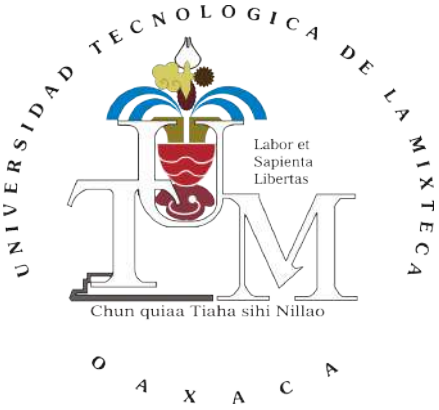


UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Corrección de rotación y traslación del sistema de locomoción del robot NAO.

T E S I S

Para obtener el grado en:

MAESTRO EN ROBÓTICA

Presenta:

ING. ARMANDO LEVID RODRÍGUEZ SANTIAGO

Director de Tesis:

DR. JOSÉ ANÍBAL ARIAS AGUILAR

Co-Director de Tesis:

DR. ROSEBET MIRANDA LUNA

Huajuapan de León, Oaxaca, México, Marzo de 2018

Tesis presentada en Marzo de 2018
ante los siguientes sinodales:

Dr. Antonio Orantes Molina.
Dr. Fermín Hugo Ramírez Leyva.
M.C. Verónica Rodríguez López.
M.C. Felipe Santiago Espinosa.

Director de Tesis:
Dr. José Aníbal Arias Aguilar.

Co-Director de Tesis:
Dr. Rosebet Miranda Luna.

A mi Madre
Desde el fondo de mi corazón con la mayor gratitud,
por tu amor sin límites y tus esfuerzos para ayudarme a
alcanzar un sueño más y enseñarme a luchar por lo que se quiere
y nunca rendirme. Por guiar mi camino y estar siempre junto a mí.
¡Gracias Mamá!

Agradecimientos

Gracias a Dios por darme la oportunidad de terminar mis estudios de maestría y por poner en mi camino personas que, de alguna manera u otra, directa o indirecta participaron con sus valiosos consejos y apoyo.

Gracias mamas Alberta y Lourdes, quienes con su gran amor, esfuerzo y cariño me han apoyado y motivado para alcanzar éste y muchos más de mis propósitos profesionales y personales. Por hacer de mí una mejor persona cada día y siempre procurar lo mejor para mí. **¡Gracias Mamas!**.

A mis tíos y tías por estar siempre conmigo brindándome su valioso e incondicional apoyo en mis estudios y en todos los aspectos de mi vida. Gracias a su ejemplo he aprendido a seguir adelante y alcanzar lo que me proponga.

Gracias a mis hermanos y hermana, a mis primos, primas y sobrinas, quienes siempre me han apoyado regalándome momentos increíbles de alegrías y felicidad. A Inés por siempre regalarme una sonrisa, darme ánimos, apoyo y motivación para continuar con éste trabajo.

A Jezabel por toda la ayuda, las enseñanzas, consejos y los importantes e inigualables momentos compartidos, por el valioso tiempo que me has regalado, por iluminarme y acompañarme todo este tiempo, por todo eso y más. Gracias.

A mis directores de tesis el Dr. José Aníbal Arias Aguilar y el Dr. Rosebet Miranda Luna por su valiosa dedicación, infinita paciencia e invaluable tiempo dedicado a la realización de este trabajo. A los sinodales el Dr. Antonio Orantes Molina, Dr. Fermín Hugo Ramírez Leyva, M.C. Verónica Rodríguez López y M.C. Felipe Santiago Espinosa por sus consejos y exigencias para realizar un trabajo de gran calidad.

Al Dr. Alberto Elías Petrilli Barceló, por los oportunos consejos y conocimientos brindados. Por la oportunidad de colaborar con el equipo Aztlán y a los compañeros que conformamos el equipo por los excelentes momentos en RoboCup.

A mi amiga Rocío por su apoyo, paciencia y gran amistad durante 12 años. A mis amigos Omar, Roberto, Daniel, Oscar y Magdiel por su ayuda y consejos durante el desarrollo de éste trabajo, el tiempo compartido y sobre todo por su amistad.

Por los oportunos consejos y valiosa ayuda en diferentes aspectos durante todo este tiempo agradezco sinceramente a la profesora M.C. María De La Luz Palacios Villavicencio, al I.E. Heriberto I. Hernández Martínez, el Dr. Jorge Luis Barahona Ávalos y el Dr. Felipe De Jesús Trujillo Romero.

A la Universidad Tecnológica de la Mixteca por los conocimientos adquiridos, la formación profesional y permitirme desarrollar mis estudios de maestría y este trabajo de tesis. A mis profesores de maestría, por su enseñanza y sabiduría. A mis compañeros Marcelino, Vidal y David por el buen ambiente de trabajo y de estudio. Al Consejo Nacional de Ciencia y Tecnología (CONACyT), el apoyo económico brindado para mis estudios de maestría.

A Todos
¡GRACIAS!

Resumen

En esta investigación se presenta el desarrollo y validación de dos propuestas de solución mediante técnicas de procesamiento de información visual para la estimación y corrección del error en el desplazamiento del robot humanoide NAO debido a factores inherentes a su sistema de locomoción.

La primera propuesta se basa en la detección de líneas mediante la transformada de Hough y la segunda se basa en el registro de imágenes utilizando como información la textura del suelo. El espacio de pruebas donde se desarrollan las técnicas es un laberinto fabricado en madera con una superficie de 180 cm x 300cm y paredes de 60 cm de altura.

Se evalúa cada técnica al determinar el error de posición tanto en desplazamientos como en rotaciones después de recorrer una distancia determinada para alcanzar una posición final deseada, evaluando los tiempos de procesamiento para finalmente determinar cuál de las dos técnicas es más adecuada para realizar la corrección de posicionamiento dentro del entorno estructurado seleccionado.

En el penúltimo capítulo de esta tesis se describen algunas aplicaciones que se han podido lograr gracias al desarrollo de este trabajo de investigación, donde se destaca la participación en la competencia internacional RoboCup 2017 en Nagoya Japón.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

En los últimos años ha ocurrido un avance significativo en la implementación y variedad de los sistemas robóticos desarrollados, que van desde el uso en la industria hasta aplicaciones para la automatización de un comercio o la convivencia en hogares. En estas tareas es necesario la integración de numerosas áreas, como son la Navegación Autónoma y la Visión Artificial, sólo por mencionar algunas, ya que dentro de estas también intervienen otras más específicas.

Dentro de los tipos de robots están los manipuladores y móviles, de estos últimos, se encuentran con ruedas, aéreos y con extremidades, lo que implica una complejidad distinta en su control, ya que cada modelo cinemático es único y tiende a ser bastante complejo, además del control que se desee implementar en ellos.

En esta investigación se presentan dos propuestas para estimar y corregir mediante técnicas de procesamiento de imágenes el error en el desplazamiento para el tipo de robots móviles con extremidades, específicamente en un robot humanoide NAO (Figura 1.1), el cual, debido a diferentes factores, presenta errores en desplazamiento, principalmente al realizar movimientos de traslación y realizar rotaciones. En este documento se revisan trabajos relacionados con este problema y se desarrollan y validan dos propuestas de solución para la corrección de las traslaciones y rotaciones del robot en un entorno controlado.

Figura 1.1: Robot NAO

1.1. Planteamiento del Problema

Dentro del estudio de la robótica móvil existen numerosas áreas de investigación, de las cuales resalta la navegación autónoma de robots humanoides. Como parte del equipamiento de la universidad, en los laboratorios de robótica se cuenta con varios ejemplares del robot humanoide NAO.

El robot puede rotar e inclinar la cabeza con precisión, además puede ser programado para realizar desplazamientos a lo largo del plano XY, y rotaciones sobre su propio eje, sin embargo y a pesar de ser una plataforma ampliamente sofisticada, es incapaz de realizar desplazamientos y rotaciones con exactitud, de manera que al realizar un desplazamiento y/o rotación programados, termina en una ubicación y con una orientación diferente a lo esperado, es decir, el robot es incapaz de trasladarse con precisión de una posición a otra.

Con base en lo descrito, se propone el diseño e implementación de un sistema de visión por computadora, empleando una o ambas cámaras del robot, que determine los errores en traslación y rotación que se generan cuando se desplaza desde un punto inicial hasta un punto final dentro de un entorno estructurado.

1.2. Justificación

La estimación de las desviaciones durante el desplazamiento programado del robot NAO es esencial para que se puedan corregir los errores al desplazarse de un punto a otro ya que es la base para realizar tareas más complejas de forma autónoma tales como actividades de entrenamiento y de servicio que impliquen la interacción con niños, demostraciones o incluso para la aplicación en competencias tecnológicas como la resolución de laberintos o juego de futbol.

Se pretende que esta investigación ayude en el desarrollo de sistemas más complejos que necesiten de un sistema de locomoción robusto, exacto, rápido y de adaptación factible.

1.3. Hipótesis

En un escenario $3D$ estructurado, es posible realizar la navegación autónoma del robot humanoide NAO, corrigiendo rotaciones y desplazamientos mediante un sistema de visión.

1.4. Objetivos

1.4.1. Objetivo General

Implementar y comparar dos estrategias de visión artificial para la estimación del error en traslación y rotación del robot humanoide NAO durante un desplazamiento programado, una de ellas basada en la detección de líneas y otra en la distribución de niveles de gris del suelo, de manera que le permitan corregir desviaciones en su desplazamiento dentro de un laberinto con suelo de duela oscura y muros blancos.

1.4.2. Objetivos Específicos

Para conseguir el objetivo principal se presentan los siguientes objetivos específicos.

1. Caracterizar los errores de rotación y traslación del robot NAO.
2. Implementar en Python la estrategia de visión artificial basada en la detección de líneas de manera que le permita al robot NAO determinar y corregir errores de translación y rotación.
3. Implementar en Python la estrategia de visión artificial basada en la distribución de niveles de gris del suelo de manera que le permita al robot NAO determinar errores de translación y rotación.
4. Implementar un módulo de locomoción en software que permita reorientar y trasladar al NAO a un destino final prestablecido.
5. Evaluar y comparar las dos estrategias de visión artificial.

1.5. Metas

1. Implementar un software para la adquisición de imágenes, así como aprender el funcionamiento del robot.
2. Realizar la calibración de la(s) cámara(s) del robot.
3. Implementar una etapa de acondicionamiento de las imágenes y detección de líneas basado en la transformada de Hough.
4. Calibrar los parámetros obtenidos por la transformada de Hough para conocer su relación con el mundo real.
5. Implementar la etapa de acondicionamiento de imágenes para el algoritmo de registro de imágenes.
6. Implementar un algoritmo de registro de imágenes basado en la distribución de los niveles de gris del suelo.
7. Calibrar los parámetros determinados por el algoritmo de registro de imágenes basado en distribución de niveles de gris para conocer su relación con el mundo real.
8. Integrar el software de estimación de errores de traslación y rotación mediante la transformada de Hough al software de locomoción del robot.
9. Integrar el software de estimación de errores de traslación y rotación mediante el algoritmo de registro de imágenes basado en la distribución de niveles de gris de la textura del suelo del laberinto al software de locomoción del robot.
10. Evaluar los las técnicas las implementadas para la corrección del sistema de locomoción del robot.

1.6. Metodología de Desarrollo

En este trabajo se emplea una metodología en cascada (1.2), la cual conlleva a realizar todas y cada una de las etapas de forma ordenada hasta alcanzar los objetivos establecidos inicialmente. A continuación se presentan las etapas de la metodología utilizada para el desarrollo de esta investigación.

1.6.1. Análisis de requerimientos

Como se ha descrito en el planteamiento del problema, se requiere de un sistema de reconocimiento que permita determinar o estimar los errores de desplazamientos del robot y de esta manera estar en condiciones de reorientar el robot. Para poder resolver este problema es necesario contar con información útil para la implementación de dicha solución. Es por esto que en la primera etapa de desarrollo se implementa el software para realizar la adquisición de información, que en este caso es la adquisición de imágenes del medio en el cual el robot se desenvolverá.

1.6.2. Concepción de la solución

El sistema debe tener la capacidad de estimar desviaciones en el desplazamiento de manera que esté en condiciones de realizar la corrección de la posición del robot, después de realizar movimientos de traslación y rotación respecto a una posición de referencia.

Para realizar esta estimación, es necesario determinar los desplazamientos y rotación relativos existentes entre dos imágenes, una de ellas adquirida desde la posición inicial (imagen de referencia) y la otra adquirida después de hacer avanzar al robot (imagen objetivo o a transformar). Además de esto, es necesario conocer la relación entre pixeles y centímetros.

El robot cuenta con dos cámaras, una ubicada en la parte frontal superior de la cabeza, con la cual se tiene una perspectiva de la parte superior del entorno, mientras que la segunda cámara está ubicada en la parte frontal inferior de la cabeza y permite conocer la información de la superficie sobre la cual el robot se encuentra situado. Por lo tanto, Se busca una relación entre la cámara y la información del entorno en el cual se trabaja.

Así mismo se propone la implementación de dos estrategias de procesamiento visual una basada en la transformada de Hough y otra en el registro de imágenes, con las cuales se pueden determinar los parámetros necesarios para reorientar al robot en el entorno estructurado.

1.6.3. Implementación

Una vez realizada la adquisición de la información del medio y el conocimiento de la relación que existe entre la información visual y el mundo real, se procede a la implementación de los módulos de software, con estos se obtiene la información necesaria para poder implementar posteriormente la corrección de traslación y rotación con ambas técnicas. Este proceso se realiza simultáneamente con la etapa de pruebas, ya que estos módulos se actualizan y/o modifican de acuerdo a los resultados que el robot presenta en la aplicación.

1.6.4. Pruebas

Esta etapa consiste en la realización de pruebas con el sistema, las cuales consisten en poner en funcionamiento el sistema en el entorno seleccionado[1] de manera que se asegure que, para cada determinada entrada de datos, el sistema y el robot, proporcionen los resultados que realmente corresponden a los esperados y que den solución al problema planteado. Como se citó anteriormente, esta etapa se realiza simultáneamente con la etapa de implementación.

1.6.5. Resultados

Finalmente, como última etapa de la metodología de desarrollo está la etapa de resultados. Esta etapa corresponde a la presentación de él o de los módulos que dan solución al problema planteado. Dado que en ocasiones el sistema requiere de modificaciones para adaptarse a nuevas aplicaciones, los módulos deben ser presentados de tal manera que tengan la capacidad de integrarse a otro sistema donde sean requeridos, modificándolos según sea necesario.

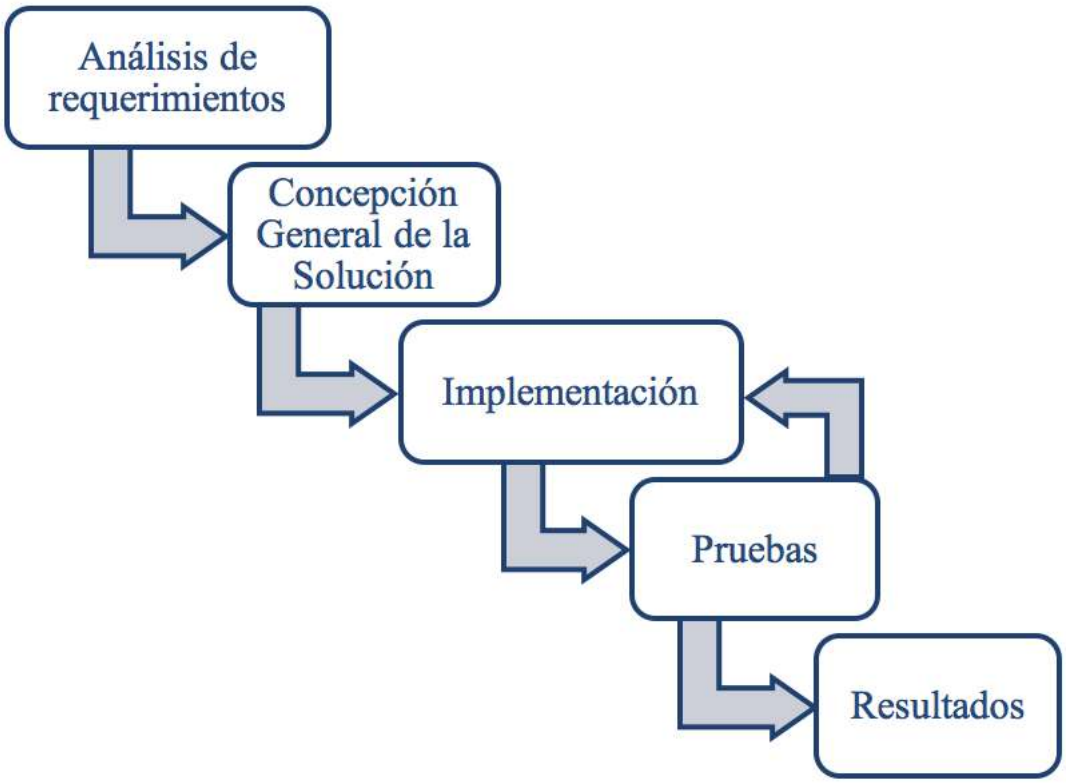


Figura 1.2: Diagrama de Flujo de la Metodología de Desarrollo

[1]Laberinto de los laboratorios de posgrado de robótica de la UTM

Capítulo 2

Marco Teórico

La robótica es una rama de la tecnología que, en los últimos años ha presentado un auge sumamente considerable. Un robot es toda aquella máquina capaz de realizar tareas de diversas clases de forma autónoma, además de ser reprogramables. Dentro de la robótica se encuentran diversos tipos de robots dependiendo del propósito de su aplicación, cuya complejidad va desde aquellos estacionarios que se utilizan en tareas industriales (e. g. ensambladores de autos), móviles o de locomoción (e. g. exploradores), hasta llegar a los de aspecto humano ("humanoides").

Como se describió en secciones anteriores, este último tipo de robot, es en el cual se enfoca la presente investigación, razón por la cual a continuación se realiza una revisión de los tópicos científicos necesarios para comprender mejor esta propuesta de tesis.

2.1. Visión Artificial

El objetivo de la visión artificial (o visión computacional) es obtener información del entorno mediante el procesamiento de imágenes o video con la cual identificar objetos de interés y su posición en un ambiente, de forma que mediante la interpretación de las imágenes adquiridas se puedan reconocer los diversos objetos en el ambiente y su posición en el espacio. Actualmente existen múltiples áreas de aplicaciones prácticas de la visión computacional, algunas son:

- ✓ Robótica móvil y vehículos autónomos.
- ✓ Tecnología avanzada de manufactura.
- ✓ Análisis e interpretación de imágenes médicas.
- ✓ Interpretación de escritura, dibujos, planos, etc.

La visión artificial está estrechamente relacionada con el procesamiento de imágenes, que a pesar de compartir información en común sus objetivos son diferentes.

El procesamiento de imágenes se encarga del mejoramiento de la calidad de las imágenes para su posterior interpretación, por el contrario, la visión computacional se encarga de extraer características de una imagen para su descripción e interpretación [Sucar, 2012]. Para la adquisición de una imagen se requiere de un dispositivo físico que sea sensible a una determinada banda del espectro electromagnético como lo son: cámaras fotográficas, de video (vidicón o de estado sólido - CCD), digitalizadores (scanners), sensores de rango (franjas de luz, laser) y sensores de ultrasonido (sonares).

Las imágenes son digitalizadas y pueden ser procesadas una vez almacenadas en la computadora, o en el mismo dispositivo de captura. En la práctica se utilizan diferentes tipos de imágenes tales como: Imagen binaria, Imagen en tonos de gris o monocromática, imágenes a color e imagen multi-espectral [Armendariz., 2003].

Por lo tanto, la visión consiste en partir de una imagen (pixeles) y llegar a una descripción adecuada de acuerdo a nuestro propósito. Estos sistemas son muy complejos por lo cual se divide en varias etapas o niveles de visión. En cada una se va refinando y reduciendo la cantidad de información hasta llegar a la descripción deseada. Se consideran generalmente tres niveles:

- ✓ Procesamiento de nivel bajo - se trabaja directamente con los pixeles para extraer propiedades como contornos, gradiente, profundidad, textura, color, etc.
- ✓ Procesamiento de nivel intermedio - consiste en agrupar los elemento obtenidos en el nivel bajo, para obtener líneas, regiones, con el propósito de segmentación.
- ✓ Procesamiento de alto nivel - esta generalmente orientada al proceso de interpretación de los entes obtenidos en los niveles inferiores y se utilizan modelos y/o conocimiento a priori del dominio.

Aunque estas etapas son aparentemente secuenciales, esto no es necesario, y se consideran interacciones entre los diferentes niveles incluyendo retroalimentación de los niveles altos a los inferiores.

2.2. Operaciones de procesamiento de imágenes

El procesamiento de imágenes es una forma de tratamiento de la señal, donde la señal de entrada es una imagen (por ejemplo, una foto o un vídeo) y la salida es ya sea una imagen o un conjunto de parámetros asociados a ella. La mayoría de las técnicas de procesamiento de imágenes en niveles de gris tratan la imagen como una señal de dos dimensiones donde x e y son las coordenadas espaciales de la imagen y la amplitud de I se llama intensidad o nivel de gris. El procesamiento de imágenes es un campo de estudio considerablemente extenso el cual comprende una gran cantidad de operaciones como son:

- ✓ Filtrado, para mejor una imagen o detectar bordes.
- ✓ Restauración de la imagen y la reconstrucción.
- ✓ Wavelets y procesamiento de multi-resoluciones.

- ✓ Transformaciones geométricas euclidianas como la ampliación, reducción y rotación.
- ✓ Correcciones de color como ajuste de brillo y contraste, cuantificación o conversión de color a uno diferente.
- ✓ El registro de imágenes (para la alineación de dos o más imágenes).
- ✓ Reconocimiento de imágenes (por ejemplo, la extracción de una cara en la imagen mediante el uso de algún algoritmo de reconocimiento facial).
- ✓ Segmentación de la imagen (la partición de la imagen en regiones características de acuerdo al color, bordes, u otras características)

Una vez que la imagen ha sido procesada se está en disposición de detectar los objetos en ella, para ello se buscan en la imagen aquellas características que definen al objeto que se desea encontrar, algunas operaciones de procesamiento de imágenes más relevantes en robótica son el filtrado y el alisado de la imagen y detección de bordes entre otras.

2.2.1. Segmentación

En cualquier imagen se encontrará presente uno o varios objetos localizados en un entorno. El objetivo de la segmentación es separar dichos objetos del medio en el que se encuentran y distinguirlos entre sí. Para ello se utilizan algunas de las propiedades o características como son: niveles de gris, color, textura o bordes, entre otros. Después de la separación estará el proceso de descripción individualizada de los objetos encontrados y su reconocimiento e interpretación. Las técnicas para obtener la segmentación se basan en la búsqueda de las partes uniformes de la imagen o justo lo contrario, aquellas donde se produce un cambio. La segmentación se basa en tres propiedades:

1. Similitud: Cada uno de los pixeles de un elemento tiene valor parecido para alguna propiedad.
2. Discontinuidad: Los objetos destacan del entorno y tienen por tanto unos bordes definidos.
3. Conectividad: Los pixeles pertenecientes al mismo objeto tienen que ser contiguos, es decir, deben estar agrupados.

Umbralización

La umbralización se realiza para eliminar información irrelevante o considerada como ruido dentro de una escena. Con la umbralización se pasa de una imagen en escala de grises a una imagen binaria, es decir, en solo dos niveles. De manera que los objetos de interés son separados del fondo o el resto de la información contenida en la imagen. Para realizar este procedimiento, es necesario encontrar un umbral de intensidad que pueda realizar la separación de los objetos de interés. El valor de umbral es aquel que pueda separar claramente los objetos de la información irrelevante, seleccionando de esta manera solo los de interés.

Básicamente se realiza una selección de pixeles con valor mayor o igual (o menor o igual) al umbral seleccionado, con esto se determina que los que cumplen dicha condición pertenecen al objeto de interés y aquellos que difieren de dicha condición son pertenecientes al fondo de la imagen.

2.2.2. Detección de bordes

Mediante la extracción de bordes se delimitan los objetos y así se separan del resto del contenido de la imagen. Las técnicas usadas tienen por objetivo la localización de los puntos en los que se produce una variación de intensidad, empleando para ello métodos basados en la derivada.

Los bordes de una imagen definen regiones en el plano en donde ocurre un cambio significativo en el brillo de la imagen. La utilización de bordes detectados en las imágenes en lugar de utilizar la imagen completa reduce significativamente la cantidad de información, reduciendo los tiempos de post-procesamiento, y por lo tanto es una opción útil durante la interpretación de imágenes. Existen diversos algoritmos que determinan los bordes en una imagen, por ejemplo, se encuentran entre otros: el Gradiente, Sobel, Laplaciano y Canny.

Algoritmo de Canny

El algoritmo de Canny [Canny, 1986] está considerado como uno de los mejores métodos de detección de contornos mediante el empleo de máscaras de convolución, este algoritmo consiste en tres grandes pasos [Rebaza, 2007] :

- ✓ Obtención del gradiente: En este paso se calcula la magnitud y orientación del vector gradiente en cada píxel. Para lo cual se realiza la convolución de la fila y columna de cada punto de la imagen original con las mascaras para filas y columnas respectivamente (ecuación 4.3) con lo cual se obtiene la magnitud y dirección (orientación) del gradiente (2.2) . (Algoritmo 1 , Apéndice A).

$$\begin{aligned} G_F(i,j) &= F(i,j) \otimes H_F(i,j) \\ G_C(i,j) &= F(i,j) \otimes H_C(i,j) \end{aligned} \tag{2.1}$$

$$\begin{aligned} |G(i,j)| &= \sqrt{G_F^2 + G_C^2} \approx |G_F| + |G_C| \\ \phi(i,j) &= \tan^{-1}\left(\frac{G_C}{G_F}\right) \end{aligned} \tag{2.2}$$

- ✓ Supresión no máxima: En este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho, obteniendo una imagen I_n con los bordes adelgazados, es decir, $I_m(i,j)$ (Ver Algoritmo 2, Apéndice A).

- ✓ Histéresis de umbral: En este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos. Este procedimiento se logra siguiendo el Algoritmo 3, mostrado en el Apéndice A

2.3. Transformada de Hough

La transformada de Hough fue presentada por Paul Hough en 1962 y está reconocida como una técnica muy efectiva para la detección de líneas rectas y otras formas geométricas. Es una técnica utilizada para aislar características de forma particular dentro de una imagen.[Climent and Mares, 2003]

Analíticamente se puede describir un segmento de línea en varias formas. Sin embargo, una ecuación conveniente para describir un conjunto de líneas es la notación paramétrica o normal (ecuación 2.3) [Duque and Ospina, 2004][Aggarwal and Karl, 2006].

$$\rho = x\cos\theta + y\sin\theta \tag{2.3}$$

Donde ρ y θ son la longitud y la orientación, respectivamente, del vector normal a la recta desde el origen de coordenadas de la imagen. Cada línea recta está definida de forma única por ρ y θ, y para cada punto (x, y) en la imagen original se puede crear una correspondencia entre la imagen y el espacio transformado (Figura 2.1).

Con este enfoque, los pixeles se asignan a líneas en un espacio de parámetros 2D. Cada celda del espacio de parámetros acumula el número de líneas rasterizadas sobre él. Al final, las celdas con los números acumulados más grandes (por votos) representan las líneas que mejor se ajustan al conjunto en los pixeles de entrada [Fernandes and Oliveira, 2008].

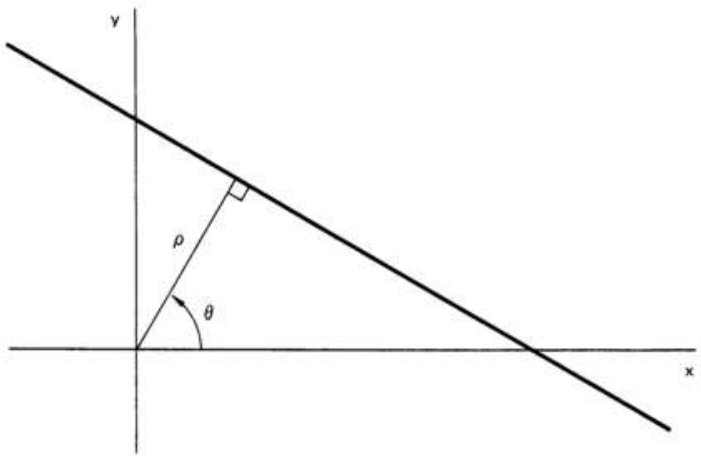


Figura 2.1: Representacón de lineas por la transformada de Hough

Duda y Hart proponen un algoritmo donde sustituyen la intersección de pendiente con una parametrización de ángulo-radio basada en la ecuación normal de la línea (ecuación 2.3), manejando naturalmente las líneas verticales y reduciendo significativamente los requerimientos de memoria.

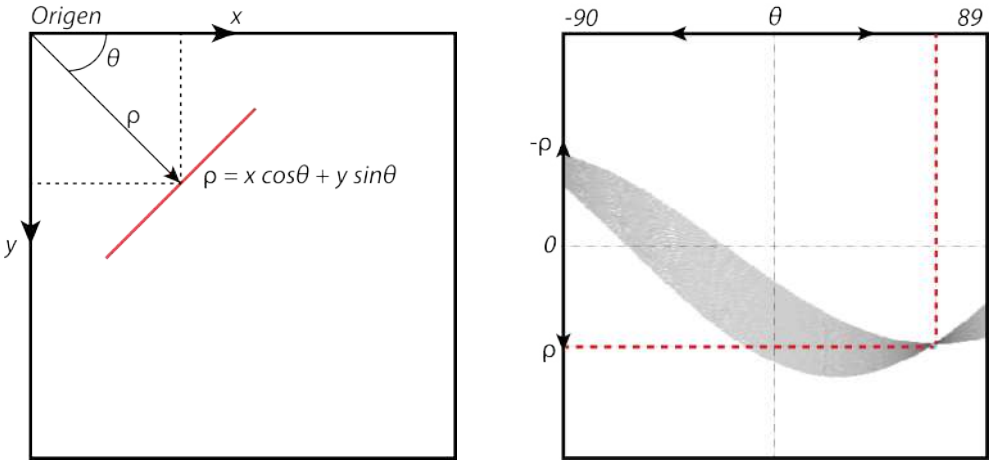
En el algoritmo de Duda y Hart, x y y son las coordenadas del pixel en la imagen, ρ es la distancia perpendicular desde el origen del sistema de coordenadas de la imagen a la línea, y θ es el ángulo entre el eje de la imagen y la normal a la línea.

Al restringir a θ $[0^o, 180^o)$ y a $\rho\,[-R, R]$, donde $R = \sqrt{w^2 + h^2}/2$ con w y h como la anchura y altura de la imagen, respectivamente. Todas las líneas posibles en la imagen tienen una representación única en el espacio de parámetros. El Algoritmo 4 (Apéndice A) resume el proceso de votación propuesto por Duda y Hart [Duda and Hart, 1972].

La discretización de la imagen y del espacio de parámetros hace que las líneas se intersectan en muchas celdas, creando picos secundarios. Este efecto se ve reforzado por el hecho de que los pixeles de cada segmento no son exactamente colineales. Como resultado, el algoritmo puede recuperar algunas líneas falsas. Para solventar este problema el sistema de votación de fuerza bruta por pixel descrito en el Algoritmo 4 se sustituye por una estrategia de voto por segmento con lo cual se evita la detección de líneas falsas.

2.3.1. Interpretación de la transformada de Hough

Los resultados de la aplicación de la transformada de Hough se representan a través de una matriz bidimensional cuyos máximos locales representan líneas en la imagen original. Para mejor interpretación de estos puntos se presenta la Figura 2.2, en donde podemos observar la representación de las coordenadas de una línea (color rojo) presente en una escena, tanto en el plano imagen (a), como en el espacio de Hough (b).



(a) Plano Imagen (b) Espacio Hough

Figura 2.2: Interpretación en las imágenes de la transformada de Hough

En el plano imagen, de forma general los sentidos de las coordenadas se toman con el origen en la esquina superior izquierda, y positiva hacia abajo y x positiva hacia la derecha. Mientras que el ángulo θ se mide positivo en sentido horario desde el eje x. La distancia a la línea desde el origen se representa por ρ. Por otra parte, tenemos el espacio de Hough donde, los ejes ahora son ρ y θ, y un conjunto de puntos pertenecientes a una línea. En el plano imagen se asigna a un conjunto de sinusoides que se intersectan en un punto en el espacio de Hough. Así el problema de detectar una línea en una imagen se convierte en un problema de detectar un punto en el espacio de Hough.

2.4. Registro de Imágenes

El registro de imágenes puede ser considerado como un procedimiento de alineamiento espacial de imágenes, pudiendo ser estas de la misma o de diferente modalidad [Li et al., 1995]. Los métodos de registro de imágenes son necesarios cuando buscamos la fusión de imágenes que contienen información complementaria (fusión de datos), o simplemente para comparar el contenido entre dos imágenes de la misma escena o sus cambios relativos (mediante la puesta en correspondencia de su información) [Li et al., 1995].

Geométricamente esto supone alinear una de las imágenes contra otra. Este tema tiene multitud de aplicaciones, tales como, detección de rostros, codificación de vídeo, predicción del tiempo, codificación de bases de datos, y la estimación de traslaciones y rotaciones relativas de un sistema de adquisición de imágenes entre muchas otras. Las diferencias en el contenido de dos imágenes de la misma escena adquiridas con el mismo sistema pueden ser de diversa naturaleza: desplazamientos, giros, escala, distinto contraste, etc. los cuales pueden llega a ser realmente complejo. [García Capel, 2007]

2.4.1. Principio matemático del registro de imágenes.

Sean dos imágenes I_r (referencia) e I_T (a transformar). Poner en correspondencia la información de dos imágenes implica determinar los parámetros de la transformación $\widetilde{T}$ (Ecuación 2.4) mediante un método de optimización ($arg\ opt$) que maximice la semejanza (medida de similitud S) de la información entre imágenes. Las funciones f_1 y f_2 representan los algoritmos de segmentación de las imágenes.

$$\widetilde{T} \quad = \quad arg \quad opt_T S(f_1(I_r), T(f_2(I_T))) \tag{2.4}$$

La medida de similitud, el tipo de la transformación geométrica T y el método de optimización a emplear se definen en función de la aplicación, de la naturaleza de las imágenes, de los requerimientos en términos de precisión, de la robustez deseada, etc. Las diferentes componentes del esquema matemático clásico del registro de imágenes constituyen criterios que permiten clasificar los algoritmos de registro de imágenes [Posada et al., 2004].

2.4.2. Componentes del esquema clásico del registro de imágenes.

✓ Información común.
La información en común (estructuras homólogas) es extraída de las imágenes gracias a algoritmos de segmentación (funciones f_1 y f_2 de la Ecuación 2.4) o calculada directamente con los valores de los niveles de gris. La información que se extrae de las imágenes es de dos tipos.

- Imágenes primitivas:
  Las imágenes primitivas son objetos geométricos que pueden ser extraídos de las imágenes, son por ejemplo puntos particulares, contornos con forma de segmentos de rectas o de curvas, superficies, etc.

Es posible utilizar varios tipos de primitivas al mismo tiempo, por ejemplo contornos y puntos para volver más robusto al procedimiento de registro de imágenes [Li et al., 1995].

- Niveles de Grises:
  Los niveles de gris de las imágenes son con frecuencia utilizados para calcular una medida de similitud en el caso en el que ninguna primitiva significativa pueda ser extraída de las imágenes. En tal caso, la medida de similitud es calculada gracias a una relación común entre los niveles de gris de las imágenes, o a partir de criterios estadísticos.

✓ Medida de similitud.
Consideremos por ejemplo un caso ideal de registro de imágenes en el que las dos imágenes son adquiridas con la misma cámara. Supongamos que los puntos de vista de las dos adquisiciones son casi idénticos, que las condiciones de iluminación permanecen relativamente constantes y que el ruido que afecta a las imágenes es despreciable. Bajo tales condiciones, un píxel de una imagen y su correspondiente en la segunda imagen deberían tener niveles de grises idénticos o muy parecidos si los dos pixeles representan un mismo punto de la escena. A partir de esta hipótesis de correlación (relación = identidad) es fácil deducir una medida de similitud:

"la suma de la diferencia entre pixeles correspondientes es nula o mínima cuando las imágenes se encuentran alineadas perfectamente"

✓ El tipo de transformación.
Según la aplicación, existen transformaciones T más o menos complicadas entre imágenes que modifican la relación espacial entre pixeles.

La naturaleza de la T a utilizar está ligada en general al sistema de adquisición de las imágenes, a los tipos de escenas adquiridas y a sus posiciones relativas. Algunos ejemplos de trasformaciones son los mostrados en la tabla 2.1. En la mayoría de las aplicaciones, siendo necesario considerar una relación geométrica mas compleja. Dependiendo de la naturaleza de las imágenes, a menudo es necesario determinar el tipo de transformación que modela mejor las diferencias entre imágenes [Rodríguez Santiago, 2013].

✓ El método de optimización.
El objetivo de toda optimización es encontrar de manera rápida, robusta y precisa los parámetros de transformación que maximicen o minimicen, según sea el caso, una métrica (medida de similitud) relacionada con el grado de correspondencia o solapamiento entre las imágenes. El tipo de método de optimización se selecciona de acuerdo a diferentes criterios:

- Tiempo de cálculo disponible,
- Ubicación de la posición inicial en el espacio de parámetros,
- Naturaleza de los datos (discretos o continuos),

- Forma del espacio de parámetros (extremo global, con o sin, numerosos extremos locales), etc.

En el caso en el que el espacio de parámetros contiene además del extremo global buscado numerosos extremos locales bien pronunciados, y que las condiciones iniciales están relativamente alejadas de la solución buscada, resulta difícil, y a veces imposible, encontrar la solución con una técnica de optimización. En estos casos extremos, una búsqueda exhaustiva en el espacio de parámetros es necesaria.

2.4.3. Criterio de Woods

El histograma conjunto

Para designar el espacio que caracteriza las estructuras extraídas de las imágenes y desde el cual se realiza el registro se utiliza el llamado histograma conjunto. Este histograma es una generalización de las matrices de concurrencia utilizadas en el análisis de textura. Para una transformación dada T, el histograma está definido por:

$$H_T = D_f \times D_r \rightarrow \mathbb{R}^+ \tag{2.5}$$

En el histograma conjunto las imágenes se consideran como variables aleatorias (VA) y los píxeles corresponden a los eventos elementales cuyo dominio es el conjunto D_r o D_f de las intensidades. El cálculo se realiza de forma iterativa y utilizando un procedimiento de interpolación [Sarrut, 2000]. Denotamos n_{ij} el valor de $H_T(i,j)$ que representa el número de píxeles, $n_i = \sum_j n_{ij}$ y $n_j = \sum_i n_{ij}$ como los números marginales, y $n = \sum_i \sum_j n_{ij}$ el tamaño total, de modo que $I_f(x) = j$ e $I_r(T(x)) = i$ representan las intensidades (ver Figura 2.3).

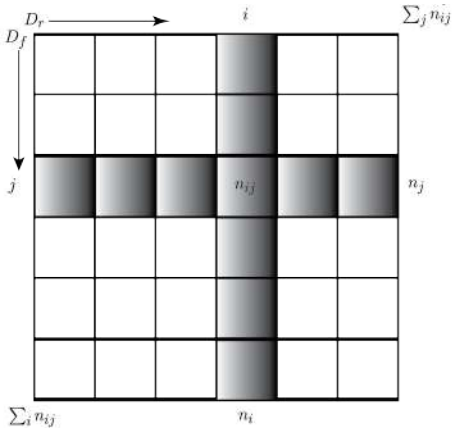


Figura 2.3: Diagrama del histograma conjunto

Al igual que el histograma unidimensional de los niveles grises de una imagen, las cantidades n_i, n_j y n_{ij} se normalizan para obtener distribuciones elementales y marginales. El histograma conjunto es entonces el equivalente de una tabla de contingencia estadística convencional.

Tabla 2.1: Clasificación de Modelos de Transformación

Transformación	Grados de Libertad	Modelo de Transformación	Ejemplo
Euclidiana	Traslaciones + Rotaciones	$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & \pm\sin\varphi \\ \mp\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$	
Rígida	Euclidiana + Factor de Escala Uniforme	$\begin{bmatrix} x' \\ y' \end{bmatrix} = k \begin{bmatrix} \cos\varphi & \pm\sin\varphi \\ \mp\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$	
Afin	Rigida + Factor de Escala No Uniforme	$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} k_x\cos\varphi & \pm S_x\sin\varphi \\ \mp S_y\sin\varphi & k_y\cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$	
Proyectiva	Afin + Proyección Perspectiva	$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \end{bmatrix} \Lambda \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	

Tomada de [Rodríguez Santiago, 2013].

Criterio de Woods

Dada una intensidad en una imagen, es decir, todos los pixeles con ese valor, el principio fundamental de esta medición es considerar la variación de todas las intensidades correspondientes entre las imágenes. La hipótesis planteada es considerar que esta variación es mínima cuando las imágenes se encuentran registradas es decir, que el criterio de Woods es un criterio a minimizar (ver Figura 2.4). Al introducir las definiciones y la notación de medias y varianzas condicionales (ecuación 2.6). El criterio de Woods se define en la ecuación 2.7:

$$\begin{aligned} m_{J|i} &= \frac{1}{p_i}\sum_j jp_{ij}; \qquad \sigma^2_{J|i} = \frac{1}{p_i}\sum_j (j - m_{J|i})^2 p_{ij} \\ m_{I|j} &= \frac{1}{p_j}\sum_i ip_{ij}; \qquad \sigma^2_{I|j} = \frac{1}{p_j}\sum_i (i - m_{I|j})^2 p_{ij} \\ Var(J \mid I = i) &= \sigma^2_{J|i}; \qquad Var(I \mid J = j) = \sigma^2_{I|j} \end{aligned} \tag{2.6}$$

$$Woods(I \mid J) = \sum_j \frac{\sigma_{I|j}}{m_{I|j}} p_j \tag{2.7}$$

Donde $m_{J|i}$ y $m_{I|j}$, representan el promedio de los elementos del histograma conjunto, para los elementos normalizados de columnas y renglones p_i y p_j respectivamente, mientras que p_{ij} indica el elemento (i, j) del histograma, $\sigma^2_{J|i}$ y $\sigma^2_{I|j}$ representan las varianzas correspondientes.

Este criterio mide la media de las desviaciones estándar normalizadas y la relación entre la desviación estándar y la media. Se debe tener en cuenta que Woods no es una medida simétrica por lo que se debe hacer una elección para determinar la imagen que mejor describa a la otra [Sarrut, 2000].

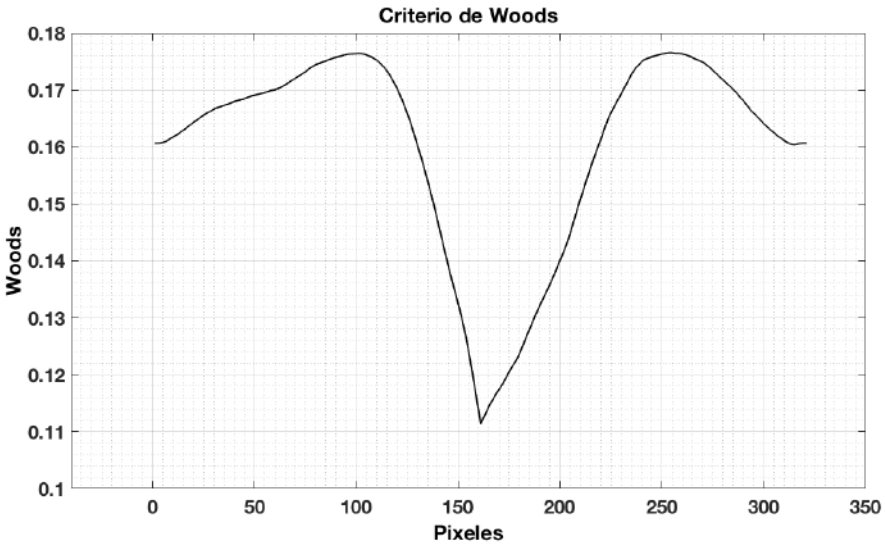


Figura 2.4: Aplicación del criterio de Woods

2.5. Información contenida en las Imágenes

Una señal, dependiendo del número de variables independientes que la describan, puede ser unidimensional, bidimensional, etc., por ejemplo, una imagen en niveles de gris se considera una señal bidimensional. Además, dependiendo de las dimensiones pueden ser escalares como en blanco y negro o vectoriales como las imágenes a color, que tienen tres componentes. Una imagen digital esta por tanto almacenada en una matriz de M x N x K elementos [De la Escalera Hueso, 2001]. A continuación se describe la información encontrada en las imágenes.

2.5.1. Color

El color es una de las características más importantes que definen a los objetos, sin embargo, hace poco tiempo se le prestaba poca atención debido al costo computacional y a la memoria necesaria para procesar imágenes en color. Los parámetros de la percepción del color son el brillo o luminosidad que incorpora la noción cromática de intensidad. El tono es un atributo asociado con la longitud de onda dominante en una mezcla de ondas de luz.

El tono representa el color dominante tal como lo percibe un observador. Por lo tanto, cuando llamamos a un objeto rojo, naranja o amarillo, estamos especificando su tono. La saturación se refiere a la pureza relativa.

El uso del color en el procesamiento de imágenes es motivado debido a que el color es un poderoso descriptor que a menudo simplifica la identificación de objetos y extracción de una escena. Además los seres humanos pueden discernir miles de tonos de colores e intensidades diferentes, en comparación con aproximadamente dos docenas de valores de niveles de gris. Esto es fundamentalmente importante cuando el análisis de imágenes es realizado por seres humanos [Rafael C. Gonzalez, 2002].

2.5.2. Texturas

Por otra parte, al observar cierto tipo de materiales se nota que presentan un aspecto homogéneo a pesar de que no tienen un nivel de gris constante. Así, las texturas se pueden definir como patrones visuales homogéneos que se observan en ciertas materias (telas, piedras, madera). La importancia de su estudio radica en que la textura forma parte de las características que definen a los objetos. Para estudiar la textura se pueden seguir dos aproximaciones: estadística y frecuencial. En el primer caso se analizan los valores estadísticos de primer orden o superior de los niveles grises. En el segundo caso, al ser la textura la repetición espacial de un patrón, aparecerán valores altos o picos en su transformación de Fourier relativos a la frecuencia de repetición.

2.6. El robot NAO

NAO es un robot humanoide programable y autónomo desarrollado por la empresa francesa Aldebaran Robotics (hoy SoftBank Robotics). Esta plataforma de desarrollo cuenta con un sistema de locomoción bípeda que posee 25 grados de libertad, dos cámaras de alta definición (una ubicada en la parte superior y otra en la parte inferior de la cabeza, con ángulo de visión horizontal y vertical de 239°, 68°, respectivamente), cuatro micrófonos, sensores ultrasónicos, infrarrojos, de contacto y de tope (ver Figura 2.5).

Además de contar con giroscopios y acelerómetros. Cuenta también, con una microcomputadora integrada con CPU ATOM Z530 1.6 GHz, Memoria Flash de 256 MB SDRAM / 2 GB y sistema operativo Linux [Grupo Mediatec, 2012]. Todo esto nos permite programar y manipular al robot para realizar así la tarea que se desee.

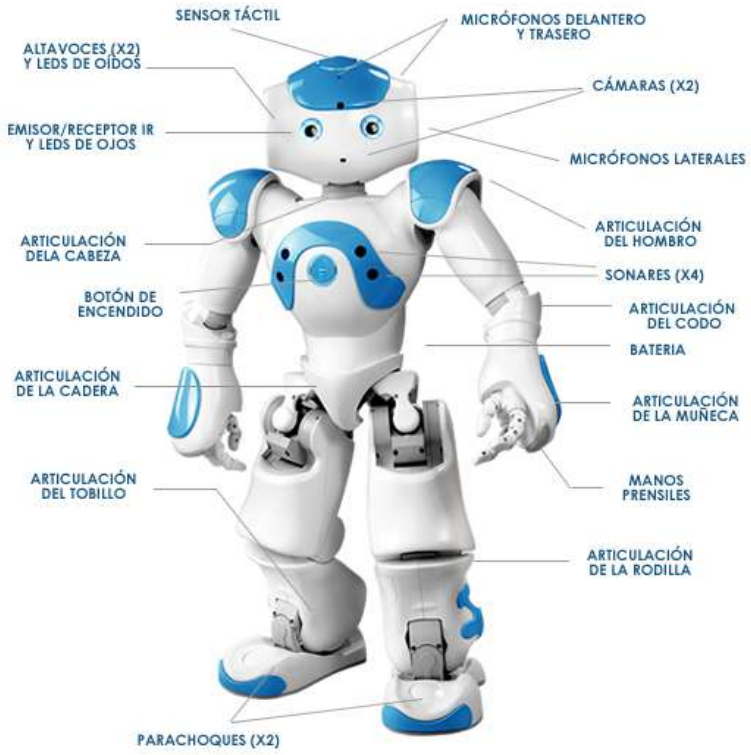


Figura 2.5: Características de robot NAO

2.6.1. Programación de robot

Para el desarrollo de aplicaciones con el robot NAO, SoftBank Robotics proporciona el SDK NAOqi, el cual cuenta con múltiples bibliotecas que permiten programar al robot para realizar diversas tareas, en distintos lenguajes de programación, además de ser compatible con los principales sistemas operativos. NAOqi están organizado en módulos y proxies; éstos son objetos (en el ámbito de la programación orientada a objetos) que nos permiten activar o acceder a ciertas funcionalidades del robot, como moverse o grabar audio.[Robotrónica, 2013].

En lo que respecta a la programación del robot NAO podemos utilizar diferentes lenguajes de programación como Choreographe, Python y C++, los cuales nos permiten optimizar la plataforma y desarrollar aplicaciones de alto grado de complejidad.

Choreographe

Este entorno gráfico de programación es sencillo e intuitivo, además permite fácilmente visualizar el porcentaje de carga de la batería y controlar el volumen de las bocinas del robot. La programación se realiza mediante bloques y posee una gran variedad de acciones, que van desde simples, hasta complejas tales como mantener diálogos e incluso reconocimiento de objetos y de personas a través de sus cámaras.

Las acciones de NAO se programan a través de diagramas de flujo, encadenando bloques uno tras otro, pudiendo realizar múltiples tareas de manera simultánea. Las acciones están contenidas en bloques de librerías y el usuario puede crear nuevas, modificarlas y hasta borrarlas.

Python

Python es el lenguaje de programación orientado a objetos. A través de Python es posible modificar e incluso generar Scripts propios de Choreographe, mediante los cuales indicar al robot patrones de comportamiento y ordenar acciones determinadas, tanto desde la terminal de Python como desde Choregraphe.

Este lenguaje destaca por su fácil legibilidad, mantenibilidad y rapidez en el aprendizaje, por encima de otros lenguajes de programación orientado a objetos, lo que hace de Python un excelente lenguaje para desarrollar scripts personalizados para la programación de NAO, además de esto posee muchas de las ventajas de otros lenguajes tales como Java, siendo su sintaxis mucho más sencilla y visual.

C++

C++ es el lenguaje de programación orientado a objetos más estandarizado en Robótica. Este lenguaje está indicado para desarrolladores experimentados por su complejidad. Permite crear nuevos módulos para NAO, como reconocimiento facial, de objetos, y es posible crear nuevas funcionalidades utilizando información que el robot recibe de forma concurrente a través de sus sensores. Para generar estos programas es necesario realizar una compilación cruzada, la cual consiste en generar programas para una estructura de microprocesador diferente a la que se está utilizando para programar. Para ello, podemos utilizar las herramientas que nos proporciona Softbank Robotics en su página web [Robotrónica, 2013]. Este lenguaje de programación es utilizado en los sistemas operativos con núcleo Linux (como NAO), y en frameworks como OpenCV (utilizado en NAO) y el meta-sistema operativo ROS (compatible con NAO).

OpenCV

OpenCV [Kaebler, 2008] es una biblioteca de visión artificial de código abierto. La librería está escrita en C y C++ y se ejecuta bajo Linux, Windows y Mac OS X, en interfaces para Python, Ruby, Matlab y otros lenguajes. Fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. OpenCV puede aprovechar los procesadores multi-core. Uno de los objetivos de OpenCV es proporcionar una infraestructura de visión por computadora simple que ayude a las personas a construir aplicaciones de visión bastante sofisticadas rápidamente.

La biblioteca OpenCV contiene más de 500 funciones que abarcan muchas áreas de visión, incluyendo inspección de productos de fábrica, imágenes médicas, seguridad, interfaz de usuario, calibración de cámara, visión estéreo y robótica. Debido a que la visión por computadora y el aprendizaje automático de la máquina van de la mano, OpenCV también contiene una Biblioteca de Aprendizaje Automático (Machine Learning Library o MLL) de uso general. La licencia de código abierto para OpenCV se ha estructurado de tal forma que se puede crear un producto comercial utilizando todo o parte de OpenCV.

Capítulo 3

Estado del Arte

La necesidad de interactuar con las máquinas es de gran importancia cuando se quiere realizar tareas industriales o de servicio; podemos entonces destacar numerosas investigaciones sobre estas áreas, que dan solución a problemas específicos en el control de posición o desplazamiento de robots, tal es el caso de Dieter et al, que en el año 2000 presentan un algoritmo estadístico de colaboración entre robots donde se utiliza el método de Markov para la localización de muestras [Fox et al., 2000]. El método utiliza funciones de densidad de probabilidad y un esquema universal basado en muestras. La implementación es realizada en plataformas móviles, equipadas con cámaras y sensores laser, los cuales colaboran entre si para poder detectar otros robots. Los resultados mostrados demuestran una considerable reducción de la incertidumbre en la localización, sin embargo las limitaciones de este sistema son grandes ya que el sistema solo es capaz de procesar detecciones positivas, es decir cuando es posible "ver" a otro robot.

Stephen y Lowe en el año 2002, presentan la implementación de un algoritmo para la localización de un robot móvil en un ambiente dinámico basado en un sistema de visión estéreo que utiliza la transformación de características invariantes en escala (SIFT). La invariancia de características en la imagen se utiliza como referencia para la localización y la creación de un mapa [Se et al., 2002]. Para poder aplicar este algoritmo en entornos dinámicos se hace uso del filtro de Kalman para realizar el seguimiento de las marcas, con lo cual se obtiene una base de datos del entorno. Se logran importantes resultados en la construcción de mapas logrando además estimar su posición en el entorno y el uso eficiente de la memoria del sistema.

En el año 2003 Ramírez describe un algoritmo para el aprendizaje de un mini-robot (Figura 3.1), para realizar navegación dentro de un ambiente desconocido mediante una red neuronal asociativa [Ramírez, 2003]. Para la red se presenta una metodología de entrenamiento, que se puede programar en un sistema FPGA, logrando que a través de las fases de entrenamiento y navegación el móvil pueda realizar la navegación autónoma en el entorno. Este tipo de estrategia ofrece una rápida solución al problema de la navegación, siendo una solución sencilla pero poco eficiente porque a pesar del previo entrenamiento (conocimiento) para obtener un conjunto de estrategias útiles para poder cumplir la misión de navegación, el mini-robot colisiona.

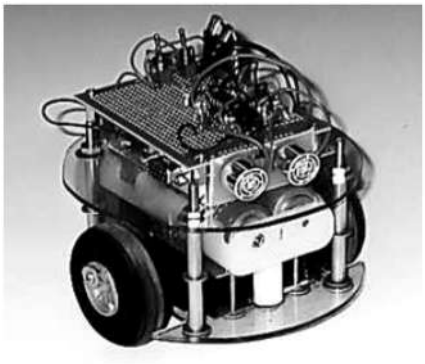


Figura 3.1: Mini-robot móvil de tracción diferencial Controlado por FPGA

La tesis de Vázquez presentada en 2005 tiene como objetivo mejorar la auto-localización de un robot móvil con ruedas en un ambiente estructurado [Vázquez Jiménez, 2005]. El sistema básicamente adquiere una imagen en la posición actual por medio de una cámara de video montada sobre el móvil (Figura 3.2). La estimación de la posición del robot se realiza mediante la comparación de las características extraídas de la imagen actual, de la cual se extrae el histograma de intensidad de iluminación y es convertido en un vector numérico, el cual es relacionado con un conjunto de vectores que permite estimar la ubicación actual del robot.

El sistema presentado requiere de un entrenamiento previo, es decir, de la captura de imágenes en distintas poses del ambiente además que estas imágenes deben ser capturadas con mucha precisión y cuidado, ya que de lo contrario no se podrá estimar su ubicación en el entorno.

Figura 3.2: Robot Golem UNAM

Gifford en 2009 utiliza registro de imágenes para determinar giros y desplazamientos de un móvil, el cual debe poder conocer su localización conforme se desplaza, usando una cámara web enfocada hacia el suelo. El registro lo basan en la FFT con lo cual se determinan distancias de desplazamiento en 2D y ángulos de rotación respecto a la posición de partida entre imágenes consecutivas [Gifford, 2009]. El plano imagen de la cámara se desplaza siempre paralelo al suelo. Este método solo permite estimar translaciones y rotaciones[Anuta, 1970] [Chen et al., 1994] [Reddy and Chatterji, 1996] y al no ser robusto ante la perspectiva, solo se aplica a sistemas donde ésta sea nula.

Hernández en su tesis de 2011 presenta un sistema tanto para ubicación como para navegación de un humanoide (Figura 3.3) en un ambiente estructurado [Cruz Hernández, 2011]. El principal sensor que se utiliza para adquirir información del entorno es una cámara que captura imágenes del ambiente. En la investigación presentada el sistema de visión es capaz de adquirir y procesar multiples imágenes por segundo, de manera que se asegura que el sistema robótico cuente con información oportuna en un ambiente dinámico como lo es por ejemplo un partido de futbol.

Debido a que los objetos de interés (porterías, marcas de referencia, pelota) tienen por reglamento colores previamente especificados, la segmentación de pixeles se hace en base al color. Se propone una técnica de segmentación por color basada en la agrupación de regiones elipsoidales rotadas en el espacio y se determinan las distancias hacia su objetivo a partir del análisis del número de pixeles.

Figura 3.3: Robot Humanoide utilizado por Hernández

Rodríguez en 2013 presenta un sistema de visión por computadora con la capacidad de corregir la ubicación de la plataforma móvil Zagros (Figura 3.4) respecto a una imagen de referencia colocada en la pared [Rodríguez Santiago, 2013]. Utiliza un software de registro de imágenes 2D basado en un modelo de transformación que comprende traslaciones, escala y perspectiva y que mediante una calibración previa de las cámaras [Rodríguez Santiago et al., 2012], se relaciona respectivamente con desplazamientos paralelos al plano imagen, distancias hacia dicho plano y rotaciones sobre su propio eje. Para determinar los parámetros de transformación utiliza una búsqueda exhaustiva en combinación con la estrategia de optimización piramidal, con lo que es capaz de estimar la ubicación de la plataforma.

Figura 3.4: Plataforma Movil Zagros

Bazeille en 2013 presenta un nuevo framework para un robot cuadrúpedo (Figura 3.5), que realiza la navegación orientada a un objetivo en un terreno desconocidos mediante el uso de datos de medición inercial y visión estereoscópica [Bazeille et al., 2013]. Este framework incluye la percepción y el control y permite al robot navegar en línea recta. El sistema de navegación desarrollado no necesita ningún tipo de cartografía o de planificación de trayectoria ya que las cámaras estéreo se utilizan para guiar visualmente el robot y evaluar el terreno. Esto asegura la locomoción mediante la combinación de una respuesta visual directa y mediciones inerciales.

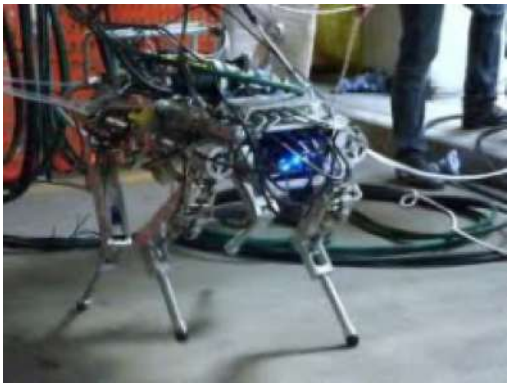


Figura 3.5: Robot Cuadrúpedo Utilizado por Bazeille

Yáñez presenta un sistema de detección de objetos para robots NAO basado en información visual obtenida por las cámaras del robot [Arancibia, 2013]. El ambiente de trabajo es una cancha de futbol para robots, con esto se busca siempre detectar líneas y arcos del entorno, este sistemas se basa en el ajuste de una serie de reglas organizadas en cascada con las cuales se determina si los modelos encontrados corresponden a objetos buscados. Gracias a que la geometría y dimensiones de los objetos son conocidas es posible determinar su posición relativa respecto al robot NAO.

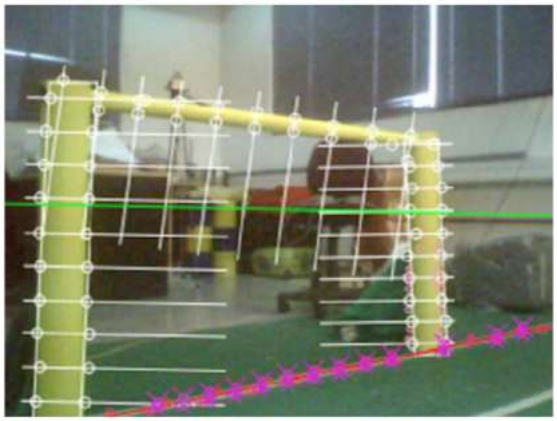


Figura 3.6: Escenario de implementación para Yáñez

Delfín presenta un artículo sobre la locomoción del robot humanoide NAO, basado en estrategias de control visual [Delfín et al., 2014]. Este sistema determina la diferencia que existe entre un par de imágenes, una inicial de referencia (siempre constante) y una obtenida desde la posición actual, respectivamente. La información necesaria es obtenida mediante un detector de esquinas y la búsqueda de correspondencias entre los puntos característicos de las imágenes objetivo e inicial. La estrategia visual permite desplazamientos omnidireccionales sin información 3D del entorno.

Figura 3.7: implementación del sistema de Delfín

Vargas en 2015 presenta un artículo donde desarrolla un algoritmo para controlar un par de robots humanoides, los cuales conforman un sistema multi-agente para la manipulación de objetos previamente conocidos [Vargas Torres and Castillo Estepa, 2015]. El sistema de visión se encarga de filtrar, detectar contornos y clasificar la información obtenida visualmente. Debido a que los objetos poseen formas geométricas conocidas y los robots son también marcados por figuras geométricas conocidas, el algoritmo de visión se encarga de la extracción de figuras cuadradas y triangulares mediante la detección de contornos. En base a la información obtenida de la detección de las figuras geometrías es posible conocer la orientación de los robots así como de los objetos a manipular.

El sistema de visión únicamente es utilizado para la detección robusta de los marcadores de los robots y los objetos a manipular sin ser afectados por el ruido y la iluminación. A pesar de contar con un buen sistema de visión, el cual es capaz de procesar hasta 30 cuadros por segundo, es susceptible a la variación inducida por las oscilaciones de los robots al desplazarse y las condiciones del ambiente de operación, con lo cual el tiempo de cumplimiento de las tareas se extiende inadecuadamente.

Caramazana en 2015 [Zarzosa, 2015] presenta un algoritmo para la corrección de la odometría de un vehículo empleando un sistema visual en combinación con el filtro de Kalman el cual realiza la estimación de la posición, orientación y trayectoria, a partir del análisis de una secuencia de imágenes adquiridas por un sistema de visión estéreo. Con este método es posible eliminar los efectos de la no linealidad en el proceso de la odometria e incluso es posible estimar el movimiento del móvil sin necesidad de realizar una reconstrucción 3D del escenario.

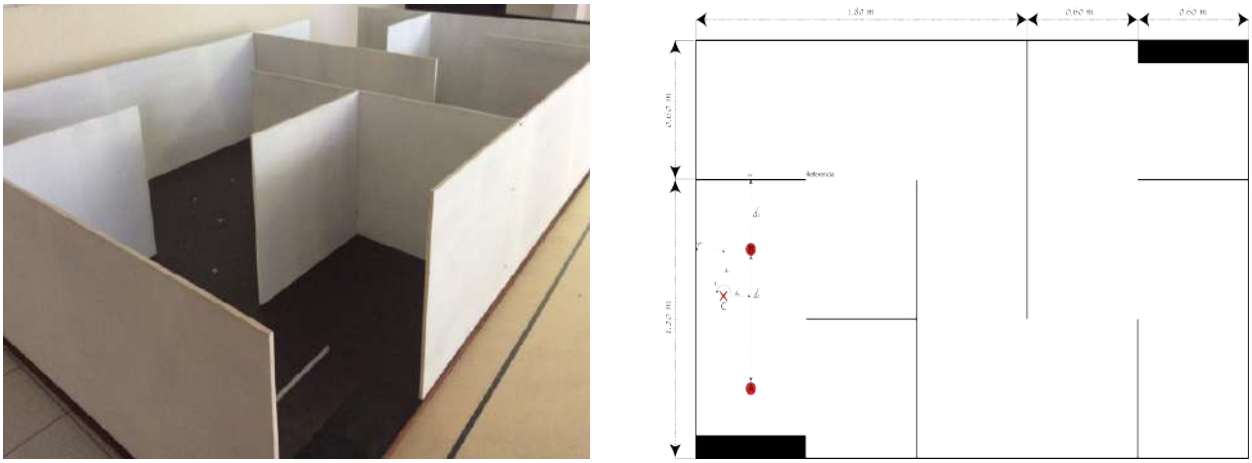
Capítulo 4

Desarrollo del sistema de visión.

Con base en la metodología en cascada propuesta en la introducción, se presenta el desarrollo y la implementación del sistema de visión que pretende dar solución al problema de referenciación para la locomoción del robot NAO.

4.1. Análisis de requerimientos y concepción de la solución

Se requiere de un sistema que permita estimar los errores en desplazamientos y rotaciones del robot, de manera que se pueda estar en condiciones de reorientar el mismo en un entorno conocido. Se considera como entorno estructurado un laberinto fabricado de madera con paredes blancas y piso color madera obscura (Figura 4.1a). Este laberinto mide 300cm de largo, 180cm de ancho y 60cm de altura, con pasillos de 60cm de amplitud(4.1b).



(a) (b)

Figura 4.1: Laberinto de los laboratorios de Robótica.

4.1.1. Características del robot

El sistema de coordenadas del robot NAO tiene como eje positivo x frente a él, y hacia el lado izquierdo y z hacia arriba del robot. En el eje z, referente a las rotaciones sobre su propio eje, se consideran ángulos positivos en dirección contraria al movimiento de las manecillas del reloj y negativos en direccion opuesta. En la Figura 4.2 se muestra el sistema de coordenadas del robot o locales y las del mundo real o globales. El sistema de coordenadas globales se mantiene siempre fijo, por lo tanto, los movimientos que el robot realiza están en función de las coordenadas globales. Tomando como origen la referencia de orientación o la posición que se desea alcanzar, las distancias d_x, d_y y rotaciones r_z que el robot realizará serán sobre los ejes XW, YW, ZW respectivamente.

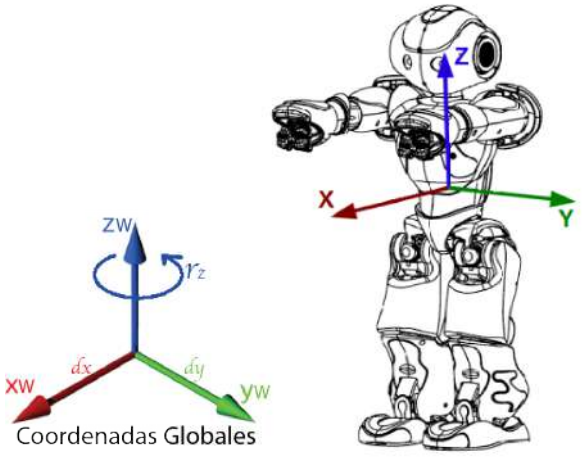
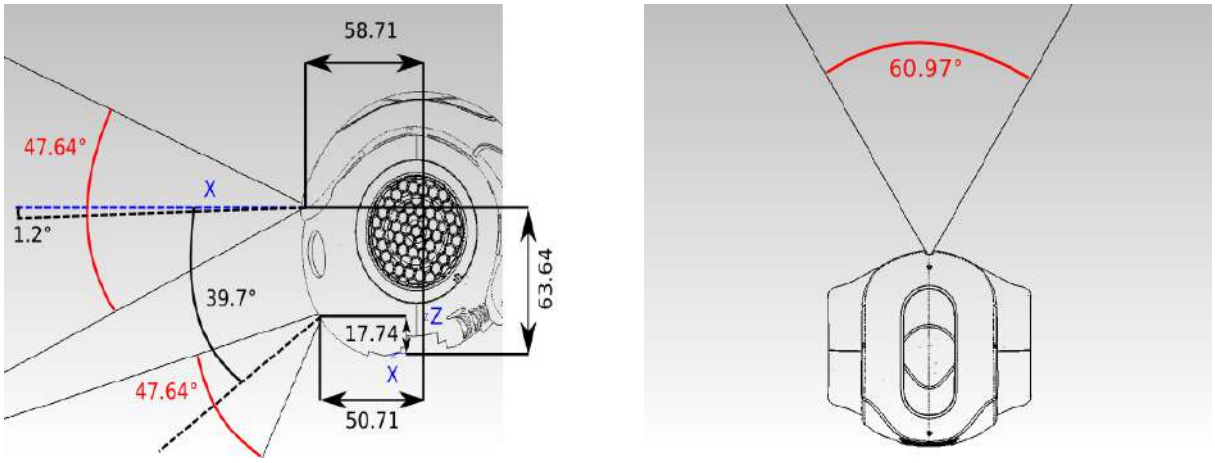


Figura 4.2: Sistema de Coordenadas del Robot NAO

La información necesaria es adquirida mediante la(s) cámara(s) ubicadas en la cabeza del robot y sus ángulos de rotación son $HeadPitch$ y $HeadYaw$, referentes a rotaciones sobre el eje y y z respectivamente (ver Figura 4.4), estos ángulos determinan la orientación de la cabeza del robot y a su vez el de las cámaras, las cuales cuentan con las características mostradas en la Figura 4.3 con dimensiones en milímetros y ángulos en grados, mientras que la profundidad de campo (d_x) varía según su posición (valores de los ángulos $HeadPitch$ y $HeadYaw$) de la cabeza del robot (ver Figura 4.5). Tomando el estado inicial del robot (*"StandInit"*), la cámara inferior tiene una profundidad aproximada de 125 cm mientras que la cámara superior alcanza más de los 300 cm medidas tomadas desde el centro de los pies del robot.

Analizando la información proporcionada por las cámaras del robot, se determina utilizar únicamente la cámara inferior, ya que ésta proporciona información relevante a la aplicación, por el contrario, la cámara superior, proporciona, en gran medida, información externa e irrelevante al entorno estructurado seleccionado (ver Figura 4.6).



(a) (b)

Figura 4.3: Características de las cámaras del robot.

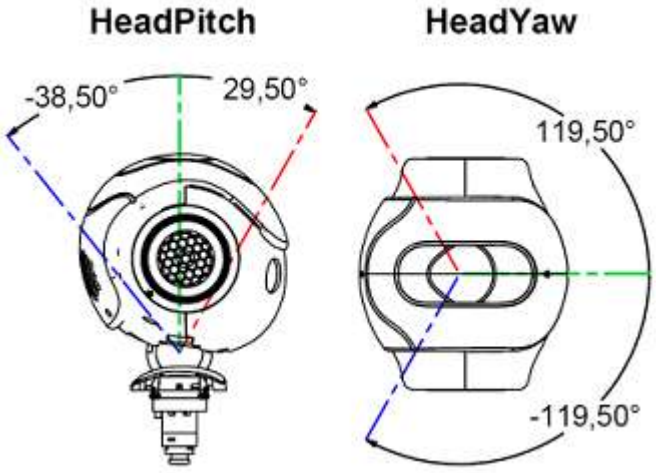


Figura 4.4: Ángulos de movimiento de la cabeza del robot NAO

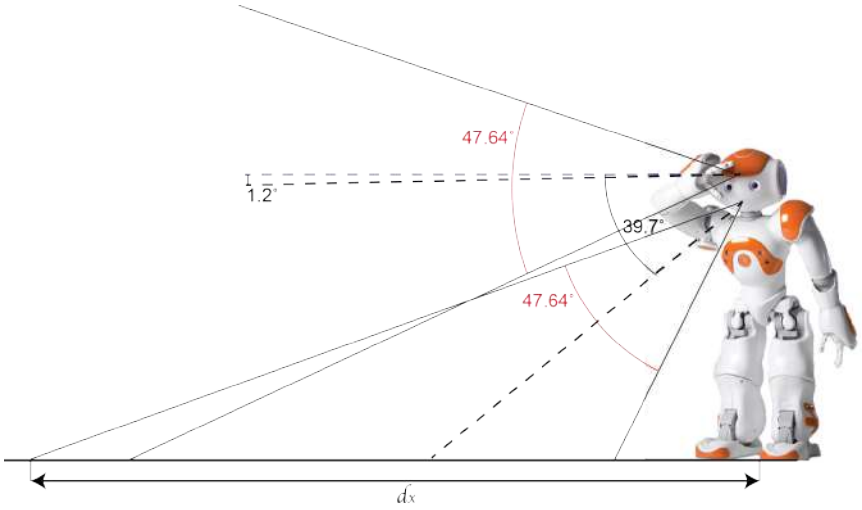


Figura 4.5: Profundidad del campo de visión de las cámaras del robot

(a) Imagen porporcionda por cámara superior (b) Imagen porporcionda por cámara inferior

Figura 4.6: Captura de imágenes de las cámaras del robot

Programación del Robot.

Como se ha mencionado, para la programación del robot se disponen de herramientas graficas como Choregraphe, herramientas de simulación como Webots for Nao y directamente el SDK NAOqi. Es posible programar al robot haciendo uso de lenguajes de programación como C++, Python, Matlab o Java.

En esta investigación se usa el lenguaje de programación Python. En cuanto a las ventajas que ofrece para programar al robot, se destaca la capacidad de indicar patrones de comportamiento y acciones determinadas desde la terminal de Python o incluso combinar los scripts con bloques de Choregraphe.

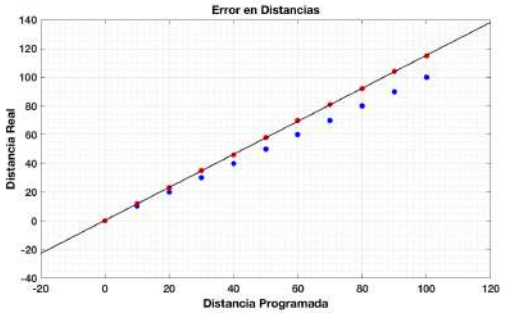
Para poder programar al robot haciendo uso de Python, es necesario obtener e instalar el SDK directamente de la página de Aldebaran Robotics [Aldebaran, 2016] y una vez instalado se puede programar libremente al robot. Para codificar cualquier programa a ser ejecutado en el robot es necesario importar ALProxy del naoqi, crear el módulo que se desea manejar, y llamar algún método de este módulo.

4.1.2. Errores en traslaciones y rotaciones

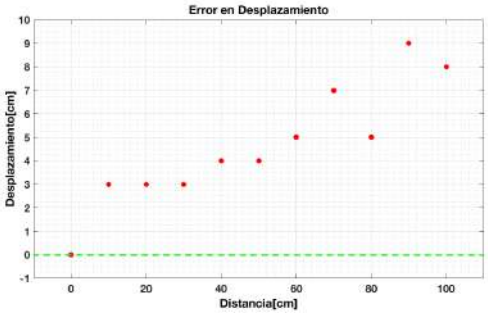
Al desplazar al robot desde una posición inicial **A** a una final **B**, idealmente el robot debería llegar a dicha posición orientado de manera correcta. Desafortunadamente el robot no puede alcanzar la posición deseada, presentando imprecisiones en su posición final. Las gráficas de la Figura 4.7 muestran en azul los movimientos programados, en rojo los movimientos realizados y en negro una aproximación lineal que podría describir el comportamiento de cada fenómeno. Se observa que además del error de distancias en direccion del eje x del robot presenta desplazamientos en el eje y, es decir, al avanzar en dirección del eje x el robot presenta una desviación hacia la izquierda (eje $+y$). Analizando los resultados se observa que, mientras la distancia de desplazamiento programada aumenta, el error presente tanto en distancia como en desviación también aumenta considerablemente. De igual forma los movimientos de rotación presenta un comportamiento similar. Con la caracterización de los movimientos se determina un error promedio de 7.8 cm, máximo de 15 cm y desviaciones de la media de 5 cm en distancias sobre el eje x. Desplazamientos sobre el eje y con media de 4.6 cm, máximo de 9 cm y 2.6 cm de desviaciones de la media. Para las rotaciones se tiene un error promedio de 2.3^o, con máximo de 6^o y desviación estándar de 2^o.

La caracterización de los errores, se realiza con el robot recién energizado y la cargada de batería desde 100 % hasta 75 %, dando un tiempo de operación de 60 *minutos* aproximadamente, esto con la finalidad de que el sobrecalentamiento de los motores por el tiempo de operación y los niveles de batería no influyan en los errores de desplazamiento y rotación.

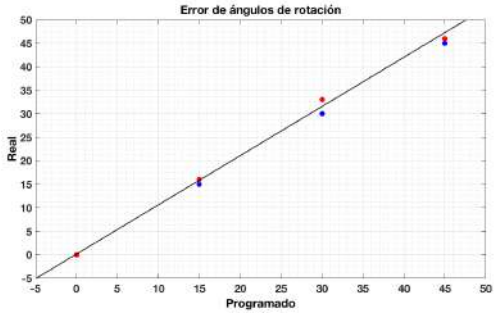
Con estos resultados podemos observar que a pesar de encontrar una aproximación lineal para los errores de desplazamientos en los ejes x, y y de rotación, no existe una retroalimentación de los movimientos realizados por el robot, y las probabilidades de controlar el robot, únicamente con movimientos programados resulta ser sumamente difícil. Todo esto dificulta la reorientación del robot y su corrección respecto a la posición final que se desea alcanzar, por lo cual, es necesario implentar un sistema para corregir los desplazamientos y giros.



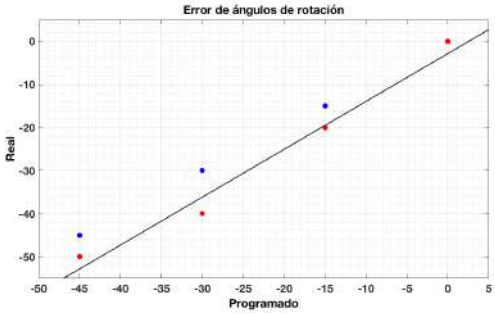
(a) Errores en Distancias



(b) Errores en Desplazamientos



(c) Error en rotaciones positivas



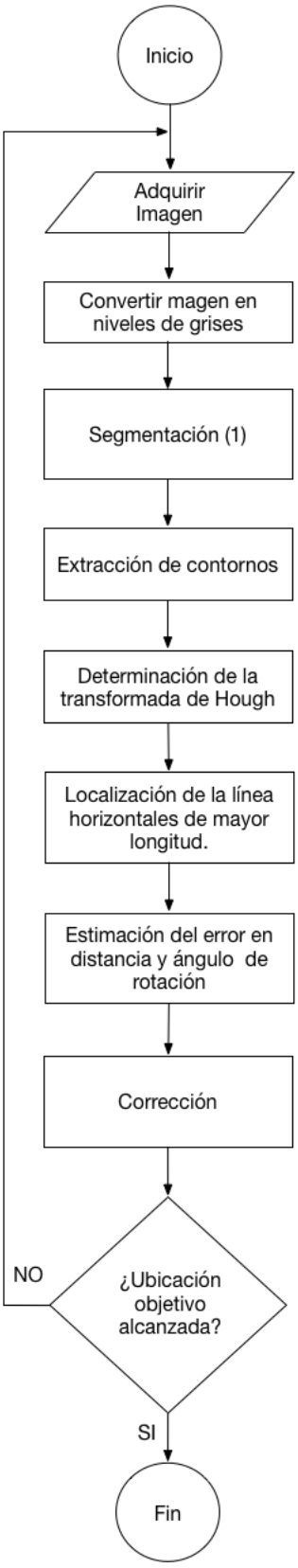
(d) Error en rotaciones negativas

Figura 4.7: Errores en Traslaciones y Rotaciones.

4.2. Estimación de errores en traslación y rotación basada en la Transformada de Hough

La primer estrategia propuesta para la estimación de los errores de traslación y rotación del sistema de locomoción, se basa en la transformada de Hough, que consiste en la detección de líneas, con la cual se buscará una referencia conocida que permita estimar los errores en traslación y rotación generados al desplazar el robot desde un punto inicial hasta uno final, de manera que sea posible reorientarlo dentro del entorno estructurado.

Para la implementación de esta estrategia se diseña una serie de etapas para la correcta detección de la referencia buscada y a su vez la corrección de los errores en la orientación. Este proceso se muestra en el diagrama de flujo de la Figura 4.8. Donde se muestran las etapas a seguir, que parten desde la adquisición de la información, el acondicionamiento de la misma hasta llegar a la aplicación de la transformada de Hough y la determinación los parámetros necesarios para realizar la corrección correspondiente.



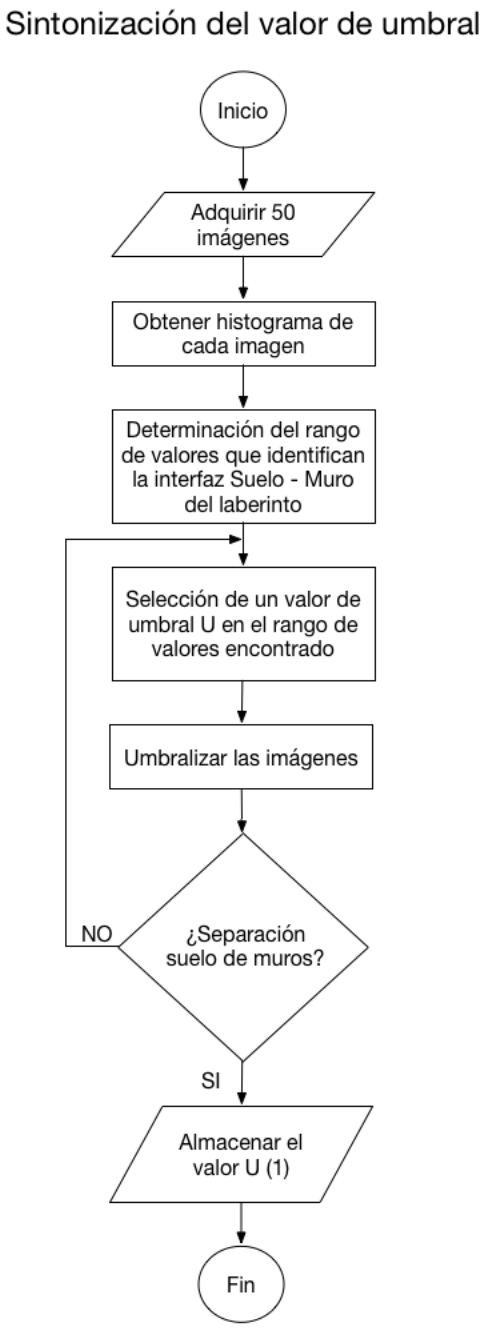


Figura 4.8: Procedimiento de corrección con la detección de líneas obtenidas usando la transformada de Hough.

4.2.1. Segmentación y extracción de contornos

Analizando la información con la que se cuenta, una referencia confiable sería la intersección de los muros con el suelo del laberinto, por lo cual es necesario conocer la interfaz suelo-muro que determina la separación entre ambos objetos y con esto contar con una referencia conocida. Con la umbralización se separa el suelo de los muros, definiéndose perfectamente las interfaces de interés.

Para la umbralización, es necesario un nivel de umbral que determine la separación entre objetos, para ello se analizó el histograma de 50 imágenes adquiridas en condiciones de iluminación estandarizadas, encontrando un rango de valores donde es posible diferenciar los objetos en la imagen. En la Figura 4.9 se pude apreciar que el rango entre los 150 y 220, es posible diferenciar el suelo de los muros del laberinto, por lo tanto se selecciona un valor de 200 y los resultados de la aplicación se muestran en la Figura 4.10a, donde, se muestran las interfaces entre las paredes y el piso del laberinto.

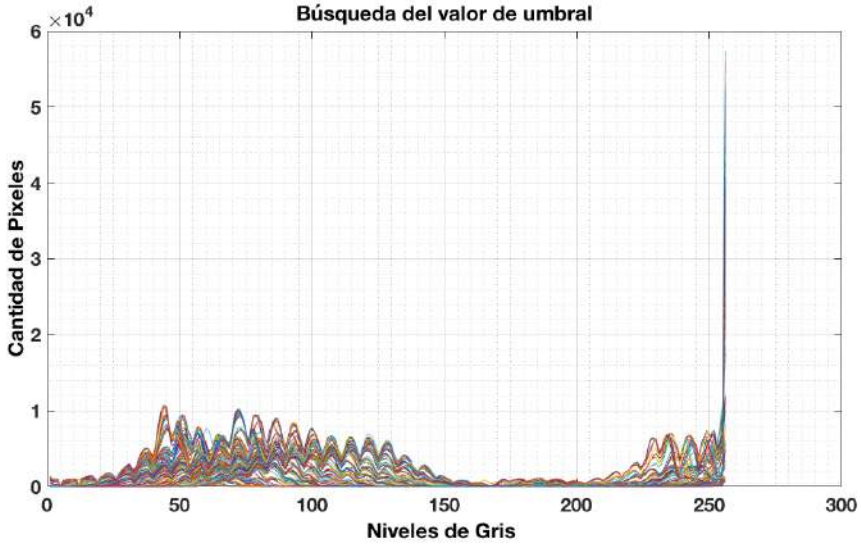
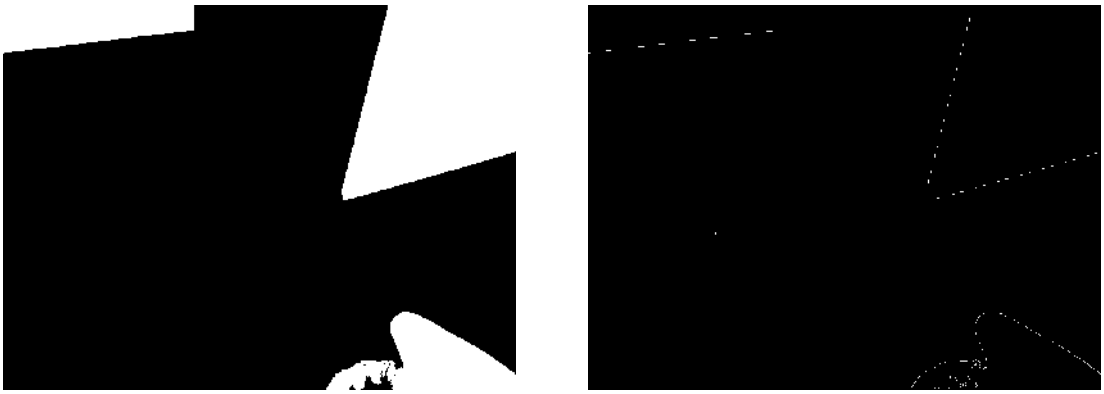


Figura 4.9: Histogramas de imágenes

La detección de contornos se realiza mediante el algoritmo de Canny, el cual utiliza la detección de bordes mediante el criterio de la primera derivada, que toma el valor de cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad.

El algoritmo de Canny puede mostrarse complejo de implementar, afortunadamente, al ser una herramienta ampliamente utilizada por los resultados que ofrece, el algoritmo se encuentra implementado y optimizado en diversos lenguajes de programación tales como C++ y Python con OpenCV. Los resultados de la extracción de contornos se muestran en la Figura 4.10b, donde se muestran las interfaces buscadas con las que se realiza la detección de líneas mediante la transformada de Hough.



(a) Segmentación de la imagen (b) Detección de contornos

Figura 4.10: Preprocesamiento de la imagen.

4.2.2. Detección de líneas con la transformada de Hough

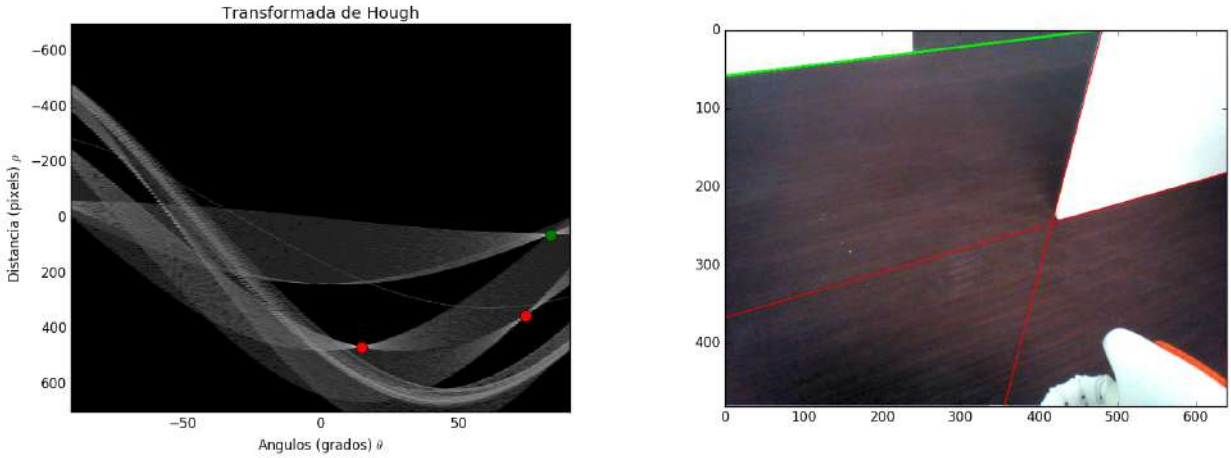
Con las imágenes segmentadas, ahora es posible aplicar la transformada de Hough. La transformada de Hough proporciona una matriz bidimensional de los valores máximos (Figura 4.11a) representan las líneas en la imagen original (Figura 4.11b).

Los resultados obtenidos son correctos, sin embargo, no todos son de utilidad para realizar la reorientación del robot, por lo tanto, de los puntos maximos (o mas brillantes) se realiza una selección de los puntos con la mayor probabilidad de ser una línea horizontal de referencia es decir, se seleccionan los de ángulos cercanos al margen izquierdo o derecho del espacio de Hough.

Con el vector de puntos encontrado cerca de los márgenes izquierdo y derecho, se puede seleccionar cualquiera porque la diferencia en distancias y ángulo de las líneas que representan es despreciable, y por practicidad se elige de manera sistemática la primera (el punto que aparece en primer lugar en el vector de puntos encontrado). En la Figura 4.11 se puede observar la detección de la líneas, tanto en el espacio de Hough (4.11a) como en la imagen original (4.11b) donde en color rojo se muestran todas las líneas detectadas, mientras que la línea horizontal de referencia se destaca en color verde.

4.2.3. Estimación de errores y correccción

En la estimación de los errores de desplazamiento y rotación para su respectiva corrección, es necesario conocer la relación de los parámetros obtenidos por el software con el mundo real. Por lo tanto, para conocer la relación existente entre los pixeles de la imagen capturada por la cámara del robot y las distancias en centímetros del mundo real, es necesario una aproximación que permita conocer la relación entre ellas. El método desarrollado en [Rodríguez Santiago, 2013] es utilizado como referencia para encontrar los parámetros de interes.



(a) Líneas detectadas en el espacio de Hough

(b) Líneas detectadas en la imagen original

Figura 4.11: Aplicación de la transformada de Hough

El procedimiento empleado consiste en la captura de múltiples imágenes colocando al robot en el laberinto frente a un muro a diferentes distancias $(100cm, 90cm, 80cm, ..., 10cm)$ respecto de éste, en la cual el robot captura de imágenes. En seguida se calcula la transformada de Hough para cada una de las imágenes adquiridas. De esta manera se obtienen los parámetros ρ y θ de cada línea presente en las imágenes, considerando solo el parámetro ρ, ya que el robot no ha realizado rotaciones, la variación en el parámetro θ de las líneas detectadas es despreciable.

Conociendo el parámetro ρ para las diferentes distancias, se busca una aproximación polinomial que mejor describa el comportamiento de los datos experimentales. El polinomio que describe mejor la relación entre pixeles y distancias se presenta en la ecuación 4.1, donde $D(cm)$ corresponde a la distancia real respecto a la referencia, en función de la cantidad de pixeles ρ. La gráfica de la Figura 4.12 muestra esta ecuación (negro) y los datos experimentales (rojo).

$$D = 0.0003\rho^2 - 0.3251\rho + 100.0675 \tag{4.1}$$

Para determinar la orientación del robot, se busca una aproximación polinomial que relaciona el ángulo de la transformada de Hough y el ángulo real al cual se encuentra orientado el robot en el entorno estructurado respecto a la línea de referencia.

De forma similar a la búsqueda de la relación para distancias el robot es ubicado a múltiples distancias respecto a la línea horizontal de referencia, y se realizan rotaciones de la cámara del robot en un rango de $\pm 45^o$ capturando imágenes cada grado. Los datos obtenidos describen correctamente la orientación respecto a la referencia establecida, por lo cual es posible determinar una expresión matemática que describa este comportamiento. La función definida en 4.2 describe la relación buscada. Donde α_{real} representa el ángulo real al cual se encuentra el robot respecto a la referencia, en función del ángulo θ determinado por Hough.

$$\alpha_{real} = 1.5441\theta - 139.4118 \tag{4.2}$$

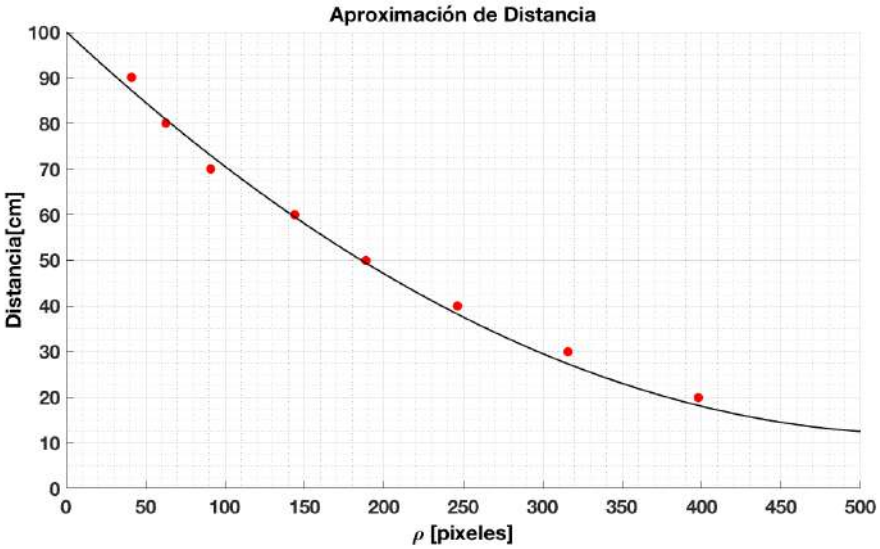


Figura 4.12: Aproximación para distancias

Con esta aproximación ahora es posible determinar el ángulo de orientación real del robot y reorientar al robot correctamente. La gráfica de la Figura 4.13, muestra en línea la aproximación mientras que en puntos los datos experimentales.

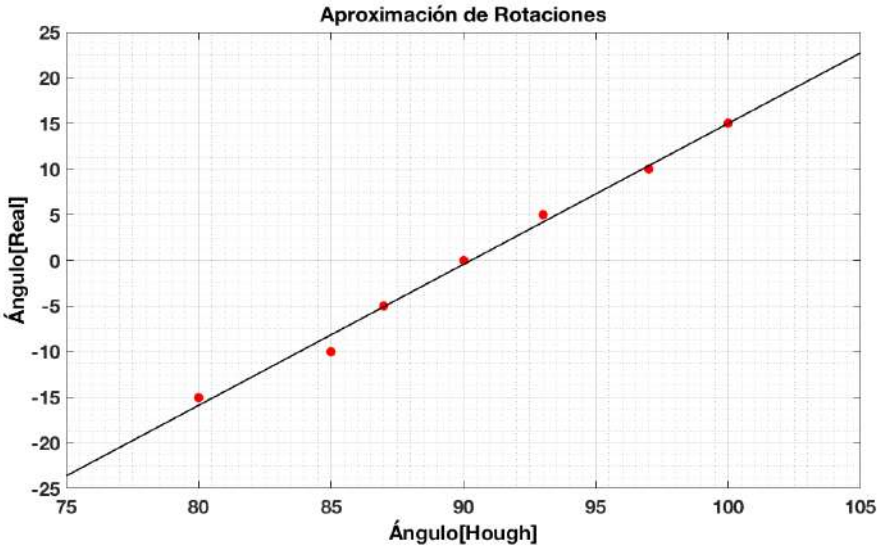


Figura 4.13: Aproximación para rotaciones

Las relaciones matemáticas encontradas, la detección y clasificación de líneas se implementan los respectivos módulos de corrección de los errores de desplazamiento y rotación, la cual se muestra en el Apéndice B. Cabe destacar que esta corrección se puede llevar a cabo solamente cuando la referencia se encuentra presente en la escena capturada por el robot.

4.3. Estimación de errores en traslación y rotación basada el Registro de Imágenes

La segunda técnica propuesta para estimación y correccion de los errores en el desplazamiento del robot es mediante el registro de imágenes, que consiste en la búsqueda de los parámetros de transformación que realicen la correspondencia entre dos imágenes una de referencia (I_R) adquirida en la posición inicial del robot y la segunda la imagen a transformar (I_T) adquirida en la posición alcanzada después del desplazamiento del robot.

De acuerdo a las características presentadas en las imágenes adquiridas del escenario estructurado, las paredes son totalmente blancas y la variación en su textura no es visible. Sin embargo, el suelo del laberinto, al ser de madera en color obscura si es posible observar una variación de textura entre imágenes, lo cual podrá aportar mayor información para la determinación de los parámetros de transformación. Por lo tanto, se buscará la variación de los niéveles de gris en el suelo del laberinto para la estimación de los parámetros de transformación. El proceso para la corrección mediante el registro de imágenes se muestra en el diagrama de flujo de la Figura 4.14.

4.3.1. Acondicionamiento de las imágenes

La imagen de referencia I_R es capturada en la posición inicial del robot con la cámara ubicada a 0^o con respecto al plano horizontal (Figura 4.15a) y el centro de la región observada corresponde al punto donde se desea que el robot llegue. La imagen a transformar I_T es capturada después de que el robot se ha trasladado, intentando alcanzar la posición deseada. Esta última es adquirida con la cámara inclinada 20^o (Figura 4.15b).

Puesto que ambas imágenes I_R e I_T son adquisiciones del suelo con la cámara a diferentes ángulos con respecto al plano horizontal, hay una diferencia en perspectiva que es necesario corregir antes de la implementación del algoritmo de registro de imágenes. De manera que entre las dos imágenes adquiridas solamente halla diferencias entre traslaciones y rotación.

Corrección de la perspectiva

Para poder registrar las imágenes I_R e I_T utilizando un modelo de transformación que incluya solamente desplazamientos en x y y y rotaciones es necesario corregir la diferencia en perspectiva entre ambas imágenes. Para ello se correge la perspectiva en ambas imágenes, lo cual se realiza aplicando las matrices H_{ref} y H_{transf} (ecuación 4.3) a cada imagen respectivamente. Los parámetros de la matriz de trasformación se determina de forma empírica, sintonizando los valores de la matriz hasta que un patrón cuadrado sobre el suelo se observará como un patrón cuadrado en las imágenes y estos fueran del mismo tamaño. Los resultados se muestran en la Figura 4.16, donde se observa la corrección de la perspectiva de ambas imágenes, en el Apéndice C se presenta el código correspondiente a la corrección de la perspectiva.

$$H_{ref} = \begin{bmatrix} 1.4545 & 0.0116 & 0 \\ 0 & 2.0625 & 0 \\ 0.00006 & -0.00363 & 1 \end{bmatrix}; \qquad H_{transf} = \begin{bmatrix} 1.0679 & -0.0122 & 0 \\ 0.0122 & 1.1699 & 0 \\ 0.00002 & -0.00165 & 1 \end{bmatrix} \tag{4.3}$$

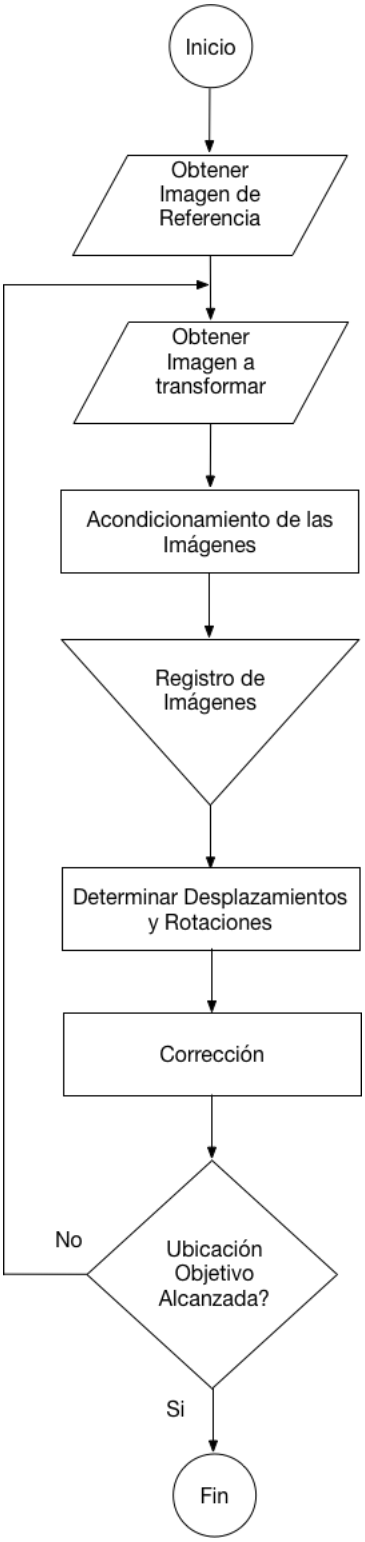
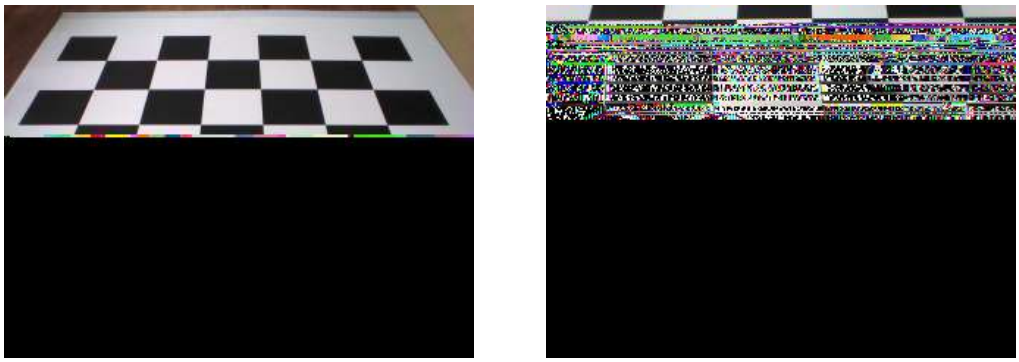
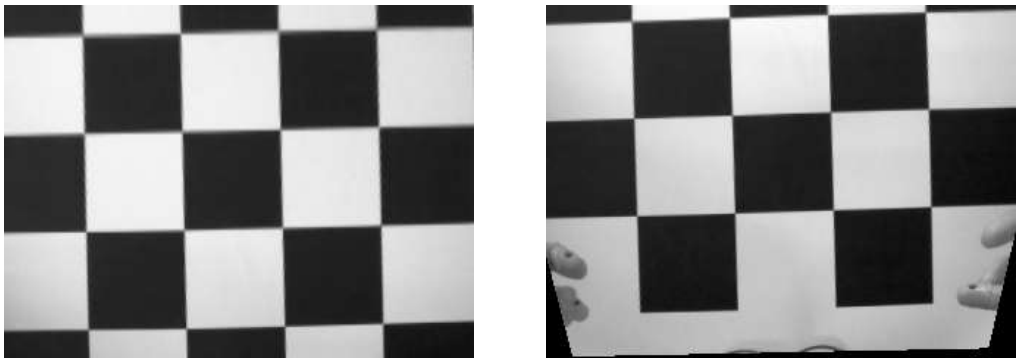


Figura 4.14: Estimación de errores con el Registro de Imágenes



(a) Imagen de referencia (b) Imagen a transformar

Figura 4.15: Información para el Registro de Imágenes



(a) Posición inicial corregida (b) Posición final corregida

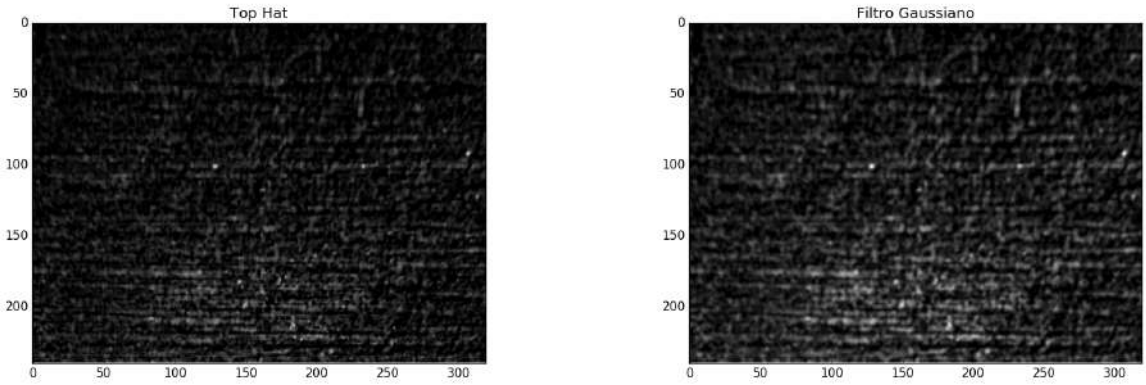
Figura 4.16: Corrección de perspectiva

Filtrado

Dada las características de la textura del suelo del laberinto, es necesario resaltar los detalles en las imágenes, para ello, una técnica conocida y muy utilizada en morfología matemática para resaltar objetos localmente brillantes (u oscuros) de una imagen en niveles de gris es llamada "Top-Hat" [Pastore et al., 2007]. En este caso se desea resaltar los objetos brillantes en la textura del suelo del laberinto, por lo tanto, la diferencia entre el mapa original y el filtrado por apertura filtra las imágenes eliminando y aumentando el contraste de algunos elementos para la detección de los objetos brillantes de interés en la imagen [Ortiz Zamora, 2002].

Por otra parte, es necesario suavizar la imagen con la intención de mejorar la imagen de manera que el ruido presente en la imagen se elimine [Díaz et al., 2006]. Para ello la aplicación de un filtro Gaussiano para alisar la imagen es lo más indicado.

Los resultados de la aplicación de este acondicionamiento de las imágenes se muestran en la Figura 4.17a y 4.17b, donde se puede observar el realce de la textura en la imagen con el "Top-Hat" así como el alisado de la imagen por medio de un filtro gaussiano para la eliminación del ruido.



(a) Aplicación del Top-Hat

(b) Aplicación del filtro Gaussiano

Figura 4.17: Corrección de perspectiva

4.3.2. Registro de imágenes

El registro de imágenes tiene por objetivo determinar los parámetros de transformación que pongan en correspondencia dos imágenes. Para lograrlo, se emplea una búsqueda exhaustiva, que se encarga de estimar los parámetros de transformación que minimice el criterio de Woods entre las dos imágenes. Para lo cual, es necesario realizar todas las combinaciones posibles de los parámetros de transformación y la comparación elemento a elemento, es decir, I_T se modifica con una combinación de los parámetros buscados y es comparada con I_R, determinando la similitud entre ellas mediante el criterio de Woods.

Este proceso tendrá un alto coste computacional, por lo cual, es necesario aplicar la estrategia piramidal para acelerar la búsqueda de los parámetros.

Las etapas necesarias para la implementación del registro de imágenes en combinación con la estrategia piramidal utilizando el criterio de Woods se muestra en el diagrama de flujo de la Figura 4.18. Donde se muestra como en una primera iteración se crea una pirámide de imágenes y se establecen los valores iniciales de búsqueda, para posteriormente emplear la búsqueda exhaustiva, utilizando el nivel más alto (imágen más pequeña) seleccionando los parámetros de transformación que minimicen el error de similitud del criterio de Woods y de acuerdo a éste, los parámetros de búsqueda son escalados para la búsqueda en el siguiente nivel de la pirámide.

Con el escalamiento la nueva búsqueda se realiza en la vecindad de los parámetros de transformación previamente estimados, con lo cual los parámetros buscados son refinados y el tiempo de procesamiento es minimizado. De esta forma se continúa en todos los niveles de la pirámide hasta llegar al último nivel (tamaño original de la imagen), y de esta manera determinando los parámetros de transformación buscados y finalizando el registro de imágenes.

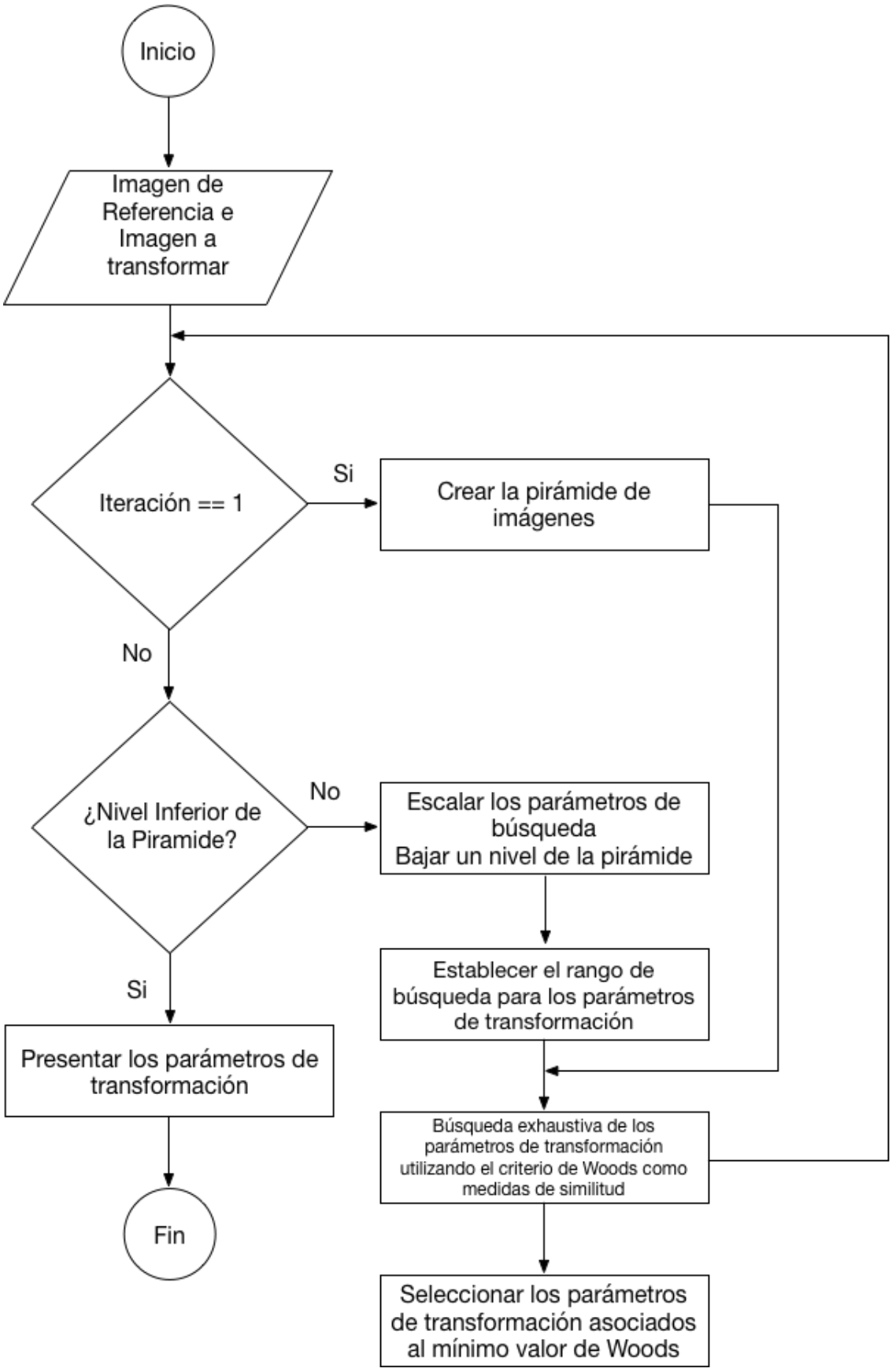


Figura 4.18: Procedimiento para el Registro de Imágenes

Estrategia Piramidal

La estrategia piramidal, como su nombre lo indica, crea una pirámide de imágenes, donde en el nivel superior se encuentra la imagen de menor tamaño y en la base de la pirámide se ubica la imagen original. La representación de esta reducción se muestra en la Figura 4.19.

Esta reducción está en función de la información que la imagen puede aportar, por lo tanto, después de múltiples pruebas experimentales se encuentra el número de niveles al cual se puede llegar en la reducción. La implementación de la estrategia se realiza haciendo uso de las funciones de OpenCV.

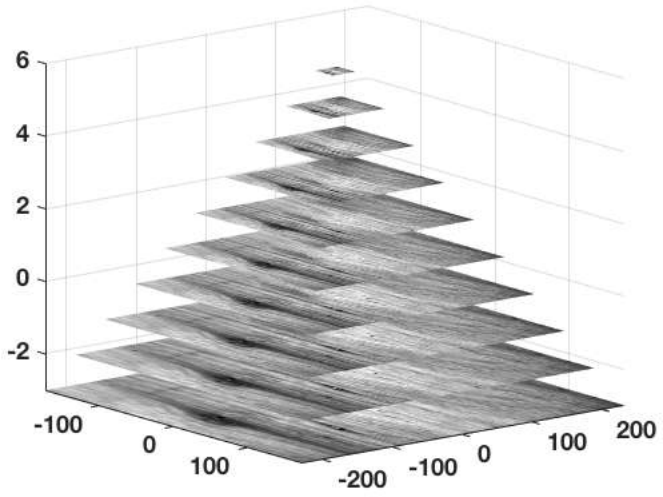


Figura 4.19: Reducción de imágenes con la estrategia piramidal

Estimación de errores y correccción

Al inicio de este método se realizó la corrección de perspectiva y una corrección en escalamiento, de tal forma que la cantidad de pixeles que representa un objeto en I_R sea la misma en I_T. Con esto se obtuvo la relación *pixeles-centímetros* que permite conocer la cantidad de pixeles equivalente a una distancia del mundo real, en este caso $70 pixeles \backsim 10.8 cm$. Con la misma corrección se encuentra la relación de ángulos $1:1$ es decir, que el ángulo proporcionado por el registro de imágenes es el mismo que el ángulo de orientación real del robot. Con esta relación se determinan los movimientos que el robot debe realizar para reorientarse respecto a la posición deseada.

Capítulo 5

Pruebas y resultados

En un inicio el robot es ubicado en una posición de partida **A** a una distancia d_i respecto a la posición **B**. Sin embargo, el robot es incapaz de alcanzar dicha posición y se ubica en alguna posición incorrecta **C** (ver Figura 5.1). Es entonces cuando el sistema desarrollado deberá determinar las distancias d_x, d_y sobre los ejes x_w, y_w respectivamente así como el ángulo de rotación r_z al rededor del eje z_w (ver Figura 4.2), necesarios para reorientar al robot y alcanzar la posición **B** utilizando como referencia una interfaz suelo-muro o la textura del suelo para la técnica basada en la transformada de Hough y el registro de imágenes respectivamente.

Para determinar el funcionamiento y los alcances del sistema desarrollado se implementan diferentes pruebas dentro de uno de los pasillos del entorno estructurado donde se determinan los parámetros necesarios para la corrección de la posición y orientación del robot y de esta manera poder alcanzar la posición final deseada.

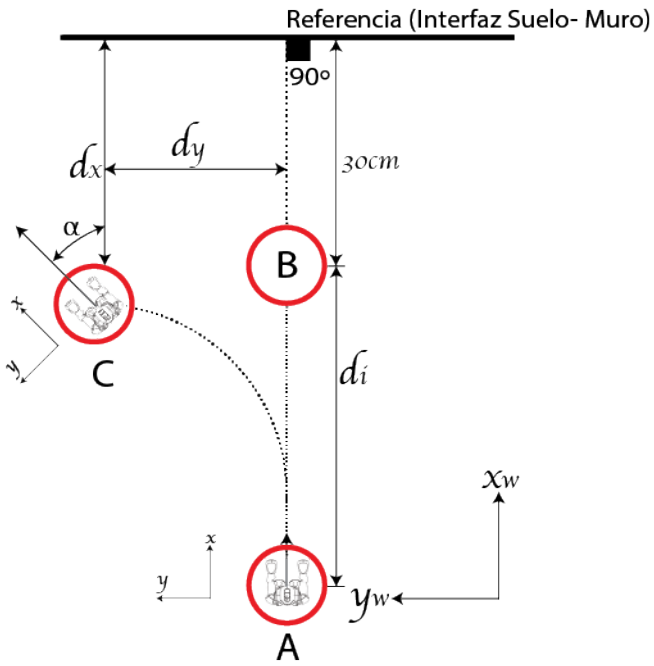


Figura 5.1: Movimientos del robot.

5.1. Corrección de errores de desplazamiento basado en la transformada de Hough

El método implementado basado en la transformada de Hough permite determinar el error de desplazamiento sobre el eje x_w del laberinto, es decir la distancia d_x respecto a la línea de la interfaz suelo-muro. Por lo tanto, aun cuando el robot también realice desplazamientos sobre el eje y_w, sólo se evalúa el desempeño para la estimación y corrección de errores en de desplazamiento sobre el eje x_w y ángulos de rotación r_z alrededor del eje z_w. En la sección 5.1.1 se presenta el protocolo de pruebas para la validación del método de corrección de desplazamiento basado en la transformada de Hough, y en la sección 5.2.1 se presenta el protocolo de prueba correspondiente a la metodología basada en el registro de imágenes.

5.1.1. Protocolo de pruebas para la metodología basada en la transformada de Hough: Distancias

Considerando la distancia conocidas a desplazarse y tomando en cuenta los errores de desplazamiento determinados en la sección 4.1.2, se colocó el NAO en diferentes ubicaciones (ver Figura 5.2) a distancias conocidas respecto al punto **B** y sin considerar la rotación. Los resultados promedio de 10 repeticiones obtenidos con el algoritmo basado en la transformada de Hough por cada ubicación, se muestran en la Tabla 5.1 donde la segunda y tercer columna muestran las distancias d_x y d_y respecto a la interfaz suelo-muro de referencia a la cual el robot se encuentra, mientras que en la cuarta columna se muestran las distancias promedio estimadas obtenidas por el algoritmo implementado, así mismo en la última columna se muestran los errores encontrados, es decir la diferencia entre la distancia real (en la dirección del eje x_w) y la determinada por el algoritmo implementado.

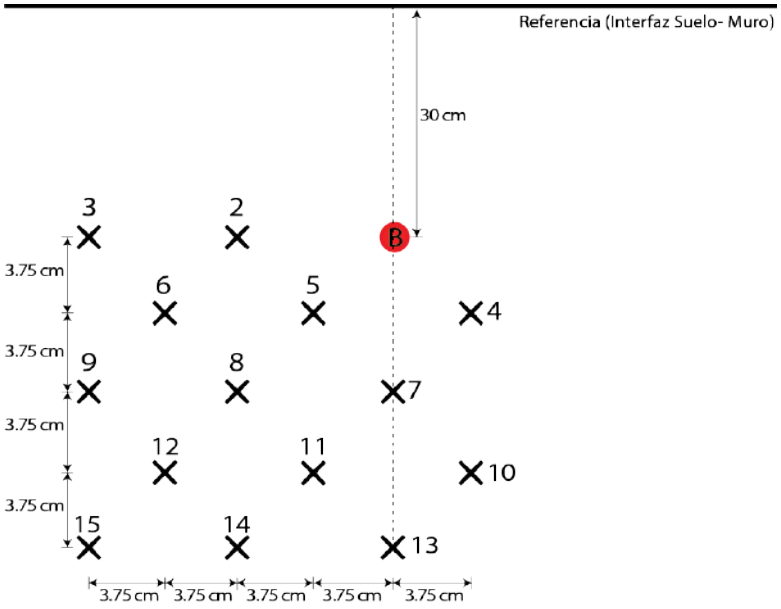


Figura 5.2: Escenario de pruebas para la metodología basada en la transformada de Hough.

Después de estas pruebas se obtiene una media del error de 6.7 cm, una desviación estándar de 1.5 cm y errores máximos y mínimos de 8.7 cm y 4.1 cm respectivamente presentando un tiempo promedio de procesamiento de 3.85 ms. En este experimento solo se muestran los resultados de las pruebas realizadas en ubicaciones sobre el eje y_w positivo ademas de dos ubicaciones del lado sobre el eje y_w negativo ya que de acuerdo a los resultados obtenido en la sección 4.1.2, el robot siempre se desvía en esta dirección, sin embargo, la técnica desarrollada también puede ser aplicada en ambos sentidos del eje y_w.

Con los resultados obtenidos se demuestra que el método basado en la transformada de Hough resulta sumamente confiable para la determinación de los errores en el desplazamiento en la dirección del eje x_w ya que al presentar un error máximo en la ubicación más lejana (13) con respecto al punto **B**, aún es posible corregir el error realizando más iteraciones de la técnica y con ello disminuir el error de ubicación.

Tabla 5.1: Estimación de distancias con respecto al punto **B**

N^o	d_x Real	d_y Real	d_x Hough	E_x
1	30.0	0.0	25.0	5.0
2	30.0	7.5	25.9	4.1
3	30.0	15.0	25.1	4.9
4	33.8	-3.75	27.3	6.5
5	33.8	3,75	27.4	6.4
6	33.8	11.25	28.7	5.0
7	37.5	0.0	29.1	8.4
8	37.5	7.5	30.1	7.4
9	37.5	15.0	30.5	7.0
10	41.3	-3.75	33.7	7.5
11	41.3	3.75	33.2	8.1
12	41.3	11.25	35.9	5.4
13	45.0	0.0	36.3	8.7
14	45.0	7.5	36.7	8.3
15	45.0	15.0	36.5	8.5
Promedio				6.7
Varianza				2.4
Desv. Estándar				1.5
Mínimo				4.1
Máximo				8.7

Medidas en centímetros

5.1.2. Protocolo de pruebas para la metodología basada en la transformada de Hough: Rotaciones

Nuevamente se considera la distancia conocida (d_x) y tomando en cuenta la rotación involuntaria del robot durante su desplazamiento y que fue caracterizada en la sección 4.1.2, el robot NAO es ubicado en las posiciones $1 - 15$ (ver Tabla 5.2) considerando únicamente rotaciones en un rango de $\pm 45^o$ con intervalos de 5^o en cada ubicación y realizando 10 repeticiones en cada una de las ubicaciones. En la Tabla 5.2 se muestra el promedio de los resultados obtenidos. Donde la segunda columna muestra los ángulos reales a los cuales se encuentra orientado el robot, mientras que la tercer columna representa los ángulos de rotación determinados por el algoritmo implementado, así mismo en la última columna se presentan los errores encontrados, es decir la diferencia entre el ángulo real y el estimado por el algoritmo desarrollado.

Es posible observar que con el método basado en la transformada de Hough, en el rango de distancias establecido, el algoritmo implementado puede estimar el ángulo de orientación del robot (α) para la corrección de rotación en un rango de $\pm 15^o$. Con las pruebas realizadas se obtiene un error promedio de 1.1^o, desviación estándar de 0.7^o y errores máximos y mínimos de 2.4^o, 0.1^o presentes en los ángulos 15^o y 10^o respectivamente.

Tabla 5.2: Corrección de rotaciones

N^o	Ángulo Real	Ángulo Hough	E_z
1	-45.0	-	-
2	-30.0	-	-
3	-20.0	-	-
4	-15.0	-13.9	1.1
5	-10.0	-8.7	1.3
6	-5.0	-5.6	0.6
7	0.0	-0.4	0.4
8	5.0	3.6	1.4
9	10.0	9.9	0.1
10	15.0	12.6	2.4
11	20.0	-	-
12	30.0	-	-
13	45.0	-	-
Promedio			1.1
Varianza			0.6
Desv. Estándar			0.7
Mínimo			0.1
Máximo			2.4

Medidas en grados

5.1.3. Corrección de distancias y rotaciones

Nótese que en las pruebas anteriores (secciones 5.1.1 y 5.1.2) el error de desplazamiento sobre el eje y_w no es considerado, ya que la técnica basada en la trasformada de Hough solo permite determina las distancias perpendiculares respecto a la línea de referencia. Sin embargo, ejecutando los algoritmos de estimación y corrección del ángulo de rotación y estimación y corrección de distancia d_x de manera iterativa, mientras el robot se desplaza sobre la distancia del punto **A** al punto **B** se verificó empíricamente que el robot no se desvía en la dirección del eje y recorriendo trayectorias rectas. Por lo tanto, el NAO es colocado en diferentes posiciones respecto a la posición final deseada. Estas distancias son recorridas en 10 repeticiones cada una de ellas, determinando el error de ubicación respecto al punto **B**. Los resultados son mostrados en la Tabla 5.3 donde la segunda columna muestra las posiciones iniciales en las cuales el robot ha sido colocado, la columnas tres y cuatro indican los errores en la estimación de la ubicación final en la dirección de los ejes x_w y y_w respectivamente con un promedio de 1.7 cm y 4.4 cm respectivamente, la última columna indica el error en el ángulo estimado por el algoritmo con un promedio de 3.3^o.

Tabla 5.3: Corrección de distancias y rotaciones con la transformada de Hough

N^o	D Inicial	E_x	E_y	E_z
1	100	2.3	4.5	3.1
2	90	1.7	4.5	3.3
3	80	1.5	3.5	2.6
4	70	2.4	3.5	2.4
5	60	1.4	4.4	3.4
6	50	1.3	4.0	3.1
7	40	1.3	5.5	4.4
8	30	1.6	4.5	3.3
9	20	2.1	5.5	4.4
Promedio		1.7	4.4	3.3
Varianza		0.2	0.5	0.5
Desv. Estándar		0.4	0.7	0.7
Mínimo		1.3	3.5	2.4
Máximo		2.4	5.5	4.4

Distancias en centímetros y ángulos en grados

Los resultados obtenidos también se representan en la Figura 5.3 en donde la zona en verde indica en donde si fue posible realizar la estimación de posición y orientación con respecto al punto **B**, en amarillo se indica la zona en donde los errores fueron moderados pero aún con posibilidades de ser reducidos con más iteraciones de la técnica, por lo cual no se muestran zonas en color rojo.

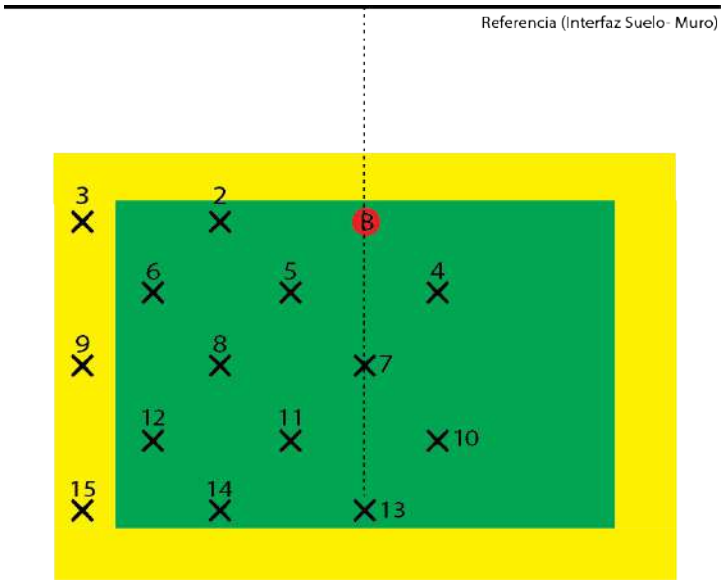


Figura 5.3: Zona de de funcionamiento de la metodología basada en la transformada de Hough.

Los errores presentes son prácticamente imperceptibles, pues la desviación máxima sobre los ejes x_w y y_w son menores al tamaño de un pie del robot, el cual mide 16.0 cm de longitud y 9.0 cm de amplitud. De forma similar ocurre con el error de orientación ya que un ángulo de 4.4^o no presenta complicaciones para que el robot pueda desplazarse de forma segura en el entorno estructurado. El tiempo promedio de procesamiento y corrección de la técnica es de 6.58 ms por cada iteración.

5.2. Corrección de errores con el Registro de Imágenes

El segundo método implementado se basa en el registro de imágenes y permite determinar los errores de desplazamiento y rotación , sobre los ejes x_w, y_w y al rededor del eje z_w respectivamente, es decir las distancias y ángulos necesarios para corregir y reorientar al robot para que este pueda alcanzar la posición **B**, con la orientación perpendicular a la linea de referencia. Para evaluar el desempeño del sistema, el NAO es colocado a una distancia conocida **A**, sobre el eje x_w respecto a la posición **B** desde donde se adquiere la imagen de referencia I_R, con la cabeza del NAO alineada horizontalmente (ángulo $HeadPitch = 0^o$).

Posteriormente el robot avanza intentando alcanzar la posición deseada **B** ubicándose en alguna posición incorrecta **C**. En esta ubicación el robot inclina su cabeza (ángulo $HeadPitch = 20^o$) y adquiere la imagen a transformar I_T y se aplica el registro de imágenes. En el caso ideal en el que el NAO se haya desplazado a la posición **B**, la imagen I_T se encontraría alineada con la imagen I_R, en cualquier otro caso, el registro de imágenes permitirá estimar los parametros de transformación que alineen ambas imagenes , permitiendo a su vez la estimación de las distancias d_x y d_y así como el ángulo de rotación r_z. En las secciones 5.2.1 y 5.2.2 se detallan las pruebas realizadas para la validación del método.

5.2.1. Corrección de y desplazamientos sin rotaciones

Con la distancia a desplazarse y nuevamente retomando los errores en el desplazamiento del robot caracterizados en la sección 4.1.2, el robot es ubicado a distancias conocidas respecto a la posición **B** y sin considerar rotaciones (ver Figura 5.1). Empíricamente se observó que el registro de imágenes tiene un rango de operación más limitado respecto al rango encontrado para la metodología basada en la transformada de Hough durante las pruebas realizadas. Por lo tanto, el rango para las pruebas del registro de imágenes, tomando como origen la posición deseada **B** es 20 cm en la dirección del eje x_w y ± 10 cm en la dirección del eje y_w con pasos de 5 cm (ver Figura 5.4).

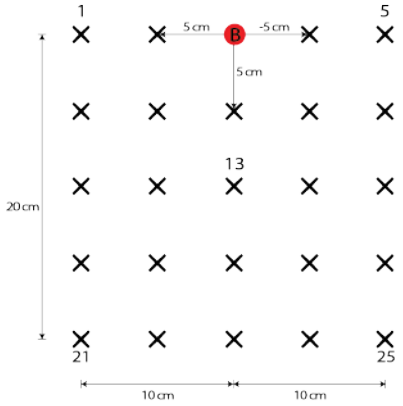


Figura 5.4: Escenario de pruebas para la metodología basada en el Registro de Imágenes.

En la Tabla 5.4 se muestran los resultados obtenidos al realizar 10 repeticiones en cada ubicación y donde la primer columna hace referencia a la numeración de la Figura 5.4, la segunda y tercer columna indican la ubicación real del robot respecto a la posición **B**, mientras que las siguientes dos indican las correspondientes distancias determinadas por la técnica basada en el registro de imágenes y las últimas dos columnas indican los errores correspondientes en cada ubicación del robot. Con los datos obtenidos se puede determinar que el registro de imágenes presenta un error promedio en la estimación de desplazamientos sobre el eje x_w y y_w respectivamente de 1.8 cm y 1.1 cm. Los errores máximos son 3.5 cm y 2.0 cm en las direcciones de los ejes x_w y y_w respectivamente y tiempo promedio de procesamiento de 12.52 s.

5.2.2. Corrección de desplazamientos con rotaciones

Nuevamente el robot es colocado en las ubicaciones indicadas en la Figura 5.4 pero en esta ocasión se consideraron ángulos de $\pm 45^o$. En la Tabla 5.5 se muestran los resultados promedio obtenidos para la determinación del rango de rotación en el cual es posible la estimación de los desplazamientos mediante el registro de imágenes, la segunda columna indica el ángulo real al cual el robot está orientado respecto al eje x_w, la tercer columna indica los ángulos determinados por la técnica implementada y en la última columna se indican los errores correspondientes, obteniendo un rango de rotación de $\pm 5^o$ en donde la técnica permite estimar la rotación.

Tabla 5.4: Errores de desplazamiento mediante la técnica de registro de imágenes y sin incluir rotaciones.

N^o	D_x Real	D_y Real	D_x Registro	D_y Registro	E_x	E_y
1	0.0	10.0	-	-	-	-
2	0.0	5.0	-	-	-	-
3	0.0	0.0	3.5	2.0	3.5	2.0
4	0.0	-5.0	-	-	-	-
5	0.0	-10.0	-	-	-	-
6	5.0	10.0	-	-	-	-
7	5.0	5.0	7.0	4.5	2.0	0.5
8	5.0	0.0	6.0	1.0	1.0	1.0
9	5.0	-5.0	-	-	-	-
10	5.0	-10.0	-	-	-	-
11	10.0	10.0	7.5	8.5	2.5	1.5
12	10.0	5.0	9.0	4.0	1.0	1.0
13	10.0	0.0	9.5	0.5	0.5	0.5
14	10.0	-5.0	-	-	-	-
15	10.0	-10.0	-	-	-	-
16	15.0	10.0	-	-	-	-
17	15.0	5.0	13.5	4.5	1.5	0.5
18	15.0	0.0	18.0	1.0	3.0	1.0
19	15.0	-5.0	-	-	-	-
20	15.0	-10.0	-	-	-	-
21	20.0	10.0	-	-	-	-
22	20.0	5.0	-	-	-	-
23	20.0	0.0	18.5	1.5	1.5	1.5
24	20.0	-5.0	-	-	-	-
25	20.0	-10.0	-	-	-	-
Promedio					1.8	1.1
Varianza					1.0	0.3
Desv. Estándar					1.0	0.5
Mínimo					0.5	0.5
Máximo					3.5	2.0

Distancias y desplazamientosen centímetros

Tabla 5.5: Rango de ángulos de rotación para el registro de imágenes tomado desde la posición 13

N^o	Ángulo Real	Ángulo Registro	E_z
1	-6.0	-	-
2	-5.0	-4.9	0.1
3	-4.0	-3.7	0.3
4	-3.0	-2.9	0.1
5	-2.0	-1.9	0.1
6	-1.0	-0.6	0.4
7	0.0	0.0	0.0
8	1.0	0.4	0.6
9	2.0	1.7	0.3
10	3.0	2.6	0.4
11	4.0	3.6	0.4
12	5.0	4.3	0.7
13	6.0	-	-
Promedio			0.3
Varianza			0.0
Desv. Estándar			0.2
Mínimo			0.0
Máximo			0.6

Ángulos en grados

Por otra parte la Tabla 5.6 muestra el resultado promedio al realizar 10 repeticiones en cada ubicación para la determinación de los desplazamientos con rotación, en las primeras tres columnas se muestran el número de prueba y las ubicaciones reales del robot, mientras que las siguientes dos indican las distancias determinadas por la técnica del registro de imágenes y en las ultimas tres columnas se indican los errores promedio para cada ubicación y rotación respectivamente.

Los resultados obtenidos también se representan en la Figura 5.5 .En verde se indica la zona desde donde si es posible realizar la corrección mediante el registro de imágenes. En amarillo se indican las zonas en donde los errores obtenidos son moderados y en rojo las zonas donde no es posible determinar las distancias y/o rotaciones.

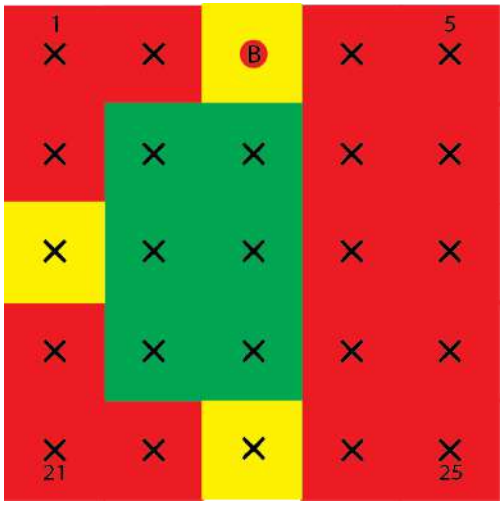


Figura 5.5: Zona de funcionamiento para la metodología basada en el Registro de Imágenes.

Para la estimación de la ubicación del robot cuando se consideran distancias d_x y d_y, así como rotación r_z, el método presenta un error promedio en la estimación de distancias sobre el eje x_w y y_w de 1.9 cm y 1.6 cm respectivamente y 1.2^o en rotaciones. Los errores máximos en la estimación de distancias son de 3.5 cm, 3.0 cm respectivamente y 2.0^o de rotación cuando el robot se encuentra en la posición **B**. El tiempo de procesamiento promedio de 18.71 s.

Tabla 5.6: Corrección de distancias y desplazamientos con rotaciones con el registro de imágenes

N^o	D_x Real	D_y Real	D_x Registro	D_y Registro	E_x	E_y	E_z
1	0.0	10.0	-	-	-	-	-
2	0.0	5.0	-	-	-	-	-
3	0.0	0.0	3.5	3.0	3.5	3.0	2.0
4	0.0	-5.0	-	-	-	-	-
5	0.0	-10.0	-	-	-	-	-
6	5.0	10.0	-	-	-	-	-
7	5.0	5.0	7.0	6.5	2.0	1.5	1.0
8	5.0	0.0	6.0	1.5	1.0	1.5	1.0
9	5.0	-5.0	-	-	-	-	-
10	5.0	-10.0	-	-	-	-	-
11	10.0	10.0	7.5	7.5	2.5	2.5	1.5
12	10.0	5.0	9.0	4.0	1.0	1.0	1.0
13	10.0	0.0	9.5	0.5	0.5	0.5	0.5
14	10.0	-5.0	-	-	-	-	-
15	10.0	-10.0	-	-	-	-	-
16	15.0	10.0	-	-	-	-	-
17	15.0	5.0	13.5	4.0	1.5	1.0	1.5
18	15.0	0.0	18.0	1.0	3.0	1.0	1.0
19	15.0	-5.0	-	-	-	-	-
20	15.0	-10.0	-	-	-	-	-
21	20.0	10.0	-	-	-	-	-
22	20.0	5.0	-	-	-	-	-
23	20.0	0.0	22.5	2.5	2.5	2.5	1.5
24	20.0	-5.0	-	-	-	-	-
25	20.0	-10.0	-	-	-	-	-
Promedio					1.9	1.6	1.2
Varianza					1.0	0.7	0.2
Desv. Estándar					1.0	0.9	0.4
Mínimo					0.5	0.5	0.5
Máximo					3.5	3.0	2.0

Distancias, desplazamientos en centímetros y ángulos en grados

Capítulo 6

Aplicaciones

6.1. Búsqueda de líneas

La estimación de los errores de traslación y rotación durante el desplazamiento del robot basados en la transformada de Hough se ha desarrollado con la cabeza del robot ubicada a 0^o respecto al plano horizontal y la cámara inferior, utilizando una línea horizontal como referencia la cual siempre será visible, por lo tanto, la profundidad de la cámara se limita a una distancia máxima de 100 cm, sin embargo, en el laberinto existen pasillos donde para detectar una línea de referencia las distancias son mayores al limite de la profundidad de la cámara por lo tanto, al robot no le es posible estimar y corregir los errores.

Una solución sería utilizar la cámara superior del robot la cual, presenta una profundidad de campo más amplio pero es necesario realizar un nuevo acondicionamiento de la información para detectar las líneas de interés. Por lo tanto, se propone utilizar la misma cámara inferior del robot y variar su posición de manera que sea posible encontrar una línea de referencia, es decir se realiza una búsqueda de líneas mediante la variación del ángulo $HeadPitch$ de la cabeza del robot. Logrando encontrar líneas a distancias mayores al límite de la profundidad. El cambio de ángulo de elevación de la cámara repercute en la estimación de la distancia a la cual se encuentra la línea de referencia, por lo que la expresión 4.1 ya no es válida, pues ésta aproximación solamente es útil cuando el robot se encuentra a 0^o respecto al plano horizontal.

Para conocer la distancia a la cual se encuentra el robot respecto a la referencia encontrada en función del ángulo de la cabeza ($\Delta\rho$) del robot y la altura h a la cual ésta se encuentra ubicada la cámara del robot, se proponen dos métodos. El primero, con las dimensiones del robot y los ángulos de inclinación de sus cámaras, se realiza un análisis trigonométrico (ver Figura 6.1) para determinar la distancia a la cual se encuentra el robot (6.1).

$$d_x = h \tan(90 - \alpha + \frac{\theta}{2} + \Delta\rho + 1.2) \tag{6.1}$$

El segundo, de forma similar al procedimiento realizado en la sección 4.2.3, se realiza una aproximación de forma experimental para determinar la relación, en este caso, ángulo $HeadPitch$ – distancia d_x.

$$d_x = -1678.3\Delta\rho^3 + 1151.8\Delta\rho^2 - 355.8\Delta\rho + 102.5 \tag{6.2}$$

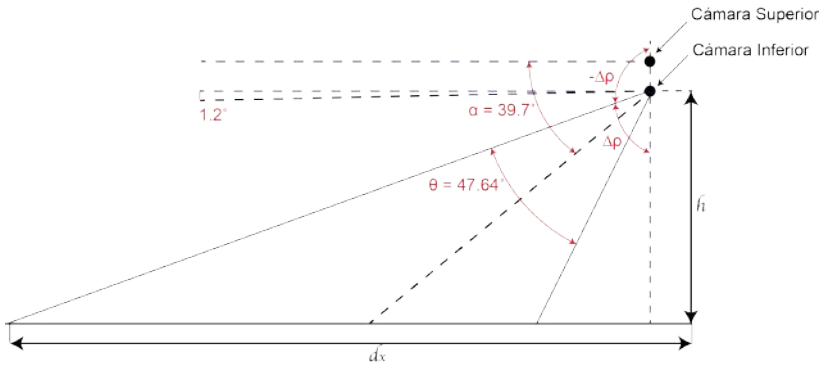
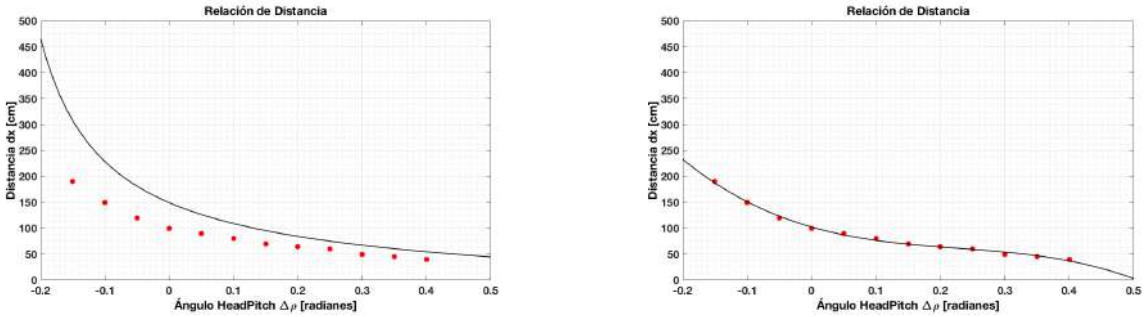


Figura 6.1: Análisis trigonométrico para la determinación de distancia.

Los resultados se muestran en las gráficas 6.2a y 6.2b para el primero y segundo método respectivamente. Donde en color rojo se muestran las distancias reales a las cuales se encuentra ubicado el robot, mientras que en negro se muestra la aproximación que podría describir el comportamiento de los datos. Es posible observar que la relación encontrada con la primera aproximación el error de estimación a la distancia real es mayor en comparación con el segundo método el cual, presenta un error despreciable respecto a los datos reales. Por lo tanto, se utiliza la aproximación 6.2 para conocer la distancia d_x que el robot deberá recorrer para reorientarse respecto a la referencia utilizada. Cabe destacar que esta expresión trabaja de forma simultánea con 4.1, 6.2, el robot avanza verificando en todo momento si es posible conocer la referencia con un ángulo de 0^o, de ser posible se utiliza 4.1 para verifica la distancia. En caso contrario la distancia del robot continúa determinándose mediante la aproximación 6.2.



(a) (b)

Figura 6.2: Aproximación para distancias en función del ángulo HeadPitch $\Delta\rho$

Para corregir la orientación del robot respecto a la referencia, se calcula la diferencia del ángulo determinado por la transformada de Hough (θ) y 1.5708 *radianes* (90^o). Esta diferencia determina el error de rotación e del robot, mismo que el robot rota para corregir su orientación. La forma normalizada para la corrección de la orientación se expresa en la ecuación 6.3.

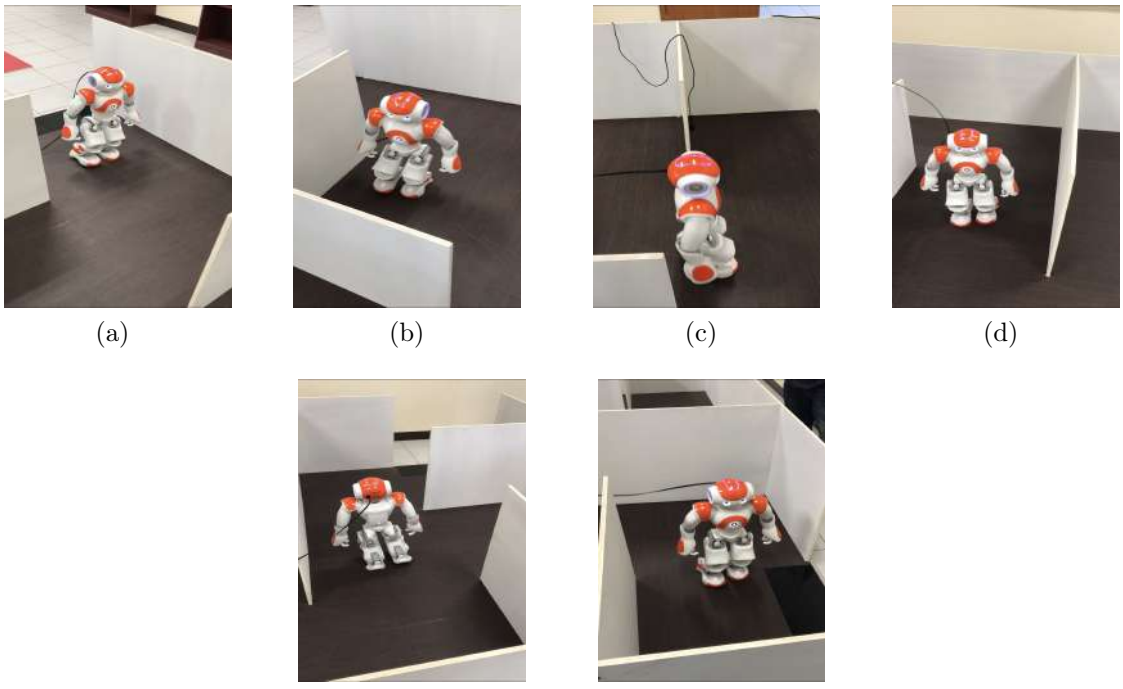
$$e = \frac{2(1.5708 - \theta)}{\pi} \tag{6.3}$$

6.1.1. Resolución de Laberinto

Con la ecuación 6.2 ahora es posible estimar los errores de desplazamientos y rotaciones del robot así como la reorientación en distancias mayores a 125 *cm*. Para la comprobación se realiza la navegación dentro del entorno estructurado corrigiendo los errores de orientación y de ser posible dar solución al laberinto con la detección de líneas, tomando una distancia de seguridad respecto a los muros de 30 *cm* así como rotaciones de 90^o *grados* siempre a izquierda o derecha.

Las imágenes mostradas en la Figura 6.3 muestran al robot durante un recorrido por el laberinto, obteniendo resultados favorables en cuanto a la corrección de los errores de orientación y navegación, es decir, el robot realizó la navegación corrigiendo su orientación respecto a la referencia encontrada y respetando la distancia respecto a los muros incluso en pasillos con longitudes mayores a 200 *cm*. Con esta corrección se logró disminuir el error en los desplazamientos durante el recorrido en el laberinto y tener una navegación fluida.

En cuanto a la resolución del laberinto el robot realizó el recorrido de al menos la mitad de éste sin problemas, sin embargo, a consecuencia de un algoritmo de resolución simple, el robot en repetidas ocasiones recorrió un mismo camino, lo que multiplicó el tiempo de resolución, teniendo un tiempo promedio de 10 *minutos*. Por lo tanto, se sugiere aplicar un algoritmo más sofisticado y la combinación de más sensores del robot que trabajen en conjunto con la cámara para poder obtener mejores resultados.



(a) (b) (c) (d)

(e) (f)

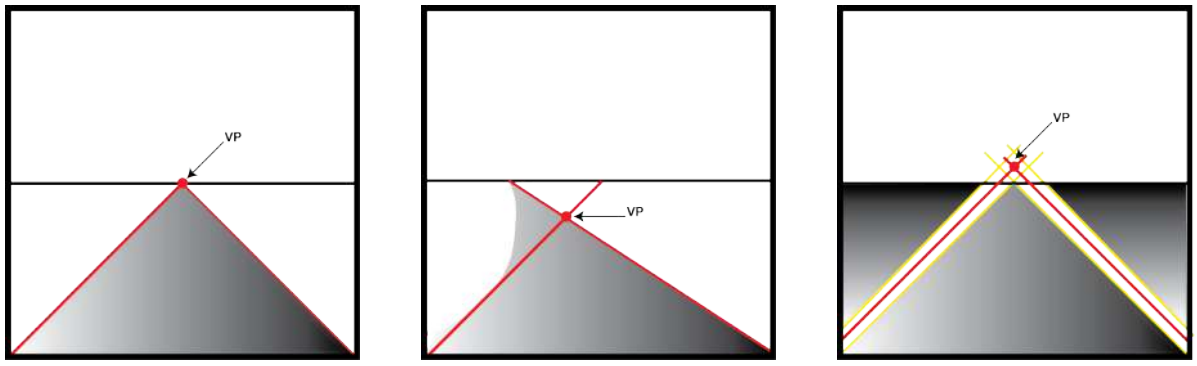
Figura 6.3: Recorrido del laberinto con el robot NAO

6.2. Punto de fuga

Un grupo de líneas en el plano de la imagen que corresponde a un conjunto de líneas paralelas sobre una superficie en el espacio 3D convergen a un punto común en el espacio de la imagen conocido como punto de fuga VP, este punto en común en ocasiones suele encontrarse en el infinito [Moghadam et al., 2012] [Rother, 2002]. Los puntos de fuga se usan ampliamente en una amplia gama de aplicaciones de visión por computadora, como la interpretación tridimensional, donde el problema de encontrar líneas paralelas en 3D se reduce a encontrar puntos de fuga en el plano de la imagen [Ebrahimpour et al., 2012].

En general, en las estructuras arquitectónicas tales como edificios y paredes tienen dos o tres puntos de fuga, que corresponden a diferentes conjuntos de líneas paralelas en dirección vertical u horizontal. Sin embargo, en el contexto de la navegación de robots móviles, independientemente de los tipos de carreteras o pistas o entornos de navegación, se puede determinar un único punto de fuga asociado con las partes rectas más inmediatas en la dirección del eje óptico de las cámaras.

El punto de fuga juega un papel importante como una restricción global para detectar la dirección en la navegación, ya que todas las líneas fronterizas paralelas o los bordes de la pista parecen converger en un único punto de fuga (ver Figura 6.4a). En el caso de una pista curva, se puede estimar un único punto de fuga a lo largo de las direcciones tangentes de las regiones de la pista inmediata frente al móvil (ver Figura 6.4b). En el caso de la existencia de múltiples líneas con pendiente y dirección semejante, el promedio de ellas dará la resultante que se interseque con el punto de fuga (VP). Nótese también que con la proyección de las rectas paralelas el VP puede estar localizado en el infinito por lo cual se encuentra fuera de la ventana de la imagen (ver Figura 6.4c).



(a) (b) (c)

Figura 6.4: Detección del Punto de Fuga

La técnica planteada para la determinación del punto de fuga consta de tres etapas, en la primera se detectan las líneas sobre el plano imagen aplicando la transformada de Hough y almacenando únicamente las líneas que son de interés para nuestra aplicación. Se considera aquellas líneas con una inclinación en el intervalo de $[\pm 30^o, \pm 40^o]$ para líneas presentes en el lado izquierdo (ángulos positivos) y derecho (ángulos negativos) del plano imagen.

En la segunda etapa se determinan las ecuaciones de las rectas, con las cuales se localizará el punto de fuga. Cada línea en el plano imagen es representada mediante dos variables, la pendiente de la recta y el coeficiente de posición (m, b) en coordenadas cartesianas. Mientras que las mismas líneas en el espacio de Hough se representan mediante las variables (ρ, θ) en coordenadas polares. Por lo tanto, podemos representar las rectas mediante las ecuaciones 6.4 y 6.5 en coordenadas cartesianas y polares en el espacio de Hough respectivamnte. Para determinar la ecuación de cada recta se utilizan los parámetros (ρ, θ) determinados por la transformada de Hough, tomando en cuenta que se utiliza la media de las rectas presentes en el plano imagen.

$$\bar{y}_{l,r} = \bar{m}_{l,r} x_{l,r} + \bar{b}_{l,r} \tag{6.4}$$

$$\bar{y}_{l,r} = -\left(\frac{\cos\theta_{l,r}}{\sin\theta_{l,r}}\right) x_{l,r} + \left(\frac{\rho_{l,r}}{\sin\theta_{l,r}}\right) \tag{6.5}$$

El punto de fuga es el punto donde las líneas presentes en el plano imagen convergen, por lo tanto, para nuestra aplicación se busca el punto donde las líneas a izquierda y derecha del robot se intersecan, es decir dónde $y_l = y_r$ y de esta forma determinar las coordenadas del punto de fuga (6.6)

$$VP_x = \frac{\bar{b}_r - \bar{b}_l}{\bar{m}_l - \bar{m}_r}, \quad VP_y = (\bar{m}_l VP_x) + \bar{b}_l \tag{6.6}$$
$$Donde: \ \bar{m}_{l,r} = -\left(\frac{\cos\theta_{l,r}}{\sin\theta_{l,r}}\right), \ \ \bar{b}_{l,r} = \left(\frac{\rho_{l,r}}{\sin\theta_{l,r}}\right)$$

Al estimar la ubicación del punto de fuga, la dirección que el robot sigue puede determinarse de forma aproximada, con lo cual se puede realizar la navegación autónomo, en este caso del robot NAO. En la última etapa se busca la alineación del centro geométrico de la escena capturada con el punto de fuga determinado. Utilizando las coordenadas del punto de fuga es posible la orientación correcta del robot y la navegación trazando trayectorias rectas.

Para la navegación del robot debe considerarse la posibilidad de que el punto de fuga pueda estar localizado en el infinito y en consecuencia estará fuera del plano imagen, lo cual se debe considerar al momento de la implementación ya que será necesario realizar los ajustes necesarios para la alineación y orientación necesaria del robot. En nuestra aplicación la prioridad es que el robot recorra trayectorias rectas por lo tanto se considera alinear el eje x del centro geométrico de la imagen con el eje x del punto de fuga.

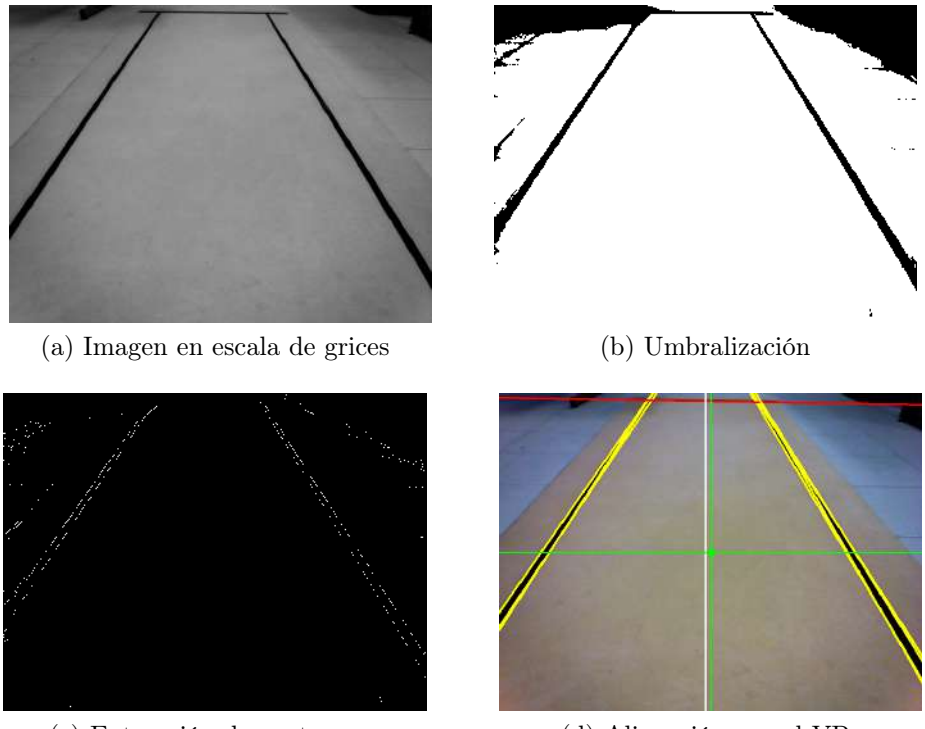
6.2.1. Carrera Individual

Con la detección del punto de fuga es posible realizar la navegación del robot en el laberinto, sin embargo, por la configuración del laberinto la técnica del punto de fuga trabaja de forma secundaria, ya que solo es aplicada cuando se presenta el caso de la no detección de línea horizontal y las líneas presentes en el plano imagen cumplen las restricciones de ángulos para líneas paralelas.

Por lo tanto, para determinar los alcances de la aplicación, esta misma técnica se utiliza para recorrer una pista de carrera individual para robots NAO y utilizando la detección de líneas horizontales se localiza la línea de meta para determinar la distancia respecto a ella y poder recorrer la pista.

La implementación del punto de fuga es puesta a prueba, obteniendo resultados satisfactorios en cuanto a la detección del VP así como la respectiva alineación del robot con las coordenadas del punto de fuga. El robot recorre la pista de carrera individual en un tiempo promedio de 60 s, a pesar de no ser tiempos óptimos, estos aún pueden ser mejorados mediante la programación del robot y poder recorrer la pista para una competencia tecnológica. La Figura 6.5, muestra los resultados del proceso de la implementación, que parte desde el preprocesamiento de la información (6.5a, 6.5b, 6.5c) el cual inicia con la transformación de la información a niveles de grises, umbralización, extracción de contornos y detección de líneas con la transformada de Hough con las cuales localizar el punto de fuga.

En 6.5d podemos observar en color amarillo las líneas para determinar el punto de cruce mismo que se encuentra fuera de la ventana de la imagen, sin embargo, solamente es de interés la coordenada x misma que se indica en color blanco, en color verde se indica el centro geométrico de la escena capturada por la cámara inferior del robot y en rojo se indica la línea de meta que se debe alcanzar.



(a) Imagen en escala de grices

(b) Umbralización

(c) Extracción de contornos

(d) Alineación con el VP

Figura 6.5: Implementación del punto de fuga en la pista de carreras para NAO

6.3. RoboCup SPL

RoboCup es una iniciativa internacional que fomenta la investigación en robótica e inteligencia artificial, a través de diversas competiciones como Robot Soccer, en la cual se incluye una serie de diferentes ligas de fútbol de robots que se centran en diferentes desafíos de investigación.

La liga de plataformas estándar o SPL (por sus siglas en ingles) es una liga de fútbol en la que todos los equipos compiten con robots idénticos que operan de manera totalmente autónoma, es decir, no hay control externo, ni por los humanos ni por las computadoras. La plataforma estándar actual utilizada es el robot humanoide NAO de SoftBank Robotics y los juegos se desarrollan en una cancha de fútbol (Figura 6.6) de 900 *cm* de largo, 600 *cm* de ancho y demás dimensiones que se especifican en la tabla 6.1 [RoboCup, 2017].

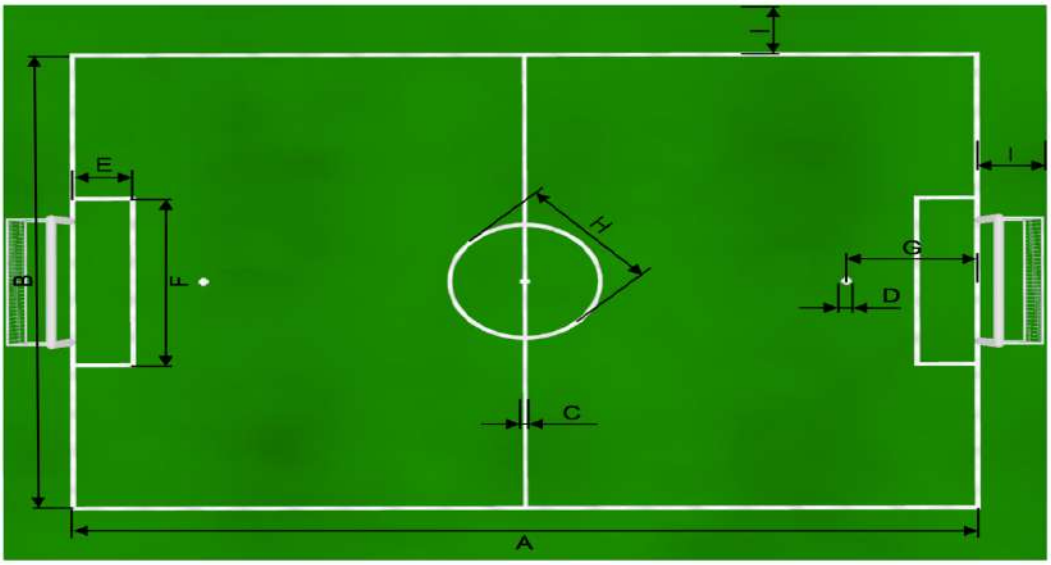


Figura 6.6: Campo de fútbol para RoboCup SPL

Tabla 6.1: Dimensiones del campo de fútbol

ID	Descripción	Longitud (en cm)
A	Longitud del campo	900
B	Amplitud del campo	600
C	Amplitud de línea	5
D	Tamaño transversal del punto penal	5
E	Longitud del área de penalty	60
F	Amplitud del área de penalty	220
G	Distancia transversal de penalty	220
H	Diametro del círculo central	220
I	Amplitud de la banda fronteriza	70

Para competir en la categoría SPL es necesario resolver diversas tareas, de las cuales destacamos la entrada y orientación correcta en el campo de juego. Por lo tanto, con el conocimiento del entorno de desarrollo, se propone una solución basada en la detección de las líneas ya que con la información del entorno esta técnica puede presentar resultados satisfactorios. El proceso desarrollado para la solución de las tareas planteadas se muestra en el diagrama de la Figura 6.7. Las pruebas necesarias se realizan en la cancha de fútbol de los laboratorios de robótica inteligente del instituto de posgrado de la UTM (Figura 6.8), posteriormente esta técnica fue aplicada durante la competencia del torneo RoboCup 2017 celebrado en la ciudad de Nagoya en Japón con el equipo Aztlán.

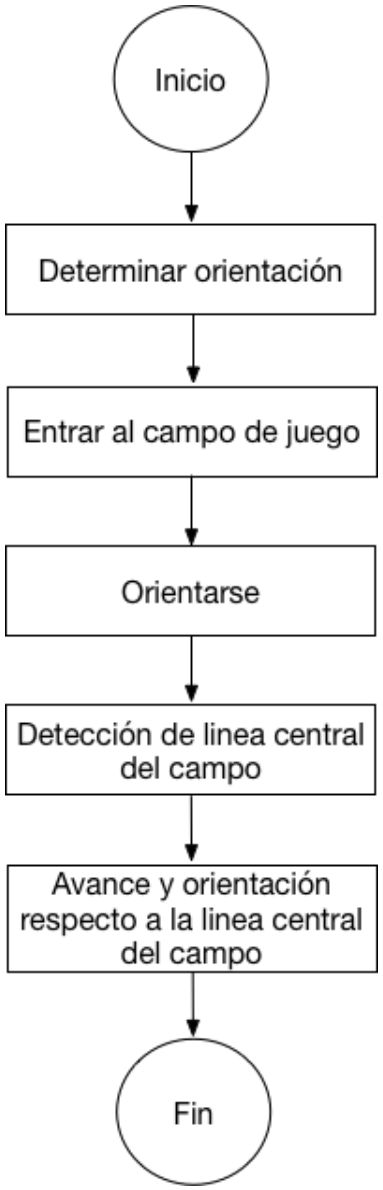
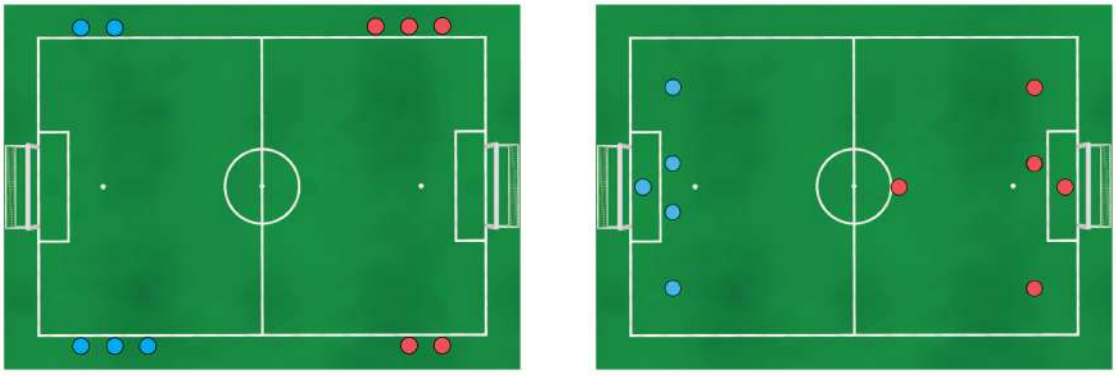


Figura 6.7: Procedimiento para la solución de las tareas

Figura 6.8: Campo de fútbol de laboratorios de posgrado de la UTM

6.3.1. Entrada y orientación en el campo de juego

Antes de iniciar el juego y al ser penalizados los robots son manualmente ubicados fuera de la cancha (Figura 6.9a), desde esta posición los robots deberán entrar al campo para tomar su posición en el campo de juego y poder dar inicio al encuentro (Figura 6.9b) o en su defecto continuar con el partido. En esta aplicación nos centramos únicamente en la entrada y orientación de forma correcta en el campo de juego antes de iniciar el encuentro, es decir el robot al entrar al campo de juego se deberá orientar en dirección de la línea central del campo de juego De manera que este orientado en dirección de la mitad de la cancha del equipo contrario.



(a) (b)

Figura 6.9: Rutina de entrada al campo de juego

De acuerdo a la información del entorno en el cual se encuentra el robot, se propone una solución basada en el número de líneas detectadas a derecha e izquierda del robot (ver Figura 6.12) con lo cual determinar la orientación que el robot deberá tomar dentro del campo de juego, el proceso se muestra en la Figura 6.11 el cual se repite tanto para la detección de líneas en el lateral izquierdo como derecho del robot.

(a) Observación derecha del robot

(b) Observación izquierda del robot

Figura 6.10: Observación del robot en el campo de juego

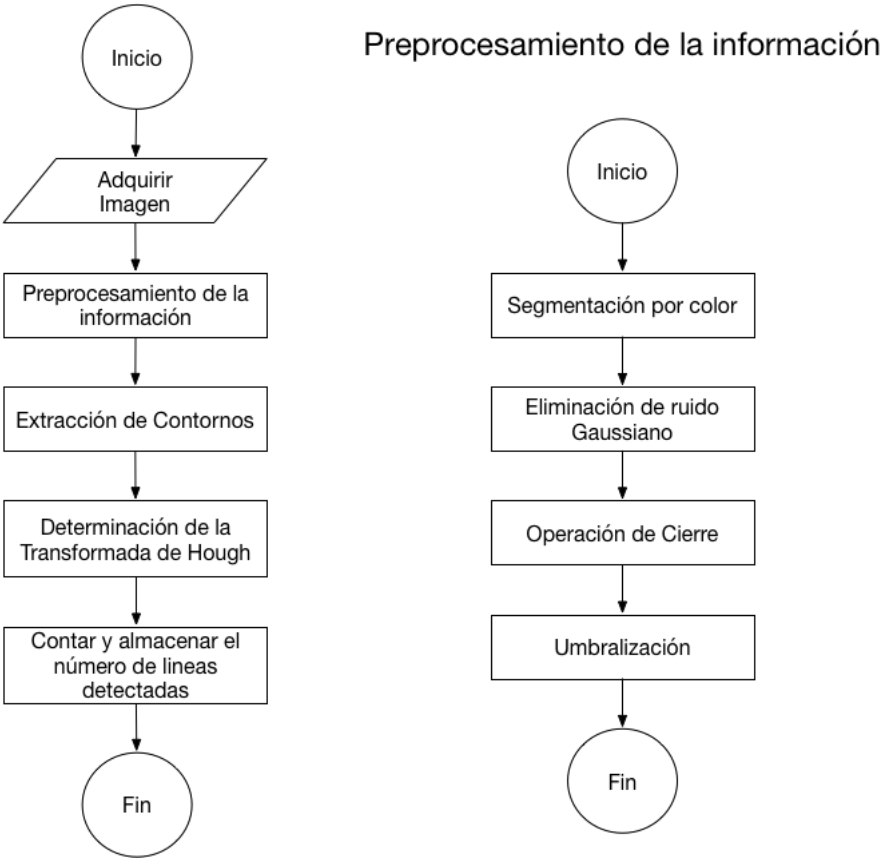


Figura 6.11: Determinación de la orientación del robot en el campo de juego

Para el conteo de líneas el robot rota su cabeza hacia su lateral derecho y posteriormente en el izquierdo y realiza el procedimiento diseñado para la detección de líneas en cada lateral y almacena el número de líneas encontradas y una vez inspeccionado ambos laterales el sentido de giro para la orientación se determina de acuerdo al número máximo de líneas, es decir, si el número de líneas detectadas en el lateral izquierdo es mayor que las líneas detectadas en el lateral derecho, dentro del campo de juego, el robot deberá rotar 90^o en sentido de las manecillas del reloj, por el contrario, si el número de líneas detectadas es mayor en el lateral derecho el robot girara en el sentido contrario a las manecillas del reloj (ver Figura 6.12).

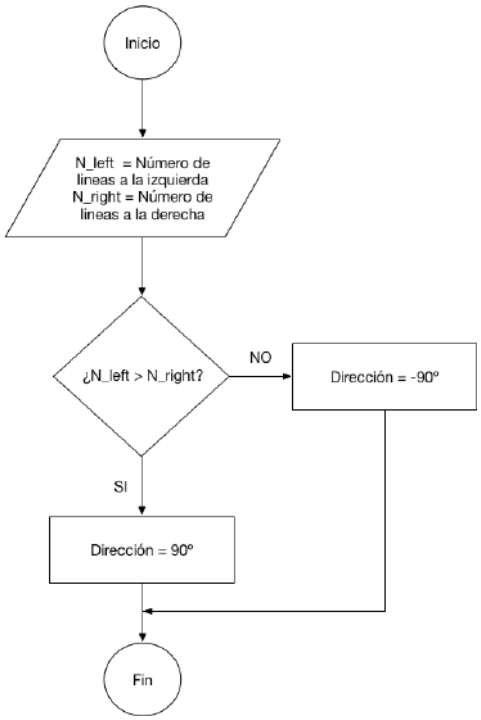


Figura 6.12: Determinación de la orientación del robot en el campo de juego

Preprocesamiento de la información

De acuerdo a la información proporcionada del entorno de desarrollo es necesario realizar una extracción del área de juego. El modelo de color HSV es un modelo motivado por el sistema visual humano. Es una herramienta ideal para desarrollar algoritmos de procesamiento de imágenes basados en descriptores de colores que son naturales e intuitivos de percibir para los humanos. En este caso lo que se desea es la descripción del color verde del campo de juego [Chen et al., 2007] [Sural et al., 2002] [Rafael C. Gonzalez, 2002]. Los resultados de la segmentación por color basado en el modelo de color HSV se muestran en 6.13a. Donde se puede observar la extracción de la zona de interés de manera satisfactoria.

Para la eliminación del ruido y suavizar la imagen se aplica un filtro gaussiano, ademas de la aplicación de los operadores de dilatación y erosión (llamada operación de cierre) se utiliza para realizar la conexión de componentes de forma más sofisticada

Antes de la aplicación de la transformada de Hough es necesario realizar una umbralización y extracción de contornos. Siguiendo el procedimiento desarrollado en la sección 4.2, la umbralizacion se realiza con un valor de umbral fijo de 200 y la extracción de contornos se realiza aplicando el algoritmo de Canny, obteniendo los resultados mostrados en 6.13b y 6.13c.

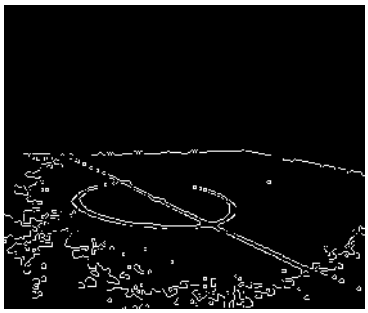
Transformada de Hough y conteo de líneas

Después del preprocesamiento de la información se aplica la transformada de Hough para detectar las líneas presentes en la imagen. En esta aplicación el interés principal es determinar el número de líneas encontradas por Hough, por lo tanto, se almacenan todas las líneas presentes en la escena y mediante las dimensiones del vector de datos obtenido se determinan la cantidad de líneas encontradas a izquierda o derecha del robot según se haya aplicado.

Los resultados del desarrollo de la técnica propuesta se muestran en la Figura 6.13d y 6.13e, para el lado izquierdo y derecho del robot respectivamente cuando el robot se encuentra fuera de la cancha, en este caso en alguna de las tres posiciones en color azul de la parte baja de la cancha (ver Figura fig:Apps301). Se puede observar que con los resultados obtenidos es posible determina el sentido de rotación correcto respecto a la línea central del campo de juego.

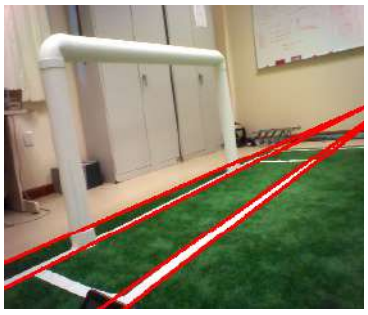
(a) Segmentacíon del color verde

(b) Filtrado y umbralización



(c) Extracción de contornos

(d) Detección derecha de líneas



(e) Detección izquierda de líneas

Figura 6.13: Detección de líneas en el campo de juego con la transformada de Hough

En la Figura 6.9b se observa la posición que cada robot deberá tomar en el campo al inicio del juego o al ser sacado, por lo cual una vez dentro de la cancha cada robot deberá corregir su orientación y distancia respecto al centro del campo, se considera dos casos, el primero cuando es necesario realizar un *kick-off* para iniciar el juego y el segundo la búsqueda del balón en el centro del campo igualmente para iniciar el juego. Por lo tanto, es necesario la identificación de la línea central del campo y referenciarse a ella de acuerdo a una distancia determinada la cual dependerá del caso que se presente.

Analizando el problema plantado y considerando la información con la cual contamos en el entorno, se diseña una serie de etapas para dar solución a nuestro caso de estudio las cales son mostradas en el diagrama de flujo de la Figura 6.14. Donde se muestra que cada una de las etapas ya han sido desarrolladas, por ejemplo, las etapas de preprocesamiento se desarrolló en la sección en la 6.3.1 sin la etapa de conteo de líneas, por otra parte, en la sección 4.2.2 y 6.1 se desarrolló la detección de líneas horizontales y la mejora a la profundidad de campo de búsqueda de líneas horizontales con la cámara inferior del robot.

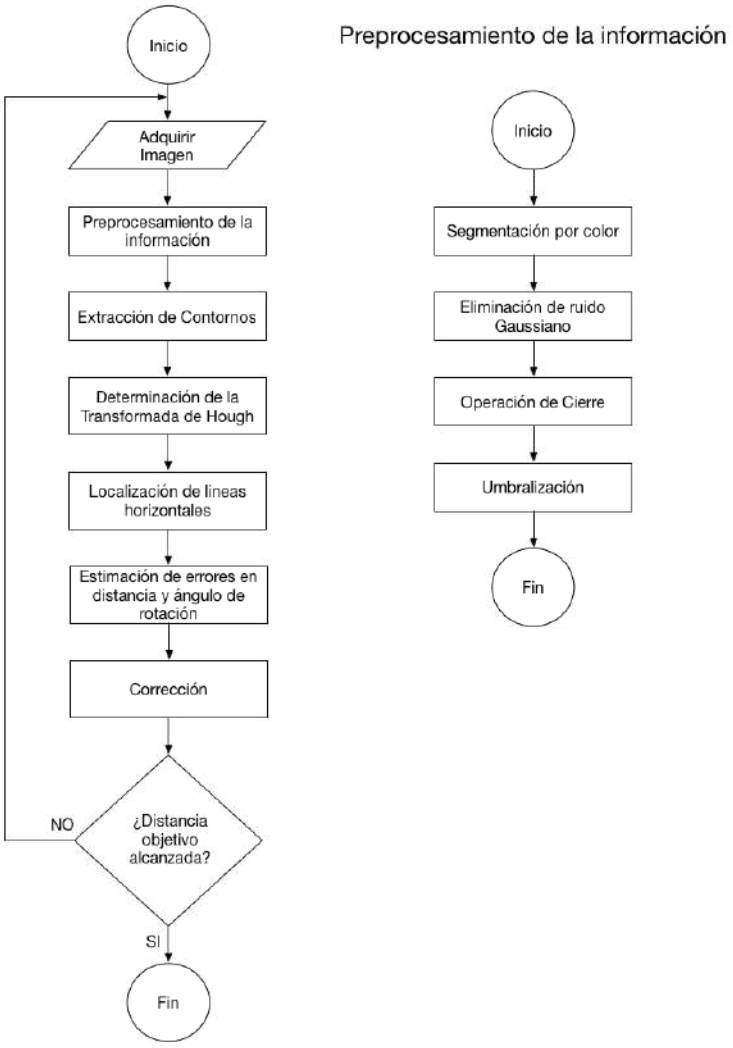


Figura 6.14: Determinación de la línea central del campo de juego

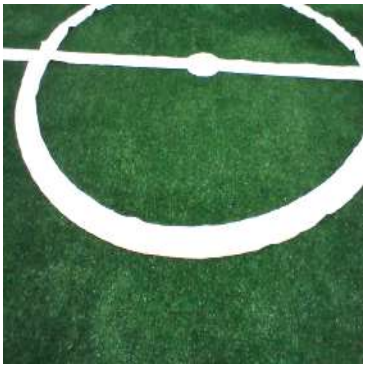
Con el acondicionamiento e integración de cada etapa, los resultados de la aplicación se muestran en la figura 6.15c donde se puede observar la detección correcta de la línea central del campo. Por otra parte, también se observa la detección de líneas ajenas a la de interés. Sin embargo el módulo de búsqueda de líneas reduce el espacio de búsqueda ya que al trabajar conjuntamente con el módulo de corrección distancia y rotación el ángulo de la cámara se modifica de tal manera que la profundidad de campo de la cámara se limita de tal forma que mientras se dirige a la línea central del campo la detección se limita solo a la línea de interés (6.15f). De esta forma es posible ubicar al robot a la distancia deseada respecto a la línea del centro del campo como se muestra en 6.15d.

(a) Posicion inicial del robot

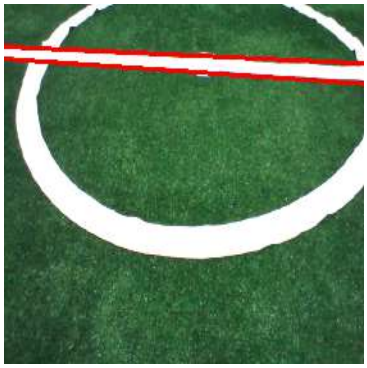
(b) Escena en posicion inicial del robot

(c) Líneas en posicion inicial

(d) Acercamiento del robot



(e) Escena del acercamiento del robot



(f) Líneas en acercamiento

Figura 6.15: Detección de la línea central del campo de juego

Capítulo 7

Conclusiones y Trabajos Futuros

7.1. Conclusiones

La primera técnica implementada está basada en la transformada de Hough, la cual consta de diferentes etapas para la detección correcta de líneas con la intención de ser utilizadas como referencia poder estimar y corregir los errores de orientación del robot dentro del entorno estructurado seleccionado.

Con esta información se ha podido reorientar al robot de forma correcta dentro del entorno estructurado contemplando únicamente la corrección de rotaciones y distancias sobre el eje x. Al corregir la orientación del robot conjuntamente con la corrección de distancias, el robot es capaz de desplazarse en línea recta con un error medio de 1.7 cm en distancias sobre el eje x, 4.4 cm para desplazamientos en el eje y, 3.3^o en las rotaciones, con un tiempo promedio de procesamiento de 6.58 ms. Con los resultados obtenidos se ha realizado navegación en el entorno estructurado, además se demuestra la efectividad de las ecuaciones (4.1) y (4.2) para relacionar los parámetros del algoritmo con las medidas del mundo real y con ello estimar y corregir los errores en distancias y rotaciones del sistema de locomoción del robot.

Aún cuando la resolución del laberinto no es un objetivo en esta investigación, con la técnica desarrollada se ha propuesto un algoritmo simple para la navegación y solucion del mismo. Sin embargo, en principio la corrección realizada es aplicada solamente cuando es posible detectar alguna línea de referencia en la profundidad de campo encontrada (100 cm) para la cámara inferior del robot. Por lo tanto, se desarrolló una aproximación para la estimación de la distancia respecto a una línea de referencia en función de la orientación de la cámara, logrando un tiempo promedio de resolución del laberinto de 15 m lo cual, si bien no es un tiempo óptimo, se ha podido mostrar la navegación del robot en el entorno estructurado con la corrección de los errores del sistema de locomoción con la búsqueda y detección de líneas de referencia.

Siguiendo el algoritmo desarrollado originalmente y mínimas modificaciones en la selección de líneas, se desarrolla e implementa una técnica para la detección del punto de fuga, la cual trabaja como ayuda para la navegación del laberinto ante la ausencia de una línea horizontal de referencia. Sin embargo, una aplicación para esta técnica es en la carrera individual de robots NAO logrando tiempos de recorrido promedios de 60 s.

Con esto se demuestra que los módulos desarrollados pueden ser adaptados para trabajar en diferentes entornos como también ha sido posible la aplicación en la categoría SPL de la competencia internacional RoboCup 2017, aportando una solución sencilla y de rápida implementación para la entrada y orientación al campo de juego de futbol de robots. Donde se ha logrado la orientación y ubicación correcta de los robots utilizando el conteo de líneas del campo de juego como un criterio de decisión para la determinación de la orientación, la búsqueda de líneas para localización de la línea central del campo y utilizarla como referencia para corregir la orientación y posición del robot dentro del campo de juego.

La segunda técnica basada en el registro de imágenes que determina los parámetros necesarios para la puesta en correspondencia de dos imágenes, en esta aplicación es optimizada médiate una estrategia piramidal y el criterio de Woods como medida de similitud, tomando en cuenta solamente la corrección de desplazamientos en el eje x, y y rotaciones. La distorsión de perspectiva y la variación de escala entre imágenes se corrige empíricamente encontrado una matriz de corrección proyectiva para I_R y una más para I_T.

Con los resultados obtenidos por la técnica basada en el registro de imágenes podemos destacar primeramente que el rango de corrección tanto en distancias como en rotaciones en comparación con la técnica basada en la transformada de Hough. Para que el registro de imágenes pueda determinar los errores de ubicación es necesario que las dos imágenes contengan al menos un 30 % de información en común, de lo contrario el sistema no es capaz de estimar los errores de ubicación, tal como se muestra en los experimentos realizados, donde al estar en la posición deseada, debido a no contar con suficiente información en común. Por otra parte, los tiempos de procesamiento son en segundos mientras que la técnica basada en la transformada de Hough realiza la corrección en el orden de milisegundos.

Analizando los resultados obtenidos por el registro de imágenes podemos determinar que la combinación propuesta de búsqueda exhaustiva, con el criterio de Woods como medida de similitud y considerar únicamente desplazamientos y rotaciones para la búsqueda de correspondencia, no es suficiente para la determinación de los parámetros necesarios para la estimación de los errores de ubicación, ya que una ligera variación en escala o la presencia de distorsión de perspectiva entre las imágenes proporciona resultados incorrectos. Por tanto, se debería considerar todos los parámetros de búsqueda y una corrección dinámica de perspectiva que pueda ser adaptable en cualquier orientación de la cámara del robot, ya que una imperfección de la superficie sobre la cual el robot se encuentra, por más imperceptible que pueda ser, al adquirir la información se presenta un cambio en la perspectiva lo cual implica corregir la distorsión con una nueva matriz de perspectiva.

7.2. Trabajos Futuros

Las técnicas empleadas para solucionar el problema planteado, ha resultado útil para determinar los parámetros de reorientación del robot, sin embargo, se requiere de realizar mejoras, tales como, la optimización de los tiempos de procesamiento del software, en especial para el registro de imágenes, ya que el tiempo de operación resulta ser un factor determinante cuando se desea trabajar en aplicaciones con demanda de tiempos de respuesta en el orden de milisegundos o menores, tal es el caso de las competencias tecnológicas por ejemplo la resolución de laberinto y el juego de futbol. Para ellos se propone la migración a código en C++, el cual ofrece tiempos de ejecución y acceso a los recursos del sistema más rápida que Python [Gouy, 2018] lo cual se ha comprobado en las aplicaciones que se han desarrollado las cuales se han codificado en C++.

En cuanto al procesamiento de la información se propone realizar una búsqueda con todos parámetros de transformación, un criterio de similitud más robusto ante los cambios de perspectiva y otras distorsiones, así como también la aplicación de técnicas como la calibración de las cámaras para obtener una mejor exactitud en la relación de los parámetros determinados por software y las medidas del mundo real, con lo cual poder realizar una mejor corrección de los errores del sistema de locomoción.

Bibliografía

[Aggarwal and Karl, 2006] Aggarwal, N. and Karl, W. C. (2006). Line detection in images through regularized Hough transform. *IEEE transactions on image processing*, 15(3):582–591.

[Aldebaran, 2016] Aldebaran, R. (2016). Nao documentation. Online: [http://doc.aldebaran.com/2-1/home_nao.html].

[Anuta, 1970] Anuta, P. E. (1970). Spatial registration of multispectral and multitemporal digital imagery using fast Fourier transform techniques. *IEEE Transactions on Geoscience Electronics*, 8(4):353–368. ISSN: 0018-9413.

[Arancibia, 2013] Arancibia, J. M. Y. (2013). Detección robusta de objetos en robots humanoides. Master's thesis, Universidad De Chile, Facultad De Ciencias Físicas De Matemáticas Departamento De Ingeniería Eléctrica, Santiago De Chile.

[Armendariz., 2003] Armendariz., M. A. M. (2003). *Visión Artificial Estéreo Con Aplicación Al Control De Un Brazo De Robot*. PhD thesis, Instituto Politécnico Nacional, México, D.F.

[Bazeille et al., 2013] Bazeille, S., Barasuol, V., Focchi, M., Havoutis, I., Frigerio, M., Buchli, J., Semini, C., and Caldwell, D. G. (2013). Vision enhanced reactive locomotion control for trotting on rough terrain. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–6. IEEE.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

[Chen et al., 1994] Chen, Q.-s., Defrise, M., and Deconinck, F. (1994). Symmetric phase-only matched filtering of Fourier-mellin transforms for image registration and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12). ISSN: 0162-8828.

[Chen et al., 2007] Chen, W., Shi, Y. Q., and Xuan, G. (2007). Identifying computer graphics using hsv color model and statistical moments of characteristic functions. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1123–1126. IEEE.

[Climent and Mares, 2003] Climent, J. and Mares, P. (2003). Real-time tracking system for assisted surgical operations. *IEEE Latin America Transactions*, 1(1):8–14.

[Cruz Hernández, 2011] Cruz Hernández, E. R. (2011). Desarrollo de un sistema de visión para la localización y navegación de robots humanoides. Tesis para obtener el grado de maestro en ciencias de la ingeniería, Instituto Tecnológico y de Estudios Superiores de Monterrey, Atizapan de Zaragoza, Edo. De México.

[De la Escalera Hueso, 2001] De la Escalera Hueso, A. (2001). *Visión por Computador, Fundamentos y Métodos*. ISBN: 84-205-3098-0. Prentice Hall, Madrid.

[Delfín et al., 2014] Delfín, J., Arechavaleta, G., and Becerra, H. M. (2014). Locomoción humanoide basada en estrategias de control servo visual. *XVI Congreso Latinoamericano de Control Automático*, pages 438–444.

[Díaz et al., 2006] Díaz, C. A., Torres, A., Ramírez, J. I., García, L. F., and Álvarez, N. (2006). Descripción de un sistema para la medición de las presiones plantares por medio del procesamiento de imágenes: Fase i. *Revista EIA*, (6):43–55.

[Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

[Duque and Ospina, 2004] Duque, J. P. U. and Ospina, E. (2004). Implementación de la transformada de Hough para la detección de líneas para un sistema de visión de bajo nivel. *Scientia Et Technica*, 1(24):79–84.

[Ebrahimpour et al., 2012] Ebrahimpour, R., Rasoolinezhad, R., Hajiabolhasani, Z., and Ebrahimi, M. (2012). Vanishing point detection in corridors: using Hough transform and k-means clustering. *IET computer vision*, 6(1):40–51.

[Fernandes and Oliveira, 2008] Fernandes, L. A. and Oliveira, M. M. (2008). Real-time line detection through an improved Hough transform voting scheme. *Pattern Recognition*, 41(1):299–314.

[Fox et al., 2000] Fox, D., Burgard, W., Kruppa, H., and Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous robots*, 8(3):325–344.

[García Capel, 2007] García Capel, L. E. (2007). Aplicación de Medidas Estadísticas de Similitud al Registro de Imagen Multimodo. Proyecto de fin de carrera. ingeniero de telecomunicación, Universidad Politécnica de Cartagena.

[Gifford, 2009] Gifford, C. M. (2009). Low-cost mobile robot localization using only a downward-facing webcam. Technical report, University of Kansas, Lawrence.

[Gouy, 2018] Gouy, I. (2018). The computer language benchmarks game. Online: [http://benchmarksgame.alioth.debian.org/].

[Grupo Mediatec, 2012] Grupo Mediatec, S. d. C. (2012). Nao-h25. Online: [http://www.grupo-mediatec.com/robotica/h25.html].

[Kaebler, 2008] Kaebler, G. . A. (2008). *OpenCV, L. Computer vision with the OpenCV library.* ISBN: 978–0–596–51613–0. O'Reilly, Sebastopol, CA.

[Li et al., 1995] Li, H., Manjunath, B., and Mitra, S. K. (1995). A contour-based approach to multisensor image registration. *IEEE Transaction on Image Processing*, 4(3):320–334. ISSN: 1057-7149.

[Moghadam et al., 2012] Moghadam, P., Starzyk, J. A., and Wijesoma, W. S. (2012). Fast vanishing-point detection in unstructured environments. *IEEE Transactions on Image Processing*, 21(1):425–430.

[Ortiz Zamora, 2002] Ortiz Zamora, F. G. (2002). *Procesamiento morfológico de imágenes en color: aplicación a la reconstrucción geodésica.*

[Pastore et al., 2007] Pastore, J. I., Moler, E., and Ballarin, V. (2007). Multiscale morphological operators and geodesic distance applied to computed axial tomography segmentation. *IEEE Latin America Transactions*, 5(1):28–31.

[Posada et al., 2004] Posada, R., Daul, C., and Miranda, R. (2004). Towards a fractioned treatment in conformational radiotherapy using 3d-multimodal data registration. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 3, pages 1911–1914, Vandoeuvre-lés-Nancy, France. ISSN: 1522-4880.

[Rafael C. Gonzalez, 2002] Rafael C. Gonzalez, R. E. W. (2002). *Digital Image Processing.* ISBN: 0201180758, 9780201180756. Prentice Hall, Michigan,USA.

[Ramírez, 2003] Ramírez, G. (2003). Método de aprendizaje simple para navegación de minirobots móviles rodantes. *Dyna*, 70(138):59–66.

[Rebaza, 2007] Rebaza, J. V. (2007). Detección de bordes mediante el algoritmo de canny. *Escuela Académico Profesional di Informática. Universidad Nacional de Trujillo.*

[Reddy and Chatterji, 1996] Reddy, B. S. and Chatterji, B. (1996). An fft-based technique for translation, rotation, and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(5):1266–1271. ISSN: 1057-7149.

[RoboCup, 2017] RoboCup (2017). The robocup standard platform league. Online: [http://spl.robocup.org].

[Robotrónica, 2013] Robotrónica, A. R. b. (2013). Manejo y programación de nao. Online: [http://aliverobots.com/nao/].

[Rodríguez Santiago, 2013] Rodríguez Santiago, A. L. (2013). Implementación de un sistema de visión mono-cámara basado en el registro de imágenes 2d.

[Rodríguez Santiago et al., 2012] Rodríguez Santiago, A. L., Miranda Luna, R., Arias Aguilar, J. A., and Antonio García, A. (2012). Calibración de una Cámara para un Sistema de Visión Mono-Cámara Económico. *VIII Semana Nacional de Ingeniería Electónica SENIE 12*, pages 348–356. ISBN: 978-607-477-902-8.

[Rother, 2002] Rother, C. (2002). A new approach to vanishing point detection in architectural environments. *Image and Vision Computing*, 20(9):647–655.

[Sarrut, 2000] Sarrut, D. (2000). *Recalage Multimodal et Plate-Forme d' Imagerie Médicale à Accès Distantimagerie Médicale à Accès Distant*. PhD thesis, Université Lumière Lyon 2.

[Se et al., 2002] Se, S., Lowe, D., and Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, 21(8):735–758.

[Sucar, 2012] Sucar, L. E. (2012). Visión computacional. Online: [http://ccc.inaoep.mx/ esucar/Libros/vision-sucar-gomez.pdf].

[Sural et al., 2002] Sural, S., Qian, G., and Pramanik, S. (2002). Segmentation and histogram generation using the hsv color space for image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 2, pages II–II. IEEE.

[Vargas Torres and Castillo Estepa, 2015] Vargas Torres, G. A. and Castillo Estepa, R. A. (2015). Development of an algorithm for humanoid robot detection & command in gathering tasks. *Tecnura*, 19(45):127–139.

[Vázquez Jiménez, 2005] Vázquez Jiménez, I. (2005). Autolocalización de un robot móvil por medio de visión computacional en espacios cerrados. Tesis para obtener el título de ingeniero en computación, Universidad Nacional Autónoma De México, San Juan de Aragón, Estado de México.

[Zarzosa, 2015] Zarzosa, L. C. (2015). Corrección de la odometría visual basada en la detección de cierre de lazo.

Apéndice A

Algoritmos

Algoritmo 1 Obtención de gradiente

Entrada: I Imagen
Entrada: H máscara de convolución, con media cero y desviación estándar σ
Salida: I_m imagen de la magnitud del gradiente
Salida: I_O imagen de la orientación del gradiente

1: Suavizar la imagen I con H mediante un filtro gaussiano y obtener J como imagen de salida.
2: **para** Para cada píxel (I, J) en J, obtener la magnitud y orientación del gradiente basándose en las siguientes expresiones **hacer**
3: El gradiente de una imagen $f(x, y)$ en un punto (x, y) se define como un vector bidimensional dado por la ecuación:
4: $G\left[f(x,y)\right] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$
5: siendo un vector perpendicular al borde, donde el vector G apunta en la dirección de variación máxima de f en el punto (x, y) por unidad de distancia, con la magnitud y dirección dadas por:
6: $|G(i,j)| = \sqrt{G_F^2 + G_C^2} \approx |G_F| + |G_C|$
7: $\phi(i,j) = \tan^{-1}\left(\frac{G_C}{G_F}\right)$
8: Obtener I_m a partir de la magnitud de gradiente y E_o a partir de la orientación, de acuerdo a las expresiones anteriores.
9: **fin para**

Algoritmo 2 Supresión no máxima

Entrada: I_m imagen de la magnitud del gradiente
Entrada: I_O imagen de la orientación del gradiente
Salida: imagen I_n

1: Considerar: cuatro direcciones d_1, d_3, d_3, d_4 identificadas por las direcciones de $0^o, 45^o, 90^o, 135^o$ con respecto al eje horizontal
2: **para** cada píxel (I, J): **hacer**
3: Encontrar la dirección d_k que mejor se aproxima a la dirección $I_o(i, j)$, que viene a ser la perpendicular al borde.
4: **si** $I_m(i, j)$ es más pequeño que al menos uno de sus dos vecinos en la dirección d_k **entonces**
5: Al píxel (i, j) de $I_n(i, j)$ se le asigna el valor $0, I_n(i, j) = 0$ (supresión)
6: **si no**
7: $I_n(i, j) = I_m(i, j)$
8: **fin si**
9: **fin para**
10: **devolver** I_n

Algoritmo 3 Histéresis de umbral a la supresión no máxima

Entrada: I_n imagen obtenida del paso anterior
Entrada: I_O imagen de la orientación del gradiente
Entrada: t_1 umbral
Entrada: t_2 umbral donde $t_1 < t_2$
Salida: I_n imagen con los bordes conectados de contornos

1: **para** todos los puntos de I_n y explorando I_n en orden fijo: **hacer**
2: Localizar el siguiente punto de borde no explorado previamente, $I_n(i, j)$, tal que $I_n(i, j) > t_2$
3: Comenzar a partir de $I_n(i, j)$, seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde, siempre que $I_n > t_1$
4: Marcar todos los puntos explorados y, salvar la lista de todos los puntos en el entorno conectado encontrado.
5: **fin para**
6: **devolver** G formada por el conjunto de bordes conectados de contornos de la imagen, así como la magnitud y orientación, describiendo las propiedades de los puntos de borde

Algoritmo 4 Proceso de votación de la transformada de Hough.

Entrada: I Imagen Binaria
Entrada: δ Paso de desrastización para el espacio de parámetros
1: $Votos \leftarrow 0$ Inicialización de la matriz de votación
2: **para** Cada píxel de $I(x, y)$ **hacer**
3: **para** $0^o < \theta < 180^o$, usando un paso discreto de δ **hacer**
4: $\rho \leftarrow x \cos\theta + y \sin\theta$
5: $Votos(\rho, \theta) \leftarrow Votos(\rho, \theta) + 1$
6: **fin para**
7: **fin para**

Apéndice B

Código para correción con la transformada de Hough.

```
import cv2
import numpy as np

def HoughLineDetector(Srcimage):
    img_gray = cv2.cvtColor(Srcimage, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(img_gray, 50, 100, apertureSize=3)
    lines = cv2.HoughLines(edges, 1, np.pi / 180, 150)
    return lines
    pass

def LineClassifier(lines, angle_min, angle_max):
    Hlines = []
    for rho, theta in lines[0]:
        if np.pi / 180 * angle_min < theta < np.pi / 180 * angle_max:
            Hlines.append((rho, theta))
            pass
        pass
    return Hlines
    pass

def HoughLinesCorrection(img):
    lines = HoughLineDetector(img)
    linesH = LineClassifier(lines, 80, 100)
    # Si hay lineas horizontales, es posible la correccion
    if len(linesH)!=0:
        rho = linesH[0][0]
        theta = linesH[0][1]
        # Error de rotacion para alineacion con la referencia
        errorRotacion = 1.5441 * theta - 139.4118
        # Determinacion de la distancia del robot respecto a la referencia
        errorDistancia = ((0.0003 * (rho * rho)) - (0.3251 * rho) + 100.0675)
        pass
    # No hay lineas horizaontales, No es posible corregir
    else:
        errorRotacion = -1
```

```
36            errorDistancia = -1
37            pass
38        return errorRotacion, errorDistancia
39        pass
```

Apéndice C

Código para correción con el Registro de Imágenes.

```
import cv2
import numpy as np

def setAffineAndRotationMatrix(img, Dx, Dy, Theta):
    rows, cols = img.shape
    M = cv2.getRotationMatrix2D((cols / 2, rows / 2), Theta, 1)
    dst = cv2.warpAffine(img, M, (cols, rows))
    M = np.float32([[1, 0, Dx], [0, 1, Dy]])
    dst = cv2.warpAffine(dst, M, (cols, rows))
    return dst
    pass

def getPyramidOfImges(img, levels):
    ImgCopy = img.copy()
    ImgPy = [ImgCopy]
    for i in xrange(levels):
        ImgCopy = cv2.pyrDown(ImgCopy)
        ImgPy.append(ImgCopy)
        pass
    return ImgPy
    pass

def getImgeFromPyramid(PyramidImage, level):
    return cv2.pyrUp(PyramidImage[level])
    pass

def woods(im1, im2):
    [sy, sx] = np.shape(im1)
    H_Conjunto = cv2.calcHist([im1, im2], [0, 1], None, [256, 256], [0, 256, 0,
    256])
    H_Conjunto = H_Conjunto[1:, 1:]
    pij = np.divide(H_Conjunto, np.sum(H_Conjunto))
    pj = np.nan_to_num(np.sum(pij, axis=1).reshape(255, 1))
    i = np.arange(0, 255).reshape(1, 255)
    SUMipij = np.dot(pij, i.transpose())
```

```
35      mI_J = np.zeros((255, 1))
36      np.seterr(divide='ignore', invalid='ignore')
37      mI_J = np.divide(SUMipij, pj).transpose()
38      i_mI_J = np.zeros((255, 255))
39      for j in xrange(255):
40          i_mI_J[j, :] = i - mI_J[0, j]
41          pass
42      i_mI_J2 = i_mI_J * i_mI_J
43      i_mI_J2pij = i_mI_J2 * pij
44      SUMi_mI_J2pij = np.sum(i_mI_J2pij, axis=1).reshape(255, 1)
45      Varianza = np.divide(SUMi_mI_J2pij, pj)
46      np.seterr(divide='ignore', invalid='ignore')
47      Woods = np.nan_to_num(np.divide(np.sqrt(Varianza), mI_J.transpose()))
48      return np.sum(Woods * pj, axis=0)
49      pass
50
51  def getPerpectiveCorrection(img, a):
52      inv_a = np.linalg.inv(a)
53      sy, sx = img.shape
54      im2 = np.zeros((sy, sx))
55      xc = (sx / 2) + 0.5
56      yc = (sy / 2) + 0.5
57
58      for j in range(sy):
59          for i in range(sx):
60              x = i - xc
61              y = yc - j
62
63              xyz = np.array([[x], [y], [1]])
64              uvw = np.dot(inv_a, xyz)
65              xi = np.int(np.divide(uvw[0], uvw[2])[0])
66              yi = np.int(np.divide(uvw[1], uvw[2])[0])
67
68              xo = np.int(xi + xc)
69              yo = np.int(yc - yi)
70
71              xa = np.floor(xo)
72              xb = np.ceil(xo)
73              ya = np.floor(yo)
74              yb = np.ceil(yo)
75
76              Dxa = xo - xa
77              Dxb = xb - xo
78              Dya = yo - ya
79              Dyb = yb - yo
80
81              A11 = Dxa * Dya
82              A12 = Dxb * Dya
83              A21 = Dxa * Dyb
84              A22 = Dxb * Dyb
85
86              if (xo >= 0) & (xo <= sx) & (yo >= 0) & (yo <= sy):
87                  if (Dxa == 0) & (Dya == 0):
```

```
88                      im2[j, i] = img[yo - 1, xo - 1]
89                      pass
90                  elif (Dxa == 0) & (Dya != 0):
91                      im2[j, i] = (img[ya - 1, xo - 1] * Dyb) + (img[yb - 1, xo -
      1] * Dya)
92                      pass
93                  elif (Dxa != 0) & (Dya == 0):
94                      im2[j, i] = (img[yo - 1, xa - 1] * Dyb) + (img[yo - 1, xb -
      1] * Dya)
95                      pass
96                  else:
97                      im2[j, i] = (img[ya - 1, xa - 1] * A22) + (img[ya - 1, xb -
      1] * A21) + (
98                          img[yb - 1, xa - 1] * A12) + (img[yb - 1, xb - 1] * A11
     )
99                      pass
100                 pass
101             pass
102         pass
103     return im2
104     pass
105
106 def RecalageCorrectionFunction(ImgR, ImgT):
107     th = 0
108     tx = 0
109     ty = 0
110     tic()
111     # Convert images to grayscale
112     ImgRGray = cv2.cvtColor(ImgR, cv2.COLOR_BGR2GRAY)
113     ImgTGray = cv2.cvtColor(ImgT, cv2.COLOR_BGR2GRAY)
114     # Images preprocessing
115     # Perspective correction
116     H2R = np.array(([1.4545, 0.0116, 0],
117                     [0, 2.0625, 0],
118                     [0.00006, -0.00363, 1.0000]))
119     H2T = np.array(([1.0679, -0.0122, 0],
120                     [0.0122, 1.1699, 0],
121                     [0.00002, -0.00165, 1.0000]))
122     ImgRGray = np.uint8(getPerpectiveCorrection(ImgRGray, H2R))
123     ImgTGray = np.uint8(getPerpectiveCorrection(ImgTGray, H2T))
124     # Top Hat transformation
125     kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (6, 6))
126     ImgRPyramidInLevel = cv2.morphologyEx(ImgRPyramidInLevel, cv2.MORPH_TOPHAT,
      kernel)
127     ImgTPyramidInLevel = cv2.morphologyEx(ImgTPyramidInLevel, cv2.MORPH_TOPHAT,
      kernel)
128     ImgRPyramidInLevel = cv2.GaussianBlur(ImgRPyramidInLevel, (3, 3), 0)
129     # Gaussian filter
130     ImgTPyramidInLevel = cv2.GaussianBlur(ImgTPyramidInLevel, (3, 3), 0)
131     # Get images pyramid
132     levels = 4  # levels in pyramid
133     n_Iteration = 0  # Number of Iteration
134     # Create pyramid
```

```
135     ImgRPyramid = getPyramidOfImges(ImgRGray, levels)
136     ImgTPyramid = getPyramidOfImges(ImgTGray, levels)
137     # Start parameters
138     theta_Start = 0
139     theta_End = 0
140     theta_Step = 1
141     ty_Start = 0
142     ty_End = 0
143     ty_Step = 1
144     tx_Start = 0
145     tx_End = 0
146     tx_Step = 1
147     print "Start..."
148     while levels > 1:
149         levels -= 1
150         # Get Image of the last one level in pyramid
151         ImgRPyramidInLevel = getImgeFromPyramid(ImgRPyramid, levels)
152         ImgTPyramidInLevel = getImgeFromPyramid(ImgTPyramid, levels)
153         # Size of Image in this level
154         rows, cols = ImgRPyramidInLevel.shape
155         # Set Initial Parameters
156         if n_Iteration == 0:
157             theta_Start = -5
158             theta_End = 5
159             theta_Step = 1
160             ty_Start = -rows / 4
161             ty_End = rows / 4
162             ty_Step = 1
163             tx_Start = -cols / 4
164             tx_End = cols / 4
165             tx_Step = 1
166             pass
167         Dx = np.linspace(tx_Start, tx_End, ((tx_End - tx_Start) / tx_Step) + 1)
168         Dy = np.linspace(ty_Start, ty_End, ((ty_End - ty_Start) / ty_Step) + 1)
169         Th = np.linspace(theta_Start, theta_End, ((theta_End - theta_Start) /
    theta_Step) + 1)
170         W = np.zeros((len(Dx) + 1, len(Dy) + 1, len(Th) + 1))
171         # Exhaustive Search
172         for k in xrange(len(Th)):
173             for i in xrange(len(Dx)):
174                 for j in xrange(len(Dy)):
175                     A = setAffineAndRotationMatrix(ImgTPyramidInLevel, Dx[i],
    Dy[j], Th[k])
176                     W[i, j, k] = woods(ImgRPyramidInLevel, A)
177                     print Dx[i], Dy[j], Th[k], W[i, j, k]
178                     pass
179                 pass
180             pass
181         W = W[:-1, :-1, :-1]
182         [i, j, k] = np.where(W == np.min(W))
183         i = i[0]
184         j = j[0]
185         k = k[0]
```

```
186             th = Th[k]
187             tx = Dx[i]
188             ty = Dy[j]
189             theta_Start = th if th == 0 else 2 * th - 1
190             theta_End = 2 * th + 1
191             ty_Start = ty if ty == 0 else 2 * ty - 1
192             ty_End = 2 * ty + 1
193             tx_Start = tx if tx == 0 else 2 * tx - 1
194             tx_End = 2 * tx + 1
195             n_Iteration += 1
196             pass
197         Dx_ = ((tx * 108) / 70) / 10 if tx != 0 else 0
198         Dy_ = ((ty * 108) / 70) / 10 if ty != 0 else 0
199         Th_ = th
200         return Dx_, Dy_, Th_
201         pass
```

Apéndice D

Código en C++ para las aplicaciones.

```
1  // Opencv includes.
2  #include <opencv2/opencv.hpp>
3  #include <opencv2/core/core.hpp>
4  #include <opencv2/highgui/highgui.hpp>
5  #include <opencv2/imgproc/imgproc.hpp>
6  #include <qi/os.hpp>
7  // C/C++ includes
8  #include <iostream>
9  #include <string>
10 #include <stdio.h>
11
12 /*=====Variables Globales=====*/
13 cv::vector<cv::Vec2f> lines;
14 std::vector<cv::Vec2f> Hlines;
15 std::vector<cv::Vec2f> VPlines_left;
16 std::vector<cv::Vec2f> VPlines_right;
17 std::vector<cv::Vec2f> VP;
18
19 // Detecta las lineas en la imagen
20 void HoughLinesDetector(cv::Mat Srcimage, double threshold_value, double
       max_BINARY_value, double threshold_type,
21   double lowThreshold, double max_lowThreshold, double kernel_size,
22   double rhoResolution, double thetaResolution, double thresholdDetect){
23   cv::Mat grayImg, threshImg, canny_img;
24   cv::cvtColor(Srcimage, grayImg, CV_BGR2GRAY);
25   threshold(grayImg, threshImg, threshold_value, max_BINARY_value,
       threshold_type);
26   cv::Canny( threshImg, canny_img, lowThreshold, max_lowThreshold, kernel_size)
       ;
27   cv::HoughLines( canny_img, lines, rhoResolution, thetaResolution,
       thresholdDetect, 0, 0);
28 }
29
30 // Clasifica Lineas Horizontales y Verticales para punto de fuga
31 void LineClassifier(double Anglemin2Hlines,double AngleMax2Hlines,
32   double Anglemin2Vlines,double AngleMax2Vlines){
33   Hlines.clear(), VPlines_left.clear(), VPlines_right.clear();
```

```
34    for( size_t i = 0; i < lines.size(); i++ ){
35      float rho = lines[i][0], theta = lines[i][1];
36      if (theta>CV_PI/180*Anglemin2Hlines && theta<CV_PI/180*AngleMax2Hlines){
37        Hlines.push_back(cv::Vec2f(lines[i][0], lines[i][1]));
38      }
39      else if ((theta>CV_PI/180* Anglemin2Vlines && theta<CV_PI/180*
      AngleMax2Vlines) ||
40        (theta>CV_PI/180* - AngleMax2Vlines && theta<CV_PI/180* - Anglemin2Vlines
      )){
41        if (theta>CV_PI/180* Anglemin2Vlines && theta<CV_PI/180* AngleMax2Vlines)
      {
42          VPlines_left.push_back(cv::Vec2f(lines[i][0], lines[i][1]));
43          m1 += (-cos(theta)/sin(theta)), b1 += ((rho)/sin(theta));
44        }
45        if (theta>CV_PI/180* - AngleMax2Vlines && theta<CV_PI/180* -
      Anglemin2Vlines){
46          VPlines_right.push_back(cv::Vec2f(lines[i][0], lines[i][1]));
47          m2 += (-cos(theta)/sin(theta)), b2 += ((rho)/sin(theta));
48        }
49      }
50    }
51  }
52
53  // Determina las coordenadas del punto de fuga
54  void Vanishing_Point(double Anglemin2VPlines_left, double AngleMax2VPlines_left,
55              double Anglemin2VPlines_right, double AngleMax2VPlines_right ){
56    float m1, b1, m2, b2, VPX, VPY;
57    VP.clear();
58    for( size_t i = 0; i < lines.size(); i++ ){
59      float rho = lines[i][0], theta = lines[i][1];
60      if ((theta>CV_PI/180*Anglemin2VPlines_left && theta<CV_PI/180*
      AngleMax2VPlines_left) ||
61          (theta>CV_PI/180*Anglemin2VPlines_right && theta<CV_PI/180*
      AngleMax2VPlines_right)){
62        if (theta>CV_PI/180*Anglemin2VPlines_left && theta<CV_PI/180*
      AngleMax2VPlines_left){
63            VPlines_left.push_back(cv::Vec2f(lines[i][0], lines[i][1]));
64            m1 += (-cos(theta)/sin(theta)), b1 += ((rho)/sin(theta));
65          }
66        if (theta>CV_PI/180*Anglemin2VPlines_right && theta<CV_PI/180*
      AngleMax2VPlines_right){
67            VPlines_right.push_back(cv::Vec2f(lines[i][0], lines[i][1]));
68            m2 += (-cos(theta)/sin(theta)), b2 += ((rho)/sin(theta));
69          }
70      }
71    }
72    m1 /= VPlines_left.size(), b1 /= VPlines_left.size();
73    m2 /= VPlines_right.size(), b2 /= VPlines_right.size();
74    VPX = (b2-b1)/(m1-m2);
75    VPY = ((m1 * VPX) + b1);
76    VP.push_back(cv::Vec2f(VPX,VPY));
77  }
78
```

```
79 int houghLinesCounter(cv::Mat Srcimage){
80   cv::Mat hsvImg, threshImg, canny_img;
81   int lowH = 40, lowS = 70, lowV = 70;  // Color Verde Low
82   int highH = 80, highS = 200, highV = 200;  //Color Verde High
83   // Transformacion de RGB al espacio de color HSV
84   cv::cvtColor(Srcimage, hsvImg, CV_BGR2HSV);
85   // Extraccion del color verde
86   cv::inRange(hsvImg, cv::Scalar(lowH, lowS, lowV), cv::Scalar(highH, highS,
      highV), threshImg);
87   // Filtrado
88   cv::GaussianBlur(threshImg, threshImg, cv::Size(3, 3), 0);       //Blur Effect
89   // Clossing
90   cv::dilate(threshImg, threshImg, 0);             // Dilate Filter Effect
91   cv::erode(threshImg, threshImg, 0);
92   // Separacion de objetos en la imagen
93   threshold(threshImg, threshImg, 200, 255,0 );
94   // Extraccion de Contornos
95   cv::Canny( threshImg, canny_img, 200, 100,3);
96   // Aplicacion de la transformada de Hough
97   cv::HoughLines( canny_img, lines, 1, CV_PI/180, 100 );
98   // Numero de lineas encontradas
99   return lines.size();
100 }
```