Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Localización y mapeo simultáneos con marcadores y un robot móvil

Tesis que presenta

Andres Cureño Ramirez

para obtener el Grado de

Maestro en Ciencias en Computación

Director de Tesis

Dr. Luis Gerardo de la Fraga

Ciudad de México **Septiembre de 2022**

Resumen

En este trabajo se demuestra la fiabilidad de los marcadores de tipo de orden para resolver el problema de localización y mapeo simultáneos. Teniendo una cámara ya calibrada, los marcadores de tipo de orden, como son un objeto conocido, permiten conocer la posición métrica del robot con respecto al marcador usando directamente los puntos reconocidos en la imagen que ve el robot. El problema consiste en realizar un procedimiento para construir un mapa 2D de un entorno desconocido utilizando los marcadores y donde también se estima la trayectoria de un robot al desplazarse dentro del entorno que va conociendo al recorrerlo. Los marcadores de tipo de orden también permiten asignar un identificador distinto a cada marcador. En el ambiente a reconocer se asignan marcadores con un mismo identificador a las paredes del ambiente y otros identificadores distintos a los objetos dentro del ambiente. El robot móvil tiene forma de coche y está equipado con una RaspberryPi 3 B+, el módulo de cámara de la RaspberryPi y un sensor ultrasónico. El entorno donde el robot se desplaza, tiene las siguientes dimensiones: 1.2 m $\times$ 2 m. Dentro del entorno, hay objetos que pueden cambiarse de posición de forma manual para crear diferentes entornos; las paredes y los objetos tienen marcadores para su detección. Para resolver el problema se realizan los siguientes pasos: 1) se coloca el robot dentro del ambiente, 2) el robot realiza una etapa de exploración, 3) en la etapa de exploración el robot detecta los marcadores en el ambiente y los procesa, 4) el robot estima su posición y crea o actualiza el mapa, el cuál se almacena en un árbol binario, 5) cuando termina la etapa de exploración se optimiza el mapa obtenido, disminuyendo el error de reproyección de los marcadores detectados y 6) se traza el mapa final utilizando los datos del árbol binario.

Abstract

This work demonstrate the reliability of order type markers for solving the simultaneos localisation and mapping problem. Having a camera already calibrated, order type markers, being a known object, allow to know the metric position of the robot with respect to the marker by directly using the recognised points in the image seen by the robot. The problem consists of a procedure to construct a 2D map of an unknown environment using the markers and where the trajectory of a robot is also estimated as it moves through the environment that it learns as it moves through it. The order type markers also allow a different identifier to be assigned to each marker. In the environment to be recognised, markers with the same identifier are assigned to the walls of the environment and different identifiers are assigned to the objects within the environment. The mobile robot is shaped like a car and is equipped with a RaspberryPi 3 B+, the RaspberryPi camera module and an ultrasonic sensor. The environment in which the robot moves has the following dimensions: 1.2 m $\times$ 2 m. Within the environment, there are objects that can be manually repositioned to create different environments; walls and objects have markers for detection. To solve the problem the following steps are performed: 1) the robot is placed inside the environment, 2) the robot performs an exploration stage, 3) in the exploration stage the robot detects the markers in the environment and processes them, 4) the robot estimates its position and creates or updates the map, which is stored in a binary tree, 5) when the exploration stage is finished the obtained map is optimized, decreasing the re-projection error of the detected markers and 6) the final map is drawn using the data from the binary tree.

Agradecimientos

A CONACYT por el apoyo económico para estudiar la maestría y al Cinvestav por los conocimientos y espacios brindados.

A mi Mamá y a mi Papá que me han apoyado durante toda mi vida.

A las Doctoras y los Doctores que me impartieron clases, en especial a mi asesor y a mis sinodales.

A mis amigas y amigos que me acompañaron durante esta etapa.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Planteamiento del problema

El problema de localización y mapeo simultáneo [3] consiste en que se tiene un robot móvil que navega en un ambiente desconocido y comienza en una posición inicial arbitraria. Los movimientos del robot generan cierta incertidumbre lo que hace que sea cada vez más complicado determinar su posición actual en coordenadas globales conforme se desplaza dentro del entorno. Mientras el robot se desplaza puede utilizar un sensor para reconocer lo que lo rodea. Por lo tanto el problema es ir construyendo un mapa del entorno mientras que simultáneamente se estima la posición del robot en el mapa con cierta incertidumbre, esto debido al ruido que puede existir de los datos obtenidos por los sensores del robot.

En esta tesis se resolverá el problema de localización y mapeo simultáneo utilizando marcadores. Por lo tanto se tiene un robot móvil que se mueve en un entorno donde hay marcadores, los cuáles ya se conocen. El robot cuenta con una cámara que ya está calibrada, por lo que se conocen de antemano los valores de los parámetros de la cámara y esta es el sensor que le permite al robot reconocer el entorno, en donde hay marcadores. A partir de la imagen que captura el robot de un marcador se puede calcular su rotación y traslación con respecto al marcador (o viceversa, la rotación y posición del marcador con respecto a la cámara). Con esta información el robot debe construir un mapa del entorno con marcadores e ir estimando su posición actual dentro del mapa.

Una vez se termina de crear el mapa del entorno con marcadores con el robot, este debe ser optimizado, disminuyendo el error de reproyección, ya que el desplazamiento del robot y la cámara pueden crear una incertidumbre alta de la posición de los objetos en el mismo. Por lo tanto una vez terminada la optimización mapa, el robot debe ser capaz de navegar en el.

1.2. Objetivos

Objetivo general

Realizar la localización y mapeo, utilizando marcadores de tipo de orden, para un robot móvil en un entorno con iluminación uniforme.

Objetivos específicos

- Estudiar y utilizar o modificar el software de desarrollo del robot, para realizar las tareas necesarias.
- Construir un entorno de pruebas y los obstáculos del entorno.
- Reducir la incertidumbre que se crea al elaborar el mapa.
- Realizar estrategias de reconocimiento y mapeo en un entorno sin obstáculos.
- Realizar estrategias de reconocimiento y mapeo en un entorno con obstáculos.
- Realizar estrategias para navegar en el mapa generado en un entorno con obstáculos.

1.3. Estado del arte

Actualmente, existen diversos algoritmos para realizar la localización y mapeo, también conocidos como *localización y mapeo simultáneo (LMS)*, en inglés es *Simultaneous localization and mapping (SLAM)*. Uno de ellos es el SPM-SLAM [4], en donde únicamente se usan marcadores para realizar el mapa y la localización. En este algoritmo la corrección del mapa se realiza utilizando la detección de ciclos, lo cual permite corregir la deriva y distribuye el error a lo largo del ciclo y luego se realiza una optimización.

Otro tipo de algoritmo es el LSD-SLAM [5], que en lugar de utilizar las características obtenidas de la imagen, para la estimación de la posición de la cámara, utilizan un enfoque directo que consiste en minimizar el error fotogramétrico, utilizando un enfoque probabilístico para incorporar el ruido en los mapas de profundidad estimados. Para resolver los problemas de relocalización y cierre de ciclos, utilizan las características de los puntos clave y también se realiza al final una optimización del mapa.

Uno de los algoritmos para realizar la LMS más conocidos y utilizados es el ORB-SLAM [6]. Éste utiliza la extracción de características utilizando ORB-Keypoints, que se usan para seguimiento y relocalización y hace corrección de ruido cuando encuentra un ciclo en el grafo que se va creando. Uno de los problemas es que no se conoce la escala del mapa y

se requiere que exista mucha textura, que puede definirse como una función de variación espacial de la intensidad de brillo de los píxeles [7].

El siguiente algoritmo, UcoSLAM [8], combina la extracción de características de la escena y la búsqueda de marcadores. Al encontrar algún marcador dentro de la escena, le permite conocer la escala del mapa y hacer correcciones a la deriva que pueda ir presentando el mapa, creado con la nube de puntos obtenidos de la extracción de características. También utiliza la detección de ciclos para corregir el error generado, pero puede detectar un ciclo utilizando las características obtenidas de la imagen o con los marcadores. Una vez corregido el error, también realiza una optimización del mapa.

Uno de los más actuales es Blitz-SLAM [9], pensado principalmente para entornos dinámicos. Está basado en ORB-SLAM2 y se combina con BlitzNet, que es una red de aprendizaje profundo, para obtener información semántica de los objetos en el entorno. Al combinar las máscaras de los objetos en movimiento obtenidas con BlitzNet, con una imagen de profundidad, se pueden eliminar esos objetos de la nube de puntos del mapa. Al estar basado en ORB-SLAM, toma las mismas técnicas para hacer la corrección del error generado.

1.4. Antecedentes

Los marcadores de tipo de orden se propusieron en la tesis de doctorado de Heriberto Cruz Hernández [10], *Solucion Simultánea de Varios Subproblemas de Visión por Computadora*. Estos marcadores permiten encontrar automáticamente el emparejamiento de los puntos del modelo del marcador con los puntos del marcador reconocidos en la imagen que ve el robot. El emparejamiento se realiza con la matriz lambda que representa el tipo de orden de los puntos. En esta tesis se describirán unos marcadores con seis puntos optimizados [11] para que resistan más el posible ruido en la captura de la imagen y en el reconocimiento del marcador, como se describirá en el capítulo 3.

En la tesis *Prototipo de un sistema de ojo de halcón* [1], se realiza una propuesta para la detección de los marcadores de tipo de orden. En la figura 1.1 se muestra un diagrama del clasificador propuesto para la detección de los marcadores. En esta tesis se usa un modelo distinto, de alto nivel, para detectar el marcador: el marcador es un objeto obscuro con cuatro esquinas (la proyección de un cuadrado).

En la tesis *Sistema de odometría visual e inercial con un marcador* [2], también se utilizaron los marcadores de tipo de orden para obtener el sistema de odometría visual e inercial, como se puede ver en la figura 1.2.

Ya se ha intentado reconocer los marcadores basados en tipo de orden con redes pro-

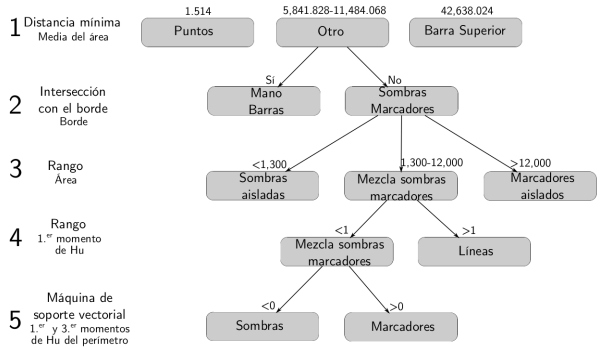


Figura 1.1: Diagrama del clasificador de marcadores de tipo de orden [1].

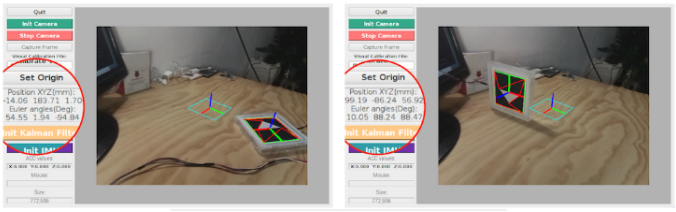


Figura 1.2: Sistema de odometría visual e inercial con un marcador [2].

fundas en la tesis de maestría [12].

1.5. Organización de la tesis

En el capítulo 1 se plantea el problema a resolver, los objetivos, el estado del arte relacionado al problema y los antecedentes para realización de esta tesis. En el capítulo 2 se presentan los conocimientos previos necesarios para la realización de esta tesis. En el capítulo 3 se explica el análisis y cómo se resolvieron los objetivos planteados en el capítulo 1. En el capítulo 4 se muestran los resultados obtenidos al realizar diferentes tareas. Finalmente, en el capítulo 5 se describen las conclusiones obtenidas de este trabajo y los posibles trabajos a futuro que se podrían realizar.

Capítulo 2

Marco teórico

En este capítulo se describirá el modelo de la cámara oscura, en el que se basa todo el desarrollo de la tesis. También se describirán los marcadores de tipo de orden, la cinemática de un robot diferencial, para entender el movimiento del robot que se usa, y el posicionamiento en un sistema global de varios sistemas locales, que es como se va construyendo el mapa que va viendo el robot. Al final del capítulo se describe el ajuste total, que es la optimización no lineal del mapa realizado.

2.1. Modelo de la cámara oscura

Tenemos un mundo tridimensional que se proyecta a una imagen, a un plano en 2D. El modelo de la cámara oscura nos describe la ecuación para realizar esa proyección. La cámara se sitúa en el origen del sistema de coordenadas, viendo hacia el eje z negativo y podemos dibujar las dos proyecciones, como se muestra en la figura 2.1 y se define lo siguiente $P = [x\, y\, z]^T$ y $p = [u\, v]^T$, esto es, P es un punto tridimensional del mundo y p su punto bidimensional proyectado en el plano de proyección, que es el plano de la imagen que toma la cámara.

Viendo la figura 2.1, por triángulos semejantes:

$$\frac{y}{z} = \frac{v}{-f}, \quad v = \frac{-fy}{z} \tag{2.1}$$

$$\frac{x}{z} = \frac{u}{-f}, \quad u = \frac{-fx}{z} \tag{2.2}$$

Así f, la distancia focal, se puede ver como un factor de escala. El valor del punto p en el plano de proyección es inversamente proporcional al valor de z, como se ve en las ecuaciones (2.1) y (2.2). Entonces, las cosas más alejadas se verán más pequeñas en la

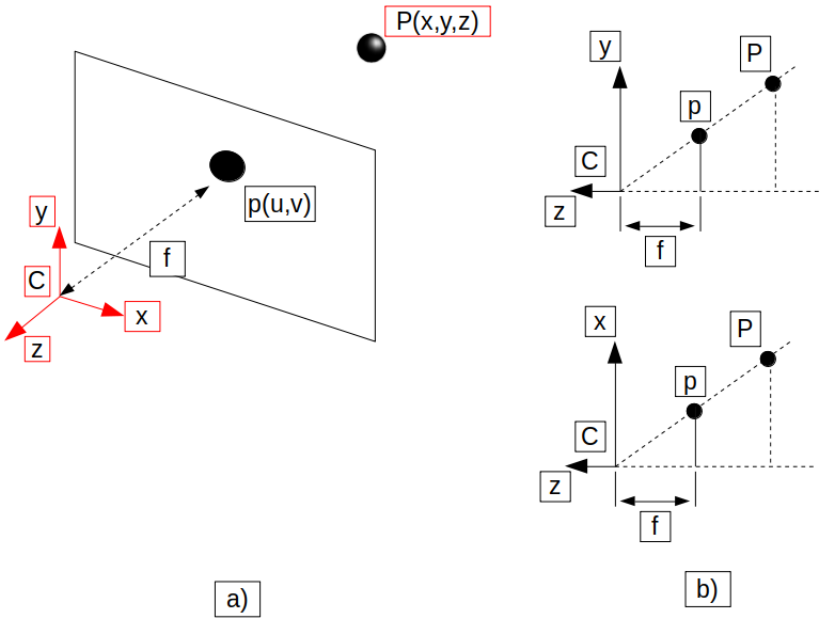


Figura 2.1: Esquema de la proyección en perspectiva. El punto tridimensional P se proyecta en el punto p sobre el plano de proyección. El plano de proyección está situado a una distancia $-f$ hacia el eje z negativo.

imagen. A esta deformación se le llama escorzado. Las ecuaciones (2.1) y (2.2) se pueden representar de forma matricial como se muestra a continuación:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.3}$$

$$\lambda p = KP \tag{2.4}$$

En la ecuación (2.3), a la coordenada z se le cambia el signo, que es otra forma de representar el cambio de signo en la distancia focal de las ecuaciones (2.1) y (2.2). Pero en la ecuación (2.4) el punto 3D no es homogéneo; si se hace homogéneo tenemos lo siguiente:

$$\lambda p = KAP \tag{2.5}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.6}$$

$$\lambda p = K[R|t]P \tag{2.7}$$

En la ecuación (2.7), R es una matriz de rotación y t es un vector de traslación en $\mathbb{R}^3$. En esta misma ecuación, el punto 3D P se rota por R y luego se traslada por los valores del vector t.

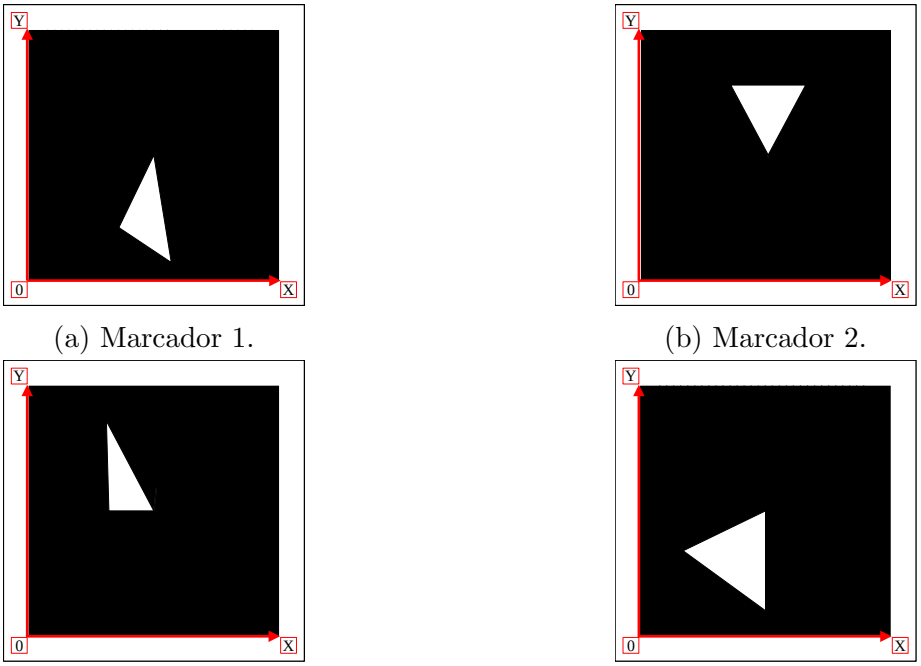
2.2. Marcadores de tipo de orden

Los marcadores de tipo de orden [13] están compuestos de tres elementos principales: un área sin ruido, un área de datos y los puntos de la etiqueta. El área sin ruido es una zona blanca que contiene a los demás elementos del marcador. Su propósito es ayudar a que el área de datos pueda ser detectada completamente dado que sirve como un separador de el área de datos y de otros objetos de la escena. El área de datos es un cuadrado negro que funciona simultáneamente como patrón de búsqueda. Su propósito es servir como el objeto más fácil de identificar (se supone que es el objeto negro más grande de la escena). Al mismo tiempo sirve para delimitar el segmento de la imagen donde se encuentran los puntos de etiqueta. Los puntos de etiqueta constituyen el conjunto de puntos y su disposición define el tipo de orden y un identificador único (un valor entero) para el marcador. Los puntos se dan como vértices de triángulos para ayudar al sistema a encontrarlos en una imagen. Los triángulos permiten calcular las posiciones de los vértices como la intersección de los lados con una precisión inferior a un píxel. Esto se logra porque cada lado se estima con la dirección del componente principal de todos los píxeles que lo forman.

Estos marcadores están representados por seis puntos, las cuatro esquinas del área de datos (cuadrado negro) y dos vértices del triangulo que se forma dentro del área de datos, el vértice del centro del área de datos no se toma en cuenta, en la figura 2.2 se muestran los marcadores de tipo de orden utilizados.

El tipo de orden de un conjunto de puntos se puede representar mediante una matriz lambda, la cuál se crea de la siguiente forma:

- Se crea una matriz de $n \times n$ para n puntos,
- n columnas de los puntos p_1, p_2, ..., p_n,
- n filas de los puntos p_1, p_2, ..., p_n,
- se traza una línea de p_i hacia p_j $(i \neq j)$
- y se cuentan los puntos que están a la izquierda de la línea.



(a) Marcador 1. (b) Marcador 2.

(c) Marcador 3. (d) Marcador 4.

Figura 2.2: Marcadores de tipo de orden utilizados.

Esto se hace para cada punto de la cubierta convexa del conjunto de puntos, por lo tanto para los marcadores de seis puntos se obtienen cuatro matrices lambda. Si existe una matriz lambda mínima lexicográfica, las etiquetas de los puntos obtenida es única y tiene las mismas etiquetas que el modelo del marcador sin estar rotado, trasladado y proyectado.

En la tabla 2.1 se muestran las coordenadas de los puntos de los marcadores que se utilizaron en este trabajo. Con estos puntos se crearon los marcadores que se observan en la figura 2.2.

Marcador 1	
X	Y
0	0
255	0
145	18
92	54
255	255
0	255

Marcador 2	
X	Y
255	255
0	255
90	199
165	199
0	0
255	0

Marcador 3	
X	Y
255	255
0	255
80	221
81	127
0	0
255	0

Marcador 4	
X	Y
0	255
0	0
44	86
127	25
255	0
255	255

Tabla 2.1: Coordenadas de los marcadores utilizados.

La primera matriz lambda para el marcador de la figura 2.2a, es la siguiente:

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

Debido a la relación de la parte superior de la matriz con la parte inferior, $\lambda_{ij} = n - 2 - \lambda_{ji}$, no es necesario calcular las matrices completas, por lo tanto la matriz lambda queda de la siguiente forma:

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

La matriz superior triangular de la matriz lambda se puede almacenar en un vector, comenzando por la primera fila. A continuación se muestran los vectores que representan las cuatro matrices lambda del marcador de la figura 2.2a. La matriz lambda anterior corresponde al vector v_2.

$$v_1 = [4, 3, 2, 1, 0, 1, 2, 4, 3, 1, 3, 2, 2, 1, 4]$$
$$v_2 = [4, 3, 2, 1, 0, 4, 2, 1, 3, 3, 2, 4, 3, 2, 1]$$
$$v_3 = [4, 3, 2, 1, 0, 4, 3, 2, 1, 2, 3, 4, 3, 2, 3]$$
$$v_4 = [4, 3, 2, 1, 0, 2, 3, 4, 1, 3, 2, 2, 3, 3, 4]$$

Se puede observar que el vector v_1 es el mínimo lexicográfico, por lo tanto los vectores que representan las matrices lambda mínimas lexicográficas de los marcadores de la figura 2.2 son los siguientes:

Marcador 1: $[4, 3, 2, 1, 0, 1, 2, 4, 3, 1, 3, 2, 2, 1, 4]$
Marcador 2: $[4, 3, 2, 1, 0, 2, 1, 4, 3, 2, 3, 2, 2, 1, 4]$
Marcador 3: $[4, 3, 2, 1, 0, 1, 3, 4, 2, 2, 3, 1, 2, 3, 4]$
Marcador 4: $[4, 3, 2, 1, 0, 1, 3, 4, 2, 3, 2, 1, 3, 3, 4]$

Para asignarles un identificador único a cada marcador es posible usar los valores en los índices 8 y 9 de cada vector.

2.3. Robot diferencial

La dirección de un robot diferencial controla de forma independiente la velocidad de cada una de las ruedas motrices fijas, es decir que no se puede cambiar su dirección y si la velocidad de las llantas no es igual, el vehículo va a girar [14]. Los robots diferenciales muy simples pueden estar compuestos por dos ruedas motrices fijas y una rueda que se ancla a la superficie con una estructura, la cuál tiene un eje en su centro anclado a la rueda, que le permite girar libremente de forma perpendicular a la rotación de la rueda. Un ejemplo de vehículos diferenciales más complejos son los tanques militares que utilizan orugas.

En la figura 2.3 se muestran los marcos referencia del robot utilizado, el marco de referencia del robot es MR, el robot solo tiene dos grados de libertad, es decir que únicamente puede desplazarse de forma lineal a lo largo del eje x y rotar a lo largo del eje z. El marco de referencia del servomotor uno es MS1, este únicamente puede rotar alrededor del eje z' y es relativo a MR y controla la orientación de la cámara. El marco de referencia del servomotor dos es MS2 el cuál solamente puede rotar sobre el eje y" y es relativo a MS1 y controla la inclinación de la cámara. Las llantas que están del lado izquierdo del robot se mueven con la misma velocidad angular ω_1 y las llantas que se encuentran del lado derecho del robot se mueven con la misma velocidad angular ω_2, por lo tanto solo existen dos velocidades angulares ω_1 y ω_2 para las llantas.

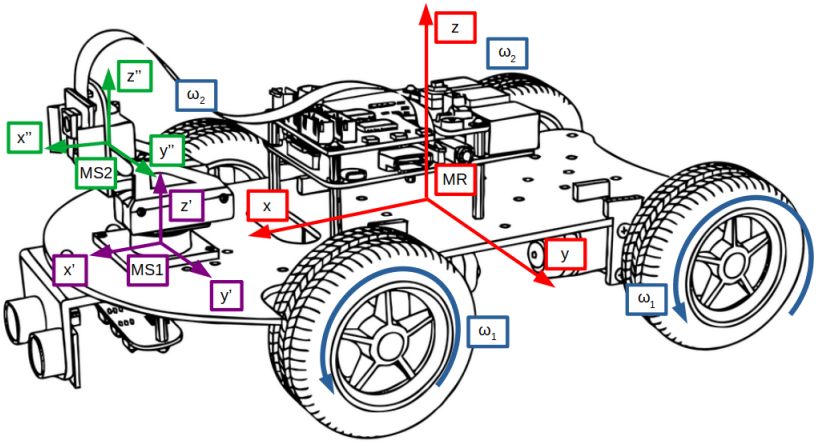


Figura 2.3: Marcos de referencia del robot utilizado. MR es el sistema local al robot. Los sistemas MS1 y MS2 indican la rotación e inclinación, respectivamente, de la cámara.

En la figura 2.4 se muestra un diagrama de los movimientos que se pueden hacer con el robot utilizado, si las 4 ruedas giran en la misma dirección el robot va avanzar o retroceder, si el sentido de rotación de las ruedas de la izquierda es diferente al de las ruedas de la

derecha el robot va a girar y si las ruedas de un lado no se mueven y las del lado contrario giran en cualquier sentido el robot va a avanzar y girar. Es importante mencionar que el movimiento del robot no depende tan sólo del sentido de giro de las ruedas, depende también de la magnitud de sus velocidades angulares.

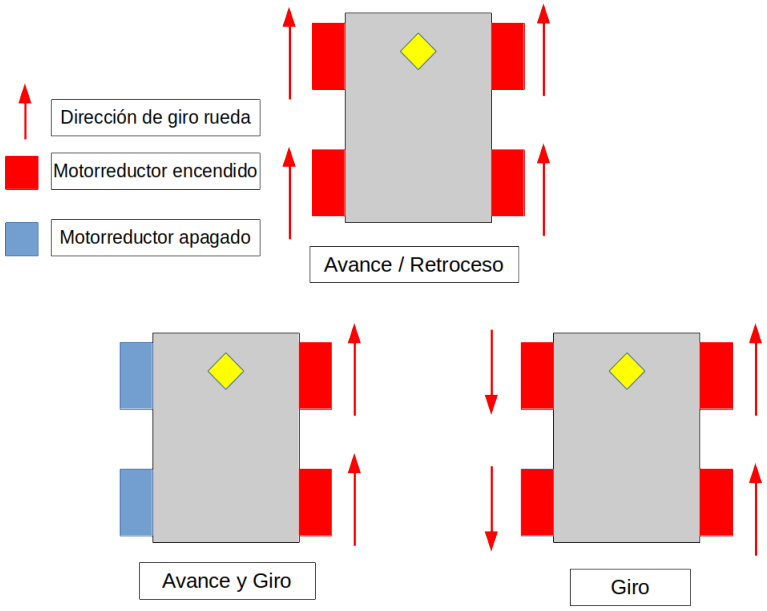


Figura 2.4: Tipos de movimientos del robot.

2.4. Posicionamiento en un sistema de coordenadas global de varios sistemas locales

En la figura 2.5 se presenta el siguiente problema en donde se tienen los marcos de referencia locales de 3 marcadores (P_1, P_2 y P_4) en rojo, y en verde se representan distintas posiciones de una cámara (F_1 y F_2) con sus marcos de referencia locales en azul. En la cámara F_1 se genera una imagen donde se ven los marcadores P_1 y P_2 y en la cámara F_2 se genera una imagen donde se ven los marcadores P_3 y P_4. Entonces lo que se busca es situar todos los sistemas de coordenadas locales en un mismo marco de referencia.

Tal vez una forma más intuitiva de la ecuación (2.7) es escribirla de la siguiente forma:

$$\lambda p = KR[I|-c]P \tag{2.8}$$

donde c es el origen de la cámara, con respecto a las coordenadas locales de cada marcador y o es un punto delante de la cámara, el eje y local se calcula como $y = (o-c)/\|o-c\|$ y

el eje x se calcula como $x = y \times [0, 0, 1]^T$. Y la matriz de rotación será:

$$R = \begin{bmatrix} x^T \\ y^T \end{bmatrix}$$

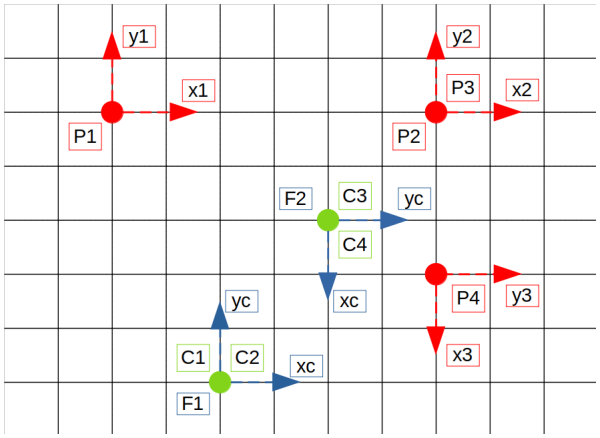


Figura 2.5: Diagrama del problema, en rojo los marcos de referencia locales de los marcadores y en verde las posiciones de la cámara con sus marcos de referencia locales.

A partir de la figura 2.5 para la cámara 1 se tiene $c_1 = [2, -5]^T$ con $o_1 = [2, 1]^T$ y $c_2 = [-4, -5]^T$ con $o_2 = [-4, 2]^T$, de donde se obtiene lo siguiente:

$$R_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, -c_1 = \begin{bmatrix} -2 \\ 5 \end{bmatrix},$$

$$R_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, -c_2 = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

A partir de la figura 2.5 para la cámara 2 se tiene $c_3 = [-2, -2]^T$ con $o_3 = [1, -2]^T$ y $c_4 = [-1, -2]^T$ con $o_4 = [-1, 2]^T$, de donde se obtiene lo siguiente:

$$R_3 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, -c_3 = \begin{bmatrix} 2 \\ 2 \end{bmatrix},$$

$$R_4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, -c_4 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Se puede definir lo siguiente, en donde R es una matriz de rotación y $T(a)$ es una matriz de traslación:

$$R = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

$$T(-c) = \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

$$M = R\,T(-c) \tag{2.11}$$

$$M^{-1} = T(c)R^T \tag{2.12}$$

Con la ecuación (2.11), podemos definir las matrices de transformación geométrica de cada marcador de la forma $M_i = R_i T(-c_i)$, para $i = \{1, 2, 3, 4\}$.

Sin embargo, una matriz del tipo

$$\begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix},$$

que se usa en el modelo de la cámara oscura en la ecuación (2.7), tiene la interpretación que el punto P primero se rota por R y luego se traslada por el vector t, así:

$$T(t)R = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Una vez definidas estas transformaciones geométricas, es posible situar todos los sistemas de coordenadas locales en un mismo sistema de coordenadas global. La situación de este sistema de coordenadas global es arbitrario. Aquí se escogerá el sistema local 1 como el de referencia para construir el mapa global:

$$M_1' = M_1^{-1}\, M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} M_2' &= M_1^{-1}\, M_2 \\ &= T(c_1)\, R_1^T\, R_2\, T(-c_2) \\ &= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Se observa en la figura 2.5 que $P_2 = P_3$. Por lo tanto M_2 y M_3 son transformaciones para un mismo modelo, entonces lo que queremos es una matriz que nos produzca el mismo efecto que M_2, como se muestra a continuación:

$$
\begin{aligned}
P_2' &= M_2 P_2, \\
P_3' &= M_3 P_3, \\
P_3' &= M_5 M_3 P_3, \text{ y de aquí} \\
M_2 &= M_5 M_3, \\
M_5 &= M_2 M_3^{-1}, \\
&= R_2\, T(-c_2)\, T(c_3)\, R_3^T, \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & -2 \\ -1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}, \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & -2 \\ -1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Es posible observar que M_5 nos indica la rotación de la cámara y es la que nos permite relacionar las dos imágenes 1 y 2.

$$
\begin{aligned}
M_4' &= M_1^{-1}\, M_5\, M_4 \\
&= T(c_1)\, R_1^T\, M_5\, R_4\, T(-c_4) \\
&= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} = \\
&= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 4 \\ -1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 4 \\ -1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 6 \\ -1 & 0 & -3 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

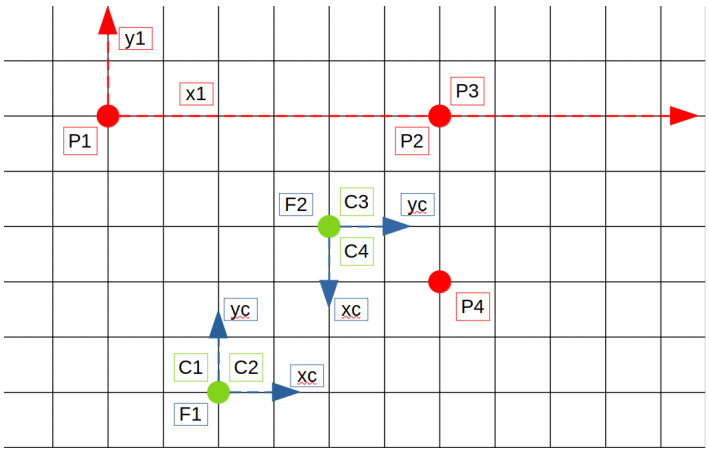


Figura 2.6: Diagrama en un mismo sistema de coordenadas global P_1.

Al hacer el cambio al sistema de coordenadas global de P_1 como se ve en la figura 2.6, es posible observar que M_1' no tiene ninguna rotación, ni traslación ya que ahora es el origen del sistema de coordenadas. M_2' representa que P_2 no tiene ninguna rotación respecto a P_1, pero si una traslación en el eje x_1. M_5 representa la rotación y traslación de 2 a 1. M_4' muestra una rotación de $-90°$ respecto a P_1, seguido de la traslación a $[6, -3]^T$.

2.5. Ajuste total

Se considera la siguiente situación con un conjunto de puntos 3D, P_j, vistos por un conjunto de cámaras con matrices de proyección (homografía) M^i. Si p_j^i son las coordenadas de el j-ésimo punto visto por la i-ésima cámara, y se quiere resolver el siguiente problema de reconstrucción: dado un conjunto de coordenadas de la imagen p_j^i, encontrar las matrices de la cámara M^i y los puntos P_j tal que $M^i P_j = \lambda\, p_j^i$.

Si las mediciones de la imagen tienen ruido, entonces las ecuaciones $M^i P_j = \lambda\, p_j^i$ no se van a satisfacer de manera exacta. Lo que se busca es estimar las matrices de proyección, $\hat{M}^i$, y los puntos 3D, $\hat{P}_j$, que se proyectan exactamente a puntos de la imagen $\hat{p}_j^i$. Como se muestra a continuación $\hat{p}_j^i = \hat{M}^i \hat{P}_j$ y también se busca minimizar la distancia de la imagen entre el punto reproyectado y los puntos de la imagen detectados (medidos) p_j^i, para cada imagen en la que aparece el punto 3D.

Esta estimación que consiste en minimizar el error de reproyección se conoce como ajuste total. Este método generalmente se usa al final de cualquier algoritmo de reconstrucción y tiene la ventaja de ser tolerante a datos faltantes, mientras que proporciona una verdadera estimación de máxima verosimilitud [15].

Dado que en los marcadores ya se conoce el modelo, P_j, y que solo se utilizará una cámara para estimar las rotaciones y traslaciones, entonces $i = 1$ y esto simplifica el problema. Por lo tanto, al usar marcadores, el método de ajuste total es más parecido al método que propone Zhang [16]. El ajuste total es la optimización no lineal para refinar las rotaciones y traslaciones de cada marcador, a partir de la solución inicial encontrada al problema lineal con el método de Zhang, en donde se propone la ecuación (2.13) para calcular el error de reproyección:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \|p_{ij} - \hat{p}(K, R_i, t_i, P_j)\|^2 \tag{2.13}$$

donde n es el número total de imágenes, m el número de puntos del modelo P, R es una matriz de rotación parametrizada en un vector de tres valores, t es un vector de traslación y K es la matriz de calibración de la cámara. Por lo tanto $\hat{p}(K, R_i, t_i, P_j)$ es el punto P_j reproyectado en la imagen i y p_{ij} es el punto j en plano imagen en la imagen i.

Capítulo 3

Desarrollo

3.1. Descripción de la infraestructura utilizada

Se construyó un entorno de pruebas con madera, de dimensiones 2 m $\times$ 1.2 m, el cual se puede armar y desarmar. Se eligió madera para que cuando se pusieran los marcadores estuvieran sobre una superficie plana y para los objetos se utilizarán cajas de cartón como se observa en la figura 3.1.

Figura 3.1: Entorno de pruebas.

El robot se compró y es de la empresa UCTRONICS el modelo K0073 [17] que se puede observar en la figura 3.2 y cuenta con lo siguiente:

- 2 servomotores que controlan la inclinación y la orientación de la cámara.
- 4 motoreductores para las llantas.
- 1 sensor ultrasónico.
- 1 cámara para Raspberry Pi.

- 1 módulo para seguidor de linea.

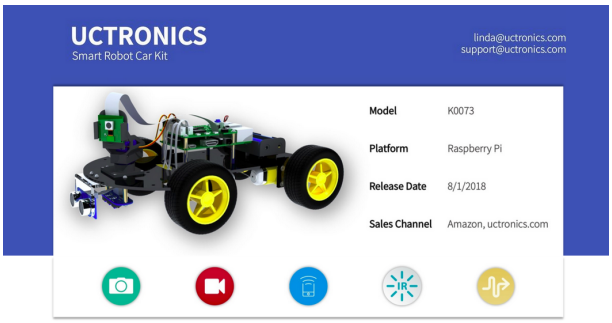


Figura 3.2: Modelo del robot.

También se utilizó una Raspberry Pi 3 B+, una computadora con GNU/Linux y un rutedador Wifi.

3.2. Calibración de la cámara del robot

Para la calibración de la cámara del robot se utilizaron las funciones de OpenCV [18] con un patrón de tablero de ajedrez de 8 columnas × 6 filas. Los lados de los cuadrados del tablero miden 25 mm. Se tomaron 37 fotografías moviendo el tablero dentro del área de visión de la cámara, como se observa en la figura 3.3.

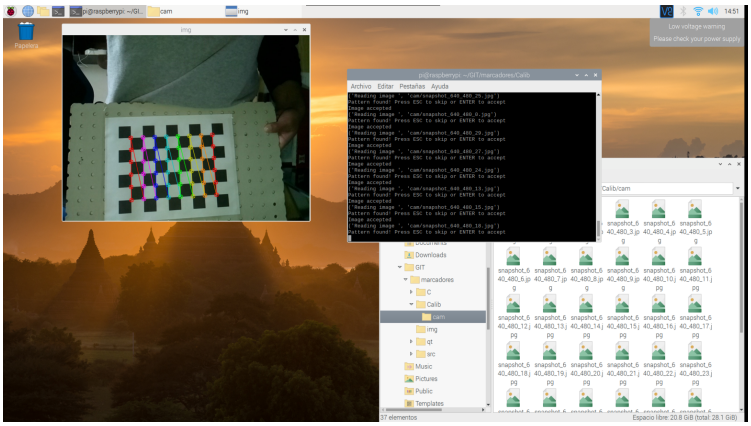


Figura 3.3: Detección del patrón de tablero de ajedrez de 8 × 6 para la calibración de la cámara del robot.

Una vez terminada la calibración se obtuvo la matriz K de la cámara (del modelo de la cámara oscura (2.7) o (2.8), en la página 5) y el vector de los coeficientes de distorsión d.

$$K = \begin{bmatrix} 595.03 & 0 & 311.20 \\ 0 & 599.44 & 237.44 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$d = [0.03866 \quad 0.2162 \quad -0.00378 \quad -0.00045 \quad -1.03014] \tag{3.2}$$

3.3. Instalación y modificación del software de desarrollo del robot

El robot incluye una tarjeta microSD con su software de desarrollo (SDK) preinstalado y este se puede controlar usando una aplicación para android [19] como se observa en la figura 3.4 y Scratch [20], el cuál es un lenguaje de programación que está diseñado especialmente para jóvenes de entre 8 y 16 años sin importar que no tengan conocimientos profundos de programación. También cuenta con la opción de compilar el SDK, descargando sus bibliotecas de su Github [21] y esta fue la opción que se utilizó.

Figura 3.4: Aplicación para android.

Se instaló Raspberry Pi OS con escritorio en la RaspberryPi 3 B+ y también se instaló OpenCV para el procesamiento de imágenes en la RaspberryPi 3 B+. Una vez instalado esto, se instaló el software de desarrollo para el robot obtenido del Github.

El software de desarrollo se divide en dos partes principales:

1. **Un servidor de imágenes:** que solo se encarga de mandar las imágenes que obtiene la cámara a la aplicación en android o a un servidor que se puede acceder con un explorador de internet.

2. **Un servidor de control:** que se encarga de el movimiento de los motoreductores, servomotores y la lectura de los sensores.

El software de desarrollo está escrito en C y también se puede utilizar con la aplicación o con scratch, ya que se conectan por medio de un servidor que se crea en la RaspberryPi.

Se realizaron pruebas utilizando la aplicación en android y se observó que enviaban valores al servidor de control, por lo que se creo una conexión usando la computadora que se conectará al servidor que crea el SDK del robot y la computadora ahora mandaría los comandos de movimiento en lugar de la aplicación por medio de un socket que se explica en la sección 3.5. en la página 23, en la tabla 3.1 se muestran los valores que de adoptaron y deben enviarse al servidor de control para realizar las distintas acciones.

Del SDK del robot solo se utilizó el servidor de control, el cuál ya incluía las siguientes funciones: giro izquierda, adelante, giro derecha, atrás y detenerse. Y se agregaron las siguientes:

- Giro y avanza izquierda/derecha, se explica en la sección 3.6, en la página 30.
- Pedir distancia, se explica en la sección 3.7, en la página 40.
- Orientación de la cámara, se explica en la sección 3.7, en la página 43.

Valor	Acción
0	Giro izquierda
1	Adelante
2	Giro derecha
3	Atrás
5	Detenerse
d	Pedir distancia del sensor ultrasónico
20	Giro y Avanza izquierda
21	Giro y Avanza derecha
22	Orientación de la cámara

Tabla 3.1: Tabla de valores para las acciones del servidor de control.

3.4. Detección y localización de marcadores

Aquí se utilizaron trabajos de tesis anteriores [1] [2] [10] [12], pero se adaptaron para resolver este problema, las bibliotecas están escritas en C/C++:

- Se redimensionaron los marcadores, en los marcadores el cuadrado negro mide 10 centímetros por lado.
- Se recálculo la normalización a media 0 y desviación estándar 1 de los valores en x y y de los seis puntos que componen el marcador que se muestran en la tabla 2.1. Los puntos normalizados se usan para el cálculo de la homografía entre una imagen y el modelo del marcador.
- Para visualizar en OpenGL los marcadores, se tenían que invertir las imágenes, en nuestra aplicación no se necesita esta inversión y se modificó el código.
- Se agregó la detección de los cuatro marcadores, ya que en los trabajos anteriores solo se detectaban uno o dos marcadores.
- Se agregó la capacidad de detectar múltiples marcadores en una misma imagen.

Unos marcadores de la figura 2.2 se colocaron sobre un muro y el carro se colocó a una distancia aproximada de 50 cm, como se observa en la figura 3.5. Posteriormente se agregaron dos cajas a los laterales con marcadores en uno de sus lados y se realizó el procesamiento de la imagen para detectar los marcadores, como se puede observar en la figura 3.6a. Es posible observar que las etiquetas de cada marcador conciden con las de la figura 2.2, algunos de los marcadores no fueron detectados ya que no cumplen con el criterio de tener sus cuatro esquinas dentro de la imagen. En la figura 3.6b los puntos marcados como "Puntos de homografía"son los puntos del modelo del marcador transformados por la homografía.

De cada marcador se obtienen 6 puntos que son: las 4 esquinas del marcador y 2 esquinas del triangulo al interior, estos puntos son guardados en una tupla con sus coordenadas $p_{ij} = (x_j, y_j)$, $j = \{0, 1, 2, 3, 4, 5\}$ junto con su etiqueta del marcador correspondiente $m = \{1, 2, 3, 4\}$ estas etiquetas se obtienen como se explica al final de la sección 2.2 e $i = \{1, \ldots, n\}$ donde n el vector de características para un marcador es entonces de tamaño 7 y es igual a:

$$caracteristicas_i = (m, p_{i0}, p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5})$$

En la tabla 3.2 se puede observar en la columna del lado izquierdo la etiqueta del marcador encontrado y en la columna del lado derecho la matriz de rotación R y el vector de traslación t de ese marcador. La imagen se analiza de izquierda a derecha y de arriba hacia abajo, el orden en la tabla muestra cómo se fueron encontrando los marcadores en la imagen.

En los primeros cuatro marcadores encontrados es posible observar que en los marcadores 3, 1 y 4 en su matriz de rotación, la diagonal es casi igual a 1, lo cuál indica que

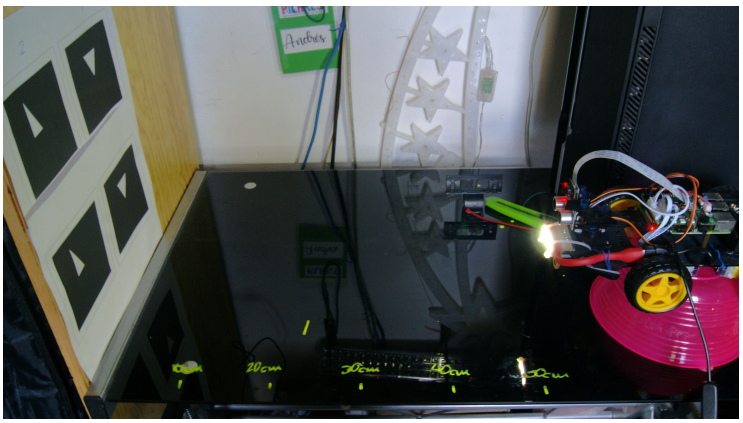
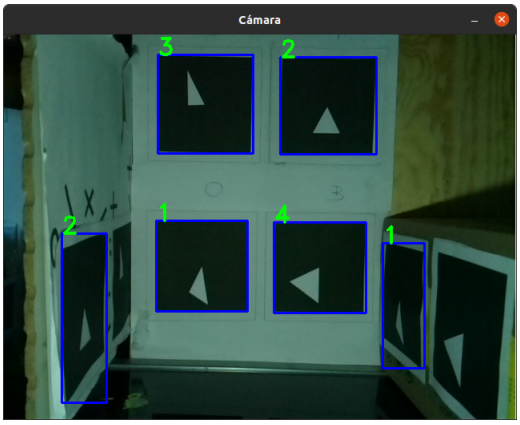
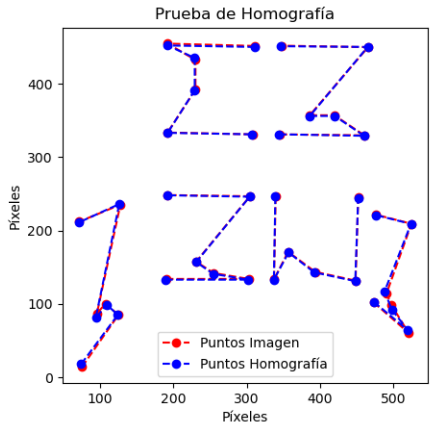


Figura 3.5: Prueba de localización y detección de marcadores.



(a) Resultado de la localización y detección de marcadores.



(b) Resultado de la homografía.

Figura 3.6: Resultados de la prueba.

casi no existe ninguna rotación respecto a la cámara del robot, pero para el marcador 2 es posible observar que en la diagonal hay unos signos negativos esto quiere decir que los ejes X y Y están en dirección contraria a como se muestran los ejes en la figura 3.7 para el marcador 2, en otras palabras el marcador está rotado 180° sobre el eje Z. Por otro lado es posible observar que los en cuatro marcadores son similares los valores del vector $t = [t_x, t_y, t_z]^T$ variando principalmente en t_x y t_y, indicando si está a la izquierda o a la derecha, arriba o abajo respecto al eje de visión de la cámara. Por último podemos ver que el valor de t_z de los cuatro marcadores se encuentra cerca de -50. En la figura 3.5 se puede observar que la cámara se encuentra aproximadamente a 50 cm de los marcadores, por lo tanto se puede ver que se tiene una buena precisión de la distancia de la cámara respecto al marcador en el mundo tridimensional.

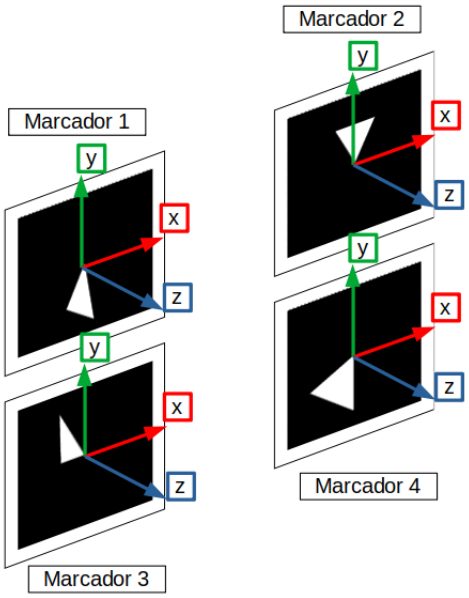


Figura 3.7: Sistema de ejes de coordenadas de los marcadores utilizados.

En los últimos dos marcadores 2 y 1, es posible observar que en su matriz de rotación de la diagonal (r_{11}, r_{22}, r_{33}) ya no es casi igual a 1, en el caso del marcador 2, se puede observar la combinación de rotaciones ya que está rotado 180° sobre el eje Z y también tiene una rotación sobre el eje Y y el marcador 1 se observa que sus valores son positivos por lo tanto solo tiene una rotación sobre el eje Y. Ahora al analizar el vector $t = [t_x, t_y, t_z]^T$ de cada uno de los marcadores que el marcador 2 está más cerca de la cámara $t_z = -30$ y que el marcador 1 se encuentra más lejos de la cámara $t_z = -50$. Se debe recordar que primero se aplica la rotación R y después la traslación por los valores del vector t.

3.5. Conexión entre el robot y la computadora

A la RaspberryPi montada en el robot se le asignó una IP estática, del SDK del robot únicamente se utilizó el servidor de control que se menciona en la sección 3.3, en la página 19. En la figura 3.10 se muestra un máquina de estados de este servidor. Se creó un servidor visión para detectar y localizar los marcadores como se explica en la sección 3.4, en la figura 3.11 se muestra un máquina de estados de este servidor y también se creó un cliente que se encarga de controlar el robot y crear el mapa que se explican en las secciones 3.7 y 3.8, en la figura 3.12 se muestra un máquina de estados del cliente. La conexión se realizó por medio de sockets. En la figura 3.9 se muestra un esquema general

Marcador	$[R\|t]$			
3	1.005212	0.028110	0.023292	−10.326097
	−0.013075	0.982973	−0.148987	8.312488
	−0.025658	0.147497	0.988464	−51.792332
2	−0.996733	−0.018968	−0.011949	13.068924
	0.016208	−0.998048	−0.098030	17.940838
	−0.010372	−0.098549	0.995095	−50.430107
1	0.995137	0.014657	0.073031	−10.6914161
	−0.012438	1.001919	−0.013849	−9.039403
	−0.073051	0.012840	0.997228	−52.524281
4	0.997008	0.016695	0.000319	2.327449
	−0.014406	1.000950	−0.059860	−9.193698
	−0.001183	0.060020	0.998196	−53.163261
2	0.031665	−0.123527	0.967216	−11.299984
	−0.113021	−1.005785	−0.103246	−0.019769
	0.923303	−0.341274	−0.045810	−36.529331
1	0.076883	0.020239	−0.994671	14.253532
	−0.063231	1.033231	0.017482	−11.677636
	0.961112	0.025626	0.080718	−52.259094

Tabla 3.2: Matrices de rotación R y vectores de traslación t obtenidos de la figura 3.6a.

de la conexión entre el los servidores y el cliente.

Figura 3.8: El robot en el entorno de pruebas como servidor de control y visión y la laptop utilizada como cliente.

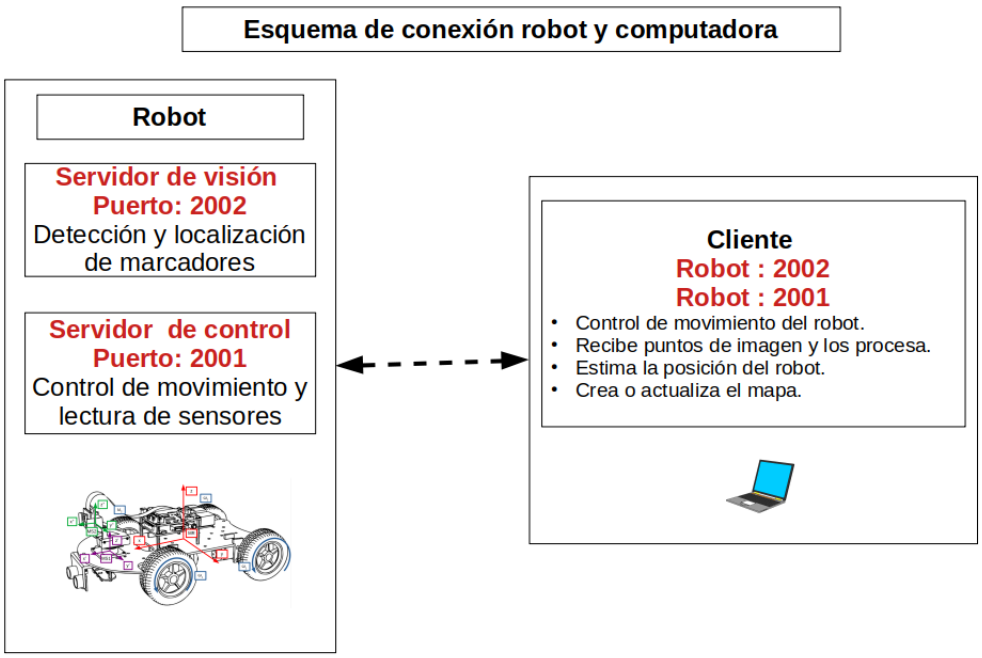


Figura 3.9: Esquema de conexión robot y el cliente. Las comunicaciones entre el robot y la máquina cliente se realizan a través de sockets. Los servidores en el robot y el cliente se diseñaron usando máquinas de estados.

3.5.1. Servidor de control del robot

En el servidor de control (SC), el robot recibe instrucciones del cliente para moverse dentro del entorno y enviar información del sensor ultrasónico.

Dentro del servidor de control del robot, que se muestra en la figura 3.10, se realiza lo siguiente:

- Estado SC_{in}: se verifica que todos los sensores y actuadores del robot estén funcionando correctamente y abre el puerto 2001 e inicializa el servidor, si todo está bien avanza al estado de espera SC_a en caso contrario termina y va al estado SC_{fin}.

- Estado SC_a: el robot se mantiene esperando alguna instrucción enviada por la computadora de las mencionadas en la tabla 3.1, si el valor recibido es d avanza al estado SC_b, si el valor recibido es 1 o 3 avanza al estado SC_c, si el valor recibido es 0 o 2 avanza al estado SC_d, si el valor es 5 avanza al estado SC_e, si el valor es 20 o 21 avanza al estado SC_f, si el valor es 22 avanza al estado SC_g y si la computadora da por terminada las tareas avanza al estado SC_{fin}.

- Estado SC_b: se toma la lectura del sensor ultrasónico en ese momento y se envía a

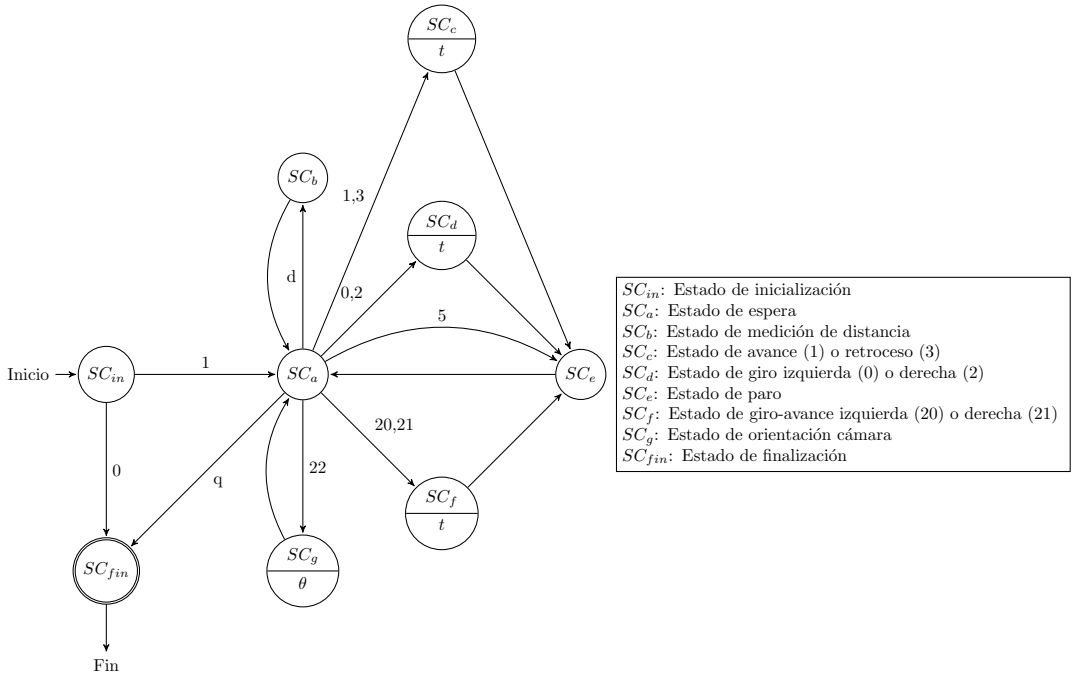


Figura 3.10: Máquina de estados del servidor de control del robot.

la computadora y regresa al estado SC_a.

- Estado SC_c: se realiza el movimiento de avance si el valor es 1 o retroceso si el valor es 3 por un tiempo t que determina la computadora y al terminar el periodo de tiempo avanza al estado SC_e.
- Estado SC_d: se realiza el movimiento de giro que se explica en la sección 3.6, en la página 30, si el valor es 0 realiza un giro a la izquierda o si el valor es 0 realiza un giro a la derecha por un tiempo t que determina la computadora y al terminar el periodo de tiempo avanza al estado SC_e.
- Estado SC_e: se detienen los cuatro motoreductores y regresa al estado SC_a.
- Estado SC_f: se realiza el movimiento de avance-giro que se explica en la sección 3.6, en la página 30, si el valor es 20 realiza un giro y avance a la izquierda o si el valor es 21 realiza un giro y avance a la derecha por un tiempo t que determina la computadora y al terminar el periodo de tiempo avanza al estado SC_e.
- Estado SC_g: se controla la orientación de la cámara que se explica en la sección 3.7 en la página 43 y el ángulo θ de orientación depende de un valor enviado por la computadora y al terminar regresa al estado SC_a.
- Estado SC_{fin}: se finaliza el servidor y se termina el programa.

3.5.2. Servidor de visión del robot

En el servidor de visión (SV), el robot se encarga de detectar y localizar los marcadores en el entorno como se explicó en la sección 3.4 y obtiene las coordenadas (u, v) en píxeles de los puntos p de los marcadores, su número de etiqueta o ID que puede ser $ID = \{1, 2, 3, 4, 5\}$, el $ID = 5$ se utiliza cuando el marcador detectado no pertenece a ninguno de los otros marcadores y también envía el número de foto al cliente.

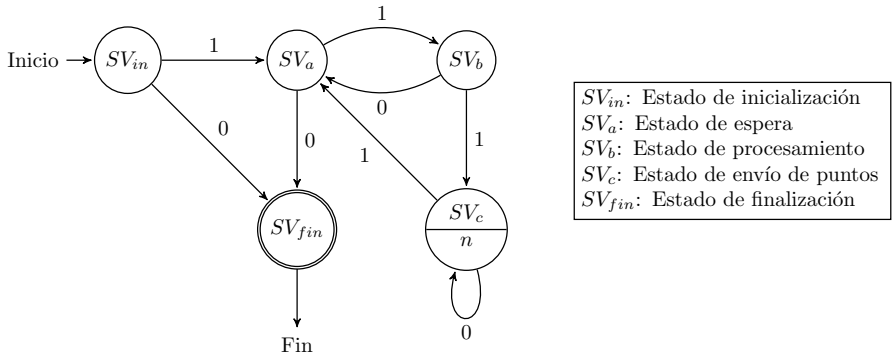


Figura 3.11: Máquina de estados del servidor de visión del robot.

Dentro del servidor de visión del robot se realiza lo siguiente:

- Estado SV_{in}: verifica que la cámara se encuentre funcionando y la inicializa; si está todo correcto avanza al estado SV_a y abre el puerto 2002 e inicializa el servidor, en caso contrario avanza al estado SV_{fin}.
- Estado SV_a: el robot se mantiene en espera de instrucciones de la computadora, si la computadora le pide una foto avanza al estado SV_b y si la computadora termina las tareas avanza al estado SV_d.
- Estado SV_b: en este estado el robot procesa la foto tomada en ese momento y en caso de que existan marcadores el robot envía por medio del socket de servidor:2002 a la computadora que existen marcadores y avanza al estado SV_c y si no hay marcadores en la imagen el robot envía por medio del socket de servidor:2002 a la computadora que no hay marcadores y regresa al estado SV_a.
- Estado SV_c: en este estado el robot envía las tuplas de características que se describió en la sección 3.4 a la computadora por medio del socket de servidor:2002, a la tupla se le agrega el número de foto y una vez que termina regresa al estado SV_a.
- Estado SV_{fin}: es el estado final y se llega a él cuando la computadora da por terminadas las tareas del robot o cuando la cámara no funciona.

3.5.3. Cliente

El cliente es una laptop con GNU/Linux y se encarga de recibir las coordenadas de los puntos encontrados por el robot y procesarlos para poder construir el mapa, crea o actualiza el mapa del entorno y también controla los movimientos del robot para poder navegar en un mapa ya construido.

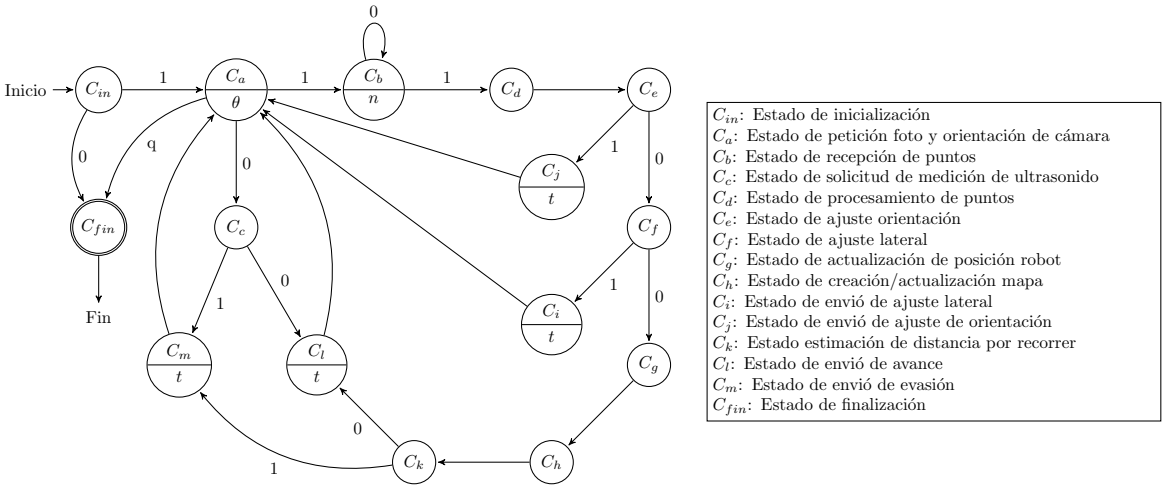


Figura 3.12: Máquina de estados del cliente.

Dentro del cliente se realiza lo siguiente:

- Estado C_{in}: se verifica que se pueda realizar la conexión al servidor de visión y al servidor de control del robot, si es correcta avanza al estado C_a y en caso contrario avanza al estado C_{fin}.

- Estado C_a: este estado se comunica con el estado SV_a y SV_b del servidor de visión del robot y le indica al estado SV_a que tome una foto en ese momento y el estado SV_b le indica al estado C_a si hay marcadores en la foto o no, si hay marcadores en la foto avanza al estado C_b y si no avanza al estado C_c. Este estado también se comunica con el estado SC_a y le indica el ángulo θ al que debe orientarse la cámara. Por último en este estado se determina si el robot completó el reconocimiento o llegó al último punto de la ruta y avanza al estado C_{fin}.

- Estado C_b: este estado se comunica con el estado SV_c del servidor de visión del robot y recibe los puntos de los n marcadores detectados en la foto tomada por el robot, cuando termina de recibir los puntos avanza al estado C_d.

- Estado C_c: este estado se comunica con el estado SC_b y recibe la lectura del sensor ultrásonico en ese momento, si es menor a un umbral de distancia U_{dist} avanza al estado C_m y si no avanza al estado C_l.

- Estado C_d: en este estado se procesan los puntos recibidos y se obtiene su homografía y se almacenan en una lista de la estructura de `Caracteristicas` que se explicará más adelante al final de esta sección, también aquí se selecciona el marcador de referencia que se explica en la sección 3.7 en la página 36, una vez terminado se avanza al estado C_e.

- Estado C_e: en este estado se calcula si es necesario realizar un ajuste de orientación que se explica en la sección 3.7 en la página 41, si no es necesario realizar ningún ajuste se avanza al estado C_f y si es necesario realizar algún ajuste se avanza al estado C_j.

- Estado C_f: en este estado se calcula si es necesario realizar un ajuste lateral que se explica en la sección 3.7 en la página 42, si no es necesario realizar ningún ajuste se avanza al estado C_g y si es necesario realizar algún ajuste se avanza al estado C_i.

- Estado C_g: en este estado se realiza la estimación de la posición del robot en el entorno como se explica en la sección 3.7 en la página 37 y una vez actualizada se avanza al estado C_h.

- Estado C_h: en este estado se crea o se actualiza el mapa, esto se explica a detalle en la sección 3.8 en la página 44 una vez que termina avanza al estado C_k.

- Estado C_i: este estado se comunica con el estado SC_a para realizar el ajuste lateral y al terminar avanza al estado C_a.

- Estado C_j: este estado se comunica con el estado SC_a para realizar el ajuste de orientación y al terminar avanza al estado C_a.

- Estado C_k: en este estado se determina si el robot aún puede avanzar o no utilizando la distancia del robot hacia el marcador de referencia, si es menor a un umbral de distancia U_{dist} avanza al estado C_m y si no avanza al estado C_l.

- Estado C_l: este estado se comunica con el estado SC_a del servidor de control del robot para realizar el avance constante y variable que se explica en la sección 4.2.1 en la página 49 y una vez terminado avanza al estado C_a.

- Estado C_m: este estado se comunica con el estado SC_a del servidor de control del robot para realizar la maniobra de evasión que se explica en la sección 3.7.3 en la página 40 y una vez terminada avanza al estado C_a.

- Estado C_{fin}: en este estado se guarda la lista de la estructura de `Caracteristicas` en un archivo **lista.dat** y también se guarda el mapa en un archivo **Mapa.dat**. Por último envía mensajes a los estados SV_a y SC_a de los servidores de control y visión del robot, para finalizar los servidores.

```
1    struct Caracteristicas {
2      int num_foto;
3      int ID_marcador;
4      int num_marcador_foto;
5      float puntos[12];
6      int x_centroide;
7      float MatrixRt[16];
8    };
```

Figura 3.13: Elementos de la estructura Caracteristicas.

A continuación se describe la estructura Caracteristicas que se muestra en la figura 3.13 y en la figura 3.14 se muestra se muestra un ejemplo, hay tres marcadores en la imagen 1 y se puede observar que en el número de foto 1 en la terminal están tres marcadores con ID 3 y 1.

El elemento `num_foto` tiene el número de foto en el que fue tomada por el robot y permite agrupar los marcadores que fueron vistos en la misma imagen, el elemento `ID_marcador` indica que número de marcador es, el elemento `num_marcador_foto` lleva el conteo de cuantos marcadores con el mismo ID hay en la misma foto, el elemento `puntos[12]` contiene las coordenadas (u, v) en píxeles de las seis esquinas del marcador que recibe la computadora del robot, el elemento `x_centroide` la coordenada en x en píxeles tiene la ubicación del centroide del marcador en la imagen que permite saber si un marcador está a la izquierda, derecha o al centro de la imagen y el elemento `MatrixRt[16]` contiene la matriz de rotación R y el vector de traslación t del marcador calculados a partir de las coordenadas de las seis esquinas recibidas y el ID de ese marcador.

3.6. Caracterización del movimiento del robot

Se requiere poder controlar el movimiento del robot, pero este solo cuenta con cuatro motoreductores en donde están las llantas, que no cuentan con una forma de retroalimentación para saber su posición o velocidad. Dentro del software de desarrollo del robot hay una función que permite controlar controlar su movimiento por medio de intervalos de tiempo.

Por lo tanto se decidió caracterizar el movimiento del robot, utilizando distintos intervalos de tiempo para girar y para avanzar y ver si existía alguna relación entre los grados de giro y la distancia avanzada con el tiempo transcurrido.

Se decidió utilizar un marcador para saber de una mejor forma la rotación y el des-

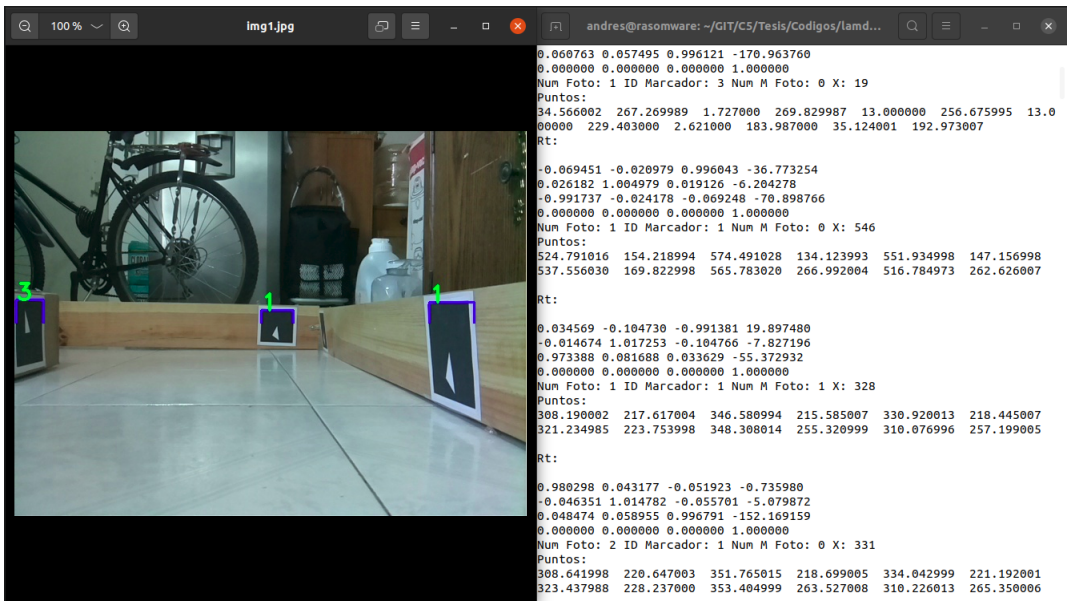


Figura 3.14: Captura de pantalla de la lista de la estructura Caracteristicas.

plazamiento. Se colocó el marcador 1 en la parte trasera del robot de tal forma que no interfiriera con la rotación de las llantas traseras. Posteriormente se colocó una cámara usb encima del carro, como se muestra en la figura 3.15. Esta cámara también se calibró utilizando el tablero de ajedrez como se explica en la sección 3.2 para poder obtener la homografía del marcador sobre el carro.

Figura 3.15: Experimento de caracterización del movimiento del robot.

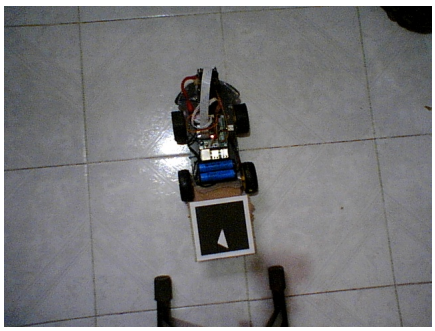
Se quería conocer el comportamiento de la función de Giro, Avance-Giro y Avance. En la figura 2.4 se observa cuál es la diferencia de estas funciones dentro del software de desarrollo del robot. En la función de Avance o Retroceso las 4 llantas giran en el mismo sentido, en la función de Giro las llantas que están de lado derecho del carro giran en un sentido y las llantas que están del lado izquierdo del robot giran en el sentido contrario y por último la función de Avance-Giro las llantas del lado derecho giran en la misma dirección mientras que las llantas del lado izquierdo no se mueven. En el caso de las funciones de Giro y Avance-Giro, para girar a la izquierda o derecha es la combinación de los movimientos explicados anteriormente.

Para la realización de las pruebas se utilizó el intervalo de tiempo de 0.1 a 1 segundos, con incrementos de 0.1 s. En el código fuente estos intervalos deben estar definidos en microsegundos (μs).

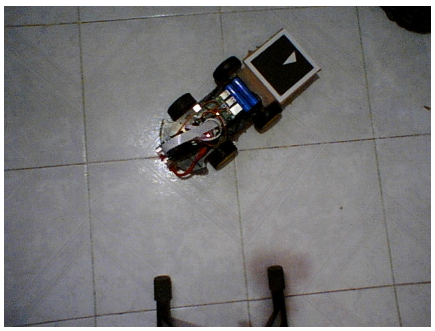
Para cada intervalo de tiempo se tomó una foto antes de comenzar el movimiento y una al terminar el movimiento, como se muestra en la figura 3.16 y este procedimiento se repitió diez veces para cada intervalo de tiempo, en cada una de las funciones de Giro y Avance-Giro. Una vez obtenidas las fotos, se obtenía la posición del marcador en la foto inicial y después la posición del marcador en la foto final, posteriormente se calculó la posición relativa de la posición final del marcador respecto a su posición inicial. Con esto fue posible estimar la orientación y el desplazamiento durante la rotación del robot.

En la tabla 3.3 se muestran los promedios y la desviación estándar de la orientación y su desplazamiento en Y de la función Giro y en la tabla 3.4 se muestran los promedios y la desviación estándar de la orientación y su desplazamiento en Y de la función Avance-Giro. En la figura 3.17 se muestra una comparación de los resultados de ambas funciones y se toman como referencia 2 resultados similares de orientación para la función de Giro 90.19 ° y para la función Avance-Giro 88.78 °, para el caso de la función de Giro es posible observar que este valor se alcanza a los 0.6 s y tiene una desviación estándar de 7.29 ° y que su desplazamiento es de 19.36 cm aproximadamente y para el caso de la función de Avance-Giro es posible observar que este valor se alcanza a los 0.8 s y tiene una desviación estándar de 3.32 ° y que su desplazamiento es de 28.39 cm aproximadamente. Por lo tanto es posible concluir que las principales diferencias es que la función de Giro hace incrementos mayores de orientación cuando rota que la función de Avance-Giro, pero la función de Avance-Giro hace incrementos mayores en el desplazamiento en Y mientras gira y también la desviación estándar de la función de Avance-Giro es menor que la de la función de Giro.

Para la función de Avance se medía la distancia recorrida utilizando un flexómetro con diez mediciones por cada intervalo de tiempo, en la tabla 3.5 se muestran los promedios y la desviación estándar de los resultados obtenidos. Es posible observar en la figura 3.18 que el comportamiento de la función de avance es casi lineal.



(a) Fotografía posición inicial.



(b) Fotografía posición final.

Figura 3.16: Fotografías de giro con 0.8 s.

Giro				
Tiempo (s)	Grados (°)	σ (°)	Desplazamiento en Y (cm)	σ (cm)
0.1	8.41	2.73	-0.4	0.17
0.2	20.72	1.32	-0.19	0.25
0.3	37.06	2.81	1.86	0.60
0.4	51.11	2.86	5.07	0.90
0.5	70.00	5.32	11.15	2.06
0.6	90.19	7.29	19.36	2.83
0.7	105.96	8.38	26.95	4.06
0.8	110.65	6.15	29.56	3.45
0.9	129.88	3.91	38.51	1.91
1	148.88	7.95	45.85	3.10

Tabla 3.3: Caracterización de la función Giro.

Estos movimientos están muy relacionados con el porcentaje de batería disponible. Durante la realización de las pruebas reconocimiento y mapeo fue posible observar que cuando las baterías estaban completamente cargadas el valor orientación que corresponde a los 0.5 s ahora le correspondía el valor de los 0.6 s y cuando las baterías estaban apunto de agotarse el valor orientación que corresponde a los 0.5 s ahora le correspondía el valor de los 0.4 s.

Por lo tanto solo se debe ajustar valor de tiempo en el código al realizar la prueba. Las tablas mostradas esta sección permiten tener una referencia de como ajustar el valor. Este ajuste se podría hacer automático si se pudiera conocer el porcentaje de batería disponible.

Avance-Giro				
Tiempo (s)	Grados (°)	σ (°)	Desplazamiento en Y (cm)	σ (cm)
0.1	4.59	0.95	0.59	0.16
0.2	13.15	0.90	1.86	0.18
0.3	23.07	1.38	4.01	0.35
0.4	36.19	3.57	7.43	1.21
0.5	46.19	2.02	10.89	0.75
0.6	62.17	3.22	16.70	1.57
0.7	75.73	6.18	22.75	2.76
0.8	88.78	3.32	28.39	1.38
0.9	97.74	1.91	32.08	0.81
1	107.75	4.86	36.64	2.17

Tabla 3.4: Caracterización de la función Avance-Giro.

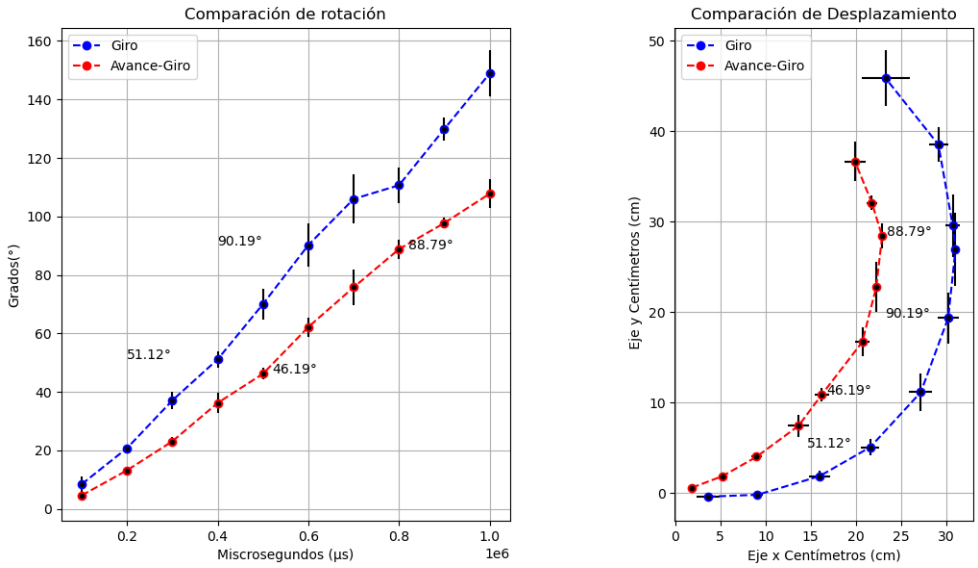


Figura 3.17: Resultado de la caracterización de la función Giro y Avance-Giro.

Una forma de tener mayor control sobre el movimiento del robot sin usar intervalos de tiempo, sería utilizar el sensor encoder que permite saber el número de vueltas de una llanta del robot y en función del radio de la misma estimar la distancia recorrida y tener control de la velocidad con la que giran las ruedas para hacer un movimiento más controlado y rápido. Estas opciones no se utilizaron debido a que el software de desarrollo del robot no explica muy bien como se realiza el control de la etapa de potencia de las

Avance		
Tiempo (s)	Desplazamiento en Y (cm)	σ (cm)
0.1	3.25	1.0
0.2	7.75	0.48
0.3	13.1	0.73
0.4	18.8	1.58
0.5	23.4	0.84
0.6	29.1	0.56
0.7	34.95	0.86
0.8	40.0	1.71
0.9	44.95	1.51
1	49.9	1.19

Tabla 3.5: Caracterización de la función Avance.

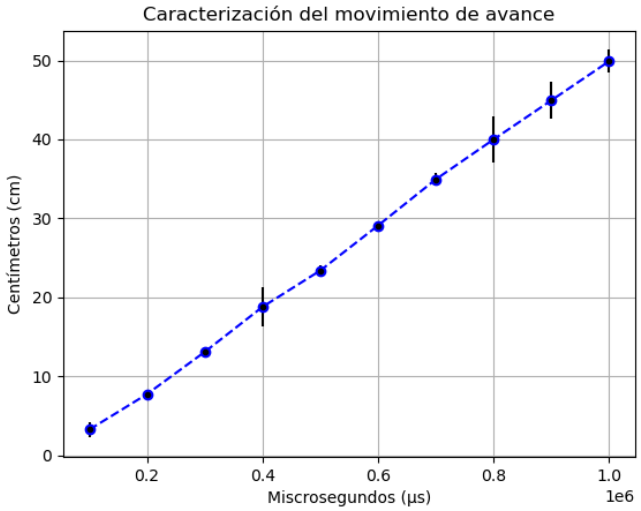


Figura 3.18: Resultado de la caracterización de la función Avance.

ruedas y en la placa de potencia del robot no se contempla el uso de enconders, por lo tanto no era posible usarlo sin usar otra placa de control y sin tener que crear las funciones dentro del software de desarrollo, lo cuál no era parte de los objetivos de este trabajo.

3.7. Sistema de control del robot

Dentro del entorno se tienen marcadores en la paredes del entorno como se ve en la figura 3.19 con $ID = 1$ que será el conjunto P y también se tiene un conjunto de obstáculos

dentro del entorno con $ID = 2$ del marcador es O_2, con $ID = 3$ del marcador es O_3 y con $ID = 4$ del marcador es O_4.

Figura 3.19: Entorno de pruebas utilizado con marcadores de tipo de orden.

3.7.1. Elección de marcador de referencia

Para elegir el marcador de referencia el robot toma una foto y se analizan todos los marcadores detectados en ella, como se muestra en la figura 3.20(a) donde la posición y orientación del robot está representada con la letra C.

El marcador de referencia debe ser únicamente del conjunto P que son los marcadores que se encuentran en las paredes del entorno, por lo tanto los marcadores que no pertenecen a este conjunto se descartan. Se puede observar en la figura 3.20(a) que hay tres marcadores del conjunto P, pero cada uno tiene una posición y orientación relativa diferente a la cámara del robot. Por lo tanto para seleccionar el marcador de referencia se busca el marcador P que tiene la orientación más similar a la orientación de C y en caso de que existan más de un marcador P con una orientación similar a C, se seleccionará el marcador P con la menor distancia relativa a C .

En la figura 3.20(b) se muestra que el marcador P_0 es el marcador de referencia. Este proceso se repite antes de que el robot avance, excepto cuando el robot está en la etapa de evasión. El marcador de referencia permite estimar la posición del robot C que se explicará en la subsección 3.7.2, la orientación del robot que se explicará en la subsección 3.7.4, el ajuste lateral que se explicará en la subsección 3.7.5 y la función para comenzar la etapa de evasión que se explicará en la subsección 3.7.3.

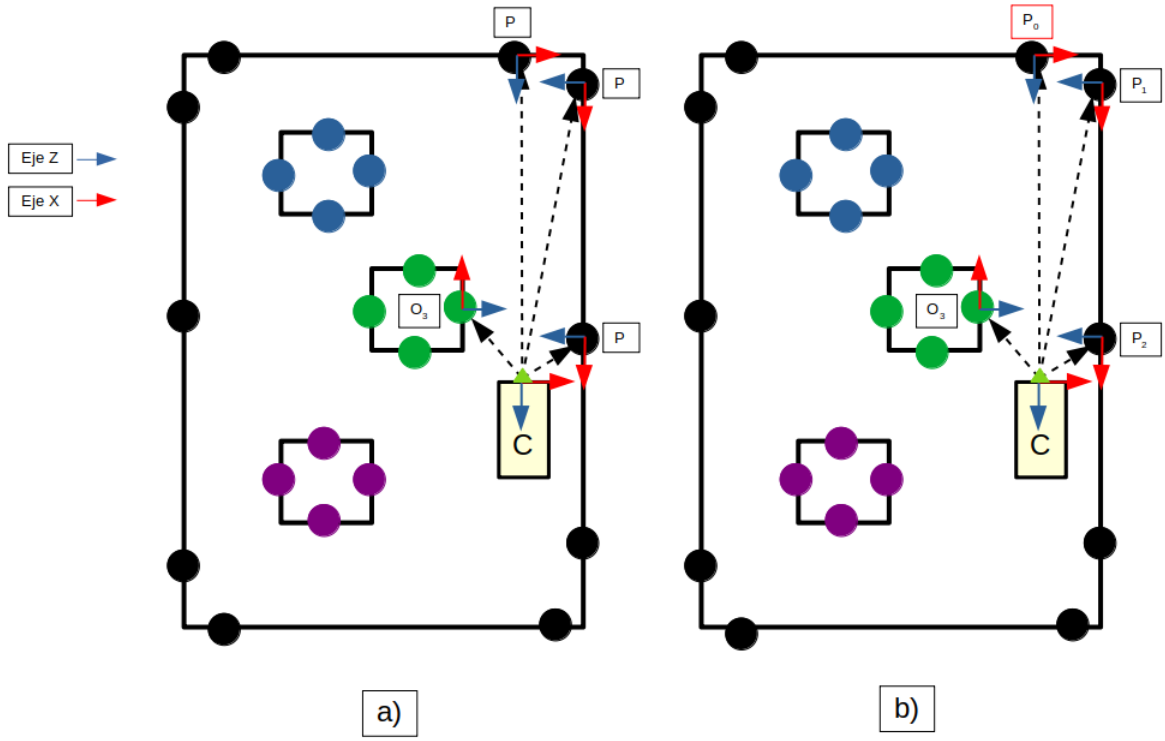


Figura 3.20: Representación de la foto para elegir el marcador de referencia: a) Foto sin marcador de referencia seleccionado y b) Foto con marcador de referencia seleccionado.

3.7.2. Estimación de posición

Una vez que se tiene seleccionado el marcador de referencia, se guarda la matriz de rotación R y el vector de traslación t del marcador de referencia P_0 a la cámara del robot C, que podemos definir de la siguiente forma:

$$M = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

Tomando en cuenta los resultados presentados en la sección 3.4 para estimar posición y rotación de los marcadores, se hizo un sistema de estimación visual utilizando los marcadores. Para realizar la estimación de la posición del robot existen dos casos como se puede ver en la figura 3.21, en el primer caso el robot ve el origen del mapa marcado en color azul y en el segundo caso el robot no ve el origen del mapa marcado en color rojo, el origen del mapa es el primer marcador de referencia que selecciona el robot.

Para el caso uno en donde el robot ve el origen del mapa P_0 como se muestra en la figura 3.21, la posición del robot se estima de la siguiente forma:

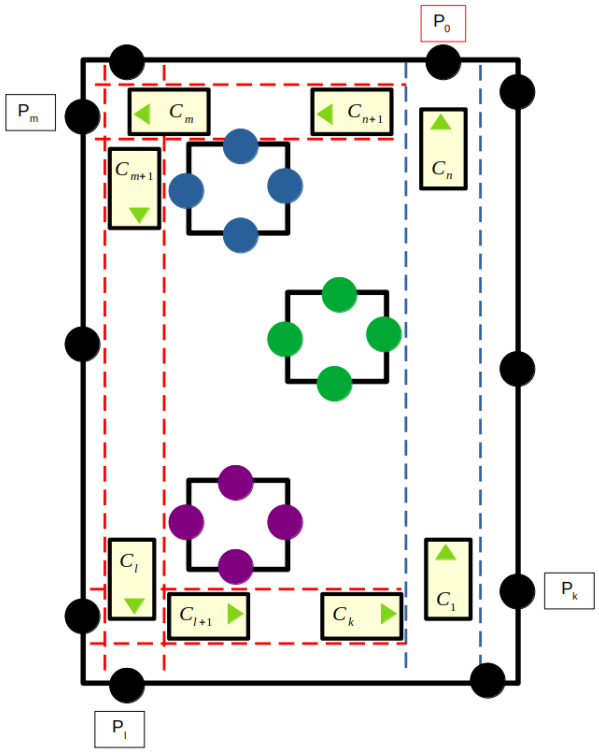


Figura 3.21: En azul se representa el camino en que el robot ve el origen del mapa y en rojo el camino donde no ve el origen del mapa.

$$C_i = (M_i^0)^{-1} \tag{3.4}$$

donde $i = \{1, \ldots, n\}$ y n es el número de veces que el robot ve al origen del mapa.

Para el caso en donde el robot no ve el origen del mapa, la ecuación 3.4 no se puede utilizar, para este caso se estima una posición a partir de la última vez que el robot ve al origen del mapa que es C_n.

En la figura 3.22 se muestra el robot en dos posiciones distintas C_{i+1} y C_{i+2}, en ambas posiciones el marcador de referencia es P_0, por lo tanto se tienen dos posiciones relativas distintas de P_j a la cámara del carro que se pueden definir usando la ecuación 3.3 de la siguiente forma M_{i+1}^j es la posición relativa de P_j a C_{i+1} y M_{i+2}^j es la posición relativa de P_j a C_{i+2}, donde C_{i+2} es la posición actual y C_{i+1} es la posición pasada, por lo tanto para estimar C_{i+2} es necesario calcular el desplazamiento D del robot.

Si se calcula de la inversa de M_{i+2}^j, la cuál definimos de la siguiente forma M_j^{i+2} que

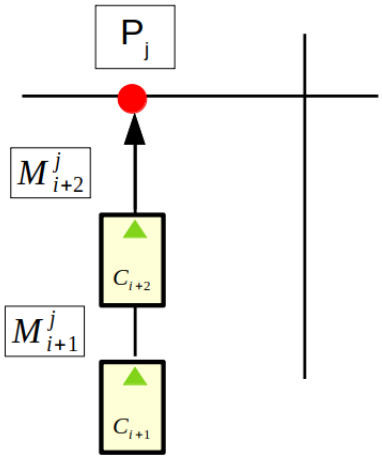


Figura 3.22: Robot en posición distinta respecto al marcador P_j.

representa la posición del robot C_{i+2} respecto a P_j.

$$M_j^{i+2} = (M_{i+2}^j)^{-1}$$

Al multiplicar M_{i+1}^j y M_j^{i+2}, se obtiene el desplazamiento D de C_{i+2} respecto a C_{i+1}.

$$D = M_j^{i+2}\, M_{i+1}^j$$

Una vez que se obtiene el desplazamiento D se aplica el desplazamiento a C_{i+1} para estimar posición C_{i+2}.

$$C_{i+2} = D\, C_{i+1}$$

Entonces podemos generalizar la estimación del desplazamiento de la siguiente forma:

$$D = M_{act}^{-1}\, M_{ant} \tag{3.5}$$

$$C_{act} = D\, C_{ant} \tag{3.6}$$

donde M_{act} es la posición relativa actual del marcador al robot C_{act} y M_{ant} es la posición relativa anterior del marcador al robot C_{ant}.

Por lo tanto para estimar la posición del robot C_i cuando $i = \{1, \ldots, n\}$ se utiliza la ecuación (3.4) y cuando $i = \{n+1, \ldots, k\}$ se utilizan las ecuaciones (3.5) y (3.6), en este último caso es posible observar la dependencia de la posición del robot C_n.

3.7.3. Evasión

La función de evasión permite que el robot rote alrededor el eje z del marco de referencia MR de la figura 2.3 para evitar chocar contra algún obstáculo o pared y también es un complemento de la función de estimación de la posición. En la figura 3.21 es posible observar los casos en los que es necesaria la función es decir en C_n, C_m, C_l y C_k. Como ya se explicó anteriormente, en estos casos es la última vez que se detecta el marcador, durante la realización de las pruebas se observó que la mínima distancia en la que se detecta el marcador son 35 centímetros entonces este será nuestro umbral δ_z para decidir si es necesario realizar la función de evasión o el robot puede continuar avanzando.

Para la función de evasión se decidió realizar un giro de 90° cada vez que la posición del robot t_z fuera menor a δ_z. Pero debido a que no se puede controlar de una forma precisa la rotación del robot como ya se explicó en la sección 3.6, se optó por utilizar la función de Avance-Giro que se muestra su caracterización en la tabla 3.4 y es posible observar que con un tiempo de 0.8 s se tienen una rotación de 88.78° con una desviación de 3.32°, pero se decidió utilizar dos veces la función de Avance-Giro de 0.5 s ya que tiene una rotación de 46.19° con una desviación de 2.02°, principalmente porque la desviación es menor y la suma es casi 90°.

En cada una de las posiciones C_n, C_m, C_l y C_k el robot debe de estimar su nueva posición C_{n+1}, C_{m+1}, C_{l+1} y C_{k+1}, esto debido a que como ya se explicó deja ver al origen, entonces se toma la ultima posición en la que el robot vió al marcador de referencia y se mutiplica por una matriz de Avance-Giro que llamaremos G que le permite estimar su nueva posición:

$$G = \begin{bmatrix} 0.0 & 0.0 & -0.99 & -7.071 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.99 & 0.0 & 0.0 & 17.071 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz G se obtiene a partir de los valores de rotación y traslación de la tabla 3.4 para 0.5 s. Entonces para C_i cuando $i = \{n, m, l, k\}$:

$$E = G\, C_i^{-1} \tag{3.7}$$

$$C_{i+1} = E^{-1} \tag{3.8}$$

Estas ecuaciones permiten estimar la nueva posición del robot cuando hace la evasión, pero debido a que es un control por tiempo y la falta de precisión del control de los giros se va insertando una incertidumbre en la posición real del carro y va creciendo entre más veces se utilice la función de evasión, esto hasta que vuelva a ver el origen del mapa. En caso de que el robot no de bien la vuelta de 90° al terminar la función de evasión, el

ajuste de orientación terminará de corregir el giro en caso de ser necesario como se ve en la figura 3.12.

3.7.4. Ajuste de orientación del robot

Al igual que en la estimación de posición, se creó un sistema de orientación visual utilizando el marcador de referencia. El ajuste de orientación permite que el robot rote sobre el eje z del marco de referencia MR que se muestra en la figura 2.3, esto para controlar que avance lo más derecho posible. El robot utiliza la orientación θ en el eje y como se muestra en la figura 3.7 del marcador de referencia respecto a la cámara en ese momento como entrada de control.

El marcador de referencia M_{ref} tiene una matriz de rotación R:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

Una matriz de rotación R puede parametrizarse de la siguiente forma:

$$R = R_{z2}(\phi)R_y(\theta)R_{z1}(\psi) \tag{3.9}$$

donde cada matriz de rotación se define como:

$$R_{z2}(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$R_{z1}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(\phi,\theta,\psi) = \begin{bmatrix} c(\phi)c(\theta)c(\psi) - s(\phi)s(\psi) & -c(\phi)c(\theta)s(\psi) - s(\phi)c(\psi) & c(\phi)s(\theta) \\ c(\psi)c(\theta)s(\phi) + c(\phi)s(\psi) & -s(\phi)c(\theta)s(\psi) + c(\psi)c(\phi) & s(\phi)s(\theta) \\ -c(\psi)s(\theta) & s(\psi)s(\theta) & c(\theta) \end{bmatrix} \tag{3.10}$$

Por lo tanto el valor en grados de θ que es la rotación en el eje y se puede obtener de la siguiente forma:

$$\theta = \frac{\arccos(r_{33}) \times 180}{\pi} \tag{3.11}$$

En la sección 3.6 se explican los dos tipos de movimientos para realizar vueltas y en la tabla 3.3 se muestra el valor mínimo con el que puede rotar el robot en 0.1 s. Para la función de ajuste de orientación se utilizó la función de Giro, ya que solo se quiere que el robot se oriente y no es necesario ningún desplazamiento.

Tomando en cuenta el valor en 0.1 s de la tabla 3.3 se creó el umbral δ para la ecuación de control de orientación con la entrada de θ de la ecuación (3.11), la cuál es una función de apagado y encendido de la siguiente forma:

$$u_\theta = \begin{cases} \theta < -\delta_\theta & u_\theta = 1 \text{ Giro derecha 0.1 s} \\ -\delta_\theta \geq \theta \leq \delta_\theta & u_\theta = 0 \text{ Sin ajuste} \\ \theta > \delta_\theta & u_\theta = -1 \text{ Giro izquierda 0.1 s} \end{cases} \tag{3.12}$$

El ajuste de orientación se realiza antes de avanzar, el umbral de orientación que se utilizó es $\delta_\theta = 6°$, lo que da un umbral completo de $12°$ y dado que el valor mínimo de la tabla 3.3 es $8.41°$ está dentro y permite que no existan oscilaciones.

3.7.5. Ajuste lateral del robot

Para el ajuste lateral, también se utilizó un sistema visual basado en el marcador de referencia, pero a diferencia de la estimación de la posición y el ajuste de orientación, en el ajuste lateral se utilizan los marcadores que no fueron seleccionados como el marcador de referencia, si observamos la figura 3.20 se utilizarían los marcadores O_3, P_1 y P_2.

Por lo tanto del conjunto de n marcadores encontrados en una foto:

$$S = \{M_{ref},\ M_0, \ldots,\ M_{n-1}\},$$

se tiene un subconjunto A en donde no existe el marcador de referencia M_{ref}, entonces con los marcadores del subconjunto A se realiza el ajuste lateral de la siguiente forma, de cada marcador del subconjunto A se revisa el vector de traslación $t = [t_x, t_y, t_z]^T$ y se separan en marcadores que se encuentran a la izquierda y a la derecha de la cámara, en la figura 3.20 se observa que los marcadores a la derecha de la cámara tienen un valor positivo en t_x y los marcadores que se encuentran a la izquierda tienen un valor negativo en t_x.

Una vez separados en el subconjunto de los marcadores de la izquierda A_{izq} y el subconjunto de los marcadores de la derecha A_{der}, en cada uno de estos subconjuntos se busca el marcador más cercano a la cámara, para esto se calcula la distancia al marcador d_m siguiente:

$$d_m = \sqrt{t_x^2 + t_z^2} \tag{3.13}$$

Ya que se tiene el marcador izquierdo más cercano a la cámara M_{imin} y el marcador derecho más cercano a la cámara M_{dmin} se revisa en cada uno de estos el valor de t_x, que es la distancia lateral del carro hacia algún objeto o pared, sea t_x^{izq} la distancia lateral del marcador a la izquierda más cercano a la cámara del robot y sea t_x^{der} la distancia lateral del marcador a la derecha más cercano a la cámara del robot, entonces la ecuación de control queda definida de la siguiente forma:

$$u_x = \begin{cases} t_x^{izq} < -\delta_x & u_x = 1 \text{ Ajuste lateral derecha} \\ (t_x^{izq} > -\delta_x) \wedge (t_x^{der} > \delta_x) & u_x = 0 \text{ Sin ajuste} \\ t_x^{der} < \delta_x & u_x = -1 \text{ Ajuste lateral izquierda} \end{cases} \tag{3.14}$$

El valor de δ_x es 10, es decir que el robot debe de mantener una distancia mínima hacia los objetos o las paredes de 10 centímetros de distancia, esto para que cuando se realicen las vueltas el robot pueda girar libremente sin chocar. Para evitar que el robot oscile usando la ecuación de control lateral (3.14), entre las paredes y los objetos debe existir una distancia mínima de 30 centímetros, esto quiere decir que existe un espacio de 10 centímetros en los que el robot va cumplir las condiciones para no necesitar un ajuste evitando que oscile.

La maniobra de ajuste lateral se realiza de la siguiente forma utilizando valores de las tablas 3.3 y 3.5, siendo t_θ el tiempo de giro y t_a el ¸ de avance:

- Ajuste derecha: El robot realiza un giro derecha t_θ , avanza t_a y realiza un giro izquierda t_θ.
- Ajuste izquierda: El robot realiza un giro izquierda t_θ, avanza t_a y realiza un giro derecha t_θ.

3.7.6. Orientación de la cámara

La orientación de la cámara se realizando una rotación de la cámara rota sobre el eje z en el marco de referencia MS1 de la figura 2.3. El servomotor puede tener valores desde 400 hasta 2420 pulsos, la computadora debe enviar al robot el angulo deseado θ_{cam} que está dado de 0° a 90° y puede ser positivo o negativo, si el valor es negativo la cámara rota a la derecha y si es positivo rota a la izquierda, como se muestra a continuación:

$$u_{cam} = \begin{cases} \theta_{cam} > 0 & u_{cam} = 1410 + \frac{\theta_{cam} \times 1010}{90} \text{ La cámara gira a la izquierda} \\ \theta_{cam} == 0 & u_{cam} = 1410 \text{ La cámara ve al frente} \\ \theta_{cam} < 0 & u_{cam} = 1410 - \frac{\theta_{cam} \times 1010}{90} \text{ La cámara gira a la derecha} \end{cases} \tag{3.15}$$

Siendo 400 pulsos igual -90° y 2420 pulsos igual a 90°. La orientación de la cámara se utiliza en caso de que se necesite verificar algún marcador, la cámara rota hacia la dirección del marcador y se toma una foto.

3.8. Creación del mapa

Para la creación del mapa es necesario tener seleccionado un marcador de referencia, el primer marcador de referencia que se selecciona se convierte en el origen del mapa y a los otros marcadores que se detectan, se les debe calcular su posición relativa al origen.

El encargado de relacionar los marcadores detectados con el origen del mapa es el robot, este toma una foto y de los m marcadores detectados en la foto obtiene sus posiciones y orientaciones relativas a la cámara del robot que llamaremos M_i^m siendo i-esima posición del robot y m el número de marcadores en esa posición. Como habíamos definido antes tenemos un conjunto de marcadores por cada foto $S_i = \{M_i^{ref},\ M_i^0, \ldots,\ M_i^{m-1}\}$ donde en la primera foto el marcador de referencia M_1^{ref} se convierte en el origen. En la sección 3.7, en la página 37, se explica como se estima la posición del robot, por lo tanto para obtener las posiciones globales de todos los marcadores del conjunto $Q = \{P, O_2, O_3, O_4\}$ identificados en el entorno por el robot C, donde C es la posición y orientación del robot relativa al origen del mapa y M_i^m la posición relativa al robot C de algún marcador del conjunto Q visto por la cámara del robot C en ese momento i, de donde $w = \{0, \ldots, |Q| - 1\}$, el número de marcadores del conjunto Q depende del número de marcadores que se coloquen dentro del entorno. Por lo tanto la ecuación para definir las posiciones globales de los marcadores es la siguiente:

$$MG_i^w = C_i\ M_i^m \tag{3.16}$$

Dado que el valor $w = 0$ es para el marcador que se seleccionó como el origen (los demás valores de w se asignan dependiendo el orden en el que se agreguen al árbol binario que se explicará más adelante en esta sección), también se sabe que C_i es la posición del robot relativa al origen en ese instante i y que M_1^{ref} es la posición relativa del marcador seleccionado como origen al robot C_1, entonces el valor de MG_i^w es:

$$MG_1^0 = C_1\ M_1^{ref}$$

como el robot ve al origen se utiliza la ecuación (3.4) por lo tanto:

$$MG_1^0 = (M_1^{ref})^{-1}\ M_1^{ref}$$

dando como resultado la identidad y así definiendo el origen del sistema de coordenadas globales:

$$MG_1^0 = I$$

Es posible observar que el marcador P_{j+1} puede ser detectado n veces, por la cámara del robot, como se observa en la figura 3.23, por lo tanto se tiene una lista de las posiciones relativas del marcador P_{j+1} a la posición del robot C, que podemos definir de la siguiente forma $LM_{j+1} = (M_{i+1}^{P_{j+1}}, \ldots, M_{i+3}^{P_{j+1}})$, por lo tanto también existe

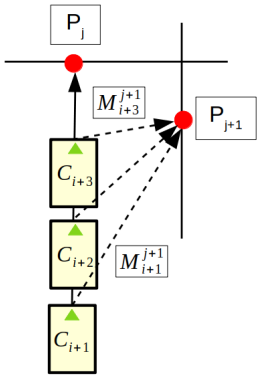


Figura 3.23: Lista de n imágenes en las que se detecta P_{j+1} por la cámara del robot C_i.

```
1     struct Nodo {
2       int ID;
3       float Orientacion;
4       float MatrixRT[16];
5       float sumX;
6       float sumZ;
7       int VecesVisto;
8       int vueltas;
9       list<float datos[60]>;
10      Nodo *izquierda;
11      Nodo *derecha;
12    };
```

Figura 3.24: Elementos de la estructura Nodo.

una lista de posiciones globales definidas por la ecuación (3.16) de la siguiente forma $LMG^w = (MG^w_{i+1}, \ldots, MG^w_{i+3})$. Entonces podemos decir que para cada marcador w en el entorno existe una lista posiciones y orientaciones relativas al robot LM_w y una lista de posiciones y orientaciones globales LMG^w, estas listas están relacionadas por la posición del robot C_i y también es importante resaltar que los elementos de la lista de posiciones globales LMG^w deben ser parecidos, pero debido al error que se genera al tomar las fotos existen pequeñas diferencias en la posición y orientación.

Para almacenar los datos del mapa se creó la estructura `Nodo` que se muestra en la figura 3.24. El elemento `ID` contiene la etiqueta del marcador detectado, el elemento `Orientacion` es la orientación relativa al origen estimada, el elemento `MatrixRT[16]` es la matriz de rotación R y el vector de traslación t de la posición global del marcador MG^w, el elemento `sumX` es la suma de los valores de t_x del vector de traslación t, el elemento `sumZ` es la suma de de los valores de t_z del vector de traslación t, el elemento `VecesVisto` es el número de

veces que ese marcador ha sido visto, el elemento `vueltas` es el número de vueltas que ha dado el robot cuando fue visto ese marcador, el número de vueltas incrementa cada vez que el robot utiliza la función de evasión, el elemento `list <float datos[60]>` este elemento contiene la matriz de posición relativa del marcador al robot M_i^w que consiste de 16 elementos, la matriz de la posición del robot C_i que consiste de 16 elementos, la matriz de la posición global del marcador visto MG_i^w y las coordenadas (u, v) de los seis puntos del marcador M_i^w que consiste de 12 elementos, `Nodo *izquierda` y `Nodo *derecha` son apuntadores a la estructura Nodo.

En la figura 3.25 se muestra un esquema en dos dimensiones utilizando únicamente los ejes x y z de lo marcadores, los puntos P_i con $i = \{0, \ldots, 9\}$ y O_2, O_3 y O_4 son únicamente el centro del marco de referencia global de los marcadores MG^w, siendo P_0 el origen el mapa global, por lo tanto los otros marcadores están relativos a P_0.

La estructura para almacenar el mapa es un árbol binario, la raíz del árbol va a hacer el primer marcador de referencia que se encuentre en este caso es P_0, a la izquierda del nodo raíz van a estar todos los marcadores con etiqueta o ID = 1 y a la derecha del nodo raíz todos los otros marcadores.

Para el caso de los marcadores con etiqueta o ID = 1 es decir los puntos P_i se verifica la orientación respecto al origen si tienen la misma orientación se agregan a la izquierda y si no a la derecha, por otro lado para los marcadores que están en los objetos O_2, O_3 y O_4 si tienen la misma etiqueta se agregan a la derecha y si no se agregan a la izquierda. El máximo número de nodos del árbol binario dependerá del máximo número de marcadores colocados dentro del entorno.

3.9. Evaluación del error de reproyección para el mapa

Para la evaluación del error de reproyección se utiliza la ecuación (2.13), siendo $m = 6$, ya que solo se tienen 6 puntos de las esquinas de los marcadores. El número de imágenes es n y como se explicó en la sección 3.8, un marcador puede ser visto n veces y hay q marcadores que pueden ser detectados y almacenados en el árbol binario, esto quiere decir que por cada nodo w en el árbol hay n imágenes.

Por lo tanto la ecuación (2.13) se puede reescribir de la siguiente forma tomando en cuenta la estructura que almacena el mapa:

$$\sum_{w=1}^{q}\sum_{i=1}^{n}\sum_{j=1}^{6} \|p_{wij} - \hat{p}(K, R_{wi}, t_{wi}, P_j)\|^2 \tag{3.17}$$

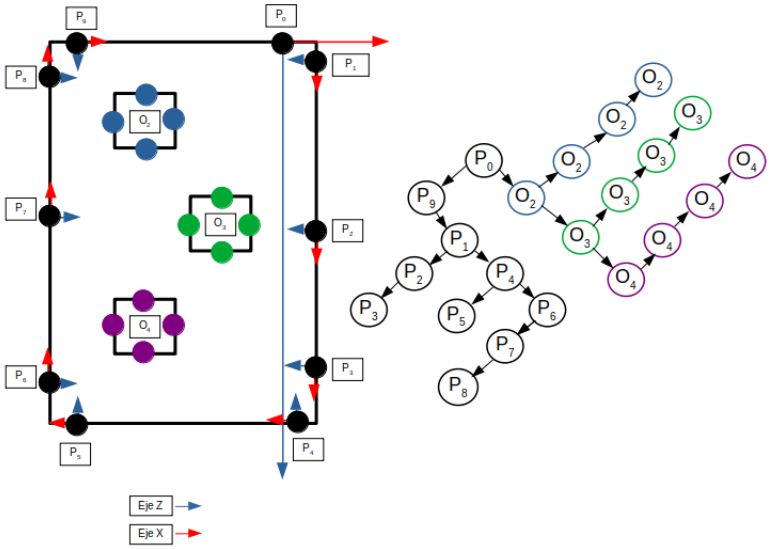


Figura 3.25: Esquema del árbol binario utilizado para almacenar el mapa.

Siendo q el número de nodos en el árbol binario, n el número de elementos en la lista `list <float datos[60]>` de la estructura Nodo, la matriz de calibración K ya se conoce en la ecuación (3.1), la matriz de rotación R y el vector de traslación t y los puntos p_{wij} están en el vector `datos[60]` y por último hay cuatro modelos P que son los marcadores de la figura 2.2, por lo tanto se utiliza el `ID` de la estructura Nodo para decidir con que modelo se realizará el cálculo de la reproyección del error.

Capítulo 4

Resultados

4.1. Descripción de la heurística para la exploración del entorno

Como se explicó el robot desconoce el entorno en que está, es por eso que se implementó la siguiente heurística para realizar la exploración del entorno:

- Avanzar mientras no exista un objeto enfrente y si existe un objeto enfrente a menos de 35 centímetros gira a la izquierda 90°.
- Para determinar si debe avanzar o evadir depende los marcadores detectados y también si la distancia del sensor ultrásonico es menor a 30 centímetros.
- El avance debe ser lo más recto posible, es por eso que se implementó el control de orientación del robot.
- Al terminar un giro de 90 grados, con ayuda del ajuste de orientación con la cámara el robot debe compensar el giro de 90 grados. El ajuste de orientación también permite compensar el giro de 90° en caso de que no se pueda dar de forma correcta.

Esta heurística está implementada en la máquina de estados del cliente en la figura 3.12.

4.2. Descripción de la heurística para el trazo inicial del mapa

Para realizar el mapa una de las principales tareas es saber que ese marcador ya ha sido detectado anteriormente, esta tarea puede ser resuelta si todos los marcadores en el entorno son distintos ya que solo es necesario identificar su etiqueta, pero la forma en como

se planteó el problema debido a que solo se cuenta con cuatro tipos de marcadores fue colocar un mismo tipo de marcador en las paredes y otro tipo en los objetos. Por lo tanto ya no bastaba con identificar la etiqueta del marcador, es por eso que se decidió hacer un promedio de la posición global del marcador y si, se detectaba un nuevo marcador y el error no era mayor a 10 centímetros podríamos decir que el marcador es el mismo y entonces solo se actualiza la matriz de posición global MG_i^w de la ecuación (3.16) que en la estructura `Nodo` de la figura 3.24, es el elemento `MatrixRT`.

$$MG_n^w = \begin{bmatrix} R_n & \bar{t} \\ 0 & 1 \end{bmatrix} \tag{4.1}$$

De donde n es el número de veces que fue visto el marcador w, por lo tanto R_n la última matriz de rotación calculada del marcador w y el vector de traslación $\bar{t} = [\overline{t_x} \quad t_y^n \quad \overline{t_z}]^T$, el cuál tiene la última posición calculada t_y^n en el eje y del marcador w y las posiciones promediadas del marcador w de en el eje x $\overline{t_x}$ y el eje z $\overline{t_z}$, que describen a continuación:

$$\overline{t_x} = \frac{\sum_{i=1}^{n} t_x^i}{n} \tag{4.2}$$

$$\overline{t_z} = \frac{\sum_{i=1}^{n} t_z^i}{n} \tag{4.3}$$

El valor de n esta guardo en la estructura de `Nodo` en el elemento `VecesVisto` y la sumatoria de $\overline{t_x}$ y $\overline{t_z}$ está guardada en los elementos `sumX` y `sumZ`. Se observó que el cálculo de los promedios permite ir reduciendo el error en la posición.

Cada vez que se detecta un nuevo marcador, se recorre la estructura del árbol como se describió en la sección 3.8 y si el error de posición en el eje x y z es menor a 10 (se estableció el número 10 ya que los marcadores miden 10 centímetros), entonces es el mismo marcador y se incrementa n y la sumatoria y también se almacenan en la lista `list <float datos[60]>` los puntos del marcador p, la posición en ese momento del robot C, la matriz rotación y traslación del marcador w relativa al robot M_i^w y la matriz de rotación y posición global del marcador w al origen MG_i^w.

Por lo tanto esta matriz global inicial MG_n^w es la que nos permite saber si es un marcador ya antes reconocido o es un nuevo marcador y así decidir si solo se tienen que actualizar datos de los elementos de la estructura `Nodo` o hay que agregar un nuevo nodo a la estructura del árbol binario que almacena el mapa.

4.2.1. Mapeo y localización con avance con velocidad variable

Para realizar esta prueba se obtuvo un mapa inicial y se seleccionaron unos puntos dentro del mapa para trazar un ruta. Primero es necesario estimar la distancia de la

posición actual del robot (px_a, pz_a) hacia las coordenadas del punto deseado (px_d, pz_d) como se muestra en la ecuación (4.4)

$$d_a = \sqrt{(px_d - px_a)^2 + (pz_d - pz_a)^2} \tag{4.4}$$

Ahora utilizando la tabla 3.5, los valores valor de tiempo de avance y la distancia recorrida en ese tiempo se almacenaron una matriz TA[10][2] en donde el primer columna es la distancia y la segunda columna el tiempo en el que se recorre esa distancia.

Algorithm 1 Avance variable

Require: d_a, TA
  if $d_a > 10$ **then**
    return $t_a = 0$
  else
    for $i = 0$ $i < 10$ $i++$ **do**
      if d_a mód TA[i][0] ≥ 1 **then**
        return t_a = TA[i][1]
      end if
    end for
  end if

Por lo tanto al final de la función de avance variable se devuelve un tiempo t_a que será 0 en caso de que la distancia por recorrer sea menor a 10 centímetros y en caso contrario se irá ajustando conforme avance el robot, esto le permite alcanzar más rápido el punto deseado de una forma más precisa. En la figura 4.1 se muestra la comparación del avance constante en azul y el avance variable en rojo, es posible observar que con el avance variable en menos pasos llega los puntos deseados marcados con una equis verde.

Es importante resaltar que las posiciones graficadas en la figura 4.1 se estiman utilizando lo explicado en la sección 3.7, es decir que la estimación de la posición del robot únicamente es visual, pero se demuestra que es posible que el robot se localice dentro de un mapa y que estime su posición dentro del mismo.

4.3. Comparación de los mapas inicial y optimizado

Para realizar la optimización es necesario utilizar el árbol binario que tiene la información del mapa y una vez que se guarda se realizan los siguientes pasos:

1. Se lee el archivo **Mapa.dat** en donde está guardado el mapa del reconocimiento.
2. Mientras se carga el mapa se cuenta cuantos nodos hay en el árbol binario y ese es el número de marcadores q y también se cuenta el número de imágenes por marcador en el árbol y ese es n de la ecuación (3.17).

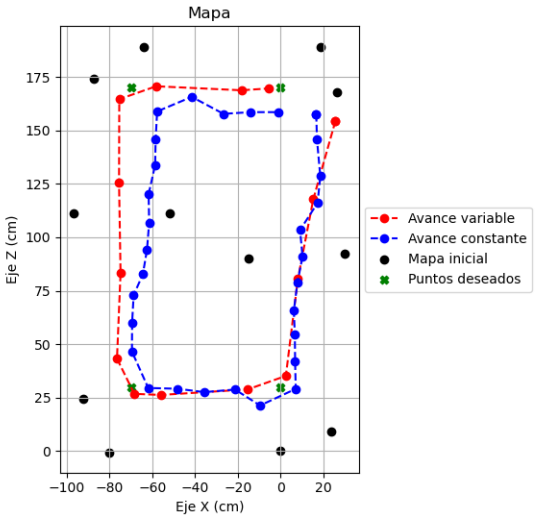


Figura 4.1: Comparación del avance constante y el avance variable.

3. Se reserva memoria para un vector de la función de error de la ecuación (3.17) el tamaño del vector es $e = 2 \times img \times 6$, el número 2 es porque se tienen dos coordenadas (u, v), img es el número de imágenes en el árbol binario y el número 6 es por el número de puntos que tiene cada marcador.

4. Se reserva memoria para un vector de variables $v = 6 \times img$, de donde el número 6 es porque la matriz M_i^w que está guardada con 16 elementos en `list <float datos[60]>` puede ser representada con 6 elementos, como se describió en la sección 3.7 la matriz de rotación R puede ser descrita con tres ángulos por lo tanto $M_i^w = (R_i(\psi, \theta, \phi), t_x^i, t_y^i, t_z^i)$ y img es el número de imágenes en el árbol binario.

5. Se calcula el error de reproyección con la ecuación (3.17) y posteriormente se calcula la norma.

6. Se introducen los valores del vector de variables, el vector de la función de error y la función de error de la ecuación (3.17), en la función `lmdif` de la bibliotca cminpack [22], que es el algoritmo de Levenberg-Marquardt y se optimizan los valores del vector de variables es decir de todas las posiciones relativas al robot $M_i^w = (R_i(\psi, \theta, \phi), t_x^i, t_y^i, t_z^i)$.

7. Se crea un nuevo árbol binario y se guardan los nuevos valores optimizados de M_i^w que llamaremos OM_i^w.

8. Para obtener la posición global optimizada OMG_i^w es necesario utilizar la ecuación (3.16), por lo tanto se deben optimizar las C_i almacenadas en el árbol binario.

9. Se crea el árbol binario final del mapa con las matrices globales OMG_i^w, relativas OM_i^w y las posiciones del robot OC_i optimizadas.

A continuación se presentan dos resultados del mapa inicial y el mapa optimizado en la figura 4.2 y en la tabla 4.1. La prueba se realizó en un entorno con dos objetos dentro, como se muestra en la figura 3.19. Es posible observar que que en la prueba 1 solo se detectaron 10 marcadores y se usaron 42 imágenes para hacer la optimización que requirió 2785 iteraciones y el error RMS inicial de 35.43 bajo a 0.87 lo que significa que es menor a un píxel, los valores de la máxima incertidumbre son menores a 10 centímetros. En la prueba 2 es posible ver que se detectaron 12 marcadores y se usaron 44 imágenes para hacer la optimización que requirió 4517 iteraciones y el error RMS inicial de 31.10 bajo a 0.85 lo que significa que es menor a un píxel y los valores de la máxima incertidumbre son menores a 10 centímetros. En los marcadores donde se tiene la máxima incertidumbre es debido a que el robot lo detectó únicamente dos veces y creemos que son necesarios más datos (más imágenes) para corregir el error.

Es importante resaltar que la detección de los marcadores se ve afectada por la cantidad de luz en el entorno debido a que la imagen se debe segmentar para la detección de estos, también por el ángulo de visión de la cámara, ya que pueden no estar en su campo de visión y la correcta estimación de la posición del robot dentro del entorno y aunque se busca reducir el error en la estimación de la posición del robot con los controles que se describieron en la sección 3.7, es importante recordar que son controles por tiempo, lo que significa que va existir una incertidumbre si está bien orientado o si dio el giro de 90 grados correctamente o al avanzar. A pesar de estos factores es posible ver que las dimensiones de los mapas obtenidos en la prueba 1 y 2, son similares a las dimensiones reales del entorno de pruebas que mide 200 cm $\times$ 120 cm, siendo las dimensiones aproximadas del mapa de la prueba 1 184 cm $\times$ 120 cm y para la prueba 2 196 cm $\times$ 120 cm.

Datos	**Prueba 1**	**Prueba 2**
Número de marcadores detectados	10	12
Número de imágenes	42	44
Número de iteraciones	2785	4517
Error RMS inicial	35.430919	31.100716
Error RMS final	0.877721	0.851547
Máxima incertidumbre en X	9.7271 cm	2.4207 cm
Máxima incertidumbre en Z	5.4692 cm	5.9410 cm

Tabla 4.1: Comparación de resultados de la prueba 1 y 2.

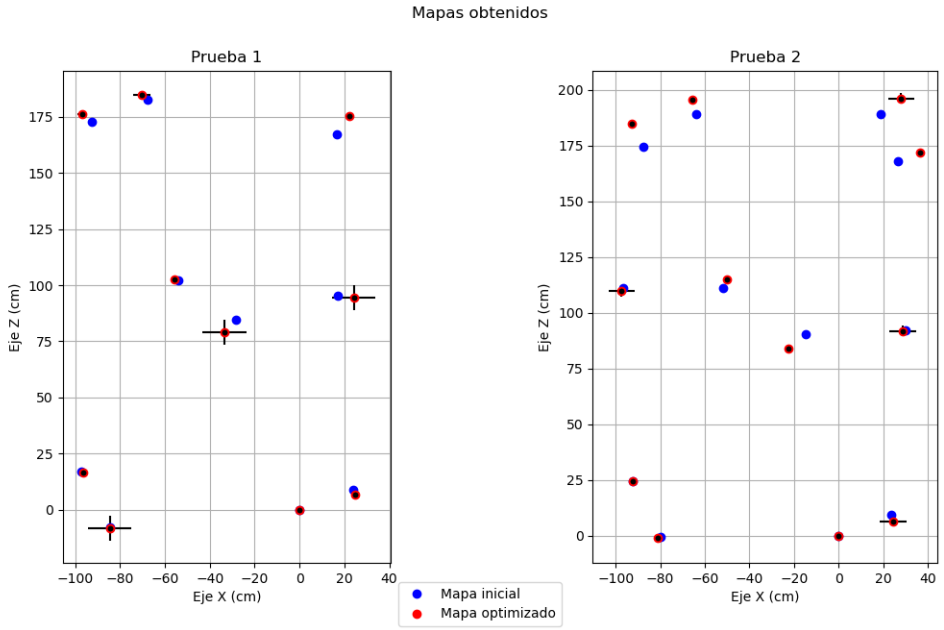


Figura 4.2: Comparación del mapa inicial y optimizado de la prueba 1 y 2.

Por último se muestran los mapas finales, los cuáles se muestran en la figura 4.3, estos se obtienen trazando líneas (que representan las paredes) de acuerdo a la orientación en el eje y del marcador, esta orientación está almacenada en la estructura `Nodo` en el elemento `Orientacion`, el tamaño de los objetos se conoce por lo que es posible trazarlos si se detecta un marcador que pertenezca al conjunto de obstáculos. Para determinar si el marcador es pared u obstáculo se utiliza el elemento `ID` de la estructura `Nodo`. Los límites del mapa se obtienen de acuerdo a la posición global mínima y máxima en los ejes x y z de todos los marcadores almacenados en el árbol. Actualmente el proceso de trazar los mapas finales se realiza fuera de linea, pero con la estructura utilizada debería ser sencillo trazar el mapa en tiempo real cada que se actualice.

Si se compara la figura 4.2 y 4.3 es posible observar que en los puntos optimizados donde existe mayor incertidumbre es donde hay más separación entre las líneas que representan las paredes del mapa final. Por otro lado es posible observar que en el mapa final de la prueba 1, los obstáculos se sobreponen, esto es debido a la incertidumbre que puede generar el robot al ir avanzando. En ambos resultados es probable que la posición de los obstáculos se pueda mejorar detectando los marcadores en al menos dos caras distintas, durante las pruebas se observó que por el tamaño del entorno de pruebas, la cámara del robot no alcanzaba a ver todas las caras de los objetos ya que no estaban en su campo de visión.

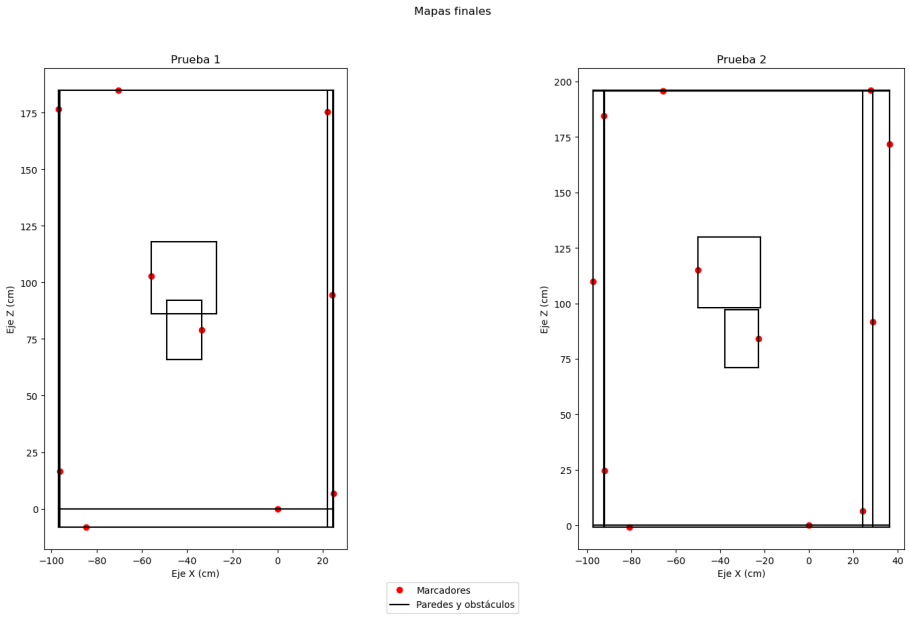


Figura 4.3: Mapas finales después de la optimización de la prueba 1 y 2.

Capítulo 5

Conclusiones

Se obtuvo un algoritmo para resolver el problema de la localización y mapeo simultáneo utilizando marcadores de tipo de orden. La estructura del árbol binario propuesta para almacenar el mapa podría utilizar cualquier tipo de marcadores, ya que se basa en la posición global y no en su etiqueta. Una forma de hacer más general la estructura propuesta sería acomodar los marcadores detectados de acuerdo a su distancia al origen, en lugar de como se tienen actualmente, que se acomodan de acuerdo si son objetos o paredes. El algoritmo de reconocimiento y detección de marcadores que se presentó en este trabajo, se puede implementar en cualquier robot con una cámara y también el sistema de visión para estimar la posición, solo deben adaptar los sistemas de control del robot. Por lo tanto el algoritmo localización y mapeo simultáneo basado en marcadores de tipo de orden propuesto, se podría utilizar con cualquier robot que tenga una cámara (haciendo las adaptaciones necesarias para el robot), en un entorno controlado y estático (que no cambien los obstáculos de posición) con marcadores de tipo de orden. Actualmente el robot cada vez que se ejecuta el algoritmo de mapeo y localización, siempre elige un origen, es decir que si se quiere utilizar un mapa ya generado se tendría que usar en la misma posición inicial para que el origen sea siempre el mismo que el del mapa generado.

Aunque el problema de realizar el mapa parece sencillo de resolver utilizando la ecuación (3.16), en la página 44, no es nada evidente como almacenar y utilizar los datos de los marcadores en un algoritmo para realizarlo de forma automática. Esto es debido a que no sabe que tamaño tendrá el mapa y la estructura para almacenar los datos del mapa va influir en la forma de realizar las búsquedas de los nodos dentro del mapa, tanto para agregar o actualizar nodos, sin que sea lenta. También es importante saber que datos son necesarios de guardar en cada nodo y cuáles no. Todo esto sin olvidar otras estructuras u otros algoritmos que se encuentran entre el proceso de detección de los marcadores y la creación o actualización del mapa, los cuales permiten procesar los datos de los marcadores de tipo de orden.

Para reducir la incertidumbre del mapa, se intentó que el robot avanzara los más derecho posible y corregir los giros de 90 grados al hacer la evasión. Durante las pruebas se observó que si el robot avanzaba sin tener control de orientación el mapa se desfasaba debido a este error, en los mapas que se muestran en la figura 4.2 es posible observar que en los puntos más alejados del origen es donde se presenta este error y esto debido a la incertidumbre que va incrementando cuando el robot avanza y gira al no ver al origen. Es probable que este error disminuya teniendo un mejor control en la orientación del robot. Por otra parte al hacer el proceso de optimización del mapa, observamos que los datos de los marcadores se optimizan de forma local, pero al pasarlos al mapa global aún dependen de la posición del robot como se ve en la ecuación (3.16), esto es debido principalmente a que la forma en como se planteó el problema y se realizó el entorno, a veces no había una forma de relacionar las fotos, es decir no había un marcador en común en dos fotografías diferentes, es probable que si se hubieran puesto más marcadores en el entorno se podría realizar esta relación entre marcadores, en lugar de relacionarlos únicamente por la posición del robot en el entorno. Entonces lo marcadores se habrían relacionado entre si, en lugar de solo estar relacionados con el origen, por lo tanto al optimizarse de forma local al depender el uno del otro la optimización tal vez podría ser mejor. Aún así como se muestra en la sección 4.3 el resultado de los mapas obtenidos es bueno.

Para navegar en el mapa se seleccionaron los puntos de una ruta de forma manual para que el robot la siguiera, los puntos de esta ruta se podrían obtener de forma automática obteniendo el esqueleto del mapa una vez terminada la optimización y la etapa de reconocimiento. La prueba de avance variable utiliza una tabla de consulta para avanzar, pero si tuviera un mejor control en los motorreductores se podría utilizar una ecuación de control para avanzar más rápido y con mayor precisión hacia el punto deseado.

Con los marcadores de tipo de orden y una cámara usb fue posible realizar la caracterización de los movimientos del robot, con lo que fue posible realizar un control por tiempo de los movimientos del robot y esto permitió obtener los resultados que se presentan en el capítulo 4, esto a pesar de que el SDK del robot utilizado es muy restringido en las funciones de movimiento del robot. Es importante resaltar que las secciones de evasión en la página 40, de orientación en la página 41, de ajuste lateral en la página 42, y de orientación de la cámara en la página 43, están diseñadas específicamente para funcionar con este robot, con los componentes y el SDK sin modificaciones, por lo tanto estas secciones se deben adaptar para otro robot.

La heurística de reconocimiento del entorno implementada es buena, pero si se tuviera mayor control al dar los giros del robot, se tendría un resultado mejor, con menor error de incertidumbre en el mapa. Para la detección de los marcadores en los obstáculos y en las paredes fue necesario tratar de mantener el entorno con luz lo más uniforme posible, ya que esto afectaba mucho la detección de los marcadores y por esto fue necesario agre-

gar una estrategia de validación ya que a veces se detectaban marcadores que no eran (positivos falsos), para corregir este problema se decidió que un marcador debe de ser reconocido al menos dos veces para que se pueda almacenar en el árbol binario. También se utilizó una velocidad de avance baja en el robot para el reconocimiento del entorno, ya que al ir más rápido podía no ver marcadores, debido a que se toma una foto cada que se detiene y el robot no hace ningún procesamiento de imagen mientras avanza. Algunos lados de los obstáculos no los detectaba ya que al avanzar, estos no estaban en su campo de visión.

El diseño como máquinas de estados del servidor de control, de visión y del cliente, es debido a que algunos estados están compuestos de distintos algoritmos para el procesamiento de los datos y algunos algoritmos pueden tardar más en terminar que otros y también pueden depender de la respuesta de algoritmos anteriores. Es por eso que se diseñó como una máquina de estados ya que permite que exista una sincronización entre los datos procesados y las acciones a realizar por el robot, ya que aunque el servidor de control y de visión están en el robot, estos no están comunicados y el cliente es el que se encarga de analizar, gestionar y utilizar de forma ordenada los datos de los servidores.

El entorno de pruebas se construyó pensando en que se trabajaría en casa por la pandemia, pero el algoritmo propuesto para resolver el problema de la localización y mapeo simultáneo utilizando marcadores de tipo de orden con el robot utilizado en este trabajo debe funcionar si el entorno es más grande y con más objetos dentro, siempre y cuando el entorno sea un polígono rectangular, ya que la maniobra de evasión está diseñada para dar giros de 90 grados a la izquierda.

5.1. Trabajo a futuro

Una lista de trabajos a futuro en los que se ha pensado incluye los siguientes puntos.

- Se puede desarrollar una biblioteca y una etapa de potencia, donde se incluyan sensores para estimar posición y controlar la velocidad de los motores del robot.
- Se puede crear un entorno más general, en lugar de que los marcadores estén sobre una sola madera, se podrían poner como señalamientos de tránsito y de esta forma resultaría más fácil moverlos y crear nuevos escenarios.
- Realizar el mapa utilizando una relación entre marcadores, en lugar de la relación con la posición del robot dentro del entorno que se presenta en la ecuación (3.16).
- Realizar la optimización del mapa en línea, durante la etapa de reconocimiento.

- Realizar una estrategia de reconocimiento utilizando un barrido del entorno con la cámara del robot. En lugar de mantener la cámara viendo siempre al frente, se hace un barrido del entorno antes de avanzar con los siguiente giros de la cámara -90°, -60°, -30°, 0°, 30°, 60° y 90°, con 3 umbrales de segmentación de la imagen distintos en cada ángulo del barrido para el problema de la iluminación, así el robot podría detectar más marcadores, además de que se podrían validar y descartar marcadores durante el barrido.
- Implementar la obtención automática de rutas del mapa una vez terminado el reconocimiento y la optimización del mapa.
- Implementar ecuaciones de control para el avance y la orientación del robot de una forma más precisa.
- Realizar la interfaz gráfica para visualizar el mapa y la posición del robot en tiempo real.
- Implementar la capacidad de detectar en línea un cambio en el entorno, actualizar el mapa y el mapa de rutas.
- Implementar la capacidad de localizarse dentro de un mapa ya generado sin importar donde se coloque el robot dentro del entorno.
- Probar el algoritmo propuesto en otros robots.

Bibliografía

[1] Daybelis Jaramillo Olivares. Prototipo de un sistema de ojo de halcón. Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco, Departamento de Computación, 2018.

[2] Sergio Alberto Herrera Castro. Sistema de odometría visual e inercial con un marcador. Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco, Departamento de Computación, 2019.

[3] Cyrill Stachniss, John J Leonard, and Sebastian Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 1153–1176. Springer, 2016.

[4] Rafael Munoz-Salinas, Manuel J Marin-Jimenez, and Rafael Medina-Carnicer. Spm-slam: Simultaneous localization and mapping with squared planar markers. *Pattern Recognition*, 86:156–171, 2019.

[5] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[6] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[7] Laleh Armi and Shervan Fekri-Ershad. Texture image analysis and texture classification methods-a review. *arXiv preprint arXiv:1904.06554*, 2019.

[8] Rafael Munoz-Salinas and Rafael Medina-Carnicer. Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers. *Pattern Recognition*, 101:107193, 2020.

[9] Yingchun Fan, Qichi Zhang, Yuliang Tang, Shaofen Liu, and Hong Han. Blitz-slam: A semantic slam in dynamic environments. *Pattern Recognition*, 121:108225, 2022.

[10] Heriberto Cruz Hernández. *Solución Simultánea de Varios Subproblemas de Visión por Computadora*. PhD thesis, Centro de Investigación y de Estudios Avanzados del

Instituto Politécnico Nacional, Unidad Zacatenco, Departamento de Computación, 2019.

[11] Luis Gerardo de la Fraga and Heriberto Cruz Hernández. Optimizing the maximal perturbation in point sets while preserving the order type. *Mathematical and Computational Applications*, 24(4):97, 2019.

[12] Gonzalo Adán Chávez Fragoso. Reconocimiento de marcadores con redes profundas. Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco, Departamento de Computación, 2020.

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

[14] Peter I Corke and Oussama Khatib. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011.

[15] Andrew Zisserman Richard Hartley. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd ed edition, 2003.

[16] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[17] UCTRONICS. (sku: K0073)uctronics robot car kit for raspberry pi v1.0. `https://www.uctronics.com/wiki/(SKU:_K0073)UCTRONICS_Robot_Car_Kit_for_Raspberry_Pi_V1.0`, 2019. Acceso 24 de abril del 2022.

[18] OpenCV. Camera calibration. `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html`, 2020. Acceso 22 de junio del 2022.

[19] UCTRONICS. Robot car kit. `https://play.google.com/store/apps/details?id=com.uctronics.Robot_Car_Kit`, 2018. Acceso 24 de abril del 2022.

[20] SCRATCH. Acerca de scratch. `https://scratch.mit.edu/about`, 2022. Acceso 22 de julio del 2022.

[21] UCTRONICS. Uctronics / uctronics_smart_robot_car_raspberrypi. `https://github.com/UCTRONICS/UCTRONICS_Smart_Robot_Car_RaspberryPi`, 2020. Acceso 24 de abril del 2022.

[22] Frédéric Devernay. C/c++ minpack. `http://devernay.github.io/cminpack`, 2007.