UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo e implementación de algoritmo de visión por computador en una mesa de pruebas para la experimentación con micro-robots móviles en robótica de enjambre

Trabajo de graduación presentado por André Josué Rodas Hernández
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2019

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

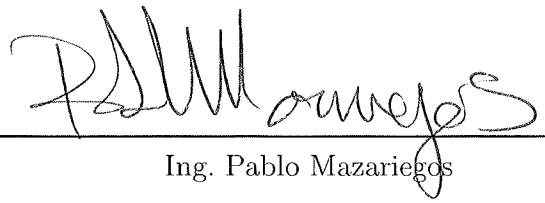
Desarrollo e implementación de algoritmo de visión por computador en una mesa de pruebas para la experimentación con micro-robots móviles en robótica de enjambre

Trabajo de graduación presentado por André Josué Rodas Hernández
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2019

Vo.Bo.:

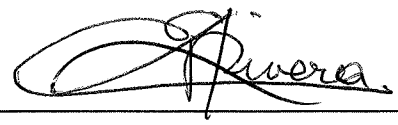
(f) 

Ing. Pablo Mazariegos

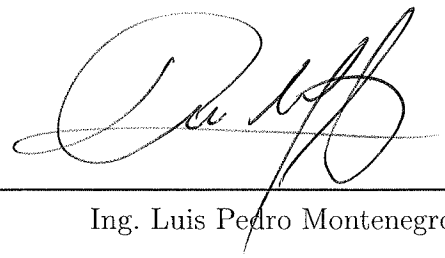
Tribunal Examinador:

(f) 

Ing. Pablo Mazariegos

(f) 

Dr. Luis Rivera

(f) 

Ing. Luis Pedro Montenegro

Fecha de aprobación: Guatemala, 4 de diciembre de 2018.

Prefacio

Son varias personas que han contribuido al proceso y culminación de este trabajo de graduación. En primer lugar, quiero agradecer a Dios por cuidarme, darme las fuerzas e inteligencia durante todo este tiempo. Por colocar a las personas indicadas para mi crecimiento intelectual y espiritual.

Gracias a mis padres por su trabajo constante y amor incondicional. El apoyarme económicamente y emocionalmente, ellos fueron el ejemplo durante todos estos años y les quiero dedicar este trabajo.

Gracias a mis profesores. A Miguel Zea por proponer este proyecto, mantener una revisión constante para entregar un trabajo de calidad, el exigir lo mejor de mi persona y creer en mis habilidades desde el inicio hasta la culminación de este proyecto. Al igual que Pablo Mazariegos quien guió el trabajo desde sus inicios, puliendo cada fase del proyecto para entregar un trabajo de excelencia, que no podría haberse logrado sin su amplio conocimiento en electrónica y diseño mecánico.

Finalmente gracias a la Universidad del Valle de Guatemala y a mis amigos por crear un ambiente ideal para la creación de grandes proyectos a partir de pequeñas ideas.

Índice

Lista de figuras

Lista de cuadros

Resumen

El proyecto plantea fabricar una mesa de pruebas para micro-robots que permita evaluar aplicaciones de robótica de enjambre para tal efecto, se estará desarrollando un estándar de reconocimiento para los micro-robots que se encuentren en la mesa de pruebas a través de una cámara montada, mediante algoritmos de visión por computador se obtendrá el identificador, posición y ángulo de cada micro-robot.

Antes de comenzar el diseño e implementación de los identificadores y el algoritmo de visión por computador, se buscaron diferentes métodos para la detección en mesas de pruebas. Uno de esos cambios fue replantear el diseño de los identificadores de tres dimensiones *ArUco Fmarkers* de la librería OpenCV con el fin de diseñar identificadores para robots planares (reconocimiento solamente de dos dimensiones) con el objetivo de reducir la complejidad del algoritmo. Para ello se proponen diferentes diseños y en paralelo se llevaron a cabo los algoritmos de visión de computador que permiten la obtención de información del identificador. Todos los algoritmos se desarrollaron con software gratuito, en busca de la replicabilidad de ejecución.

Posteriormente el algoritmo de reconocimiento se unirá con el protocolo de comunicación multi-robot realizado por el estudiante Marlon Castillo de la Universidad del Valle, para finalmente diseñar, fabricar y poner en funcionamiento la mesa de pruebas. Este banco de pruebas busca ser una solución más simplificada a los modelos de investigación multi-robot de otras universidades, que resultan ser costosos, complejos, lentos de desarrollar y operar; esto mediante un diseño eficiente, intuitivo, de bajo costo y replicable.

Abstract

The project proposes to manufacture a test table for micro-robots that allow the evaluation of swarm robotics applications. For this purpose, it will be developing a standard of recognition for the micro-robots that are in the test table through a mounted camera. Using computer vision algorithms the identifier information is extracted. This information includes the position, angle and robot number for each micro-robot.

Before starting the design and implementation of the identifiers and the algorithm of vision by computer, different methods were sought for the detection in tables of tests. One of these changes was to rethink the design of the three-dimensional identifiers *ArUco markers* from the OpenCV library to design identifiers for planar robots (recognition only in two dimensions) with the aim of reducing the complexity of the algorithm. For this purpose, different designs are proposed and in parallel the computer vision algorithms were carried out that allow the obtaining of information of the identifier. All the algorithms were developed with free software, looking for the replication of execution.

Later the recognition algorithm will join the multi-robot communication protocol conducted by student Marlon Castillo of the University of Valle in order to finish the design, manufacture and operate the test table. This test bench seeks to be a more simplified solution to the multi-robot research models of other universities, which are expensive, complex, slow to develop and operate; this is an efficient, intuitive, low-cost, replicable design.

CAPÍTULO 1

Introducción

El control coordinado de sistemas multi-robot ha recibido atención significativa durante la última década, para resolver una amplia variedad de tareas, desde monitoreo ambiental para manejo de materiales colectivos hasta robótica de enjambre; utilizando una serie de algoritmos de control y colaboración colectiva.

A pesar de estos avances sigue siendo un proceso lento y complejo la simulación a partir de la teoría. Por ello el despliegue en tiempo real de información es vital para el análisis e investigación multi-robot que es cada vez más complejo simular fielmente los problemas asociados con múltiples robots que realizan diferentes tareas coordinadas, específicamente en la robótica de enjambre.

El banco de pruebas está diseñado para abordar esta brecha entre la teoría y la simulación práctica, permitiendo plantear una serie de diferentes preguntas científicas con base a algoritmos de coordinación.

Buscando el acceso remoto, eficiencia, precisión, exactitud y diseño económico, se plantea una estandarización de reconocimiento para cualquier micro-robot planar utilizado en la mesa de pruebas, para ello se establece un identificador capaz de proporcionar la información y diseño necesario para que algoritmo de visión por computador sea capaz de extraer; en consecuencia se analizan diferentes técnicas de filtrado y detección de bordes que permiten aislar el identificador en cuestión y leer las características del patrón.

Por último, se diseñó la mesa de pruebas con una cámara montada que permita correr la interfaz programada, los datos se estarán actualizando en una base de datos cada cierto periodo establecido y con ello el módulo de comunicación entre computador y micro-robot podrá ejecutarse y así realizar algoritmos en la mesa de pruebas.

CAPÍTULO 2

Antecedentes

2.1. El *Robotarium*

El *Robotarium* es una plataforma remota y accesible, creada con el propósito del estudio de robótica de enjambre. Esta fue desarrollada por estudiantes de Georgia Institute of Technology en el año 2016 [1]. El impulso detrás de Robotarium es que los bancos de prueba de múltiples robots constituyen una parte integral y esencial del ciclo de investigación de múltiples robots. Sin embargo, son costosos, complejos y lentos de desarrollar, operar y mantener. Estas restricciones de recursos, a su vez, limitan el acceso para grandes grupos de investigadores y estudiantes, que es lo que el Robotarium está remediando al proporcionarles a los usuarios acceso remoto a un centro de pruebas avanzado.

Este documento detalla el diseño y el funcionamiento del Robotarium y analiza las consideraciones que se deben tomar al hacer que el hardware complejo sea accesible de forma remota. En particular, la seguridad debe integrarse en el sistema que ya se encuentra en la fase de diseño sin limitar excesivamente los programas de control coordinado que los usuarios pueden cargar y ejecutar, lo que requiere rutinas de seguridad mínimamente invasivas con garantías de rendimiento comprobables. [1]

Entre las consideraciones de diseño se plantean los siguientes puntos:

- Capaz de ser replicada una versión de bajo costo, con diseño de robots *open-source*.
- Interacción intuitiva con los datos recolectados a travez de una web pública, facilitando el envío de código y recuperación de datos.
- Permitir cambio sin interrupciones entre el desarrollo en la simulación y ejecución.
- Minimizar el costo y complejidad de los robots, mediante carga automática y seguimiento.
- Integrar medidas de seguridad y protección mediante prevención de colisiones.

2.2. *Zooids*

Los Zooids consisten en la investigación acerca de las interfaces de usuario de enjambre, una nueva clase de interfaces humano-computadora compuesta por muchos robots autónomos que manejan tanto la pantalla como la interacción. Los *Zooids* son una plataforma de open hardware con código abierto para el desarrollo de interfaces de enjambre en una mesa de pruebas. [2]

La plataforma consiste en una colección de micro robots móviles sobre ruedas diseñados con un tamaño de 2,6 cm en diámetro, una estación base de radio, un proyector de luz estructurada DLP de alta velocidad para seguimiento óptico y un el software para el desarrollo y control de aplicaciones.

El potencial de las interfaces de usuario de enjambre de mesa a través de un conjunto de escenarios de aplicación desarrollados con Zooids y se discute consideraciones de diseño generales exclusivas de las interfaces de usuario de enjambre, entre ellas:

- La primera plataforma de hardware y software de código abierto para experimentar con interfaces de usuario de enjambre de mesa
- Permite la creación de múltiples escenarios para ilustrar las posibilidades sin precedentes que ofrece el sistema y el usuario de sobremesa
- Discute principios generales de diseño y desafíos para las interfaces de usuario de enjambre

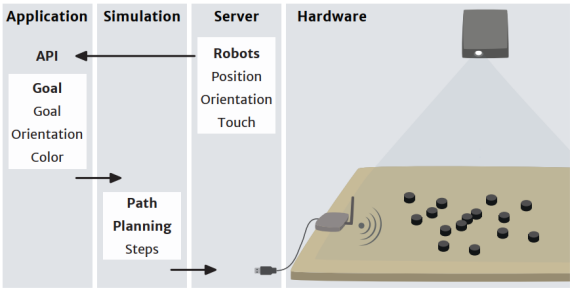


Figura 1: Arquitectura de software de los Zooids. [2]

2.3. *Kilobots*

El robot Kilobot, diseñado específicamente para operar en un gran colectivo. Si bien el Kilobot es relativamente simple en comparación con otros robots, al compararse con otros diseños del mercado, este es capaz de ejecutar SDASH, así como otros comportamientos colectivos. Esta simplicidad permite un bajo costo, que combinado con operaciones escalables, permite colectivos mucho más grandes que los disponibles durante la investigación. [3]

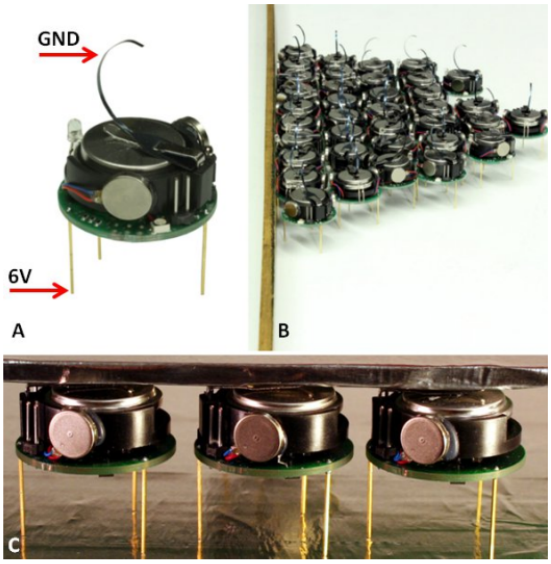


Figura 2: Kilobots. [3]

2.4. OpenCV: *ArUco marker*

Sistema de marcador *fiducial* especialmente apropiado para la localización de la cámara en aplicaciones de realidad aumentada o robótica. En lugar de emplear un conjunto predefinido de marcadores, se ha propuesto un método general para generar diccionarios configurables en tamaño y número de bits.

El algoritmo se basa en una búsqueda probabilística que maximiza dos criterios: las distancias entre marcadores y el número de transiciones de bits. Además, se ha derivado la distancia teórica máxima entre marcadores que puede tener un diccionario con creadores de cuadros. El documento también propone un método automático para detectar los marcadores y corregir posibles errores. En lugar de utilizar bits redundantes para la detección y corrección de errores, el enfoque se basa en una búsqueda en el diccionario generado. [4]

Figura 3: Ejemplos de diferentes identificadores AruCo. [4]

2.5. Librería *OpenCV*

OpenCV (Open Source Computer Vision Library) es una biblioteca de visión de computadora y *machine learning*. OpenCV fue construido para proporcionar una infraestructura común para aplicaciones de visión de computadora y para acelerar el uso de la percepción en máquinas de productos comerciales. Al ser un producto con licensia BSD, OpenCV facilita a las empresas utilizar y modificar el código [5].

La biblioteca cuenta con más de 2500 algoritmos optimizados. Estos algoritmos se pueden usar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en vídeos, rastrear movimientos de cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, entre otros.

OpenCV tiene más de 47 mil personas de usuarios es su comunidad. La biblioteca se usa ampliamente en compañías, grupos de investigación y por organismos gubernamentales.

CAPÍTULO 3

Justificación

Es esencial estudiar la rama de robótica de enjambre fabricando una plataforma que permita la recolección de datos de forma rápida y concisa para su análisis posterior; acelerando el proceso de simulación en computador y comprobando de manera física los algoritmos. La cama de pruebas permitirá crear micro-robots sin sensores de distancia entre otros módulos para el reconocimiento de su entorno, con lo cual se estará minimizando los costos, miniaturización y tiempo de fabricación de los micro-robots. Siendo esta una parte clave para el desarrollo de robótica tipo enjambre que se basa en la coordinación y desarrollo de un trabajo conjunto entre robots simples; esto ya que el comportamiento de muchos seres vivos depende de la interacción que se presenta con sus vecinos,como por ejemplo buscar alimento o evadir depredadores.

La aplicación de un algoritmo de reconocimiento se presenta como una alternativa a las mesas de pruebas multi-robot desarrolladas por otras instituciones de investigación que cuentan con una gran cantidad de sensores analógicos y digitales. No obstante, tienden a ser amplias, complejas y costosas.

La mesa por desarrollar contará con una sola cámara que mediante el procesamiento de imágen obtenga las posiciones y ángulos de cada robot; permitiendo un diseño económico, desmontable y replicable. El procesamiento de imágenes es una técnica moderna de análisis para la obtención de información mediante imágenes digitales.

Dependiendo del tipo de problema y su complejidad la aplicacion de una cama de pruebas resulta beneficiosa y eficiente, por cuestiones que se discutirán más adelante. De esta forma se estará yendo a la vanguardia con temas de visión por computador.

La cama de pruebas resulta ser una herramienta educacional e investigativa, permitiendo su utilización en una alta gama de carreras como biología, electrónica, mecatrónica y electrónica converger. Con ello se podrá competir a nivel internacional con otras universidades que desarrollan aplicaciones de robótica de enjambre y multi-robot.

Por el momento no existe en Latinoamérica una mesa de pruebas para la experimentación

con micro-robots. Con ello permitirá a una siguiente generación de investigadores estudiar el diseño de robots (tanto a nivel físico, como de sus conductas de comportamiento), para lograr que se puedan obtener patrones de comportamiento colectivo predeterminados por las interacciones entre robots y de estos con su entorno, siguiendo patrones de comportamiento como los que se observan en los insectos sociales, llamados inteligencia de enjambre.

En la actualidad, esta área de investigación tipo robótica de enjambre ha tomado gran relevancia debido a las variedad de temas no explorados o diferentes los diferentes enfoques que pueden ser tomados.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar e implementar un algoritmo de visión por computador para ser utilizado en una mesa de pruebas para la experimentación de micro-robots móviles en robótica de enjambre.

4.2. Objetivos específicos

- Utilizar procesamiento de imágenes para la detección de cada micro-robot en la mesa de pruebas, con su respectivo identificador, posición y ángulo con respecto a la cámara
- Desarrollar e implementar un estándar de generación de identificadores para el reconocimiento de cada uno de los micro-robots que se encuentren dentro de la mesa de pruebas
- Diseñar y fabricar la estructura de la cama de pruebas para micro-robots móviles
- Integrar el procesamiento de imágenes con el estándar de comunicación para permitir la experimentación en la cama de pruebas

CAPÍTULO 5

Alcance

Esta mesa de pruebas contará con un espacio de 130x90cm, con una cámara montada en la parte superior se estará obteniendo las posiciones de los códigos, siempre y cuando se esté capturando toda la superficie de la mesa a una resolución específica y se estarán auto calibrando las esquinas mediante software para que el usuario solo tenga que colocar la cámara en una posición que capture y enfoque la totalidad de la mesa.

Mediante una interfaz gráfica se le dará la oportunidad al usuario de variar el tamaño de su identificador en cm, para poder ser adaptado al tamaño de sus robots, siempre y cuando esté entre los rangos capaces de reconocimiento. Los robots tendrán como requisito ser planares debido a la naturaleza para la que fue programada la detección y se tendrán un máximo de 254 códigos diferentes.

Cada cierto periodo se estarán subiendo a una base de datos los valores de los robots para que la parte de comunicación pueda tener acceso a estas de manera independiente al programa de detección. Mediante una prueba estadística se verá si el algoritmo permite una medición precisa de valores, para ello se trabajará con enteros para los centímetros y grados.

CAPÍTULO 6

Marco teórico

6.1. Visión de computadora

Es una rama de la inteligencia artificial que desarrolla tanto la teoría como tecnologías necesarias para emular la percepción visual humana, su principal objetivo es el estudio de los procesos que permiten reconocer y localizar objetos en el ambiente. [6]

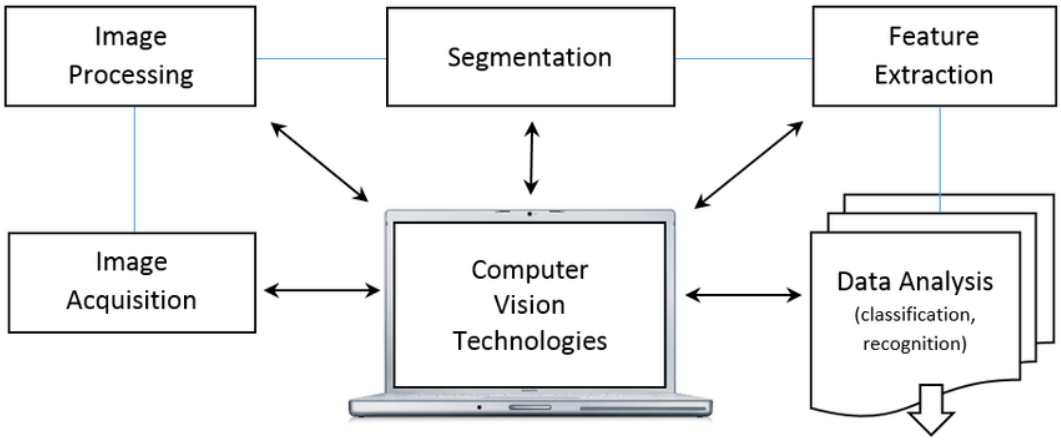


Figura 4: Diagráma de visión por computadora. [7]

6.2. La cámara digital

6.2.1. Cámara

La cámara de un sistema de visión por computadora es el dispositivo que recibe la luz reflejada por la escena y la utiliza para generar imágenes. Su elemento más importante es el sensor, formado por una capa fotosensible que transforma la luz incidente en una señal eléctrica. [6]

6.2.2. Píxeles

Un píxel se representa mediante un punto o cuadrado en la pantalla de visualización de un monitor de computadora. Los píxeles son los componentes básicos de una imagen o pantalla digital y se crean utilizando coordenadas geométricas. Según la tarjeta gráfica y el monitor de visualización, la cantidad, el tamaño y la combinación de colores de los píxeles varía y se mide en términos de la resolución de la pantalla.

6.2.3. Iluminación

La iluminación es la característica que utilizan las técnicas de visión por computadora para extraer información de los objetos de una escena [6].

La iluminación se puede realizar con luz natural o artificial. Las más comúnmente utilizadas son lamparas halógenas, lámparas fluorescentes, diodos LED y láser [6].

6.2.4. Óptica

La óptica es uno de los elementos más relevantes para cualquier sistema de visión. Tiene el objetivo de conectar la luz reflejada por los objetos de la escena en el sensor de la cámara para generar la imagen [6].

6.3. Modelos matemáticos y distorsión de una óptica

Existen una variedad de modelos matemáticos que definen la transformación que aplica la óptica de un sistema de visión a la luz que recibe. Se pueden mencionar los siguientes: modelo de agujero, modelo de lente delgada, modelo de lente gruesa y modelo *pinhole*. Utilizaremos el último modelo debido a su sencillez, siendo el más utilizado para eliminar distorsiones no lineales de ópticas. [6]

6.3.1. Modelo *pinhole*

En este modelo se considera que la óptica del sistema está formada por una única lente representada por un punto infinitesimal, denominado foco, a través del cual pasan los rayos de luz procedentes de la escena al sensor de la cámara. De esta forma, un punto (X,Y,Z) en el espacio se proyecta a un punto (x,y) en el plano de la imagen. El plano de la imagen se encuentra a una distancia f del foco, esta se conoce como distancia focal.

Las desventajas de este modelo es que no cubre la mayor parte de características y parámetros de las cámaras reales, tales como el *blur* debido al desenfoque y el control de la cantidad de luz incidente en el sensor[8].

La figura (5) muestra la representación de este modelo [7].

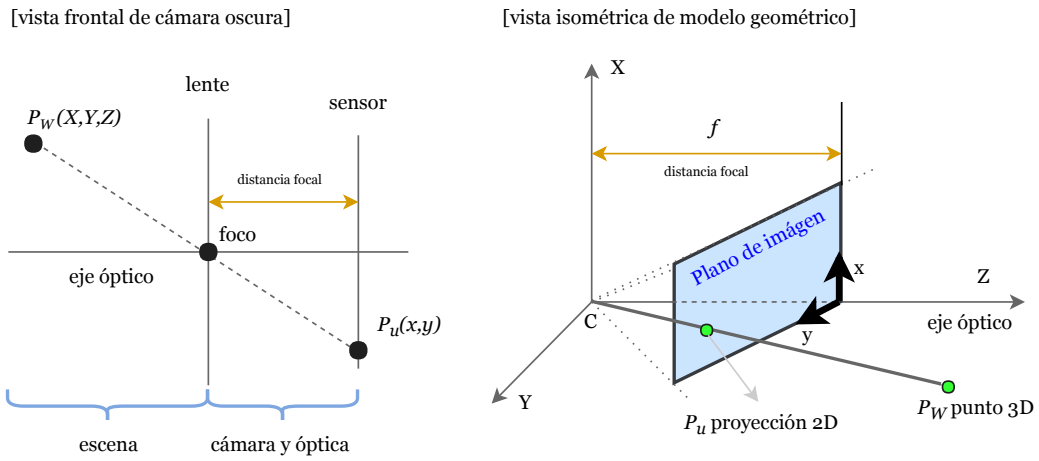


Figura 5: Modelo *pinhole*. [Elaboración propia]

Mediante la ecuación (1) en coordenadas homogéneas se encuentra la relación entre los puntos $(X,Y,Z)^\intercal$ y $(x,y)^\intercal$:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

esta puede ser escrita de forma matricial como:

$$\lambda \mathbf{m} = \mathbf{PM} \tag{2}$$

Donde $\mathbf{M} = [X\ Y\ Z\ 1]^\intercal$ y $\mathbf{m} = [x\ y\ 1]^\intercal$ las coordenadas homogéneas de $M = P_W$ y $m = P_u$ respectivamente, y la matriz de 3×4 denominada *matriz de proyección perspectiva* de la cámara. El factor λ es un factor de escala para mantener la igualdad y es igual a Z.

El modelo geométrico de la cámara oscura el orificio de ingreso de luz corresponde al centro óptico C ya que por él pasa todos los haces de luz que conforman el plano de la imagen. Sin embargo, debido a que el centro óptico se ubica entre el objeto y el plano de imagen, ocurre una inversión de la imagen, es decir $X/Z = -x/f$ y $Y/Z = -y/f$. Por lo que es necesario cambiar f por $-f$ [7], obteniendo así la siguiente expresión:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3}$$

6.3.2. Distorsión del lente

La curvatura del lente de las cámaras digitales introducen una deformación en la imagen. Debido a esta distorsión las líneas que en el espacio 3D son rectas ya no son vistas como tal en la proyección, sino como líneas curvas tal como lo ilustra la Figura (6). El efecto puede ser despreciable en el centro de la imagen, sin embargo es considerable en los extremos de la imagen, donde la normal de la superficie del lente no es paralela al eje óptico de la proyección [7].

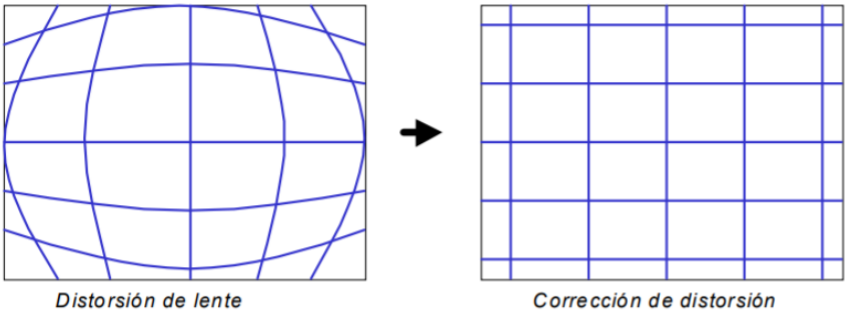


Figura 6: Rectificación de distorsión de lente. [7]

Generalmente la distorsión es modelada como una componente *radial* δ_r y otra *tangencial* δ_ϕ. Ilustradas en la Figura (7) en la que se observa un punto de distorsión nula (x_0, y_0), donde $x = x'$ e $y = y'$. Descomponiendo (x, y) en coordenadas polares (r, ϕ) con centro en (x_0, y_0) se puede escribir la distorsión como la suma de ambos componentes:

$$\delta_x(x, y) = \delta_r(r)\cos(\phi) - r\delta_\phi(\phi)\sin(\phi) \tag{4}$$

$$\delta_y(x, y) = \delta_r(r)\sin(\phi) + r\delta_\phi(\phi)\cos(\phi) \tag{5}$$

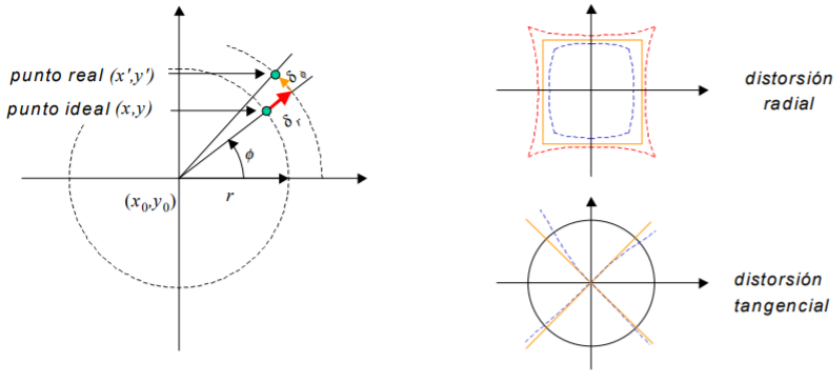


Figura 7: Modelo de distorsión de lente en componente radial y tangencial. [7]

6.3.3. Calibración de modelo *pinhole*

La calibración de la cámara *pinhole* requiere dos pasos y debe de mencionarse que esta no incluye los efectos no lineales de un lente de distorsión (i.e., radiales y tangenciales). El primer paso es encontrar la relación entre los puntos $(X_i, Y_i, Z_i)^T$ en el espacio de trabajo con los puntos correspondientes de la imagen $(x_i, y_i)^T$ [8] mediante la obtención de la matriz homogénea de dimensiones 3x4, denotada como A.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} K_x \\ K_y \\ K \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{7}$$

6.4. Coordenadas homogéneas

Las coordenadas homogéneas de un punto tridimensional con coordenadas cartesianas (X, Y, Z) se definen como el punto tetradimensional (kX, kY, kZ, k), donde k es una constante arbitraria distinta de 0 (habitualmente se suele utilizar $k = 1$). Por tanto, un punto P del espacio, representado mediante coordenadas cartesianas, se expresa en forma vectorial como:

$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{8}$$

y representando mediante coordenadas homogéneas se expresa como:

$$\mathbf{P}_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \tag{9}$$

Para aplicar ciertas transformaciones elementales sobre un punto del espacio es necesario que dicho punto esté expresado en coordenadas homogéneas.

6.5. Transformaciones elementales

En esta sección se presentan las transformaciones elementales que permiten aplicar técnicas básicas de visión de computadora en la imagen capturada [6].

6.5.1. Traslación

La traslación de un punto del espacio $\mathbf{P} = [X\ Y\ Z]^\intercal$ expresada mediante coordenadas homogéneas, en donde $\mathbf{P} = [X_0\ Y_0\ Z_0]^\intercal$ es el vector de traslación.

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{10}$$

6.5.2. Escalado

El escalado de un punto del espacio $\mathbf{P} = [X\ Y\ Z]^\intercal$ con los factores S_X, S_Y y S_Z en los ejes cartesianos de sus respectivos subíndices, expresado en coordenadas homogéneas expresa como:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_X & 0 & 0 & 0 \\ 0 & S_Y & 0 & 0 \\ 0 & 0 & S_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{11}$$

En el caso de aplicar un escalado uniforme en los ejes cartesianos se debe tener en cuenta la restricción $S_X = S_Y = S_Z$

6.5.3. Rotación

La rotación de un punto del espacio $\mathbf{P} = [X\ Y\ Z]^\intercal$ con respecto a cada uno de los ejes cartesianos, de forma que α es un ángulo de rotación en el eje X, β el ángulo en el eje Y y γ el ángulo del eje Z. Esto expresado en coordenadas homogéneas está dada por las matrices de rotación R_α, R_β y R_γ para los ejes X, Y y Z respectivamente.

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$R_\beta = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\cos\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$R_\gamma = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

6.6. Transformaciones proyectivas 2D

La geometría proyectiva 2D es el estudio de las propiedades del plano proyectivo S^2 que son invariantes bajo un grupo de transformaciones denominandas *proyectividades*.

La proyectividad es una transformación invertible dada por $h : S^2 \to S^2$ de tal forma que la línea recta es transformada como una línea recta, siendo definido cómo:

$$h(\mathbf{m}) = \mathbf{m}' = \mathbf{H}\mathbf{m} \tag{15}$$

donde $\mathbf{H}$ es una matriz 3x3 no singular. Se dice entonces que $\mathbf{H}'$ es la transformación lineal $\mathbf{H}$ de $\mathbf{m}$. Esta transformación 2D tiene una única correspondencia en cada punto de otro plano 2D, de $\mathbf{m}$ a $\mathbf{m}'$.

La ecuación (15) puede escribirse de forma explícita como:

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{16}$$

Esta transformación de coordenadas no son afectados si se cambia $\mathbf{H}$ por $k\mathbf{H}$, para $k \neq 0$, por lo cual $\mathbf{H}$ es una matriz homogénea.

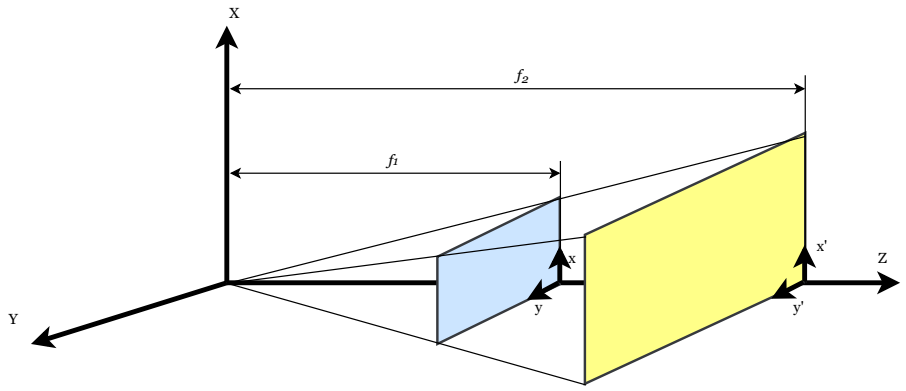


Figura 8: Proyección en dos planos paralelos. [Elaboración propia]

Utilizando el Teorema de Thales, se obtienen la siguientes relaciones:

$$\frac{x'}{f_2} = \frac{x}{f_1}; \frac{y'}{f_2} = \frac{y}{f_1}$$

Estas ecuaciones se pueden escribir usando la forma matricial $\mathbf{m'} = \mathbf{Hm}$, con $\mathbf{m} = [xy1]^\intercal$, $\mathbf{m'} = [x'\ y'\ 1]^\intercal$ y

$$\mathbf{H} = \begin{bmatrix} f_2/f_1 & 0 & 0 \\ 0 & f_2/f_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

Para el caso de dos planos no paralelos como se muestra en la Figura (9) se pueden establecer dos características de la transformación proyectiva [7]:

1. Correspondencia biunívoca entre los puntos pertenecientes a ambos planos

2. Existe al menos una línea recta en un plano que corresponde a una línea recta en el otro plano

De esta manera se puede afirmar que la relación proyectiva entre los planos R_1 y R_2 esta dada por la ecuación general (16).

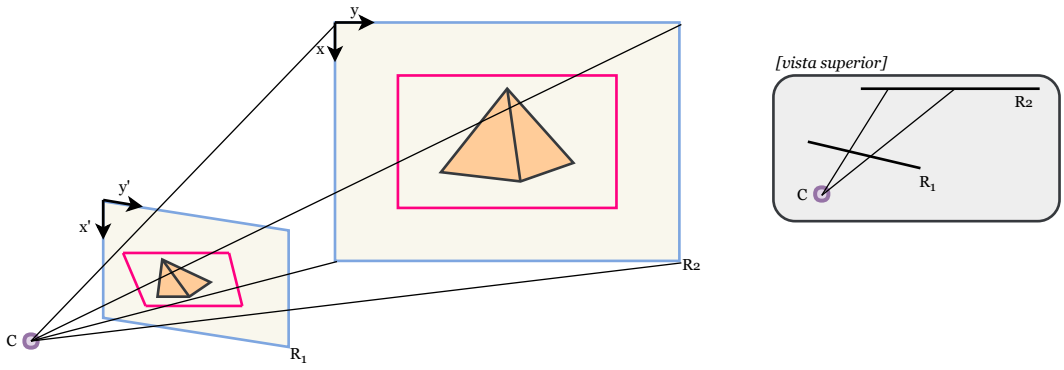


Figura 9: Proyección de dos planos no paralelos. [Elaboración propia]

Por lo general se escogen puntos (x_i, y_i) que pertenezcan a un rectángulo. A partir de la ecuación (16) se obtienen la correspondencia (x', y') de cada punto (x, y) donde:

$$x' = \frac{x'_1}{y'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{x'_2}{y'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Son nueve incognitas de $\mathbf{H}$ que se desean encontrar. De esta manera es posible dividir cada elemento de $\mathbf{H}$ por h_{33} para obtener una matríz $\mathbf{H}$ de solo 8 incógnitas, ya que $h_{33} = 1$. Escribiendo las últimas dos ecuaciónes de manera matricial se obtiene:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{18}$$

$$\mathbf{Ah} = \mathbf{b} \tag{19}$$

Se observa que de cada correspondencia de puntos se obtienen dos ecuaciones. Suponiendo n pares de puntos correspondientes se puede establecer el siguiente sistema de $2n$ ecuaciones y 8 incógnitas.

$$\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ : \\ \mathbf{A}_n \end{bmatrix} \mathbf{h} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ : \\ \mathbf{b}_n \end{bmatrix} \tag{20}$$

Donde $\mathbf{A}_i$ y $\mathbf{b}_i$ son la matriz $\mathbf{A}$ y el vector $\mathbf{b}$ obtenido de la ecuación (20) para el punto i. Para el sistema anterior si $n = 4$ existe una solución directa dada por dada por $\mathbf{h} = \tilde{\mathbf{A}}^{-1}\mathbf{b}$. De ser $n > 4$ el sistema queda sobredeterminado [7].

6.7. Modelos de color

Un modelo de color es un modelo matemático abstracto que describe la forma en que se representan y distribuyen los colores como matrices de números, utilizando normalmente de tres o cuatro valores o componentes de color. [9]

Estos modelos los podemos dividir en dos clases: el primero orientado a equipos (e.g. cámaras, monitores, televisiones) denominados *modelos sensoriales*; el segundo que se asemeja más a la percepción humana, orientados al procesamiento de imágenes y visión, denominados *modelos perceptuales* [10].

6.7.1. Modelos sensoriales

RGB

El modelo RGB se basa en los tres sensores humanos, considerando que todos los colores son una combinación de tres colores básicos: R (rojo), G (verde), B(azul) [10]. Estos colores se representan en coordenadas cartesianas dentro de un cubo unitario representado en la Figura (10).

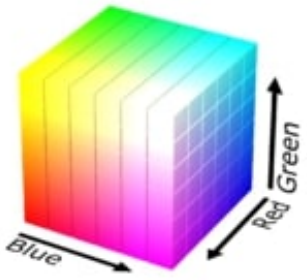


Figura 10: Espacio de color RGB representado en forma de cubo. [7]

6.7.2. Modelos perseptuales

HSV

El modelo "HSV"(*Hue, Saturation, Value*) se obtiene cambiando el espacio geométrico donde existen los colores, convirtiéndolo en una un pirámide hexagonal invertida. Donde V es el eje vertical que representa la brillatez, S es el eje horizontal y H es el ángulo de proyección [10]. El modelo *HSV* se ilustra en la Figura (11)[10].

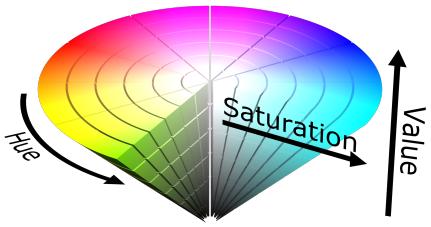


Figura 11: Espacio de color HSV representado en forma de cono, blanco en el centro superior del cono y negro en el vértice inferior. [7]

La conversión de RGB a HSV se logra mediante las siguiente ecuaciones [10]:

$$V = M; [0, 1] \tag{21}$$

$$Si : M = m, S = 0; sino\ S = \frac{M - m}{M}; [0, 1] \tag{22}$$

$$Si : m = B, H = 120\frac{G - m}{R + G - 2m}; [0, 360] \tag{23}$$

$$Si : m = R, H = 120\frac{B - m}{B + G - 2m}; [0, 360] \tag{24}$$

$$Si : m = G, H = 120\frac{R - m}{R + B - 2m}; [0, 360] \tag{25}$$

6.8. Robótica

6.8.1. Robótica móvil

Sistema de coordenadas

El movimiento en robótica consiste principalmente como el cambio local de un objeto rígido en relación con otro objeto rígido. Bajo esta premisa, la traslación se define como el

movimiento de toda la masa de un objeto rígido con la misma dirección y velocidad paralelo al eje, y la rotación como puntos de masa que giran a lo largo de un eje concéntrico. En consecuencia, cada movimiento de un robot puede ser modelado como una combinación de una rotación y de traslación [8].

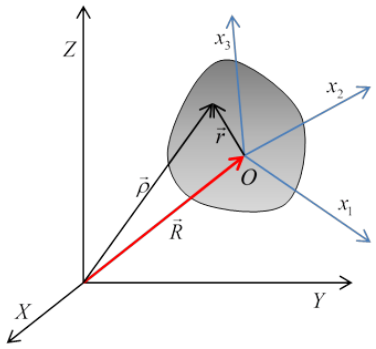


Figura 12: Modelo del cuerpo rígido. [11]

Cinemática

La cinemática es el estudio más básico de cómo se comportan los sistemas mecánicos. Para los robots móviles es necesario entender el comportamiento del sistema mecánico para saber como realizar las tareas designadas mediante el control del hardware mediante software.

Para ello comenzamos definiendo el espacio de trabajo, el cual es crucial para definir los rangos donde el robot se estará moviendo y con ello un marco inercial el cual nos servirá como referencia. Una vez definidos los efectores del robot podemos construir un modelo que permita determinar su pose en el espacio de trabajo en relación al valor de sus efectores. Esta pose es fácil limitar cuando contamos con robots manipuladores los cuales cuentan con una posición fija y solamente es necesario determinar la posición de sus juntas. A diferencia de un robot móvil que puede moverse con respecto a su ambiente.

Para entender el movimiento de un robot móvil es necesario entender el rol que desempeñan las ruedas del mismo, la cual añade restricciones a la cinemática del robot. Para ellos describimos primero la cinemática de una rueda y luego describimos el movimiento del chasis con dos ruedas colocadas. [11]

Modelos cinemáticos y restricciones

Para derivar el modelo que explique el movimiento del robot móvil se establece un proceso ascendente. Cada rueda contribuye al movimiento del robot y, al mismo tiempo impone restricciones sobre el mismo. Las ruedas están unidas según la geometría del chasis del robot y por lo tanto sus restricciones se combinan para formar restricciones en el movimiento general del chasis del robot. Pero para ellos las fuerzas y restricciones deben expresarse

con respecto a un marco de referencia claro y consistente. Un mapeo claro entre marcos de referencia globales y locales es requerido.[11]

El mínimo de ruedas requerido para un robot móvil con ruedas estable son como mínimo tres, debido a consideraciones del centro de masa, torque de los motores y diámetros de rueda seleccionamos dos ruedas de tracción independientes ubicadas en la parte trasera y una rueda omnidireccional sin alimentación en la parte frontal. Basados en el modelo propuesto se desarrolla una notación y terminología necesaria para la siguiente figura

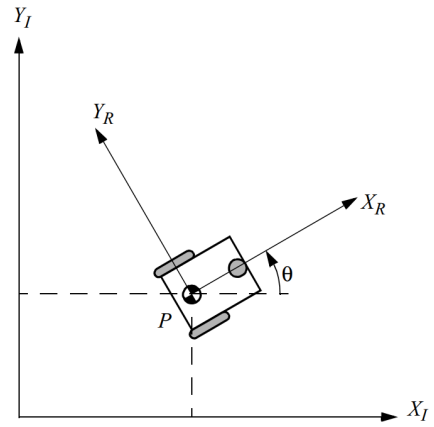


Figura 13: Modelo del robot móvil de tres ruedas. [11]

Representación de la posición del robot

Modelamos el robot como un cuerpo rígido sobre ruedas que opera en un plano horizontal. Las variables para expresar la pose del robot son tres, dos para la posición en el plano y una para la orientación a lo largo del eje vertical, el cual es ortogonal al plano. Por supuesto que existen grados de libertad adicionales y flexibilidad debido a los ejes de las ruedas, las juntas de dirección de las ruedas y las articulaciones de las ruedas. Sin embargo en el chasis de un robot nos referimos únicamente al cuerpo rígido del robot, ignorando las articulaciones y los grados de libertad internos del robot y sus ruedas.

Los ejes X_I y Y_I definen una base inercial arbitraria sobre el plano como marco de referencia global sobre el mismo origen $O\{X_I, Y_I\}$. Escogemos un punto P sobre el chasis del robot como punto de referencia para especificar la posición del robot . Definimos los ejes X_R, Y_R que son los ejes relativos a P sobre el chasis del robot como marco de referencia local del robot. La posición P en el marco de referencia global está dado por las coordenadas x y y de la diferencia angular entre el marco de referencia local y global está dado por θ. La pose del robot se puede escribir como un vector entre estos tres elementos.

Para obtener la rotación del robot en términos de estos componentes es necesario mapear el movimiento a lo largo de los ejes del marco de referencia global al movimiento a lo largo de los ejes del marco de referencia local del robot. Este mapeo se realiza mediante una matriz de rotación ortogonal. [11][12]

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{26}$$

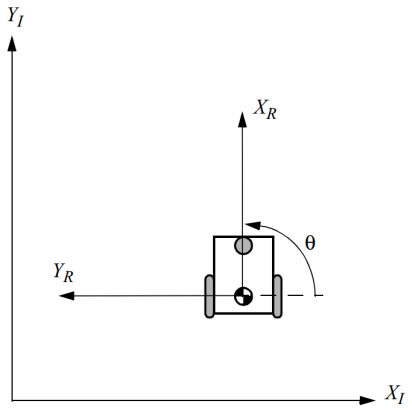


Figura 14: Estableciendo ejes necesarios y variables para el robot móvil de tres ruedas. [11]

Modelado en un marco de referencia absoluto

Para el robot móvil de tipo monociclo, el modelo cinemático (Figura 13) utilizado a partir de ahora es:

$$\dot{x} = u_1 \cos\theta \tag{27}$$

$$\dot{y} = u_1 \sin\theta \tag{28}$$

$$\dot{\theta} = u_2 \tag{29}$$

Donde x y y representa las coordenadas del punto P_m ubicado a media distancia de las ruedas accionadas, y el ángulo caracteriza la orientación del chasis del robot (Figura 14). En esta ecuación, u_1 representa la intensidad de la velocidad longitudinal del vehículo, y u_2 es la velocidad instantánea de rotación del chasis. Las variables u_1 y u_2 están ellos mismos relacionados con la velocidad angular de las ruedas accionadas a través de las relaciones de uno a uno. [13]

$$v = \frac{v_L + v_R}{2}$$
$$w = \frac{v_R - v_L}{2l}$$

Donde r son el radio de las llantas

6.8.2. Robótica de enjambre

La robótica de tipo enjambre se basa en la coordinación y desarrollo de un trabajo conjunto entre robots simples para llegar a un fin concreto cooperativo sin tener que realizar tareas largas. Mejorando así los tiempos de respuesta en la ejecución del proceso establecido.

Algunos ejemplos de publicaciones acerca de comportamientos en robótica de enjambre emergente y autoorganizada son: sistemas dinámicos que surgen de redes biológicas en múltiples niveles de resolución, desde interacciones moléculas y células hasta la ecología conductual de grupos de animales. Bandadas de aves y bancos de peces que viajan en formación y actúan como una sola unidad, lo que permite a estos animales defenderse de los depredadores y proteger sus territorios. Ciertos comportamientos de búsqueda de alimento que incluyen animales que se asignan roles para dividirse en su entorno en zonas no superpuestas. Se ha dedicado extensa investigación esta última década a los problemas de asignación distribuida de tareas.[14]

6.9. Aplicaciones de visión por computadora

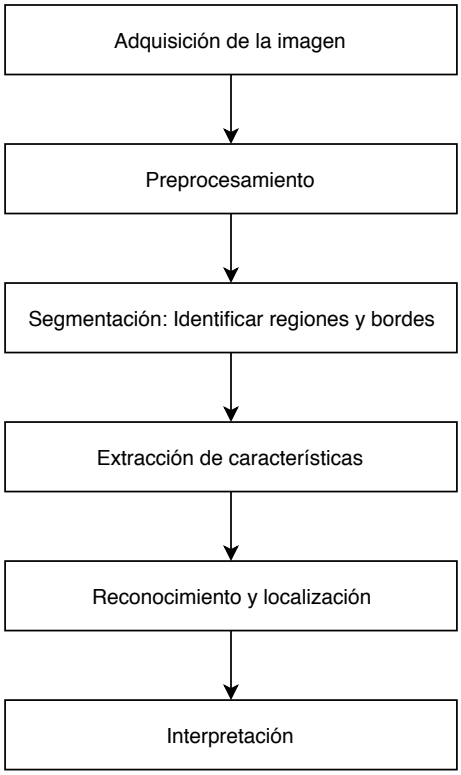


Figura 15: Etapas para algoritmo de visión de computadora. [Elaboración propia]

La visión por computadora es el proceso mediante el cual una máquina, generalmente una computadora digital, procesa automáticamente una imagen e informa "lo que está en

la imagen". Es decir, reconoce el contenido de la imagen. Por ejemplo, el contenido puede ser una pieza mecanizada, y el objetivo puede ser no solo ubicar la pieza sino también inspeccionarla.

Los estudiantes tienden a confundirse con otros términos que a menudo aparecen en la literatura, como procesamiento de imágenes, visión artificial, comprensión de imágenes y reconocimiento de patrones [15]. Cada uno de ellos son procesos diferentes, para ello se elaboró un diagrama de las etapas que caracterizan a un algoritmo de visión por computador (Figura 15).

Se describe de forma breve que es cada una de estas etapas:

1. Adquisición de la imagen: Se captura una proyección de dos dimensiones de la luz que se refleja en los objetos del espacio de trabajo
2. Preprocesamiento: Se eliminó todo aquello que sea ruido, en este caso se recorta y rectifica la imagen
3. Segmentación: Se aplican diferentes filtros y funciones para aislar los elementos deseados
4. Extracción de características: Se obtiene una representación formal de los elementos que deseamos de la escena
5. Reconocimiento y localización: Localizamos cada objeto deseado en la escena
6. Interpretación: Con la información obtenida tenemos conocimiento acerca de lo que se interpreta de la escena

Otra distinción clave es la de procesamiento de imágenes de bajo nivel y de alto nivel. Si interpretamos estos procesos desde la perspectiva de la señal y los sistemas, es más claro describir su diferencia y similitud con el formato de entrada / salida del sistema. Cuando el sistema de procesamiento de imágenes de bajo nivel procesa una imagen de entrada, la salida sigue siendo una imagen, pero una imagen algo diferente. Por ejemplo, puede ser una imagen sin ruido, una imagen que no ocupa tanto espacio de almacenamiento como la imagen de entrada, una imagen más nítida que la imagen de entrada, etc. Aunque el procesamiento de imágenes de bajo nivel no proporciona información sobre el objeto que se está visualizando, es probable que sea necesario, ya que permite que los algoritmos de nivel superior funcionen correctamente.

Los términos visión artificial, comprensión de imágenes y procesamiento de imágenes de alto nivel a menudo se utilizan para designar visión por computadora. Estos son procesos que operan en una imagen de entrada, pero la salida ya no es una imagen; en cambio, es una interpretación de la imagen. [7]

6.10. OpenCV

OpenCV es una biblioteca de visión por computadora de código abierto disponible online. La biblioteca está escrita en C y C ++ y se ejecuta en Linux, Windows y Mac OS X. Este es un desarrollo activo en las interfaces para Python, Ruby, Matlab y otros lenguajes. OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. OpenCV está escrito en C optimizado y puede aprovechar los procesadores multinúcleo. Si desea una optimización automática adicional en las arquitecturas Intel, se pueden comprar las bibliotecas de Inteligencia de rendimiento integrada de Intel (IPP), que consisten en rutinas optimizadas de bajo nivel en muchas áreas algorítmicas diferentes.

Automáticamente se ejecuta la biblioteca IPP apropiada en tiempo de ejecución si esa biblioteca está instalada. Uno de los objetivos de OpenCV es proporcionar una infraestructura de visión por computadora fácil de usar que ayude a las personas a desarrollar aplicaciones de visión bastante sofisticadas rápidamente. La biblioteca OpenCV contiene más de 500 funciones que cubren muchas áreas en visión, incluida: la inspección de productos de fábrica, imágenes médicas, seguridad, interfaz de usuario, calibración de cámaras, visión estéreo y robótica.

Debido a que la visión por computadora y el aprendizaje automático (*Machine Learning*) van de la mano, OpenCV también contiene una Biblioteca de Aprendizaje de Máquina (MLL) completa y de propósito general. Esta sub-biblioteca se centra en el reconocimiento estadístico de patrones y la agrupación. MLL es muy útil para las tareas de visión que son el núcleo de la misión de OpenCV, pero es lo suficientemente general como para ser utilizado en cualquier problema de aprendizaje automático. [15]

6.10.1. Thresholding

Esta función consiste en aplicar un umbral de nivel fijo a una imagen. Aunque es posible utilizar esta función con una imagen multicanal, generalmente se utiliza en una imagen de un solo canal (o escala de grises) para crear una imagen binaria con los píxeles aceptados que superan el nivel umbral y los que no. El marco OpenCV proporciona una serie de funciones para tratar la segmentación de imágenes en general.

La función de *threshold* de OpenCV simplemente nos devuelve una imagen binaria de aquellos píxeles con un valor mayor al parámetro del umbral.

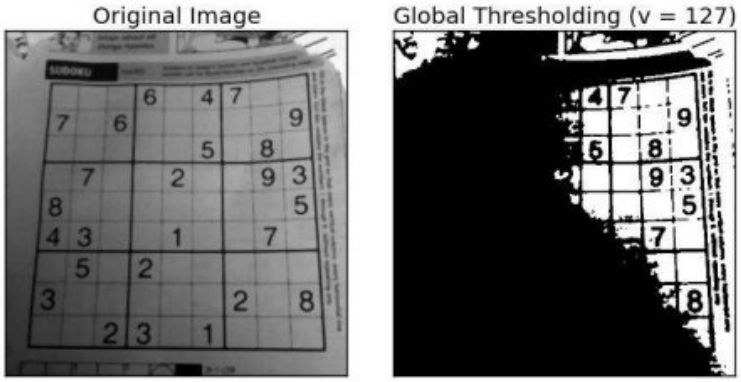


Figura 16: Aplicación de threshold en una imagen. [7]

La otra función *adaptiveThreshold* se puede usar para aplicar un umbral adaptativo a una imagen en escala de grises. Esta función, según el método adaptativo que se le haya pasado, se puede usar para calcular el valor umbral de cada píxel de forma individual y automática.

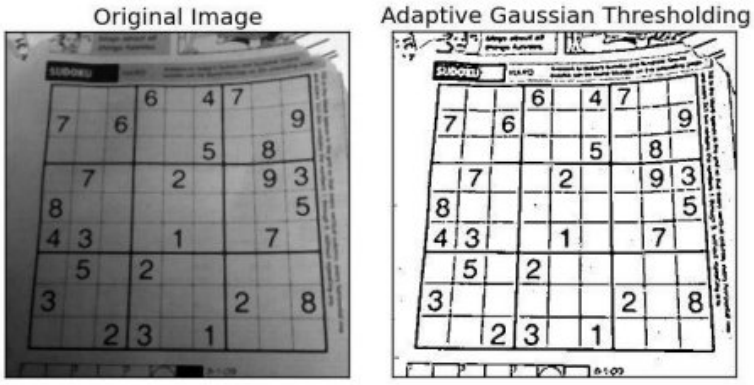


Figura 17: Aplicación de threshold adaptativo en una imagen. [7]

6.10.2. Detección de bordes mediante Canny

El detector de bordes Canny trata de aliviar todos los diversos problemas de los detectores basados en gradiente y curvatura. Canny primero formalizó un conjunto de propiedades que un detector de bordes óptimo debería tener y luego trabajó hacia un método que satisficiera estas propiedades. Según Canny, un detector de borde óptimo debe tener las propiedades de buena detección, buena localización y respuesta mínima. Una buena detección dice que el filtro debe responder sólo a los bordes y no al ruido. Por lo tanto, los bordes deben encontrarse con el mínimo de *spurious edges*. Una buena localización significa que el borde detectado está cerca del borde verdadero. Finalmente, la respuesta mínima dice que la ubicación exacta del borde está marcada con una respuesta de punto único. Hay una compensación que se logrará entre estos diferentes objetivos. En cualquier imagen real, el ruido desempeñará un papel. El filtrado suavizado o de paso bajo mejorará la detección a costa de la localización y la respuesta mínima [15].

Sobre la base de los objetivos anteriores, Canny propuso un método de cuatro pasos: (a) Suprimir el ruido utilizando el filtrado de paso bajo; (b) Calcular imágenes de gradiente de magnitud y dirección; (c) Aplicar supresión no máxima a la imagen de magnitud de gradiente; (d) Usa la histéresis y análisis de conectividad para detectar bordes. Esto significa que hay dos umbrales, uno superior y uno inferior. Si un píxel tiene un gradiente mayor que el umbral superior, entonces se acepta como un píxel de borde; si un píxel está por debajo del umbral inferior, es rechazado. Si el gradiente del píxel está entre los umbrales, solo se aceptará si está conectado a un píxel que está por encima del umbral alto [15].

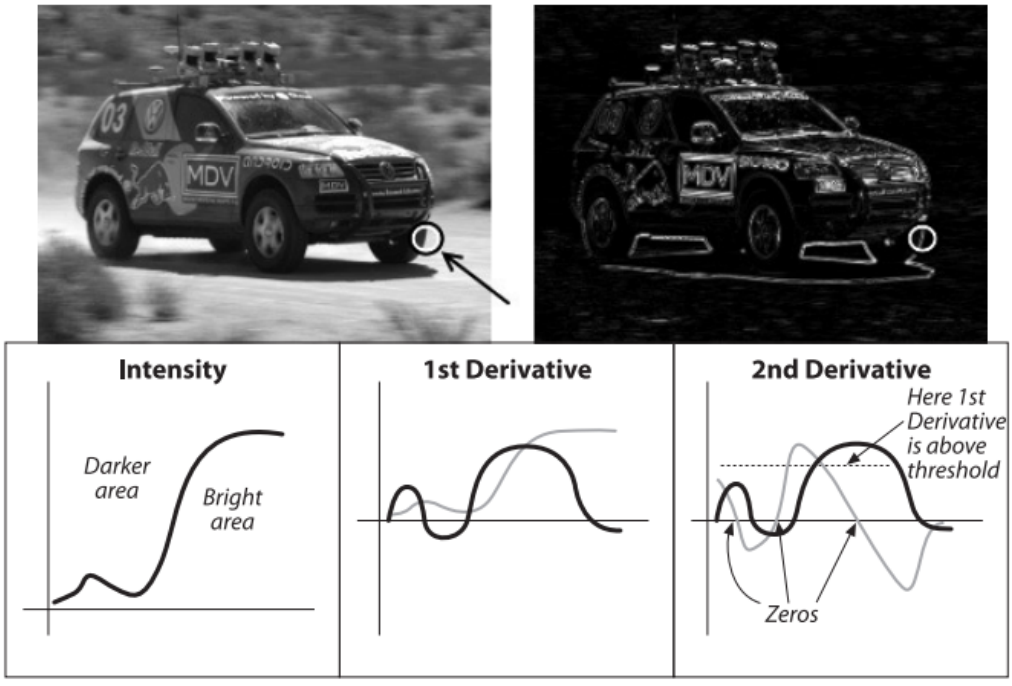


Figura 18: Aplicación de detector de bordes Canny una imagen. [7]

6.10.3. Especificaciones de la cámara

Se escogió la cámara Logitech c920 con las siguientes especificaciones:

Especificaciones de Webcam	
Tipo de conexión	USB
Protocolo USB	USB 2.0
Tipo de enfóque	Automático
Resolución óptica	Verdadera: 3MP
Captura de vídeo (16:9 W)	320p, 480p, 720p, 1080p
Máximo Frame Rate	1080p@30fps
Lentes y tipo de sensor	Vídrio

Cuadro 1: Especificaciones técnicas de HD Pro Webcam C920

CAPÍTULO 7

Metodología

7.1. Requerimientos del sistema

Para la creación de una cama de pruebas de robots móviles se analizó cual era el modelo que describía la posición en el espacio de un robot móvil, se obtuvo que la pose del robot puede representarse con las variables x, y y θ. Bajo esta premisa es necesaria solamente una cámara para tomar los datos necesarios para describir la posición del objeto en el tablero mediante una sola imagen. Suponemos que nuestra cámara es un modelo *pinhole*. Más adelante veremos que el modelo *pinhole* no puede describirnos las distorsiones radiales y tangenciales.

Cabe mencionar que con base a los antecedentes se tomaron ideas con el objetivo de observar las ventajas y desventajas de cada diseño. Construyendo un diseño propio que sea simple, replicable y de bajo presupuesto. A continuación se enumeran las desventajas en base a los antecedentes de otras camas de pruebas implementadas:

- Los *AruCo makers* utilizados en el *Robotarium* contienen un numero de hasta 1024 identificadores en su librería, a pesar de la flexibilidad en la cantidad de códigos, cabe mencionar que la implementación de la detección de estos algoritmos resulta complejo, por lo que se busca una alternativa que sea más sencilla de interpretar y detectar. Específicamente para el propósito de la mesa de pruebas. Ya que los *AruCo markers* al ser identificadores para aplicaciones de realidad aumentada obtienen una dimensión más en el eje z que no nos interesa, ya que para obtener la información necesaria de los robots móviles sobre la mesa son necesarios únicamente los ejes x y y de nuestro plano de trabajo.
- Los Zooids cuentan con un código de alto rendimiento para detectar las posiciones de los robots (muestreo de 3000Hz), esta puede ser adaptable a una cualquier su-

perficie siempre y cuando exista la cantidad de luz necesaria para reflejarse en los foto-diodos del proyector. Este proyector resulta ser la desventaja, ya que el proyector DLP LightCrafter de Texas Instruments Inc. tiene un precio aproximado de 600USD, lo que resulta en un plataforma de alto presupuesto en comparación con una cámara Logitech c920 valorada en un precio aproximado de 50USD.

- Se busca un diseño barato como los Kilobots con la diferencia que la plataforma permita cambiar el diseño del robot móvil manteniendo su rastreo. Es importante poder variar los efectores de movimiento de estos robots móviles, ya que al ser motores vibradores los del Kilobot tienen un rango de desplazamiento muy bajo ($1cm/s$). Lo cual aumenta los tiempos de ejecución de un algoritmo.

La fuente principal del desarrollo de reconocimiento se basa en gran parte en el Robotarium con ciertas modificaciones. Comenzando por el uso de diferentes identificadores que los *arUco markers* y la posibilidad de auto-calibración de la cámara mediante un algoritmo que obtenga las esquinas del tablero.

Para llegar a la implementación final fue necesario realizar diferentes pruebas para obtener el performace, precisión y exactitud y así comparar cual era el algoritmo adecuado para la cama de pruebas, a continuación se discuten en los siguientes capítulos las diferentes implementaciones y métodos encontrados.

7.2. Seleccionando el lenguaje de programación

Para la selección del lenguaje de programación se investigaron las ventajas y desventajas de los lenguajes de programación Python, Matlab y C++. Se elaboró una tabla que resumiera sus características:

MATLAB	**Ventajas:** -Rápido para implementación y debugging. -Familiaridad con el lenguaje y funciones. **Desventajas:** -No posee plataformas para creación de IDEs. -Es de paga.
C++	**Ventajas:** -Contiene librería OpenCV. -Lenguaje de alto rendimiento y eficiencia. -Soporte amplio en diferentes plataformas. -Cuenta con ambientes de desarrollo de IDEs amigables. -Es portable. **Desventajas:** -Lenguaje complejo.
Python	**Ventajas:** -Contiene librería OpenCV. -Documentación y aplicaciones variadas. -Lenguaje amigable para desarrollar aplicaciones complejas. **Desventajas:** -Tiempos de ejecución altos comparados con C++.

Cuadro 2: Ventajas y desventajas de diferentes lenguajes de programación

Se seleccionó el lenguaje de C++ por motivos de performance. Ya que se esta buscando obtener la mayor frecuencia de muestreo posible. Python es un lenguaje interpretado mientras que C++ es compilado a código máquina por lo que existe una diferencia en cuanto a la velocidad de ejecución de cada uno. A pesar de ello cabe mencionar el código en OpenCV de las librerías de Python ya se encuentra convertido a código de maquina.

CAPÍTULO 8

Implementando los identificadores

8.1. Diseño el identificador

Para la estandarización del reconocimiento se realizaron múltiples diseños que cumplieran la funcionalidad de presentar un identificador y también fuera de fácil de lectura para el algoritmo. Era necesario crear un diseño fácil de aplicarle visión de computadora y que fuera capaz de dar la posición y ángulo de rotación del robot. Se pensaron el diseños circulares, rectangulares y cuadrados.

El primer diseño que se creó teniendo como referencia los códigos QR, pero al ya imprimir la figura en papel se obtuvo poca resolución para identificar el patrón en binario que establecían la numero el identificador, siguiendo la regla de "mínimo de 3 píxeles para obtener la información". La aplicación resulta ser diferente ya que mientras la cámara se encuentra a más de un metro de distancia en la mesa de pruebas, cuando se captura un código QR es necesario acercar el celular para tomar una foto del elemento con alta resolución y así poder determinar el color de los patrones. Por ello el modelo fue abandonado.

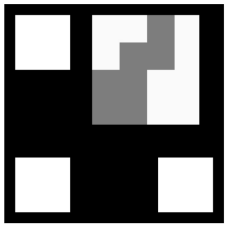


Figura 19: Primer prototipo de identificador.

Se estableció otro modelo y se terminó de escoger como el estándar de reconocimiento debido a su simplicidad y mayor rango de píxeles para el análisis posterior de la imagen, a continuación se explica su estructura.

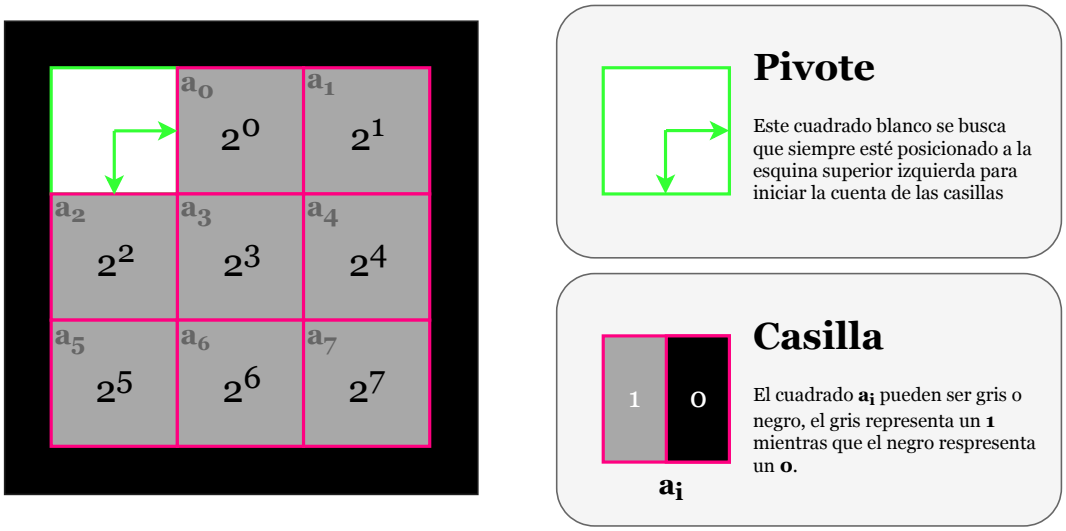


Figura 20: Estándar para reconocimiento de ID y pose del robot.

Para calcular el identificador del robot (ID) de la Figura (21) se calcula de la siguiente manera:

$$ID = (a_0)2^0 + (a_1)2^1 + (a_2)2^2 + (a_3)2^3 + (a_4)2^4 + (a_5)2^5 + (a_6)2^6 + +(a_7)2^7 \qquad (30)$$

Donde a_i puede tomar el valor de uno o de cero, dependiendo si es gris o negro, respectivamente. A continuación se muestra una identificador con su ID calculado.

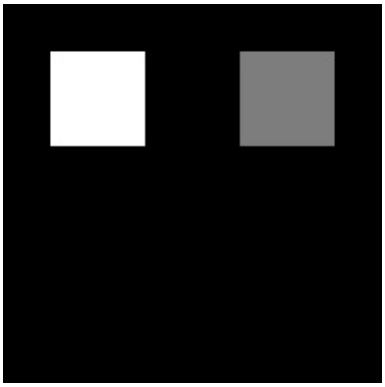


Figura 21: Lectura del patrón binario, con $ID = 2^1 = 2$.

CAPÍTULO 9

Métodos para detección y extracción de información del identificador

Los siguientes métodos tienen como finalidad extraer la información de los identificadores, las cuales son su posición x y y, orientación θ y número del identificador que los diferencia de los demás. Para ello es necesario aplicar los filtros que permitan diferenciar la superficie de los identificadores en la mesa y posteriormente otros filtros para la lectura del patrón binario.

A continuación se discuten los diferentes métodos que se usaron y se selecciona el más conveniente.

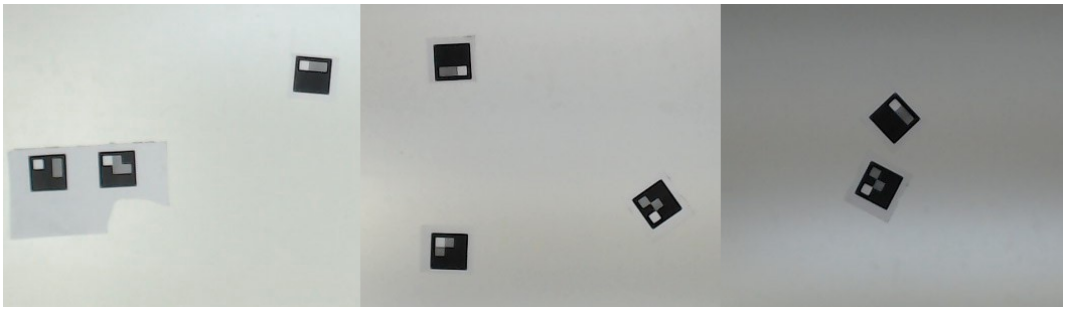


Figura 22: Imagen del tablero en distintas horas del día.

9.0.1. Mediante thresholding

En esta prueba se utilizó la función *thresholding* de OpenCV, a pesar de tener iluminada la plataforma en donde se capturaba la imagen en el transcurso del día el treshold comenzaba a aplicar el umbral el localidades donde no se encontraban identificadores de color oscuro, esto era más notable conforme se oscurecía el día. A continuación se muestra el thresholding

aplicado a distintas horas del día, manteniéndose los valores de umbral constantes.

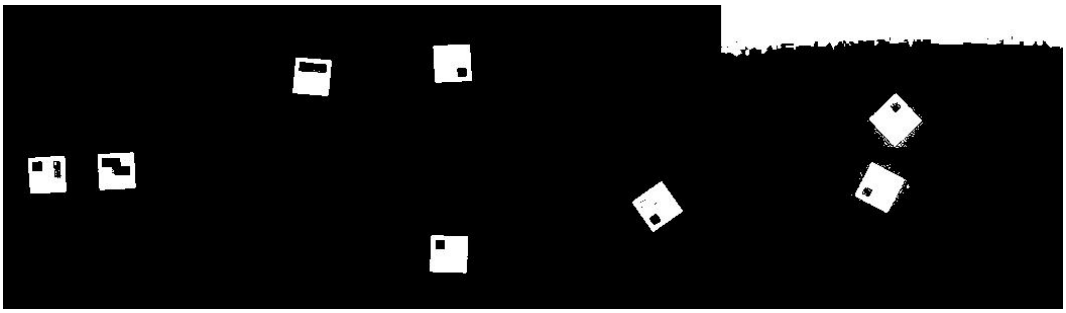


Figura 23: Aplicación de threshold en la imagen del tablero en distintas horas del día.

Si el resultado del threshold permitía aislar únicamente los cuadrados en el siguiente paso del algoritmo se detectaban los bordes de los identificadores mediante la función *findContours* de OpenCV. Si el resultado era como el de la derecha de la Figura 23 los detectaria de forma incorrecta. Inicialmente se realizó gran parte del código con este método para luego pasar a uno con menor posibilidades de errores.

9.0.2. Mediante Thresholding Adaptativo

Con los errores que se encontraron usando el threshold se buscó otro método que permitiera identificar los contornos de los identificadores y fuera menor la incidencia en el resultado la cantidad de luz que existiere en el ambiente. Bajo esa premisa se encontró la función *adaptativeThreshold*, en donde se observaron mejores resultados, a pesar de ya no existir una dependencia de la cantidad de luz significativa en el resultado existían detección de contornos no deseados. A continuación se muestran los resultados obtenidos mediante este método:

Figura 24: Aplicación de threshold adaptativo en la imagen del tablero en distintas horas del día.

9.0.3. Mediante detección de bordes por algoritmo de Canny

Luego de investigar extensamente detectores de bordes se cambió la metodología y se decidió usar un detector de bordes para aislar directamente los identificadores, esto mediante

el detector de bordes de Canny. A pesar de ser más demandante este algoritmo contaba con un mejor resultado para la detección de los contornos sin importar la cantidad de luz en la imagen. Debido a que cuenta con un umbral inferior y superior se pueden obviar bordes con un diferencial establecido en la imagen, detectando solamente los cambios bruscos de color como lo son el identificador en el tablero.

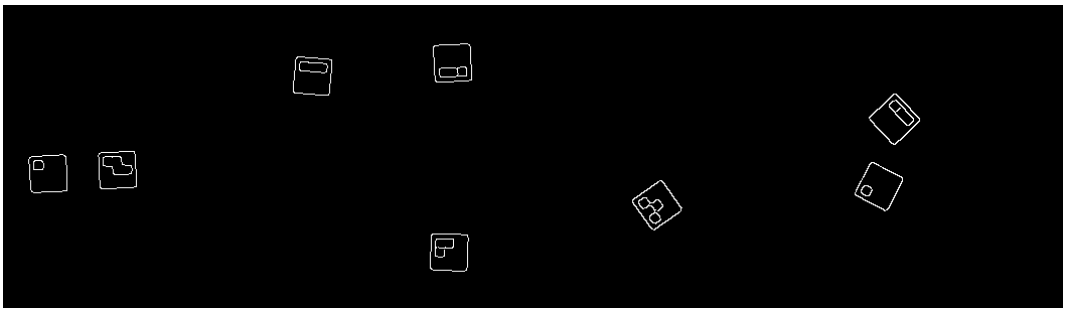


Figura 25: Aplicación de detección de bordes Canny en la imagen del tablero en distintas horas del día.

A pesar de ya tener el contorno que encierra el identificador se percató que antes de realizar el método de procesamiento en la imagen, es necesario eliminar la perspectiva, de lo contrario los identificadores encontrados tendrían forma de polígonos de cuatro lados donde todos sus lados serian diferentes, cuando el resultado deseado es un cuadrado.

9.1. Corrección de perspectiva

Hasta este momento las imágenes que se habían analizado eran tomadas por una cámara colocada en un trípode a noventa grados de una cartulina blanca, teniendo ya las dimensiones de la mesa se pasó a realizar el tamaño de una cartulina que tuviera las dimensiones de la mesa (130x90cm).

El siguiente desafío en el análisis de los identificadores en la mesa era rectificar la imagen, ya que existía la posibilidad que la persona que montara la cámara no la colocase ortogonal a la mesa, así que mediante una corrección de perspectiva, es decir una proyección de planos no paralelos podemos realizar una transformación a nuestra imagen para eliminar la perspectiva bajo la premisa de que la mesa es un rectángulo rígido por lo que no cambian sus esquinas.

El algoritmo además de quitar la perspectiva también recorta la imagen a las 4 esquinas establecidas, esto nos permite analizar únicamente el espacio de trabajo donde están los micro-robots.

Para la aplicación de este algoritmo se usó la función *getPerspectiveTransform* de OpenCV, que devuelve la matriz de transformación necesaria para rectificar la imagen. Para usarla es necesario establecer cuatro pares de puntos, un par de estos son las cuatro esquinas del tablero por el momento estas fueron seleccionadas de forma manual y el otro par es un rectángulo con vértices en $\{0, 0\}$, $\{X_{max}, 0\}$, $\{0, Y_{max}\}$ y $\{X_{max}, Y_{max}\}$. Donde X_{max} es la distancia máxima en el eje x entre dos puntos del primer par de cuatro puntos y Y_{max} para

el eje y.

Una vez calculada la matriz de transformación de perspectiva se usa la función *warpPerspective* de OpenCV, donde se ingresa la imagen original y la transformación que quiere aplicarse y se guarda el resultado. La siguiente figura ilustra la transformación aplicada una vez determinados las cuatro esquinas.

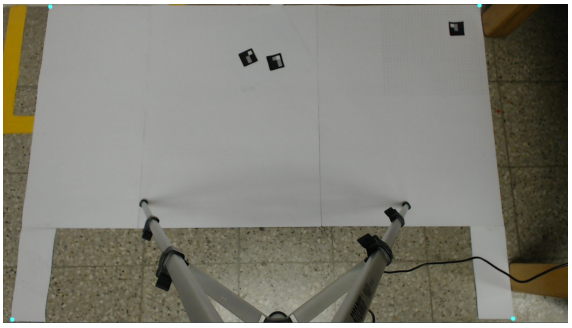


Figura 26: Imagen de trípode en perspectiva antes de la transformación proyectiva no paralela.

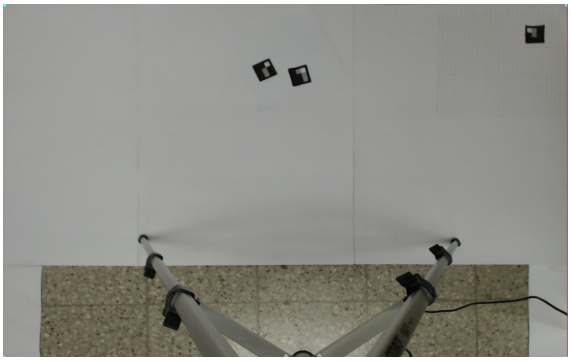


Figura 27: Imagen de trípode en perspectiva después de la transformación proyectiva no paralela.

Este proceso solo es necesario aplicarlo al comienzo del algoritmo, ya que en teoría la cámara se mantendrá fija en la plataforma una vez sea colocada.

Los efectos de distorsión de lente no fueron tomados en cuenta ya que después de encontrar la matriz de distorsión y aplicarla en una imagen del tablero se obtuvieron resultados parecidos. Se investigó que las cámaras web antiguas poseen altos coeficientes de distorsión pero los modelos modernos han disminuido significativamente, por ello ya no es aplicamos esta corrección a nuestra imagen.

9.2. Obteniendo la información del identificador

Una vez rectificada la imagen se pueden comenzar a usar los filtros necesarios para la extracción de los diferentes identificadores que se encuentran en la mesa y para cada uno de estos identificadores interpretar el patrón binario para extraer su posición x, y, ángulo de rotación y numero de ID. Para ello la imagen se le aplica el detector de bordes Canny y la ingresamos a la función *findContours* de OpenCV, esta nos retorna una lista de contornos encontrados, cada contorno posee el arreglo de puntos que lo hacen un contorno. Mediante una asignación condicional podemos filtrar aquellos contornos que tengan un perímetro que se encuentre en el rango del identificador, se imprime el resultado (Figura 28).

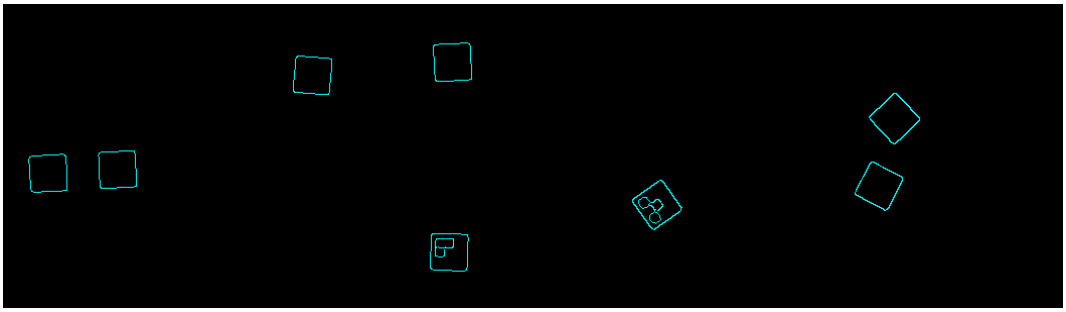


Figura 28: Aplicación de *findCountourns* en la imagen rectificada del tablero con detección de bordes Canny aplicado.

Para obtener el centro del identificador se calculó el centro de masa recorriendo la colección de puntos que contenía el contorno en la lista, pero se abandonó la metodología para encontrar una función en OpenCV que obtuviera las dimensiones de un cuadrado a partir de una colección de puntos. En la búsqueda de esta función se encontró *minAreaRect*, el cual estima el rectángulo mínimo que puede existir en la región de puntos ingresada, devolviendo así lo siguiente: posición del centro de la figura en píxeles, el ancho y largo en píxeles, y el ángulo de rotación con respecto a las esquinas inferiores izquierdas (Figura 29).

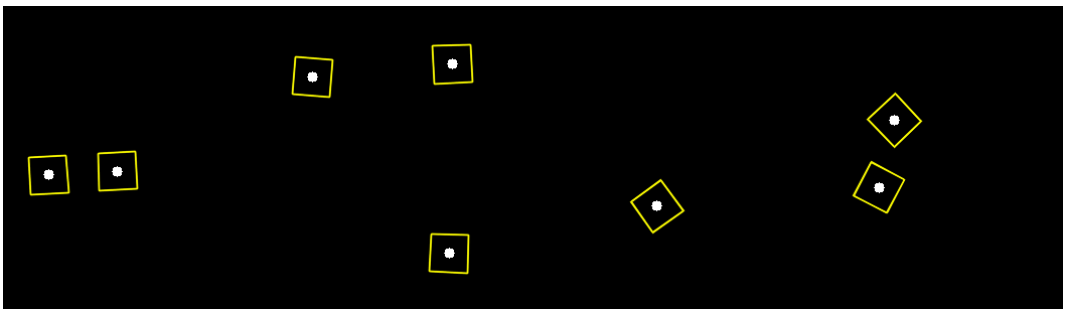


Figura 29: Aplicación de *minAreaRect* despues de aplicar *findCountourns*.

Ya se tiene la posición del robot pero para obtener el numero de ID y el ángulo de rotación es necesario leer la combinación de colores que tiene el identificador. Para ello será necesario seguir los siguientes pasos:

1. Transformar la escala de colores a blanco y negro, ya que facilitará las operaciones y no nos vemos afectados en la lectura del patrón de colores ya que el identificador se encuentra en diferentes escalas de grises (Figura 20).

2. Recortar N veces la imagen en escala de grises, donde N es el número de robots en la mesa. Esta imagen recortada contendrá únicamente un identificador, para recortarse será necesario que las dimensiones de ancho y largo de aproximadamente $\sqrt{2}L$ donde L es el ancho o largo en píxeles (Figura 30).

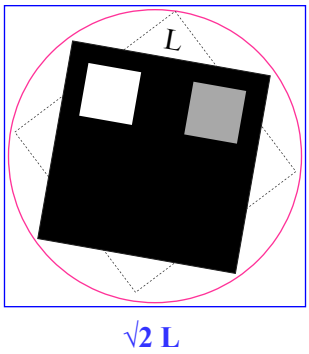


Figura 30: Relación de tamaño geométrico de la imagen recortada en relación a un identificador de lado L.

La relación de $\sqrt{2}L$ se obtuvo por análisis geométrico bajo la premisa que L es un lado del cuadrado y que éste tiene circunscrito un circulo de radio indefinido, al despejar el radio y multiplicarlo por dos obtenemos la relación. Esto lo hacemos ya que no sabemos cómo va rotado el identificador con respecto a la imagen, por ello cortamos a partir del centro $\pm\frac{\sqrt{2}L}{2}$ píxeles.

3. Rotamos la imagen con el ángulo que nos da *minAreaRect*. Con este procedimiento lo que queremos es colocar el identificador sin rotación a pesar que el pivote no se encuentre en la esquina superior izquierda.

4. Rotar la imagen sino se encuentra en la posición *Default*. Para ello leemos el color de las esquinas buscando donde se encuentra el valor más alto (blanco) y a partir de este se le dará una rotación más a la imagen recortada para iniciar la lectura del ID. Además nos ayuda a obtener la rotación del robot con respecto la mesa. Es decir que el ángulo de rotación que nos da *minAreaRect* le sumamos $\frac{\pi}{2}$, π o $\frac{3\pi}{2}$ de ser necesario.

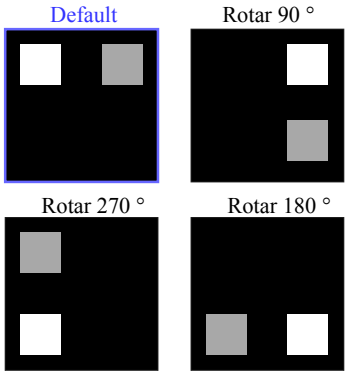


Figura 31: Distintas posibilidades de identificador con su respectiva rotación para llevarlo a la posición *Default*.

5. Leer el patrón de colores del identificador para obtener su ID (Figura 20). Esto se logra leyendo el valor del píxel que se encuentra en el centro de cada cuadro.

6. Convertir la posición en píxeles a unidades en centímetros. Esto se realiza mediante una sencilla regla de 3. Sea $\{x_{pix}, y_{pix}\}$ la posición del identificador en píxeles, a este punto solo será necesario multiplicarlo por 130cm / *ImageWidth* píxeles, para así obtener en cm el valor. El *ImageWidth* es el ancho de la imagen con la corrección de perspectiva ya realizada.

7. Guardar los valores del identificador. Más adelante se estará implementando una clase de tipo *robot* que contenga todos los atributos necesarios para el control del micro-robot.

CAPÍTULO 10

Implementación

10.1. Montando la mesa de pruebas

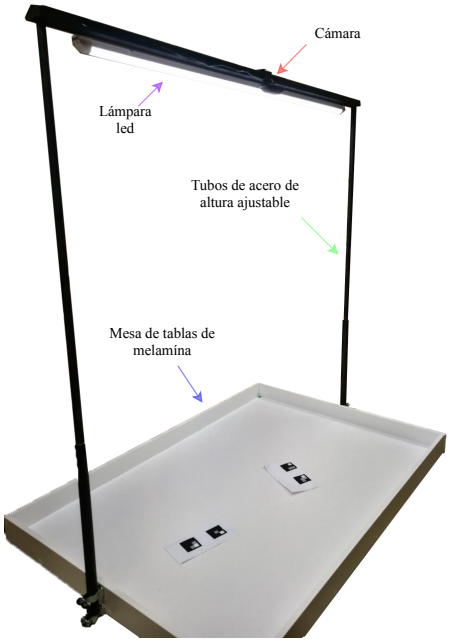


Figura 32: Mesa de pruebas.

Para las dimensiones de la mesa se decidió trabajar con las mismas que trabaja el Robotarium que son de 130x90cm [1]. Para ello se compró una tabla de melamina blanca y se

dieron las especificaciones para corte a modo de formar una especie de cajón (Figura 32). Se compraron y diseñaron dos tubos de acero con altura ajustable mediante un perno en cada uno; los cuales pueden alcanzar una altura de hasta 1.5m, lo suficiente para capturar toda la escena a una resolución de 920x720 píxeles.

Ya que se escogieron las funciones necesarias para el algoritmo de visión por computadora a continuación se muestra la estructura principal que seguirán los programas para obtener las posiciones de los identificadores:

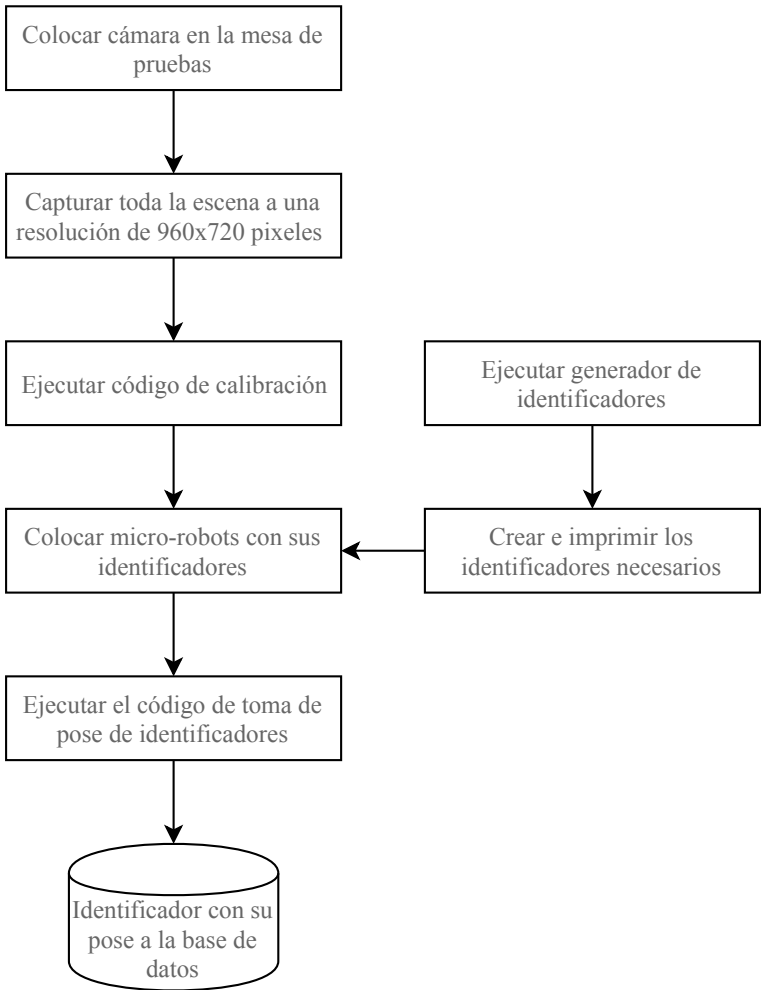


Figura 33: Diagrama de los distintos procesos para la obtención de poses de los identificadores.

Una vez establecida la metodología se comenzaron a crear los diferentes programas con sus métodos principales.

10.2. Creación de la clase principal

Para la manipulación de los micro-robots es necesario establecer una clase que contenga los datos para operar y categorizar los robots. Para ello se utiliza un *header* de C++ que contiene la clase publica *robot*. Como se estarán tomando datos de diferentes identificadores se establece otra clase que contenga un vector de *robot* así como métodos de búsqueda y actualización de información de los identificadores encontrados, a esta clase le estaremos llamando *vectorrobots*. A continuación se realiza un diagrama UML que explica los atributos y métodos de estas clases.

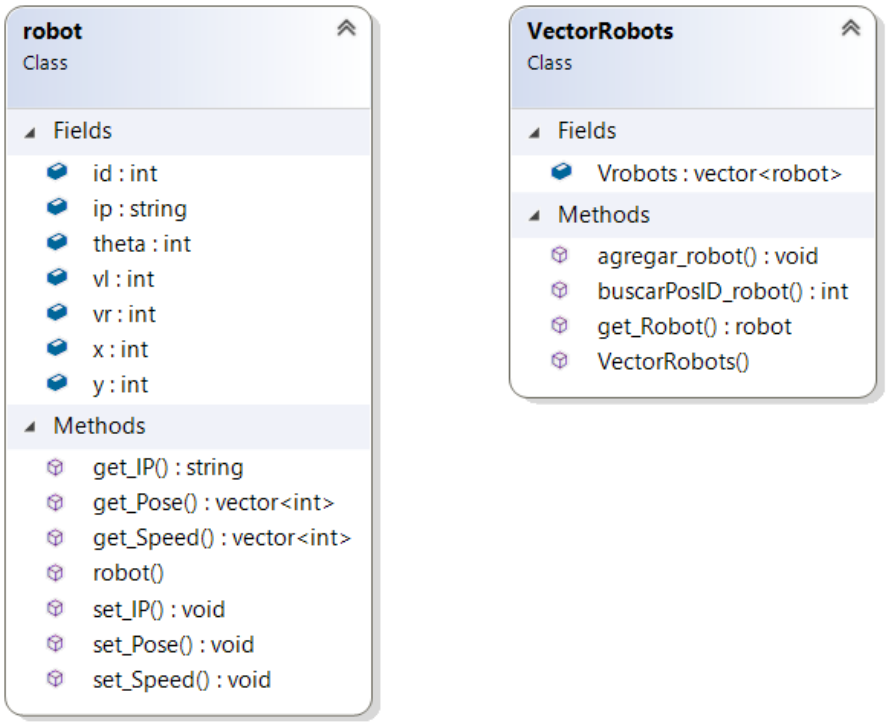


Figura 34: Diagrama UML de la clase *robot* y *vectorobots*.

10.3. Creación de programas y sus respectivas funciones

Se establecen tres programas diferentes que cumplan con el objetivo general de crear un algoritmo de visión de computador para la experimentación de micro-robots móviles:

- Programa para generación de códigos: En este código colocamos el valor de 1 a 255 para que cree un identificador que el algoritmo sea capaz de detectar, generando un archivo .jpg para ser copiado en un editor de texto y ser impreso a la medida deseada, siempre y cuando sea mayor igual a 3cm.

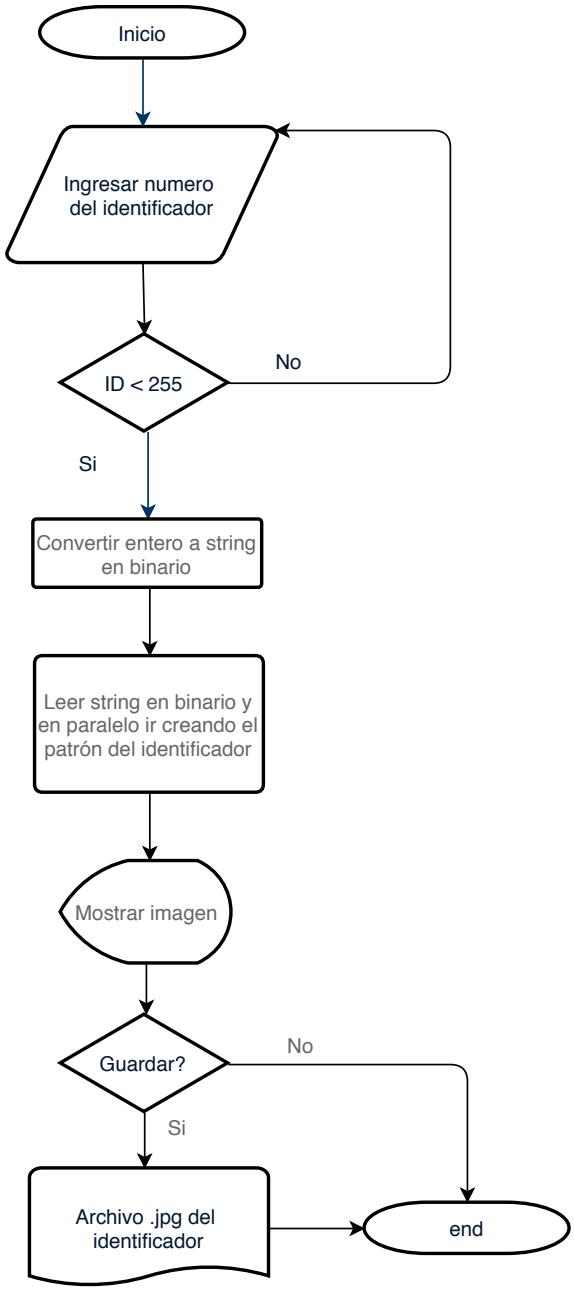


Figura 35: Diagrama de flujo de programa para generación de códigos.

- Programa para calibración de cámara: Este toma una foto de la mesa una vez es colocada la cámara y su objetivo es detectar las esquinas de la mesa para luego guardar en un archivo de texto la matriz de transformación de *lambda* necesaria para remover la perspectiva y recortar la imagen a solo el plano de trabajo de interés.

  Para la implementación de este programa fue necesario colocarle al tablero unos círculos verdes de 1.6cm de diámetro, con el objetivo que estos se conviertan en las esquinas del tablero reconocibles por el algoritmo (Figura 36).

Figura 36: Imagen del tablero con esquinas colocadas.

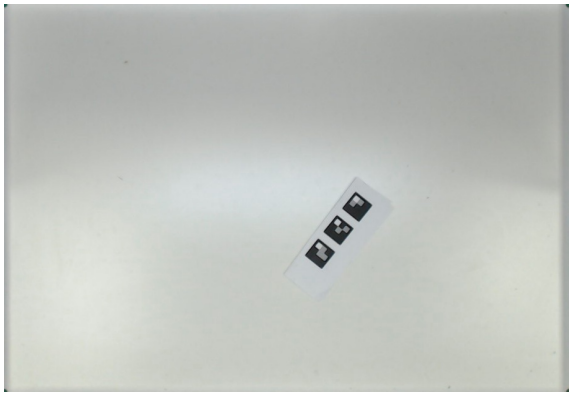


Figura 37: Imagen del tablero con esquinas colocadas luego de aplicarle el algoritmo de calibración.

El siguiente diagrama de flujo ilustra como se obtiene las esquinas del tablero y como es guardada la información en un archivo de texto.

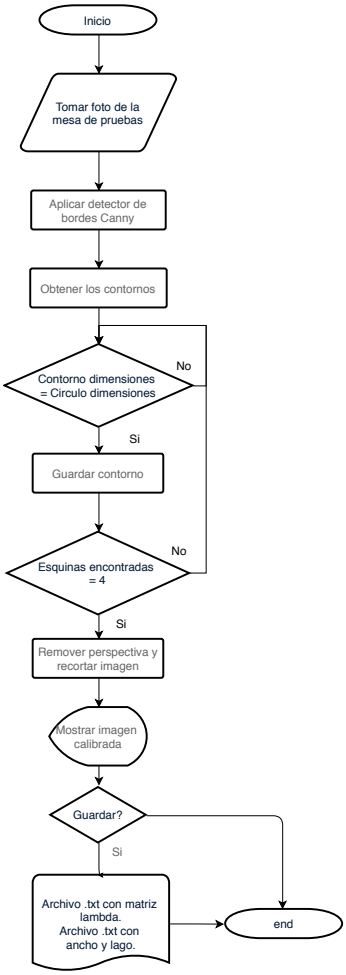


Figura 38: Diagrama de flujo de programa para calibración de tablero.

- Programa para obtención de pose: Es el principal, éste detecta los identificadores de la imagen y los guarda en un objeto tipo *vectorrobot*, estos datos son también almacenados en una base de datos en PostgreSQL para que la información pueda ser accesada desde cualquier máquina que se encuentre conectada al servidor.

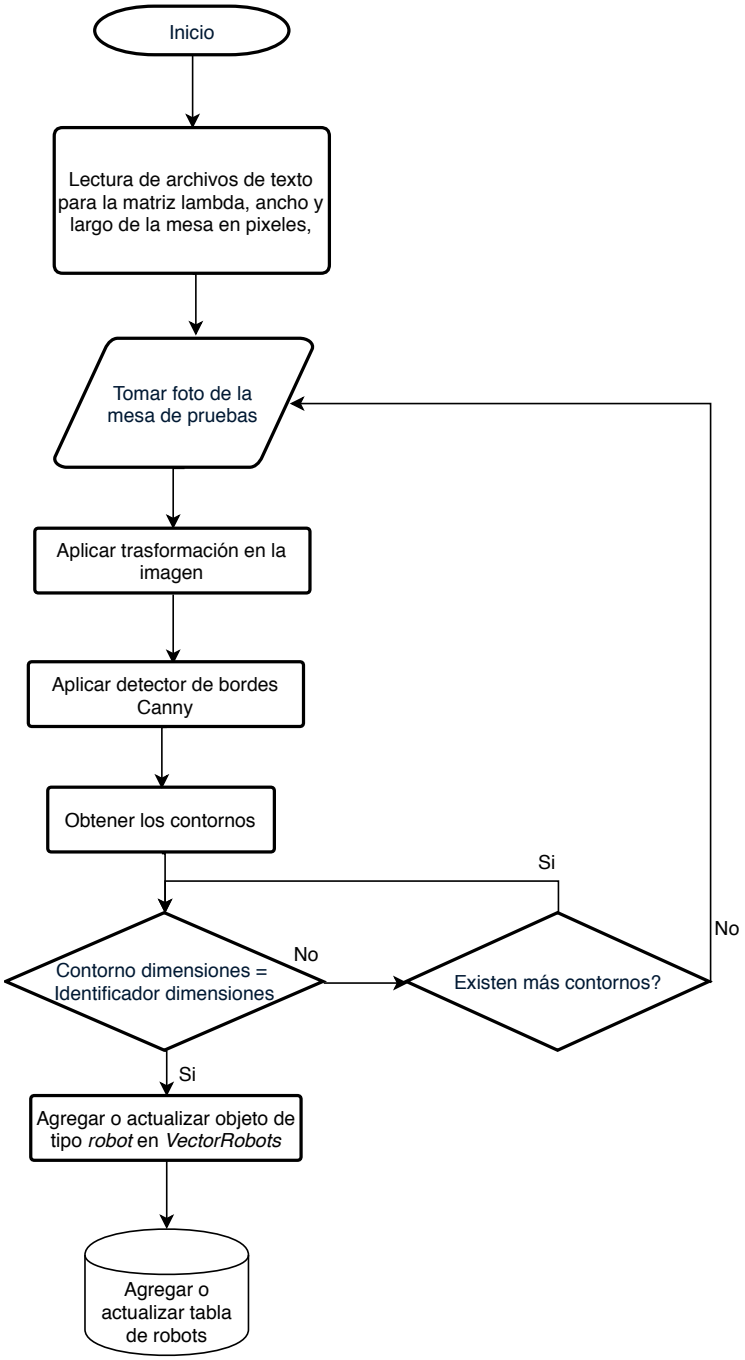


Figura 39: Diagrama de flujo de programa para obtención de pose.

10.4. Creación de interfaces gráficas mediante QT5

La idea es colocar la cámara en la mesa de trabajo, ésta solo tiene como objetivo capturar su totalidad, se pasa a ejecutar el programa para calibrar la cámara y se guardan los datos para generar la matriz *lambda*. Se ejecuta el programa principal ingresando el tamaño del identificador. Se ingresan los identificadores para tomar las posiciones y una vez terminada esta selección se pasa a la captura de datos. Se obtiene el numero del identificador y se sube a la base de datos la pose con respecto a la mesa. Los datos se toman a una frecuencia de 10Hz.

Todo esto se realizó en IDE de Qt5. Qt 5 es la última versión de Qt. Permite desarrollar aplicaciones con interfaces de usuario intuitivas para objetivos múltiples. Con la integración de código en lenguaje C++.

Las ventajas de QT para nuestra aplicación es que cuenta con: *Timers*, necesarios para la ejecución a un periodo constante de los métodos; librería de bases de datos en código *PostgreSQL* que nos permite el insertar, eliminar y actualizar elementos en una tabla.

A continuación se muestran las interfaces gráficas finales:

10.4.1. Creador de códigos

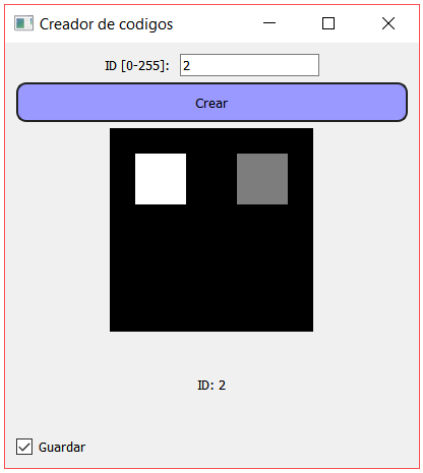


Figura 40: Interfaz gráfica de creador de identificadores.

10.4.2. Calibración de tablero

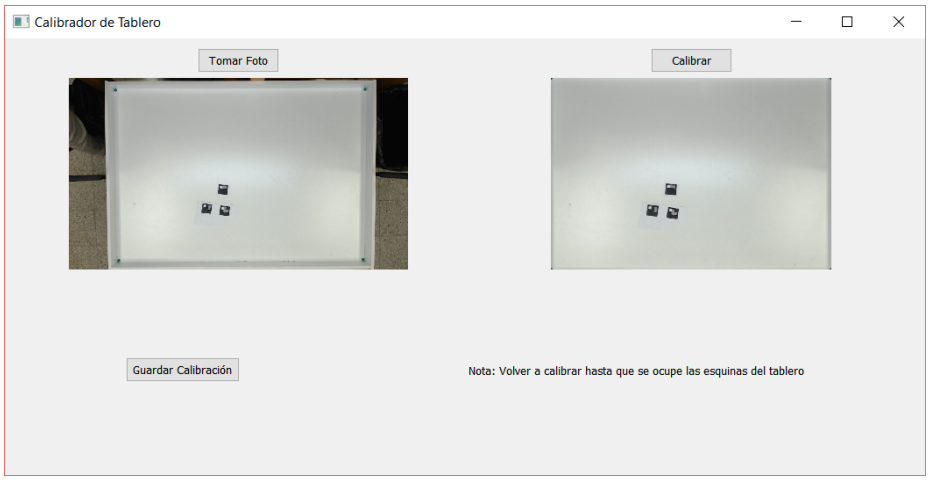


Figura 41: Interfaz gráfica de calibración de tablero.

10.4.3. Obtención de pose

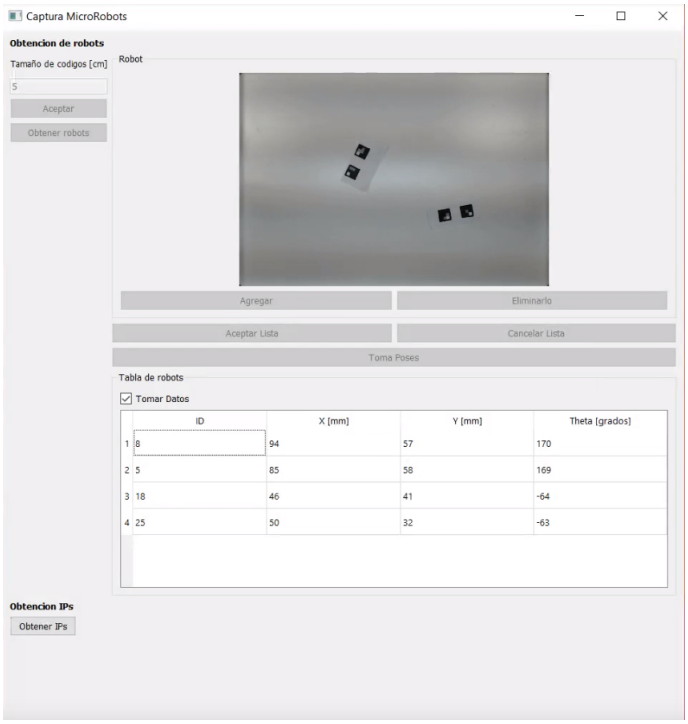


Figura 42: Interfaz gráficas de obtención de pose.

Conexión a la base de datos mediante PostgreSQL

Para la conexión con el protocolo de comunicación solo fue necesario establecer una red LAN donde las dos computadoras estuvieran conectadas. Mediante la librería en C++ de PostgreSQL se tuvo acceso a la tabla que contenía los datos, para ello fue necesario ingresar el usuario, contraseña, nombre del servidor y nombre de la tabla. Con esta prueba dio por finalizada la fase de conexión con el protocolo de comunicación.

CAPÍTULO 11

Tiempo de muestreo y precisión y exactitud de algoritmo

11.1. Medición de rendimiento

Para esta prueba se estableció un numero de tres agentes en la mesa, los cuales se estuvieron cambiando de posición durante la ejecución. Se tomaban los tiempos de máquina antes y después de la ejecución de la obtención de las poses. Luego de establecer la frecuencia de muestreo de 10Hz y ejecutando el código durante 10 segundos, se obtuvo el promedio de 100 datos los cuales mostraban el desempeño del algoritmo, estos son los resultados:

Promedio de tiempo	0.061s
Desviación estándar	0.005s
Valor máximo	0.069s

Cuadro 3: Tabla de estadísticas de tiempos de ejecución

De la tabla 3 se llegó a la conclusión que un muestreo de 10Hz era suficiente para capturar el movimiento "continuo" para el control de los tres robots. Este tiempo de muestreo es mayor al valor máximo que se obtuvo.

11.2. Pruebas estadísticas para el programa

Ahora la siguiente parte era comprobar estadísticamente si el algoritmo al obtener la posición y ángulo del identificador se acercaba al valor medido físicamente.

Para ello fue necesario realizar una prueba T pareada para identificar si existen diferencias estadísticamente significativas entre los grupos de datos medidos físicamente y los

obtenidos con el software. Se estableció un mínimo de 30 datos los cuales fueron generados aleatoriamente en los rangos de las dimensiones del tablero, a continuación se muestran los resultados obtenidos:

La prueba T pareada nos permite conocer si esa variable se debe al efecto de la variable independiente y si se acepta o rechaza la hipótesis de trabajo.

$$n = 30$$

datos medidos: x (cm), y (cm), θ (°)
datos software: x_s (cm), y_s (cm), θ_s (°)
error en x: $\sum_{i=1}^{30} x_i - x_{s_i} = 0$
error en y: $\sum_{i=1}^{30} y_i - y_{s_i} = 0$
error en ángulo: $\sum_{i=1}^{30} \theta_i - \theta_{s_i} = 0$

La suma de los errores en $x - x_s$, $y - y_s$ y $\theta - \theta_s$ fueron en los tres casos 0. A partir de ello se concluyó que no existe ningún efecto de la variable independiente (medición a mano) con la medición del software. Por lo que se puede asegurar la exactitud y precisión del algoritmo, siempre y cuando se trabaje con números enteros en los tres casos.

CAPÍTULO 12

Conclusiones

1. Se observó que el detector de bordes Canny es el que menos influencia tiene sobre la luz que existe en el ambiente para detectar los identificadores. A pesar de tener una lámpara que apuntara directamente a la mesa seguía existiendo una influencia del ambiente sobre ella.

2. Se encontró estadísticamente que no existe una diferencia significativa en la medida obtenida por el algoritmo en comparación con la medida de forma física, en consecuencia la medida del software tiene una alta precisión y exactitud, siempre y cuando las variables se trabajen con números enteros, donde x y y estaban dados en centímetros y θ en grados.

3. Trabajar con una resolución de 920x760 permite alcanzar tamaños de identificadores hasta de 3x3 cm, de lo contrario el patrón binario no pueden ser detectado correctamente.

4. Se obtuvo que la frecuencia de muestreo es dependiente de la cantidad de objetos en la mesa así como la resolución de la imagen, lo que implica en que si se tienen más micro-robot se tardará más el algoritmo en encontrar la información de los identificadores.

5. Utilizar una base de datos permite aislar el sistema de visión por computadora e implementar el código de control de los micro-robots en otra computadora, acelerando el proceso de diseño, minimizando la posibilidad de errores y asegurar la tasa de muestreo constante en la computadora.

CAPÍTULO 13

Recomendaciones

1. Existen otros filtros en OpenCV que no fueron utilizados para la extracción del identificador por cuestiones de tiempo. Con la bibliografía mencionada en el trabajo se pueden investigar otras alternativas que pueden resultar más eficientes.

2. Es necesario escoger una cámara que no genere distorsiones radiales o tangenciales en la imagen. Con una cámara de gamma media es posible obtener fotos con baja distorsión sin sacrificar tiempo de ejecución para la corrección de la imagen.

3. Existen otros métodos para detección de micro-robots que no involucran una cámara, por ejemplo un proyector con foto-receptores como el que cuentan los Zooids, que les permite altas tasas de muestreo sin dependencia de la cantidad de robots y sin comprometer el rendimiento. El fin sería desarrollar uno de estos dispositivos a un precio más accesible para así aplicarlo en una cama de pruebas.

4. Es posible utilizar *threads* en QT5 para aumentar el muestreo de los identificadores, estos corren procesos en diferentes partes del núcleo siendo independientes de otros. A estos se les puede asignar su nivel de importancia. Su desventaja radica en que pueden suceder múltiples errores en su aplicación, ya que eventos pueden colisionar o variables pueden eliminarse mientras otros quieren su acceso.

5. Utilizar arquitectura CUDA para la reducción de tiempos en los algoritmos de procesamiento de imágenes.

CAPÍTULO 14

Bibliografía

[1] D. Pickem, L. Wang, P. Glotfelter, Y. Diaz-Mercado, M. Mote, A. Ames, E. Feron y M. Egerstedt, "Safe, remote-access swarm robotics research on the robotarium", *arXiv preprint arXiv:1604.00640*, 2016.

[2] M. Le Goc, L. H. Kim, A. Parsaei, J.-D. Fekete, P. Dragicevic y S. Follmer, "Zooids: Building Blocks for Swarm User Interfaces", en, en *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*, Tokyo, Japan: ACM Press, 2016, págs. 97-109, ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984547. dirección: http://dl.acm.org/citation.cfm?doid=2984511.2984547 (visitado 17-09-2018).

[3] M. Rubenstein, C. Ahler y R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors", en *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, págs. 3293-3298.

[4] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas y M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion", *Pattern Recognition*, vol. 47, n.º 6, págs. 2280-2292, jun. de 2014, ISSN: 00313203. DOI: 10.1016/j.patcog.2014.01.005. dirección: http://linkinghub.elsevier.com/retrieve/pii/S0031320314000235 (visitado 06-06-2018).

[5] G. Bradski y A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[6] J. Molleda Meré, *Técnicas de visión por computador para la reconstrucción en tiempo real de la forma 3D de productos laminados*. 2008.

[7] D. Mery, "Visión por computador", *Santiago de Chile. Universidad Católica de Chile*, 2004.

[8] S. S. Mohammadi, "Computer vision for autonomous mobile robotic applications", 2011.

[9] J. M. G. Hernández, "Sistema de visión para agricultura de precisión: identificación en tiempo real de líneas de cultivo y malas hierbas en campos de maíz", Tesis doct., Universidad Complutense de Madrid, 2015.

[10] L. E. Sucar y G. Gómez, "Visión computacional", *Instituto Nacional de Astrofísica, óptica y Electrónica. México*, 2011.

[11] R. Siegwart e I. R. Nourbakhsh, *Introduction to autonomous mobile robots*, en, ép. Intelligent robots and autonomous agents. Cambridge, Mass: MIT Press, 2004, ISBN: 978-0-262-19502-7.

[12] D. Robot, "Robot Daro: plataforma robótica", es, *Revista de la Facultad de Ingeniería*, pág. 17, 2017.

[13] *Springer handbook of robotics*, en, 2nd edition. New York, NY: Springer Berlin Heidelberg, 2016, ISBN: 978-3-319-32550-7.

[14] F. Bullo, J. Cortés y S. Martínez, *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*, en. Princeton: Princeton University Press, ene. de 2009, ISBN: 978-1-4008-3147-0. DOI: 10.1515/9781400831470. dirección: https://www.degruyter.com/view/books/9781400831470/9781400831470/9781400831470.xml (visitado 17-09-2018).

[15] G. R. Bradski y A. Kaehler, *Learning OpenCV: computer vision with the OpenCV library*, en, 1. ed., [Nachdr.], ép. Software that sees. Beijing: O'Reilly, 2011, OCLC: 838472784, ISBN: 978-0-596-51613-0.