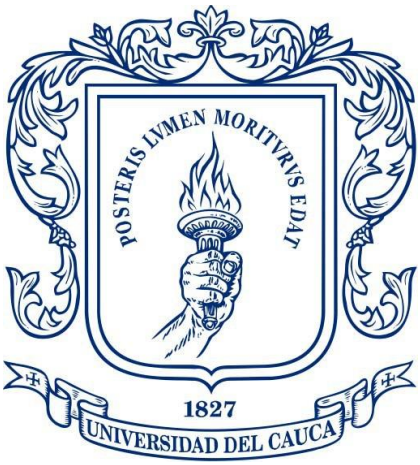
DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA DE DETECCIÓN Y RECONOCIMIENTO DE OBSTÁCULOS PARA INTERACCIÓN DE ROBÓTS MÓVILES POR VISIÓN DE MÁQUINA



CRISTIAN ANDRES TOBAR MOSQUERA
JOSE DANIEL MONTENEGRO RIVERA

Universidad del Cauca

Facultad de Ciencias Naturales, Exactas y de la Educación

Ingeniería Física

Popayán

2018

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA DE DETECCIÓN Y RECONOCIMIENTO DE OBSTÁCULOS PARA INTERACCIÓN DE ROBÓTS MÓVILES POR VISIÓN DE MÁQUINA

Tesis de Grado para optar al título de:

Ingeniero Físico

CRISTIAN ANDRES TOBAR MOSQUERA
JOSE DANIEL MONTENEGRO RIVERA

Director:

Mg. Leonairo Pencue Fierro

Universidad del Cauca

Facultad de Ciencias Naturales, Exactas y de la Educación

Ingeniería Física

Popayán

2018

Nota de Aceptación

Director----------
MSc. LEONAIRO PENCUE FIERRO

Jurado----------
MSc. PABLO JAVIER SALAZAR VALENCIA

Jurado----------
MSc. EDUARDO ANDRÉS CAÑOLA SOTELO

Fecha de Sustentación: Noviembre 01 de 2018

AGRADECIMIENTOS

A Dios por permitirme culminar una etapa más en mi vida, a mis padres por el apoyo incondicional a lo largo de mi carrera y del diario vivir, a mi hermano por siempre estar pendiente de mí, a mi novia por su confianza y motivacion en estos últimos semestres y a toda mi familia por el apoyo constante. ***Cristian A. Tobar.***

Primero agradecer a Dios por ayudarme en la culminación de una etapa tan importante como lo es la universidad, a mi familia por su apoyo incondicional, especialmente a mi madre y mi abuela ya que sin ellas hubiera sido muy difícil conseguir este logro, y por ultimo pero no menos importante a mi novia por ser uno de los mayores apoyos y motivaciones durante todo el proceso de formación. ***Daniel Montenegro.***

A la Universidad del Cauca por habernos brindado una educación integra y de calidad, a su cuerpo docente en especial a los vinculados al departamento de Física por su ardua labor de enseñanza, a nuestro director Mg. Leonairo Pencue Fierro por su tiempo y dedicación mostrada a lo largo del proyecto y a nuestros compañeros por las experiencias compartidas a lo largo de la carrera.

RESUMEN

Este proyecto muestra como a través del procesamiento de una imagen panorámico cenital del entorno de movimiento de un robot móvil, el sistema es capaz de guiarlo de manera eficiente y dinámica a través de una serie de obstáculos hasta el punto de destino indicado desde el computador, de tal modo que no colisione con los objetos presentes en la plataforma.

En el proyecto se evaluaron tres fases fundamentales que permitieron llegar a su culminación. Inicia con un algoritmo de reconocimiento de obstáculos por color. El color elegido fue el blanco, ya que contrasta fácilmente con el fondo uniforme de la base negra. Cuando el reconocimiento de los obstáculos fue bueno, se procedió al desarrollo del algoritmo principal del sistema para identificación de trayectorias a partir de esta información.

La segunda fase fue el acondicionamiento del ambiente de trabajo, en este se construyó una plataforma con la iluminación adecuada para toda la experimentación requerida a lo largo del proyecto. En esta fase también se hizo el reconocimiento del carro por marcadores visuales de color, en este caso verde y rojo. Finalmente se procede a la recepción y transmisión de datos entre el computador y la Raspberry. Gracias a la buena segmentación tanto de los objetos como de los marcadores, se procede a calcular el ángulo de inclinación del carro. Este ángulo es enviado constantemente a la Raspberry para que esta pueda comparar la posición angular actual del robot móvil con la requerida y así iniciar el movimiento. Además de esto no solo se trabajó con objetos estáticos, sino también dinámicos, es decir cuando están o no en movimiento, pues cuando un objeto interfiere en la trayectoria del carro, este se detiene hasta que el sistema encuentre una nueva ruta.

Palabras clave: Imagen panorámica cenital, Segmentación, Marcadores visuales.

ABSTRACT

This project shows how through the processing of a zenith panoramic image of the movement environment of a mobile robot, the system is able to guide it efficiently and dynamically through a series of obstacles to the point of destination indicated from the computer, in such a way that it does not collide with the objects present on the platform.

In the project, three fundamental phases were evaluated that allowed to reach its culmination. Start with an obstacle recognition algorithm by color. The color chosen was white, as it contrasts easily with the uniform background of the black base. When the recognition of the obstacles was good, we proceeded to develop the main algorithm of the system to identify trajectories from this information.

The second phase was the conditioning of the work environment, in this a platform was built with adequate lighting for all the experimentation required throughout the project. In this phase, the car was also recognized by visual color markers, in this case green and red. Finally we proceed to the reception and transmission of data between the computer and the Raspberry. Thanks to the good segmentation of both objects and markers, we proceed to calculate the angle of inclination of the car. This angle is constantly sent to the Raspberry so that it can compare the current angular position of the mobile robot with the required one and thus start the movement. In addition to this, not only were static objects worked, but also dynamic, that is, when they are in motion or not, because when an object interferes with the trajectory of the car, it stops until the system finds a new route

CONTENIDO

PÁG.

LISTA DE FIGURAS

LISTA DE TABLAS

PÁG.

LISTA DE SIMBOLOS

$\subset$ Inclusión

‘ Complemento

$\cup$ Unión

$\cap$ Intersección

¥ Translación

$\ominus$ Resta de Minkowski

$\rightarrow$ Implicación

{A : B} = { A |B} notación constructora de conjuntos

$\forall$ Para todo

$\in$ Pertenece

GLOSARIO

Imagen panorámica cenital: Es aquella donde el punto de vista de una cámara se encuentra perpendicular respecto del suelo y la imagen obtenida ofrece un campo de visión orientado de arriba abajo.

Segmentación: Proceso mediante el cual en una imagen se establecen dos niveles, generalmente ceros y unos o blanco y negro. Usualmente se separan los objetos de interés del fondo presente en la escena.

Marcadores visuales: Son un sistema de componentes que permiten obtener una referencia del estado inmediato de un robot. Son componentes de mucha relevancia a partir de los cuales es posible establecer parámetros a un cuerpo, como por ejemplo un robot.

CAPITULO 1

INTRODUCCIÓN

La visión por computador es una de las ramas de la inteligencia artificial que estudia cómo procesar, analizar e interpretar imágenes de forma automática, junto con la robótica móvil ha permitido implementar una serie de sistemas que satisfacen muchas necesidades. Esta es una disciplina que trata de sistemas y máquinas que se comportan de una forma inteligente brindando aplicaciones a la seguridad y mejorando la calidad de vida de las personas.

En este trabajo vamos a crear un sistema que permita orientar un carro a través de una serie de obstáculos estáticos o dinámicos en un ambiente controlado, usando una plataforma de detección y reconocimiento de obstáculo haciendo uso de la visión de máquina. Esto permitirá reducir el gasto innecesario de energía en procesos que consistan en el guiado de robots móviles en un entorno incierto.

1.1 Objetivos del proyecto

El objetivo de este proyecto es diseñar e implementar un sistema para detección y reconocimiento de obstáculos que permita la interacción de robots móviles con su entorno, haciendo uso de la visión de máquina, además buscar una forma de generación de trayectorias para estos robots partiendo de la información del entorno físico inmediato en el que se encuentran sin necesidad de contar con sensores empotrados en la máquina. En este proyecto se busca no solo generar trayectorias a partir de un entorno secuencial de imágenes panorámico cenitales, sino también generar información, y permitir al robot moverse sufrir choques, de tal forma que pueda llegar a un determinado destino sin problema.

Los objetivos específicos son:

- Desarrollar un algoritmo de reconocimiento de obstáculos.
- Implementar un programa de toma de decisiones que pueda guiar el móvil de manera eficiente y dinámica.
- Diseñar e implementar el sistema de locomoción del robot móvil.

- Establecer y enlazar el sistema de comunicación entre el robot móvil y el ordenador.
- Diseñar una plataforma con un ambiente controlado que permita verificar el correcto funcionamiento de cada una de las etapas del proceso tanto de software como de hardware.

CAPITULO 2

MARCO TEÓRICO

2.1 Visión por computador

2.1.1 Generalidades

La visión por computador es una de las ramas de la inteligencia artificial que estudia cómo procesar, analizar e interpretar imágenes de forma automática. Es una ciencia que arranca con la aparición de los primeros computadores en los años cincuenta. Tras unos comienzos esperanzadores que produjeron unas expectativas demasiado ambiciosas fue abandonada una década después por lo limitado de los recursos conseguidos. Hubo que esperar el desarrollo de la informática durante los años ochenta para que, con un mayor realismo aparecieran las primeras aplicaciones industriales de la visión por computador [1]. La función principal de la visión por computador es lograr ver el mundo físico como lo puede observar el ojo humano.

2.1.2 Cámaras

El funcionamiento de las cámaras web es bastante similar al de las cámaras digitales, su mayor diferencia se centra en el tipo de software que tiene la cámara web. Ambos tipos de cámara son la evolución de las cámaras de película fotográficas, ya que estas reemplazaron las películas, por una matriz de elementos fotosensibles que convierten la luz que capta en señales eléctricas.
Cada uno de los elementos fotosensibles que compone el sensor es denominado píxel, el número de píxeles del sensor generalmente se mide en millones de pixeles (Megapíxeles), estos elementos se pueden definir en términos de cuánto podemos ampliar la imagen sin perder nitidez, por lo que esta característica es determinante pero no la más importante al momento de seleccionar una cámara.

Al tratarse de una cámara digital las señales emitidas por los elementos fotosensibles son interpretadas y posteriormente mostradas en el Display de la

misma, al tratarse de una cámara web lo que se tiene es que la señal obtenida puede ser llamada, interpretada y procesada desde varios programas del computador.

2.1.2 Segmentación

La información o tareas que un sistema de visión artificial puede realizar van desde la simple detección de objetos sencillos en una imagen, hasta la interpretación tridimensional de escenas complicadas [2], para ello se usan técnicas como la segmentación, que consiste en aislar las regiones u objetos de interés [3], siguiendo métodos como el de discontinuidad o similitud entre los niveles de gris de píxeles vecinos, con el fin de procesar esta información y aplicarla en un sistema dado, obteniendo como resultado imágenes con dos niveles claramente distinguidos. Las imágenes binarias son aquéllas que tienen dos niveles, generalmente blanco y negro. Por lo tanto, pueden ser representadas mediante conjuntos. Por ejemplo, el conjunto de todos los píxeles blancos de una imagen blanco y negro constituyen una descripción completa de la imagen [4].

2.1.3 Transformaciones morfológicas

- Transformaciones morfológicas en imágenes binarias

Las transformaciones morfológicas son aquellas que modifican la estructura o forma de los objetos que están presentes en la imagen. Inicialmente estas técnicas se desarrollan para imágenes binarias (dos niveles de gris) aunque después se extendieron los mismos conceptos a imágenes con varios niveles de gris. Estas herramientas además de ser útiles para la extracción de características permiten la eliminación de ruido que se produce en todo el proceso de segmentación [5][6].

La morfología matemática se basa en la teoría de conjuntos y en la topología. Algunas consideraciones previas son:

Inclusión. Un conjunto Y estará incluido en el otro X.$Y \subset X$, si todo elemento de Y pertenece a X,

$$\forall\, y \in Y \rightarrow y \in X.$$

Complemento. El complemento de X, X', son todos los elementos que no pertenecen a X.

Unión. La unión de dos conjuntos X e Y, X ∪ Y, son todos los elementos que están incluidos en uno de los dos, y se define como:

$$X \cup Y = |x|\ x \in X\ o\ x \in Y.$$

Intersección. La intersección de dos conjuntos son los elementos que tienen comunes y se define utilizando los conceptos de unión y complemento.

$$X \cap Y = (X' \cup y')'$$

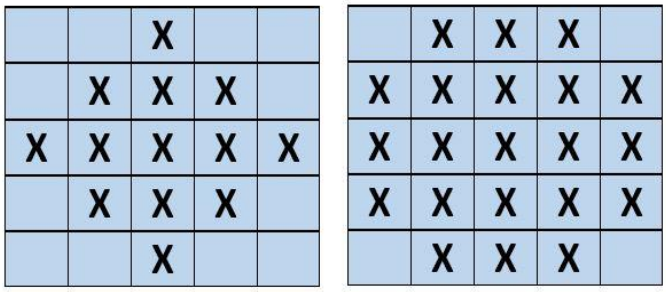
Traslación. Un conjunto X es trasladado por un vector μ, cuando cada uno de los elementos de ese conjunto sufre esa traslación. Se denominara al nuevo conjunto X μ.

A cada transformación, ¥(X), se le puede asociar una transformación dual, ¥ *(X), que cumple:

$$¥^*(X) \rightarrow (¥^*(X'))'$$

- Elemento estructural

Los elementos estructurantes son un conjunto de puntos que servirán para determinar la estructura de un conjunto X. Uno de ellos constituirá el centro del elemento. Como ejemplo se tienen dos elementos estructurantes uno en forma de rombo y el otro de circunferencia con el origen de ambos en el pixel central (*Figura 2.1*).



a) b)

Figura 2.1. Elementos estructurales. a) Rombo. b) Circunferencia (Fuente: Propia)

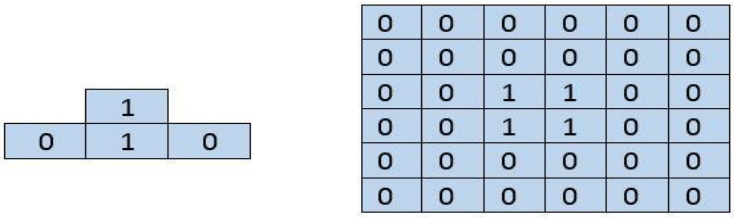
Una transformación de acierta o falla (hit or miss), se aplicará a cada punto de un conjunto X de la siguiente manera:

- Del elemento estructural B se formará un conjunto de Bx que será ese elemento estructural desplazado por todo el elemento x perteneciente al conjunto X. Además Bx, estará formado por dos subconjuntos B1x y B2x, correspondiendo el primero a los elementos del primer plano y el segundo a los del fondo. Obviamente su intersección será el conjunto vacío.

- Un punto x pertenece a la transformación de acierta o falla, X⊗B, si y sólo si B1x está incluido en X y B2x está incluido en X':

$$X \otimes B = |x| \; B1x \subset X1; \; B2x \subset X'$$

En otras palabras esta transformación indica donde coincide exactamente el elemento estructural B dentro del conjunto X y de ahí proviene el nombre <acierta o falla>. Se tiene por ejemplo en la *Figura 2.2* la siguiente imagen y elemento estructural.

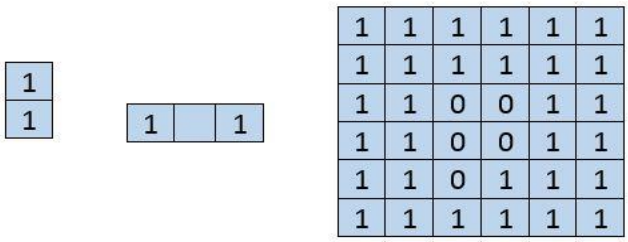


B Imagen X

a) b)

Figura 2.2. Esquema de trabajo del elemento estructural. a) Elemento estructural. b) Imagen. (Fuente: Propia)

Se obtendrán los elementos estructurales para la parte del objeto y del fondo, así como el complemento del conjunto X (*Figura 2.3*).

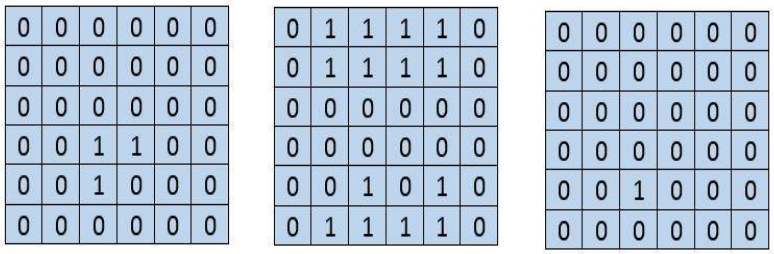


B1x B2x Imagen X’

a) *b)*

Figura 2.3. Formación de subconjuntos del elemento estructural. a) Elementos estructurales del objeto y del fondo. b) Complemento de la imagen. (Fuente: Propia)

Por último, se realizan las operaciones que define a la transformación acierta, representados en la *Figura 2.4.*



$B1x \subset X1$ $B2x \subset X2$ $X \otimes B$

Figura 2.4. Resultado de la transformación acierta o falla. (Fuente: Propia)

Comprobando que efectivamente solo aquel punto coincide con el elemento estructural se mantiene a nivel alto. Con esto se puede pasar a la definición de las dos operaciones morfológicas principales, la erosión y la dilatación [1].

- Erosión

Es la degradación progresiva de uno de los campos (0 o 1). Un elemento del campo a degradar seguirá perteneciendo al mismo si está rodeado de elementos iguales a él. En caso contrario, pasará al otro campo. En un proceso de iterativo terminará por destruir la imagen. Matemáticamente se expresa de la siguiente forma: si se toma el elemento estructural simétrico respecto al origen de B, B', la erosión de un conjunto X respecto al elemento B es:

$$X \ominus B' = \{x | Bx \subset X\}$$

Lo que es igual a una transformación acierta o falla donde B2x es un conjunto vacío.

El símbolo $\ominus$ representa la resta de Minkowski.

Dependiendo del tamaño del elemento estructural, la erosión será más pronunciada o no. Por otro lado a veces es necesario realizar varias erosiones seguidas. Por ello se ha encontrado la relación entre el tamaño del elemento y el número de veces que se aplica:

$$d = 1 + (n-1) \text{ Número de iteraciones}$$

Siendo n la dimensión del elemento inicial, y la del elemento equivalente. Así aplicar dos erosiones de un elemento de 3x3 es igual a una 5x5, y tres a otra de 7x7 [1][7].

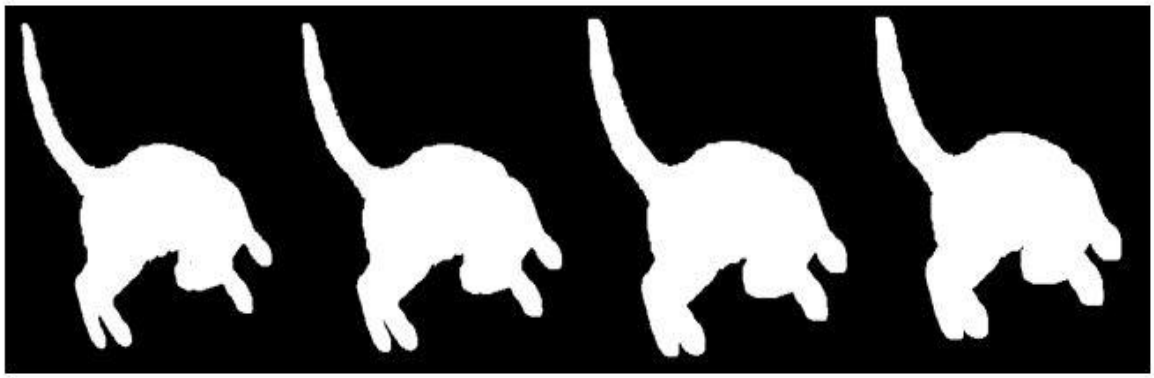
Una interpretación gráfica de la erosión puede verse en la *Figura 2.5*. Donde dependiendo del elemento estructurante, la erosión es más pronunciada.

Original 3x3 7x7 10x10

Figura 2.5. Erosión iterativa con diferente kernel. (Fuente: Propia)

- Dilatación

Es el crecimiento progresivo de uno de los campos (0 o 1). Un elemento del campo contrario a crecer será convertido se posee algún vecino perteneciente al campo que se expansiona. En caso contrario, permanecerá igual. Los elementos pertenecientes al campo a expansionar evidentemente no se modifican. Si se aplicase un número elevado de veces, terminaría por destruir la imagen, ya que todos los píxeles estarían a nivel alto. En la *Figura 2.6*, un objeto se vuelve cada vez más grande al aplicar diferentes elementos estructurantes [1][7].



Original 3x3 7x7 10x10

Figura 2.6. Dilatación iterativa con diferente kernel. (Fuente: Propia)

- Esqueletización

La Esqueletización busca representar una región u objeto por un grafo. Es útil para la representación y es una característica que define el objeto. Hay que resaltar que pequeñas variaciones en la región afectan en gran medida a su esqueleto. Esto puede convertirse en una ventaja si lo que se propone es un control de calidad, ya que será fácil detectar las piezas defectuosas.

Una manera de obtenerlo se basa en la denominada transformación de eje medial (*medial axis transformation* o MAT). Para cada elemento de la región se busca el punto de borde más cercano. Si más de un punto del borde está a esa distancia el elemento pertenece al eje medial que se equipara al esqueleto. Gráficamente se explica mediante el siguiente ejemplo: si el objeto fuera un campo al que se pegase fuego a la vez en todos los puntos del borde, y las llamas avancen a la misma velocidad, el esqueleto lo formarían los puntos a los que lleguen las llamas de dos frentes distintos al mismo tiempo.

Otro enfoque similar es sustituir cada píxel por la distancia más pequeña al borde. El esqueleto vendría dado por los máximos locales

Como puede deducirse de la definición, el coste computacional es prohibido. Por ello se han desarrollado algoritmos permiten encontrar el esqueleto de un objeto de una forma más rápida. Uno de ellos es el que se basa en la aplicación sucesiva de una serie de filtros morfológicos [1]. Un ejemplo del esqueleto de un objeto se muestra en la *Figura 2.7*, donde el los puntos que tienen más tres pixeles unidos se

conocen como bifurcaciones. Los puntos que están a los extremos se denominan terminaciones. En la imagen se aprecian cinco terminaciones y tres bifurcaciones.

Figura 2.7. Esqueleto de una imagen binarizada. (Fuente: Propia)

2.2 Robótica móvil

2.2.1 Generalidades

Existen muchas definiciones de la palabra robot. En cada una de ellas encontramos destacado algún aspecto en particular, que es el que cada autor quiere resaltar en su obra. Según la Asociación Japonesa de Robótica Industrial (JIRA), los robots son dispositivos capaces de moverse de modo flexible, análogo al que poseen los organismos vivos, con o sin funciones intelectuales, lo que permite la realización de operaciones en respuesta a órdenes recibidas por humanos. Vemos que en esta definición se encuentra resaltada la capacidad de movimiento de los robots y su analogía con los seres de la naturaleza. Sin embargo, a la JIRA no le interesa la inteligencia artificial aplicada al robot, dado que su función fundamental es ser operado por un humano. Por su parte, el Instituto de Robótica de Norteamérica (RIA) define a un robot industrial como un manipulador multifuncional y reprogramable diseñado para desplazar materiales, componentes, herramientas o dispositivos especializados por medio de movimientos programados variables, con el fin de realizar diversas tareas. En este caso, el acento está puesto en la capacidad de programación del robot y, por lo tanto, en cierta independencia de funcionamiento con respecto a la operación humana. Como dijo Joseph Engelberg, padre de la robótica industrial:

Es posible que no sea capaz de definir qué es un robot, pero sé cuándo veo uno. Particularmente, y ya que nos hemos ganado el derecho dado que estamos

escribiendo un libro sobre robótica, agregaremos una definición más de robot a la larga lista preexistente:

Un robot es un dispositivo con un determinado grado de movilidad, que puede realizar un conjunto de tareas en forma independiente y que se adapta al mundo en el que opera [7]. Los robots de detección de entorno pueden formar parte componente de máquinas herramientas o utilizarse como un sistema autónomo para un sinnúmero de aplicaciones [8], [9]. Un ejemplo de esto es el Curiosity (*Figura 2.8*), un robot enviado por la nada en el año 2011 al planeta marte, con el fin de explorar este terreno.

Figura 2.8. Robot Curiosity (Fuente: www.nasa.gov)

2.2.2 Algoritmo de Dijkstra

El problema de la trayectoria mínima es uno de los problemas básicos en la teoría de grafos y este puede ser resuelto vía el algoritmo de Dijkstra. Para ilustrar una solución con Dijkstra, consideremos el siguiente problema de rutas representado en un grafo como $G = \{N|A, W\}$; donde N son los nodos, A las aristas y W los pesos de las aristas. El grafo G tiene como nodo origen $n1 \subset n$, y cada arista es unida por un par de nodos, tal que $Np = \{nj, nk|\forall p = 1,2,3,4, \dots m\}$. Los pesos de las aristas están dados por la función peso $W = \{w_1, w_2, w_3, \dots wm|wi \in [0, \infty]\}$ y para cada wp son asociados pares Np. Podemos definir que el costo de una ruta S puede ser

determinado identificando los nodos de la ruta; como bs = … nk; y el costo del camino es la suma de las ponderaciones de las aristas que están asociadas en bs; tal que xs = wm + wn … + wp . Por consiguiente cualquier ruta admisible bs puede ser categorizada como: ds ($n1$, nr) = {xs : $n1 \rightarrow nr$ |$nr \subset n$, ∀s = 1,2 … ,t}.

Donde t es el número de trayectorias que pueden ser resueltas. Para encontrar los caminos, se parte del nodo inicial y se consideran únicamente los nodos a los cuales se puede llegar directamente. Continuando por el camino se consideran los nodos a los que se puede llegar por medio del nodo anterior y desde el nodo inicial. Así sucesivamente hasta llegar al nodo final. Cada trayectoria de nodos posee un costo asociado, el cual se va almacenado y evaluando para encontrar el costo mínimo [10], [11], [12].

La solución del conjunto de la ecuación (1) es dada por:

d = min {ds (p,)|∀s = 1,2 … ,t}, (2)

La ecuación (2) significa que dado un conjunto de trayectorias, la solución óptima es la mínima suma de las ponderaciones del conjunto (p, nr); la cual corresponde a una ruta definida en bs como la ruta mínima [8]. Un ejemplo de este algoritmo se aprecia en la *Figura 2.9*, donde la posición inicial está en A y su objetivo será llegar hasta E.

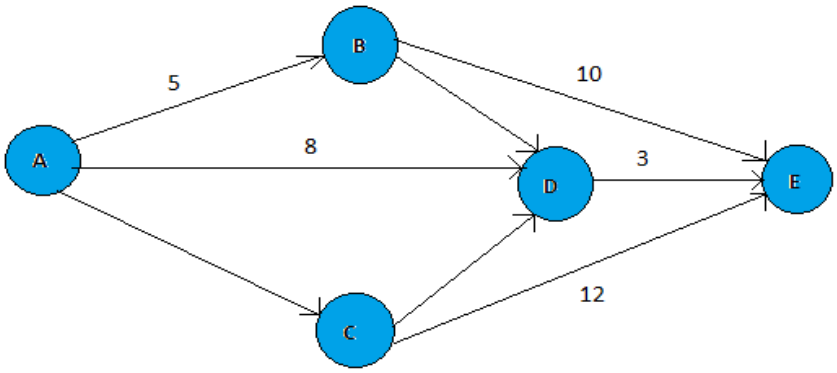


Figura 2.9. Ejemplo algoritmo de Dijkstra. (Fuente: Propia)

2.3 Sistemas de control

El control automático ha desempeñado un papel vital en el avance de la ingeniería y la ciencia. El control automático se ha convertido en una parte importante e integral en los sistemas de vehículos espaciales, en los sistemas robóticos, en los procesos modernos de fabricación y en cualquier operación industrial que requiera el control de temperatura, presión, humedad, flujo, etc. Es deseable que la mayoría de los ingenieros y científicos estén familiarizados con la teoría y la práctica del control automático [13], [14], [15].

2.3.1 Sistema de control de lazo abierto

Los sistemas en los cuales la salida no tiene efecto sobre la acción de control se denominan sistemas de control en lazo abierto. En otras palabras, en un sistema de control en lazo abierto no se mide la salida ni se realimenta para compararla con la entrada. Un ejemplo práctico es una lavadora. El remojo, el lavado y el centrifugado en la lavadora operan con una base de tiempo. La máquina no mide la señal de salida, que es la limpieza de la ropa. En cualquier sistema de control en lazo abierto, la salida no se compara con la entrada de referencia. Así, a cada entrada de referencia le corresponde una condición de operación fija; como resultado de ello, la precisión del sistema depende de la calibración. Ante la presencia de perturbaciones, un sistema de control en lazo abierto no realiza la tarea deseada. En la práctica, el control en lazo abierto sólo se usa si se conoce la relación entre la entrada y la salida y si no hay perturbaciones internas ni externas. Es evidente que estos sistemas no son de control realimentado. Obsérvese que cualquier sistema de control que opere con una base de tiempo está en lazo abierto. Por ejemplo, el control de tráfico mediante señales operadas con una base de tiempo es otro ejemplo de control en lazo abierto [13]. En la *Figura 2.10* se observa el diagrama para un sistema de control de lazo abierto.

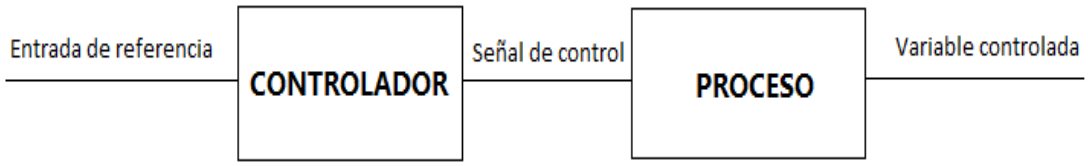


Figura 2.10. Esquema general para un sistema de control de lazo abierto. (Fuente: Propia)

2.3.2 Sistema de control de lazo cerrado

Los sistemas de control realimentados se denominan también sistemas de control en lazo cerrado. En la práctica, los términos control realimentado y control en lazo cerrado se usan indistintamente. En un sistema de control en lazo cerrado, se alimenta al controlador la señal de error de actuación, que es la diferencia entre la señal de entrada y la señal de realimentación (que puede ser la propia señal de salida o una función de la señal de salida y sus derivadas y/o integrales), con el fin de reducir el error y llevar la salida del sistema a un valor deseado [14], [15] (*Figura 2.11*).

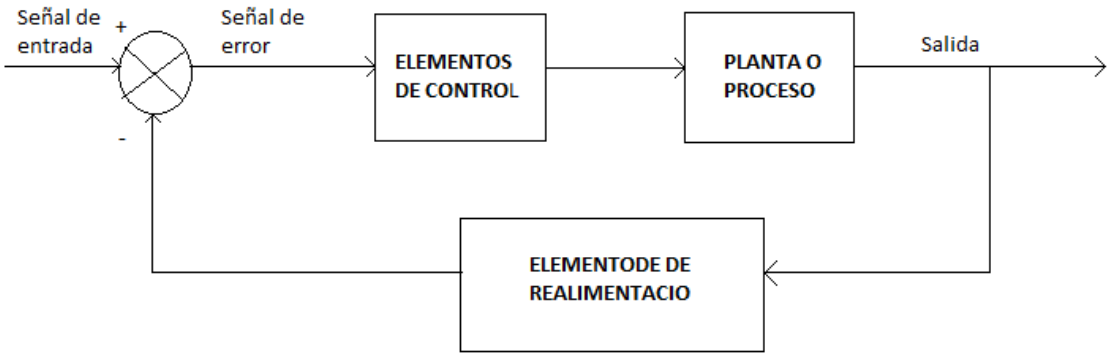


Figura 2.11. Esquema general para un sistema de control de lazo cerrado. (Fuente: Propia)

2.4 software de visión por computador

2.4.1 OpenCv

OpenCv (Open Source Computer Vision Library) es una biblioteca de software de visión abierta y software de aprendizaje automático. OpenCv fue construido para proporcionar una infraestructura común para aplicaciones de visión por computadora y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al ser un producto con licencia de BSD, OpenCv facilita a las empresas utilizar y modificar el código.

La biblioteca cuenta con más de 2500 algoritmos optimizados, que incluyen un conjunto completo de algoritmos de visión artificial y de aprendizaje automático tanto clásico como avanzado. Estos algoritmos se pueden usar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de la cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D desde cámaras estéreo, unir imágenes para producir una alta resolución imagen de una escena completa, encuentra imágenes similares de una base de datos de imágenes, elimine los ojos rojos de las imágenes tomadas con flash, siga los movimientos oculares, reconozca el escenario y establezca marcadores para superponerlo con realidad aumentada, etc. OpenCv tiene más de 47 mil personas de usuarios comunidad y número estimado de descargas que exceden los 14 millones. La biblioteca se usa ampliamente en compañías, grupos de investigación y por organismos gubernamentales [16].

Con toda la teoría descrita a lo largo de este capítulo será fácil entender conceptos mencionados en tramos posteriores. A continuación se dará a conocer el proceso de implementación del proyecto, en este se plasma el proceso que finalmente tuvo éxito y se logró culminar satisfactoriamente, desde la etapa de reconocimiento de obstáculos hasta la conexión inalámbrica para el control del robot móvil.

CAPITULO 3

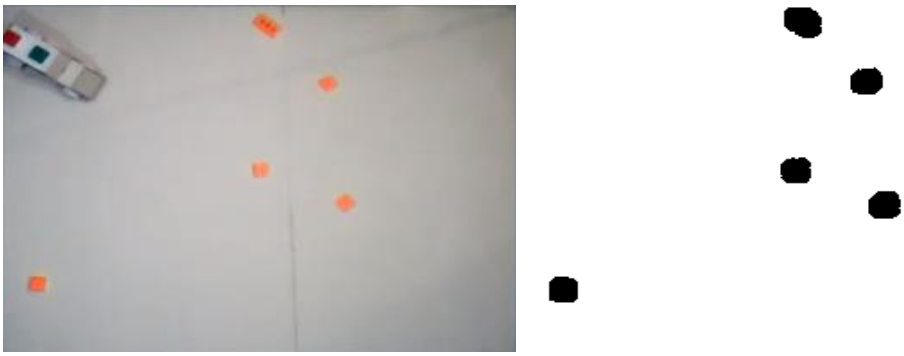
IMPLEMENTACIÓN

3.1 Reconocimiento de obstáculos

Para el reconocimiento de obstáculos se implementaron dos procesos que permitieron una adecuada identificación de la forma y posición de los mismos. Inicialmente se procede a segmentar los objetos, no obstante al trabajar con imágenes reales estas llevan inmerso cierta cantidad de ruido, por lo que se hace necesario implementar un filtrado morfológico en la imagen.

3.1.1. Segmentación de obstáculos

Para el reconocimiento de los objetos por color, se usó el proceso de segmentación con su técnica de Conectar Umbral, la cual se basa en detectar objetos que se encuentran dentro de un rango de intensidad específico, esto fue aplicado sobre un fondo uniforme, donde estos tienen un determinado color. Inicialmente se trabajó con un fondo uniforme de color blanco, los objetos eran de color naranja, y los marcadores para el carro rojo y verde (*Figura 3.1.a*), no obstante, por problemas de iluminación se opta por cambiar el fondo, pues el blanco refleja sobre los objetos y en ocasiones se perdían ambos marcadores e incluso los objetos. En la *Figura 3.1.b* se muestran todos los objetos naranjas segmentados.



a) *b)*

Figura 3.1. Base de la plataforma inicial. a) objetos de color naranja. b) segmentación de los objetos.

Una vez el carro comienza su desplazamiento, en las regiones del centro de la base los marcadores se pierden y el ángulo enviado es incorrecto, a tal punto que la trayectoria hecha por el carro no es la que se espera, por ende se cambió la base de los obstáculos, además de pintar los marcadores, base de la plataforma y objetos con colores mate, con el fin de minimizar el brillo al incidir la luz sobre la superficie lo los mismos.

En la *Figura 3.2*, se observa el resultado final de la base usada para el desplazamiento del carro, el color elegido fue el negro, pues fue el que mejor respuesta tuvo en cuando a la segmentación de los objetos. Estos últimos deben ser de color blanco, aunque si se desea, se puede implementar para otros colores agregando nuevas mascaras. En el caso del carro siempre se usó los marcadores de color verde y rojo, se intentó cambiar el rojo por un azul mate, no obstante en partes de la plataforma fue imposible lograr una buena segmentación.

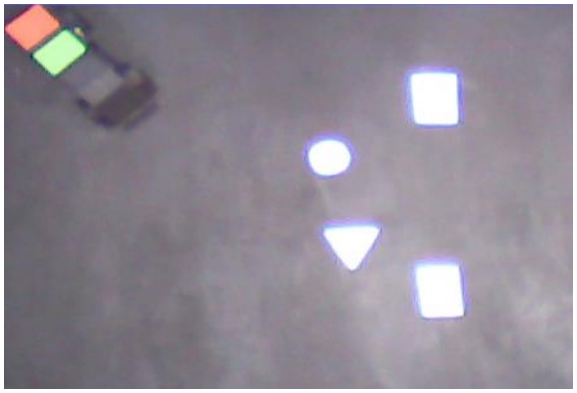
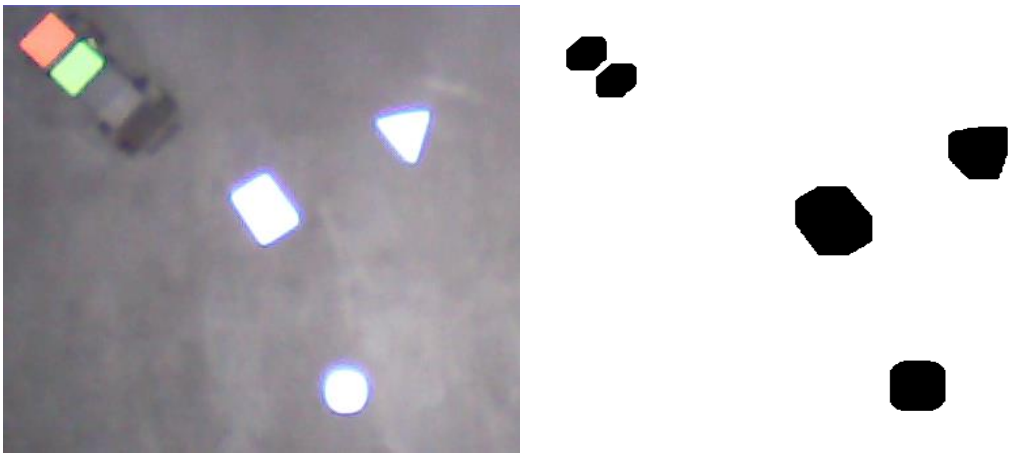


Figura 3.2. Base de la plataforma usada en el proyecto.

3.1.2 Filtros Morfológicos

Una vez se obtiene la segmentación tanto para los objetos como para los marcadores del carro, se observan pequeños puntos a lo largo de la plataforma, que a pesar de dejar el umbral de segmentación lo más cerrado posible fueron imposibles de eliminar, por ende se hizo uso de filtros morfológicos, como la erosión y dilatación, que no solo se usaron con el fin de eliminar ruido, sino también de buscar espacio entre objetos para determinar si el carro puede pasar o no por una determinada trayectoria.

Inicialmente solo se optó por aplicar la erosión a las imágenes, no obstante se encontró que naturalmente el tamaño tanto de los obstáculos como del marcador del auto fue reducido, por lo que se debió hacer el proceso inverso, es decir se debió dilatar la imagen al mismo tamaño que se erosiono. La *Figura 3.3.a*, muestra el frame inicial obtenido al compilar el programa. En la *Figura 3.3.b*, el resultado obtenido tras este proceso de filtrado. En esta imagen se muestra la suma de los objetos y los marcadores visuales usados.



a) b)

Figura 3.3. Segmentación de objetos en la base final. a) Carro con marcadores rojo y verde y objetos blancos. b) Segmentación y filtrado de objetos y marcadores

3.2 Generación de trayectoria

Cuando se cuenta con la correcta segmentación de objetos para su fácil detección, se inicia la búsqueda de un programa que permita obtener un camino óptimo y seguro para el robot móvil. Este programa debe contar con algunas características de funcionamiento, por ejemplo: se debe permitir al usuario elegir el lugar de destino del carro, el lugar de inicio será encontrado bajo la detección del robot y tener en cuenta el ancho carro para evitar choques.

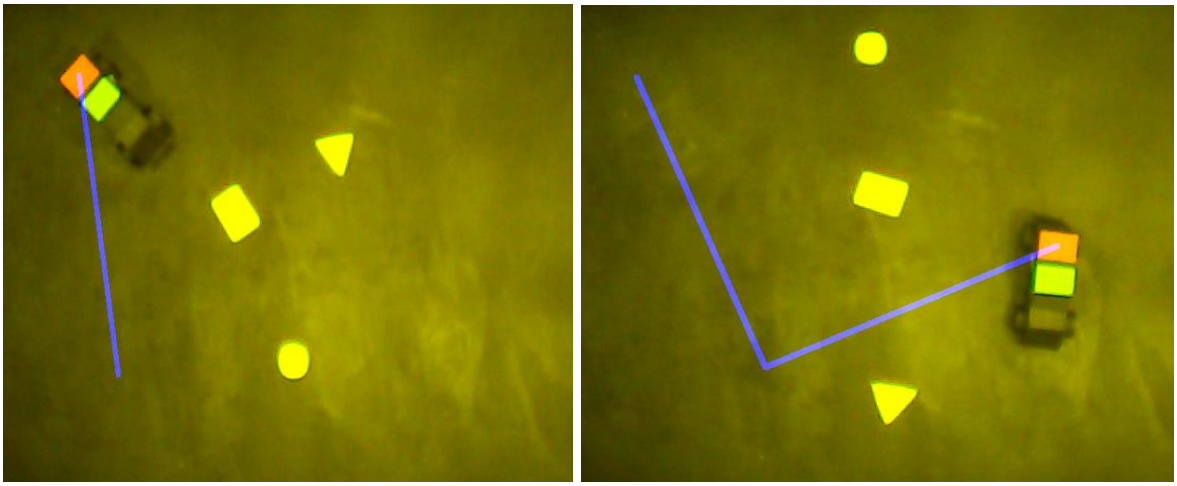
3.2.1 Posición inicial de movimiento

La posición inicial del carro en principio fue una constante, es decir, antes de compilar el programa el auto debía posicionarse en un punto determinado, no obstante se vio la necesidad de obtener un punto inicial dependiendo de la ubicación del robot móvil. Para ello se hizo uso de un marcador por color, el cual se actualiza

cada vez que el programa recalcula las trayectorias, a diferencia de la posición final que debe ser seleccionada por el usuario al iniciar la ejecución del programa y se mantiene constante hasta el final de la trayectoria.

El marcador que se usó fue el color rojo, pues fue el que menor ruido presentó al momento de la segmentación. Posteriormente fue sometido a filtros morfológicos de erosión y dilatación, con el fin de obtener únicamente en pantalla el color deseado. Una vez culminado todo el proceso anterior, se procede a encontrar el centro de masa de dicho marcador con su determinada posición en el eje coordenado, y así encontrar la posición inicial del carro.

En la *Figura 3.4* se ven algunos ejemplos de la posición inicial del carro, y la trayectoria generada desde este punto.



a) b)

Figura 3.4. Trayectoria a partir de la posición del carro. a) Trayectoria directa. b) Trayectoria con inclinación

3.2.2 Generación de puntos característicos

Para encontrar un camino optimo y seguro para el robot móvil se inicia la búsqueda de un método que permita extraer algunos puntos estratégicos en la imagen del entorno de movimiento, de tal forma que sea posible llevar el carro hasta una determinada posición final a partir de dichos puntos.

- Celda de puntos

Para la extracción de los puntos usados para el cálculo de la trayectoria se tomó como base una imagen con fondo uniforme, la cual tiene inmerso una serie periódica de puntos (*Figura 3.5*). Esta es una imagen binarizada que guarda posiciones coordenadas tanto de filas como de columnas de todos los puntos. Este sistema se emplea con el fin de no limitar el programa a trayectorias estándares, pues es independiente de a la posición del carro y gracias a todos estos puntos existe una mayor probabilidad que haya una camino hasta el destino sugerido por el usuario.

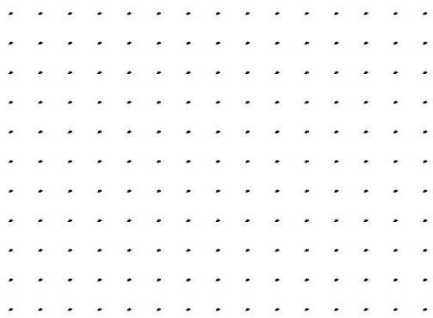


Figura 3.5. Celda de puntos binarizada

Con la celda de puntos construida, se inicia la segmentación de los obstáculos con su respectivo proceso de filtrado, con el fin de que el sistema en determinadas ocasiones no detectará objetos virtuales, es decir puntos que no corresponden a un objeto. En la *Figura 3.6* se muestra el resultado obtenido una vez finalizado el proceso. En esta imagen solo están segmentados los objetos, el carro será abordado posteriormente.

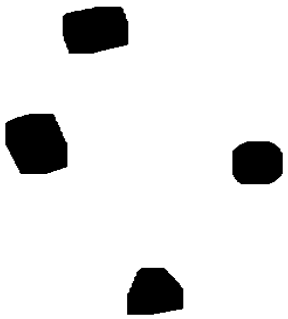


Figura 3.6. Segmentación de objetos con su respectivo proceso de filtrado

- Mascara de selección

Obtenida la imagen segmentada con los obstáculos, empieza la búsqueda de un método, una forma que permita hacer una selección de puntos y permitir obtener una trayectoria. El método data de un proceso de dilatación con un elemento estructural de gran tamaño, con el fin de obtener un único elemento, el cual delimitará el borde de las posibles trayectorias. El resultado de este proceso se muestra en la *Figura 3.7*.

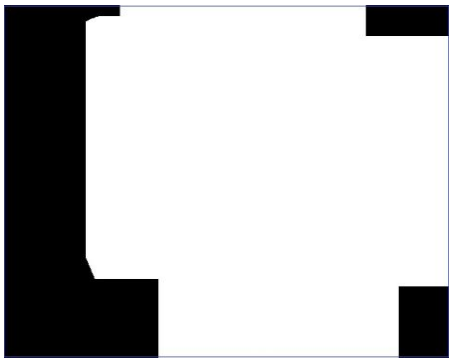


Figura 3.7. Dilatación de los objetos con un elemento estructural grande

Teniendo un área delimitadora tan grande se obtuvieron demasiados puntos que además de saturar el sistema no permitían una buena elección de la trayectoria, lo que llevo a pensar en un anillo de selección.

En la *Figura 3.8* se observan los objetos con un valor binario de uno, es decir están de color blanco. Para finiquitar la selección de puntos para una trayectoria, es necesario hacer una inversión de la imagen de los objetos segmentados, para posteriormente someterlo a una erosión. Es claro que el elemento estructural usado en este caso será más pequeño, ya que si se hace del mismo tamaño o mayor la imagen resultante sería la inicial u otra diferente que no corresponde en lo absoluto a los objetos presentes en la escena.

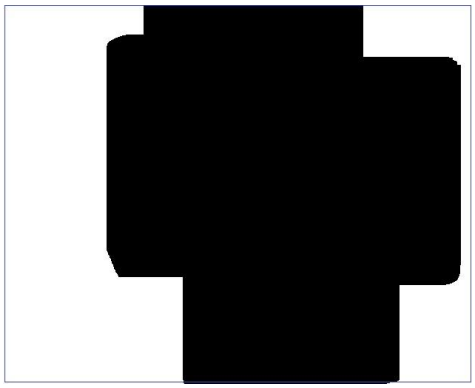
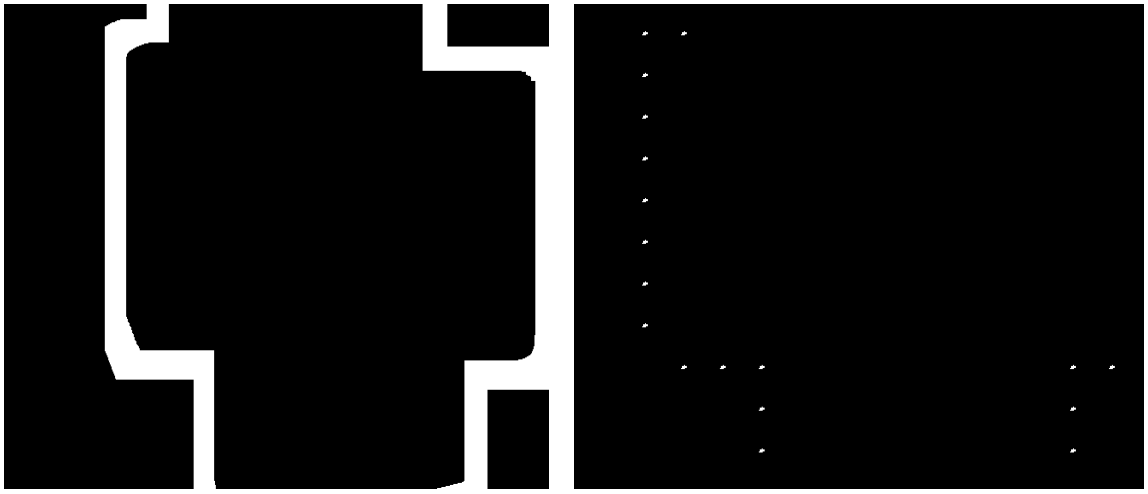


Figura 3.8. Inversión binaria de la segmentación de objetos para la erosión

Obtenidas las dos imágenes de dilatación y erosión inversa de los objetos (*Figura 3.6 y Figura 3.7*), se realiza una multiplicación punto a punto entre estas dos imágenes, el resultado es un entorno alejado de la posición de los obstáculos (*Figura 3.8.a*). Con esta imagen de contorno, se procede a multiplicar esta imagen con la celda de puntos (*Figura 3.5*), esto con el fin de hacer una selección de puntos que garantiza siempre bordear los obstáculos (*Figura 3.8.b*) y hacer más grande la posibilidad de obtener una trayectoria.



a) *b)*

Figura 3.9. Multiplicación de imágenes con la celda de puntos. a) Aro seleccionador. b) Puntos para la trayectoria

Conocido la selección de puntos (*Figura 3.3.b*), se procede a la extracción en vectores de coordenadas, es decir, cada punto tiene su respectiva posición en el

eje x,y o posición fila-columna, esto con el fin de conocer el estado de cada punto que será fundamental en el análisis posterior para encontrar una trayectoria.

3.2.3. Algoritmo de Dijkstra

El algoritmo de Disktra o conocido también como algoritmo de mínimo camino, permite extraer en una serie de puntos, con vértices y nodos la trayectoria más óptima. Además de este algoritmo está el de Bellman-Ford, que establecen el mismo criterio pero a diferencia del primero, plantean distancias negativas, lo que no es una opción en el proyecto, pues no tendría sentido una distancia menor a cero para el desplazamiento del carro. Por ende se establece el algoritmo de Dijkstra como el método principal usado en el desarrollo del programa, la ventaja de este algoritmo es proporcionar un camino de menor costo energético dada una serie de puntos en la imagen, no obstante, la gran desventaja es el coste computacional que conlleva.

- Elección punto a punto

Para el desarrollo del algoritmo de Dijkstra se tienen en cuenta tres aspectos fundamentales, el primero es el estado inicial del carro, ya que este proporciona el punto de partida a partir del cual se empieza la búsqueda de un camino. El segundo son los puntos obtenidos en el proceso descrito anteriormente. El sistema comienza a comparar distancias desde este punto al resto de puntos característicos como se muestra en la *Figura 3.10 (*Los puntos inicial y final están denotados por color rojo*)*, de tal manera que elige el de menor gasto convirtiéndose este el nuevo punto inicial. Finalmente este proceso de elección del punto de menor distancia se repite iterativamente hasta que el punto inicial coincida con el punto final, indicando que la iteración ha terminado. Cada vez que el sistema reinicia su posición inicial, las coordenadas de dicho punto se guardan en dos vectores de n posiciones, en uno las coordenadas fila y en el otro las columnas.

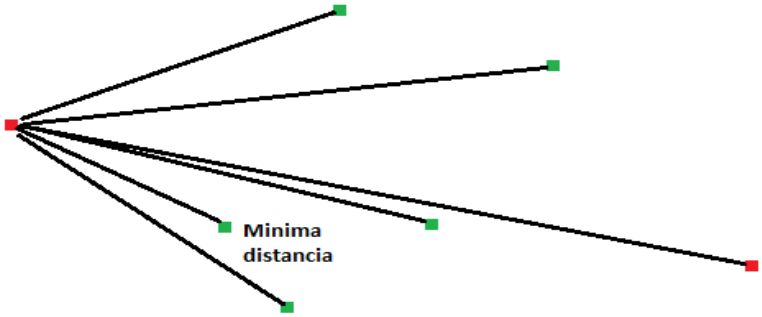


Figura 3.10. Elección de la menor distancia

- Generación del vector de trayectoria

Cuando el algoritmo de Dijkstra termina su ejecución se tiene una posible trayectoria, en la cual el algoritmo selecciona de los posibles puntos a los que pueden llegar los que estén conectados por las menores distancias, al graficar una recta entre los puntos seleccionados se obtienen resultados como por ejemplo el de la *Figura 3.11*. Este tipo de resultados se obtiene, porque el algoritmo selecciona puntos de menor distancia y no puntos de conveniencia, es decir puntos donde se de una trayectoria más óptima.

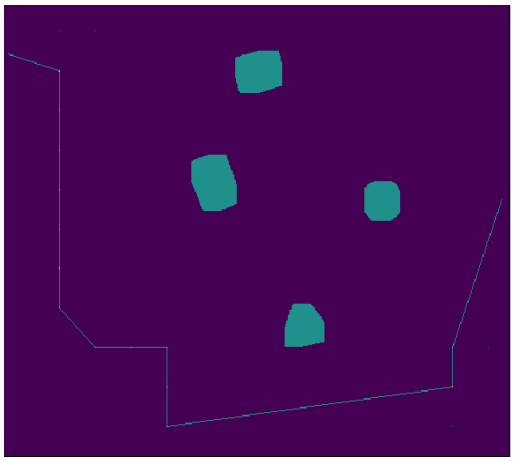


Figura 3.11. Trayectoria generada por el algoritmo de Dijkstra

3.2.4 Algoritmo de selección

Partiendo de una trayectoria base generada como por ejemplo la *Figura 3.11*, se buscó optimizar esta de la mejor manera, pues en determinados casos el camino llegaba a diferentes puntos sin sentido, lo que generaría un gasto innecesario de energía en el momento de un movimiento posterior del carro, por ende se implementó un algoritmo que mitigará este efecto y lograra llevar el auto a su destino sin necesidad de pasar por tantos puntos innecesariamente.

La funcionalidad del algoritmo es verificar desde el punto de partida si es posible unir los puntos que presentan cercanía entre ellos, es decir omitir puntos innecesarios, repitiendo iterativamente hasta encontrar el punto de llegada. En la *Figura 3.12* se representa la trayectoria generada después de pasar las coordenadas que otorga el algoritmo de Dijkstra por el algoritmo de selección, donde evidentemente es la trayectoria más óptima en este caso.

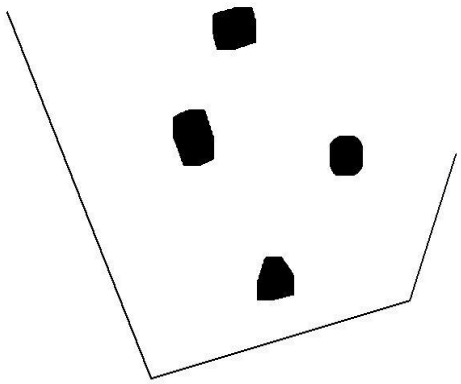


Figura 3.12. Optimización de la trayectoria generada por el algoritmo de Dijkstra

3.2.5 Distancia de seguridad

La distancia de seguridad del carro trata de un método, en el cual se evite el choque del carro con obstáculos, por ejemplo si hay una posible trayectoria pero el carro no puede pasar por motivos de espacio, el sistema no permitirá que se inicie el movimiento, de tal manera que comienza la búsqueda de un nuevo camino hasta hallar una trayectoria correcta. En la *Figura 3.13* se genera un camino por medio de dos obstáculos, aparentemente este camino es correcto y cumple el objetivo deseado por el usuario (llegar hasta el punto final), sin embargo, nótese en la *Figura*

3.14, la trayectoria generada está tocando un objeto, por ende el sistema tiene que recalcular una nueva ruta.

La distancia de seguridad se logra gracias a la dilatación de la trayectoria generada inicialmente por el computador, con un ancho máximo de treinta pixeles.

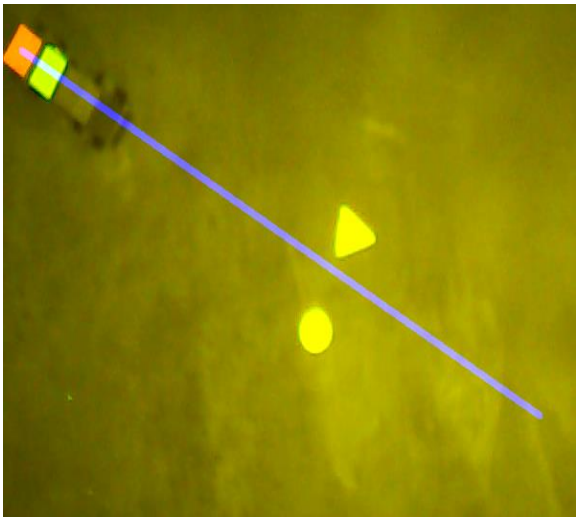


Figura 3.13 Aparente camino posible

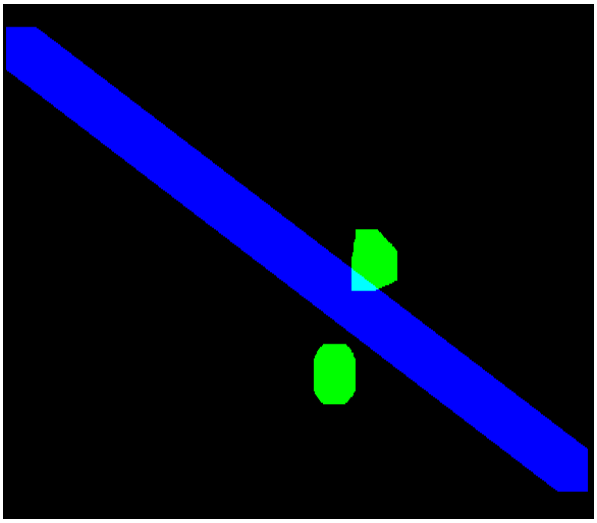


Figura 3.14 Distancia de seguridad del robot móvil

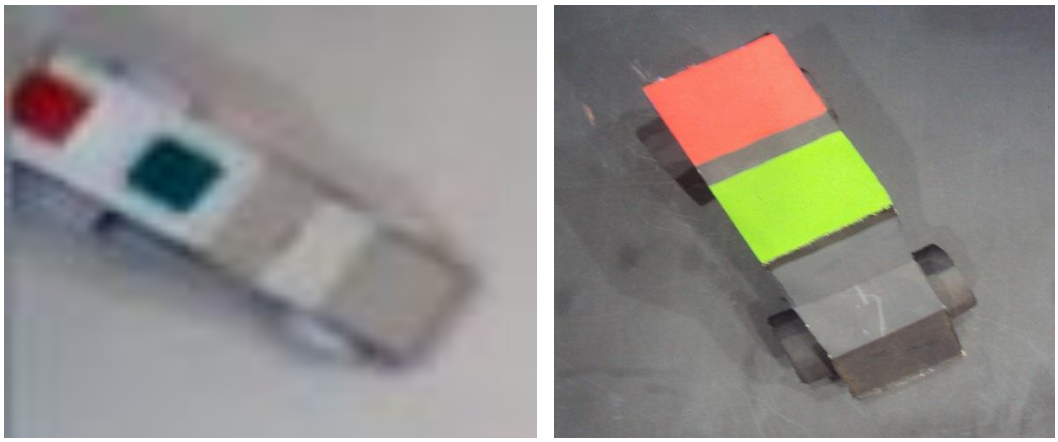
3.3 Diseño e implementación del Hardware

3.3.1 Construcción del carro

Para la elaboración del carro se tuvieron en cuenta varios aspectos, ya que el material debió tener ciertas características, por ejemplo: liviano, maleable y facilidad al pintar, por tal motivo se eligió el cartón paja. Para el diseño exterior del carro, se optó por un modelo rectangular. Esto debido a aspectos básicos como un espacio adecuado, ya que en el interior se introdujeron tres elementos, la Raspberry, la batería y el driver encargado de suministrar la potencia necesaria para que los motores se enciendan. El modelo de Raspberry utilizado fue la pi 3 model b, se escogió esta placa debido a su gran versatilidad de uso, ya que cuenta con características tales como: Un Chipset Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz, Un procesador gráfico VideoCore IV, Un módulo de 512 MB de memoria RAM, Un conector de RJ45 conectado a un integrado lan9512 -jzx de SMSC que nos proporciona conectividad a 10/100 Mbps, Salida digital de video + audio HDMI, Pines de entrada y salida de propósito general, Conector de alimentación micro USB, Lector de tarjetas SD entre otros. Además de que ya tiene módulos como el WI-Fi listo para usarse, otra de las ventajas que nos ofrece la Raspberry pi 3 model b, es que sus pines y demás módulos pueden ser utilizados desde el compilador Python, el cual ya está instalado en el sistema operativo de la Raspberry, esto fue de gran utilidad ya que para la conexión entre el computador y el robot móvil, solo se debió establecer la conexión entre dos programas Python cliente-servidor, y en la Raspberry se implementaba sin problema.

Figura 3.15. Perfil y base del carro

Inicialmente el color del carro fue el estándar dado por el cartón paja, esto aprovechando el color inicial de la base de la plataforma, es decir blanco (*Figura 3.16.a*), no obstante por mejoras en el software y obtener un buen reconocimiento de objetos, se optó por una base de color negro, por ende el carro también debió cambiar de color. La forma final de carro se expresa en la *Figura 3.16.b*, en este se ve el color final de los marcadores con pinturas mate.



a) *b)*

Figura 3.16. Robot móvil usado en el proyecto. a) Color inicial del carro. b) Color final de carro

3.3.2 Construcción de la plataforma

La plataforma fue diseñada bajo las características de una de las partes fundamentales del proyecto como lo es la cámara, tanto la altura, como el largo y el ancho de la plataforma fueron seleccionados de tal modo que la cámara tuviera una posición óptima para que la imagen fuera lo más nítida posible, además de eso abarcara un espacio suficiente para verificar que el movimiento del carro puede ser fluido, la forma de la plataforma está plasmada en la *Figura 3.17*. Está compuesta por 13 tubos de hierro, los cuales al ser ensamblados componen un prisma rectangular. En la cara superior tiene una barra diagonal, en el centro de esta barra se ubicó la cámara. La cámara usada cuenta con una resolución de 640x480 pixeles.

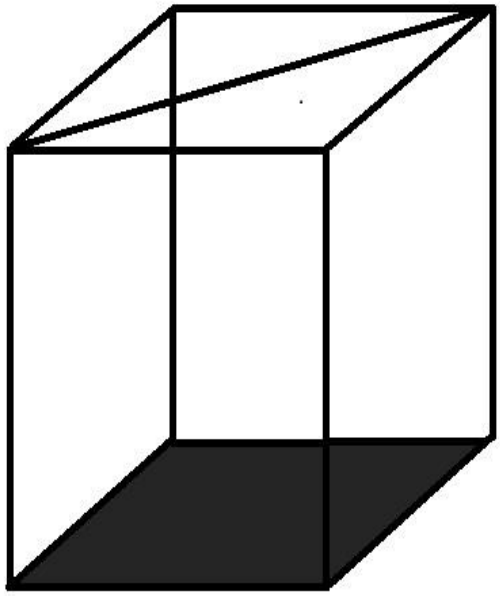


Figura 3.17. Esquema de la plataforma

Inicialmente se optó por una plataforma de un metro de alto construida con tubos de pvc y con base cuadrada de un metro. La base usada fue cartón paja, en principio se pintó blanco, como se aprecia en la *Figura 3.18.a*, sin embargo, al momento del desplazamiento del carro se notan los objetos más cerca de lo que realmente estaban, ya que la cámara usada tiene un aumento considerable, por ende se decidió ampliar el rango de visión de la cámara incrementando la altura de la plataforma. Para ello se añadieron cincuenta centímetros por todos los lados de la plataforma de tubo, no obstante esta sufrió deformaciones por el peso, lo que llevo a la construcción de una nueva plataforma de mayor resistencia y estabilidad. La plataforma final (*Figura 3.18.b*) obtenida cuenta con medidas de 1,4 metros de largo, 1,1 metros de ancho, y la altura esta con un rango de variación de 1,5 metros hasta los 1,9 metros, ya que si por algún motivo se utiliza una cámara con características diferentes es mucho más fácil adecuar el sistema.

a) *b)*

Figura 3.18. Plataformas usadas a lo largo del proyecto. a) Plataforma inicial con fondo blanco. b) Plataforma final con fondo negro

La base de la plataforma también necesito de algunas modificaciones, pues en inicio se trabajó con cartón paja, el carro se desplazaba ágilmente y llegaba hasta su destino sin problema (*Figura 3.19.a*), no obstante el color no favoreció debido a su alto grado de variación que depende en gran medida de la iluminación, generando un cambio por el color negro. Cuando se empezó a trabajar con este nuevo color en la base, la segmentación de los objetos y del carro fue buena, no obstante se notaron discrepancias de las trayectorias hechas por el carro con respecto al caso anterior. Esto sucedió porque la pintura deformo el material (cartón paja) de modo que estas imperfecciones impedían un correcto movimiento de los motores, por tal motivo se cambió el material para la base. El material final usado fue MDF, este se pintó de negro y a diferencia del cartón paja no sufrió deformación alguna (*Figura 3.19.b*).

Figura 3.19. Bases de las plataformas. a) Base inicial de color blanco. b) Base final de color negro

3.3.2.1 Condiciones lumínicas

Por motivos de estabilidad en los rangos de color que tiene tanto el carro como los obstáculos se hizo necesario tener un buen control sobre el ambiente lumínico de la plataforma, por tal motivo se hizo uso de cinco bombillos led de luz blanca de 806 Lm cada uno, estos se situaron inicialmente en cada una de las esquinas superiores de la plataforma, no obstante se notó que los objetos situados en el centro de la plataforma variaban su valor, llegando hasta el punto de desaparecer y afectar drásticamente los resultados, por ende hubo la necesidad de compensar esto con la ubicación de un quinto bombillo sobre el centro de la cara superior de la plataforma, con esto se logró una luz cuasi constante en la totalidad de la plataforma. La ubicación de los bombillos se aprecia en la *Figura 3.20*, tanto para la plataforma inicial (*Figura 3.20.a*) como para la final (*Figura 3.20.b*).

a) *b)*

Figura 3.20. Ubicación de las fuentes lumínicas. a) cuatro leds en las esquinas de la plataforma inicia. b) cinco leds en la plataforma final.

3.4 Recepción y transmisión de datos

Para lograr una constante comunicación entre el computador y la Raspberry mediante wifi, se busca un programa que permita recibir y enviar datos de posición angular del robot, con el fin de conocer el estado en que se encuentra y así iniciar la locomoción del carro. Para ello se usaran marcadores visuales por color y una comunicación TCP para la conexión.

3.4.1 Cálculo del ángulo de giro en tiempo real

Geométricamente para el cálculo de un ángulo es necesario conocer por lo menos dos puntos de interés. En el proyecto se buscó obtener dos puntos característicos de la imagen para saber el estado angular del carro, para ello se hizo uso de marcadores visuales de color rojo y verde.

Los colores se ubican sobre la parte superior del carro. El verde es el color del frente, y el rojo está en la parte trasera. El ángulo se calcula con el centro de masa de los dos marcadores, el sistema detecta la posición independiente de cada marcador y extrae la posición en el eje coordenado de los dos colores. Con esto se obtienen cuatro puntos para calcular el ángulo del robot móvil. Este cálculo es desplegado en pantalla cada vez que el sistema analice un frame.

3.4.2 Elección de marcadores visuales

Al igual que en el posicionamiento del carro se usaron marcadores de colores ubicados a 180 grados uno con respecto al otro, cabe resaltar que estos deben ser diferentes, pues el giro del robot móvil aborda los 4 ejes coordenados en un rango de 0 a 359 grados. Inicialmente se optó por marcadores de igual color, no obstante el sistema no reconocía por ejemplo cuando el carro estaba en el tercer o primer cuadrante, por ende se concluyó que debían ser diferentes. Estos marcadores fueron segmentados y sometidos a un proceso de filtrado (*Figura 3.21*).Culminado este proceso se encuentra el ángulo para cada movimiento del carro en tiempo real.

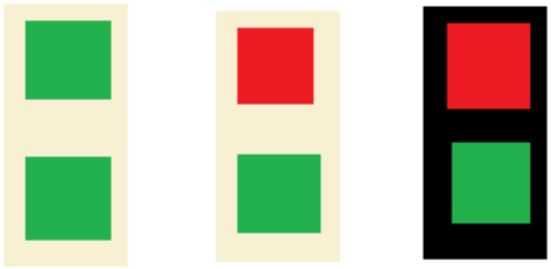
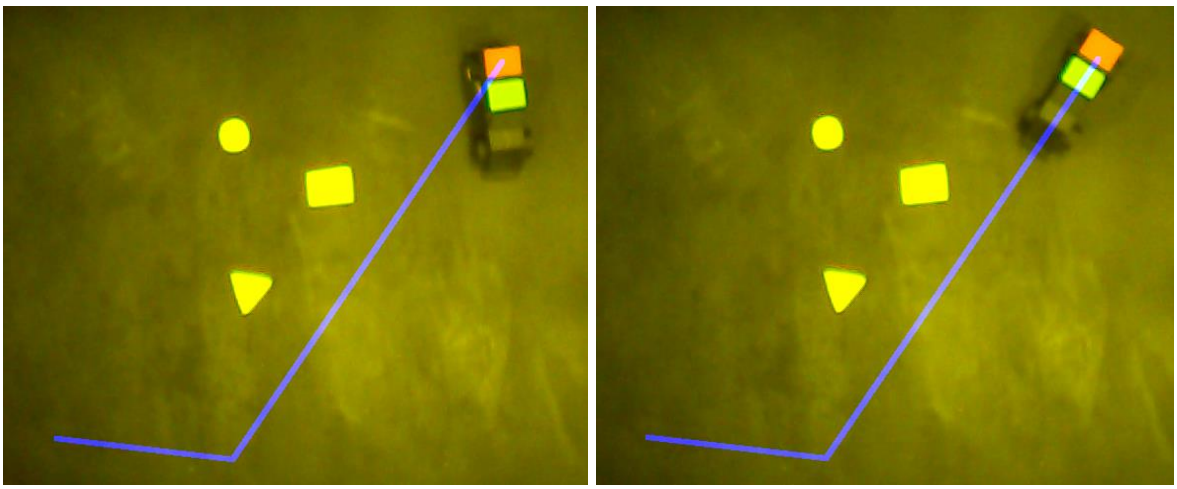


Figura 3.21. Esquema de los marcadores usados progresivamente

3.4.3 Sistema de control del robot móvil

El sistema cuenta con tres procesos de control. En dos de estos procesos se usa un sistema de control de lazo cerrado con una retroalimentación ensamblada a este. El primero es la posición angular del carro en la plataforma, pues en caso de no estar en el ángulo requerido, este se realimenta y lo vuelve a llevar a la posición angular deseada. En la *Figura 3.22.a*, el carro está inicialmente con ángulo aproximado de ochenta grados, cuando inicia el envío de datos, el sistema lo mueve hasta el ángulo deseado (*Figura 3.22.b*). Esto no solo es al iniciar el movimiento, sino que está presente durante todo el recorrido del robot móvil en la plataforma. El segundo es el reinicio del programa en caso de alguna interferencia en la trayectoria por donde va a pasar el carro. Una vez el sistema detecte un objeto encima del camino escogido este alerta al robot y lo detiene hasta recibir una nueva instrucción y reenvía los datos necesarios para iniciar un nuevo movimiento. Cabe resaltar que

no necesariamente el obstáculo debe tocar la trayectoria, basta con estar lo suficientemente cerca de esta para generar un reinicio del sistema.



a) b)

Figura 3.22. Control de ángulo del carro. a) Posición angular inicial. b) Posición angular en el ángulo requerido.

Gracias a una buena latencia en la conexión, el sistema reacciona rápido y no permite que el carro se desvíe de la posición donde debe estar, lo cual llevó a pensar en un control on-off para el posicionamiento del carro, es decir, este no tiene ninguna realimentación del sistema. En la *Figura 3.23*, está un esquema de la conexión, donde se representa el constante envío de datos entre el computador y la Raspberry.

Figura 3.23. Esquema general de Comunicación inalámbrica computador-Raspberry

Al contar con una conexión adecuada para él envió constante de datos entre el computador y la Raspberry, se inicia con las pruebas de locomoción del robot mobil tanto en lazo abierto como en lazo cerrado. Esto con el fin de analizar el comportamiento del carro ante estos tipos de sistemas y establecer cuáles son las ventajas y desventajas.

3.4.4 Diagrama de flujo del sistema

En la Figura 3.24 se presenta el diagrama de procesos ejecutados por el sistema. Inicia con la captura de la imagen panoramica cenital del entorno de movimiento, la cual tendrá un posterior procesamiento; limitar la selección del punto de destino a una zona que no salga del rango de visión de la cámara. Una vez el usuario seleccione este punto final, comienza el procesamiento de la imagen. Esto incluye segmentación de objetos y del robot, con el fin de lograr su identificación. Con ello se generan las instrucciones necesarias para guiar el robot móvil hasta su destino.

Una vez extraídas dichas instrucciones el sistema procede a conectarse inalámbricamente con el robot móvil y transferirle dicha información e iniciar su movimiento. Cuando inicia el movimiento, el sistema está verificando constantemente si algún objeto obstaculiza el trazado guía del robot, de ser así, el

sistema detiene su ejecución mientras se calcula una nueva trayectoria, por el contrario, si ningún objeto interfiere, en la pantalla se puede seguir el proceso de locomoción del robot móvil hasta alcanzar su punto de destino.

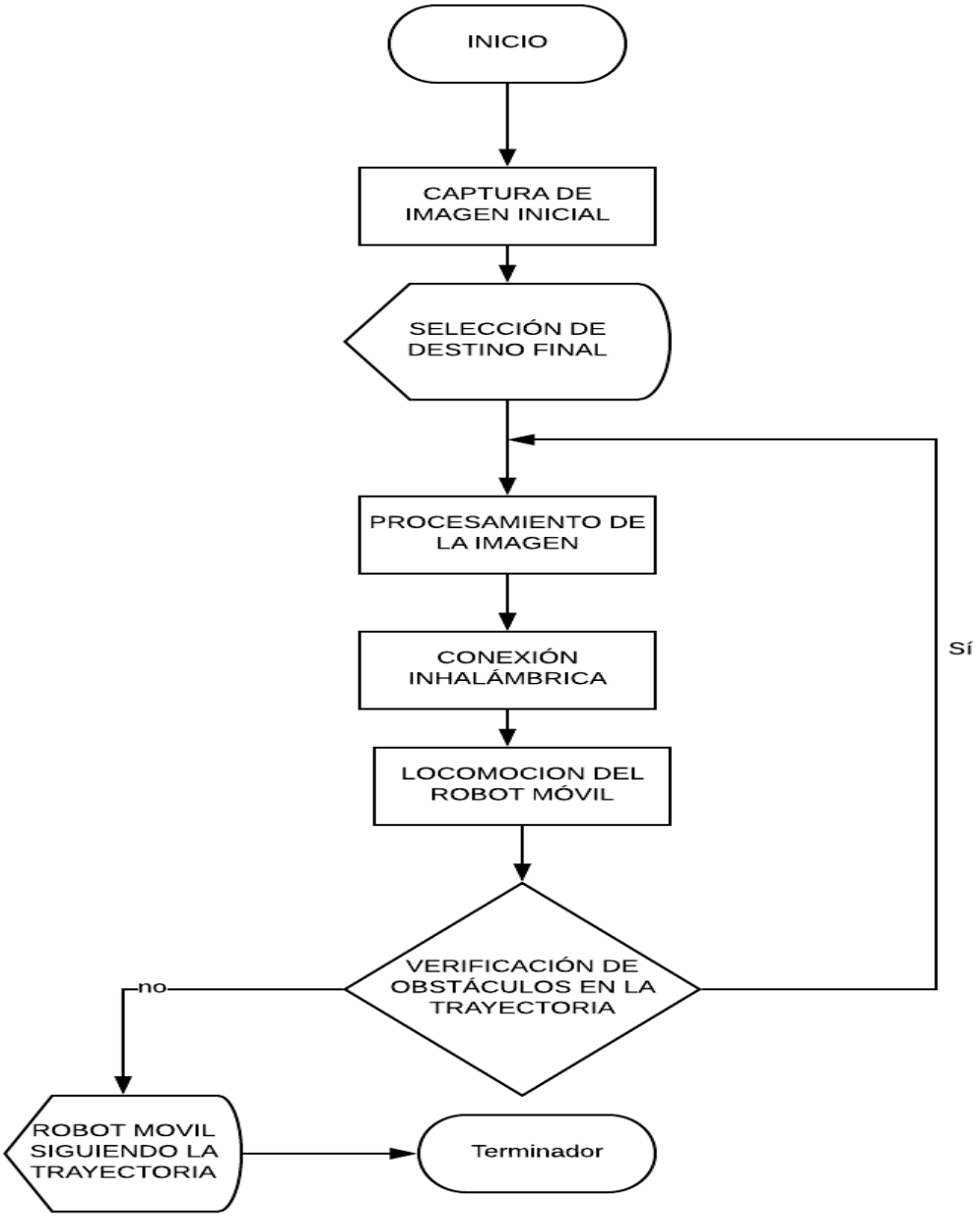


Figura 3.24 Diagrama de procesos del sistema

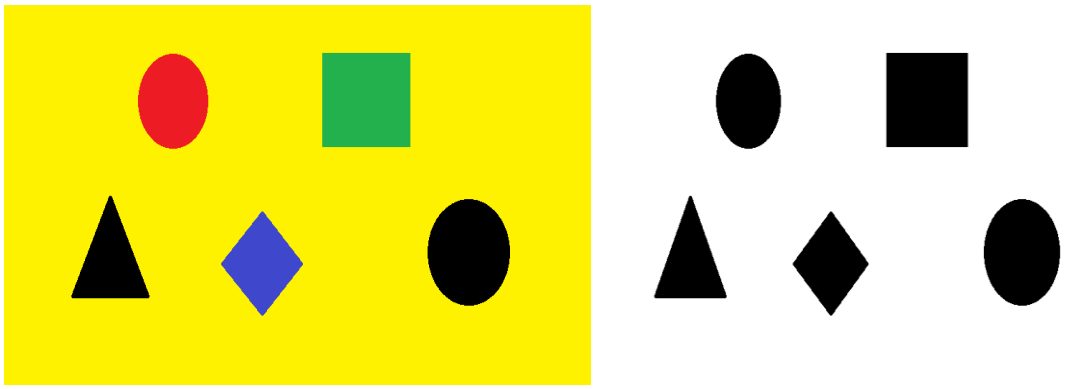
En el capítulo descrito a continuación se evidenciará todo el proceso usado para alcanzar el objetivo general del proyecto, se muestran resultados explícitos de la segmentación de obstáculos en diferentes programas de programación, se mostrara los tres programas bases usados para la generación de una trayectoria y finalmente se muestra las pruebas de locomoción del robot móvil.

CAPITULO 4

RESULTADOS

4.1 Resultados del reconocimiento de obstáculos

Como se describió en el capítulo 3 para el reconocimiento de obstáculos se empleó la técnica de segmentación. Inicialmente se trabajaron con imágenes de Paint para simular el ambiente de trabajo, en esta se presenta un fondo uniforme de color amarillo y los objetos de distintos colores y formas (*Figura 4.1a*), con su respectiva segmentación (*Figura 4.1b*).



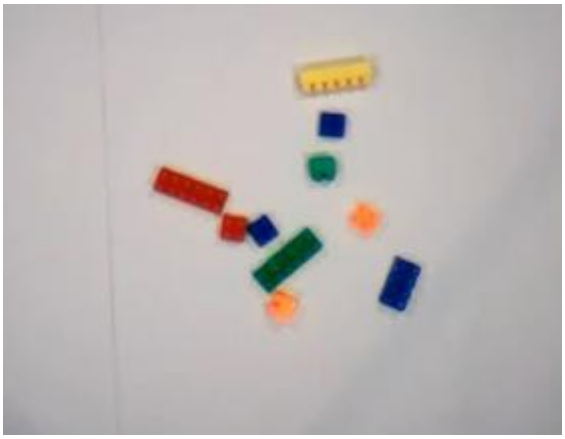
a) *b)*

Figura 4.1 Imagen generada desde el software Paint a) Imagen con diferentes obstáculos y colores b) Segmentación de la imagen

Cuando se hacen diferentes pruebas para la segmentación con imágenes creadas desde Paint se evidencia el correcto funcionamiento del programa, por ende se inicia con el trabajo de imágenes reales desde el software Python y la librería OpenCv para el procesamiento de imágenes.

Para simular imágenes como las anteriormente descritas, se decide trabajar con fichas lego de diferentes colores en un fondo uniforme de color blanco (*Figura 4.2a*). La ventaja de trabajar con estos objetos fue permitirle a la cámara caracterizar los

parámetros de segmentación en el espacio RGB, no obstante en diferentes horarios del día se presentaron problema por carencia de buena iluminación y el brillo de dichos objetos. La segmentación de estos se evidencia en la *Figura 4.2b*, donde se aprecia la distinción entre el fondo y los todos los objetos.



a)

b)

Figura 4.2 Imagen usando fichas lego de diferentes colores. a) Fichas lego en un fondo uniforme de color blanco. b) Segmentación de fichas lego.

Debido al ruido presentado por las imágenes segmentadas se usaron filtros que permitieron eliminar estas zonas parasitas de los objetos de interés, ya que se convertiría en algo problemático para posteriores análisis.

Como se describió en el capítulo 3, los colores finalmente usados para la base de la plataforma fue el negro y para los objetos el blanco.

4.2 Resultados para la generación de trayectorias

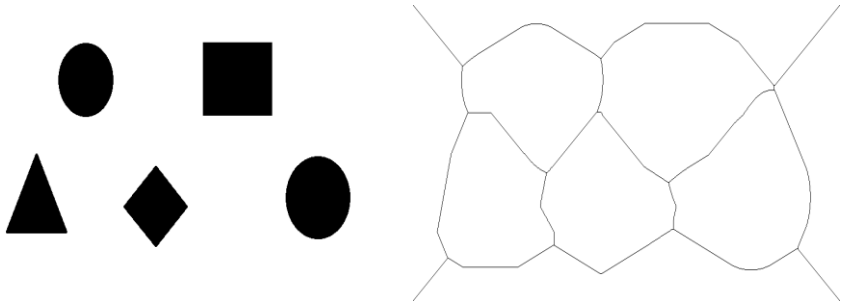
Para obtención de trayectorias se usaron tres métodos diferentes. Dos de estos procesos se llevaron hasta la etapa final de enlace computador-Raspberry, pues el restante demandaba un tiempo de compilación amplio y fue deshecho para posteriores análisis.

4.2.1 Algoritmo de esqueleto

- Algoritmo de esqueleto en Matlab

Para obtener una trayectoria a partir de este algoritmo se inicia trabajando en la plataforma Matlab para procesamiento de imágenes. En este proceso se generan todos los posibles caminos que el robot puede ejecutar hasta su destino. Cabe aclarar que en los inicios del proyecto se buscó encontrar un camino óptimo desde la esquina superior izquierda de la imagen hasta la esquina inferior derecha, es decir atravesar la plataforma, por ende la trayectoria seleccionada tendrá esta configuración.

En la *Figura 4.3b* se muestra el diagrama de las trayectorias obtenidas al aplicar el algoritmo en una imagen segmentada (*Figura 4.1a*) y creada desde Paint. Para cada obstáculo presente en la imagen se da una trayectoria que lo rodea, lo cual es una gran ventaja cuando se trata de obtener una ruta. Sin embargo una vez finalizado el proceso de selección de una trayectoria.



a) *b)*

Figura 4.3 Esqueleto de en Matlab a) Objetos de Paint segmentados. b) Esqueleto del fondo de la imagen.

Con todas las posibles trayectorias por donde se podría desplazar el robot móvil, se indaga acerca de un método de selección de camino, el cual permita extraer una trayectoria correcta hasta el punto final (Esquina inferior derecha de la imagen). Con ello se pensó en los puntos de intersección conocidos también como bifurcaciones, que es la unión de tres o más pixeles con un pixel común (*Figura 4.4*). En la imagen el pixel común es representado de color rojo y los pixeles vencidos de color verde.

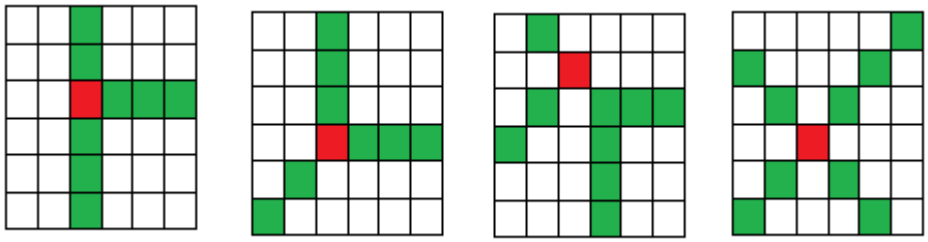


Figura 4.4 Algunas configuraciones de pixeles para hallar bifurcaciones

Para encontrar los pixeles comunes de la *Figura 4.3b*, se elabora un código el cual extrae la posición de la fila y columna de cada punto y se guardan en dos vectores. Estos puntos están representados en la *Figura 4.5.*

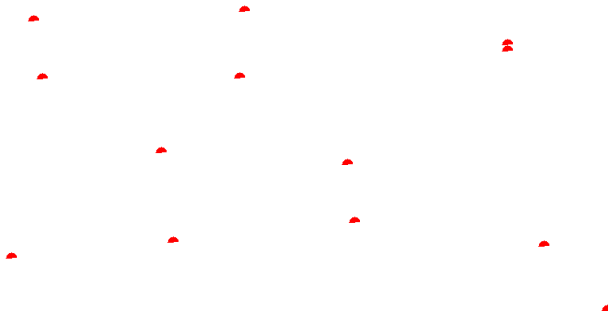


Figura 4.5 Bifurcaciones del esqueleto de la imagen

Con las bifurcaciones de la imagen obtenida se inicia el algoritmo de selección de trayectoria basado en el método de Dijkstra. Este algoritmo encuentra el mínimo camino dada una serie de puntos, en este caso bifurcaciones. Cuando se encuentra una ruta, el sistema compara esta con el esqueleto de la *Figura 4.1b* y hace una superposición. *En la Figura 4.6*, la línea roja indica el camino obtenido por el algoritmo de Dijkstra.

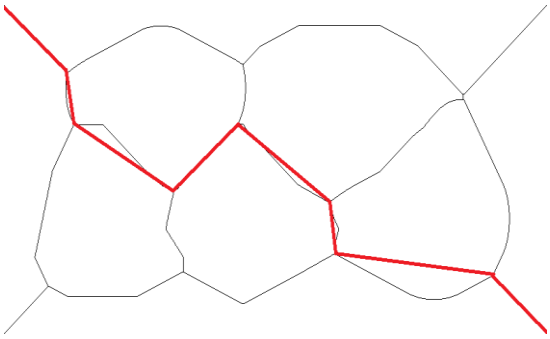


Figura 4.6 Superposición de caminos

El objetivo de la superposición de caminos fue establecer una referencia para el sistema; nótese que la línea roja se divide en ocho partes para llegar hasta el punto final, cada una de estas partes se evalúa por separado con el fin de encontrar que porción del esqueleto se asemeja a dicha ruta. Cuando se culmina el análisis en toda la imagen se obtiene la trayectoria plasmada en la *Figura 4.7*.

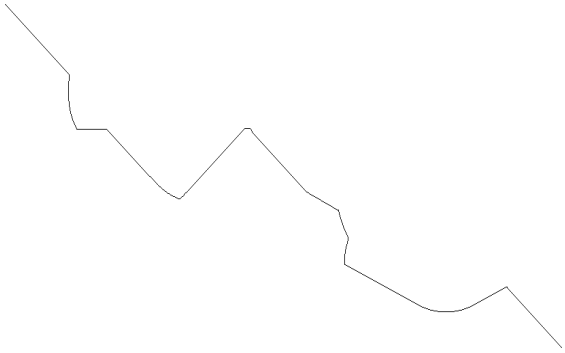
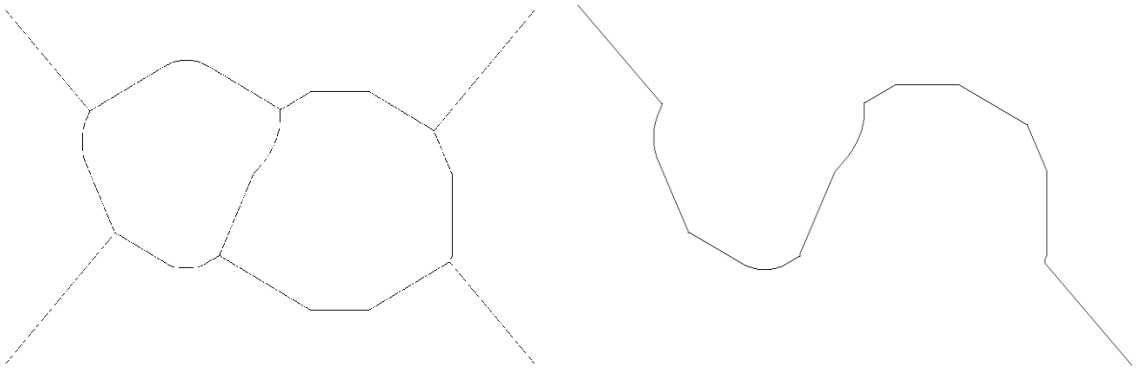


Figura 4.7 Trayectoria obtenida en Matlab

Las trayectorias obtenidas por este método en la plataforma Matlab presentaron buenos resultados, no obstante en ocasiones se daban algunas inconsistencias en la generación de caminos, lo que se convertiría en una problemática en posteriores análisis. Una de estas inconsistencias fue la elección del mínimo camino dada una serie de puntos. En la *Figura 4.8a* se muestra el esqueleto obtenido en una configuración de obstáculos. En la *Figura 4.8b*, la trayectoria seleccionada por el sistema como mínimo camino.



a) *b)*

Figura 4.8 Inconsistencia de trayectoria en Matlab. a) Todos los caminos generados. b) Trayectoria elejida por el sistema

En la *Figura 4.8b* se evidencia una inconsistencia del sistema para hallar un camino al robot movil, ya que la ruta generada es claramente la menos optima y segura. Una ruta correcta para la configuracion mostrada en la *Figura 4.6a*, se representa en la *Figura 4.9*, donde evidentemente es la mejor opcion para llegar hasta la esquina inferior derecha de la imagen.

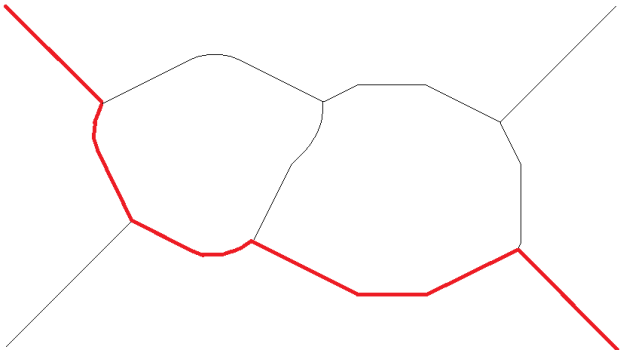


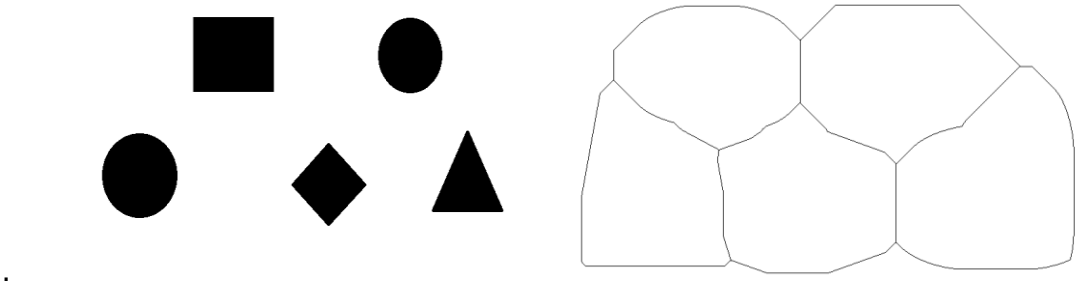
Figura 4.9 Ruta teoricamente correcta

Otra problemática presentada en el programa es el tiempo de compilación de aproximadamente 78.9 s, lo que imposibilitaría un trabajo con video, por tal motivo se inició la búsqueda de un nuevo software que mitigara estos efectos.

- Algoritmo de esqueleto en Python y OpenCv

Con el software de programación Python y la librería OpenCv al igual que el caso anterior, se inicia trabajando con imágenes de Paint, con el fin de establecer una

comparativa con la plataforma usada anteriormente. Se inicia trascribiendo el código fuente de Matlab a Python y comienza la búsqueda de un método para obtener el esqueleto de una imagen. Este fue un proceso tedioso, pues la librería OpenCv no cuenta con un algoritmo para obtener el esqueleto de una imagen, por tanto se optó por buscar otras librerías como Scikit-image que si cuenta con esta opción. Los resultados obtenidos al aplicar el algoritmo sobre una imagen segmentada (*Figura 4.10a*) se muestra en la *Figura 4.10b.*



.

a) b)

Figura 4.10 Esqueleto usando Scikit-Image a) Detección de los obstáculos b) Trayectorias posibles obtenidas mediante el esqueleto de la imagen

De la *Figura 4.8b* es posible extraer seis puntos característicos o bifurcaciones. Una desventaja en comparación con los puntos obtenidos a partir de la *Figura 4.1b* son los puntos de las cuatro esquinas, lo que disminuye la probabilidad de encontrar un camino hasta el punto final. Para asemejar el comportamiento del programa Python al creado en Matlab, se decide forzar al programa a encontrar los puntos de las esquinas por medio de una imagen con objetos agrados manualmente, representados en la *Figura 4.11*.

Figura 4.11 Imagen con objetos virtuales

Con los objetos virtuales agregados manualmente a una imagen Paint, se suma esta imagen a la trabajada por el programa para hallar el esqueleto (*Figura 4.12*).

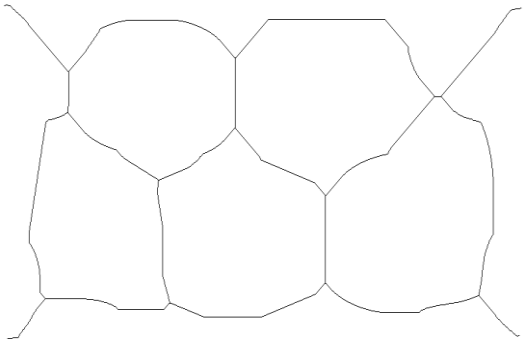


Figura 4.12 Trayectorias mejoradas en Python

La *Figura 4.12* es muy parecida a la generada en Matlab (*Figura 4.1b*), sin embargo difiere en la total intersección con las esquinas de la imagen, lo que en un inicio fue un problema, pues se requería que estuviese tocando estos bordes, pues para un posterior análisis la trayectoria quedaría cortada. Debido a la falta de herramientas en el filtrado de una trayectoria como la obtenida en Matlab, se estableció un nuevo criterio de generación de caminos; Solo se tendría en cuenta el obtenido por el algoritmo de Dijkstra. *En la Figura 4.13*, se muestra un ejemplo de un camino, donde todos los objetos están etiquetados por diferentes colores.

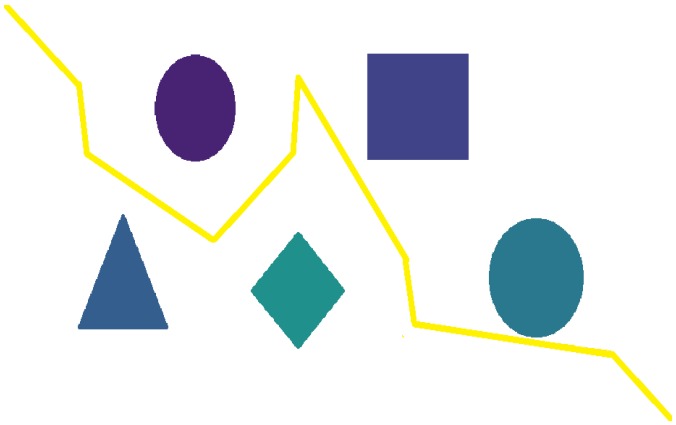
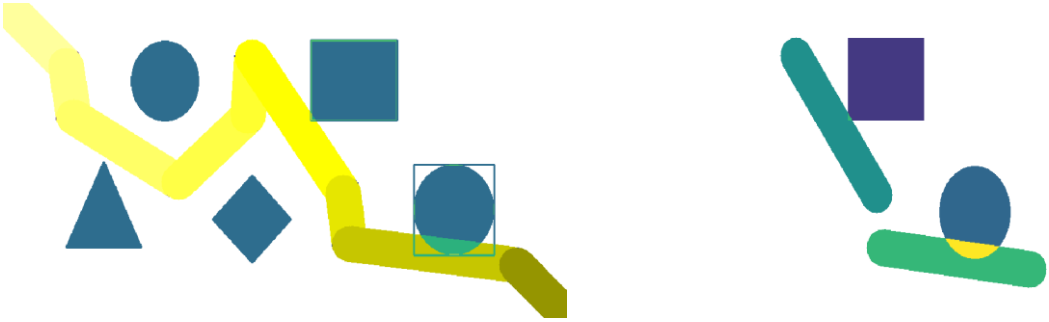


Figura 4.13 Camino obtenido desde Python

En la imagen se puede ver que a pesar de tener una trayectoria hasta el punto de destino, un segmento de camino pasa muy cerca de un obstáculo, lo que no sería muy útil cuando se inicie con la locomoción del robot móvil, por ende se vio en la necesidad de crear una distancia de seguridad; descrita con detalle en el capítulo 3

Con la distancia de seguridad plasmada en el código, se genera el mismo camino, pero esta vez claramente se evidencia la intersección de la recta con dos obstáculos (*Figura 4.14a*) de la imagen. Para suplir esta necesidad se etiqueto por color el trayecto generado y se seleccionó los puntos o zonas de análisis (*Figura 4.14b*), con el fin de mover este tramo según lo requiera el caso.



a) *b)*

Figura 4.14. Etiqueta de objetos y trayectoria. a) Detección de objetos tocando la trayectoria. b) Selección de tramos para análisis.

Con los objetos que tocan la trayectoria seleccionados se procedió a elaborar el código que permita mover los trayectos, de tal forma que el camino sea correcto. El trayecto mejorado está representado en la *Figura 4.15a*. Este camino tiene un ancho

de cincuenta pixeles y se evidencia la mejora, pues ningún obstáculo intersecta con este. En la *Figura 4.15b* está plasmado, el mismo trayecto, pero con un grosor de un pixel.

a) *b)*

Figura 4.15 Trayecto python mejorado. a) Trayecto con un grosor de 50 pixles. b) Trayecto con un grosor de 1 pixel.

Al trabajar siempre con la misma imagen en la busqueda de un metodo para encontrar un camino optimo y seguro para el robot movil, este se hizo muy particular, y al insertar nuevas configuraciones de objetos, el sistema fallaba en ocaciones, por tal motivo se hizo un nuevo programa.Este codigo incluye una nueva configuacion de objetos virtuales, con el fin de obtener mas puntos de interes(*Figura 4.16*).

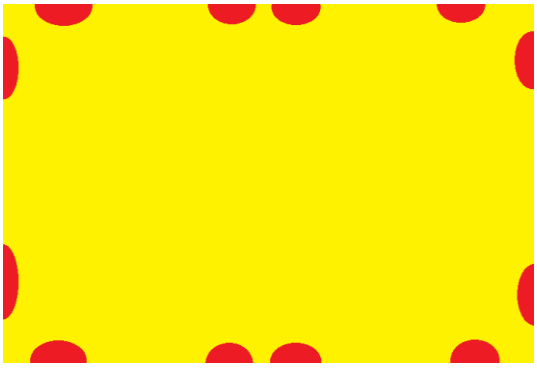


Figura 4.16 Nueva configuracion de objetos virtuales

El funcionamiento de programa trata de dilatar iterativamente los obstaculos e increnta las posibilidades de encontrar un camino (*Figura 4.17*). A continuacion algunos de los resultados obtenidos.

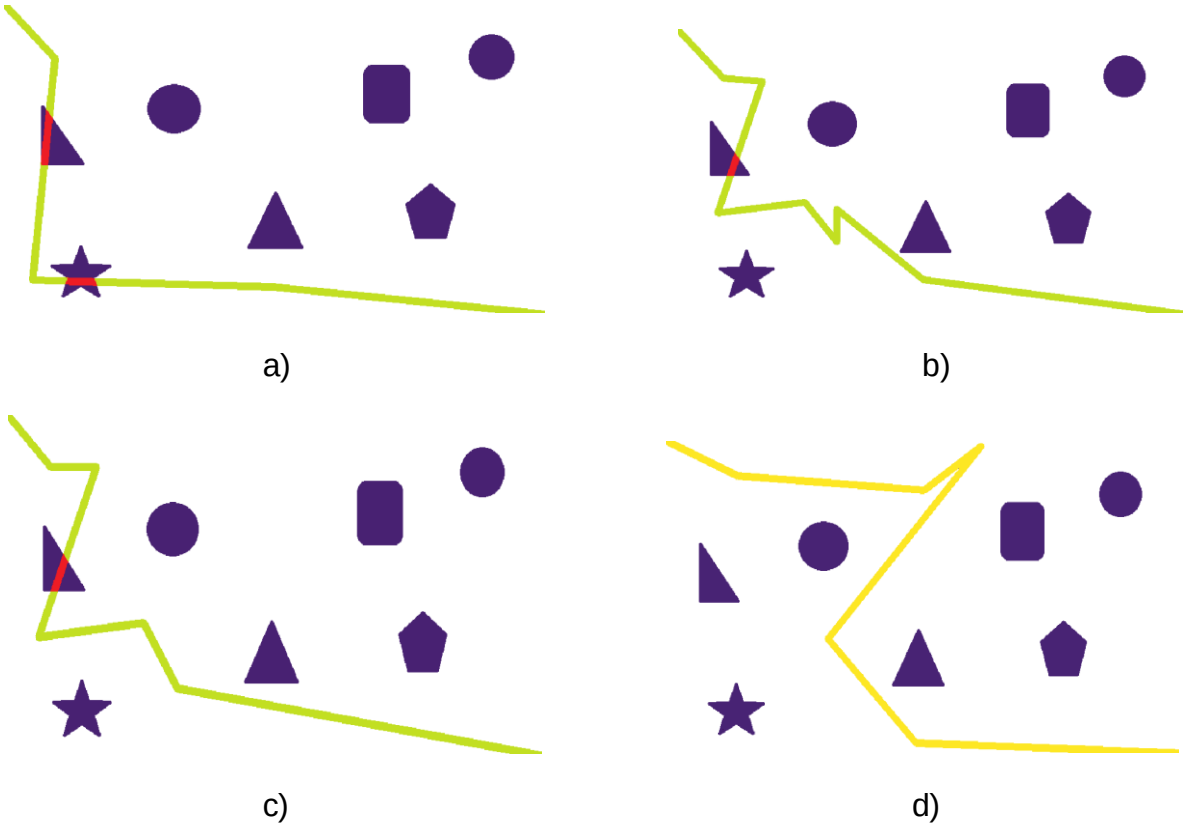


Figura 4.17 Caminos encontrados a partir de dilatación de objetos

La *Figura 4.17d* muestra el único camino seguro pero no optimizado hasta el destino requerido, por ende se buscó no solo optimizar caminos generados a partir del algoritmo de Dijkstra, sino que se inició el trabajo con imagenes reales. En estas imágenes se elimina la limitante del punto de destino, pues el usuario es quien indica donde desea que llegue el robot móvil. Para la configuración de objetos reales de color blanco en la plataforma de fondo negro (*Figura 4.18*), se tendrán una serie de caminos a partir del esqueleto generado.

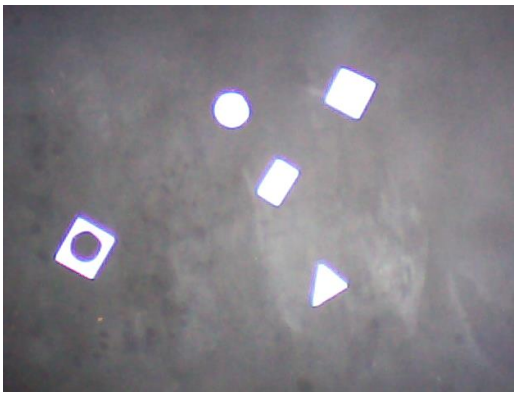
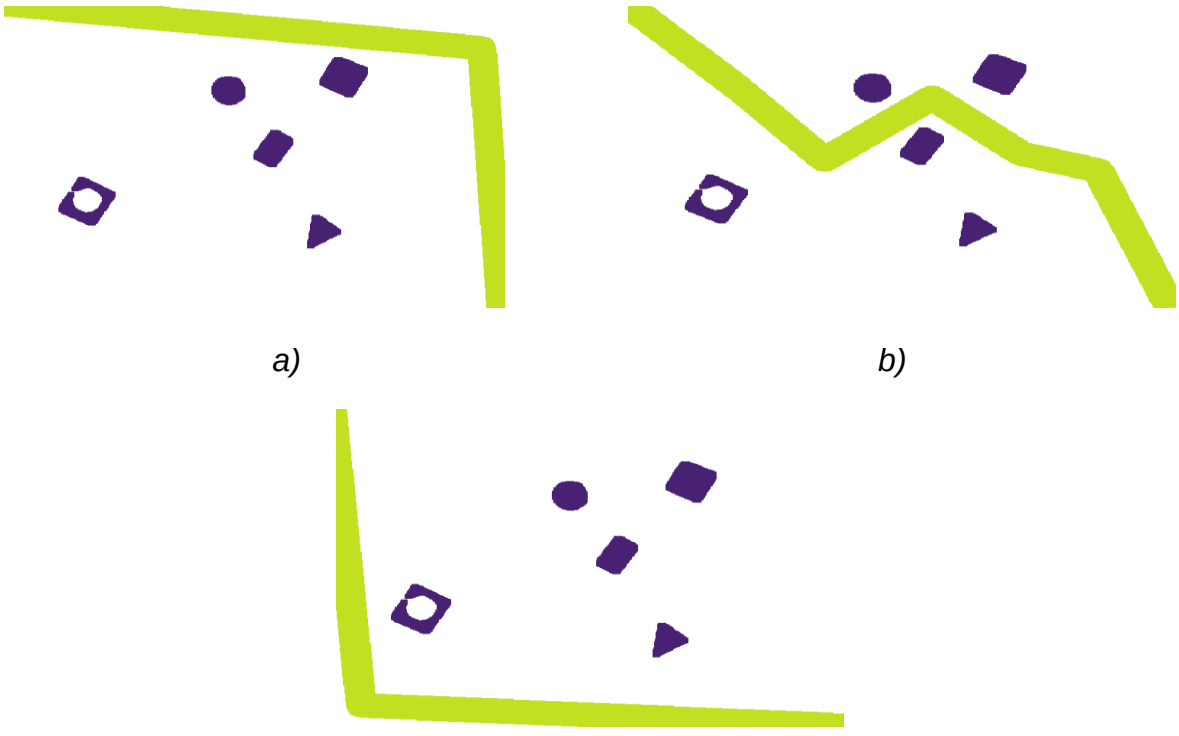


Figura 4.18 Objetos reales de color blanco y fondo negro

Algunos caminos generados por el algoritmo de esqueleto para la configuración mencionada anteriormente, se plasma en la *Figura 4.19*, donde en los tres casos la trayectoria obtenida es correcta, por ende se procede a la elección de una de estas bajo el principio de mínimo camino o menor gasto energético.



a) *b)*

c)

Figura 4.19 Posibles trayectorias generadas a partir de una configuración de objetos reales

Visualmente se puede identificar que las trayectorias a y c son muy similares. Para establecer cuál de esas configuraciones es el menor camino, el sistema compara el área en pixeles cuadrados y selecciona la de menor valor, en este caso la *Figura 4.19c.*

En el algoritmo de esqueleto se obtuvieron resultados buenos en cuanto a la generación de trayectorias para el robot móvil, no obstante por tiempo de compilación y por ende en el procesamiento de video el algoritmo no se comportó de la misma manera, pues al analizar tantos frames el sistema se saturaba, lo que llevó a un descarte definitivo del método. Este es un algoritmo muy atractivo para la generación de trayectorias, pues siempre se muestran varios caminos por donde el carro se pudiese llegar a desplazar, lo que de no haber tenido un coste computacionalmente elevado seria la opción perfecta.

Como anteriormente se mencionó, este proceso no se llevó a la etapa final de conexión y transmisión de datos, pues en caso de algún movimiento de un objeto cerca de la zona de la trayectoria generada, es decir, un objeto encima o cerca a la trayectoria, el carro se chocaría con este antes de esperar alguna respuesta del sistema, por ende comenzó la búsqueda de nuevos algoritmos.

4.2.2 Algoritmo de puntos característicos

Una vez obtenida la imagen binarizada de los obstáculos, el funcionamiento del algoritmo de puntos característicos se basa en la identificación del centro de masa de cada uno de los objetos presentes en la imagen como se puede apreciar en la *Figura 4.20*.

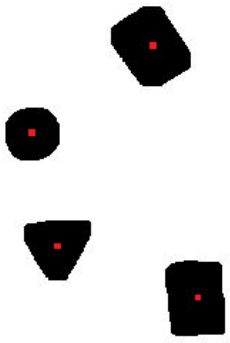
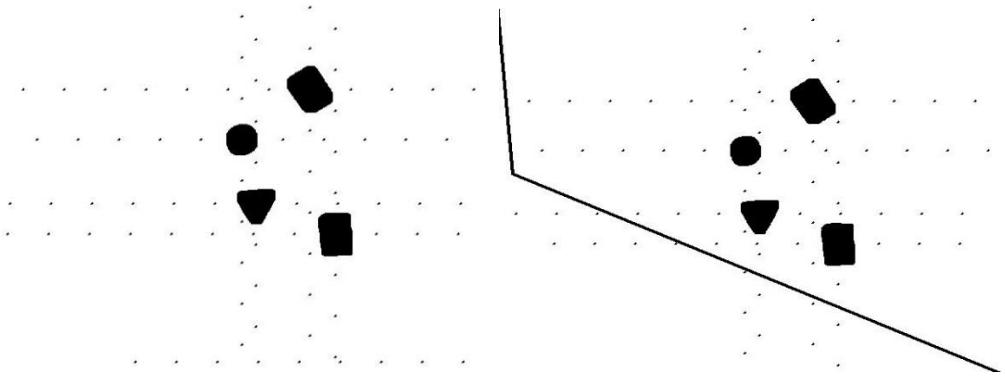


Figura 4.20. Centros de masa de los obstáculos

Posterior a la identificación de las coordenadas de dicha característica, el algoritmo inicia una serie de iteraciones que se basan en alejar periódicamente puntos en las direcciones cardinales, estos puntos están a una distancia máxima de 500 pixeles a través de 10 iteraciones de 50 pixeles (ver *Figura 4.21.a*). Al finalizar cada iteración el sistema procesaba la imagen resultante mediante el algoritmo de Dijkstra y después por el algoritmo de mejoramiento de trayectoria se obtenía una trayectoria que generalmente estaba alejada de los obstáculos (ver *Figura 4.21.b*), aunque presentaba algunas inconsistencias dependiendo de la posición de los mismos.



a) b)

Figura 4.21 Procesos del método de puntos a) Puntos generados a partir las coordenadas del centro de masa de los obstáculos b) Trayectoria final obtenida mediante el procesamiento de los puntos generados

El principal problema encontrado al utilizar este método es cuando un objeto está en una posición cercana a la de algún otro obstáculo, los puntos generados podrían quedar tan cerca de los obstáculos que en el momento en que el móvil siguiera la trayectoria los bordes podrían tocar estos.

Aunque la mejora en tiempo de ejecución fue bastante grande en comparación con el algoritmo de esqueletización, el algoritmo de puntos característicos no logro el requerimiento esperados para la correcta ejecución del movimiento, pero sentó bases que fueron de gran ayuda para el código final de ejecución que se usó en el proyecto.

4.2.3 Algoritmo celda de puntos

Como se describió en el capítulo 3, este fue el algoritmo usado en el proyecto, no solo por la velocidad en los cálculos de trayectorias, sino por la mayor eficiencia en el momento de obtener una nueva partiendo de la posición inicial del carro y llegando de forma segura hasta su destino final. Este algoritmo no tuvo ningún inconveniente en el procesamiento de video. En caso de que algún objeto tocara la trayectoria a seguir del carro, es sistema responde oportunamente y evita que este choque, de tal forma que el robot móvil permanece en estado de reposo hasta que el sistema se reinicie y recalcule una nueva trayectoria.

En el inicio del programa se despliega una imagen bordeada con un recuadro azul, esta imagen permite al usuario elegir el destino del carro, ya que inicialmente esté solo se movía desde su posición inicial hasta la esquina inferior derecha de la plataforma. En la *Figura 4.22* se muestra este recuadro y el punto donde debe dirigirse el carro.

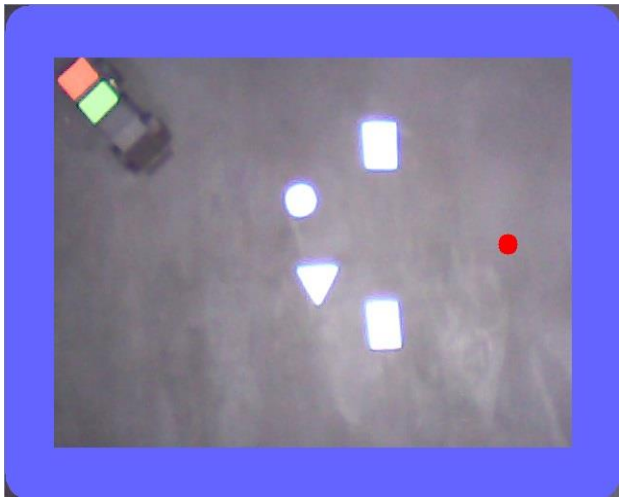
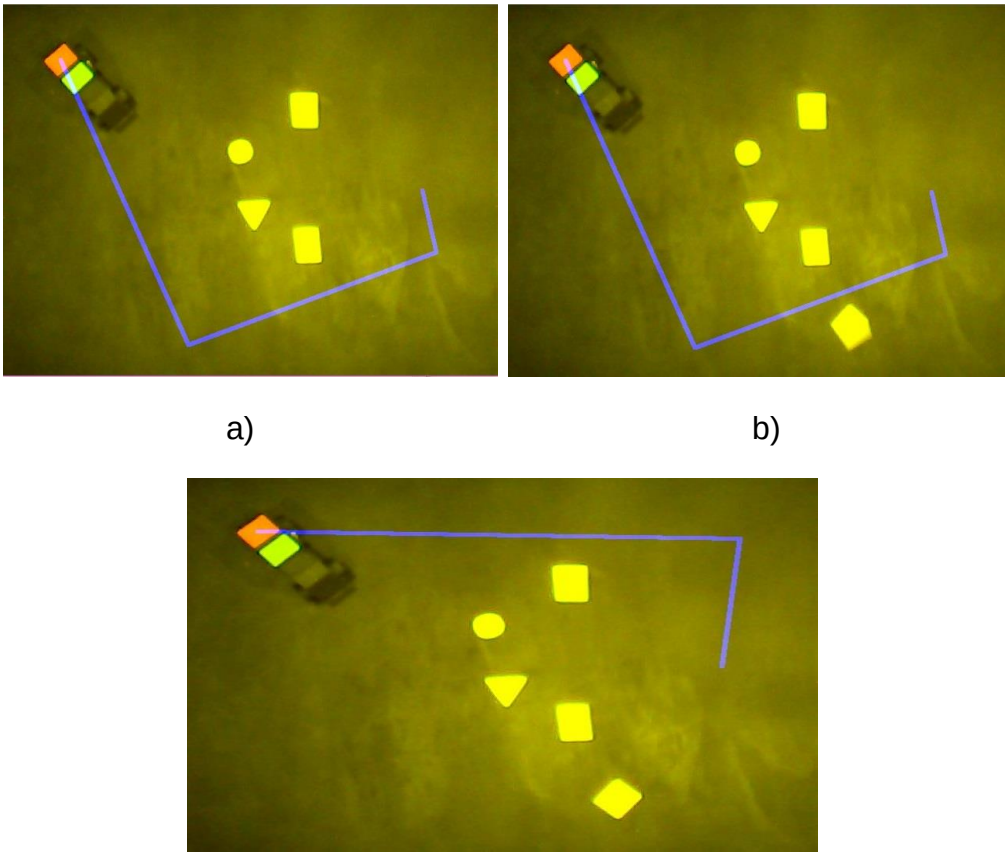


Figura 4.22 Interfaz de selección del punto final

El objetivo de insertar un marco sobre la imagen es evitar que el carro se salga del marco de visión de la cámara, es decir, si no hubiese ninguna limitación puede que el carro llegue a un punto donde los marcadores de color se pierdan, afectando drásticamente en el movimiento, ya que no habrá forma de volver a entrar los límites aceptados, pues no puede enviar un ángulo para iniciar algún movimiento.

En la *Figura 4.6.a* se muestra una primera trayectoria obtenida por el sistema, donde la línea azul representa por donde se debe mover el robot móvil para llegar hasta el final. Si un objeto tocase el trazado guía, como se aprecia en la *Figura 4.23.b*, el sistema se detiene hasta que se genere un nuevo camino (*Figura 4.23.c*). Cabe aclarar que el sistema funciona de forma adecuada aun en el caso en el que el obstáculo interfiera en la trayectoria una vez iniciado el proceso de locomoción del robot, en este caso el computador envía un comando para que el carro se detenga hasta encontrar una nueva ruta. Una vez encontrada se procede al envío de las nuevas instrucciones. Este método permitió establecer una excelente conexión entre el computador y Raspberry, pues al trabajar con video el sistema no se satura y se da un acertado envío de datos.



a) b)

c)

Figura 4.23 Corrección de la trayectoria a) Trayectoria generada inicialmente b) Muestra que un objeto interfiere en la trayectoria c) Nueva trayectoria generada por el sistema

4.2.4 Comparación de tiempo de ejecución de algoritmos

Para generar trayectorias a partir de una configuración de obstáculos, los tres algoritmos anteriormente descritos funcionaron de una forma adecuada, no obstante se decide descartar dos de estos. Uno de los criterios usados para omitir los algoritmos fue el tiempo de compilación para encontrar un camino óptimo y seguro hasta el destino. En la tabla 4.1, se muestra el tiempo de compilación para cada algoritmo.

Tabla 4.1 Tiempo de compilación de algoritmos para generación de trayectorias

Método	Tiempo(s)
Esqueleto	18.55
Puntos característicos	2.23
Celda de puntos	2.35

En la tabla se puede apreciar que claramente el algoritmo de esqueleto es descartado por su amplio tiempo de compilación, sin embargo el algoritmo de puntos característicos presenta menor tiempo que el de celda de puntos. El principio usado para elegir uno de los dos algoritmos fue la versatilidad de cada sistema para hallar una trayectoria, pues el algoritmo celda de puntos por tener más opciones de análisis, presenta una mayor probabilidad de hallar una ruta.

Cuando se logra afianzar un código que permita obtener trayectorias, tanto en condiciones estáticas como dinámicas, se inicia el proceso de conexión inalámbrica entre el robot móvil y el computador.

4.3 Comunicación y control

Para el proceso de transmisión de datos se optó por utilizar el protocolo TCP mediante la configuración de servidor y cliente, esta fue la opción final utilizada, aunque no fue la primera. Inicialmente se usó una herramienta llamada Xampp (Apache Friends) que es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases, la cual crea un servidor local en el equipo y permite compartir los archivos almacenados en una carpeta en particular, se pensó en esta opción dado que a dicha carpeta compartida se podía acceder desde un navegador web. El principal inconveniente que se tuvo al implementar este método fue la vigilancia de debía hacer el sistema sobre la base de datos creada en determinada dirección IP e identificar si en esa ruta hubo algún cambio, esto tuvo grandes complicaciones, pues para lograr tener un sistema en configuración de lazo

cerrado este debe estar constantemente enviando y recibiendo datos, por lo que esta tarea se hacía cada vez más compleja, ya que los datos debían ser guardados en archivos de texto plano antes de ser enviados.
Por tal motivo se busca una alternativa más eficiente en cuanto al envío y recepción de datos que permitiese tener una latencia menor, dado que, si el sistema tarda en responder, el robot móvil colisionaría si se agrega o se mueve un obstáculo cerca de la posición del mismo. Con esto se llegó a la implementación del protocolo de conexión TCP entre archivos compilados desde Python, esto permitió una conexión casi inmediata entre el servidor y el cliente, que en este caso es el computador y la Raspberry.

4.3.1 Locomoción del robot móvil

El sistema de locomoción del móvil pasó por dos etapas.
Inicialmente se pensó en convertir las características del trazado creado por el sistema en un vector que se iría llenando de acuerdo a la conectividad que tuviera un pixel inicial con sus vecinos, y de acuerdo al valor que tuviera el vector en cada posición el robot móvil debería interpretar dicho valor para realizar una acción determinada.
A continuación, se muestra el proceso de extracción de características. Una línea está formada por una secuencia de pixeles que para el caso del ejemplo están pintados de color negro. Al aumentar la línea a gran escala se pueden distinguir cada uno de los pixeles que la conforman (*Figura 4.24*) Como se puede ver en la imagen cada pixel (a excepción de los del borde) tienen ocho posibles lugares en los que puede estar su pixel vecino.

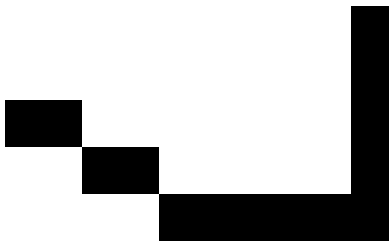


Figura 4.24 Conectividad entre pixeles vecinos

Para encontrar un patrón de movimiento para del robot, se usó una matriz de 3x3, con numeración de 1 hasta 8. Cada número indica la orientación que debe seguir el robot para llegar a su destino (*Figura 4.25*)

1	2	3
4	X	5
6	7	8

Figura 4.25 Matriz de orientación para el robot móvil.

La X representa las coordenadas (x,y) del pixel en el que está situado, así, se recorre toda la imagen extrayendo un vector característico de la siguiente manera: Se sitúa la máscara en su posición inicial (marcada en amarillo) y se obtiene el valor de la posición de la casilla de su vecino (marcada en color rojo), que en este caso es 5 (*Figura 4.26*)

Figura 4.26 Posición inicial y posición vecino

Posterior a esto, el vecino se convierte en la nueva posición inicial de la máscara, quedando situada en la siguiente posición (*Figura 4.27*)

Figura 4.27 Reinicio de la posición inicial y análisis con la posición vecino

Como se puede apreciar su siguiente vecino está en la posición con un valor de 8. Esto se repite iterativamente hasta recorrer toda la línea, para el caso del ejemplo al final se obtendría un vector como el mostrado en la *Figura 4.28.*

5	8	5	8	5	5	5	5	5	2	2	2	2	2

Figura 4.28 Vector de orientación del robot móvil

Con el vector de posiciones obtenido se debía codificar cada valor de este con una instrucción especifica como encender los motores, por ejemplo al tener valores de 5 seguidos, el sistema debía interpretar esto y encender ambos motores para avanzar, si por el contrario no eran valores constantes seguidos, sino por ejemplo, pasar de un valor de 5 a un 8, el sistema debería interpretar esto como encender el motor del lado izquierdo para que el robot móvil gire hacia la derecha. Este proceso se convierte en algo complicado de ejecutar para el robot móvil, ya que por ejemplo, el primer tramo del trazado de la trayectoria corresponde a una línea diagonal, al aumentar significativamente la imagen el sistema reconoce esta línea diagonal como una serie de escalones, por ende al robot móvil le sería muy difícil de ejecutar esta tarea. Esto solo se cumple si el robot inicia en una posición de 5, ya que de lo contrario no tendrá orientación alguna, por ende se vio la necesidad de buscar un agente externo para una constante información de la posición angular del carro. Con ello se pensó en un sensor giroscopio MPU6050 (*Figura 4.29*), que permitiera establecer la posición inicial del robot, sin embargo debido a la complejidad de conexión y movimiento bajo este método, esta opción fue descartada.

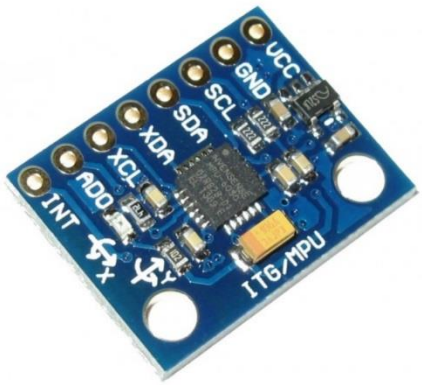
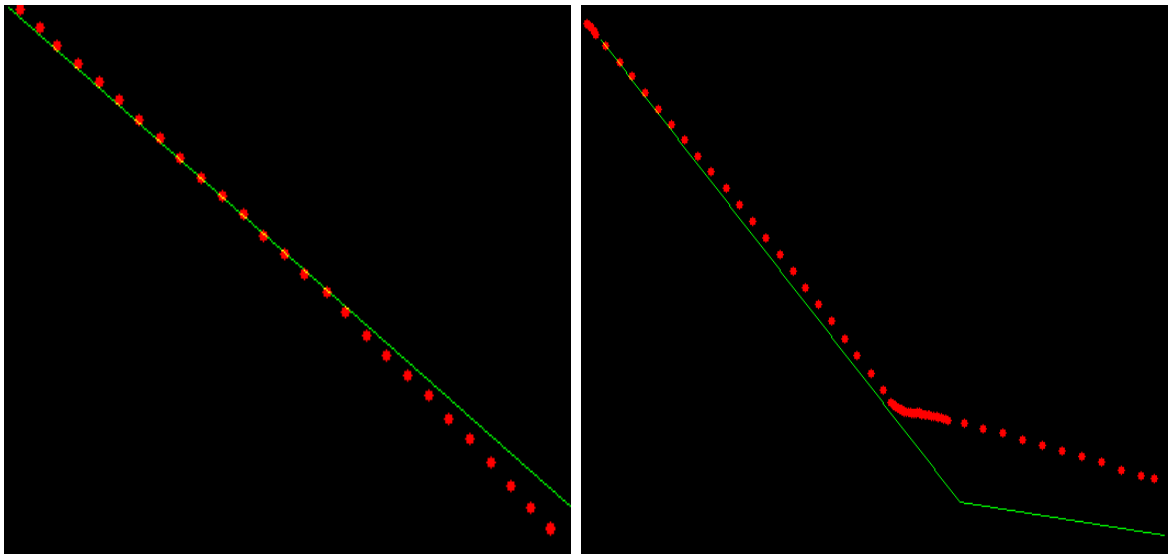


Figura 4.29 Giroscopio MPU6050 para posición angular de robot móvil (Fuente: www.prometec.net)

4.3.2 Sistema de control de lazo abierto

La segunda etapa inicia una vez procesada la imagen el entorno de movimiento y extraídas las características necesarias para guiar el robot móvil, el computador envía los datos de ángulo y distancia a la Raspberry, el carro inicia su movimiento sin parar hasta llegar a su destino, no obstante mientras se ejecuta el movimiento, el sistema no atiende información enviada por el computador. Lo que se convierte en un problema en caso de alguna eventualidad externa o dinámica de algún objeto, pues el carro no para, ya que no cuenta con alguna retroalimentación que le permita detenerse, provocando un choque del robot con un objeto en un movimiento posterior.

En la *Figura 4.30*, se muestra la comparación entre la línea ideal (línea verde) y la real trabajada por el carro (línea roja punteada). Los resultados son buenos, pues no necesita un control solo para la posición del carro, es decir enviar constantemente la posición hasta igualar la teóricamente esperada, sin embargo por la problemática anteriormente mencionado este método fue descartado, pues no concuerda en parte con los objetivos de este proyecto.



a) b)

Figura 4.30 Comparación de trayectorias lazo abierto a) Comparación de las trayectorias en un tramo recto b) Comparación de trayectorias en un tramo con curvas

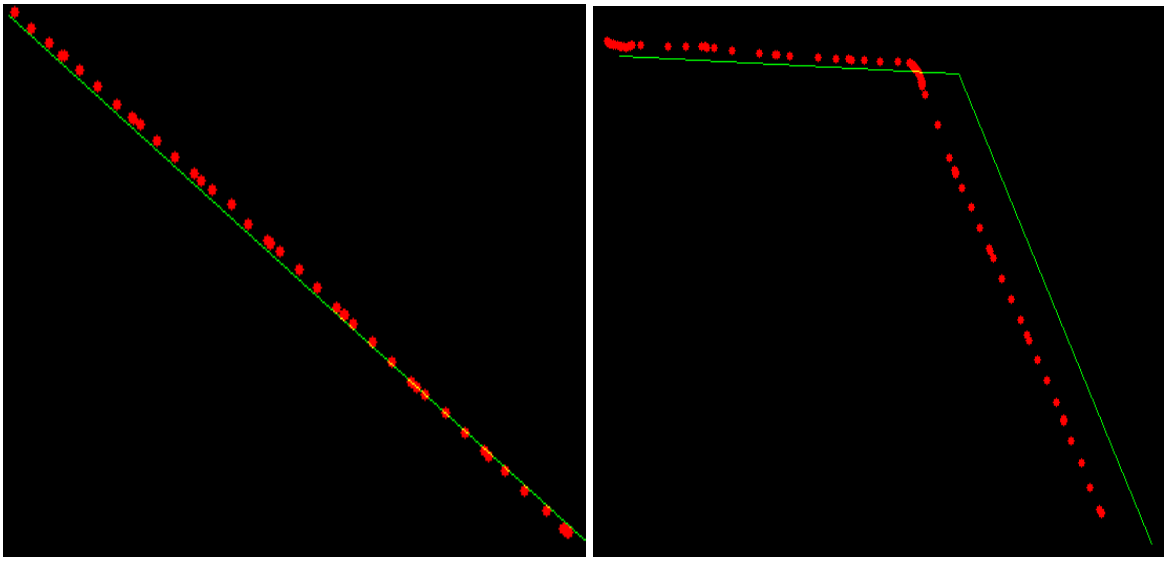
4.3.3 Sistema de control de lazo cerrado.

Para tener una mayor control sobre la posición del carro y disminuir la latencia del sistema en cuanto a cambios, ya sea por número de objetos o a la posición de los mismos, se optó por particularizar ciertos comandos de tal modo que si un obstáculo llegase a tocar la trayectoria por cualquier circunstancia, el sistema se detendrá y espera hasta que se recalcule otra ruta.

Para conseguir este control sobre el móvil se implementó una fragmentación del tiempo total del recorrido, el cual se obtuvo mediante la razón de la distancia entre los puntos y la velocidad del carro.

El control de lazo cerrado se completa debido a que el movimiento avanza bajo ciertas iteraciones y entre cada ciclo de iteración se analiza si el cliente le reporta al móvil que hubo un cambio en el sistema, la fragmentación fue conseguida haciendo una acumulación de un término que se calculó haciendo el inverso de la velocidad, con esto se obtuvo un factor particular que se sumaba iterativamente hasta llegar a ser igual al tiempo.

Los resultados de este proceso se pueden apreciar en la *Figura 4.31*, donde la línea verde representa la ruta ideal y la roja, son los puntos generados por el carro al pasar por determinado espacio.



a) b)

Figura 4.31 Comparación de trayectorias lazo cerrado a) Comparación de las trayectorias en un tramo recto b) Comparación de trayectorias en un tramo con curvas

4.3.4 Comparación de sistemas

Tanto en el sistema de lazo abierto como en el de lazo cerrado se obtuvieron buenos resultados, sin embargo, en el primer caso, no es posible saber el estado inmediato de cada objeto, ya que el sistema no cuenta con retroalimentación, limitándose solo a condiciones estáticas, es decir, para que este sistema funcione correctamente, los objetos siempre deben estar en reposo, de lo contrario presenta fallas. En el sistema de control de lazo cerrado se mejora la problemática anteriormente mencionada, pues gracias a la constante comunicación de la Raspberry con el computador, el carro no sale del ángulo requerido, llevándolo de una buena forma hasta su destino final.

En ambos casos, es evidente que para trayectorias directas, el sistema es mucho mejor que cuando tiene que hacer diferentes movimientos. Esto se debe al giro del

carro, ya que en el momento que se detiene y empieza su nueva búsqueda de ángulo, el carro puede sufrir un desplazamiento del eje de rotación, esto ocurre por qué, cuando el robot móvil gira, un motor está en HIGH y el otro está en estado LOW, por ende la fuerza que uno produce sobre el otro al girar, hace mover el carro y desviarlo un poco de su posición requerida.

La desventaja del sistema de control de lazo cerrado, en comparación con el sistema de lazo abierto, es el tiempo de movimiento de robot móvil hasta el destino, pues el carro demora un 20% más en culminar una tarea específica.

CAPITULO 5

CONCLUSIONES Y TRABAJO FUTURO

Las conclusiones del trabajo fueron:

- Con el posicionamiento de la cámara desde una perspectiva panorámico cenital del entorno de movimiento, se logró un adecuado direccionamiento para un robot móvil. Esto ayudó en gran medida a controlarlo, no solo por permitir no sufrir choques, sino por evitar movimientos innecesarios y generar un gasto innecesario de energía.

- Al trabajar con imágenes reales, se presentan problemas de segmentación de objetos debido a la baja iluminación de la plataforma y en el ambiente de trabajo. Para un correcto funcionamiento del sistema se debió combinar dos tipos de iluminación; una intensa utilizada en los bordes de la plataforma y una difusa situada en el centro de esta, esto con el fin de obtener uniformidad en los colores independiente de la posición donde estuviesen. Además, se aisló el sistema de la luz solar.

- La capacidad de detección de los objetos no solo depende del brillo en los mismos, sino también de su textura, lo que conllevo a la búsqueda de colores que minimizaran dicho efecto y mejoraran el funcionamiento del sistema, por ende, se utilizaron pinturas en tono mate.

- La elección de un fondo uniforme es esencial para lograr una detección adecuada de los obstáculos y así encontrar una trayectoria correcta.

- Para lograr una adecuada locomoción de este tipo de robots móviles, es necesario contar con una base plana y sin objetos paracitos que impida el movimiento de estos. Además, es necesario lograr un balance en el robot.

- La intensidad computacional del sistema al trabajar con procesamiento de video depende en gran medida de la instrumentación usada, pues se

requieren cálculos en tiempo real para hallar una ruta óptima y segura para el robot móvil.

Uno de los principales problemas encontrados en la elaboración del proyecto fue el coste computacional para la generación de una trayectoria, ya que siempre se buscó analizar el entorno en tiempo real, por ende, el sistema debe analizar muchos cuadros o imágenes por segundo, las cuales tienen inmerso una serie de puntos a analizar y en ocasiones saturan el programa. Para ello se establecieron condiciones de cálculo, es decir, el sistema solo analiza un nuevo frame cuando fuese necesario, con ello se aproximó en gran medida a un trabajo en tiempo real. No obstante, para ciertos procesos se encontraron tareas de mayor coste computacional, de tal forma que al sistema le tomó mayor tiempo en encontrar una trayectoria eficiente para completar la tarea designada. Otro inconveniente fue la iluminación de la plataforma, pues en el día funcionaba bajos ciertos parámetros, pero en la noche de debían recalcular estos. Esta problemática se solucionó trabajando en un ambiente controlado; se instalaron placas de cartón de color negro sobre la ventana del salón usado, además de contar con una buena iluminación sobre la plataforma, lo que permitió la correcta segmentación tanto de los objetos como del carro.

Como trabajo futuro se plantean algunos aspectos que se consideran fundamentales en caso de una posible mejora al sistema:

- Establecer una conexión entre el computador y el robot móvil mediante una red no local. Además lograr una enlace multi hilo que permita un control de lazo cerrado no solo en el ángulo, sino también en otras características del sistema como por ejemplo la posición.
- Maximizar el espacio de movimiento del carro usando un sistema multi cámara.
- Construir un móvil que presente características más similares a las de un automóvil normal en cuanto al direccionamiento del mismo, ya que en el utilizado en el proyecto el giro del robot se controló mediante la tracción en los motores ubicados en la parte posterior.

REFERENCIAS

[1]. Escalera, A. (2001). Visión por computador. Madrid. PEARSON EDUCACIÓN. S.A

[2]. Rojas Z, Teddy V., Sanz F., Wilmer, Arteaga, Francisco, Sistema de visión por computadora para la detección de objetos esféricos a través de la transformada de Hough. Revista INGENIERÍA UC

[3]. Garcia-Lamont, F; Cuevas, A y Nino, y. Segmentación de imágenes en color por características de cromaticidad utilizando mapas auto organizados. en g. investig.

[4]. Coca, Ledys, Franco, Zulay, & Pateti, Antonio. (2008). IMPLEMENTATION OF MORPHOLOGICAL FILTERS USED IN THE PROCESSING OF DIGITAL IMAGES IN A PROGRAMMABLE LOGIC DEVICE. Universidad, Ciencia y Tecnología, 12(48), 171-182

[5]. Endiola-Santibañez, Jorge D, & Terol-Villalobos, Iván R. (2005). Quantifying Contrast Methods through Morphological Gradient. Computación y Sistemas.

[6]. Peregrina Barreto, Hayde, & Terol Villalobos, Iván R. (2011). Contrast Enhancement Based on a Morphological Rational MultiscaleAlgorithm. Computación y Sistemas.

[7]. Morales Olivera, Yosbel, García Parrado, Josué, Reyes Fernández, Pablo E., & Lorenzo Ginori, Juan V.. (2012). Experiencias en la implementación de las operaciones morfológicas de erosión y dilatación para imágenes binarias empleando vecindades adaptativas. Ingeniería Electrónica, Automática y Comunicaciones.

[8]. Marinelli, Marcelo J, & De Silvestre, Enrique C. (2012). Desarrollo de un prototipo de robot móvil para la investigación y aplicación de técnicas de inteligencia artificial. Revista de Ciencia y Tecnología

[9]. Gonzalo Zabala, Robótica (2007), Banfield - Lomas de Zamora: Gradi.

[10]. Cardona, M. J., Castrillón, O. D., & Tinoco, H. A. (2017). Determinación del Método Óptimo de Operaciones de Ensamble Bimanual con el Algoritmo de Dijkstra (o de Caminos Mínimos). Información Tecnológica.

[11]. Pedraza, Luis F, López, Danilo, & Salcedo, Octavio. (2011). Enrutamiento basado en el algoritmo de Dijkstra para una red de radio cognitiva.

[12]. Restrepo, Jorge & Sanchez, John. (2004). Aplicación de la teoría de grafos y el algoritmo de Dijkstra para determinar las distancias y las rutas más cortas en una ciudad. Scientia et technica

[13] Ogata, K. (2010). Ingeniería de control moderna. Madrid. PEARSON EDUCACIÓN. S.A

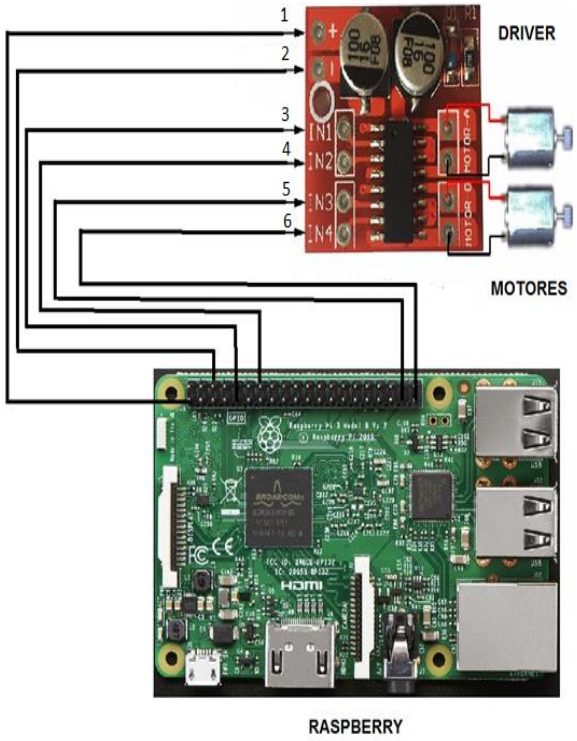
[14] Álvarez G, Carlos, Soto P, Andrés, & Watkins O, Francisco. (2009). SIMULACIÓN DE CONTROLADORES DIGITALES. Ingeniare. Revista chilena de ingeniería.

[15]. Rairán-Antolines, José Danilo, & Fonseca-Gómez, José Miguel. (2011). Doble lazo de control para regular la posición y la velocidad en un motor de corriente directa. Ingeniería y Universidad.

[16] opencv.org/about.html

ANEXOS

A. DIAGRAMA DE CONEXIÓN DEL ROBOT MÓVIL



REFERENCIA	DESCRIPCION
1	3.3 v
2	GND
3	GND
4	PIN 12 (GPIO18)
5	PIN 37 (GPIO 26)
6	GND

En el diagrama se muestran los tres elementos necesarios para la locomoción de robot móvil; Los motores que tienen un control independiente uno del otro, esto con el fin de permitir el direccionamiento del carro, el driver que se encarga de suministrar la potencia necesaria para que los motores se puedan mover y finalmente la Raspberry, que controla los motores para el giro y movimiento del robot móvil gracias a la constante comunicación con el computador.