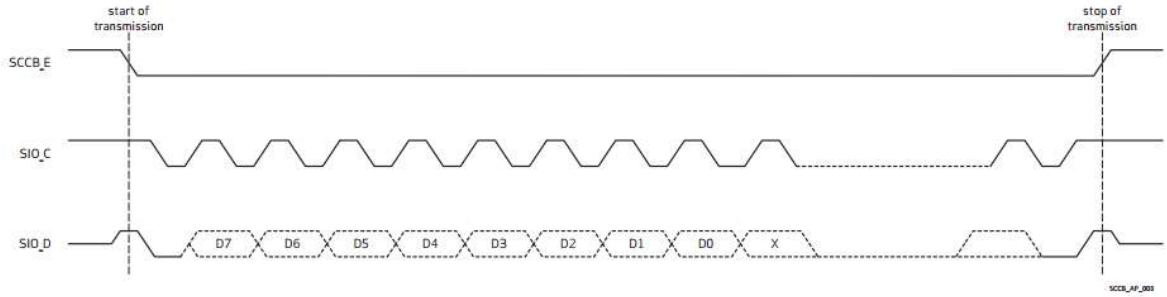


Figura 4.26. Diagrama de tiempo de trasmisión de datos

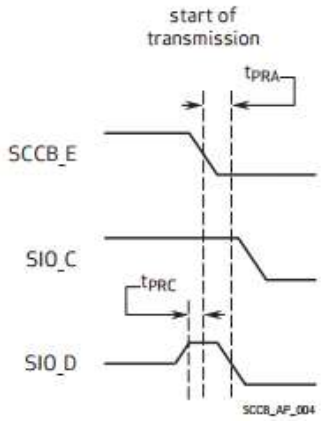


Figura 4.27. Inicio de la trasmisión de datos.

Una condición de paro de trasmisión se genera cuando se lleva a un uno lógico la señal SIO_C y posteriormente se lleva a uno la señal SIO_D. Esto se puede revisar en la figura 4.28.

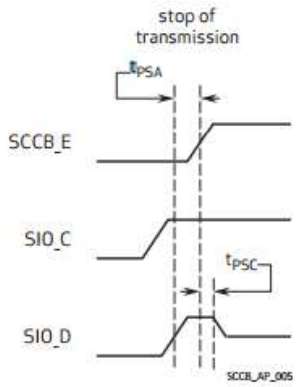


Figura 4.28. Condición de paro.

Una ventaja que tiene el protocolo SCCB es que es compatible con el protocolo i2c, el cual se encuentra en casi todos los microcontroladores, y es con el cual se hizo la interfaz de comunicación.

Obteniendo la imagen de la cámara. La cámara manda la información de los pixeles por medio de los 8 pines de datos (dato en paralelo) y el pin PCLK el cual es la señal de reloj que indica cuando se tiene un pixel. El diagrama de tiempo se puede ver en la figura.

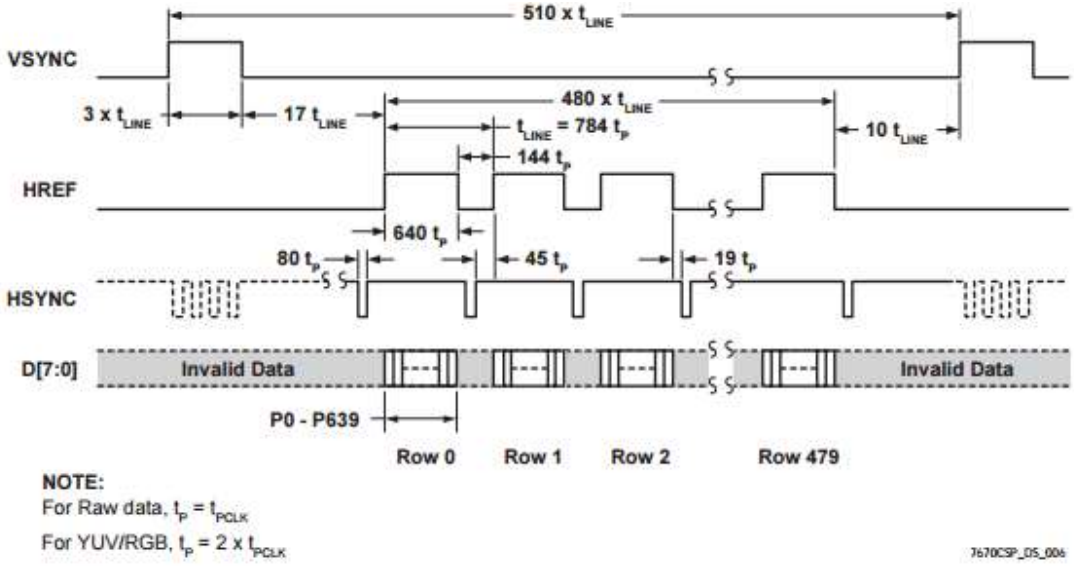


Figura 4.29. Diagrama de tiempo de una imagen.

Donde VSYNC es la señal de sincronización vertical, esto significa que esta señal indica cuando se inicia una imagen y cuando termina, la señal HREF indica el principio y fin de una línea de la imagen y D[7:0] son los datos de los pixeles. Todas estas señales están sincronizadas por el pin de PCLK, que es el reloj de salida del módulo.

4.5.6 Implementación del firmware

Para implementar el código del Arduino primero se tuvo que configurar el dispositivo esto con ayuda de la hoja de datos del ARM Cortex de Atmel [21]. En la figura 4.30 se puede ver como se conectó el módulo OV7670 con el Arduino Due.

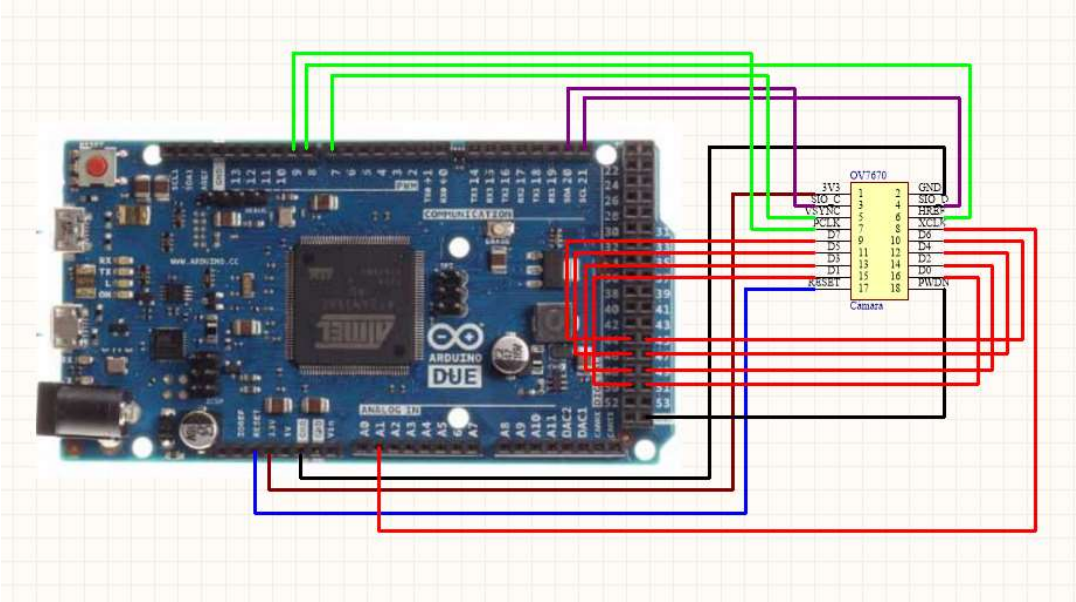


Figura 4.30. Conexión del Arduino Due con el Modulo OV7670.

En la figura 4.31 se muestra el diagrama de flujo del programa principal. Como se puede observar el programa necesita de interrupciones para llevar a cabo su tarea. Una de estas interrupciones es la del puerto UART la cual se activa cada vez que le llega un dato, si el dato es igual a "s" (en formato ascii) la subrutina de interrupción igualara la variable Estado a s1 (figura 4.32), y en este estado se habilitara la interrupción por flancos del puerto C que es el puerto donde se conectan todas las señales de datos de la cámara (datos de la imagen).

En la interrupción del puerto C se habilitan 2 pines de interrupción que corresponden a las señales VSYNC y PCLK, que como ya se mencionó son las señales de sincronismo vertical, de sincronismo horizontal y el reloj de los pixeles. Para el pin de VSYNC se habilita la interrupción cada que hay un flanco de subida y un flanco de bajada, esto es ya que el flanco de bajada indica el inicio de una imagen, mientras que el flanco de subida indica un fin de imagen. La interrupción del PCLK se configura para que ocurra cada flanco de subida ya que en los flancos de subida es cuando los datos del pixel están estables. El diagrama a flujo de la interrupción se puede ver en las figuras 4.33 y 4.34. La interrupción se encarga de almacenar la imagen en el registro FIFO. De esta manera se puede almacenar la imagen de una forma rápida, esto quiere decir que como van llegando los datos estos se van almacenando y posteriormente se podrá manejar los datos sin problemas. Al final cuando ya se tiene la imagen completa la interrupción del puerto C se deshabilita, ya que, el módulo OV7670 envía información indiscriminadamente, así se discrimina la información.

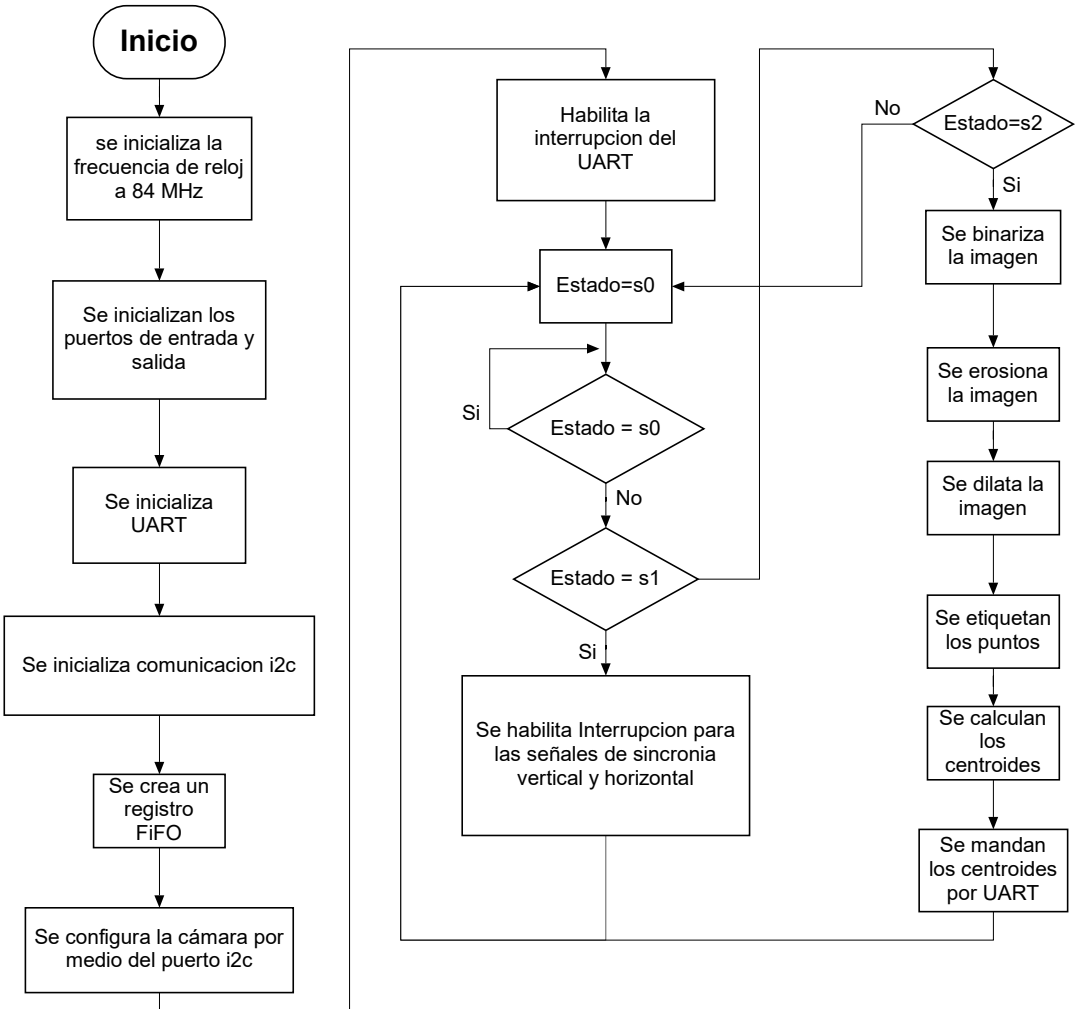


Figura 4.31. Diagrama de flujo del programa principal para el Arduino Due.

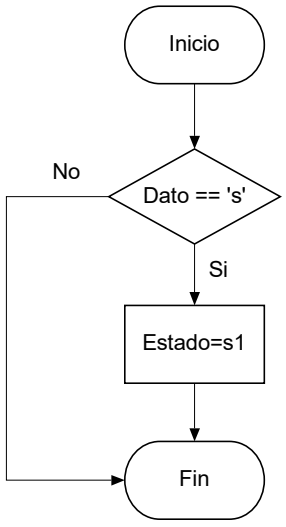


Figura 4.32. Interrupción de UART.

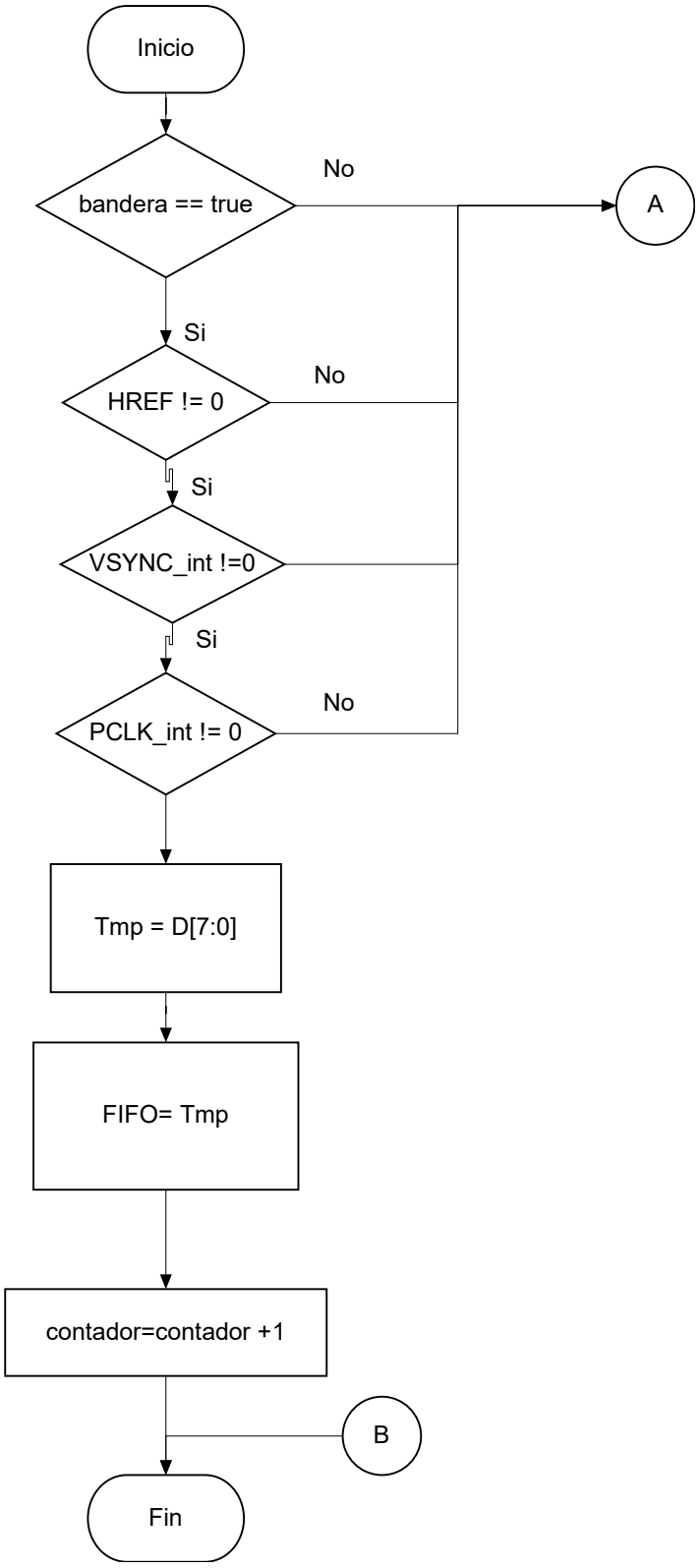


Figura 4.33. Diagrama de flujo de interrupción de las señales VSYNC y PCLK.

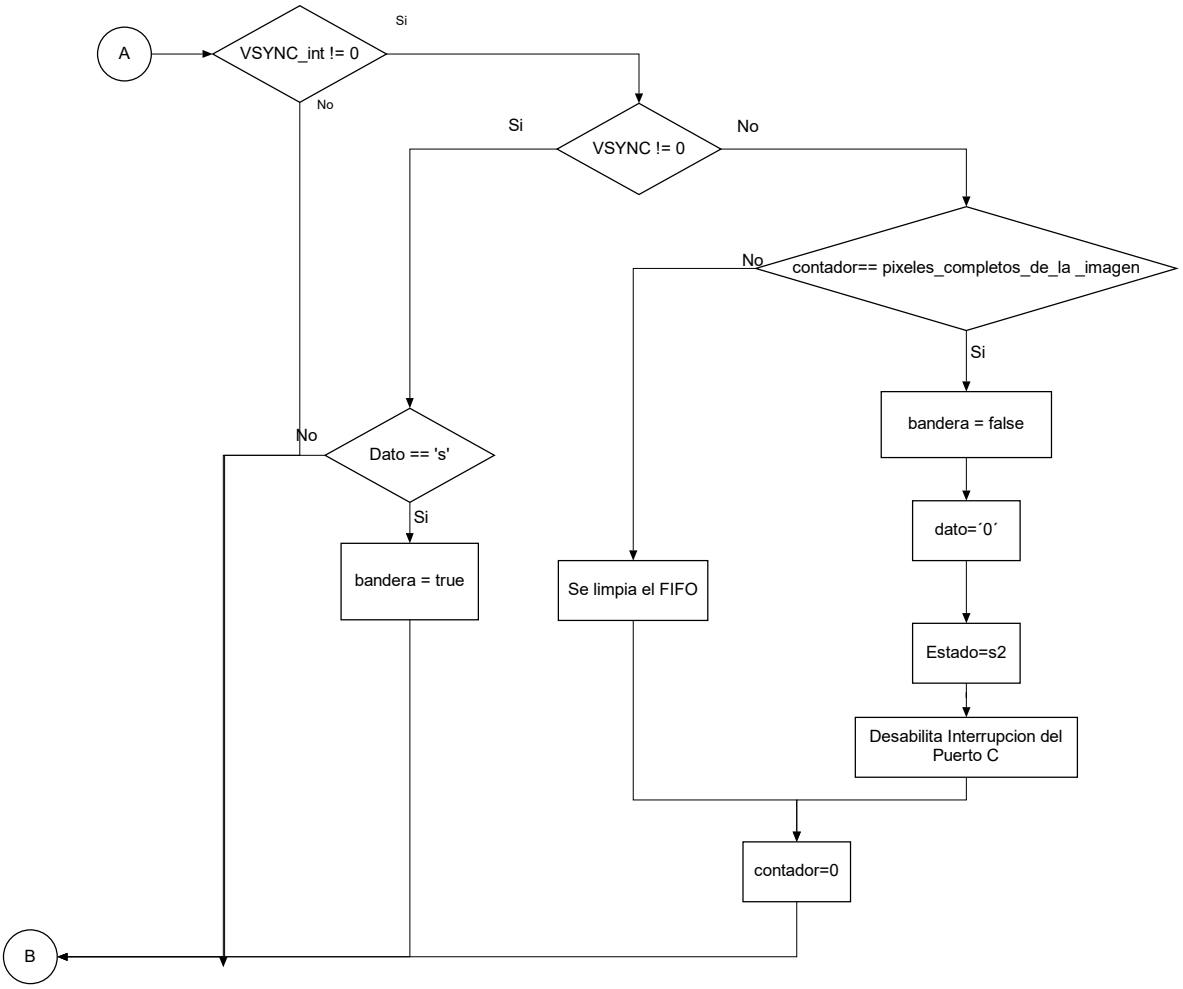


Figura 4.34 Diagrama de flujo de interrupción de las señales VSYNC y PCLK (continuación).

Una vez que se tiene la imagen completa la subrutina de interrupción del HSYNC pone la variable Estado a s2 y con esto se realiza todo el procesamiento de imagen para encontrar los centroides.

El primer paso es binarizar la imagen esto se logra haciendo leyendo byte por byte el FIFO que contiene la imagen en escala de grises, y como se van extrayendo los bytes se van comparando con el umbral para decidir si será un uno o un cero. El diagrama de flujo se presenta en la figura 4.35. Posteriormente se tiene que dilatar la imagen para esto se hace un barrido de la imagen y si se encuentra un pixel en uno, se ponen en uno los pixeles que están alrededor de él. El diagrama de flujo del proceso de dilatación se puede ver en la figura 4.36.

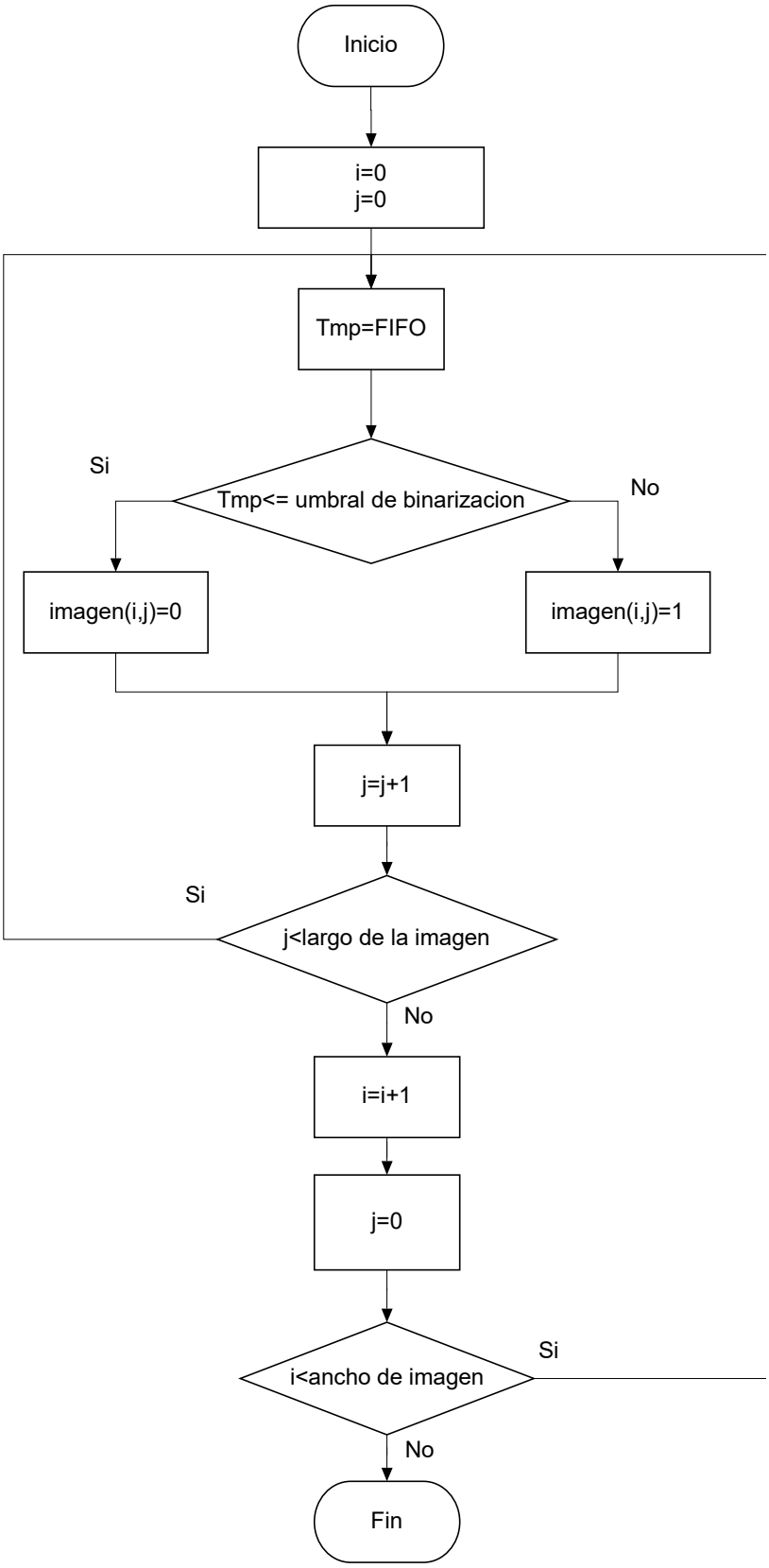


Figura 4.35. Diagrama de flujo del proceso de binarización.

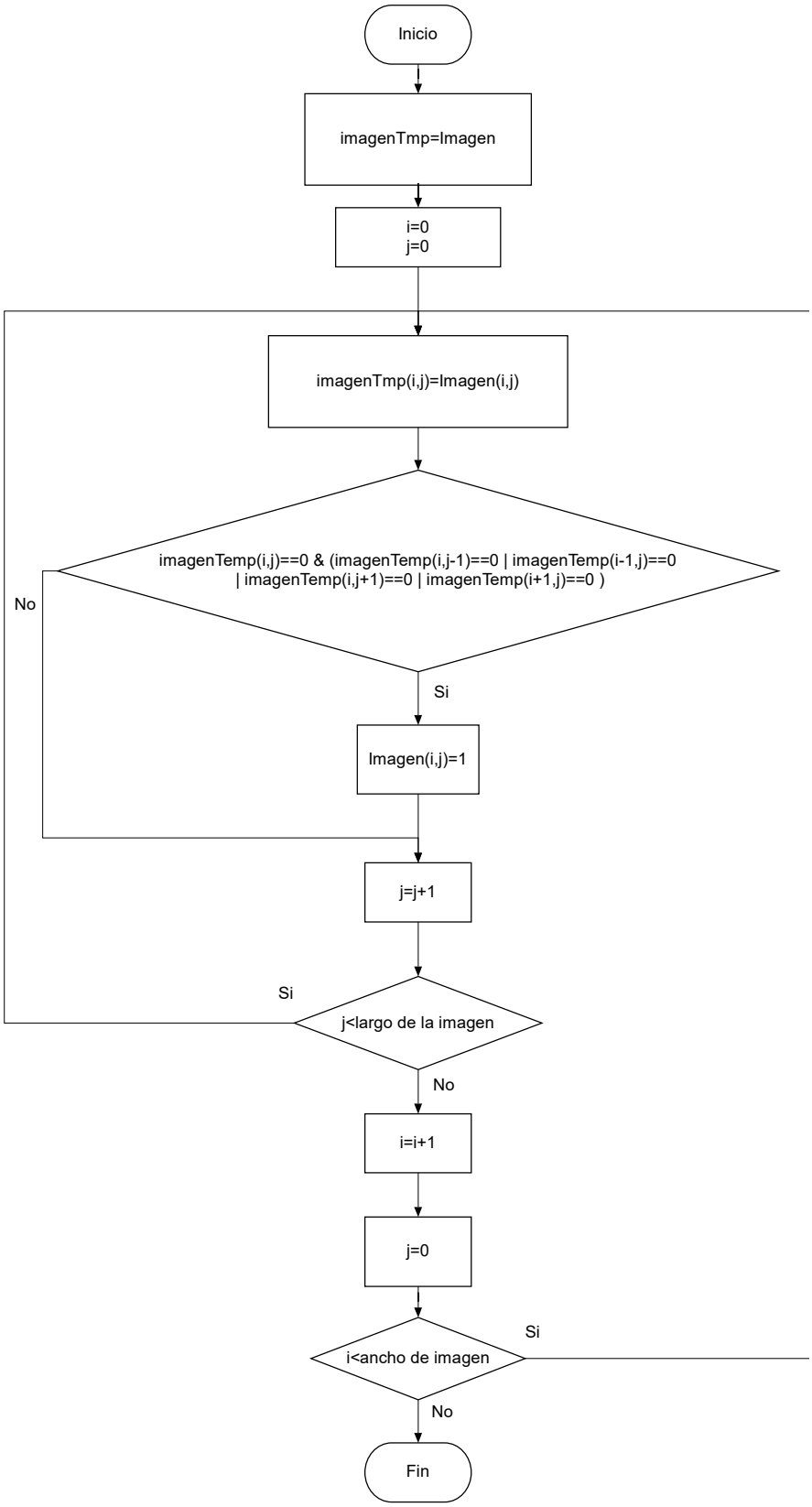


Figura 4.36. Diagrama de flujo del proceso de dilatación.

Una vez que se realizó la dilatación se procede a hacer la erosión de la imagen esto con el fin de regresar las figuras a su estado original. La figura 4.37 muestra el diagrama de flujo para el proceso de erosión.

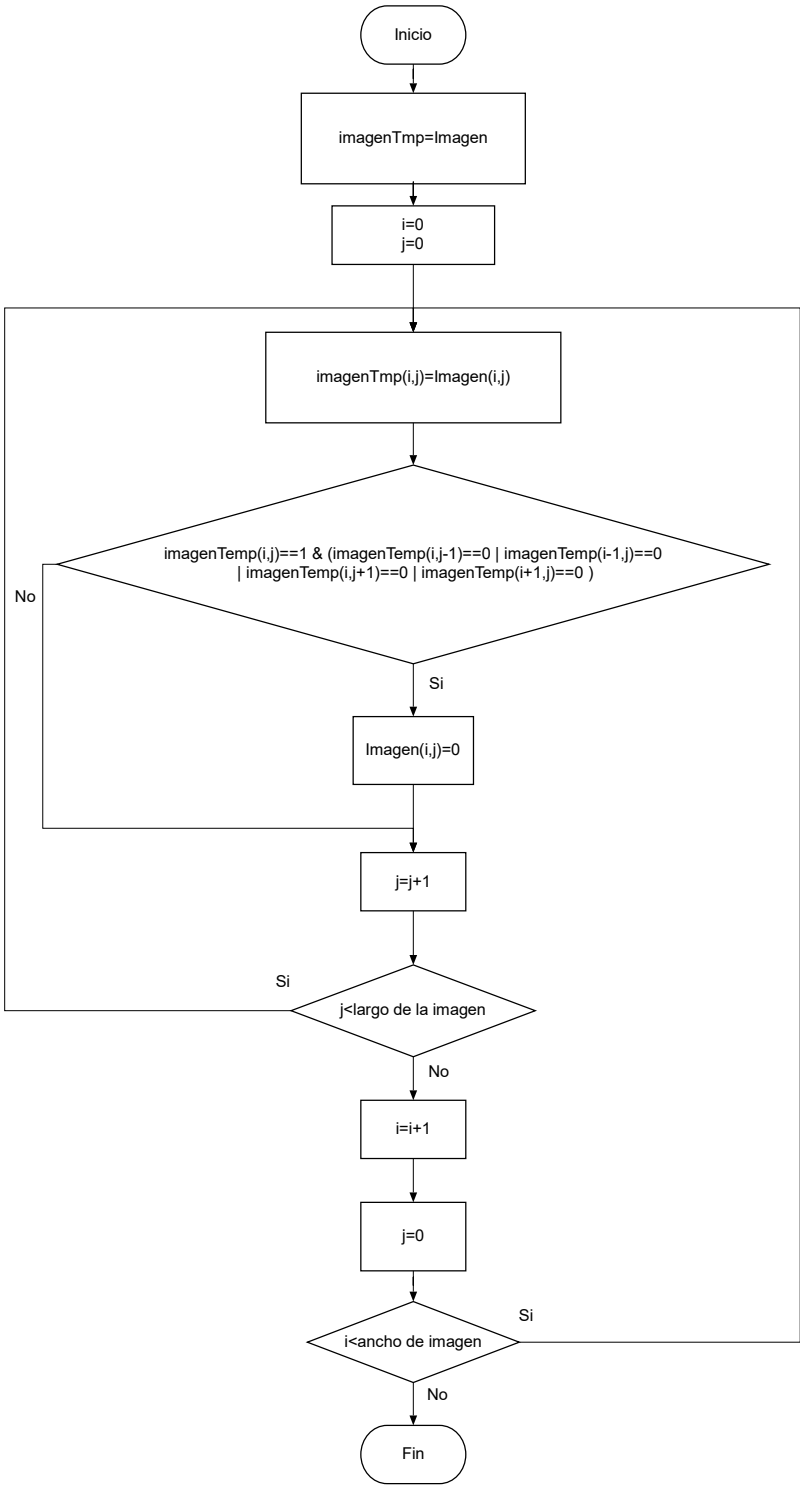


Figura 4.37. Diagrama de flujo del proceso de erosión.

Una vez que se tiene la imagen limpia se hace el etiquetado para determinar las áreas en pixeles de los puntos en la imagen. El diagrama de flujo se muestra en las figuras 4.38, 4.39y4.40.

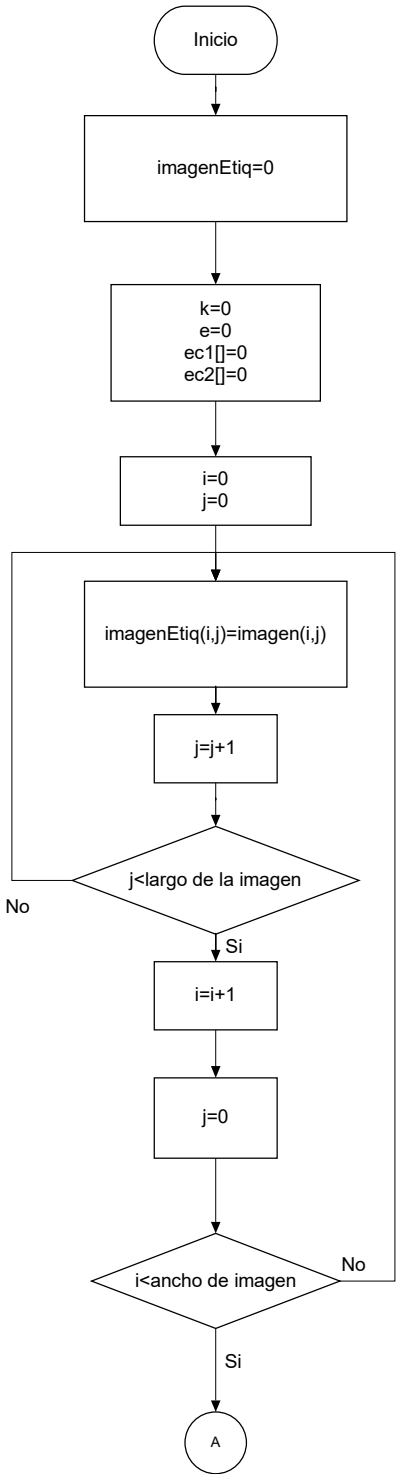


Figura 4.38. Diagrama de flujo del proceso de etiquetado.

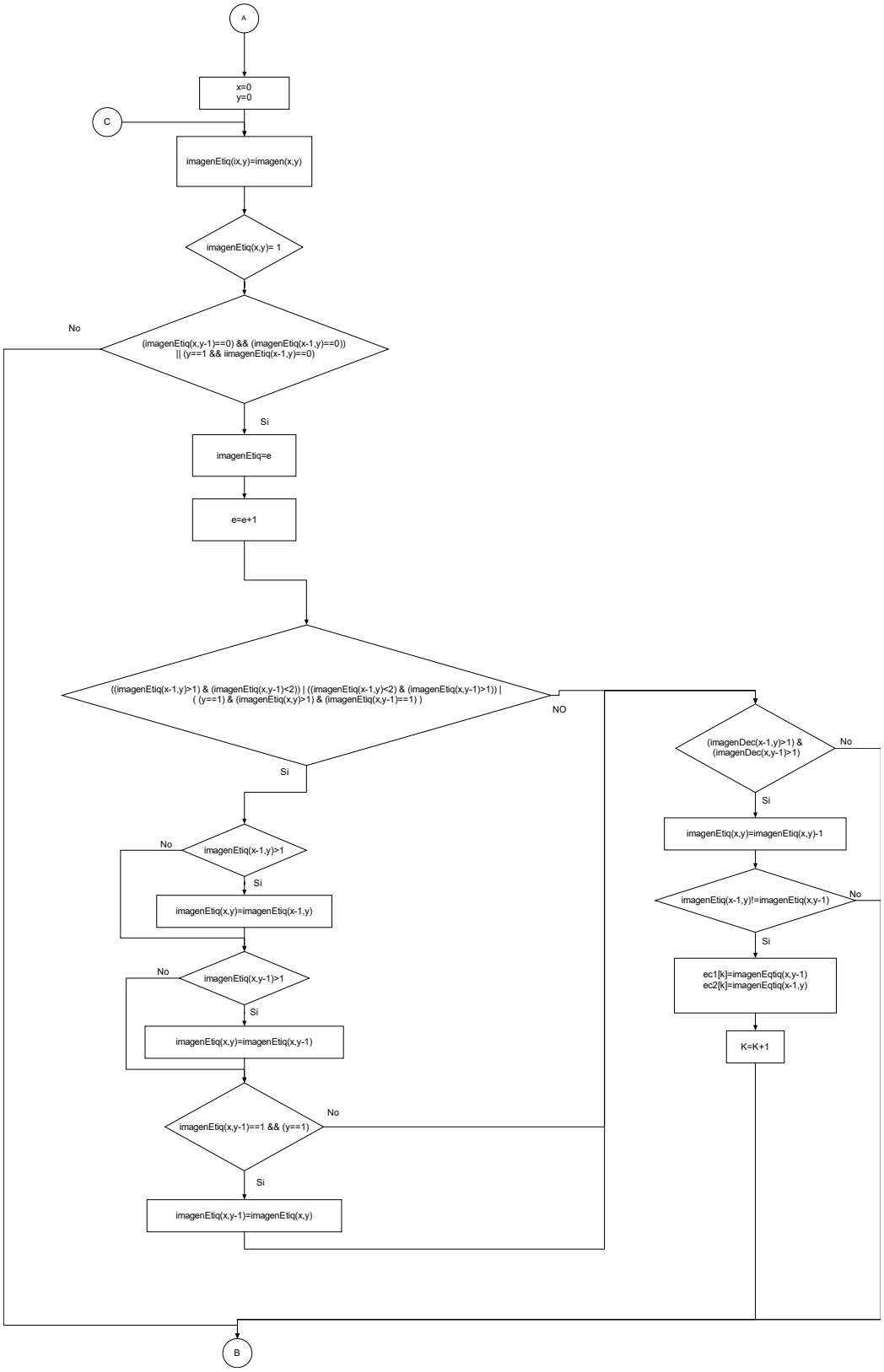


Figura 4.39 Diagrama de flujo del proceso de etiquetado. (continuación)

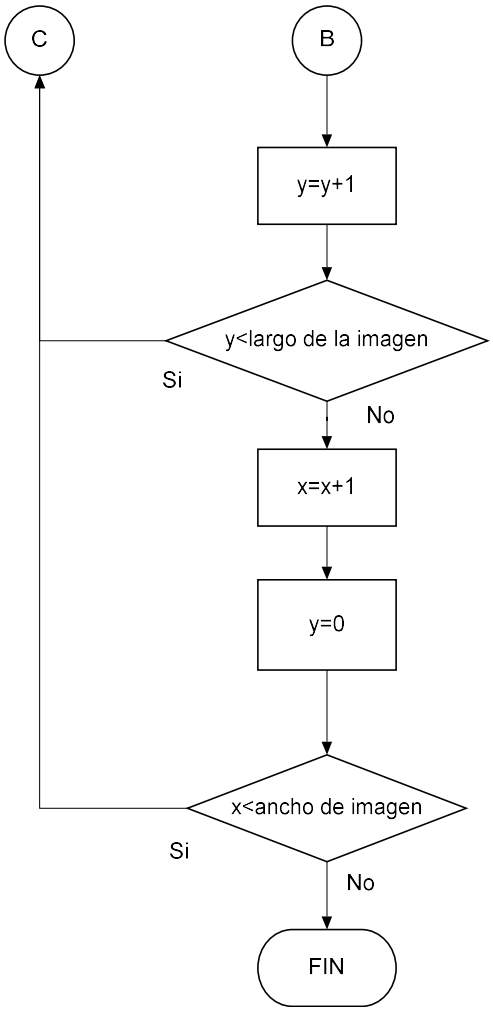


Figura 4.40. Diagrama de flujo del proceso de etiquetado. (continuación)

Al hacer el etiquetado de figuras se tiene que eliminar las colisiones, estas suceden cuando dentro de la imagen una figura queda etiquetada con dos o más etiquetas, pero estas etiquetas como ya se mencionó pertenecen a la misma figura. Para eliminar estos errores se tiene que agregar un algoritmo de colisiones, este algoritmo pertenece al mismo proceso de etiquetado. En la figura 4.41 se puede ver el diagrama de flujo algoritmo para la eliminación de colisiones.

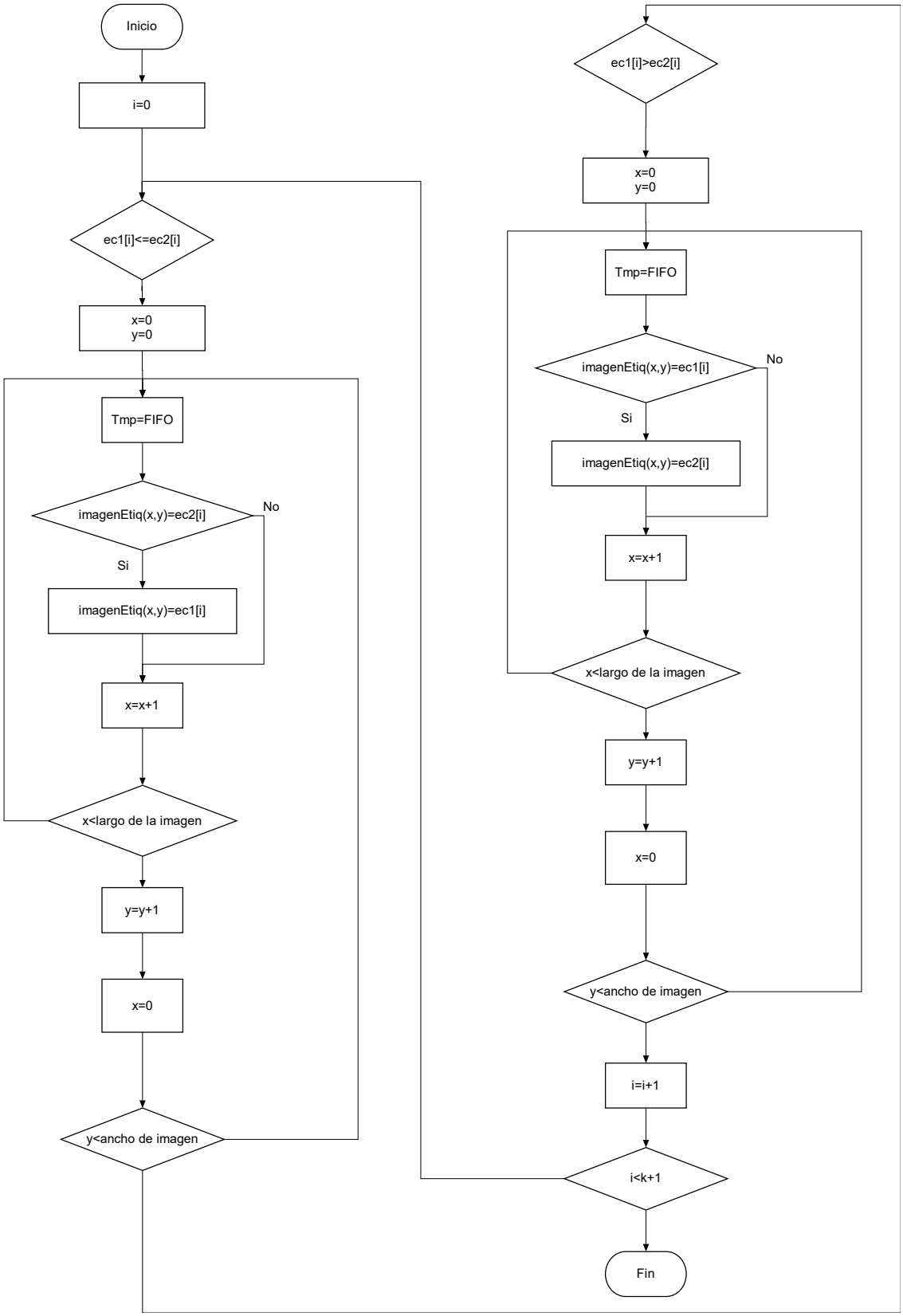


Figura 4.41. Diagrama de flujo para la eliminación de colisiones.

Por último, se presenta el algoritmo para el cálculo de los centroides el cual necesita la imagen con etiquetas para saber las áreas dentro de la imagen. Los diagramas a flujo para el cálculo de los centroides se puede ver las figuras 4.42, 4.43 y4.44

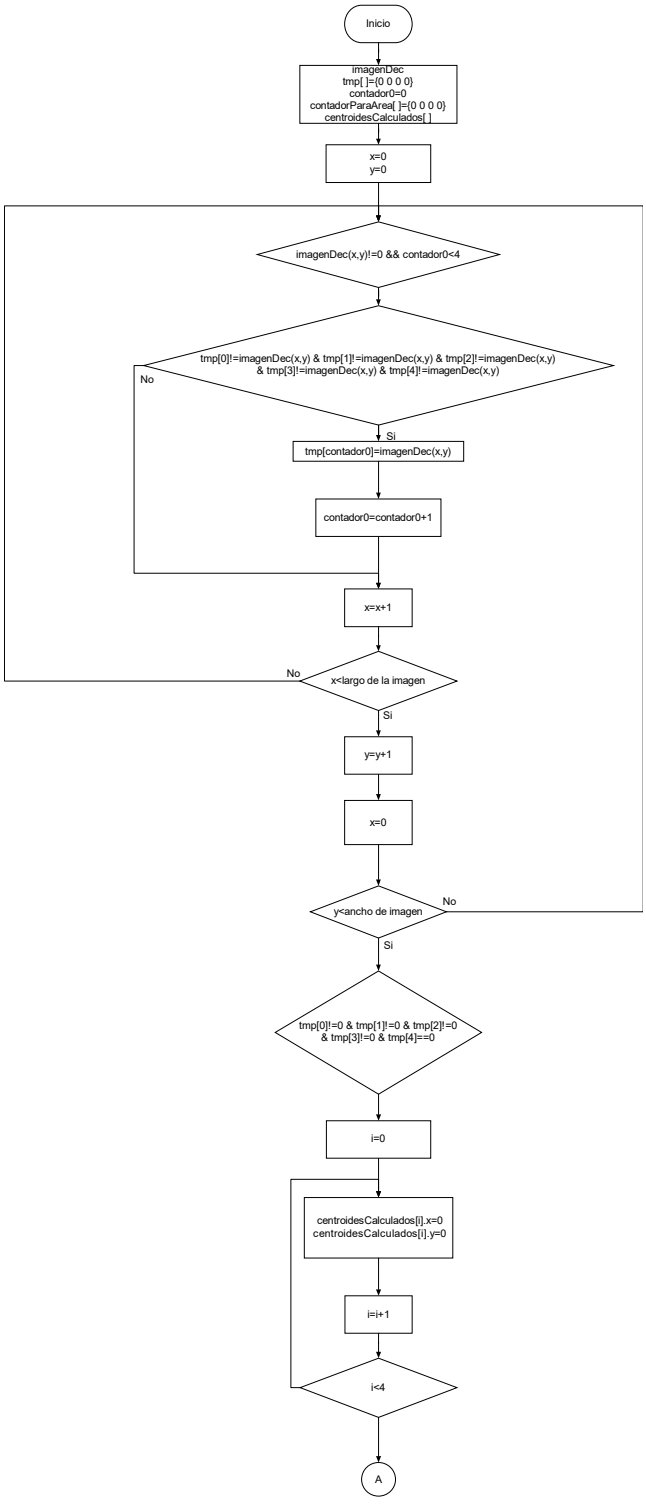


Figura 4.42. Diagrama de flujo para el cálculo de los centroides.

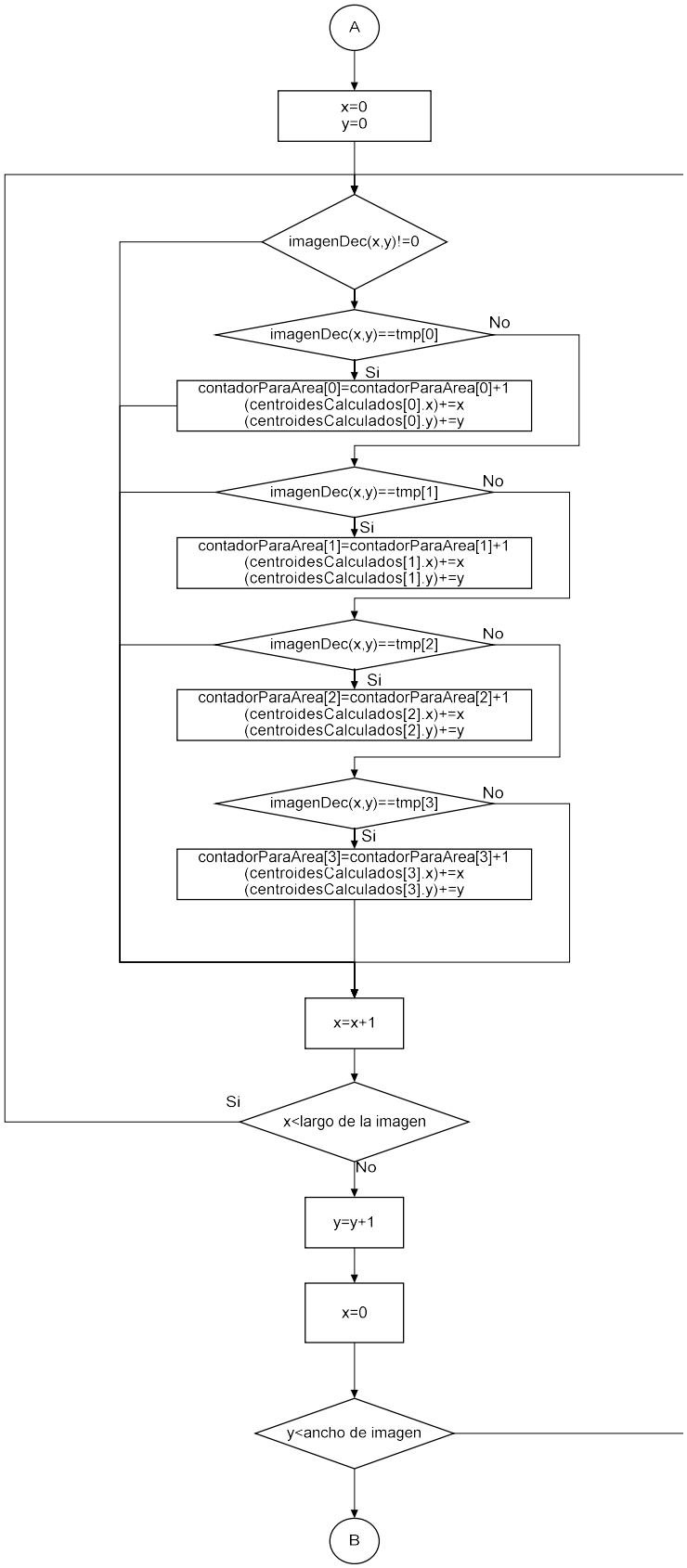


Figura 4.43. Diagrama de flujo para el cálculo de los centroides. (continuación)

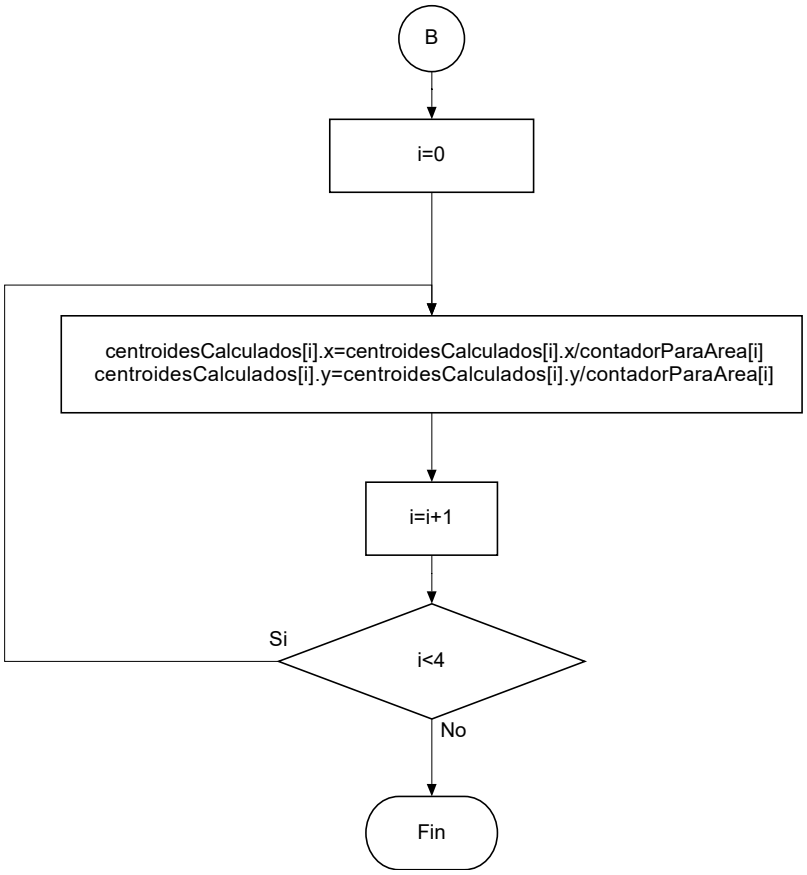


Figura 4.44. Diagrama de flujo para el cálculo de los centroides. (continuación)

En este capítulo se explicó desde el desarrollo de la teoría que respalda los algoritmos que se implementaron tanto en Matlab como en la plataforma Arduino.

Como se pudo observar el algoritmo que se desarrolló para Matlab puede correr en tiempo real y no consume demasiados recursos de la computadora. Sin embargo, el microcontrolador Arduino aunque tiene una frecuencia de 84MHz el procesamiento de las imágenes es demasiado lento y por lo tanto no es buena opción para la implementación de este tipo de tareas.

5 Resultados

En este capítulo se presentan los resultados obtenidos de la implementación del algoritmo desarrollado. Estos resultados se obtuvieron al aplicar los algoritmos en un sistema que hemos nombrado mesa experimental la cual se presenta en la figura 5.1.

La disposición de los elementos que integran este sistema experimental, se encuentran ubicados de tal forma que permita aplicar el algoritmo en función de las características que se desean obtener. La cámara se mantiene fija tanto en su ubicación como en su orientación, mientras que el marco de led's infrarrojos puede ser manipulado manualmente, con el fin de cambiar cualquiera de sus 6 gdl.

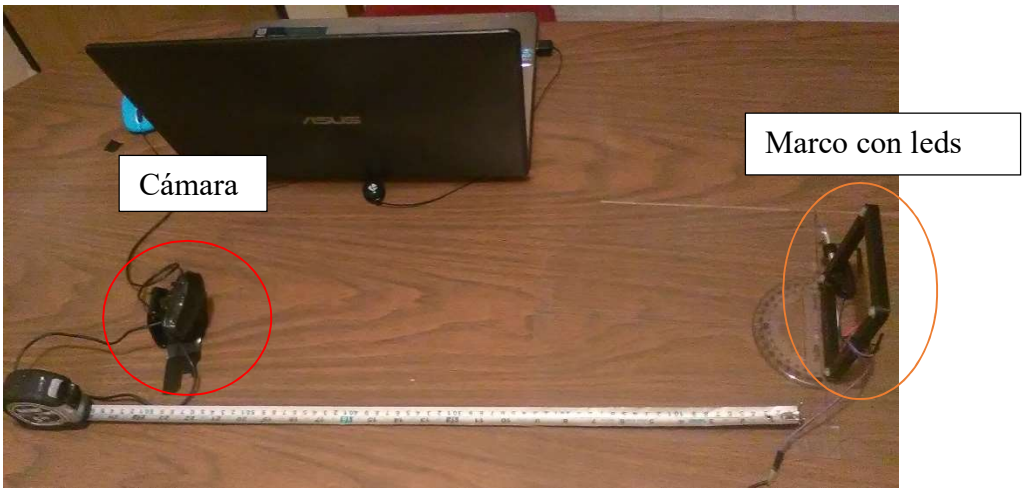


Figura 5.1 Disposición del marco con los led´s infrarrojos (círculo naranja) con respecto a la cámara (círculo rojo).

5.1 Resultados del algoritmo

En la figura 5.2 se muestra la interfaz realizada en Matlab, como se puede observar en la imagen de la izquierda se despliegan los valores tanto de los ángulos como de las traslaciones con respecto del centro del plano de la imagen, los valores se despliegan en grados para los ángulos y milímetros para las distancias o traslaciones. A la derecha se puede ver las imágenes de los puntos como los capta la cámara (escala de grises), a la derecha se pueden ver los puntos después del preprocesamiento y en la parte inferior se puede ver la proyección de un cubo, el cual representa los 4 puntos que pertenecen al plano físico, también cuatro puntos proyectados a 100mm de distancia del plano Z, estos puntos son virtuales, esto quiere decir que solo se dieron valor a estos puntos y el software los proyecta como si estuvieran realmente ahí.

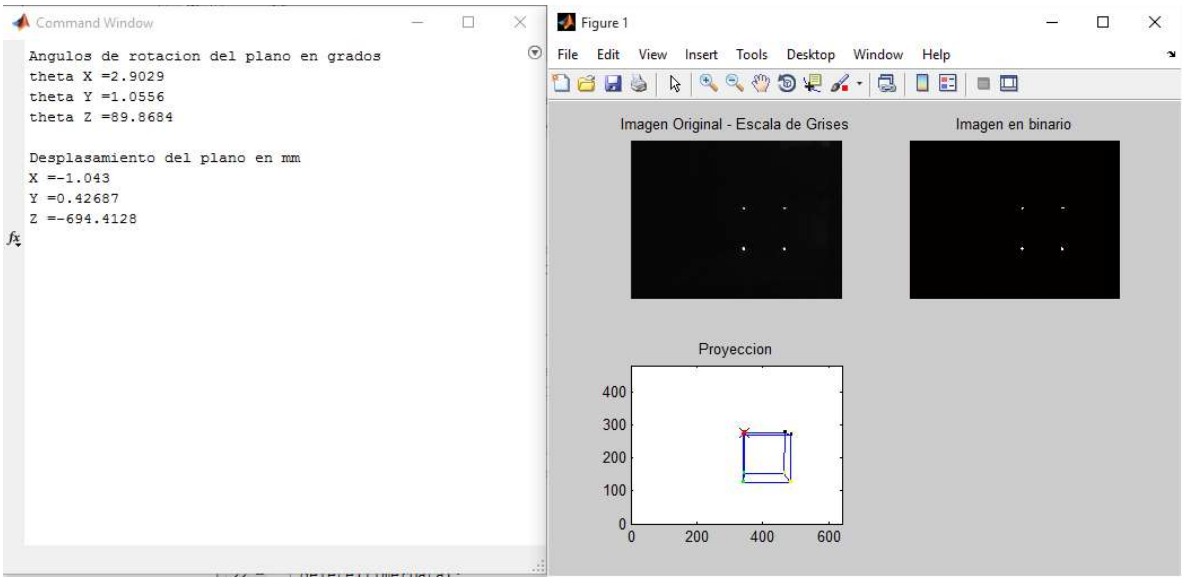


Figura 5.2. Izquierda: Medidas hechas por el algoritmo. Derecha: imágenes y proyección de el algoritmo.

Para conocer la referencia del plano físico en la imagen, se marca con una x un punto en el cubo para que visualmente se tenga idea donde se encuentra, esta x representa la coordenada (0,0) (figura5.3).

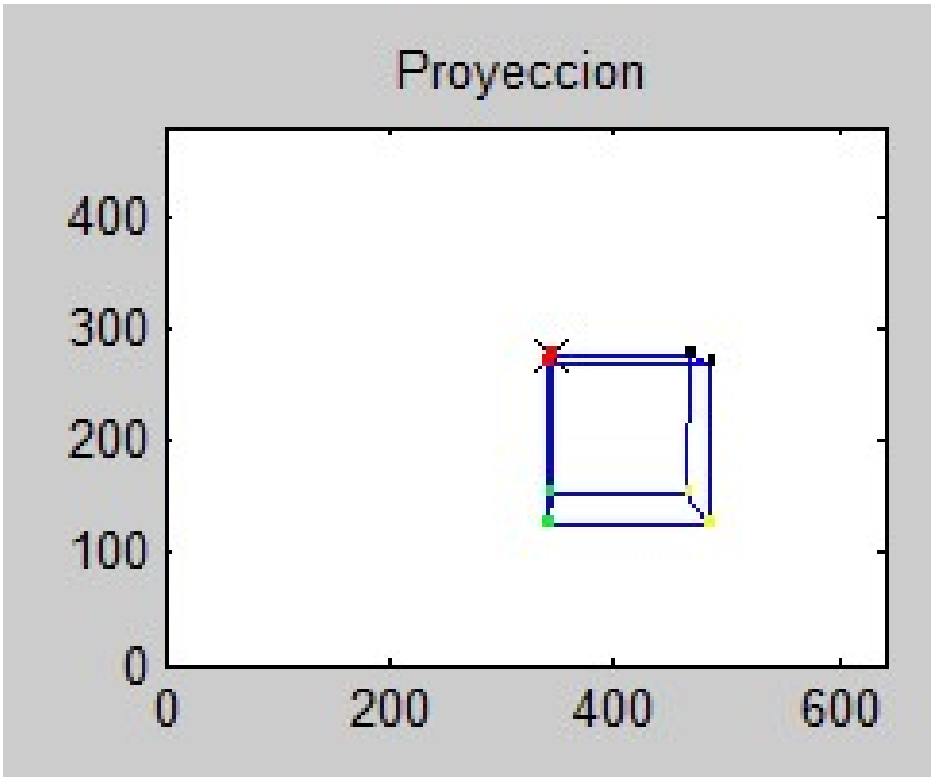


Figura 5.3. Plano de la imagen. El origen se marca con una X.

Para establecer una referencia y esta sirva para hacer las medidas y comparaciones se escogió un punto, y este se fijó, este punto tiene las medidas que aparecen en la figura 5.4, las cuales fueron verificadas con instrumentos de medición.

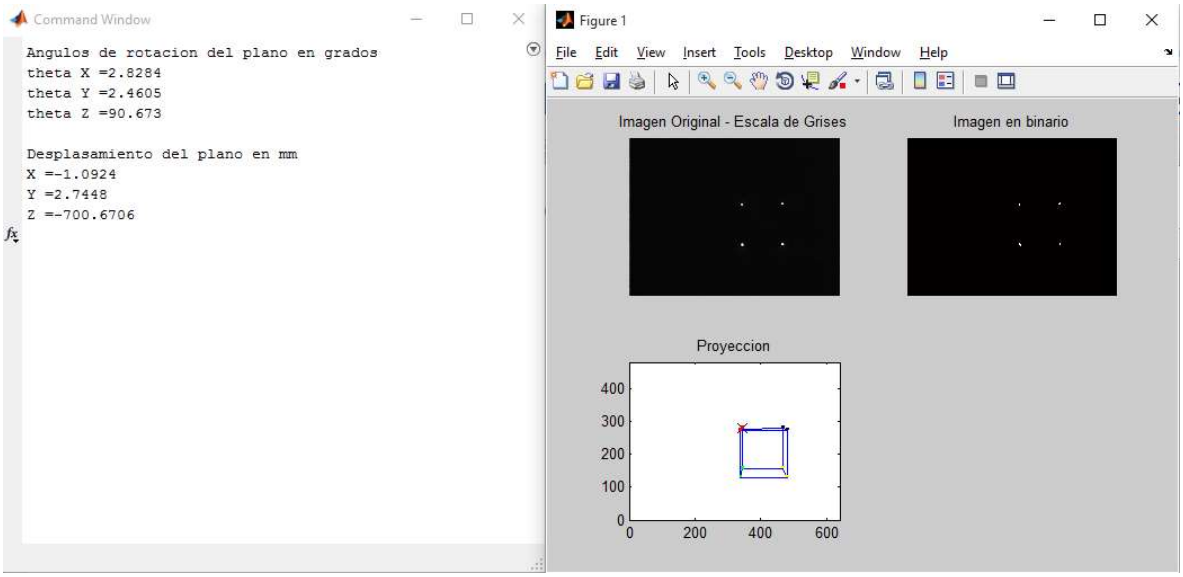


Figura 5.4. Proceso para el establecimiento de la referencia en el plano de la imagen.

La medida de la distancia Z que se propuso fue 700mm. En la figura 5.4 se puede ver encerrado en un círculo rojo la referencia que se encuentra a 700mm de distancia de la cámara, mientras que en la figura 5.5 se puede apreciar donde se comienza a medir la distancia. Mientras que las medidas de X y Y se ajustaron a 0mm. Los ángulos se ubicaron como cero, alineando el marco con la cámara (figuras 5.6, 5.7, 5.8). De esta forma se consiguió la referencia para realizar las siguientes pruebas experimentales.

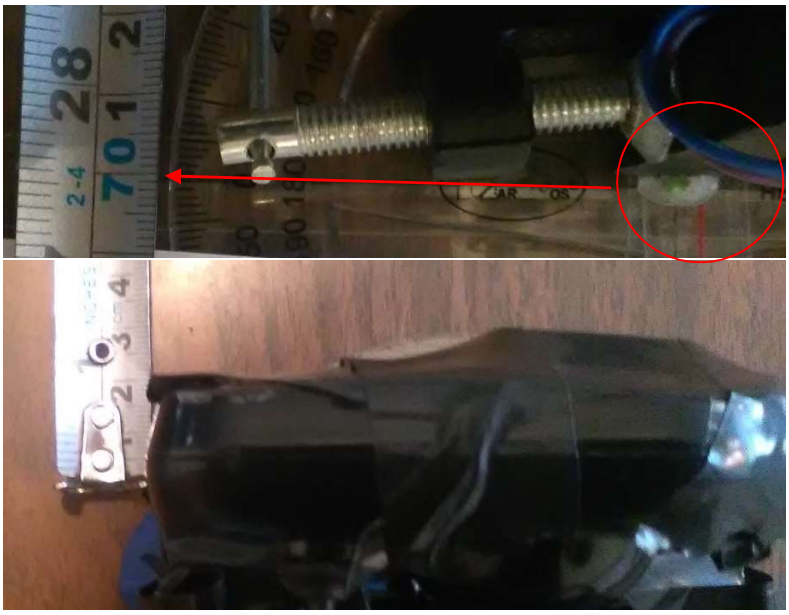


Figura 5.5. Distancia de la cámara a la referencia.

Figura 5.6. Alineación del ángulo θ_x.

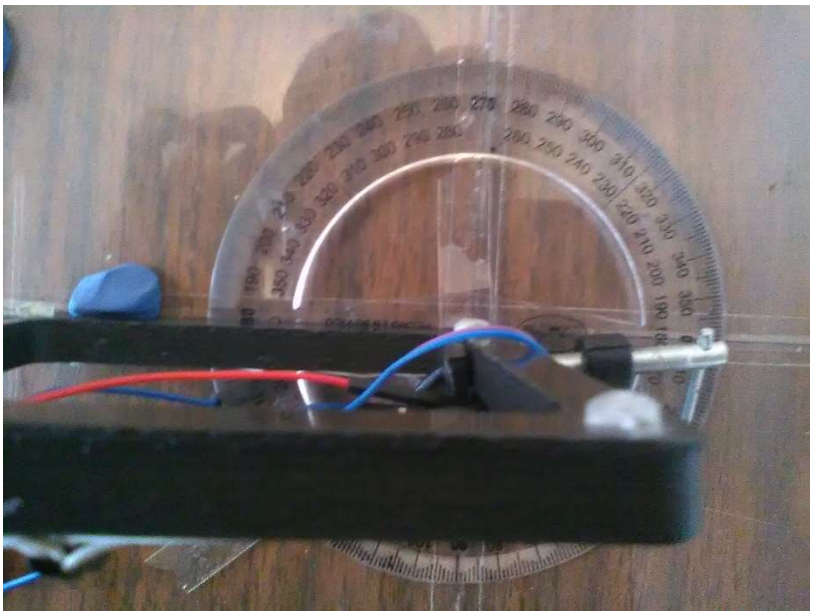


Figura 5.7. Alineación de ángulo θ_y.

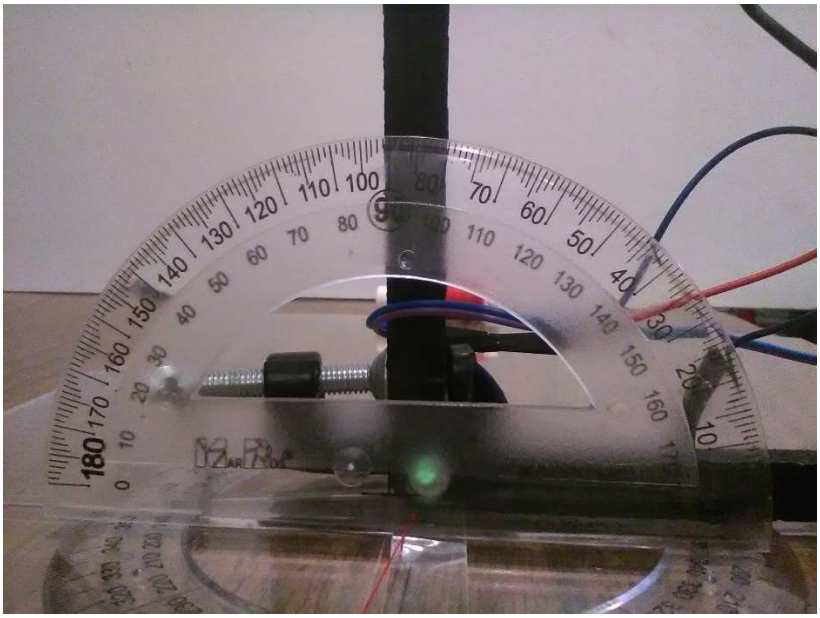


Figura 5.8. Alineación del ángulo θ_z.

Para hacer la primera comparación se cambiará la distancia del eje Z y esta se comparará con la medida realizada físicamente, también se comparará con las medidas de la referencia que se tomaron respecto a la cámara.

En la figura 5.9 se muestra la medición hecha por el software. Como se puede apreciar comprándolas con las medidas de referencia los parámetros que no son la distancia en Z siguen aproximados. En la figura 5.10 se puede ver el valor medido físicamente.

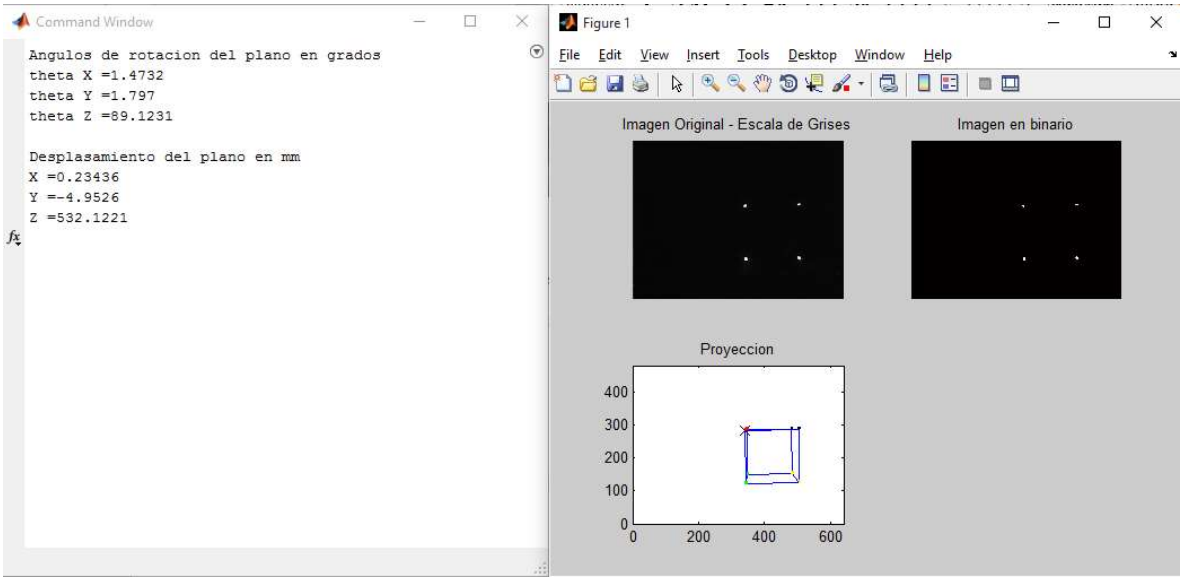


Figura 5.9. Distancia medida por el algoritmo.

Figura 5.10. Medida realizada físicamente.

La comparación entre la medida hecha por el algoritmo y hecha físicamente es muy aproximada. La comparación de las medidas se puede ver en la siguiente tabla.

Medida referencia	**Medida algoritmo**	**Medida física**
Z=2.4748mm	Z=532.1221mm	Z=530.00mm

Ahora se variará la distancia en el eje X, moviendo el marco 50mm, en la figura 5.11se muestra la medición echa por el algoritmo.

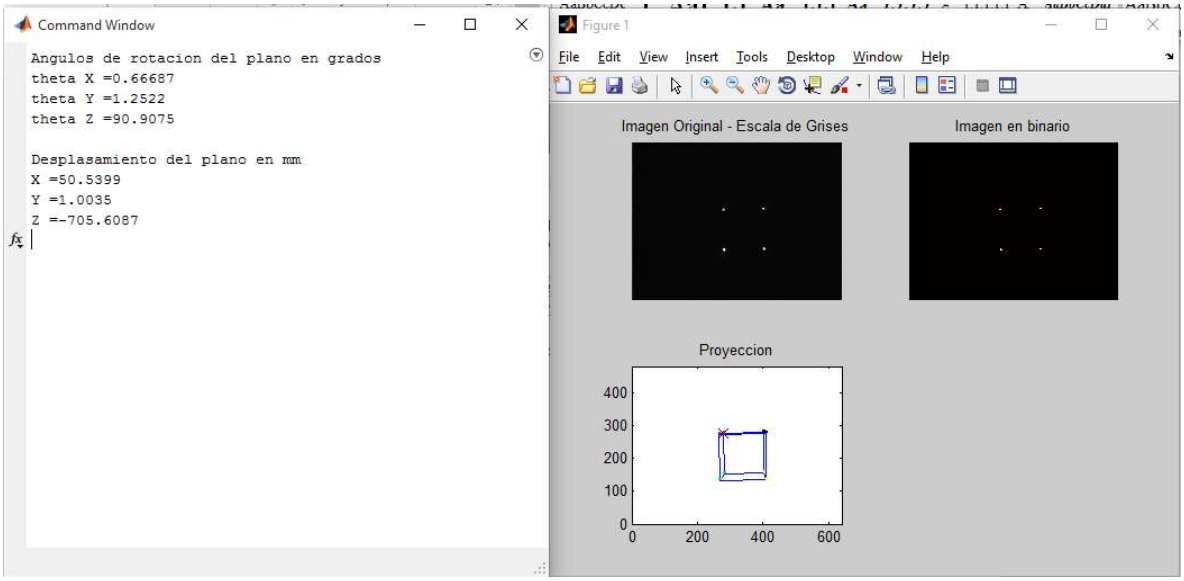


Figura 5.11. Medida del algoritmo moviendo el marco 50mm en dirección del eje X.

La medida física se puede observar en la figura 5.12: En esta figura se marca el desplazamiento que se hizo con una línea amarilla.

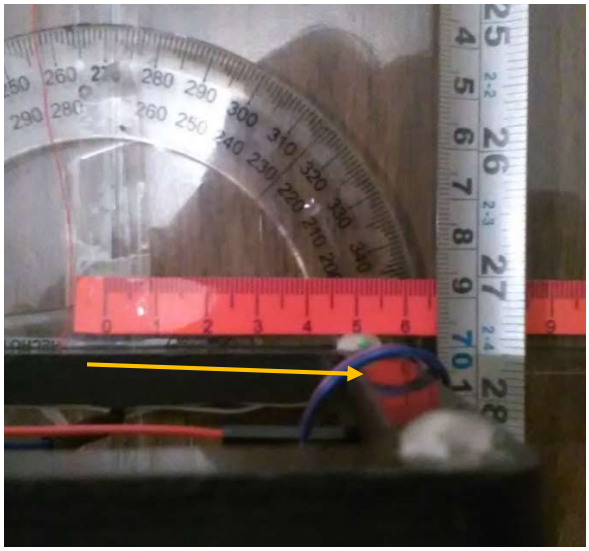


Figura 5.12. Medida realizada físicamente.

La comparación entre la medida hecha por el algoritmo y hecha físicamente puede verse a continuación.

Medida referencia	Medida algoritmo	Medida física
X=1.0924mm	X=50.5399	X=50.00mm

Para la medida del desplazamiento en eje Y el algoritmo dio como resultado lo que se muestra en la figura 5.13.

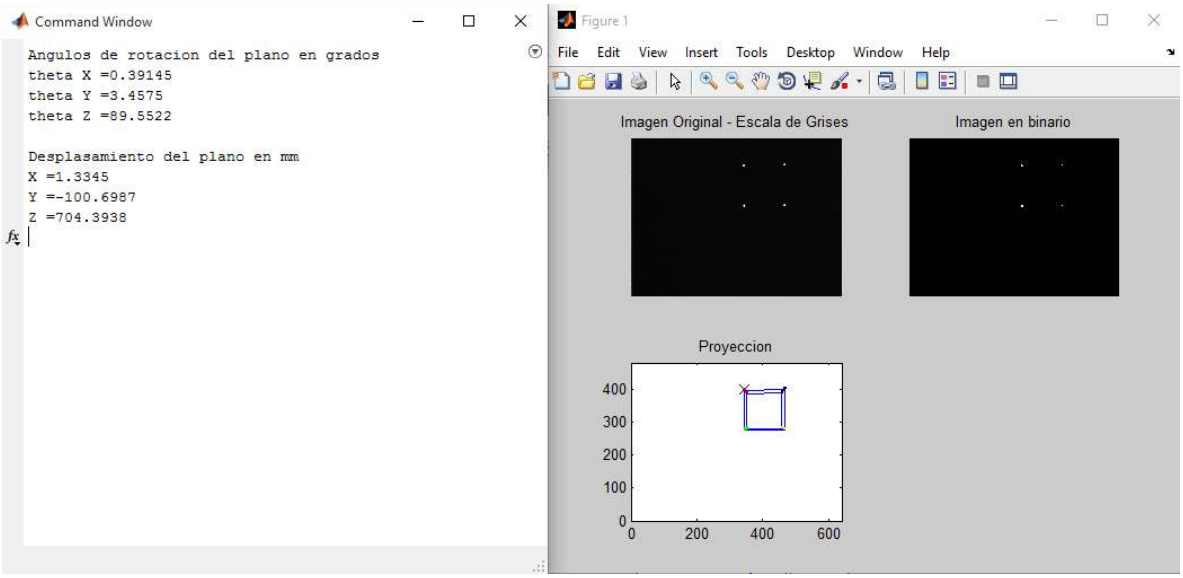


Figura 5.13. Medida del algoritmo moviendo el marco 100mm en dirección del eje Y.

Ya que el punto (0,0) del plano físico se encuentra en la parte superior fue necesario subir la cámara a esa altura para que coincidieran los orígenes, por lo tanto, primero se midió la altura del led superior del marco lo que se muestra en la figura 5.14. Como se puede observar la distancia del led superior respecto a la base es aproximadamente de 105mm

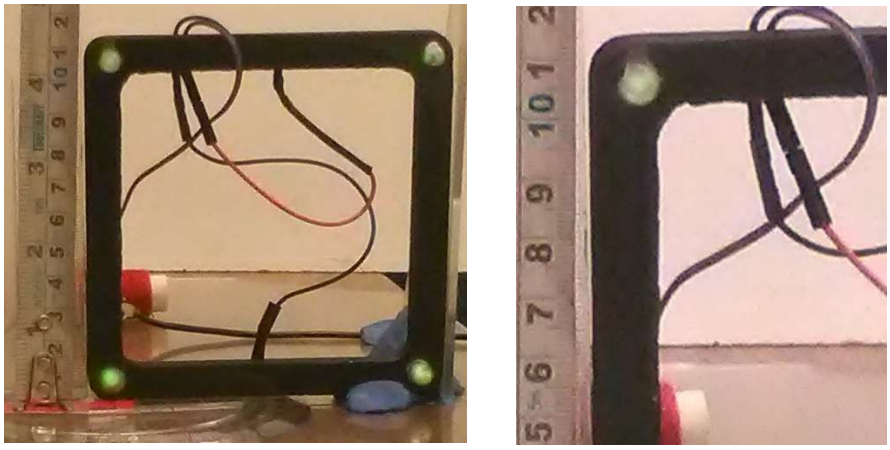


Figura 5.14. Distancia de la base al led superior.

Una vez que se mueve el marco en dirección del eje Y, la medida que se obtiene es aproximadamente 205mm esto se puede verificar en la figura 5.15.

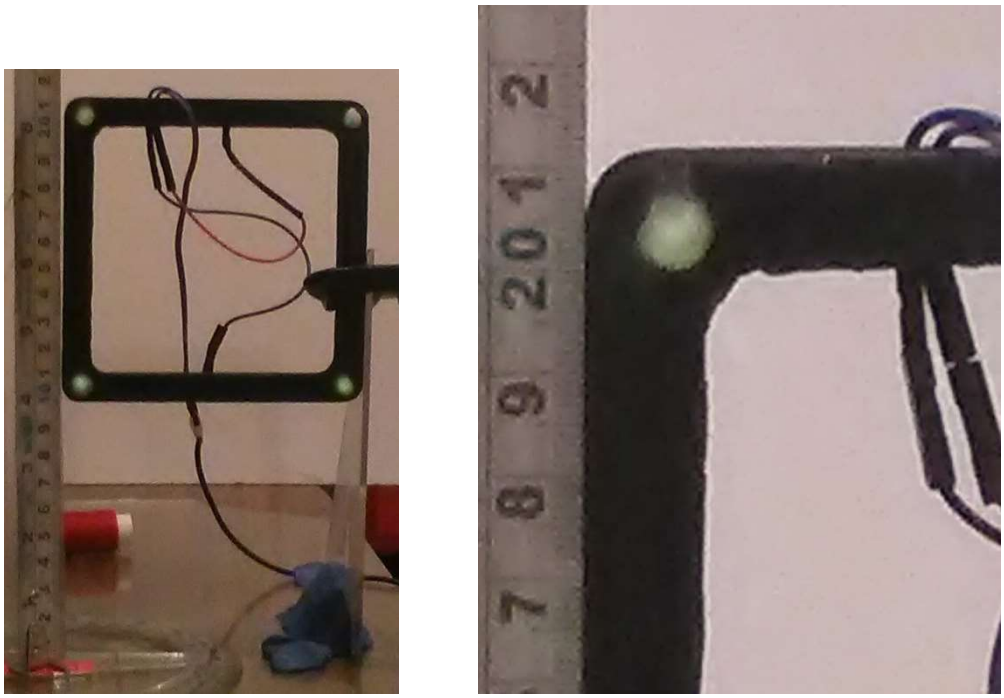


Figura 5.15. Medida de la base al led superior una vez que se movió el marco.

Ya que se obtuvieron las medidas del led superior es solo cuestión de restar las distancia para sacar la distancia del desplazamiento del marco en sentido del eje Y. Esta medida es aproximadamente 100mm, lo cual concuerda con lo medido por el algoritmo. En la tabla siguiente se hace la comparación de los resultados.

Medida referencia	**Medida algoritmo**	**Medida física**
Y=1.0924mm	Y=100.6987mm	Y=100mm

Para medir el ángulo θ_x se giró el marco 15°, el algoritmo arrojó el siguiente resultado (figura 5.16):

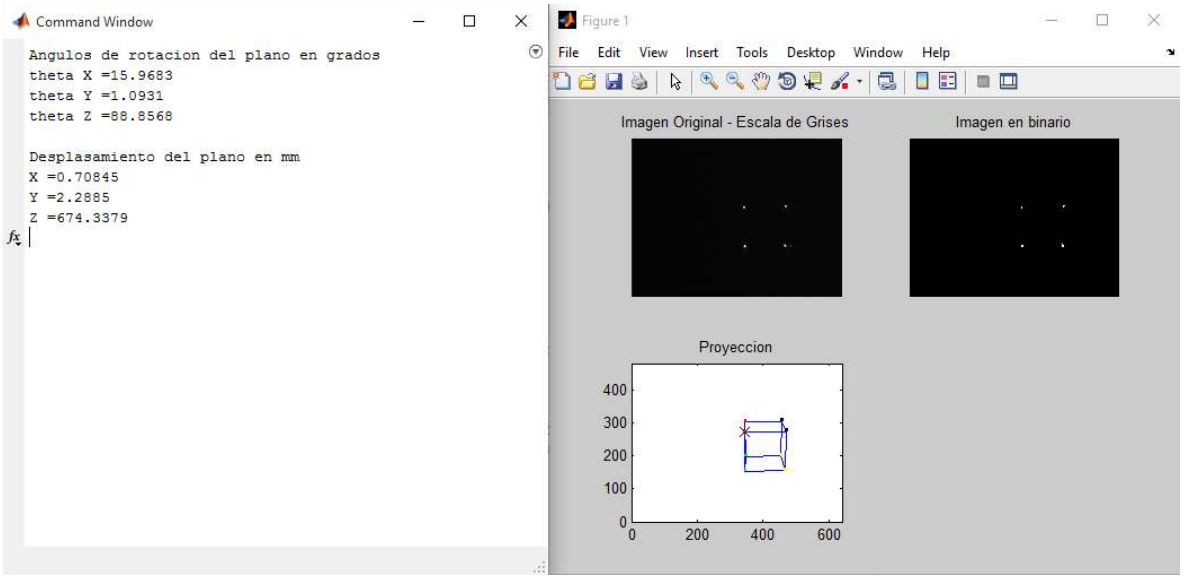


Figura 5.16. Medida de la rotación en el eje X.

Este ángulo se midió con un transportador como se muestra en la figura 5.17.

Figura 5.17. Rotación sobre el eje X.

En la tabla siguiente se muestra las comparaciones de θ_x. Como se puede observar el resultado obtenido mediante el algoritmo coincide con la medida física, con un error de 0.9 grados.

Medida referencia	**Medida algoritmo**	**Medida física**
θ_x=1.0924°	θ_x=15.9683°	θ_x=15°

En la figura 5.18.se muestra la información que nos da el algoritmo tras rotar el marco 20° sobre el eje Y. Las medidas fiscas se pueden observar en las figuras 5.19 y 5.20.

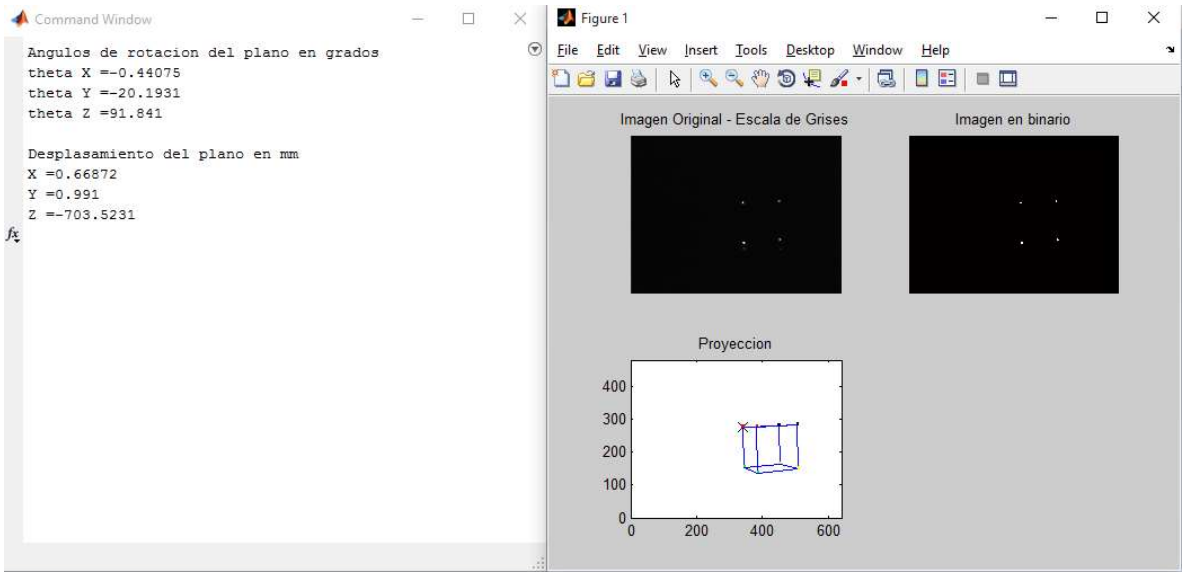


Figura 5.18. Resultados del algoritmo después de rotar sobre el eje Y.

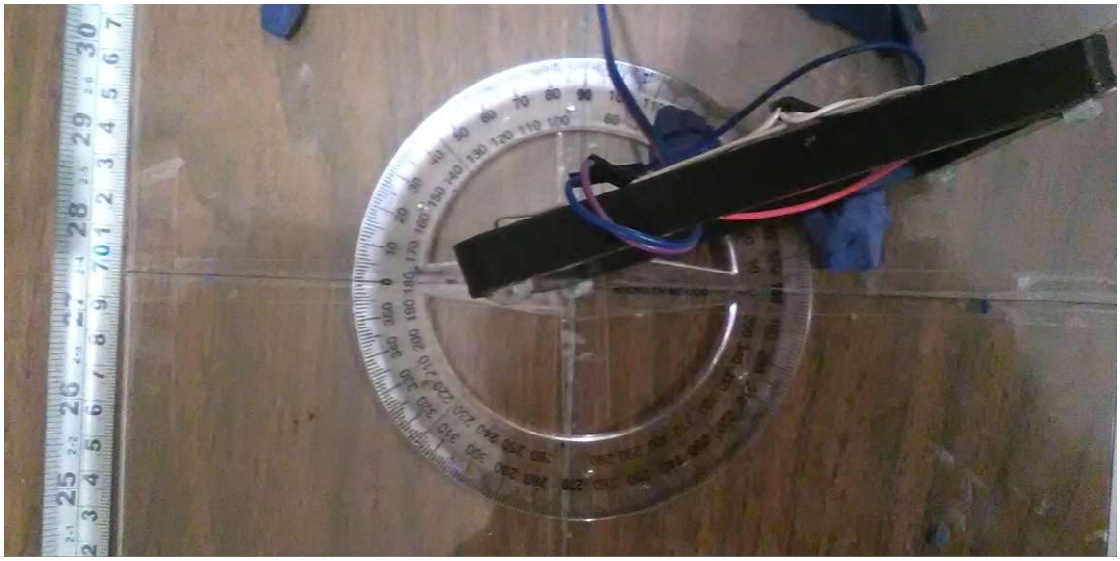


Figura 5.19. Rotación del marco sobre el eje Y.

Figura 5.20. Detalle de rotación sobre el eje Y.

En la tabla se puede ver la comparación de θ_y

Medida referencia	Medida algoritmo	Medida física
θ_y=2.4505°	θ_y=20.19°	θ_y=20°

Por último, se rotará el marco con respecto al eje Z, lo cual nos da como resultado lo que aparece en la figura 5.21. En la medición del algoritmo de referencia se puede ver que la rotación en Z es de 90° esto es debido a como se escogieron los puntos del plano físico.

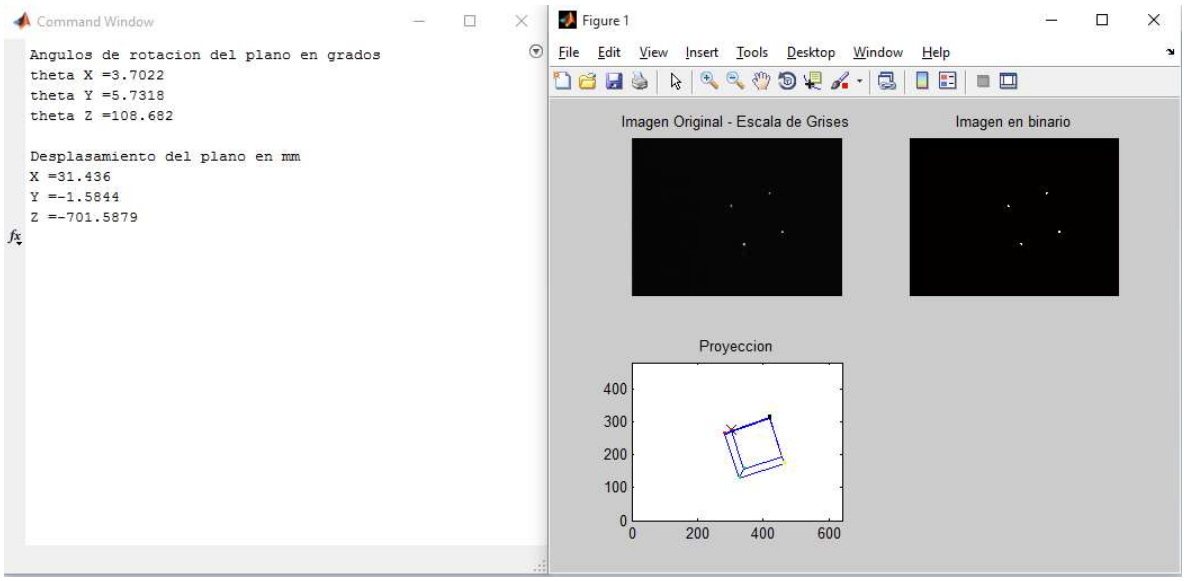


Figura 5.21. Resultado de rotación sobre eje Z.

En las figuras 5.22 y 5.23 se puede apreciar cómo se midió la rotación sobre el eje Z. El valor del ángulo de rotación fue de 19°, esto significa que a este valor hay que sumarle los 90° que salen sin rotar el eje Z.

Figura 5.22. Medición del ángulo de rotación sobre el eje Z.

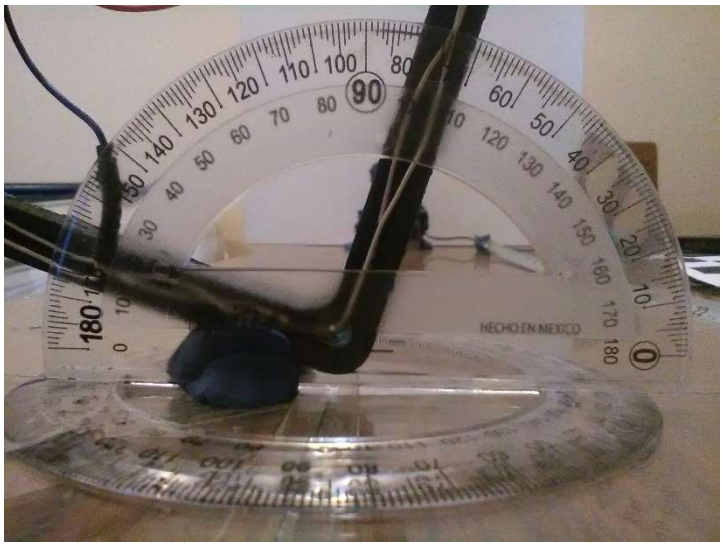


Figura 5.23. Detalle de rotación sobre el eje Z.

En la tabla siguiente se puede ver las comparaciones de los ángulos θ_z.

Medida referencia	Medida algoritmo	Medida física
θ_z=90.673°	θ_z=108.682	θ_z=19°

5.2 Resultados con el Arduino

Por otro lado, con el Arduino se comunicó con el módulo OV7670 pudiendo obtener imágenes como se muestra en la figura 5.24, Ya que la memoria del Arduino es limitada la resolución de la imagen se configuro para que fuera de 174×144.

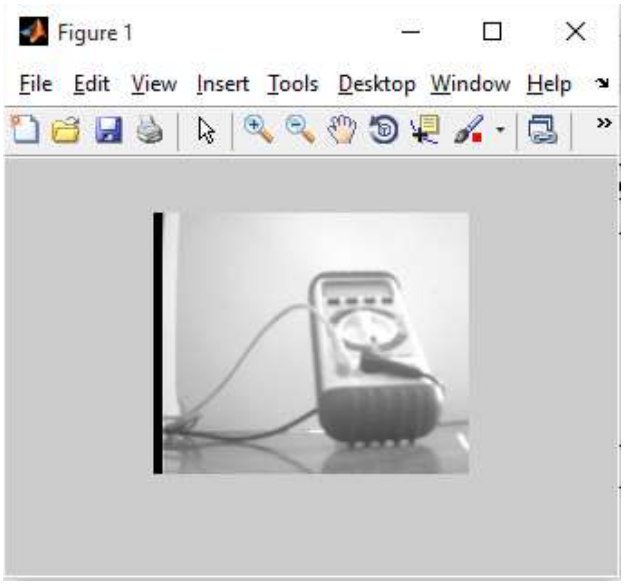


Figura 5.24. Imagen tomada con el módulo OV7670.

En la figura 5.25 se puede apreciar las señales que comprenden una imagen completa, estas señales se tomaron con ayuda de un analizador lógico. Como se puede apreciar la señal VSYNC es la que indica cuando una imagen inicia y cuando termina.

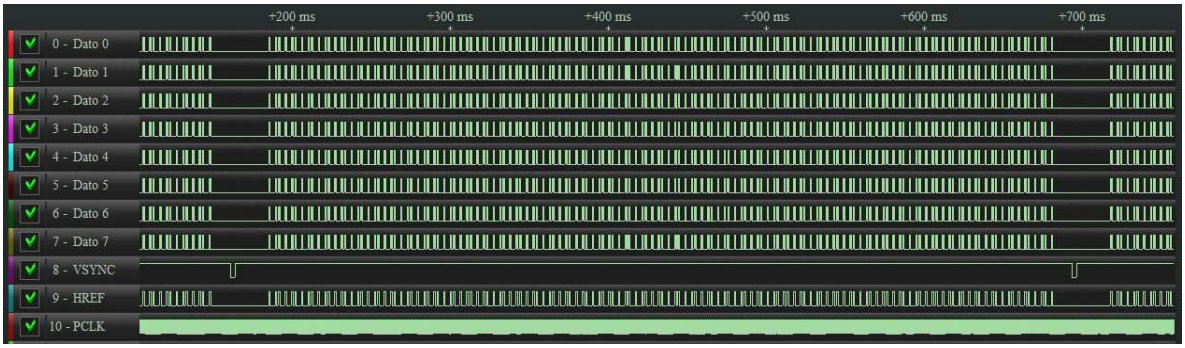


Figura 5.25. Captura de una imagen completa.

Ampliando la figura 5.25 para observar la información de una sola línea horizontal de la imagen se puede ver como la señal HREF indica el principio y fin de una línea horizontal de la imagen (figura 5.26).

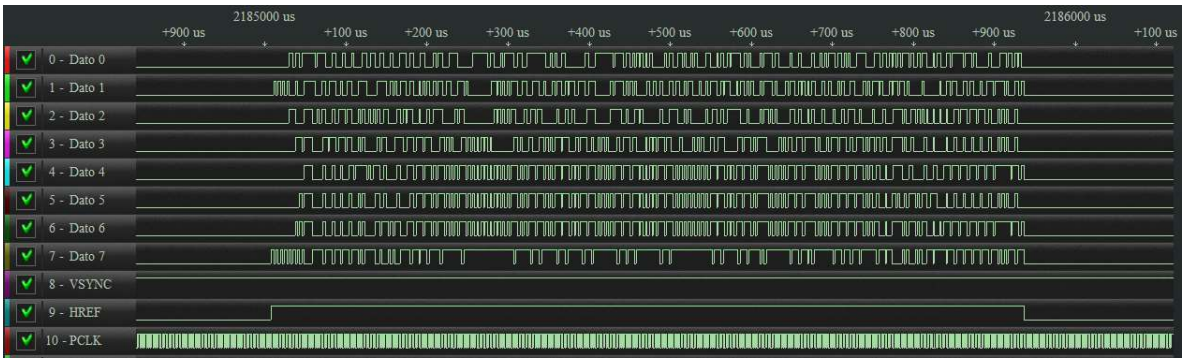


Figura 5.26. Detalle de la captura de una sola línea de la imagen.

Para la información de los pixeles se puede apreciar en la figura 5.27 como la señal reloj cambia de estado lógico 0 a estado lógico 1 para capturar la información.

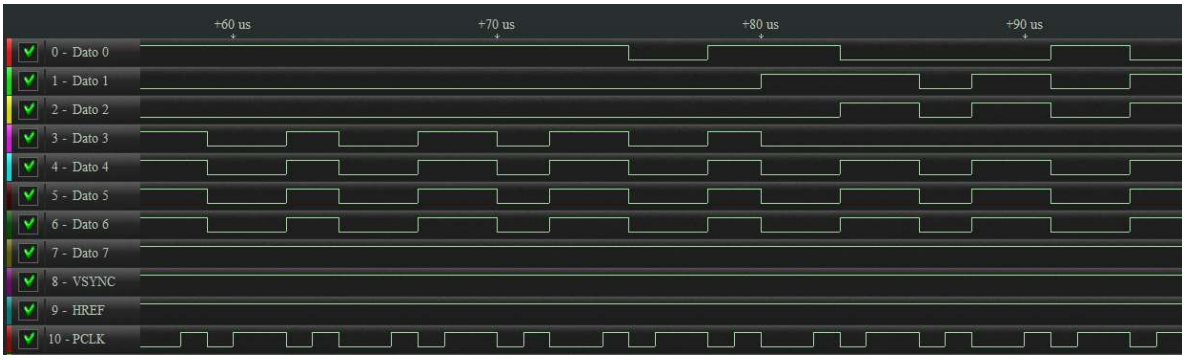


Figura 5.27. Detalle de la captura la información de los pixeles.

En la figura 5.28 se muestra el sistema completo del Arduino y el módulo OV7670 con el que se tomó la imagen. Se puede apreciar también la imagen como el analizador lógico está conectado.

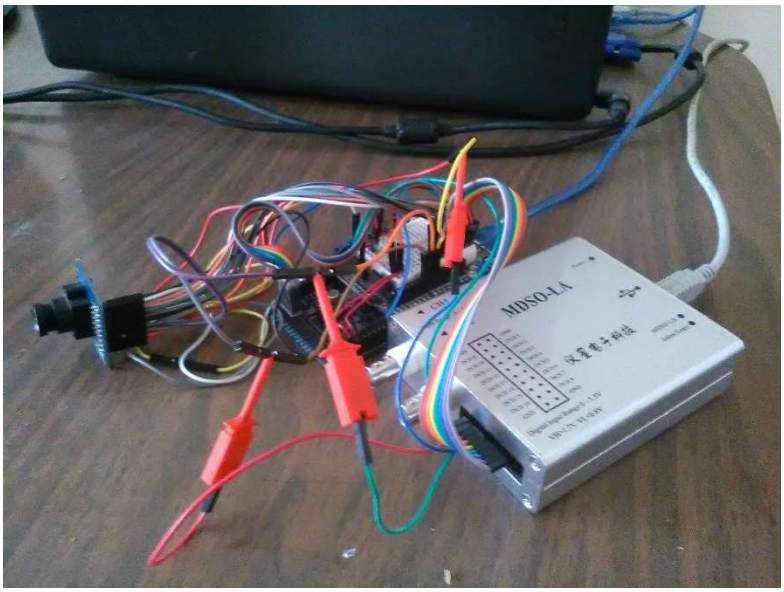


Figura 5.28. Circuito de comunicación con la cámara.

Una vez que se obtuvieron las imágenes se realizó el preprocesamiento de la imagen. Se comenzó binarizando la imagen con el algoritmo antes descrito (figura 4.34), el resultado de la binarización se puede ver en la figura 5.29. Una vez realizada la tarea de binarización se implementó tanto la dilatación como la erosión con los algoritmos de la figura 4.35 y 4.36, los resultados de estas operaciones se pueden ver en la figura 5.30.

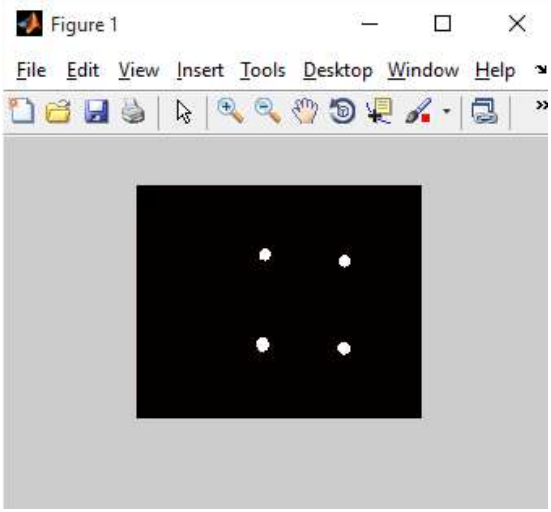


Figura 5.29. Proceso de binarización con la plataforma Arduino.

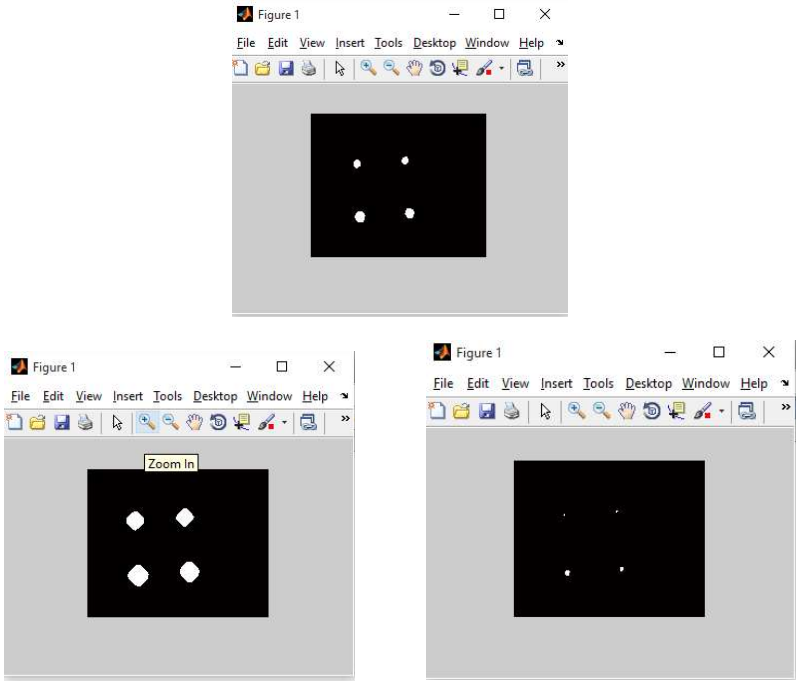


Figura 5.30. Arriba: Imagen original, Abajo izquierda: Imagen dilatada, Abajo derecha: Imagen erosionada.

Para el etiquetado de cada figura dentro de la imagen se utilizó el algoritmo de las figuras 4.37 y 4.38 el cual genera una imagen con valores diferentes en las regiones detectadas. Esto quiere decir que en una región cerrada todos los pixeles tienen el mismo valor. Utilizando la función mat2gray de Matlab a la hora de desplegar la imagen esta función

generara diferentes escalas de grises para los pixeles con el mismo valor. Por lo tanto, como se puede ver en la figura 5.31 cada punto detectado en la imagen tiene un valor de gris diferente.

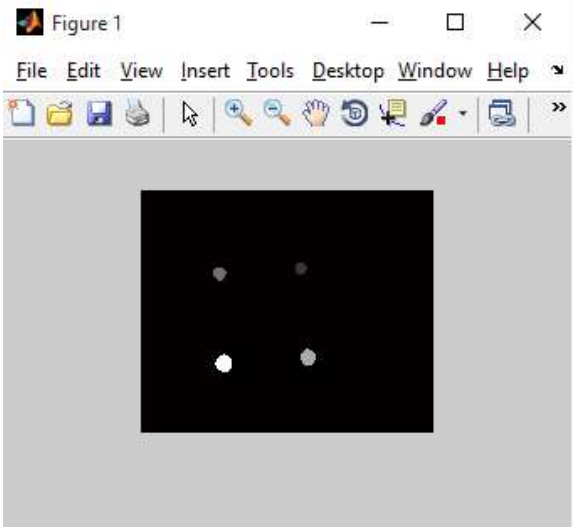


Figura 5.31. Etiquetado de la imagen.

Y como procesamiento final se calculan los centroides con el algoritmo de la figura 4.39. Para la imagen 5.31 los centroides que se recibieron desde la tarjeta Arduino fueron (153,44), (104,47), (157,97) y (108,102). Una vez que se tienen los centroides estos se mandan por el puerto USB a la computadora y esta con Matlab calcula los tres parámetros de rotación y los tres parámetros de traslación con el método descrito en la sección 5.1, pero con la diferencia de que el programa solo recibe los centroides y no hace el preprocesamiento de la imagen.

Como comparativa, el procesamiento de la imagen en la plataforma Arduino tarda alrededor de 2s en mandar los datos a la CPU, mientras que si se hace por medio de la cámara web y Matlab el tiempo máximo es de 2ms. Sin embargo, una ventaja de implementar el algoritmo en un sistema como Arduino implicaría que no se necesitaría de una computadora para que un robot móvil se pudiera posicionar con respecto de un sistema.

6 Conclusiones generales

6.1 Conclusiones

Ha sido posible lograr tanto el objetivo general como los objetivos particulares planteados en esta tesis de forma satisfactoria, puesto que se ha conseguido implementar un sistema que calcula los seis grados de libertad de un plano que se encuentra en el mundo físico, empleando para ello una cámara web de bajo costo y un filtro que permite el paso de únicamente la luz infrarroja.

Este sistema tiene la ventaja de calcular la posición y la orientación de un robot sin utilizar otro tipo de sensores, lo cual representa un avance respecto a la forma en la que actualmente se detecta la posición de un robot móvil, ya que con un solo sistema de referencia propuesto por el usuario (cuatro puntos cooplanares en el mundo físico) es posible obtener toda la información que comúnmente se obtiene empleando tres o cuatro sensores.

Por otro lado, una ventaja de este sistema es que se trata de un sistema monocular, esto implica, por ende, que se emplea únicamente una cámara y no dos o más, lo cual a su vez trae como consecuencia, un aumento en la calidad de información a manejar y mayores recursos de cómputo. Con este sistema las ventajas se traducen en un menor costo del procesamiento de la imagen en relación a los sistemas binoculares, dado que estos utilizan dos cámaras para extraer la información 3D.

Esta ventaja permite, a su vez, que sea más fácil implementarlo en sistemas embebidos ya que al ser sólo una imagen la que es necesario procesar, se necesita de menor memoria RAM para el procesamiento de la imagen y de menos recursos del sistema para procesarla.

Además, el algoritmo que se desarrolló en esta tesis consume pocos recursos de la CPU lo que lo hace bastante rápido. Esto es, gracias al filtro infrarrojo se agilizó el procesamiento de imagen ya que no fue necesario hacer reconocimiento de patrones, detección de colores o algún tipo de algoritmo para detectar los cuatro puntos.

Por otro lado, esta tesis aporta una aplicación de la geometría proyectiva a problemas reales puesto que, al utilizarla en este sistema, fue posible calcular fácilmente los seis grados de libertad del plano físico y poder llegar a un resultado óptimo.

Como se notó en los resultados el sistema comprende un error alrededor de dos milímetros en las traslaciones y de uno a dos grados en las rotaciones. Hecho que, si bien no deja de representar un error, este mismo es tolerable en cuanto al posicionamiento de robots móviles, en donde no se requiere alta precisión para su localización. Este error depende de varios factores, uno de ellos es la resolución de la imagen, ya que entre menos pixeles tenga esta, el error será más grande; otro factor reside en la correcta calibración de la cámara puesto que si no está correctamente calibrada el error puede incrementarse.

Ahora bien, los cuatro puntos pueden estar en posiciones aleatorias siempre y cuando sean cooplanares sin embargo, y con esto se asegura que las mediciones hechas sean correctas.

Este trabajo de tesis es posible aplicarlo al control de robots manipuladores cerrando el lazo de control mediante la técnica de visual servoing, con la ventaja de que solo es necesario emplear una cámara para la obtención de la posición actual del robot (sus 6 grados de libertad), a diferencia de los enfoques actuales que utilizan una combinación de otros sensores o de varias cámaras, para la obtención de la información.

6.2 Trabajo futuro

Finalmente, para proyectos futuros que continúen la temática de la presente con el fin de extender sus resultados -y dado que se comprobó que el algoritmo funciona correctamente- se recomienda implementar el algoritmo completo en un sistema embebido puesto que en la presente tesis se llegó al cálculo de los centroides en el Arduino como se había proyectado. Sin embargo, como se mencionó en el capítulo de resultados, a este le tomaba casi dos segundos desde capturar la imagen hasta mandar a la computadora el valor de los centroides, por tal razón, se recomienda implementar el sistema en un microcontrolador con más prestaciones o, en su defecto, en un FPGA, ya que este último puede trabajar a muy altas frecuencias lo cual representa un tema de un nuevo proyecto de investigación.

7 Referencias

[1] Ciwi, F. Q. (2004). The vision-based metric self-localization of indoor mobile 0robot using projective geometry. IEEE, 914-917.

[2] Dano, R. I. (2000). Reconstruction from six-point sequences. IEEE .

[3] Fernando Caballero, I. M. (2009). A Robust Head Tracking System Based on Monocular Vision. OPEN ACCESS sensors, 8925-8943.

[4] Guillermo Heredia, F. C. (2009). Multi-Unmanned Aerial Vehicle (UAV) Cooperative Fault. Open Access Sensors, 7567-7579.

[5] Marcos, A. G. (2006). Técnicas y Algoritmos Básicos de Visión Artificial. España: Universidad de La Rioja. Servicio de Publicaciones.

[6] Ren Yuan, W. H. (2009). 3D Information Acquisition from Rectangles and Circles in Single Images. World Congress on Computer Science and Information Engineering, 43-47.

[7] Peter Corke,"Robotics Vision and Control Fundamental Algorithms in Matlab". Springer, 2013.

[8] .R. Hartley, A. Zisserman, "Multiple View Geometry in Computer Vision" Cambridge University Press, 2003.

[9] J. L. Mundy , A. Zisserman ,"Geometric Invariance in Computer Vision". . The MIT Press.1992.

[10] D. Forsyth, J. Ponce, "Computer Vision, a Modern Approach". Prentice Hall, 2011.

[11] I. D García, "Visión Artificial y Procesamiento Digital de Imágenes usando Matlab". Universidad de Ecuador, 2008.

[12] A. G. Marcos, "Técnicas y Algoritmos Básicos de Visión Artificial". Universidad de la Rioja, 2013

[13] E. Cuevas, D. Zaldívar , M Pérez,"Procesamiento Digital de Imágnes con MATLAB y Similink". Alfaomega Ra-Ma, 2010.

[14] Perfec Choice, 2016, PC-320449, http://pub.perfectchoice.me/Publicaciones/WebActual/manuales/PC-320449.pdf

[15] Jean-Yves, 2016, Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

[16] MathWorks, 2016, https://www.mathworks.com/products/matlab/

[17] G. Strang, “Introduction to Linear Algebra” Wellesley Cambridge Press, 2003.

[18] Arduino.cc, 2016,Arduino Due, https://www.arduino.cc/en/Main/ArduinoBoardDue

[19] OV7670/OV7171 CMOS VGA (640 X 480) Camera Chip Implemetation Guide. Version 1.0.

[20] OmniVision Serial Camera Control Bus (SCCB) Fuctional Specification. Version 2.2

[21] Atmel.com, 2016, SAM3X/SAM3A Series. http://www.atmel.com/Images/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf

Apéndices

Apéndice A publicaciones

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS

IEEE MORELOS SECTION

CONSTANCIA

Artículo: "*Altitude and orientation sensor based on 4 points coplanar.*"

Autores: **Luis Ernesto Valencia, Amparo Palomino, Gibran Etcheverry, Juan Antonio Escareño and Sergio Vergara**

Id. artículo: **62**

El Comité Técnico del XIII Congreso Internacional Sobre Innovación y Desarrollo Tecnológico CIINDET 2016, que se llevó a cabo en la Ciudad de Cuernavaca, Morelos, México, del 7 al 9 de septiembre de 2016, hace constar que el artículo citado fue presentado de acuerdo con el programa técnico del congreso e incluido en las memorias del mismo.

La presente constancia se expide para los fines legales que a los autores convengan.
Cuernavaca, Morelos, México a 9 de septiembre de 2016.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC. FOUNDED NEW YORK 1884

Atentamente

Dr. Jorge Guillermo Calderón Guizar
Presidénte del Comité Técnico CIINDET 2016

XIII CONGRESO INTERNACIONAL SOBRE INNOVACIÓN Y DESARROLLO TECNOLÓGICO

CIINDET 2016
HACIA UN DESARROLLO SUSTENTABLE CON TECNOLOGÍA PROPIA

IEEE Sección Morelos

La Sección Morelos del Instituto de Ingenieros en Electricidad y en Electrónica

Otorga el presente

Reconocimiento

A: LUIS ERNESTO VALENCIA SEGURA

Por su participación como

Congresista

Durante el XIII Congreso Internacional sobre Innovación y Desarrollo Tecnológico, realizado del 7 al 9 de Septiembre del 2016 en la ciudad de Cuernavaca, Morelos, México.

M. C. Julio A. Hernández Galicia
Presidente IEEE Sección Morelos

Dr. Francisco Ponce Maldonado
Presidente del Ciindet 2016

Altitude and orientation sensor based on 4 points coplanar.

L. E. Valencia Segura, A. Palomino Merino, G. Etcheverry Doger, J. A. Escareño Castro, S. Vergara Limon

Abstract: **This document describes a method for locating an object in physical coordinates using a monocular system. This system take images of four points belonging to a plane and using the projective geometry theory, the position of the plane is recovered from a 2D image. With this method we obtain the 6 degrees of freedom of the plane in the physical world with respective coordinates of the camera (three translational and three rotational).**

Keywords: **image, homogeneous coordinates, projective plane, projection matrix.**

Resumen: **Este documento describe un método para la localizar un objeto en coordenadas físicas usando un sistema monocular. Este sistema captura imágenes de cuatro puntos pertenecientes a un plano y usando la teoría de la geometría proyectiva, la posición del plano es recuperada desde una imagen en 2D. Con este método obtenemos los 6 grados de libertad del plano en coordenadas físicas con respecto a las coordenadas de la cámara (tres de traslación y 3 de rotación).**

Palabras clave: **imagen, coordenadas homogéneas, plano proyectivo, matriz de proyección.**

El problema de detectar la posición y orientación de un objeto en relación a otro objeto y con base a un sistema de referencia, es una tarea difícil de realizar independientemente del sensor a utilizar. Uno de los sensores más completos es una cámara, la cual junto con un análisis adecuado de la imagen adquirida, puede proporcionar una gran cantidad de información. El problema real consiste en saber obtener la información a partir de la imagen captada por la cámara en el mundo real. En aplicaciones de localización y orientación de los sistemas basados en robots, ya sean terrestres, aéreos o submarinos el problema de detectar la posición del vehículo, respecto de su objetivo, ha sido abordado ampliamente utilizando diferentes métodos (sistemas basados en central inercial, sensores IR, acelerómetros, visión, entre otros). La selección adecuada debe estar en función de la aplicación, sin embargo el problema de la auto localización requiere de varios sensores, por ejemplo un acelerómetro nos proporciona las aceleraciones en los tres ejes (x, y, z), por consiguiente para obtener la posición se tiene que integrar dos veces, lo cual genera un error que con el pasar del tiempo aumenta cada vez más, haciéndose necesario utilizar métodos para eliminar el error como seria implementar un filtro Kalman. También se tiene el inconveniente de que el sistema por si sólo no determina la posición inicial con respecto a su sistema de referencia. El problema de la ubicación del objeto, también se podría abordar un sensor GPS (Global Positioning System) el inconveniente aquí es que el sistema GPS tiene un error medido en metros, lo cual para vuelos en interiores no es adecuado, además de que para detectar o medir las orientaciones sería necesario hacer uso de algún otro sensor. Por otro lado, se podrían utilizar un conjunto de sensores para auto localizar el sistema pero se tiene el inconveniente de que el hardware se complicaría al igual que el software o firmware.

La visión por computadora puede resolver estos problemas abordándolo desde distintos puntos de vista; en la extracción de la información visual contenida en la imagen.

Para poder obtener los parámetros de rotación y traslación de un objeto a partir de una imagen, es necesario obtener la matriz que nos relacione los puntos en el plano físico con los puntos en la imagen. Ya que esta matriz contiene toda la información de las relaciones entre los puntos de ambos planos (plano de la imagen y plano en el mundo físico), para determinar los parámetros que nos interesan, será necesario interpretar los datos de esta matriz.

I. OBTENCIÓN Y ANÁLISIS DE LA IMAGEN

A. *Obtención de la imagen*

Una imagen puede definirse como una función bidimensional $f(x, y)$ donde x y y son coordenadas en el plano y la amplitud f es llamada intensidad o nivel de gris en ese punto. Cuando (x, y) y f son todos finitos (cantidades discretas) llamamos a la función como imagen digital. Es decir, una imagen digital estará compuesta por un número finito de elementos llamados píxeles, cada uno de los cuales con un valor y una posición particular.

Una imagen normalmente es representada como $I(x, y)$ [1], donde el valor de la intensidad I se obtiene por el indexado de las coordenadas x y y. El modelo más común de representación de la imagen es por medio de una matriz, tal que:

Luis Ernesto Valencia Segura, Benemérita Universidad Autónoma de Puebla, Av. San Claudio y 18 sur, Ciudad Universitaria, Puebla, México. (e-mail: ernesto.valencia@alumno.buap.mx).

Amparo D. Palomino Merino, Benemérita Universidad Autónoma de Puebla, Av. San Claudio y 18 sur, Ciudad Universitaria, Puebla, México. (e-mail: palomino@ece.buap.mx).

Gibran Etcheverry, Universidad de las Américas Puebla, Sta. Catarina Mártir. Cholula, Puebla, México. (e-mail: gibran.etcheverry@udlap.mx).

Juan Antonio Escareño Castro, Institut Polytechnique des Sciences Avancées, 15- 21 Rue Maurice Grandcoing, 94200 Ivry-sur-Seine, Francia (e-mail: jantonio.escareno@gmail.com)

Sergio Vergara Limon, Benemérita Universidad Autónoma de Puebla, Av. San Claudio y 18 sur, Ciudad Universitaria, Puebla, México. (e-mail: svergara2@hotmail.com).

$$I(x,y) = \begin{bmatrix} I(1,1) & I(2,1) & \cdots & I(N,1) \\ I(1,2) & I(2,2) & \cdots & I(N,2) \\ \vdots & \vdots & \ddots & \vdots \\ I(1,M) & I(2,M) & \cdots & I(N,M) \end{bmatrix} \quad (1)$$

B. *Modelo de la cámara*

Al pasar de coordenadas 3D del mundo real a coordenadas 2D de la imagen, es un proceso de proyección en el cual se pierde una dimensión. La manera usual de modelar este proceso es por la proyección central, en el cual un rayo es trazado desde un punto en el espacio 3D a través de un punto fijo en el espacio (centro de proyección C) [2]. Este rayo intersecta un plano al cual se le llama el plano de la imagen Fig.(1).

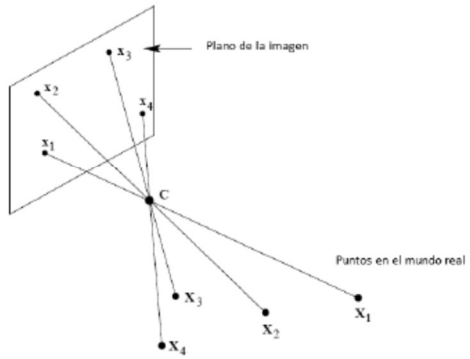


Fig. 1 Modelo de proyección de la imagen.

Este modelo concuerda con el modelo simple de una cámara, ignorando los efectos como foco y espesor del lente.

Una manera de interpretar el plano proyectivo P^2 (espacio proyectivo de dos dimensiones) es un conjunto de rayos en R^3(espacio euclidiano de 3 dimensiones). El conjunto de todos los vectores $k(x_1, x_2, x_3)^T$ con k variante, puede formar un rayo a través del origen. Así como un rayo puede ser pensado como la representación de un único punto en P^2. En este modelo, las líneas en P^2 son planos pasando a través del origen. Los puntos y líneas pueden ser obtenidas intersectando sus conjuntos de rayos y planos por el plano Π en $x_3 = 1$, como se ilustra en la Fig. (2). Los rayos representan puntos ideales y el plano representa l_∞ esta paralelo al plano x_3.

C. *Transformaciones proyectivas*

Una transformación proyectiva planar es una transformación lineal de vectores homogéneos de 3 dimensiones representada por una matriz no singular de 3x3 (2).

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \quad (2)$$

donde x_i con $i = 1,2,3$ son las coordenadas del plano de la imagen, X_i con $i = 1,,2,3$ son las coordenadas del plano en el mundo real y h_{ij} con $i = 1,2,3$ y $j = 1,2,3$ son los elementos de la matriz de transformación.

O en su forma reducida, $x = HX$

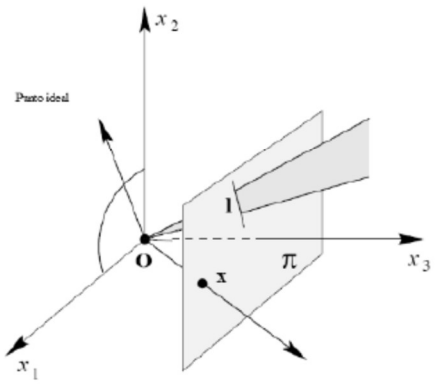


Fig. 2 . Puntos y líneas de P^2.

D. *Mapeo entre planos*

Como se observa en la Fig. (3) la proyección a lo largo del rayo a través de un punto común (el centro de proyección) define un mapeo desde un plano a otro. Este mapeo punto a punto preserva las líneas, una línea en un plano es mapeada a una línea en el otro plano. Si un sistema de coordenadas es definido en cada plano y los puntos son representados en coordenadas homogéneas, entonces el mapeo de la proyección central puede ser expresada por $x = HX$ donde H es una matriz de 3x3 no singular.

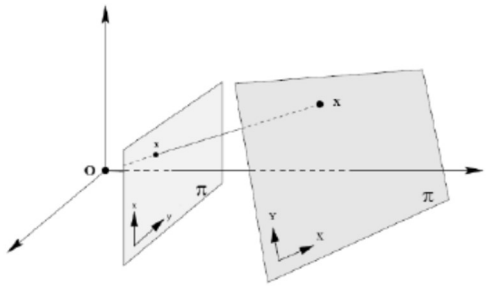


Fig. 3 Mapa de la proyección central de puntos en un plano a puntos en otro plano.

E. *Cálculo de la matriz H de transformación 2D a 2D*

Para el cálculo de la matriz de transformación H que nos relaciona el plano de la imagen con el plano que está en el mundo físico, es decir

$$x_{uv} = HX_{xy} \quad (3)$$

Donde x representa un punto en las coordenadas del plano de la imagen y X representa el mismo punto pero en coordenadas del plano en el mundo físico. La ecuación 3 en su forma explícita queda:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Conociendo los valores de las coordenadas en el plano físico y los valores en el plano de la imagen y resolviendo el sistema (4) se puede calcular los valores de la matriz de proyección [3].

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -u_2X_2 & -u_2Y_2 & -u_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -v_2X_2 & -v_2Y_2 & -v_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -u_3X_3 & -u_3Y_3 & -u_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -v_3X_3 & -v_3Y_3 & -v_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -u_4X_4 & -u_4Y_4 & -u_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -v_4X_4 & -v_4Y_4 & -v_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \quad (4)$$

F. *Cálculo de los valores extrínsecos de la cámara*

Una vez obtenida la matriz de transformación H, el paso siguiente es obtener la matriz de rotación y vector de traslación (valores extrínsecos) para obtener los valores de la posición y rotación del plano[4].

Se sabe que

$$K^{-1}H = \lambda R \quad (5)$$

donde K es la matriz de parámetros intrínsecos de la cámara, R es la matriz de rotación y λ es una constante [5].

La ecuación 4 se puede expresar como.

$$K^{-1}H = \lambda[r_1|r_2|t] \quad (6)$$

Donde r_1 y r_2 son vectores en R^3 y son los vectores ortonormales del plano físico y t es el vector de traslación en R^3. Sabiendo esto tenemos que:

$$\|r_1\| = \|r_2\| = 1 \quad (7)$$

Por lo tanto tenemos que

$$K^{-1}[h_1 \mid h_2 \mid h_3] = [\lambda r_1|\lambda r_2|\lambda t] \quad (8)$$

Utilizando las propiedades de la norma de vectores tenemos que

$$\|k^{-1}h_1\| = \|k^{-1}h_1\| = \lambda \quad (9)$$

Utilizando la igualdad de la ecuación 8 para encontrar el valor de λ nos queda expresión (10).

$$Rt = \frac{k^{-1}H}{\lambda} \quad (10)$$

G. *Cálculo de la matriz de proyección de 3D a 2D*

Para realizar el cálculo de esta matriz sólo se necesita obtener el vector perpendicular a r_1 y r_2 por lo tanto.

$$r_3 = r_1 \times r_2 \quad (11)$$

Quedándonos la matriz de proyección de la cámara de la forma (12).

$$P = [r_1|r_2|r_3|t] \quad (12)$$

Esta matriz contiene toda la información de las rotaciones y traslaciones del plano físico con respecto al plano de la cámara.

La matriz P se multiplica por la matriz de parámetros intrínsecos de la cámara; esto nos dará como resultado una matriz de proyección de la cámara de la forma (13).

$$P_K = KP \quad (13)$$

H. *Recuperación de los datos del plano físico*

Una vez obtenida la matriz P de la ecuación (12) es necesario obtener los datos de traslación y rotación del plano físico.

Los valores de la traslación se obtienen directamente del vector t esto quiere decir que se cumple (14)

$$t = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (14)$$

Por otro lado para recuperar los valores de rotación del plano físico se utilizan las matrices de rotación euclidiana donde los vectores r_i con i=1,2,3 de la matriz de rotación quedan definidas de la siguiente manera

$$r_1 = \begin{pmatrix} -\cos(\theta_z)\cos(\theta_y) \\ cos(\theta_z)sen(\theta_y)sen(\theta_x) + cos(\theta_x)sen(\theta_z) \\ -sen(\theta_z)sen(\theta_x) - cos(\theta_z)cos(\theta_x)sen(\theta_y) \end{pmatrix} \quad (15)$$

$$r_2 = \begin{pmatrix} -\cos(\theta_y)\,sen(\theta_y) \\ sen(\theta_z)sen(\theta_y)sen(\theta_x) - cos(\theta_z)cos(\theta_x) \\ cos(\theta_z)sen(\theta_x) + cos(\theta_x)sen(\theta_z)sen(\theta_y) \end{pmatrix} \quad (16)$$

$$r_3 = \begin{pmatrix} sen(\theta_y) \\ -cos(\theta_y)sen(\theta_x) \\ cos(\theta_y)cos(\theta_x) \end{pmatrix} \quad (17)$$

Siendo θ_x, θ_y y θ_z los ángulos de rotación del plano físico con respecto de los ejes x y y z respectivamente.

Utilizando álgebra para despejar los ángulos de rotación, tenemos que, de la ecuación (17)

$$\theta_y = sen^{-1}(P_{13})$$

$$\theta_x = \text{sen}^{-1}\left(\frac{P_{11}}{\cos(\theta_y)}\right)$$

$$\theta_z = \cos^{-1}\left(\frac{P_{11}}{\cos(\theta_y)}\right) \quad (18)$$

Estas ecuaciones nos regresan la información de la posición lineal y la rotacional del plano físico.

II. IMPLEMENTACIÓN DEL HARDWARE

Lo anteriormente descrito en la sección I se ha llevado a su implementación con el fin de comprobar de forma experimental la obtención de la información. El diagrama a bloques en la Fig.(4) muestra como está constituido el sistema completo.

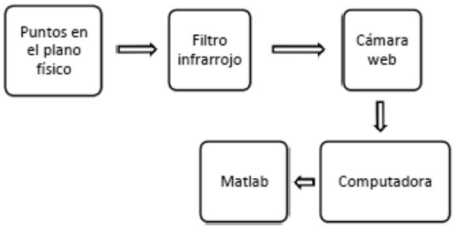


Fig. 4 Diagrama a bloques del sistema.

Para generar los puntos en el espacio 3D (plano físico) se construyó un dispositivo experimental, con base en un marco de material ABS el cual tiene 4 led´s infrarrojos ubicados en las esquinas del cuadrado y estos leds están separados por una distancia de 10 cm como lo muestra en la Fig. (5). De esta manera generamos cuatro puntos sobre un plano, el cual permitirá adquirir una imagen que pueda ser rotada-desplazada en los ejes y detectar las variables buscadas.

Fig. 5 Disposición de los led's infrarrojos en un marco de plástico

El siguiente elemento que se utiliza es un filtro de luz infrarroja Fig. (6) para discriminar toda la luz que llega del ambiente a la cámara web y únicamente dejar pasar la luz infrarroja proveniente de los 4 puntos (led's).

Fig. 6 Filtro de luz infrarroja.

En la Fig. (7) se puede ver el dispositivo experimental con los 4 led's infrarrojos tomados con la cámara sin el filtro infrarrojo, en contraste con la Fig. 8 en la que se puede observar el mismo dispositivo experimental con los led's pero esta vez con el filtro infrarrojo.

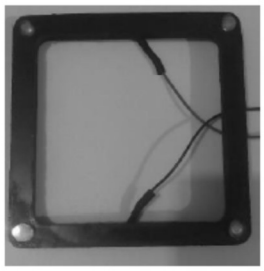


Fig. 7 Imagen obtenida desde la cámara sin filtro infrarrojo.

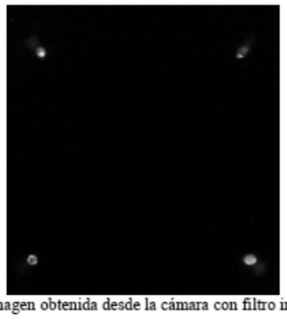


Fig. 8 Imagen obtenida desde la cámara con filtro infrarrojo.

La cámara que se utiliza para las pruebas es la cámara Web Perfect Choice PC-320449 [6] Fig. (9). Aunque esta cámara no posee las mejores características técnicas, consideramos que es suficiente como una primera aproximación para obtener resultados experimentales que permitan comprobar la teoría.

Fig. 9 Cámara utilizada para las pruebas.

Para la calibración de la cámara se hizo uso de la herramienta Camera Calibration Toolbox de Matlab [7].

La programación del algoritmo basado en cuatro puntos se realizó en código de MatLab R2014 [8], así como la adquisición de la imagen.

III. IMPLEMENTACIÓN DEL SOFTWARE

En la Fig. (10) presenta el diagrama de flujo del software que se implementó para la detección del plano y el cálculo de posición y orientación del mismo.

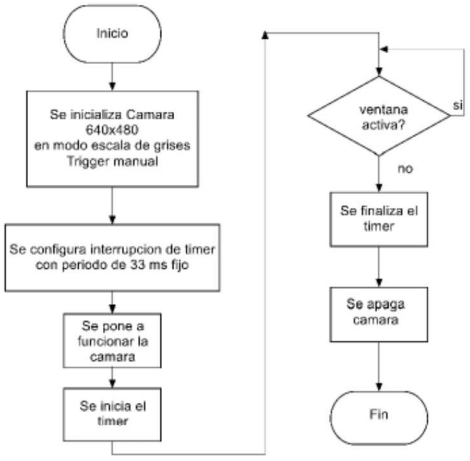


Fig. 10 Diagrama de flujo del programa principal.

El primer paso es la configuración de la cámara y adquisición de las imágenes para lo cual se utilizó la herramienta Image Acquisition Toolbox de MatLab [8]. Con esta herramienta se configuro la cámara para que capture las imágenes con una resolución de 640x480 pixeles, también se configuró para que tome las imágenes en escala de grises y que se capture la imagen en forma manual. Posteriormente se configura un timer para mandar una interrupción cada 33 ms ya que la cámara toma imágenes a 30 cuadros por segundo y esta velocidad es fijada por el fabricante.

En la Fig. 11 se muestra el diagrama de flujo de la subrutina que se ejecuta cada vez que la interrupción del timer es activada.

En este programa una vez que se adquiere la imagen se procede a realizar un preprocesamiento, con el fin de eliminar ruido, de tal manera que en el procesamiento posterior se obtengan mejores resultados. Este preprocesamiento consiste en una binarización, dilatación, erosión, etiquetado y cálculo de centroides.

A. Binarización de la imagen

La binarización de imágenes es una técnica del procesamiento de imágenes que consiste en un proceso de reducción de la información de una imagen digital a dos valores: 0 (negro) y 1(blanco) [9].

Esta técnica consiste en comparar cada pixel de la imagen con un determinado umbral (valor límite que determina si un pixel tendrá un valor de 0 ó 1 en su equivalente en color). Los valores de la imagen que sean mayores que el umbral toman un valor 1 (blanco), el resto de pixeles toman valor 0(negro) (figura 12).

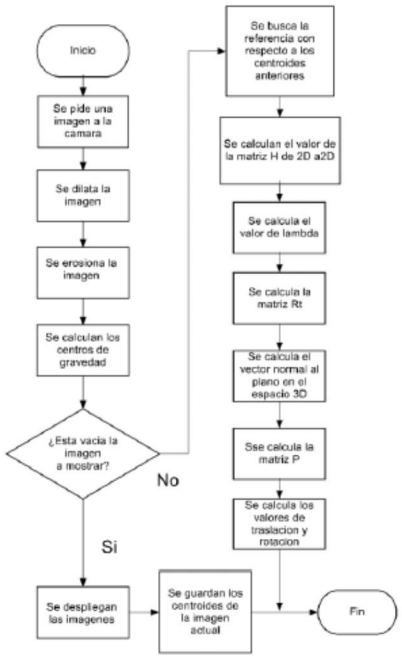


Fig. 11 Diagrama de flujo del proceso para ejecutar la interrupción.

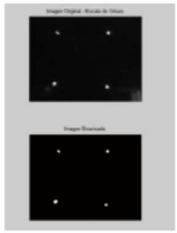


Fig. 12 Comparación de imagen en escala de grises e imagen binarizada.

B. Dilatación y erosión de imágenes

La erosión y la dilación son las operaciones morfológicas más usadas en procesamiento de imágenes [7].

La función de dilación es tomar cada píxel del objeto (con valor "1") y poner al valor "1" todos aquellos píxeles pertenecientes al fondo (background) que tienen una conectividad C (C=4, C=8, etc) con el píxel del objeto. En pocas palabras, asignar el valor 1 a los pixeles del fondo vecinos a los píxeles del objeto.

En nuestro caso la dilatación se usa para agrandar los puntos encontrados y eliminar partes de los puntos que estén separadas del punto principal, formando un solo punto, así evitando errores en la posterior cuantificación de puntos.

La erosión consiste en asignar un valor "0"a cada píxel del objeto que tiene una conectividad C con los píxeles del fondo.

En otras palabras, poner a "0" los pixeles del objeto vecinos a los pixeles del fondo. Esto se hace después de la dilación para regresar a su tamaño normal los puntos y de esa forma tener solo el número correcto de puntos (Fig. 13).

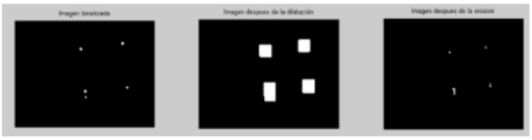


Fig. 13 Izquierda: Imagen binaria original (5 puntos). Centro: Puntos dilatados. Derecha: Puntos erosionados (4 puntos).

C. *Etiquetado*

Es una técnica que asigna una etiqueta a cada componente conexo para obtener al final una región [10]. Básicamente el algoritmo recorre una imagen binaria de izquierda a derecha y de arriba hacia abajo en busca de pixeles con valor "1" que se encuentren conectados entre sí en un rango que puede ser C=8, C=4 etc. y les asigna una etiqueta construyendo de esta manera regiones. Esta técnica, no es tan susceptible a tener errores si los caracteres están rodeados o de líneas que cubran o rodeen a los caracteres. Además no requiere que la imagen se encuentre alineada con respecto al eje para ser efectiva lo cual es de gran utilidad debido a que no siempre se pueden tener correctamente alineado el dispositivo móvil con la imagen como sería con un escáner de sobremesa o plano.

Una vez que en la imagen ya se encuentra con cuatro puntos y uno de ellos es un punto de referencia sin ruido, se procede a realizar un procesamiento, iniciando con la obtención de lo centroides.

D. *Centroides*

Ya que se tienen los puntos etiquetados es necesario encontrar sus coordenadas en la imagen, para esto se determinan los centroides [1] de los puntos u objetos en la imagen.

El centroide o centro de masa de un objeto es el punto en donde por su geometría se encuentra concentrada la masa del objeto. El centroide $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y})$ de un objeto binario se calcula como el punto medio aritmético de las coordenadas en la dirección x y y, tal que

$$\bar{x} = \frac{1}{Area(O)} \sum_{(x,y)\epsilon O} x \qquad \bar{y} = \frac{1}{Area(O)} \sum_{(x,y)\epsilon O} y \qquad (19)$$

Donde un objeto O de una imagen binaria puede ser interpretado como una distribución de puntos de valor uno $\boldsymbol{x_i} = (x_i, y_i)$ en una matriz bidimensional, esto es

$$O = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N}\} = \{(x_1, y_2), (x_2, y_2), \dots, (x_N, y_N)\} \quad (20)$$

El área de un objeto O puede ser calculada mediante el número de pixeles que lo conforman, esto es:

$$Area(O) = N = |O| \qquad (21)$$

y

$$\sum_{(x,y)\epsilon O} \qquad (22)$$

representa la sumatoria de todos los puntos que están dentro del área del objeto.

Por último se realizan los cálculos de las matrices (4), (9), (10), (11), (12) y (18) como lo indica el diagrama de flujo de la Fig. 11. A partir de estas se obtienen posiciones lineales y las rotaciones del plano.

IV. Resultados

El algoritmo fue probado con imágenes fijas y en tiempo real. Las medidas que arroja el programa están en milímetros para las medidas de traslación y en grados para las medidas de los ángulos de rotación. Los valores reales se obtuvieron de forma manual para realizar la comparativa.

La Fig. 14 muestra la disposición de los elementos físicos involucrados en la plataforma experimental, con los elementos descritos en la sección II y la programación en MatLab de los algoritmos descritos en la sección III. El dispositivo de los 4 led's infrarrojos se colocó frente a la cámara en forma paralela, logrando obtener el plano con 4 puntos infrarrojos a una distancia de 50 cm del plano de la imagen.

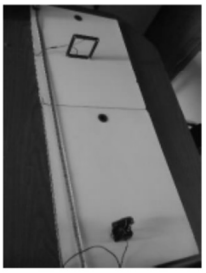


Fig. 14 Plataforma experimental.

La figura 15 muestra las imágenes obtenidas del programa hecho en MatLab. Los datos obtenidos por el algoritmo se presentan en la tabla 1.

Tabla 1 Datos experimentales

	Datos reales	*Datos obtenidos del algoritmo*
X	30mm	29.61mm
Y	60mm	62.03mm
Z	500mm	501.5mm
θ_x	2°	1.24°
θ_y	24°	25.572°
θ_z	90°	92.335°

Como se puede observar los datos de la Tabla 1 corresponden a las posiciones y ángulos del plano que se encuentra en el mundo real con respecto del plano de la imagen. Estos datos que corresponden con la realidad presentan un error máximo de ±2mm que depende de las distancias entre los pixeles y la forma de los 4 puntos en la imagen que en consecuencia afectan el cálculo de los centroides.

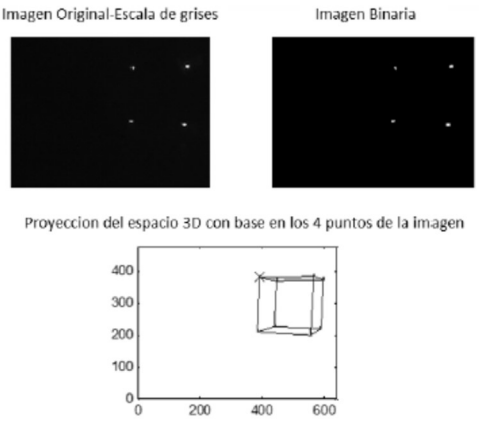


Fig. 15 . Calculando la posición y orientación de un plano con respecto a 4 putos en el plano.

V. CONCLUSIONES

El algoritmo que se a desarrollado cumple con los objetivos planteados al poder realizar el cálculo de los 6 grados de libertad de un plano que contiene 4 puntos. Al utilizar el filtro infrarrojo se logró evitar utilizar un algoritmo de preprocesamiento de imagen más complejo como lo sería la detección de esquinas o detección de formas entre otros, ayudando con esto a que el algoritmo sea más rápido y se ejecute en tiempo real.

Es importante también mencionar que este algoritmo depende de una buena calibración de la cámara, aunque esta no tiene que ser muy costosa.

Con el método presentado en este trabajo se puede ver que son sólo necesarios 4 puntos coplanares para determinar los 6 grados de libertad de un objeto abriendo la posibilidad de cerrar lazos de control sin utilizar ningún otro sensor de orientación y traslación (central, inercial, GPS, ultrasónico, entre otros).Para problemas de 3D de Visual Servoing son necesarias como mínimo 2 cámaras para determinar la posición en 3D, esto quiere decir que es necesario el procesamiento de imágenes de dos o más cámaras.

A futuro ya que este algoritmo no requiere de mucho procesamiento podría adaptarse a un microcontrolador o sistema embebido para así poderlo montar en robots móviles autónomos, haciendo con esto un sensor de grandes prestaciones.

El trabajo desarrollado aquí contribuye de manera importante, en el área de visión e instrumentación, ya que no existe un sensor que detecte estas variables físicas y con la precisión que se menciona.

VI. REFERENCIAS

[1] E. Cuevas, D. Zaldívar, M. Pérez. "Procesamiento Digital de Imágnes con MATLAB y Similink". Impreso por Alfaomega Ra-Ma. 2010.

[2] R. Hartley, A. Zisserman, "Multiple View Geometry in Computer Vision" .Cambridge University Press, 2003.

[3] J. L. Mundy, A Zisserman. "Geometric Invariance in Computer Vision". Cambridge,1992.

[4] D. Forsyth, J. Ponce, "Computer Vision, a modern approach". Prentice Hall, 2011.

[5] M. Shah, "Fundamentals of Computer Vision", University of Central Florida. 1997.

[6] Camara. Perfect Choice PC-320449. Junio 2015. Disponible en: http://pub.perfectchoice.me/Publicaciones/WebActual/manuales/PC-320449.pdf

[7] Camera Calibration. Junio 2015. Disponible en: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

[8] MatLab . Junio 2015. Disponible en : http://www.mathworks.com/help/matlab

[9] Kenneth, R. Castleman, "Digital Image Processing", Prentice Hall.

L. E. Valencia Segura. Licenciado en Ciencias de la Electrónica egresado de la Benemérita Universidad Autónoma de Puebla (BUAP); actualmente estudiante de la Maestría en Ciencias de la Electrónica opción Automatización en la Benemérita Universidad Autónoma de Puebla (BUAP).

A. D. Palomino Merino. Licenciada en Electrónica por la BUAP, Maestra en Ciencias en Ingeniería Eléctrica por el CINVESTAV-IPN y Doctora en Tecnologías de la Información y Sistemas, especialidad Sistemas de Control por la UTC, Francia. Actualmente es Profesor Investigador en la Facultad de Ciencias de la Electrónica. BUAP

G. Etcheverry, Ingeniero en Electrónica por la Universidad Autónoma de Puebla en 2000, Maestro en Electrónica Avanzada por la Universidad de Warwick en 2001 y Doctor en Sistemas Automáticos por la Universidad de Toulouse III en 2006.

J. A. Escareño Castro. Es Ingeniero Electrónico y Maestro en Ciencias en Ingeniería Eléctrica por el Instituto Tecnológico de la Laguna, Torreón, México; Doctor en Control Automático por la UTC, Francia. Actualmente es Profesor Investigador en el Instituto Politécnico de Ciencias Avanzadas (IPSA), Francia.

S. Vergara Limón. Es Licenciado en Electrónica por la BUAP, Maestro en Ciencias y Doctor en Ciencias con especialidad en Optoelectrónica. Actualmente es Profesor Investigador en la Facultad de Ciencias de la Electrónica de la BUAP, Miembro del SNI nivel I desde el 2006.

2do CONGRESO IBEROAMERICANO DE INSTRUMENTACIÓN Y CIENCIAS APLICADAS

SOMI XXXI

CONGRESO DE INSTRUMENTACIÓN

CD. DE GUATEMALA, GUA.,
26 al 28 de octubre de 2016

La Sociedad Mexicana de Instrumentación,
el Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la
Universidad Nacional Autónoma de México y la Facultad de Ingeniería
de la Universidad de San Carlos de Guatemala

Otorgan el presente

RECONOCIMIENTO

a: David Palomino Merino, Luis Ernesto Valencia Segura,
Amparo D. Palomino Merino, Sergio Vergara Limon, Jaime Cid Monjaraz

por haber presentado su trabajo

Algoritmo de detección de la altitud y orientación sensor basado en 4 puntos coplanares

CD. DE **GUATEMALA, GUA.,** 28 de octubre del 2016

Por el Comité Organizador

Dr. Rodolfo Zanella Specia
Director
Centro de Ciencias Aplicadas y Desarrollo Tecnológico

SOMI XXXI CONGRESO DE INSTRUMENTACIÓN
XXXI Edición, Cd. de Guatemala, Gua., 26 al 28 de octubre de 2016
2do CONGRESO IBEROAMERICANO DE INSTRUMENTACIÓN Y CIENCIAS APLICADAS
ISSN 2395-8499

Algoritmo de detección de la altitud y orientación basado en 4 puntos coplanares

David Palomino Merino
Centro de Ciencias Aplicadas y Desarrollo Tecnológico, UNAM
Ciudad de México
davpalomino@gmail.com,

Luis Ernesto Valencia Segura, Amparo D. Palomino Merino, Sergio Vergara Limon, Jaime Cid Monjaraz
Facultad de Ciencias de la Electrónica,
Benemérita Universidad Autónoma de Puebla
Puebla, México
luis.valencia.segura@gmail.com, palomino@ece.buap.mx, svergara2@hotmail.com, jaime.cid@correo.buap.mx

RESUMEN

En este trabajo se describe un método para la localizar un objeto en coordenadas físicas usando un sistema monocular. Este sistema captura imágenes del mundo real, formadas por cuatro puntos pertenecientes a un plano y usando la teoría de la geometría proyectiva, la posición del plano es recuperada desde una imagen en 2D. Con este método obtenemos los 6 grados de libertad del plano en coordenadas físicas con respecto a las coordenadas de la cámara (tres de traslación y tres de rotación).

PALABRAS CLAVE: imagen, coordenadas homogéneas, plano proyectivo, matriz de proyección.

1 INTRODUCCIÓN

El problema de detectar la posición y orientación de un objeto en relación a otro objeto y con base a su sistema de referencia, es una tarea difícil de realizar independientemente del sensor a utilizar. Uno de los sensores más completos es una cámara, la cual junto con un análisis adecuado de la imagen adquirida, puede proporcionar una gran cantidad de información. El problema real consiste en saber obtener la información a partir de la imagen captada por la cámara en el mundo real. En aplicaciones de localización y orientación de los sistemas basados en robots, ya sean terrestres, aéreos o submarinos el problema de detectar la posición del vehículo, respecto de su objetivo, ha sido abordado ampliamente utilizando diferentes métodos (sistemas basados en central inercial, sensores IR, acelerómetros, visión, entre otros). La selección adecuada debe estar en función de la aplicación, sin embargo el problema de la auto localización requiere de varios sensores, por ejemplo un acelerómetro nos proporciona las aceleraciones en los tres ejes (x, y, z), por consiguiente para obtener la posición se tiene que integrar dos veces, lo cual genera un error que con el pasar del tiempo aumenta cada vez más, haciéndose necesario utilizar métodos para eliminar el error como seria implementar un filtro Kalman. También se tiene el inconveniente de que el sistema por si sólo no determina la posición inicial con respecto a su sistema de referencia. El problema de la ubicación del objeto, también se podría abordar con un sensor GPS (Global Positioning System). El inconveniente aquí es que el sistema GPS tiene un error medido en metros, lo cual para vuelos en interiores no es adecuado, además de que para detectar o medir las orientaciones sería necesario hacer uso de algún otro sensor. Por otro

ISSN 2395-8499

lado, se podrían utilizar un conjunto de sensores para auto localizar el sistema pero se tiene el inconveniente de que el hardware se complicaría al igual que el software o firmware.

La visión por computadora puede resolver estos problemas abordándolo desde distintos puntos de vista; en la extracción de la información visual contenida en la imagen.

Para poder obtener los parámetros de rotación y traslación de un objeto a partir de una imagen, es necesario obtener la matriz que nos relacione los puntos en el plano físico (mundo real) con los puntos en la imagen. Ya que esta matriz contiene toda la información de las relaciones entre los puntos de ambos planos (plano de la imagen y plano en el mundo físico), para determinar los parámetros que nos interesan, será necesario interpretar los datos de esta matriz.

2 Obtención y análisis de la imagen

2.1 Obtención de la imagen

Una imagen digital puede definirse como una función bidimensional $f(x, y)$ donde x y y son coordenadas en el plano y la amplitud f es llamada intensidad o nivel de gris en ese punto. Cuando *(x, y)* y *f* son todos finitos (cantidades discretas) se le denota a la función como imagen digital, en donde estos elementos son llamados *pixeles*, cada uno de los cuales con un valor y una posición particular.

Una imagen normalmente es representada como $I(x, y)$ [1], donde el valor de la intensidad I se obtiene por el indexado de las coordenadas x y y. El modelo más común de representación de la imagen es por medio de una matriz (1), tal que:

$$I(x,y) = \begin{bmatrix} I(1,1) & I(2,1) & \cdots & I(N,1) \\ I(1,2) & I(2,2) & \cdots & I(N,2) \\ \vdots & \vdots & \ddots & \vdots \\ I(1,M) & I(2,M) & \cdots & I(N,M) \end{bmatrix} \quad (1)$$

2.2 Modelo de la cámara

Al pasar de coordenadas 3D del mundo real a coordenadas 2D de la imagen, es un proceso de proyección en el cual se pierde una dimensión. La manera usual de modelar este proceso es por la proyección central, en el cual un rayo es trazado desde un punto en el espacio 3D a través de un punto fijo en el espacio (centro de proyección C) [2]. Este rayo intersecta un plano al cual se le llama el plano de la imagen (Figura 1).

ISSN 2395-8499

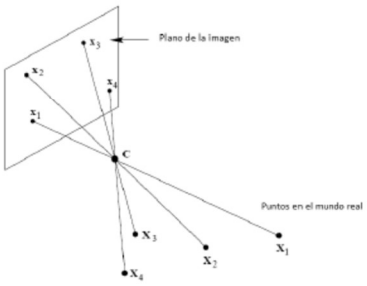


Figura 1. Modelo de proyección de la imagen.

Este modelo concuerda con el modelo simple de una cámara, ignorando los efectos como foco y espesor del lente. Una manera de interpretar el plano proyectivo P^2 (espacio proyectivo de dos dimensiones) es un conjunto de rayos en R^3(espacio euclidiano de 3 dimensiones). El conjunto de todos los vectores $k(x_1, x_2, x_3)^T$ con k variante, puede formar un rayo a través del origen. Así como un rayo puede ser pensado como la representación de un único punto en P^2. En este modelo, las líneas en P^2 son planos pasando a través del origen. Los puntos y líneas pueden ser obtenidas intersectando sus conjuntos de rayos y planos por el plano Π en $x_3 = 1$, como se ilustra en la Figura 2. Los rayos representan puntos ideales y el plano representado por l_∞ es paralelo al plano x_3.

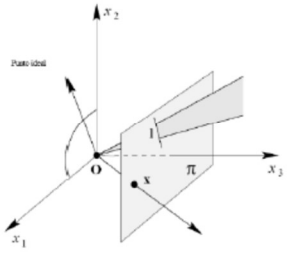


Figura 2. Puntos y líneas de P^2.

2.3 Transformaciones proyectivas

Una transformación proyectiva planar es una transformación lineal de vectores homogéneos de 3 dimensiones representada por una matriz no singular de 3x3 de la siguiente manera:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \quad (2)$$

ISSN 2395-8499

donde x_i con $i = 1,2,3$ son las coordenadas del plano de la imagen, X_i con $i = 1,,2,3$ son las coordenadas del plano en el mundo real y h_{ij} con $i = 1,2,3$ y $j = 1,2,3$ son los elementos de la matriz de transformación, lo cual en su forma reducida es $x = HX$.

2.4 Mapeo entre planos

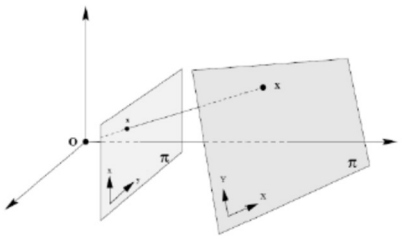


Figura 3. Mapa de la proyección central de puntos en un plano a puntos en otro plano.

Como se observa en la Figura 3, la proyección a lo largo del rayo a través de un punto común (el centro de proyección) define un mapeo desde un plano a otro. Este mapeo punto a punto preserva las líneas, es decir, una línea en un plano es mapeada a una línea en el otro plano. Si un sistema de coordenadas es definido en cada plano y los puntos son representados en coordenadas homogéneas, entonces el mapeo de la proyección central puede ser expresada por $x = HX$ donde H es una matriz de 3X3 no singular.

2.5 Cálculo de la matriz H de transformación 2D a 2D

Para el cálculo de la matriz de transformación H que nos relaciona el plano de la imagen con el plano del mundo físico, se utiliza la ecuación representada por (3)

$$x_{uv} = HX_{xy} \tag{3}$$

donde x representa un punto en las coordenadas del plano de la imagen y X representa el mismo punto pero en coordenadas del plano en el mundo físico. Esta ecuación (3) en su forma explícita queda:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Conociendo los valores de las coordenadas en el plano físico y los valores en el plano de la imagen y resolviendo el sistema (4) se puede calcular los valores de la matriz de proyección [3].

ISSN 2395-8499

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -u_2 2X_2 & -u_2Y_2 & -u_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -v_2X_2 & -v_2Y_2 & -v_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -u_3 2X_3 & -u_3Y_3 & -u_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -v_3X_3 & -v_3Y_3 & -v_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -u_4X_4 & -u_4Y_4 & -u_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -v_4X_4 & -v_4Y_4 & -v_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \quad (4)$$

2.6 Cálculo de los valores extrínsecos de la cámara

Una vez obtenida la matriz de transformación H, el paso siguiente es obtener la matriz de rotación y vector de traslación (valores extrínsecos) para obtener los valores de la posición y rotación del plano [4].

Se sabe que

$$K^{-1}H = \lambda R \quad (5)$$

donde K es la matriz de parámetros intrínsecos de la cámara, R es la matriz de rotación y λ es una constante [5].

La ecuación (4) se puede expresar como.

$$K^{-1}H = \lambda[r_1|r_2|t] \quad (6)$$

donde r_1 y r_2 son vectores en R^3 y son los vectores ortonormales del plano físico y t es el vector de traslación en R^3. Sabiendo esto, tenemos que:

$$\|r_1\| = \|r_2\| = 1 \quad (7)$$

Por lo tanto tenemos que

$$K^{-1}[h_1 \mid h_2 \mid h_3] = [\lambda r_1|\lambda r_2|\lambda t] \quad (8)$$

Utilizando las propiedades de la norma de vectores, tenemos que

$$\|k^{-1}h_1\| = \|k^{-1}h_1\| = \lambda \quad (9)$$

Utilizando la igualdad de la ecuación (8) para encontrar el valor de λ , nos queda la expresión siguiente:

$$Rt = \frac{k^{-1}H}{\lambda} \quad (10)$$

2.7 Cálculo de la matriz de proyección de 3D a 2D

Para realizar el cálculo de esta matriz sólo se necesita obtener el vector perpendicular a r_1 y r_2 por lo tanto.

$$r_3 = r_1 \times r_2 \tag{11}$$

Quedándo la matriz de proyección de la cámara de la forma siguiente

$$P = [r_1|r_2|r_3|t] \tag{12}$$

Esta matriz contiene toda la información de las rotaciones y traslaciones del plano físico con respecto al plano de la cámara.
La matriz *P* se multiplica por la matriz de parámetros intrínsecos de la cámara; lo que da como resultado una matriz de proyección de la cámara de la forma:

$$P_K = KP \tag{13}$$

2.8 Recuperación de los datos del plano físico

Una vez obtenida la matriz *P* de la ecuación (12) es necesario obtener los datos de traslación y rotación del plano físico.
Los valores de la traslación se obtienen directamente del vector *t*, esto quiere decir que se cumple:

$$t = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{14}$$

Por otro lado para recuperar los valores de rotación del plano físico se utilizan las matrices de rotación euclidiana donde los vectores r_i con *i=1,2,3* de la matriz de rotación quedan definidas de la siguiente manera

$$r_1 = \begin{pmatrix} -\cos(\theta_z)\cos(\theta_y) \\ \cos(\theta_z)\,sen(\theta_y)sen(\theta_x) + \cos(\theta_x)\,sen(\theta_z) \\ -sen(\theta_z)sen(\theta_x) - \cos(\theta_z)\cos(\theta_x)\,sen(\theta_y) \end{pmatrix} \tag{15}$$

$$r_2 = \begin{pmatrix} -\cos(\theta_y)\,sen(\theta_y) \\ sen(\theta_z)sen(\theta_y)sen(\theta_x) - cos(\theta_z)cos(\theta_x) \\ cos(\theta_z)sen(\theta_x) + cos(\theta_x)sen(\theta_z)sen(\theta_y) \end{pmatrix} \tag{16}$$

ISSN 2395-8499

$$r_3 = \begin{pmatrix} sen(\theta_y) \\ -cos(\theta_y)sen(\theta_x) \\ cos(\theta_y)cos(\theta_x) \end{pmatrix} \quad (17)$$

Siendo θ_x, θ_y y θ_z los ángulos de rotación del plano físico con respecto de los ejes x, y y z respectivamente. A partir de la ecuación (17) se obtiene

$$\theta_y = sen^{-1}(P_{13})$$
$$\theta_x = \text{sen}^{-1}\left(\frac{P_{11}}{\cos(\theta_y)}\right)$$
$$\theta_z = \cos^{-1}\left(\frac{P_{11}}{\cos(\theta_y)}\right) \quad (18)$$

Estas ecuaciones nos regresan la información de la posición lineal y la rotacional del plano físico.

3 Implementación del hardware

Lo anteriormente descrito en la sección I se ha llevado a su implementación con el fin de comprobar de forma experimental la obtención de la información. El diagrama a bloques en la Figura 4 muestra como está constituido el sistema completo.

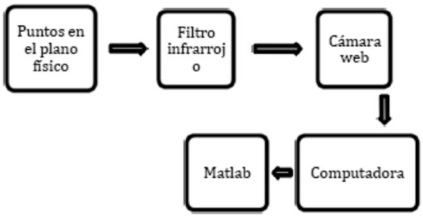


Figura 4. Diagrama a bloques del sistema completo.

Para generar los puntos en el espacio 3D (plano físico) se construyó un dispositivo experimental, con base en un marco de material ABS el cual tiene 4 led´s infrarrojos ubicados en las esquinas de un cuadrado, los cuales están separados por una distancia de 10 cm como lo muestra la Figura 5. De esta manera generamos cuatro puntos sobre un plano, el cual permitirá adquirir una imagen que pueda ser rotada-desplazada en los tres ejes y detectar las variables buscadas.

ISSN 2395-8499

Figura 5. Disposición de los led´s infrarrojos en un marco de material ABS

Figura 6. Filtro de luz infrarroja

El siguiente elemento que se utiliza es un filtro de luz infrarroja (Figura 6) para discriminar toda la luz que llega del ambiente a la cámara web y únicamente dejar pasar la luz infrarroja proveniente de los 4 puntos (led's).

En la Figura 7 se puede ver el dispositivo experimental con los 4 led's infrarrojos tomados con la cámara sin el filtro infrarrojo, en contraste con la Figura 8 en la que se puede observar el mismo dispositivo experimental pero esta vez con el filtro infrarrojo.

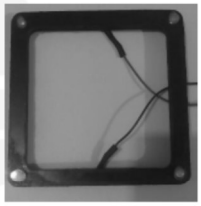


Figura 5. Imagen obtenida desde la cámara sin filtro infrarrojo.

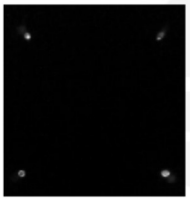


Figura 6. Imagen obtenida desde la cámara con filtro infrarrojo.

La cámara que se utiliza para las pruebas es la cámara Web Perfect Choice PC-320449 [6] (Figura 9). Aunque esta cámara no posee las mejores características técnicas, consideramos que es suficiente como una primera aproximación para obtener resultados experimentales que permitan comprobar la teoría.

Figura 7. Cámara utilizada para las pruebas.

ISSN 2395-8499

Para la calibración de la cámara se hizo uso de la herramienta Camera Calibration Toolbox de Matlab [7].

La programación del algoritmo basado en cuatro puntos se realizó en código de MatLab R2014 [8], así como la adquisición de la imagen.

4 Implementación del Software

En la Figura 10 se presenta el diagrama de flujo del software que se implementó para la detección del plano y el cálculo de posición y orientación.

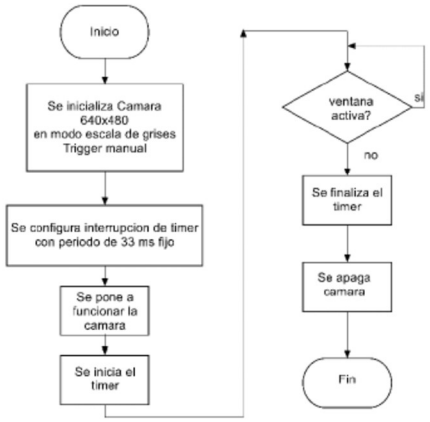


Figura 8. Diagrama de flujo del programa principal.

El primer paso es la configuración de la cámara y adquisición de las imágenes para lo cual se utilizó la herramienta Image Acquisition Toolbox de MatLab [8]. Con esta herramienta se configuro la cámara para que capture las imágenes con una resolución de 640X480 pixeles. También se configuró para que tome las imágenes en escala de grises y que se capture la imagen en forma manual. Posteriormente se configura un timer para mandar una interrupción cada 33 ms ya que la cámara toma imágenes a 30 cuadros por segundo y esta velocidad es fijada por el fabricante.

En la Figura 11 se muestra el diagrama de flujo de la subrutina que se ejecuta cada vez que la interrupción del timer es activada.

ISSN 2395-8499

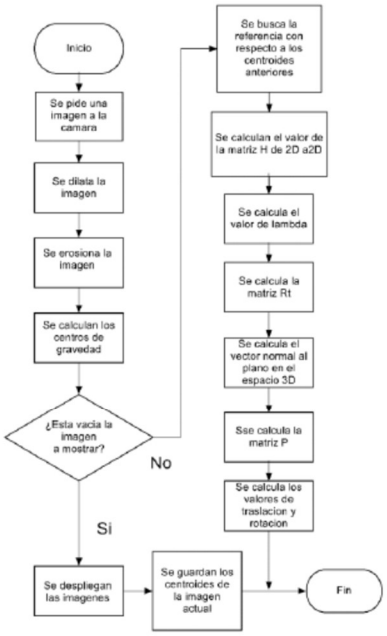


Figura 9. Diagrama de flujo del proceso para ejecutar la interrupción del timer.

En este programa una vez que se adquiere la imagen se procede a realizar un preprocesamiento, con el fin de eliminar ruido, de tal manera que en el procesamiento posterior se obtengan mejores resultados. Este preprocesamiento consiste de varias etapas: binarización, dilatación, erosión, etiquetado, para obtener finalmente el cálculo de los centroides de los 4 puntos.

4.1 Binarización de la imagen

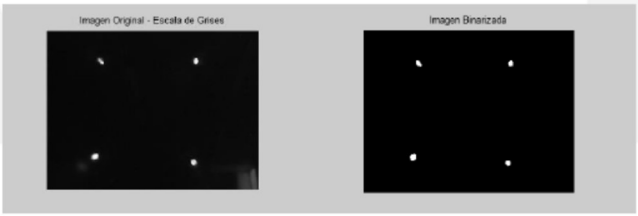


Figura 10. Comparación de imagen en escala de grises e imagen binarizada.

La binarización de imágenes es una técnica del procesamiento de imágenes que consiste en un proceso de reducción de la información de una imagen digital a dos valores: 0 (negro) y 1(blanco) [9].
Esta técnica consiste en comparar cada pixel de la imagen con un determinado umbral (valor límite que determina si un pixel tendrá un valor de "0" ó "1" en su equivalente en color). Los valores de la imagen que sean mayores que el umbral toman un valor "1" (blanco), el resto de pixeles toman valor "0" (negro) (Figura 12).

4.2 Dilatación y erosión de imágenes

La erosión y la dilación son las operaciones morfológicas más usadas en procesamiento de imágenes [7].
La función de dilación es tomar cada pixel del objeto (con valor "1") y poner al valor "1" todos aquellos pixeles pertenecientes al fondo (background) que tienen una conectividad C (C=4, C=8, etc) con el píxel del objeto. En pocas palabras, asignar el valor "1" a los pixeles del fondo, vecinos a los pixeles del objeto.

En nuestro caso la dilatación se usa para agrandar los puntos encontrados y eliminar partes de los puntos que estén separadas del punto principal, formando un solo punto, evitando con esto, errores en la posterior cuantificación de puntos.

La erosión consiste en asignar un valor "0" a cada pixel del objeto que tiene una conectividad C con los pixeles del fondo. En otras palabras, poner a "0" los pixeles del objeto vecinos a los píxeles del fondo. Esto se hace después de la dilatación para regresar a su tamaño normal los puntos y de esa forma tener sólo el número correcto de puntos (Figura 13).

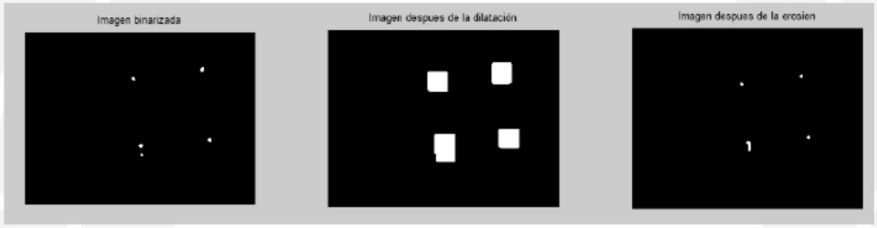


Figura 11 Izquierda: Imagen binaria original (5 puntos). Centro: Puntos dilatados. Derecha: Puntos erosionados (4 puntos).

4.3 Etiquetado

Es una técnica que asigna una etiqueta a cada componente conexo para obtener al final una región [10]. Básicamente el algoritmo recorre una imagen binaria de izquierda a derecha y de arriba hacia abajo en busca de píxeles con valor "1" que se encuentren conectados entre sí en un rango que puede ser C=4, C=8, etc. y les asigna una etiqueta construyendo de esta manera regiones. Esta técnica, no es tan susceptible a tener errores si los caracteres están rodeados o de líneas que cubran o rodeen a los caracteres. Además no requiere que la imagen se encuentre alineada con respecto al eje para ser efectiva lo cual es de gran utilidad debido a que no siempre se puede tener correctamente alineado el dispositivo móvil con la imagen como sería con un escáner de sobremesa o plano.

Una vez que en la imagen ya se cuenta con cuatro puntos y uno de ellos es un punto de referencia sin ruido, se procede a realizar un procesamiento, iniciando con la obtención de lo centroides de los puntos. Cabe aclarar que el punto de referencia no se pierde durante estas etapas de preprocesamiento de la imagen.

4.4 Obtención de Centroides

Ya que se tienen los 4 puntos etiquetados, es necesario encontrar sus coordenadas en la imagen, para esto se determinan los centroides [1] de los puntos u objetos en la imagen.
El centroide o centro de masa de un objeto es el punto en donde, por su geometría, se encuentra concentrada la masa del objeto. El centroide $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y})$ de un objeto binario se calcula como el punto medio aritmético de las coordenadas en la dirección x y y, tal que:

$$\bar{x} = \frac{1}{Area(O)} \sum_{(x,y)\epsilon O} x \qquad \bar{y} = \frac{1}{Area(O)} \sum_{(x,y)\epsilon O} y \tag{19}$$

Donde un objeto O de una imagen binaria puede ser interpretado como una distribución de puntos de valor uno $\boldsymbol{x_i} = (x_i, y_i)$ en una matriz bidimensional, esto es

$$O = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N}\} = \{(x_1, y_2), (x_2, y_2), \dots, (x_N, y_N)\} \tag{20}$$

El área de un objeto O puede ser calculada mediante el número de pixeles que lo conforman, esto es:

$$Area(O) = N = |O| \tag{21}$$

y

$$\sum_{(x,y)\epsilon O} \tag{22}$$

representa la sumatoria de todos los puntos que están dentro del área del objeto.

Por último, se realizan los cálculos de las matrices (4), (9), (10), (11), (12) y (18) como lo indica el diagrama de flujo de la Figura 11. A partir de estas se obtienen posiciones lineales y las rotaciones del plano.

5 Resultados

El algoritmo fue probado con imágenes fijas y en tiempo real. Las medidas que arroja el programa están en milímetros para las medidas de traslación y en grados para las medidas de los ángulos de rotación. Los valores reales se obtuvieron de forma manual para realizar la comparativa.

La Fig. 14 muestra la disposición de los elementos físicos involucrados en la plataforma experimental, con los elementos descritos en la sección II y la programación en MatLab de los algoritmos descritos en la sección III. El dispositivo de los 4 led's infrarrojos se colocó frente a la cámara en forma paralela, logrando obtener el plano con 4 puntos infrarrojos a una distancia de 50 cm del plano de la imagen.

ISSN 2395-8499

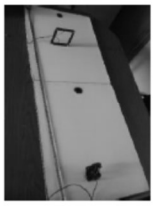


Figura 12. Plataforma experimental.

La figura 15 muestra las imágenes obtenidas del programa hecho en MatLab. Los resultados obtenidos por el algoritmo se presentan en la tabla 1.

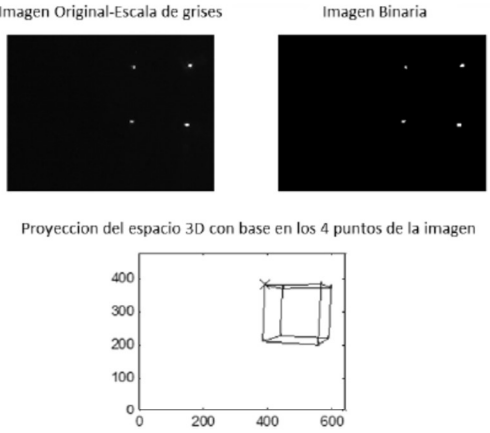


Figura 13. Calculando la posición y orientación de un plano con respecto a 4 putos en el plano.

Tabla 1 Datos experimentales

	Datos reales	*Datos obtenidos del algoritmo*
X	30mm	29.61mm
Y	60mm	62.03mm
Z	500mm	501.5mm
θ_x	2°	1.24°
θ_y	24°	25.572°
θ_z	90°	92.335°

Como se puede observar los datos de la Tabla 1 corresponden a las posiciones y ángulos del plano que se encuentra en el mundo real con respecto del plano de la imagen. Estos datos que corresponden con la realidad presentan un error máximo de ±2mm que depende de las distancias entre los pixeles y la forma de los 4 puntos en la imagen que en consecuencia afectan el cálculo de los centroides.

6 CONCLUSIONES

El algoritmo que se ha desarrollado cumple con los objetivos planteados al poder realizar el cálculo de los 6 grados de libertad de un plano que contiene 4 puntos. Al utilizar el filtro infrarrojo se logró evitar utilizar un algoritmo de preprocesamiento de imagen más complejo como lo sería la detección de esquinas o detección de formas entre otros, ayudando con esto a que el algoritmo sea más rápido y se ejecute en tiempo real.

Es importante también mencionar que este algoritmo depende de una buena calibración de la cámara, aunque ésta no tiene que ser muy costosa.

Con el método presentado en este trabajo se puede ver que son sólo necesarios 4 puntos coplanares para determinar los 6 grados de libertad de un objeto abriendo la posibilidad de cerrar lazos de control sin utilizar ningún otro sensor de orientación y traslación (central, inercial, GPS, ultrasónico, entre otros).Para problemas de 3D de Visual Servoing son necesarias como mínimo 2 cámaras para determinar la posición en 3D, esto quiere decir que es necesario el procesamiento de imágenes de dos o más cámaras.

A futuro ya que este algoritmo no requiere de mucho procesamiento podría adaptarse a un microcontrolador o sistema embebido para así poderlo montar en robots móviles autónomos, haciendo con esto un sensor de grandes prestaciones.
El trabajo desarrollado aquí contribuye de manera importante, en el área de visión e instrumentación, ya que no existe un sensor que detecte estas variables físicas y con la precisión que se menciona.

REFERENCIAS

[1] E. Cuevas, D. Zaldívar, M. Pérez. "Procesamiento Digital de Imágnes con MATLAB y Similink". Impreso por Alfaomega Ra-Ma. 2010.

[2] R. Hartley, A. Zisserman, "Multiple View Geometry in Computer Vision" .Cambridge University Press, 2003.

[3] J. L. Mundy, A Zisserman. "Geometric Invariance in Computer Vision". Cambridge, 1992.

[4] D. Forsyth, J. Ponce, "Computer Vision, a modern approach". Prentice Hall, 2011.

[5] M. Shah, "Fundamentals of Computer Vision", University of Central Florida. 1997.

[6] Camara. Perfect Choice PC-320449. Junio 2015. Disponible en: http://pub.perfectchoice.me/Publicaciones/WebActual/manuales/PC-320449.pdf

[7] Camera Calibration. Junio 2015. Disponible en: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

[8] MatLab . Junio 2015. Disponible en : http://www.mathworks.com/help/matlab

[9] Kenneth, R. Castleman, "Digital Image Processing", Prentice Hall, 1995.