UNIVERSIDAD DEL PAPALOAPAN

CAMPUS LOMA BONITA

INGENIERIA EN MECATRÓNICA

Reconocimiento de Patrones con OpenCV para la Asignación de Tareas en un Robot Móvil

Tesis para obtener el título de

INGENIERO EN MECATRÓNICA

Presenta:

JASMANIK TORRES LÓPEZ

Director de tesis:
M.C. Rafael Fernando González Zárate

LOMA BONITA, OAX

Universidad del Papaloapan

ÁREA:	**VICE RECTORIA ACADEMICA**
OFICIO NÚMERO:	**UNPA/VRA/038/2024**
ASUNTO:	**AUTORIZACION DE IMPRESIÓN DE TESIS**

Loma Bonita, Oax., a 08 de febrero de 2024.

C. JASMANIK TORRES LÓPEZ
Presente.

En base al artículo 120 del reglamento de alumnos, por medio del presente se aprueba la impresión de la tesis titulada **"Reconocimiento de Patrones con OpenCV para la Asignación de Tareas en un Robot Móvil"** así como la programación del examen profesional bajo la dirección del M.C. Rafael Fernando González Zarate.

Sin más por el momento le envío un cordial saludo.

Atentamente
terrauberrima, mensaperta
Bou lo tama, chijíjú

VICE-RECTORIA ACADEMICA

M.C HÉCTOR LÓPEZ ARJONA
VICE-RECTOR ACADÉMICA.

C.c.p. M.E. Yesenia Barrientos Arenal.- Jefa de servicios escolares.-UNPA
M.C Luis Alberto Hernández Zuccolotto.- Jefe de la Carrera de Ing. En Mecatrónica.
M.C Rafael Fernando González Zarate.- Director de Tesis.
Archivo.

UNIVERSIDAD DEL PAPALOAPAN

Campus Loma Bonita

Oficio: IMEC-24-005
Loma Bonita, Oaxaca, a 6 de febrero de 2024

M.E. Yesenia Barrientos Arenal
Jefa del Departamento de Servicios Escolares
PRESENTE

Mediante la presente, le informo que la jefatura de carrera a mi cargo, con visto bueno de la Vice-Rectoría Académica, ha designado a los siguientes profesores como sinodales para examen profesional del **C. Jasmanik Torres López**, quien defenderá su trabajo de tesis titulado ***"Reconocimiento de Patrones con OpenCV para la Asignación de Tareas en un Robot Móvil"***, para obtener el título de Licenciado en Ingeniería en Mecatrónica.

Titulares:

Presidente: **M.C. Luis Alberto Hernández Zuccolotto**
Secretario: **Dr. Hiram Netzahualcóyotl García Lozano**
Vocal: **M.C. Rafael Fernando González Zarate**

Suplentes:

Dr. Jesús Santiaguillo Salinas
M.C. Edmundo Mendieta Fernández

Atentamente
tierra uberrima, mens aperta
Bau lu-tama, chi, ji, ju

M.C. Luis Alberto Hernández Zuccolotto
Jefe de Carrera de Ingeniería en Mecatrónica

UNPA OAXACA VICE-RECTORIA ACADEMICA

M.C. Héctor López Arjona
Vice-Rector Académico

c.c.p. M.C. Héctor López Arjona. Vice-Rector académico. Para su conocimiento
c.c.p. Archivo

Agradecimientos

A mi amada madre, quien siempre me apoya y confía en mí. Gracias por todos los sacrificios que hiciste para que terminara mi ingeniería, por ser mi inspiración, por siempre creer en mí y por ser el faro que me guio en mis peores momentos. Gracias por todo lo que has hecho por mí.

A mi abuelita, mi segunda madre, quiero expresar cuánto significas para mí y agradecerte por todo lo que has hecho a lo largo de los años. Desde que era pequeño, has sido una presencia constante en mi vida, brindándome amor incondicional, sabiduría y apoyo inquebrantable.

A mi abuelito, quien me cuida desde el cielo, te extraño profundamente, pero sé que tu espíritu vive en cada rayo de sol, en cada brisa suave y en cada estrella que brilla en el cielo nocturno.

A mis hermanos por estar siempre a mi lado y compartir conmigo cada paso de este camino, los aprecio más de lo que las palabras pueden expresar.

A mi increíble novia, por su comprensión, paciencia y aliento durante esta etapa de mi vida. Tu presencia ha hecho que cada logro sea aún más significativo. Gracias por ser mi compañera en este viaje.

A toda mi familia, agradezco la bondad y la generosidad que muestran día a día, así como el amor incondicional que nos une como familia. Los momentos que hemos compartido, ya sean grandes o pequeños, están llenos de recuerdos que atesoraré para siempre.

Al profe Rafa y Dr. Hiram, quienes nunca dudaron en brindarme su tiempo, paciencia y conocimiento, lo cual agradezco sinceramente.

A todos mis profesores, quienes a pesar de saber que no soy el mejor estudiante siempre me brindaron el apoyo y la dedicación para guiarme y enseñarme.

A mis leales amigos, por ser mi red de apoyo, por compartir risas en los momentos de alegría y por brindar me consuelo en los tiempos difíciles. Su amistad me recordó la importancia de disfrutar cada paso del camino, me siento afortunado de conocerlos.

A la Universidad del Papaloapan por brindarme la oportunidad de formarme académica y personalmente.

Resumen

En esta tesis, se presenta la implementación de un **sistema robótico de visión**. Este sistema se diseñó con la finalidad de reconocer patrones geométricos que controlen los movimientos de un **robot móvil**. Se compone de un algoritmo de reconocimiento de patrones, un algoritmo de comunicación y un algoritmo de control en lazo abierto.

La primera parte del **algoritmo de reconocimiento de patrones** consiste en la verificación del número de contornos presentes en la imagen recibida, si el número es mayor a uno sigue buscado hasta detectar un solo contorno, lo que implica que la imagen contiene una sola figura. Posteriormente se detecta el número de esquinas, lo que facilita la clasificación de las figuras geométricas. Este algoritmo se programó haciendo uso de algoritmos para el procesamiento de imágenes y visión por computadora que ofrece la **librería OpenCV**. El algoritmo entrega un número real, el cual está relacionado directamente con la figura geométrica recibida en la imagen. Este número es transmitido entre el sistema embebido utilizado para el reconocimiento de patrones y el sistema que utiliza el robot móvil. El algoritmo de comunicación se ha diseñado para transmitir tal información. El robot se controla por medio del algoritmo de control en lazo abierto desplazando al robot en función de la figura detectada.

El algoritmo de reconocimiento de patrones y algoritmo de comunicación se implementaron en el sistema embebido Raspberry PI. El algoritmo de control en lazo abierto se implementó en una placa Arduino.

Se evaluó la respuesta del algoritmo de reconocimiento de patrones ante variaciones de intensidad de luz, en el laboratorio de control de la Universidad del Papaloapan, campus Loma Bonita. El algoritmo demostró ser eficaz ante pequeñas variaciones de luz en entornos controlados.

Palabras Clave: Sistema robótico de visión, Robot móvil, algoritmo de reconocimiento de patrones, librería OpenCV.

Abstract

In this thesis, the implementation of a **robotic vision system** is presented. This system is designed to recognize geometric patterns that control the movements of a **mobile robot**. The system consists of a pattern recognition algorithm, a communication algorithm, and an open-loop control algorithm.

The first part of the **pattern recognition algorithm** involves checking the number of contours present in the received image. If the number is greater than one, it continues searching until detecting a single contour, indicating that the image contains a single figure. Subsequently, the number of corners is detected, facilitating the classification of geometric shapes. This algorithm was programmed using image processing and computer vision algorithms provided by the **OpenCV library**. The algorithm outputs a real number directly related to the geometric figure detected in the image. This number is transmitted between the embedded system used for pattern recognition and the system controlling the mobile robot. The communication algorithm is designed to transmit this information. The robot is controlled by the open-loop control algorithm, moving the robot based on the detected figure.

The pattern recognition and communication algorithms were implemented on the Raspberry Pi embedded system. The open-loop control algorithm was implemented on an Arduino board.

The response of the pattern recognition algorithm was evaluated in the control laboratory of the University of the Papaloapan, Loma Bonita campus, under variations in light intensity. The algorithm proved to be effective in handling small variations in light in controlled environments.

Keywords: Robotic vision system, Mobile robot,* Rattern recognition algorithm*, OpenCV library.

Revision by: C.Cheryl Lynn Gad

Jefa del Centro de Idiomas, Campus Loma Bonita

Date: 22/02/2024

ÍNDICE

Índice de imágenes

Capítulo 1 Introducción

La robótica es la ciencia encargada del desarrollo de mecanismos capaces de realizar tareas de manera autónoma, comúnmente conocidos como robots. En el año 1920 aparece el término *Robot*, en la obra titulada *Rossum´s Universal Robots* (Čapek, 1920), empleado para definir a los androides de la novela. A partir de ese momento, diferentes autores comenzaron a adoptar el término para definir así a cualquier mecanismo autónomo.

Actualmente, se han desarrollado una gran diversidad de robots, enfocando la mayor parte del desarrollo en la fabricación de robots manipuladores (brazos robóticos). Los cuales son ampliamente usados en aplicaciones industriales (manejo de material, soldadura, inspección, entre otras) aumentando la eficiencia y producción de las fábricas, sin embargo, presentan la desventaja de ser fijos, por este motivo en la década de los sesenta se comenzó a impulsar el término *robot móvil*.

Un robot móvil es una máquina capaz de desplazarse por el espacio, con el fin de realizar una tarea de forma autónoma. Se dividen en tres tipos: aéreos, acuáticos y terrestres. Y representan un gran campo de investigación y desarrollo. Por esta razón, el presente trabajo se centra en los robots móviles terrestres, específicamente en los robots móviles a ruedas (RMR).

La eficiencia de un robot móvil para desplazarse por el espacio radica principalmente en dos aspectos; su sistema de locomoción y el medio por el cual se desplaza. Por lo tanto, es importante conocer las características que ofrecen los diferentes tipos de locomoción que pueden implementarse en los robots móviles terrestres, así como, las diferentes configuraciones cinemáticas que un RMR puede presentar. Por esta razón, el capítulo 1 está destinado a conocer dichos aspectos de los robots móviles terrestres.

Los RMR son eficientes desplazándose en terrenos controlados, sin embargo, en la práctica los terrenos tienden a ser hostiles, en ese sentido, dotarlos de un sistema de visión puede ayudarlos realizar múltiples acciones, tales como: identificar objetos, identificar terrenos irregulares por los que no sean capaces de desplazarse (entre otras acciones) y de esta forma tomar decisiones automáticamente.

La visión es la capacidad que tienen los organismos de convertir la luz, que es reflejada por los objetos, en impulsos nerviosos. Posteriormente estos impulsos son procesados y analizados por el cerebro con el fin de obtener una percepción del mundo exterior. La visión es uno de los sentidos que posee el cuerpo humano, por lo que el hombre ha tratado de imitar el proceso de visión durante más de sesenta años. Cualquier persona que esté interesada en participar en el desarrollo de la visión artificial debe aprender a interpretar la visión no solo como un sentido, sino como un proceso. Por lo tanto, el capítulo 2 está destinado a explicar el proceso de la visión humana, así como la relación que existe entre estos procesos y los procesos que realizan los sistemas de visión artificial.

El capítulo 3 está enfocado en OpenCV, una librería de software de visión por computadora y aprendizaje automático de código abierto, la cual cuenta con más de 2500 algoritmos optimizados, clásicos y de última generación. Actualmente es una de las herramientas más potentes que se tiene para el desarrollo de la visión por computadora. Este capítulo aborda la historia de la visión por computadora y el origen de OpenCV, así como, algunas de las diferentes funciones que nos ofrece la librería y que fueron útiles en el desarrollo del algoritmo para el reconocimiento de patrones.

En el capítulo 4 se presenta la descripción del hardware empleado en la implementación del algoritmo de reconocimiento de patrones. Se presenta el hardware y software que permite la interacción entre el algoritmo de reconocimiento de patrones y el robot móvil, así como los problemas y resultados obtenidos

Por último, el capítulo 5 está destinado a las conclusiones y la explicación de trabajos futuros basados en los resultados obtenidos en este trabajo.

Capítulo 2 : Robótica móvil

2.1 Introducción a la robótica móvil

La robótica móvil, un campo interdisciplinario que requiere conocimientos de diversas áreas de estudio (matemáticas, física, control, mecánica, electrónica y computación), tiene como objetivo principal, dotar a los robots con la capacidad de desplazarse eficientemente a través del espacio.

Se requiere un sistema de locomoción para permitir que un robot se desplace. Dependiendo de la aplicación y el entorno, el sistema de locomoción puede incluir ruedas, patas, orugas, alas o propulsores, entre otros. En la actualidad se han desarrollado diversos sistemas de locomoción, cada uno diseñado para operar en entornos y tareas específicas. Este capítulo, presenta una breve descripción de los sistemas de locomoción más empleados en los robots móviles terrestres.

Por otra parte, el estudio y la comprensión del movimiento, es una parte esencial en el diseño y control de los robots móviles. La forma en que los robots se desplazan esta intrínsecamente relacionada con la cinemática de los sistemas de locomoción. Por esta razón, en este capítulo se presenta una breve descripción de las configuraciones comúnmente empleadas en el análisis, diseño e implementación de RMR's, enfocándose principalmente en la cinemática del robot móvil con configuración diferencial.

2.2 Sistemas de locomoción

Los vehículos terrestres no tripulados (UGV, Unmanned Ground Vehicles) son ampliamente usados en aplicaciones espaciales, militares, educativas, agrícolas, industriales, entre otras. Las superficies en las que se desplazan suelen presentar características diferentes, por lo cual, es importante seleccionar el sistema de locomoción adecuado. A continuación, se presentan los tres sistemas de locomoción más empleados en los robots móviles terrestres: locomoción por patas, locomoción por orugas y locomoción a ruedas (Silva Ortigoza, García Sánchez, Barrientos Sotelo, & A. Molina, 2007).

2.2.1 Locomoción a patas

Los robots con **locomoción a patas** son capaces de moverse a través de superficies irregulares, sin embargo, su principal desventaja se encuentra en mantener la estabilidad, maniobrabilidad y controlabilidad del robot. Requieren un mínimo de dos grados de libertad para mover una pata. Se han logrado mejores resultados, en robots de dos patas, agregando un grado de libertad en la articulación al tobillo, permitiéndoles un contacto con el suelo más consistente al activar la postura de la planta del pie. Aunque agregar grados de libertad a la pierna aumenta la maniobrabilidad del robot, esto puede tener ciertas desventajas debido a que se requieren agregar actuadores adicionales. Los actuadores adicionales aumentan la masa, requiriendo mayor potencia-energía requerida, además de aumentar la dificultad del control. Los robots con locomoción a patas se clasifican, según el número de patas, en: bípedos, cuadrúpedos, hexápodos, y más (Siegwart & Nourba, 2004).

Los seres humanos son el ejemplo más claro de organismos **bípedos**, por lo que se han desarrollado robots con un mecanismo de locomoción que imita sus movimientos. Tal es el caso del robot ASIMO, mostrado en la Figura 2.1, un robot bípedo creado por Honda, que actualmente es capaz de caminar, correr, saltar en una o dos piernas de manera continua, además de mantener el equilibrio en superficies irregulares (Honda, 2023).

Figura 2.1 Robot ASIMO

Los **robots cuadrúpedos** son diseñados de manera que imiten la forma de moverse de los animales con cuatro patas. En la Figura 2.2 se presenta a AIBO, un perro robot desarrollado por Sony, capaz de correr e interactuar con su usuario de una manera muy similar a la de un perro (Sony, 2023).

Figura 2.2 Robot AIBO

Los insectos poseen un sistema de locomoción compuesto por seis patas, que les permite desplazarse incluso boca abajo. En este contexto, son las criaturas locomotoras más exitosas de la tierra. Por esta razón, se han desarrollado robots con las características locomotoras de hormigas, arañas, y otros insectos. Un ejemplo es el **robot hexápodo** mostrado en la Figura 2.3. El diseño de este robot imita la estructura de un escorpión, fue desarrollado en el Laboratorio de ciencia de datos de la Universidad de Gante, en Bélgica. El escorpión robótico (Figura 2.3) fue diseñado para ser lo más reproducible, modular y adaptable posible.

Figura 2.3 Escorpión robótico

2.2.2 Sistema de locomoción por orugas

Una alternativa al sistema de locomoción a patas, para terrenos poco firmes, son las orugas. Este sistema de locomoción hace uso de pistas de deslizamiento, lo que implica una mayor área de contacto con la superficie, obteniendo movilidad superior a la que ofrecen las patas. A demás de tener un diseño mecánico menos complejo, presentan gran versatilidad para operar en terrenos con poca dureza, como pueden ser superficies con arena o nieve, presentando gran capacidad de carga. En desplazamientos rectos tienen buen contacto con el suelo, por otra parte, en los giros cerrados el suelo tiende a fracturarse debido a la fricción y a las altas vibraciones (González, Rodríguez, & Guzman, 2015).

Figura 2.4 Robot PackBot 510

En la Figura 2.4 se observa a PackBot 510, un robot móvil terrestre no tripulado (UGV). Este robot participa en misiones militares de reconocimiento y de vigilancia, inspección y desactivación de bombas y explosivos. Un sistema de oruga doble le permite avanzar sobre cualquier terreno, incluso subir escaleras.

2.2.3 Sistema de locomoción a ruedas

El sistema de locomoción a ruedas es el más empleado en los robots móviles terrestres, debido a presentan un buen desempeño para desplazarse sobre superficies duras y lisas. Este sistema consta de ruedas que giran alrededor de ejes fijos o articulados, proporcionando movilidad y dirección al robot. Actualmente son empleados en diversas aplicaciones, tales como: transporte de mercancías, exploración espacial, conducción autónoma y otras aplicaciones. Los aspectos clave a considerar en los sistemas de locomoción a ruedas son el tipo de ruedas y la configuración cinemática

La diversidad de ruedas disponibles ofrece a cada robot la posibilidad de adoptar un método de desplazamiento de acuerdo con sus necesidades particulares. Las ruedas se dividen principalmente en dos categorías: rueda estándar y rueda sueca. La **rueda estándar**, Figura 2.5, se puede encontrar en diversos transportes, tales como: automóviles, bicicletas, entre otros. Estas ruedas solo se desplazan en dirección del ángulo de giro. A diferencia de una rueda estándar, una **rueda sueca** (también llamada omnidireccional) se desplaza en otra dirección diferente al ángulo de giro, esto se debe a la rotación pasiva de los rodillos ubicados a lo largo de su periferia. Los rodillos se pueden ubicar a 90 grados con respecto al eje de giro, como se muestra en la Figura 2.6, o a 45 grados, como se muestra en la Figura 2.7 (F. Muir & P. Neuman, 1986).

Figura 2.5 Rueda estándar

Figura 2.6 Rueda omnidireccional a 45°

Figura 2.7 Rueda omnidireccional a 90°

2.3 Cinemática del robot móvil a ruedas

El tipo de rueda, que se use en el diseño de un robot móvil, está relacionado con la configuración cinemática del robot. A continuación, se presentan tres de las configuraciones cinemáticas más empleadas en los robots móviles a ruedas: Akerman, Omnidireccional y Diferencial (Ollero Baturone, 2005).

La configuración **Ackerman** está diseñada para vehículos que necesitan seguir trayectorias curvas de manera suave y precisa. Hace uso de cuatro ruedas estándar, dos delanteras y dos traseras. Sus ruedas delanteras, giran a diferentes ángulos para seguir una trayectoria curva y están orientadas hacia el centro del círculo de giro. Esta configuración es empleada en los automóviles. Ejemplo de ello es el Autominy, un vehículo a escala desarrollado en la Freie Universität Berlin, mostrado en la Figura 2.8.

Figura 2.8 Autominy, configuración akerman

Los robots con configuración **omnidireccional** están diseñados para permitir movimientos en cualquier dirección sin cambiar la orientación del robot. Utiliza ruedas suecas que permiten el desplazamiento lateral sin rotar el robot y un alto grado de maniobrabilidad en espacios reducidos.

El robot omnidireccional de tres ruedas, Figura 2.9, utiliza ruedas suecas a 90°. Las tres ruedas son fijas y se colocan a la misma distancia del centro del robot con un ángulo de separación de 120° entre cada rueda, los rodillos de las ruedas le otorgan mayor maniobrabilidad.

Figura 2.9 Robot Omnidireccional con tres ruedas

El robot omnidireccional de cuatro ruedas, mostrado en la Figura 2.10, utiliza ruedas suecas a 45°. Una peculiaridad de esta configuración es que tiene dos ruedas con los rodillos orientados a 45° (ruedas derechas) y dos ruedas con los rodillos orientados a 135° (ruedas izquierdas). Se utilizan dos ruedas delanteras, una izquierda y una derecha, colocadas paralelamente. Las ruedas traseras deben colocarse de manera inversa a las delanteras para otorgarle mayor maniobrabilidad al robot.

Figura 2.10 Robot Omnidireccional Uranus (Carnegie Mellon University)

La configuración **diferencial** está diseñada para simplificar la mecánica y el control. Se compone de dos ruedas estándar motrices, colocadas paralelamente sobre el mismo eje, una a cada lado del robot y una rueda libre que otorga soporte, como se muestra en la Figura 2.11. La forma en la que se mueven las ruedas determina la forma de moverse del robot. En este caso, si las ruedas del robot se mueven en la misma dirección y a la misma velocidad, el robot avanza en línea recta, si las ruedas giran en la misma dirección, pero con diferentes velocidades, el robot sigue una trayectoria curva y si las ruedas giran en dirección opuesta, a la misma velocidad, el robot rota alrededor de su propio eje. Esta configuración se utiliza en robots móviles simples y robustos, en aplicaciones como limpieza autónoma o educación.

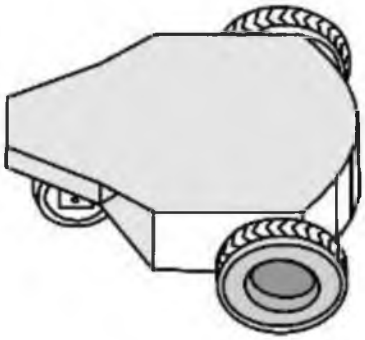

Figura 2.11 Robot diferencial

Cada configuración tiene sus propias ventajas y desventajas, y la elección depende de los requisitos específicos de la aplicación. Por ejemplo, la configuración Ackerman es ideal para vehículos que necesitan seguir trayectorias curvas con precisión, mientras que la configuración omnidireccional es más adecuada para aplicaciones que requieren movimientos multidireccionales rápidos y precisos. La configuración diferencial tiene una configuración cinemática, con menor nivel de complejidad y presenta desempeños aceptables en superficies lisas y duras.

Es posible observar este comportamiento a través de un análisis numérico, haciendo uso del modelo cinemático. Es decir, ecuaciones que describen el movimiento en función de sus parámetros geométricos y de control. La cinemática se divide en dos: cinemática directa y cinemática inversa.

La **cinemática directa** se encarga de determinar la posición y orientación del robot en el espacio en función de las velocidades de las ruedas. Para un robot móvil a ruedas, se utilizan las ecuaciones de cinemática directa para calcular cómo se mueve el robot en función de las velocidades lineales y angulares de sus ruedas. Este modelo varía según el tipo de ruedas del robot, por ejemplo, los modelos diferenciales o modelos omnidireccionales (Siciliano, Sciavicco, Villani, & Oriolo, 2009). La **cinemática inversa** se ocupa de determinar las velocidades de las ruedas necesarias para lograr una posición y orientación deseadas del robot. En otras palabras, dado un punto en el espacio al que se quiere llegar, la cinemática inversa calcula cómo deben girar las ruedas para llegar a ese punto.

Ambos modelos son importantes para el diseño, control y planificación de movimiento en robots móviles, mejorando la capacidad de navegación y ejecución de tareas. A continuación, se presenta el modelo matemático del robot móvil con configuración diferencial, para ello se hace uso de la Figura 2.12.

El modelo describe el desplazamiento y orientación de un marco de referencia móvil (XR, YR), que se encuentra sobre el robot, con respecto a un marco de referencia inercial fijo (X, Y).

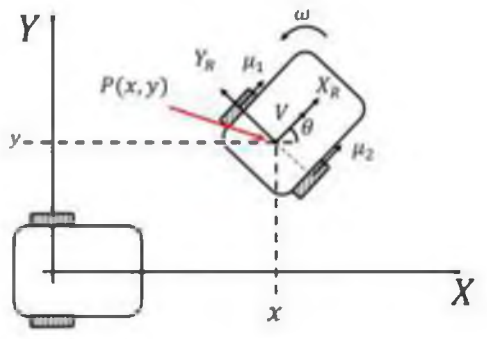

Figura 2.12 Esquema cinemático de configuración diferencial

El punto de control, denotado como $P(x, y)$, se encuentra ubicado en el origen del marco de referencia móvil (X_R, Y_R) y coincide con el punto medio del eje de las ruedas. El ángulo de orientación, representado por θ, se define como la diferencia angular entre el eje x del marco de referencia móvil (X_R, Y_R) y el eje x del marco de referencia fijo (X, Y). La velocidad angular se denota por ω y V representa la velocidad longitudinal del robot. Las velocidades longitudinales de las ruedas se representan por (μ_1, μ_2).

El modelo cinemático del robot móvil diferencial está dado por la ecuación (1).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sec(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (1)$$

Las velocidades angulares de las ruedas están dadas por la ecuación (2).

$$\begin{bmatrix} \omega_d \\ \omega_i \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & \frac{d}{2} \\ 1 & -\frac{d}{2} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (2)$$

Donde, el radio de las ruedas se denota como r y d representa la distancia entre el centro del eje de las ruedas y las ruedas.

2.3 Conclusiones

El segundo capítulo proporciona una visión general de la robótica móvil, enfocándose en los RMR. Se presentan los sistemas de locomoción y configuraciones cinemáticas empleadas en los RMR. Adicionalmente se presentan tres configuraciones: Ackerman, Omnidireccional y Diferencial.

Se introduce el modelo cinemático para el robot móvil con configuración diferencial y las ecuaciones cinemáticas directas, que nos permite observar cómo las velocidades de las ruedas afectan el movimiento y la posición del robot.

En resumen, el capítulo ofrece una base para comprender los principios de la robótica móvil, desde la selección del sistema de locomoción hasta la implementación de configuraciones cinemáticas específicas. Este conocimiento es importante para el diseño y la implementación de robots móviles en diversas aplicaciones.

Capítulo 3 Visión

"La visión es el resultado de un intrincado proceso neurológico que transforma los patrones de luz en una representación coherente y significativa del mundo que nos rodea" - Desconocido

La visión es el sentido más utilizado para obtener información del mundo exterior. A través de la visión se pueden identificar propiedades cualitativas y cuantitativas del entorno, tales como: color, forma, tamaño, distancia, posición y movimiento. La percepción de estas características es importante para la comunicación y toma de decisiones.

En el ámbito de la robótica móvil, extraer características del entorno es de gran importancia para lograr una navegación satisfactoria, debido a que, el principal objetivo es permitir a los robots desplazarse sobre entornos con características físicas y geométricas no conocidas. Este objetivo se logra a través del funcionamiento conjunto del Sistema de Control, Sistema de Locomoción y el Sistema de Adquisición de Datos (Sensores Internos y Externos).

Los robots utilizan sensores exteroceptivos para extraer características del entorno. Estos sensores tienen la capacidad de detectar y, en algunos casos, procesar información relacionada con el ambiente, permitiendo que los robots perciban estímulos externos, tales como: sonido, luz, temperatura, presión y otros factores ambientales. Algunos ejemplos comunes de sensores exteroceptivos incluyen sensores de fuerza, sensores de rango y sensores de visión. (Siciliano, Sciavicco, Villani, & Oriolo, 2009)

El uso de sensores de rango, tales como: sensores ultrasónicos e infrarrojos, proporciona a los robots la capacidad realizar tareas, tales como; evitar obstáculos, construir mapas del entorno, entre otras. Por otra parte, los sensores de visión, parte indispensable de los sistemas de visión, proporcionan información geométrica y cualitativa del entorno que les permite medir distancias, planificar trayectorias, reconocer objetos, entre otras tareas.

La robótica móvil ha encontrado en los sistemas de visión una herramienta de gran importancia. En las próximas páginas se explora el complejo sentido de la visión y se revisan hechos históricos que han contribuido al desarrollo de los modelos actuales del proceso de visión. A medida que se amplía la comprensión de este sentido fundamental, se aborda de qué manera el proceso de visión humano puede ser aplicado para desarrollar sistemas de visión artificial.

3.1 Introducción a los modelos de visión

A lo largo de la historia el sentido de la visión a fascinado a filósofos, científicos e investigadores, se ha convertido en el epicentro de un debate fundamental entre dos enfoques contrastantes: el modelo activo y el modelo pasivo. Estos enfoques representan diferentes maneras de abordar como percibimos el mundo y de entender las complejidades que se presentan al analizar el proceso visión.

En el libro titulado ***Percepción visual*** se presentan el modelo activo y pasivo de visión. En el modelo activo de la visión, se plantea que el ojo emite partículas de luz que viajan por el espacio hasta tocar un objeto, el contacto entre estas partículas y los objetos producen la sensación de la visión. Por otra parte, en el modelo pasivo de la visión se propone la idea de que son los objetos quienes envían imágenes de sí mismos hacia el espacio que los rodea. Estas imágenes, denominadas *eidola,* se introducen en el ojo humano después de viajar por el espacio, produciendo así la visión. (Alberich, Gómez Fontanills, & Ferre Franquesa, 2013)

Transcurrieron muchos años para resolver el debate de si son los ojos los que emiten partículas de luz o si son las partículas las que se introducen al ojo. En la obra *Vision How It Works and What Can Go Wrong*, John Dowling y Joseph Dowling hacen referencia al experimento realizado por el científico árabe, Ibn Al Hatham. En este experimento se expusieron los ojos de animales al sol y se descubrió que después de un periodo de tiempo, la retina resulta dañada. A partir de esto, razonó que la luz del sol que ingresa al ojo causa el daño. (Dowling & Dowling, 2016)

El historiador Beaumont Newhall (en su obra ***La Historia de la Fotografía***) hace referencia al libro ***Magiae Naturalis*** (publicado en el año 1553 por Giovanni Battista Della Porta) en el cual se presenta por primera vez una descripción documentada de la cámara oscura (el primer sistema de visión artificial). El proceso de la cámara oscura, consiste en permitir la entrada de luz en una habitación cerrada, a través de un pequeño orificio ubicado en una de las paredes. Como resultado, se proyecta una imagen invertida del mundo exterior en la pared opuesta de la habitación. (Newhall, 2002)

La creación de la cámara oscura representó un hecho fundamental en la comprensión de los principios básicos del proceso de visión. Años después de su creación, entre 1571 y 1630, se postuló que las imágenes se forman en la retina y que es el cerebro el encargado procesarlas. Esto sentó las bases para considerar al cerebro como elemento principal del proceso.

3.2 Percepción de la de visión

Según el libro **Sensation and Perception** la percepción es una secuencia de procesos que trabajan juntos para determinar nuestra experiencia y reacción a los estímulos del entorno. Se divide en cuatro categorías: estímulo, electricidad, experiencia-acción, y conocimiento. El estímulo se refiere a todo lo que hay en nuestro entorno y estimula nuestros receptores. Por su parte, la electricidad trata con las señales que son transducidas y transmitidas al cerebro. En la experiencia y acción se hace referencia al objetivo; percibir, reconocer y reaccionar ante los estímulos. El conocimiento se refiere a toda la información previa que se tiene del estímulo.

En la actualidad, el estudio de los modelos, del proceso de percepción, contemplan al menos dos enfoques: el enfoque fisiológico y el enfoque psicofísico. (Goldstein, 2010)

3.2.1 Enfoque fisiológico

El Enfoque Fisiológico es un área de investigación interdisciplinaria que combina conceptos de anatomía, neurología, y de otros campos de estudio. Busca comprender la relación entre los estímulos y los procesos fisiológicos. El proceso fisiológico de la visión comienza en el momento en que la luz se introduce en el ojo, un órgano constituido por el **globo ocular** y el **nervio óptico**. **El globo ocular** es una estructura de forma esférica de unos 25 milímetros de diámetro y consta de tres capas: externa, media e interna.

La capa externa está constituida por la **esclerótica** y la **córnea**. La esclerótica es una cubierta fibrosa protectora que constituye el área blanca del ojo, en su parte central se hace trasparente y forma la córnea. La córnea es una lente que permite el paso de luz a la parte interna del globo ocular, su función es proteger.

La capa media está constituida por la **coroides**, el **cuerpo ciliar**, el **iris**, el **cristalino** y el **humor vitreo**. La coroides es una membrana vascularizada de coloración obscura cuya función es evitar que la luz se refleje. El cuerpo ciliar está formado por humor acuoso y por un músculo. El iris es una membrana ubicada por detrás de la córnea y posee una abertura central llamada pupila, cuya función es regular la cantidad de luz que entra en el globo ocular, la pupila se contrae cuando hay mucha luz y se dilata cuando hay poca luz en el ambiente. El cristalino es una lente capaz de cambiar su tamaño y curvatura, en función de la distancia a la que se encuentran los objetos con el objetivo de enfocar la luz en la retina, gracias al musculo ciliar. El humor vitreo es un gel incoloro e inodoro que llena el espació entre el cristalino y la retina.

En la capa interna, las señales eléctricas se propagan por la **retina**, un tejido altamente organizado y funcional que desencadena el proceso de **detectar** y **transducir** la luz en señales eléctricas. Estas señales se transmiten al cerebro a través del **nervio óptico,** para su procesamiento y percepción visual. La retina es una capa delgada y compleja, está formada por tejido conectivo y 10 capas de células, las cuales se dividen en tres tipos: pigmentadas, neuronas y células de sostén. Las neuronas a destacar son las células fotorreceptoras (sensibles a la luz) las cuales se dividen en dos tipos: conos y bastones. Los bastones se activan a bajas frecuencias proporcionando la visión en blanco y negro y la detección de movimiento y los conos que se activan en situaciones de mucha luminosidad proporcionando la visión a color.

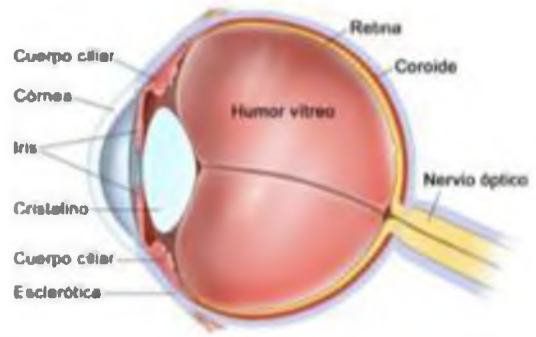

Figura 3.1 Estructura del ojo

El nervio óptico es una extensión del sistema nervioso que se origina en la retina del globo ocular, esencial para la transmisión de la información visual desde el ojo hasta el cerebro. Está compuesto por axones, prolongaciones de células nerviosas conocidas como células ganglionares, que transmiten las señales eléctricas generadas en la retina hacia el cerebro. El nervio óptico transmite la información visual al área del **cerebro** llamada **corteza visual o córtex visual.**

La información visual llega finalmente a la **corteza visual**, una región del **cerebro** dedicada a procesar y comprender lo que vemos. Aquí, las señales visuales se integran con otros datos sensoriales y cognitivos para crear una imagen coherente y significativa de nuestro entorno.

3.2.2 Enfoque psicofísico

El Enfoque Psicofísico se centra en la relación que existe entre las propiedades físicas de los estímulos y las respuestas perceptivas. A continuación, se presenta una definición, así como algunos ejemplos tomados de experiencias, de posibles respuestas perceptivas que pueden ocurrir.

Describir: Indicar las características de un estímulo. "La figura es de color rojo"

Reconocer: Colocar el estímulo en la categoría especifica. "Esa Figura es un triángulo"

Detectar: Tomar conciencia de un aspecto apenas detectable de un estímulo. "En esa imagen, están casi todas las figuras geométricas, falta el círculo.

Percibir la magnitud: Ser consciente del tamaño o la intensidad de un estímulo. "El rectángulo es el doble de largo que cuadrado"

Búsqueda: Buscar un estímulo específico entre una serie de otros estímulos. "Estoy buscando la elipse"

A través de experimentos y métodos, los psicofísicos exploran cómo nuestras respuestas perceptivas se relacionan con las propiedades de la luz (intensidad, longitud de onda, otras).

Un concepto fundamental en el enfoque psicofísico es el **umbral** perceptual, que representa el punto en el que podemos percibir un cambio mínimo en un estímulo. Por ejemplo, el umbral de luminosidad se refiere al nivel más bajo de luz que podemos detectar.

El enfoque psicofísico también examina cómo percibimos el contraste entre objetos (la relación entre diferentes longitudes de onda de luz) y cómo nuestras percepciones varían en diferentes condiciones de iluminación. A través de experimentos cuidadosamente diseñados, los psicofísicos pueden elaborar funciones de respuesta perceptual y curvas de sensibilidad, que revelan cómo nuestras percepciones se ajustan a los cambios en los estímulos visuales.

3.2.3 Procesos del sentido de la visión

Basándose en lo explicado anteriormente, se identifican las etapas principales del proceso del sentido de la visión, las cuales se mencionan a continuación:

Detección: En esta etapa, nuestros sentidos detectan la información visual disponible en el entorno. El sistema visual recoge la luz reflejada por los objetos y la enfoca en la retina.

Transducción: Los fotorreceptores en la retina convierten la luz en señales eléctricas.

Transmisión de señales: El nervio óptico es el encargado de la transmisión de señales. Las señales son conducidas desde la retina hasta el cerebro.

Procesamiento: Las señales visuales llegan a la corteza visual, una región del cerebro donde se realiza el procesamiento visual primario. Aquí, las señales se organizan en formas, colores y movimientos, formando la base de nuestra percepción visual consciente.

Fijación visual: Durante esta fase, nuestros ojos se dirigen a una región específica del campo visual para obtener una vista más detallada.

Extracción de características: En esta etapa, el cerebro establece límites o umbrales para la identificación de características más complejas, como colores, texturas y formas. Se crean mapas de activación neuronal que representan estas características.

Reconocimiento: En esta fase, el cerebro interpreta la información visual para reconocer objetos familiares y asignarles significado. La experiencia pasada y el conocimiento influyen en este proceso.

Percepción de profundidad de movimiento: El cerebro utiliza señales visuales para estimar la profundidad y la distancia de los objetos, así como para percibir el movimiento en el entorno.

Precepción constante: Esta etapa implica la capacidad del cerebro para percibir los objetos con consistencia a pesar de las variaciones en la iluminación, el ángulo de visión y otros factores.

Precepción consiente: Finalmente, todas las etapas anteriores conducen a una experiencia visual consciente, donde la información visual se integra en una representación coherente y comprensible de nuestro entorno.

3.3 Sistemas de visión artificial

Se ha visto que el sistema de visión humano está compuesto por un conjunto de elementos que permiten adquirir, procesar y analizar información del entorno para tener una percepción precisa del mundo exterior. Análogamente al sistema de visión humano es posible diseñar sistemas de visión artificial. A continuación, se presentan las etapas del proceso de visión, previamente expuestas, desde la perspectiva de la visión artificial.

3.3.1 Detección y transducción

El encardo de detectar la luz en el sistema de visión humano es el globo ocular, el dispositivo que cumple esta función en los sistemas de visión artificial es la cámara digital. La cámara digital es una cámara fotográfica que detecta, transduce y procesa la luz para posteriormente almacenarla en un arreglo de matrices numéricas.

La Figura 3.2 muestra una imagen sencilla para explicar el proceso de detección y transducción o adquisición de una imagen.

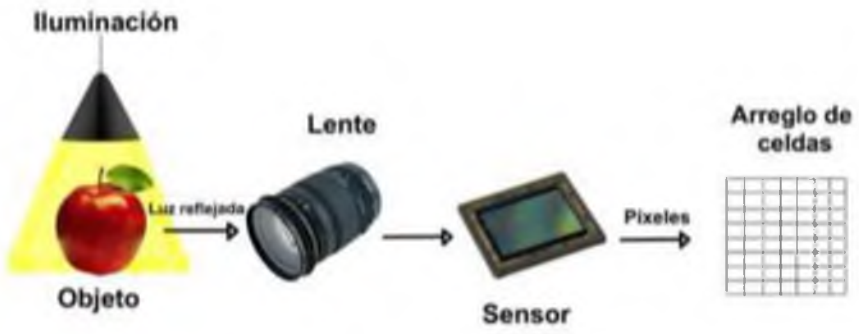

Figura 3.2 Proceso de detección y transducción

Como se aprecia en la ilustración, la luz reflejada por un objeto se introduce a través de la lente de la cámara y se proyecta sobre un conjunto de sensores, ordenados en un arreglo matricial. Los sensores responden a la intensidad lumínica y generan una señal de voltaje. La magnitud del voltaje producida, se almacena en una matriz. A cada valor individual de la matriz se le conoce como píxel. Cada píxel registra la intensidad de luz en su ubicación específica, y esta información se utiliza para componer la imagen completa. La señal de voltaje generada por cada píxel es de naturaleza analógica. Para poder procesar y almacenar esta información se utiliza un convertidor analógico digital (ADC) que convierte las señales analógicas en valores digitales. En los sistemas más simples se emplea una representación de 8 bits, lo que permite definir 256 niveles de intensidad en cada píxel. Es importante señalar que tanto las cámaras como el ojo humano capturan exclusivamente la luz, y la percepción del color es el resultado de procesos adicionales, como la interpretación de señales por los conos, en el caso del ojo. Por otra parte, los sistemas de visión artificial que requieren la captura de imágenes en color utilizan filtros. El modelo RBG es comúnmente el más empleado, y suelen integrarse en las cámaras digitales.

3.3.2 Transmisión de señales

Los datos capturados se transmiten desde la cámara al procesador a través de conexiones como cables USB, HDMI o conexiones inalámbricas, dependiendo del sistema y la aplicación. Análogamente a cómo las señales nerviosas viajan desde el ojo al cerebro, la transmisión de datos permite que la información visual se mueva del sensor de la cámara hacia el procesador.

3.3.3 Procesamiento

En el procesador, la imagen digital es procesada utilizando algoritmos de programación, con el objetivo de mejorarla o modificarla, y obtener una imagen digital con diferentes características. Algunos de los algoritmos comunes que se pueden implementar se describen a continuación.

La **segmentación de imágenes** busca separar un objeto u objetos de interés presentes en una imagen del resto de la imagen. A continuación, se muestra un ejemplo; en la Figura 3.3 se observan dos flores, una de color amarillo y otra de color rojo, ambas con tallos verdes. De fondo, el cielo azul. Se busca separar las flores de color amarillo del resto de la imagen, por lo cual se utiliza un método de segmentación. El resultado Figura 3.4 es una imagen binaria, la parte de color blanco corresponde a la flor de color amarillo.

Figura 3.3 Imagen formato RGB

Figura 3.4 imagen binaria

Una de las herramientas más importantes, en el proceso de segmentación, es el **umbral**. En este método, a partir de una imagen digital, como ejemplo una imagen en escala de grises con 256 tonos de gris, se busca obtener una imagen binaria. Si se propone un umbral con valor de 100, los pixeles con tonos de gris mayores a 100 forman parte del objeto de interés. Por lo tanto, los números mayores a 100 pasaran a tomar un valor de 255 en la escala de gris y los menores un valor de 0, como resultado se obtendrá una imagen binaria.

La **detección de bordes** es otro de los métodos ocupados en el procesamiento de imágenes, el cual consiste en evaluar un pixel con un pixel vecino, si se detecta un cambio brusco en los valores de escala de gris, el pixel se considera un borde y toma el valor más alto en la escala de grises, de lo contrario su valor es 0. De esta forma cuando se detecta un borde, el pixel es iluminado de color blanco y el resto de color negro, obteniendo como resultado una imagen binaria.

En ocasiones se utilizan algoritmos para mejorar la calidad visual de la imagen capturada. Pueden incluir técnicas de ajuste de brillo y contraste, cambios de espacio de color, corrección de color, reducción de ruido, recorte de imágenes, entre otros.

3.3.4 Fijación visual

En el contexto de visión artificial, el término fijación visual se utiliza para describir cómo un algoritmo puede enfocarse en áreas específicas de una imagen para realizar tareas específicas de procesamiento o análisis. En este caso, la fijación visual no implica un movimiento real de la cámara, sino más bien una representación de cómo el algoritmo dirige su atención a ciertas partes de la información visual.

3.3.5 Extracción de características

La extracción de características es un componente fundamental en la visión artificial y se refiere al proceso de identificar y capturar información relevante de los datos visuales para facilitar tareas de análisis y toma de decisiones. Las características extraídas pueden ser patrones, texturas, colores u otras propiedades que ayudan a describir los objetos y elementos presentes en una imagen.

3.3.6 Reconocimiento

El reconocimiento en visión se refiere al proceso de identificar y clasificar objetos, patrones o elementos presentes en imágenes o videos utilizando algoritmos y técnicas computacionales. El objetivo es que una máquina o computadora sea capaz de reconocer visualmente diferentes estímulos.

3.3.7 Precepción constante

En el campo de la visión artificial, lograr la percepción visual constante es un desafío importante y se aborda mediante el uso de algoritmos y técnicas que pueden reconocer y clasificar objetos en diversas condiciones. Esto implica la capacitación de modelos para ser robustos ante cambios en la apariencia de los objetos.

3.3.8 Percepción consiente

Las máquinas y los algoritmos no tienen conciencia ni percepción consciente en el mismo sentido que los seres humanos. Los algoritmos de visión artificial procesan datos de manera computacional y no experimentan una experiencia subjetiva.

3.4 Conclusión

A lo largo de este capítulo, se ha realizado un recorrido histórico sobre los hechos que contribuyeron significativamente a la conceptualización de los procesos y etapas que realiza en el sentido de la visión. Estos hechos históricos, abarcan desde los primeros modelos, realizados por los antiguos filósofos griegos, hasta las teorías fundamentales de la percepción visual de la época moderna.

Además, se ha llevado a cabo una exploración a grandes rasgos de los procesos y etapas que conforman el sistema de visión humano. Este recorrido abarca desde la detección inicial de la luz por parte del ojo hasta la culminación en la percepción consciente. Se han desglosado las distintas etapas que componen este proceso, que incluyen la detección, transducción, transmisión de señales, procesamiento, fijación visual, extracción de características, reconocimiento, percepción de profundidad de movimiento, percepción constante y percepción consciente.

El propósito fundamental ha sido establecer una base de conocimiento sobre el proceso de visión humano. No obstante, esta base no se limita solo a la comprensión de cómo percibimos el mundo a través de nuestros ojos, sino que se extiende hacia la aplicación de estos principios en el campo de la visión artificial. La visión artificial busca imitar los procesos de la visión humana en sistemas tecnológicos, y este capítulo establece el fundamento para comprender cómo estos procesos se traducen y adaptan en la creación y funcionamiento de sistemas de visión artificial.

Capítulo 4 Visión por computadora con OpenCV

"La visión por computadora es el arte de dotar a las máquinas con el don de la percepción, abriendo una ventana hacia un mundo digital lleno de significado visual" - Gary Bradski

4.1 Introducción a la visión por computadora

La visión por computadora es un campo de estudio interdisciplinario que busca proporcionar a las máquinas, en particular a los robots, la capacidad de interpretar y comprender su entorno a partir de imágenes digitales. En las últimas décadas, los avances en diversas tecnologías han permitido la aplicación de la visión por computadora en múltiples campos de estudio y aplicaciones prácticas, tales como: vehículos autónomos, clasificación de objetos, detección y seguimiento de personas, inspección de calidad en la industria, cirugía asistida por robots, entre otras.

Los **vehículos autónomos** utilizan tecnologías de visión por computadora para extraer información visual del entorno y así navegar de forma segura en entornos variables. La **clasificación de objetos** hace uso de las herramientas que proporciona la visión por computadora para clasificar objetos a partir de una imagen o video. La **detección y seguimiento de personas** se utiliza en sistemas de vigilancia, y en la industria del entretenimiento para el seguimiento de movimientos. En el campo de **inspección de calidad en la industria**, la visión por computadora se utiliza para inspeccionar productos en busca de defectos en líneas de producción, mejorando la calidad y reduciendo el desperdicio. En el campo de la **cirugía asistida por robots**, la medicina, robótica y visión por computadora se combinan para permitir cirugías más precisas y menos invasivas.

Estas tecnologías continúan evolucionando y encontrando nuevas aplicaciones en diferentes campos, lo que promete un futuro emocionante en términos de innovación y automatización. Esta evolución ha sido posible gracias al esfuerzo de estudiantes, investigadores, científicos y desarrolladores de todo el mundo que han diseñado métodos, técnicas, estrategias y herramientas que permiten a las máquinas percibir su entorno visual a través de una computadora.

Una poderosa herramienta en este campo es OpenCV (Open Source Computer Vision Library). Una biblioteca de software de visión artificial y aprendizaje automático de código abierto. Ofrece más de 2500 algoritmos optimizados. Algoritmos que se pueden utilizar para detectar y reconocer caras, identificar objetos estáticos o en movimiento, extraer modelos 3D de objetos, encontrar imágenes similares de una base de datos de imágenes, reconocer paisajes y otras acciones. La capacidad de esta biblioteca para optimizar el procesamiento de imágenes ha contribuido significativamente al avance de la visión por computadora y a su aplicación en numerosos campos, tales como; automatización industrial, sistemas de seguridad y visión artificial en robótica, entre otros campos (OpenCV, 2022).

La visión por computadora ha evolucionado notablemente, desde sus primeras investigaciones hasta la creación de OpenCV. Este capítulo ofrece una revisión de la evolución histórica de la visión por computadora, desde de las primeras investigaciones que le permitieron consolidarse como un campo de estudio, hasta los orígenes de OpenCV. Además, se describen algunos de los algoritmos que OpenCV ofrece a los desarrolladores, abarcando desde técnicas básicas de procesamiento de imágenes, como el cambio de espacios de color y la segmentación, hasta algoritmos avanzados que permiten la extracción de características, entre otras funcionalidades.

4.2 Evolución de la visión por computadora

En 1959 David Hubel y Torsten Wiesel publicaron su trabajo titulado: ***Campos receptivos de neuronas individuales en la corteza estriada del gato***. En ese trabajo se explora cómo la forma, tamaño, movimiento y dirección de estímulos lumínicos influyen en la formación de campos receptivos en la corteza visual del cerebro de los gatos. Se explican múltiples experimentos, en los cuales se colocaron electrodos en el área primaria de la corteza visual del cerebro de gatos anestesiados y se logró observar la respuesta neuronal ante diversos estímulos de luz. Los hallazgos de Hubel y Wiesel revelaron que en la corteza visual primaria del cerebro existen neuronas con respuestas, simples y complejas, a estímulos visuales. Este trabajo marcó la historia de la visión por computadora al proporcionar una base sólida para comprender cómo el sistema visual biológico procesa la información visual. Esta investigación sentó las bases para futuros avances en la creación de sistemas de visión artificial, convirtiéndose así en una trabajo fundamental e influyente en el campo de la visión por computadora.

En ese mismo año, Russell Kirsch se convirtió en el primer hombre en crear una imagen digital. Russell Kirsch, haciendo uso de un escáner, transformó imágenes en una matriz de números, para posteriormente traducirla a lenguaje binario y así lograr interactuar con una computadora, dando inicio a la visión por computadora. La figura 4.1 muestra la primera imagen digital, en la cual aparece el hijo pequeño de Russell, compuesta por una matriz de 176x176 pixeles.

Figura 4.1 La primera imagen digital

Años después, en 1963, Lawrence Roberts en su tesis doctoral titulada ***Machine Perception Of Three-Dimensional Solids***, expone un modelo por computadora, capaz de construir todas las percepciones visuales de un objeto tridimensional a partir de una sola imagen.

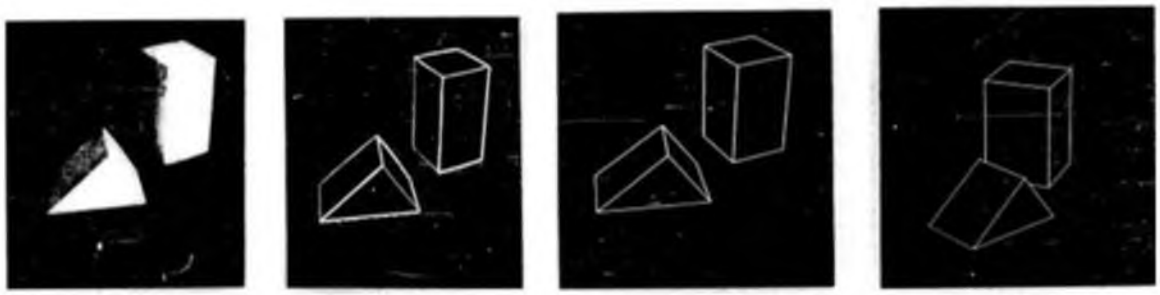

Figura 4.2 Diferentes perspectivas visuales de objetos a partir de una imagen 2D

A finales de la década de los sesenta, la visión por computadora comenzó a considerarse un campo científico. Uno de los hechos clave en esta consolidación fue la creación del proyecto ***The Summer Vision Project*** en 1966, del Instituto Tecnológico de Massachusetts (MIT). El objetivo inicial de este proyecto era involucrar a estudiantes y empleados del MIT en la tarea de desarrollar algoritmos para sistemas de visión artificial durante el verano. Sin embargo, lo que comenzó como un proyecto aparentemente simple se convirtió rápidamente en un gran desafío que, hasta la fecha, no ha sido completamente resuelto. Este proyecto demostró la complejidad de la visión por computadora y la necesidad de abordar una serie de desafíos técnicos y teóricos para lograr avances en este campo.

Posteriormente en 1974 Ray Kurzweil desarrolló la tecnología OCR (Reconocimiento Óptico de Caracteres). Esta tecnología permitió la conversión de texto impreso en formatos digitales editables. La tecnología OCR dio origen a nuevas posibilidades para la digitalización y el análisis de datos visuales.

Años más tarde, en 1982, David Marr publicó su libro "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information". Este libro estableció un marco teórico importante en el desarrollo de la visión artificial y la visión por computadora. En esta obra Marr planteó que los algoritmos de bajo nivel se utilizan como peldaños hacia una comprensión de alto nivel, además de que el procesamiento visual se realiza de manera jerárquica (Marr, 2010). El algoritmo de Marr se divide en las siguientes tres etapas:

En la **primera etapa**, la representación de la información visual se realiza en forma de un "boceto primario". Se identifican bordes, barras, límites y otras características básicas de la imagen. Estos elementos visuales son fundamentales para reconocer objetos y escenas. En la **segunda etapa** se realiza una representación en la que se unen diversos elementos de la imagen, tales como: superficies, profundidades y discontinuidades en la imagen, lo que permite obtener una apariencia tridimensional sin necesidad de una representación 3D completa. Esto es importante para comprender la estructura y la ubicación de los objetos en el espacio. Y en la **última etapa** se desarrolla un modelo 3D, el cual está organizado jerárquicamente en términos de primitivas, objetos elementales (círculos, triángulos, esferas, cilindros), superficiales y volumétricas.

En el año 1999, Davi G. Lowe en su artículo "Object Recognition from Local Scale-Invariant Features" presenta un sistema capaz de lograr reconocimiento de objetos en imágenes con múltiples objetos. En este sistema se propone un enfoque basado en características locales y escalares invariables, es decir, en lugar de analizar una imagen completa, se identifican características locales clave dentro de la imagen, estas características locales suelen ser puntos de interés, tales como: esquinas, bordes, y otros elementos distintivos que son robustos frente a cambios de escala, rotación e iluminación. Este artículo ocasionó un creciente interés en el reconocimiento de objetos basado en características locales y ha dado lugar a numerosas investigaciones y aplicaciones en diversos campos, tales como: reconocimiento de patrones, visión por computadora, robótica, visión artificial, entre otros.

Ese mismo año, Gary Bradsky, trabajador de Intel (Fundador de OpenCV), se dio cuenta que era posible aprovechar la experiencia acumulada en grandes universidades y centros de investigación en el campo de la visión por computadora. Notó que en estos lugares se contaba con infraestructuras avanzadas y algoritmos transmitidos de una generación a otra, lo que permitía a las nuevas generaciones desarrollar mejores algoritmos y soluciones en el campo de la visión por computadora. Por esta razón, en el año 1999, se lanzó OpenCV (Open Source Computer Vision Library), una iniciativa para proporcionar infraestructura común para aplicaciones de visión artificial, en colaboración con Intel. Los objetivos del proyecto fueron los siguientes: Avanzar en la investigación de la visión por computadora, proporcionando código abierto y optimizado para la infraestructura de visión básica. Difundir el conocimiento de la visión por computadora, facilitando la transferencia de código.

Actualmente OpenCV es una fundación sin fines de lucro que proporciona software libre y de código abierto, tiene una comunidad de más de 47 mil usuarios y un número estimado de descargas que superan los 18 millones. La biblioteca se utiliza ampliamente en empresas, grupos de investigación y por organismos gubernamentales.

Las aplicaciones de OpenCV son igualmente diversas y globales. Se utiliza para unir imágenes en servicios como Google Street View, para detectar intrusiones en sistemas de vigilancia en Israel, para monitorear equipos mineros en China, para habilitar la navegación y la recolección de objetos por parte de robots en Willow Garage, para la detección de accidentes de ahogamiento en piscinas en Europa, para impulsar el arte interactivo en lugares tan distantes como España y Nueva York, para inspeccionar etiquetas de productos en fábricas en todo el mundo y para realizar la detección rápida de rostros en Japón (OpenCV, 2022).

4.3 OpenCV

OpenCV es una biblioteca de visión por computadora altamente versátil y compatible que ofrece interfaces en varios lenguajes de programación, incluidos C++, Python, Java. Es compatible con los sistemas operativos Windows, Linux, Android y Mac OS, lo que la convierte en una solución accesible para una amplia gama de plataformas.

4.3.1 Tipos de datos

Comprender cómo OpenCV maneja datos visuales, es esencial conocer las estructuras de datos que utiliza (imágenes y matrices). A continuación, se presenta una visión general de estas estructuras de datos.

Las imágenes son la base de muchas aplicaciones de visión por computadora y OpenCV proporciona un conjunto de herramientas para trabajar con ellas. En OpenCV, se utilizan las matrices para adquirir imágenes y posteriormente manipular las propiedades. A continuación, se describen algunas propiedades de las imágenes digitales.

Los **píxeles** representan los valores de una imagen y se almacenan en una matriz, donde cada elemento corresponde a un píxel y contiene información sobre su intensidad de luz. La **resolución** de una imagen se refiere a sus dimensiones, determinando la cantidad de píxeles que componen la imagen y, por lo tanto, la cantidad de detalle que se puede extraer de ella. El **espacio de color**, por otro lado, es un sistema que define cómo se representan y codifican los colores, siendo esencial en la reproducción precisa del color. Ejemplos de espacios de color incluyen BGR, RGB, CMYK, entre otros. Los **canales** son representaciones de componentes de color. La mayoría de los modelos de color utilizan tres canales, como rojo (R), verde (G) y azul (B), pero también se emplean canales monocromáticos (escala de grises). Cada canal almacena información sobre la intensidad de un color específico en cada píxel, y la combinación de los canales determina el color final de un píxel. OpenCV admite varios **formatos de imagen**, como JPEG, PNG, BMP, y más, permitiendo la lectura y escritura de imágenes en diferentes formatos mediante las funciones proporcionadas por OpenCV.

En OpenCV, las matrices también se utilizan para manipular datos numéricos multidimensionales. Almacenan datos de diferentes tipos (enteros, flotantes y otros). La máxima dimensión de las matrices está limitada por las propiedades del hardware y el software del sistema. A demás, el acceso a cada uno de sus elementos es relativamente sencillo, lo que permite realizar operaciones matriciales con facilidad.

4.4 Visión y OpenCV

En el capítulo dos se presentó, como los procesos de la visión humana análogamente pueden ser representados por los que permiten la visión en los sistemas de visión artificial. A continuación se plantea la relación entre herramientas (funciones, métodos y clases) pertenecientes a OpenCV y determinados procesos de los sistemas de visión artificial. Además, se explica como implementar estas herramientas.

4.4.1. Lectura de datos (Transmisión de señales)

En los sistemas de visión artificial, la cámara digital es la encargada de adquirir información visual del entorno. Esta cámara genera una imagen digital que se transmite al procesador. OpenCV brinda acceso a múltiples funciones para leer diversos dispositivos o tipos de datos, tales como: imágenes, videos, cámara, entre otros. Estas funciones son parte de un conjunto de herramientas llamado HighGUI, que se incluye en el paquete OpenCV.

La función **imread (Nombre, Bandera)** permite cargar imágenes que se encuentran almacenadas en la computadora, admite una amplia variedad de formatos de imagen, tales como: JPEG, JPE, PNG, entre otros. La función devuelve una matriz que contiene cada uno de los pixeles de la imagen y recibe dos parámetros, los cuales se explican a continuación.

- El parámetro **Nombre** es obligatorio y debe estar formado por una cadena de caracteres que especifican la ruta completa o relativa de la imagen. (Ejemplo de ruta completa: 'C:\Users\Usuario\Documentos\imagen.jpg'). La ruta relativa especifica la ubicación de imágenes en niveles superiores al directorio de trabajo (Ejemplo: 'imagenes/imagen.jpg').
- El parámetro **Bandera** es opcional y permite especificar el espacio de color con el que se desea cargar la imagen. Si se desea cargar la imagen en el espacio de color BGR, espacio que por defecto maneja OpenCV, se utiliza **IMREAD_COLOR** o **1.** Por otra parte, si se desea cargar la imagen en el espacio de color de escala de grises**,** se utiliza **IMREAD_GRAYSCALE** o **0.**

La clase **VideoCapture (Parametro)** captura video desde diferentes fuentes, tales como: archivos de video, transmisiones en línea o dispositivos de captura (cámaras web). El parámetro que recibe la clase especifica la fuente de video que se desea utilizar, puede ser numérico o una cadena de caracteres. A continuación, se presenta una breve descripción de los parámetros que pueden utilizarse.

- En el caso de utilizar la cámara web se usa como parámetro numérico el **0**. Por otra parte, si se tiene acceso a más cámaras, se puede especificar la cámara utilizando como parámetro **1** para la segunda cámara, y así sucesivamente.
- Para acceder a un archivo de video almacenado localmente, es necesario proporcionar la ruta completa del archivo de video como argumento.
- Es posible acceder a una transmisión de video en línea proporcionando el URL como parámetro.

El método **isOpened ()** verifica si la fuente de video, ya sea un archivo de video, transmisión en línea o dispositivo de captura, se ha abierto correctamente. Este método devuelve un valor booleano true, si la fuente de video se ha abierto con éxito o false, en caso contrario.

El método **read ()** captura un solo fotograma de la fuente de video. Se utiliza comúnmente en un bucle, con el objetivo de realizar un procesamiento en los fotogramas. El método devuelve dos variables, la primera una variable booleana (que permite verificar si se ha logrado capturar con éxito el fotograma) y la segunda un arreglo que contiene los pixeles del fotograma.

La función **Imshow (parametro)** crea una ventana, si no existe, o muestra en pantalla la imagen entregada por el parámetro (contiene el valor de los pixeles de la imagen o fotograma). Es útil para visualizar imágenes y fotogramas.

4.4.2 Procesamiento

En esta etapa se realiza el procesamiento visual previó a la extracción de características. El procesamiento de imágenes permite realizar diversas operaciones, tales como: cambiar de espacios de color, aplicar transformaciones geométricas, convertir imágenes a imágenes binarias, filtrar imágenes, realizar transformaciones geométricas, detección de bordes, entre otras. Estas operaciones se usan con la finalidad de obtener características que permitan a los algoritmos identificar su entorno. OpenCV brinda múltiples funciones para el procesamiento

de imágenes. A continuación, se presentan algunas funciones encargadas de realizar procesamiento de imágenes.

La función **cvtColor (imagenEntrada, bandera)** cambia de espacio de color una imagen. Recibe dos parámetros: imagenEntrada y bandera. El primer parámetro está formado por una cadena de caracteres, la cual corresponde al nombre de la imagen o variable que contiene la imagen. El parámetro bandera indica la conversión deseada, como ejemplo; **COLOR_BGR2GRAY** convierte una imagen, que se encuentra en el espacio de color BGR, a una imagen en escala de grises.

La función **getRotationMatrix2D (centro, ángulo, escala)** obtiene una matriz de rotación empleada para girar una imagen alrededor de un punto de origen. Recibe tres parámetros. El parámetro centro indica un par de coordenadas (x, y), que especifican el punto de origen alrededor del cual se realiza la rotación. El ángulo indica los grados de rotación (un valor positivo indica una rotación en sentido antihorario, y un valor negativo indica una rotación en sentido horario). El último parámetro es un factor de escala, que se aplica a la imagen después de la rotación, si no se desea modificar la escala se usa 1.

La función **warpAffine (imagen, matrizRotación, dimensiones)** rota una imagen. Recibe tres parámetros, el primer parámetro contiene la imagen que se desea rotar, el segundo parámetro llama a la matriz de rotación y el último parámetro es un vector que contiene las dimensiones de la imagen (número de columnas, número de filas).

La función **cv.threshold(imagenGris, valorUmbral, maxValor, tipoUmbral)** convierte una imagen en escala de grises a una imagen binaria, mediante la aplicación de un umbral (devolviendo el valor de umbral y la imagen resultante). La función recibe cuatro parámetros, el primer parámetro corresponde a la imagen de entrada, el segundo indica el valor de umbral (a los pixeles con valor mayor al umbral se le asigna el valor máximo definido). El Parámetro maxValor es el valor máximo que se le establece a los pixeles con valor superior al umbral, por ejemplo, se establece un valor máximo de 255 para obtener un pixel de color blanco para BGR con resolución de 8 bits por pixel. El ultimo parámetro, especifica el tipo de umbral que se desea aplicar. Los valores comunes para este parámetro son los siguientes:

- **THRESH_BINARY**: A Los píxeles con valor superior al umbral se les asigna maxValor, y a los que se encuentren por debajo se les asigna el valor 0.
- **THRESH_BINARY_INV**: A Los píxeles con valor superior al umbral se les asigna el valor 0, y a los que se encuentren por debajo se les asigna maxValor.
- **THRESH_TRUNC**: Los píxeles con valor superior al umbral, mantienen el mismo valor, y a los que se encuentren por debajo se les asigna el umbral.
- **THRESH_TOZERO**: Los píxeles con valor superior al umbral, mantienen el mismo valor, y a los que se encuentren por debajo se les asigna el valor 0.

OpenCV tiene diversas funciones para el filtrado de imágenes. El filtrado de imágenes se utiliza para reducir el ruido. Algunas de las técnicas de suavizado más comunes son el suavizado gaussiano y el suavizado de la media. A continuación, se explica cómo realizar estos dos tipos de suavizado.

Un kernel es una unidad básica utilizada en el procesamiento de imágenes como base para aplicar una función de OpenCV. Es una matriz cuadrada de dimensión impar, relativamente pequeño comparado con la imagen de entrada.

En la función **GaussianBlur (inputArray, ksize, 0)**, conocida como desenfoque gaussiano, se debe especificar el ancho y la altura del kernel. Se especifica también la desviación estándar en las direcciones X e Y con sigmaX y sigmaY respectivamente. Si sólo se especifica sigmaX, sigmaY se toma igual que sigmaX. Si ambos se dan como ceros, se calculan a partir del tamaño del kernel. El desenfoque gaussiano es altamente efectivo para eliminar el ruido gaussiano de una imagen.

La función **medianBlur ()** toma la mediana de todos los píxeles bajo el área del kernel y el elemento central se reemplaza con este valor mediano. En los filtros anteriores, el elemento central es un valor recién calculado que puede ser un valor de píxel en la imagen o un nuevo valor, sin embargo, con la función **medianBlur ()** el elemento central siempre se reemplaza por algún valor de píxel en la imagen. Esto Reduce el ruido de manera efectiva.

4.4.3 Extracción de Características

La función **findContours()** se utiliza para encontrar contornos en una imagen binaria o en escala de grises. Los contornos son definidos como conjuntos de puntos continuos que forman una curva, correspondiente a los bordes de objetos. Esta función es especialmente útil para el análisis de formas, detección y reconocimiento de objetos.

Las esquinas son regiones de la imagen con gran variación de intensidad en todas las direcciones. Uno de los primeros métodos para la detección de esquinas fue desarrollado por Chris Harris y Mike Stephens, el cual actualmente se utiliza en la función de OpenCV **cornerHarris(Img, BlockSize, Ksize, K)**. Img es la imagen de entrada en escala de grises tipo float32. BlockSize es el tamaño del vecindario considerado para la detección de esquinas. Ksize es el parámetro de apertura de la derivada de Sobel utilizada y K es el parámetro libre del detector de Harris es la ecuación.

4.5 Conclusiones

En este capítulo, se ha explorado el campo de estudio de la visión por computadora, cuyo propósito es dotar a las máquinas con la capacidad de percibir y comprender su entorno a través de imágenes digitales.

Se explora a grandes rasgos la librería OpenCV. Se presentan algunos tipos de datos que manipula y cómo estas estructuras son empleadas para procesamiento de datos visuales. Además, se presentaron funciones clave para la lectura de datos, procesamiento de imágenes y extracción de características, utilizadas en este trabajo.

Se concluye este capítulo destacando que la visión por computadora, respaldada por herramientas como OpenCV, no solo ha revolucionado la manera en que las máquinas perciben su entorno, sino que también ha abierto la puerta a diversas aplicaciones prácticas. A medida que la tecnología continúa evolucionando, la visión por computadora promete seguir desempeñando un papel importante en la automatización y la mejora de la calidad de vida.

Capítulo 5 Implementación

"La autonomía de los robots y los vehículos autónomos no se trata de reemplazar a los humanos, sino de empoderarlos para que puedan enfocarse en tareas más significativas" - Sebastian Thrun

5.1 Introducción

El objetivo principal de esta investigación es desarrollar un algoritmo de reconocimiento de patrones capaz de identificar figuras geométricas básicas. Se establecieron siete figuras como objeto de estudio: triángulo, cuadrado, rectángulo, pentágono, estrella, círculo y elipse.

Se eligió emplear el lenguaje de programación Python en el desarrollo del algoritmo debido a su versatilidad y facilidad de uso. Además, se optó por aprovechar una de las librerías que nos ofrece Python para el desarrollo de algoritmos en el campo de la visión artificial, OpenCV.

Además, se consideró la implementación del algoritmo en un robot móvil haciendo uso de una cámara digital. Este robot móvil debe ser capaz de llevar a cabo una serie de acciones específicas, como avanzar, retroceder, girar a la derecha, girar a la izquierda y otras, en función de la figura geométrica que se identifica.

A lo largo de esta sección, se presentará en detalle los pasos seguidos para abordar la implementación del algoritmo de reconocimiento de patrones en un robot móvil. Se describirán las herramientas utilizadas en el proceso, así como los problemas y resultados obtenidos.

5.2 Algoritmo de reconocimiento de patrones

El reconocimiento de patrones es un proceso que implica la extracción de características relevantes de los datos y la construcción de modelos, para generalizar y clasificar patrones similares en datos nuevos. Estos patrones pueden presentarse en diversas formas, como imágenes, sonidos, texto o datos numéricos. El objetivo principal del reconocimiento de patrones es permitir a las máquinas reconocer y clasificar patrones.

Este campo tiene aplicaciones en variedad de disciplinas, como visión por computadora, procesamiento de señales, minería de datos, biometría, medicina y muchas otras áreas donde es crucial identificar y comprender patrones en conjuntos de datos.

En este trabajo, los objetos de estudio son figuras geométricas, por tal motivo, se planteó la siguiente pregunta: ¿Cuáles son las características que permiten identificar las figuras geométricas? Para responder esta pregunta se debe recordar las características básicas de las figuras, las cuales se presentan a continuación:

Triángulo

- Tiene tres lados.
- Tiene tres vértices.
- Tiene tres ángulos.
- La suma de los ángulos internos es igual a 180 grados.

Cuadrado

- Tiene cuatro lados de igual longitud.
- Tiene cuatro vértices.
- Tiene cuatro ángulos rectos (90 grados).

Rectángulo

- Tiene cuatro lados.
- Tiene cuatro vértices.
- Tiene cuatro ángulos rectos (90 grados).
- Los lados opuestos son paralelos y de igual longitud.

Pentágono

- Tiene cinco lados.
- Tiene cinco vértices.
- La suma de los ángulos internos es igual a 540 grados.

Estrella de 5 picos

- Tiene diez lados.
- Tiene diez vértices.

Círculo

- No tiene lados ni ángulos.
- Todos los puntos en el borde están equidistantes del centro.

Elipse

- No tiene lados ni ángulos.
- No todos los puntos en el borde están equidistantes del centro.

Se decidió clasificar las figuras a partir del número de vértices y una relación de tamaños.

Con el conocimiento adquirido y presentado en el capítulo tres, se realizó un primer algoritmo de reconocimiento de patrones. Este tomaba como entrada imágenes digitales almacenadas en la computadora, las cuales se muestran continuación.

Figura 5.1 Figuras geométricas empleadas para el reconocimiento

Con base en que el número de vértices (puntos de intersección entre líneas, segmentos o bordes) proporciona una forma para clasificar a las figuras, se consideraron las siguientes preguntas: ¿Cómo una computadora es capaz de detectar vértices?, ¿Cómo es capaz de detectar líneas? La única variable que se tenía para desarrollar el algoritmo era la intensidad

de luz que nos proporcionan los pixeles que conforman la imagen digital. Como se explicó en el capítulo dos, una imagen digital es una matriz. Se pensó que la matriz podía ser vista como un plano y cada pixel podía representar un punto en el mismo. A partir de dos puntos, es posible obtener la ecuación de una recta y a partir de la ecuación de dos rectas se puede saber si se intersecan. Por otra parte, esta idea conlleva el siguiente cuestionamiento ¿cómo le indicamos al algoritmo que pixel corresponde a la recta? Se pensó que, si la imagen fuera blanca y negro se podría distinguir fácilmente una línea recta. No se consideraron otros aspectos, como ¿Qué pasa si los pixeles del contorno y los pixeles internos tienen el mismo color? En este punto se estableció que detectar vértices no es una tarea sencilla y que probablemente requeriría hacer uso de muchos recursos a nivel de hardware.

En una revisión de las herramientas que proporciona OpenCV se encontró el detector de esquinas Harris. En esta función se requiere como entrada una imagen en escala de grises, sin embargo, la imagen de entrada a OpenCV está, por defecto, en formato BGR. Por tal motivo, se requiere la conversión de una imagen en formato BGR a una imagen en escala de grises. Para lograr esta conversión, se suelen utilizar varias fórmulas o métodos que ponderan los canales de color de la imagen BGR para obtener un solo valor de intensidad en escala de grises. A continuación, se presenta la fórmula comúnmente empleada para realizar esta conversión:

$$ValorEscalaGrises = 0.299 * R + 0.587 * G + 0.114 * B \quad (3)$$

Donde, R corresponde al valor de intensidad del pixel en el canal Rojo, G corresponde al valor de intensidad del pixel en el canal verde y B corresponde al valor de intensidad del pixel en el canal azul.

Se realizó un algoritmo que convierte la imagen en formato BGR, pixel a pixel, en una imagen en escala de grises. Este algoritmo consumía demasiados recursos del hardware lo que disminuía considerablemente la velocidad de respuesta, por lo que se buscó una alternativa. Se usó la función que proporciona OpenCV para la conversión de imágenes a escala de grises.

Terminado el algoritmo, para detectar el número de esquinas en la imagen, se clasificaron las figuras por el número de vértices, logrando así la detección de las figuras de interés.

El desarrollo del algoritmo de reconocimiento de patrones junto con el conocimiento adquirido en la revisión bibliográfica, permitieron entender el proceso mediante el cual se implementa el reconocimiento de patrones en una computadora.

La Figura 5.2 muestra el diagrama de flujo del proceso que se consideró para el reconocimiento de patrones.

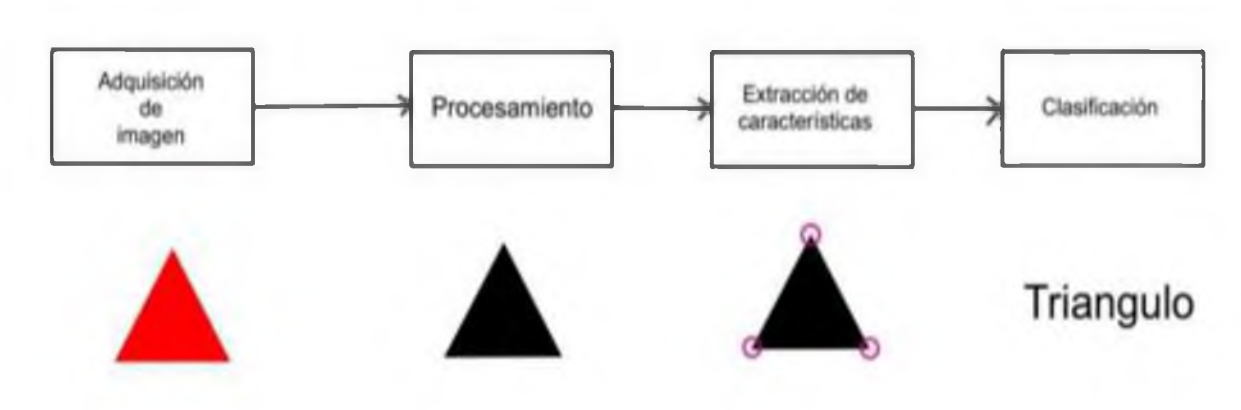

Figura 5.2 Diagrama de flujo del proceso de reconocimiento de patrones

5.3 Implementación del algoritmo en un robot móvil

Para la segunda etapa de este proyecto, se implementó el algoritmo en un robot móvil. Por tal motivo, se modificó la etapa de adquisición de imagen introduciendo una cámara digital.

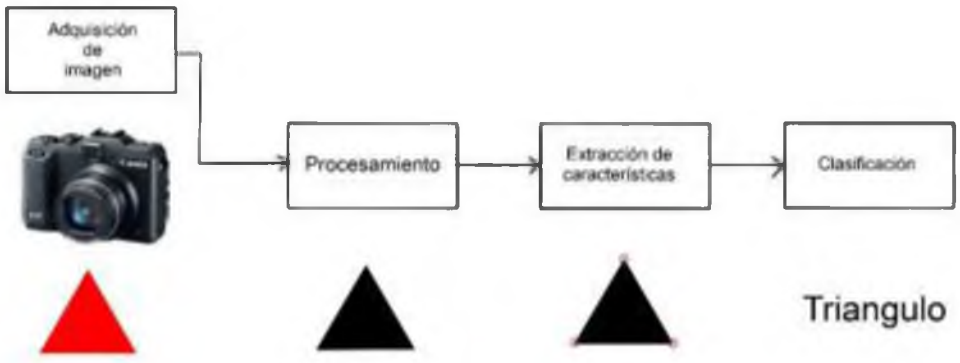

Figura 5.3 Diagrama de flujo del proceso de reconocimiento de patrones modificado

El objetivo es hacer uso de una cámara digital en la etapa de **adquisición de imagen**, como se observa en la Figura 5.3. Para tal efecto se requiere una cámara digital, pequeña con resolución aceptable y un sistema embebido para realizar el procesamiento.

5.3.1 Selección del sistema embebido

Actualmente, existen muchos módulos con cámaras digitales empleados para el diseño de visión artificial en robots móviles. Una alternativa de bajo costo, en el mercado, es el sistema embebido ESP32-CAM. El módulo ESP32-CAM es un SoC (Sistem On Chip) desarrollado por Espressif System MR, el cual permite la comunicación por medio de Wi-Fi, Bluetooth, puertos GPIO y una variedad de interfaces de comunicación. El módulo incluye una cámara OV2640 la cual está constituida por un sensor de imagen CMOS UXGA (1632 x 1232), el sensor es pequeño en tamaño y bajo en voltaje de funcionamiento, proporcionando las mismas funciones de una cámara UXGA de un solo chip y procesador de imagen, lo que permite la captura de imágenes y video de acuerdo a las necesidades de este trabajo.

Inicialmente se trabajó con este sistema embebido, debido a su versatilidad de aplicaciones, desde la captura de imágenes hasta la comunicación inalámbrica y el control de dispositivos externos. Su capacidad de integración lo hacen muy popular en proyectos de electrónica y robótica, así como en aplicaciones relacionadas con la visión por computadora.

La Figura 5.4 muestra una vista superior de tarjeta de desarrollo de módulo de cámara wifi ESP32-CAM.

Figura 5.4 ESP32-CAM

El módulo ESP32-CAM cuenta con 16 pines, los cuales están distribuidos de la siguiente manera.

- Tres pines de tierra GND
- Dos pines alimentación, uno de 5V y otro de 3,3V.

- Diez pines de entrada o salida I/O
- Un pin de salida de potencia VCC que genera 5V o 3.3V.

El módulo no cuenta con un puerto físico destinado a la carga de código, por lo que se requiere un convertidor TTL o en todo caso una placa Arduino. Utilizando para la comunicación serial los pines IO1 y IO3 del módulo ESP32-CAM, se conectan a los pines RX y TX de la placa Arduino (RX (IO1) es el pin receptor de datos, TX (IO3) el pin de transmisión de datos). Cuando el pin IO0 del módulo ESP32-CAM se conecta a tierra la tarjeta, entra en modo intermitente, es decir, se puede cargar código a la placa. La figura 5.5 muestra la el diagrama de conexión, entre la placa de Arduino y el módulo ESP32-CAM, empleado para cargar código al módulo ESP32-CAM.

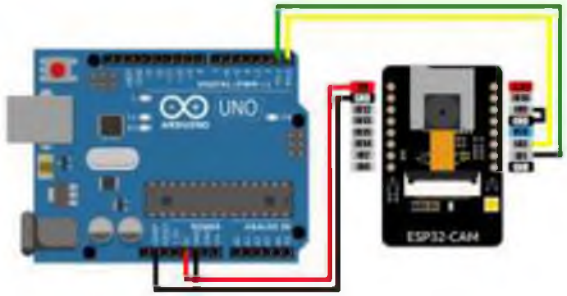

Figura 5.5 Diagrama de conexión para cargar código al módulo ESP32-CAM

En la figura 5.6 se presenta el diagrama de flujo del algoritmo, así como la conexión, de la computadora con el módulo ESP32-CAM.

Figura 5.6 Diagrama de implementación

En la etapa de adquisición de la imagen, el módulo ESP32-CAM capta la intensidad de luz y la transduce a una señal digital, a través de la cámara. La señal digital es transmitida, vía comunicación serial, a la computadora mediante el entorno de programación Python, haciendo uso de la librería OpenCV.

Una vez realizada la adquisición de imágenes, el siguiente paso es evaluar el algoritmo, sin embargo, este sistema embebido no cuenta con la capacidad de procesamiento requerida para la implementación del algoritmo de detección de vértices.

El hecho de no poder integrar todo el sistema para el robot en la ESP32_CAM, brindó la perspectiva necesaria para analizar en forma global el sentido de la visión. Dado que se desea que el robot realice el proceso de visión con las capacidades propias, de forma análoga al sistema de visión humano, una buena hipótesis es tratar de emular todos los procesos realizados por el sentido de visión en el robot. Esta hipótesis nos dio nuevas perspectivas sobre el trabajo, por lo que se decidió trabajar con un sistema embebido con mayor capacidad de procesamiento, Raspberry Pi. Este sistema sigue siendo de bajo costo, tamaño reducido y otorga mayores prestaciones, siendo la única desventaja con respecto a la ESP32-CAM que consume mayor potencia.

5.3.1.1 Raspberry Pi

La Raspberry Pi fue creada por la fundación Raspberry Pi, una organización benéfica formada con el objetivo de estimular la enseñanza de ciencias computacionales en las escuelas. El nombre, Raspberry Pi, fue combinación del deseo de seguir la tradición de empresas que utilizan nombres basados en frutas y un guiño a la constante matemática. Su comercialización empezó en el año 2012 y está diseñada para ejecutar el sistema operativo Raspberry Pi OS, sistema operativo gratuito basado en Debian, optimizado para el hardware de Raspberry Pi. El sistema operativo se ejecuta desde una tarjeta microSD, lo que permite realizar diversas tareas, tales como: programar, ejecutar aplicaciones, navegar por la web, entre otras.

La Raspberry Pi cuenta con los componentes típicos de una computadora (Figura 5.7): procesador, memoria RAM, puertos de entrada y salida, entre otros. Los puertos de entrada y salida permiten la conexión con periféricos y dispositivos externos, mediante: USB, HDMI, Ethernet, ranura de tarjetas SD, Jack 3.5 mm, GPIO, cámara (CSI), pantalla (DSI) y conexión inalámbrica (Pi, s.f.).

Figura 5.7 Puertos de la Raspberry Pi

El sistema operativo Raspberry Pi Os es compatible con amplia variedad de lenguajes de programación, uno de los más populares y ampliamente utilizados es Python, conocido por su facilidad de uso y versatilidad. Python viene preinstalado en la Raspberry pi. Es posible programar directamente desde la terminal o usando un entorno de desarrollo integrado como Thonny. Python tiene una gran cantidad de bibliotecas y módulos, a su disposición, que hacen que sea fácil realizar diversas tareas, desde la programación de hardware GPIO (entrada/salida) hasta la implementación de proyectos de visión artificial.

5.3.2 Implementación del algoritmo en la Raspberry Pi

La elección del sistema embebido Raspberry Pi simplificó significativamente la integración del proyecto. La presencia de puertos específicos para cámara en la Raspberry Pi permitió la sustitución directa del módulo ESP32-CAM por una cámara compatible con la Raspberry Pi. Además, la capacidad de la Raspberry Pi para ejecutar el sistema operativo Raspberry Pi OS, junto con su facilidad para trabajar con Python, facilitó la implementación del algoritmo de reconocimiento de patrones desarrollado previamente.

Al intentar implementar el algoritmo en la Raspberry Pi, no se tenía comprensión completa de los procesos que se realizan en el sentido de la visión humana. El sentido de la visión se utiliza de forma natural, a diario y gracias a él se percibe el entorno, permitiendo llevar a cabo diversas acciones. En este punto el conocimiento era limitado, se sabía que lo ojos captan la luz y que esta se refleja en la retina para que el cerebro procese señales eléctricas y permita la visión,

sin embargo, no se comprendía en su totalidad los procesos que ocurren desde que la luz se refleja en los objetos hasta el momento en el que somos capaces de percibir el entorno. Tampoco se era consciente que, realizar una acción es una parte integral de este sentido.

La revisión bibliográfica realizada, acerca de los procesos involucrados en el sentido de la visión, permitió concluir que, un sistema robótico con percepción visual que emule todos estos procesos y que además sea capaz de realizar las tareas visuales, no es tecnológicamente realizable aún. Requiere conocimientos profundos y complejos, así como el uso de recursos a nivel de hardware y software que posiblemente no se han desarrollado. Cada etapa presenta diversas dificultades, tal es el caso de la **percepción constante**, etapa donde la principal dificultad es desarrollar un algoritmo insensible a los cambios repentinos de intensidad de luz.

Debido a las dificultades presentadas en párrafos anteriores, se decidió simplificar la implementación del sistema de visión a los siguientes procesos:

- Detección, transducción y transmisión de señales.
- Procesamiento.
- Extracción de características.
- Reconocimiento.
- Acción y Control.

Estas etapas redefinen la implementación del algoritmo desarrollado, como se muestra en el diagrama del flujo de señales de la figura 5.9.

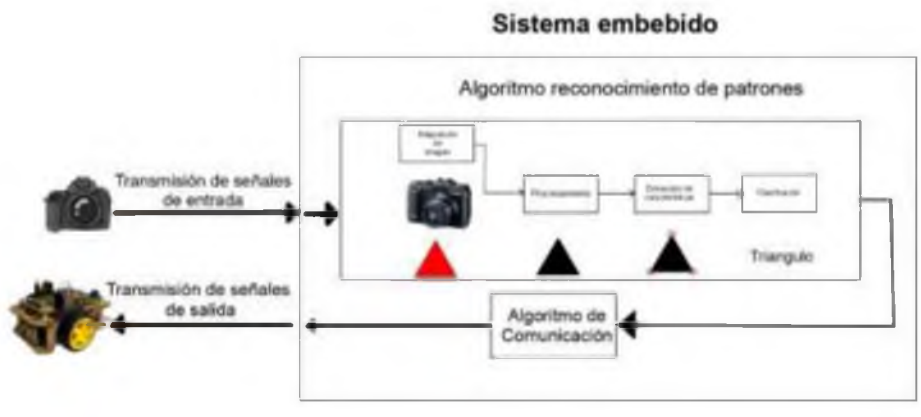

Figura 5.8 Diagrama de flujo del sistema

El proceso inicia cuando la luz se detecta y transduce en señales eléctricas, mediante una cámara digital. Las señales generadas por la cámara digital son transmitidas al sistema embebido. En este sistema, el algoritmo de reconocimiento de patrones adquiere las señales en forma de una imagen digital, para su procesamiento, extracción de características y clasificación. Esto es lo que se denomina reconocimiento de patrones. El sistema embebido ejecuta un algoritmo de comunicación, usando la información generada en el reconocimiento de patrones para que el robot tome una decisión y realice una acción.

A continuación, se explican cómo se abordó la implementación en cada una de los procesos del sistema.

5.3.3 Detección, transducción, trasmisión de señales y reconocimiento.

El algoritmo de reconocimiento de patrones, implementado en la computadora, ya tenía una etapa de **procesamiento** de imágenes. Por esta razón, se pensó que esta etapa del proceso de visión estaba completa, sin embargo, al realizar pruebas con la cámara, se observó que un sector de la imagen (las ultimas columnas) no presentaban color. Esta modificación no deseada de la imagen, provocaba que el algoritmo no funcionara correctamente. Por esta razón, se recortó la imagen digital, de tal forma que se excluyeran las columnas de pixeles que no tenían información real del entorno.

Posteriormente, la imagen recortada se transforma en una imagen en escala de grises y se utiliza la función de filtro gaussiano de OpenCV para eliminar el ruido. Una vez realizado este procesamiento, se aplica la función umbral con el objetivo de separar el fondo de la figura geométrica. Se propuso un umbral de 112, con el cual los pixeles con un valor de intensidad menor a 112 toman valores de intensidad de 0 y se considera que forman parte del fondo. En caso de que los pixeles tengan un valor de intensidad igual o superior a 112 se considera que forma parte de la figura y los pixeles toman un valor de 255.

Como resultado de estos procesos se obtienen dos imágenes, que son empleadas en etapas posteriores, una imagen en escala de grises y otra donde se aplicó la función umbral.

Con las modificaciones aplicadas al algoritmo de reconocimiento de patrones, se realizaron pruebas para evaluar la etapa de **extracción de características**. Se observó que, en

ocasiones, cuando no había ninguna figura en el campo de visión de la cámara, el algoritmo detectaba esquinas inexistentes. Los cambios repentinos de luz provocaban que se detectaran pixeles con valor de 255, haciendo que el algoritmo fallara. Para eliminar esta falla, se identificó el número de contornos presentes en la imagen digital con el uso de la función contorno. Cuando se detecta un solo contorno, el algoritmo reconoce la figura. En caso de detectar un mayor número de contornos, el algoritmo sigue identificando el número de contornos hasta que solo detecte uno. En otras palabras, para obtener un reconocimiento aceptable, el algoritmo realiza la detección de esquinas únicamente si se ha detectado un solo contorno en su campo de visión.

En el proceso de **reconocimiento**, el algoritmo clasifica las figuras, basado en el número de esquinas detectado, asignando un número a cada una de las figuras y almacenándolo en una variable.

5.3.4 Acción y control

En este caso, un robot móvil realiza una serie de movimientos, en función de la figura que se detecte. Se optó trabajar con un robot móvil de configuración diferencial perteneciente al laboratorio de Ingeniería en Mecatrónica de la Universidad de Papaloapan, campus Loma Bonita. El robot móvil está equipado con dos actuadores acoplados a las llantas, un sensor ultrasónico, tres sensores infrarrojos, una batería LiPo, una pantalla LCD y una placa Arduino Uno.

Para integrar la Raspberry Pi y la cámara se modificó parte de la estructura mecánica y electrónica del robot (Figura 5.9). La integración de estos elementos planteó la necesidad de determinar la forma de transmitir datos entre la Raspberry Pi y la placa Arduino.

Figura 5.9 fotografía del robot móvil modificado.

5.3.4.1 Comunicación entre el robot y el sistema de visión

Existen diversos métodos de comunicación para la transmisión de datos entre sistemas embebidos. Algunos que se pueden mencionar en este trabajo son: **comunicación inalámbrica**, que utiliza ondas electromagnéticas para la transmisión sin cables físicos (Wi-Fi, Bluetooth, entre otras); **comunicación por fibra óptica**, que utiliza luz para transmitir datos a través de fibras ópticas, ofreciendo velocidades de transmisión extremadamente altas e inmunidad a interferencias electromagnéticas; **comunicación paralela**, que transmite múltiples bits simultáneamente, proporcionando velocidades de transferencia rápidas, aunque puede requerir más cables y ser propensa a interferencias; y **comunicación serial**, que implica la transmisión de datos bit a bit a través de un solo canal.

Debido a su simplicidad y eficiencia en la transmisión de datos, se seleccionó la comunicación serial. Este método de comunicación ha demostrado ser una buena opción en entornos donde se requiere un flujo constante de información de manera ordenada y confiable. Este método requiere la implementación de protocolos de comunicación adecuados, los cuales desempeñan el papel primordial de definir las reglas y estándares para la transmisión de información, facilitando la sincronización y la interacción entre los dispositivos. Existe una gran variedad de protocolos de comunicación, entre los cuales podemos encontrar los siguientes: comunicación serial UART, comunicación I2C (Inter-Integrated Circuit), comunicación SPI (Serial Peripheral Interface) y comunicación serial.

La comunicación serial UART es uno de los protocolos más comunes. Utiliza dos líneas, una para transmitir datos (TX) y otra para recibir datos (RX). Opera de manera asíncrona, lo que significa que no requiere un reloj compartido entre los dispositivos. La simplicidad de UART lo hace versátil y fácil de implementar, siendo utilizado en una variedad de aplicaciones, desde la comunicación entre microcontroladores hasta la conexión de periféricos.

El protocolo SPI (Serial Peripheral Interface) está diseñado para la comunicación sincrónica entre un servidor y varios dispositivos clientes, SPI utiliza las siguientes cuatro líneas: MISO (Master In Slave Out), MOSI (Master Out Slave In), SCK (Serial Clock), y una conexión $\overline{SS/CS}$ (Slave Select/Chip Select) por cada dispositivo serial a comunicar. Aunque es más complejo que UART, SPI permite tasas de transferencia más rápidas y se utiliza comúnmente en la comunicación entre microcontroladores y sensores, así como en la programación de dispositivos como memorias flash.

I2C (Inter-Integrated Circuit) es un protocolo de comunicación serie de dos hilos que facilita la conexión entre múltiples dispositivos utilizando solo dos líneas: SDA (Serial Data) y SCL (Serial Clock). Es especialmente eficaz cuando se trata de conectar periféricos a una placa base, y su capacidad para soportar múltiples dispositivos en el mismo bus lo hace popular en sistemas embebidos y aplicaciones de electrónica.

En este trabajo, se estableció trabajar con comunicación serial a través de puertos USB-USB. Cuando se conecta una placa Arduino Uno a una Raspberry Pi a través de puertos USB, se utiliza el protocolo serial para habilitar la comunicación serial entre ambos dispositivos. Para su funcionamiento es necesario instalar los controladores apropiados para asegurar que ambos dispositivos se reconozcan y se comuniquen entre sí. La biblioteca Serial de Arduino se puede utilizar para programar la placa y gestionar la comunicación serie a través del puerto USB.

Una vez definido el tipo de comunicación, se programó el algoritmo en la Raspberry Pi, para la transmisión de datos de la Raspberry Pi a la placa Arduino. El algoritmo de reconocimiento de patrones genera un número del uno al siete. Este número indica la figura que se reconoció: si el algoritmo de reconocimiento de patrones detecta un triángulo se manda un uno, si detecta un cuadrado se manda un dos, si detecta un rectángulo se manda un tres, si detecta un pentágono se manda un cuatro, si detecta una estrella se manda un cinco, si detecta un círculo se manda un seis y si detecta una elipse se manda un siete y si no detecta figura, cero.

Posteriormente el algoritmo de comunicación envía, a la placa Arduino, cuatro bits que contienen el número correspondiente a la figura reconocida.

La placa Arduino recibe información sobre la figura detectada, escribiendo el nombre de la figura en la pantalla LCD. El robot móvil reproduce con desplazamientos la figura recibida, esto es si recibe un triángulo, el robot móvil se desplaza en línea recta, gira a la izquierda aproximadamente 60 grados, avanza en línea recta, gira a la izquierda una cantidad similar de grados, avanza en línea recta, gira nuevamente 60 grados a la izquierda para finalizar su movimiento. En forma similar para un cuadrado, el robot sigue una serie de movimientos girando 90 grados a fin de trazar una trayectoria cuadrada. Los movimientos de todas trayectorias se implementan mediante funciones básicas del robot como avanzar y girar. En el caso de que se detecte un círculo o elipse, ambas llantas avanzan en la misma dirección, pero a diferentes velocidades, esto se logra cambiando el valor de PWM, generado por la placa Arduino.

Capítulo 6 Conclusiones y trabajos futuros

El proyecto aborda el diseño, implementación y aplicación de un algoritmo de reconocimiento de patrones para identificar figuras geométricas básicas. La aplicación práctica seleccionada fue un algoritmo de control, para la ejecución de trayectorias en lazo abierto, en un robot móvil. La implementación del algoritmo dota al robot móvil con la capacidad de reconocer figuras geométricas y realizar acciones específicas. Se llevó a cabo una exhaustiva revisión bibliográfica del estado del arte, proporcionando las bases teóricas para abordar los desafíos asociados con la visión por computadora.

Se seleccionó, para desarrollar el algoritmo de reconocimiento de patrones, el entorno de programación Python y se utilizaron diversas herramientas para visión por computadora de la biblioteca OpenCV. El algoritmo desarrollado hace uso de las propiedades geométricas de las figuras, tales como número de vértices y relación de tamaño, para clasificar y reconocer figuras. La detección de esquinas se aborda con el algoritmo de esquinas Harris.

En la fase de implementación en el robot móvil se seleccionó inicialmente el sistema embebido ESP32-CAM para la adquisición de imágenes. Sin embargo, debido a limitaciones de hardware y compatibilidad con OpenCV, se decide cambiar a la Raspberry Pi. La Raspberry Pi se integra con la cámara y se realizan los protocolos de comunicación serial requeridos a fin de poder trasmitir información al robot móvil. Se estableció una comunicación serial USB-USB para la transmisión de datos entre la Raspberry Pi y la placa Arduino del robot, con lo que se terminó la integración del hardware.

El algoritmo de comunicación en la Raspberry Pi envía información codificada sobre la figura detectada a la placa Arduino. Esta información determina los movimientos del robot, siguiendo secuencias específicas para cada figura geométrica. Las acciones del robot incluyen movimientos en línea recta, giros y ajuste de velocidad de las llantas para círculos y elipses.

En términos de desempeño, se evaluó la respuesta del algoritmo de reconocimiento de patrones ante las variaciones de intensidad de luz en el laboratorio. El algoritmo demostró que era eficaz ante pequeñas variaciones de luz en entornos controlados.

6.2 Trabajos a futuro

Implementar el algoritmo de reconocimiento de patrones en un robot móvil es el primer paso en el desarrollo de un algoritmo de visión que permita al robot realizar tareas con mayor dificultad.

Este trabajo se centra en el reconocimiento de figuras geométricas, a partir del número de esquinas presentes en la imagen. La detección de esquinas es importante en el proceso de percepción visual. Estas proporcionan información valiosa sobre la estructura y los límites de los objetos. Son puntos de referencia que destacan en la escena y permiten una representación más detallada de la geometría. Al identificar esquinas, se puede inferir mejor la forma y la orientación de los objetos, contribuyendo a una representación más completa del entorno.

En el seguimiento de objetos a lo largo del tiempo, las esquinas son sensibles a cambios en la escena, como movimientos y deformaciones. Las esquinas sirven como características distintivas que ayudan a asociar correctamente los puntos en diferentes fotogramas.

En robots móviles, el reconocimiento de esquinas es esencial para la percepción del entorno y la toma de decisiones de navegación. Las esquinas pueden servir como puntos de referencia para la planificación de rutas y evitar obstáculos.

El reconocimiento de esquinas ayuda a identificar regiones de interés en una imagen, lo que es de beneficio para la segmentación de objetos y la atención visual, ya que resaltan áreas con información relevante.

En resumen, el reconocimiento de esquinas desempeña un papel importante en la percepción visual al proporcionar información estructural, facilitar la reconstrucción tridimensional, permitir el seguimiento de objetos y contribuir a la toma de decisiones en aplicaciones como la robótica y la visión por computadora.

En futuras investigaciones, se podrían abordar diversos aspectos para mejorar y ampliar la aplicación del sistema propuesto de reconocimiento de figuras geométricas y control de movimiento en entornos robóticos. Algunas áreas sugeridas para investigaciones futuras podrían incluir:

Investigar métodos para mejorar la precisión del algoritmo de reconocimiento de figuras, especialmente en condiciones desafiantes como cambios bruscos en la iluminación o la presencia de ruido en la imagen.

Evaluar la implementación de técnicas de aprendizaje profundo para el reconocimiento de patrones, lo que podría proporcionar una mejora significativa en la capacidad del sistema para generalizar a una amplia variedad de situaciones.

Extender el sistema para reconocer y manejar figuras tridimensionales, incorporando sensores 3D y explorando algoritmos que puedan analizar la geometría en el espacio tridimensional.

Diseñar algoritmos de control que consideren la planificación de trayectorias y la adaptación a cambios dinámicos en el entorno.

Realizar pruebas y evaluaciones en entornos no controlados, como exteriores, para entender mejor el rendimiento del sistema en condiciones del mundo real y desarrollar estrategias para manejar desafíos imprevistos.

Explorar la posibilidad de implementar y coordinar varios robots utilizando el sistema propuesto, permitiendo la interacción y colaboración entre ellos para realizar tareas más complejas.

Estos trabajos a futuro podrían proporcionar una base sólida para la evolución y la mejora continua del sistema propuesto, así como contribuir al avance en el campo de la visión por computadora y la robótica.

Referencias

Alberich, J., Gómez Fontanills, D., & Ferre Franquesa, A. (2013). *Percepció visual.* UOC.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV.* O'REILLY.

Čapek, K. (1920). *Rossum's Universal Robots.*

Cox, T. (2014). *Raspberry Pi Cookbook for Python Programmers.* Packt.

Díaz Hernández, M., González Montenegro, J., Hernández Beleño, R., Durán García, J., & Sánchez Sánchez, N. (2018). *Raspberry Pi 3 y pcDuino: tutorial de instalación y configuración.* Unipiloto.

Dowling, J., & Dowling, J. (2016). *Vision How It Works and What Can Go Wrong.* MIT Press.

Goldstein, E. (2010). *Sensación y percepción.* Wadsworth.

González, R., Rodríguez, F., & Guzman, J. L. (2015). Robots Móviles con Orugas: Historia, Modelado, Localizacion y Control. *Elsevier*, 10.

Honda. (2023). *Honda Asimo*. Obtenido de https://www.honda.mx/asimo

La Serna Palomino, N., & Róman Concha, U. (2009). Técnicas de Segmentación en Procesamiento Digital de Imágenes. *Revista Científica de Sistemas e Informática* , 8.

Marr, D. (2010). *VISION.* MIT Press.

Minichino, J., & Howse, J. (2015). *Learning OpenCV 3 Computer Vision with Python.* Packt.

Newhall, B. (2002). *La Historia de la Fotografia.* GG.

Ollero Baturone, A. (2005). *RÓBOTICA Manipuladores y robots móviles.* Marcombo.

OpenCV. (2022). *OpenCV-Python Tutorials*. Obtenido de https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html

Pi, F. R. (s.f.). *Raspberry Pi*. Obtenido de https://www.raspberrypi.com/documentation/

Reyes Cortés, F. (2011). *Róbotica: control de robots manipuladores.* Alfaomega.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics Modelling, Planning and Control.* Springer.

Siegwart, R., & Nourba, I. (2004). *Introduction to Autonomous Mobile Robots.* MIT Press.

Silva Ortigoza, R., García Sánchez, J., Barrientos Sotelo, V., & A. Molina, M. (2007). UNA PANORÁMICA DE LOS ROBOTS MÓVILES . *Revista Electrónica de Estudios Telemáticos*, 14.

Apéndice A. Algoritmos de reconocimiento de patrones y comunicación

```
import cv2
import numpy as np
import serial, time
#########################################
###### Algoritmo de reconocimiento de patrones #####
#########################################

###### Procesamiento #####
def convertir_BGR_a_Escala_Grises(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    bilateral = cv2.bilateralFilter(gray,9,75,75)
    return (bilateral)

def umbralizacion(img):
    ret, th1 = cv2.threshold(img,130,255,cv2.THRESH_BINARY)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
    erosion = cv2.erode(thi, kernel, iterations = 1)
    return (erosion)
########################

###### Extracción de características ######
def convertir_cany_OpenCV(img):
    canny = cv2.Canny(img,20,50)
    return (canny)
```

```
def deteccion_de_bordes(img, fig):
    contours, Hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    NC = len(contours)
    relacion_aspecto = 0.0
    if (NC > 0):
        for cnt in contours:
            cv2.drawContours(fig, cnt, -1, (255,255,255), 4)
            x,y,w,h = cv2.boundingRect (cnt)
            realcion_aspecto = float(w)/h
    return (NC, relacion_aspecto)
#########################################

##### Clasificación de figuras #####
def reconocimiento_patrones(fig, iman, relacion):
    relacion_aspecto = relacion
    corners = cv2.goodFeaturesToTrack(iman, 30, 0.05, 10)
    approx = len(corners)
    for i in corners:
        x,y = i.ravel()
        cv2.circle(fig, (x,y), 3, (0,255,0), 10)

    if approx<3:
        ObjectType = "No se reconocio la figura"
        R = 0
    elif approx==3:
        ObjectType = "Triangulo"
        R = 0
    elif approx==4:
        if relacion_aspecto == 1:
            ObjectType = "Cuadrado"
            R = 0
        else:
```

```
            ObjectType = "Rectangulo"
            R = 1
        elif approx==5:
            ObjectType = "Pentagono"
            R = 0
        elif approx==10:
            ObjectType = "Estrella"
            R = 0
        elif approx>20:
            if relacion_aspecto == 1:
                ObjectType = "Circulo"
                R = 0
            else:
                ObjectType = "Elipse"
                R = 1
        else:
            ObjectType = "No se reconoce la figura"
            R = 0

        fig = cv2.putText(fig, ObjectType, (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 2, (0,0,255),3,
cv2.LINE_AA)
        return(approx, R)
```

##################################

##

##

```
#################################
###### Algoritmo de comunicación #####
#################################
def enviar_datos(num_lados):
    if num_lados <=2:
            arduino.write(b'0')
    elif num_lados == 3:
            camara.release  ()
            arduino.write(b'1')
    elif num_lados == 4:
        camara.release  ()
        if R==0:
                arduino.write(b'2')
            elif R==1:
            arduino.write(b'3')
            elif num_lados == 5:
                camara.release()
                arduino.write(b'4')
            elif num_lados == 10:
            camara.release()
                arduino.write(b'5')
            elif num_lados > 20:
                camara.release()
                if R==0:
                        arduino.write(b'6')
                elif R==1:
                        arduino.write(b’7’)
            else:
                arduino.write(b'0')

arduino = serial.Serial('/dev/ttyUSBO', baudrate = 9600)
```

```
time.sleep(5)
apagar = 0
camara = cv2.VideoCapture(0) #El 0 indica el
time.sleep(2)

while (apagar == 0):
    cad = arduino.readline().decode('ascii')
    apagar = int (cad);
    print(apagar)
    print("**********")
    resets = arduino.readline().decode('ascii')
    reinicio = int(resets)
    print(reinicio)
    print("%%%%%%%%%%")

##################################
##################################

##### Adquisición de la imagen ######
    if (reinicio <= 2):
        camara = cv2.VideoCapture(0)
        time.sleep(2)
    while(camara.isopened()): # Mientras se este capturando una imagen el ciclo no fermina a menos que se indique por el usuario
        ret, imagen = camara. read() # La funcion read devuelve dos valores, el primer valor es un boleano (True or False)
        ###### Procesamiento #####
        Figura = imagen [0:470,0:630]
        imagenGrisesopenCv = convertir BGR a Escala Grises (Figura)
        imagenUmbralizada = umbralizacion (imagenGrises0pencv)
        ###### Extracción de características ######
        imagenCany = convertir_cany_Opencv(imagenumbralizada)
        NumeroContornos, RA = deteccion_de_bordes (imagenCany, Figura)
```

```
        print (NumeroContornos)
        lados = 0
        ##### Clasificación de figuras #####
        if (NumeroContornos == 1):
            lados,R = reconocimiento patrones (Figura, imagenGrisesOpenCv,  RA)
            print(Lados)
            ######  comunicación #####
            enviar_datos(lados)

        if cv2.waitKey(1) == ord('s')
        break

camara.release()
cv2.destroyAllWindows()

##### Fin del programa #####
```

Apendice B. Algoritmo de control

```
#include <LiquidCrystal.h>
unsigned long lastTime = 0, sampleTime = 1000;
int ValorCNY_I;
int ValorCNY_D;
int ValorCNY_C;
int x=0;
int reinicio=0;
int cont=0;

void setup()
{
   Serial.begin(9600);
   delay(2000);
   lcd.begin(16, 2);
   pinMode(MD1, OUTPUT);
   pinMode(MD2, OUTPUT);
   pinMode(MI1, OUTPUT);
   pinMode(MI2, OUTPUT);
   lastTime = millis();
}

void loop()
{
  apagar();
  reinicio = analogRead(A7);
  if (reinicio <= 2 && figura!=0)
  {
      lcd.setCursor(0,0);
      lcd.print("                ");
      figura = 0;
      cont = 0;
  }
  iniciar();
  while (figura == 0)
  {
    recibir();
  }

  NameLCD();
  lcd.setCursor(0,0);
  lcd.print(nameFigura);
  if (cont <= 1)
  {
    delay(5000);
    Trayectoria();
  }
  Stop();
}
```

```
////////////////////////////////////
void Stop()
{
  digitalWrite(MD1, LOW);
  digitalWrite(MD2, LOW);
  digitalWrite(MI1, LOW);
  digitalWrite(MI2, LOW);
}
////////////////////////////////////
void Adelante()
{
  digitalWrite(MD1, LOW);
  digitalWrite(MD2, HIGH);
  digitalWrite(MI1, HIGH);
  digitalWrite(MI2, LOW);
}
////////////////////////////////////
void Izquierda()
{
  digitalWrite(MD1, LOW);
  digitalWrite(MD2, HIGH);
  digitalWrite(MI1, LOW);
  digitalWrite(MI2, LOW);
}
////////////////////////////////////
void Derecha()
{
  digitalWrite(MD1, LOW);
  digitalWrite(MD2, LOW);
  digitalWrite(MI1, HIGH);
  digitalWrite(MI2, LOW);
}
////////////////////////////////////
void Triangulo()
{
  Adelante();
  delay(5000);
  Stop();
  delay(1000);
  Izquierda();
  delay(3000);
  Stop();
  delay(1000);
  Adelante();
  delay(5000);
  Stop();
  delay(1000);
  Izquierda();
  delay(3000);
  Stop();
  delay(1000);
  Adelante();
  delay(5000);
}
```

```
void Cuadrado()
{
  Adelante();
  delay(2000);
  Stop();
  delay(500);
  Izquierda();
  delay(1000);
  Stop();
  delay(500);
  Adelante();
  delay(2000);
  Stop();
  delay(500);
  Izquierda();
  delay(1000);
  Stop();
  delay(500);
  Adelante();
  delay(2000);
  Stop();
  delay(500);
  Izquierda();
  delay(1000);
  Stop();
  delay(500);
  Adelante();
  delay(2000);
}
////////////////////////////////////
void Rectangulo()
{
  Adelante();
  delay(3000);
  Stop();
  delay(1000);
  Izquierda();
  delay(900);
  Stop();
  delay(1000);
  Adelante();
  delay(5000);
  Stop();
  delay(1000);
  Izquierda();
  delay(900);
  Stop();
  delay(1000);
  Adelante();
  delay(3000);
  Stop();
  delay(1000);
  Izquierda();
  delay(900);
```

```
  Stop();
  delay(1000);
  Adelante();
  delay(5000);

}
////////////////////////////////
void Pentagono()
{
  Adelante();
  delay(5000);
  Stop();
  delay(1000);
  Izquierda();
  delay(1500);
  Stop();
  delay(1000);
  Adelante();
  delay(5000);
  Stop();
  delay(1000);
  Izquierda();
  delay(1500);
  Stop();
  delay(1000);
  Adelante();
  delay(5000);
  Stop();
  delay(1000);
  Izquierda();
  delay(1500);
  Stop();
  delay(1000);
  Adelante();
  delay(5000);
  Stop();
  delay(1000);
  Izquierda();
  delay(1500);
  Stop();
  delay(1000);
  Adelante();
  delay(5000);
}
////////////////////////////////////
void Estrella()
{
  Izquierda();
  delay(300);
  Adelante();
  delay(1000);
  Stop();
  delay(500);
  Izquierda();
```

```
  delay(800);
  Stop();
  delay(500);
  Derecha();
  delay(1400);
  Stop();
  delay(500);
  Adelante();
  delay(1000);
  Stop();
  delay(500);
  Izquierda();
  delay(800);
  Stop();
  delay(500);
  Derecha();
  delay(1400);
  Stop();
  delay(500);
  Adelante();
  delay(1000);
  Stop();
  delay(500);
  Izquierda();
  delay(800);
  Stop();
  delay(500);
  Derecha();
  delay(1400);
  Stop();
  delay(500);
  Adelante();
  delay(1000);
}
//////////////////////////////////////
void Circulo()
{
  digitalWrite(MD1, LOW);
  analogWrite(MD2, 255);
  analogWrite(MI1, 200);
  digitalWrite(MI2, LOW);
}
//////////////////////////////////////
void Elipse()
{
  digitalWrite(MD1, LOW);
  analogWrite(MD2, 255);
  analogWrite(MI1, 200);
  digitalWrite(MI2, LOW);
  delay(2000);
  digitalWrite(MD1, LOW);
  analogWrite(MD2, 255);
  analogWrite(MI1, 240);
  digitalWrite(MI2, LOW);
```

```
  delay(2000);
  digitalWrite(MD1, LOW);
  analogWrite(MD2, 255);
  analogWrite(MI1, 200);
  digitalWrite(MI2, LOW);
  delay(2000);
  digitalWrite(MD1, LOW);
  analogWrite(MD2, 255);
  analogWrite(MI1, 240);
  digitalWrite(MI2, LOW);
}
//////////////////////////////////////
void recibir()
{
     if(Serial.available())
     {
        char option = Serial.read();
        if (option == '0')
        {
          figura = 0;
        }
        if (option == '1')
        {
          figura = 1;
        }
        if (option == '2')
        {
          figura = 2;
        }
        if (option == '3')
        {
          figura = 3;
        }
        if (option == '4')
        {
          figura = 4;
        }
        if (option == '5')
        {
          figura = 5;
        }
        if (option == '6')
        {
          figura = 6;
        }
        if (option == '7')
        {
          figura = 7;
        }
     }
}
```

```
void apagar()
{
  Serial.println(x);
  delay(1000);
}
//////////////////////////////////////
void iniciar()
{
  Serial.println(reinicio);
}
//////////////////////////////////////
void Trayectoria()
{
  if (figura == 0)
  {
    Stop();
  }
  if (figura == 1)
  {
    Triangulo();
    tarea= 1;
  }
  if (figura == 2)
  {
    Cuadrado();
    tarea = 1;
  }
  if (figura == 3)
  {
    Rectangulo();
    tarea = 1;
  }
  if (figura == 4)
  {
    Pentagono();
    tarea = 1;
  }
  if (figura == 5)
  {
    Estrella();
    tarea = 1;
  }
  if (figura == 6)
  {
    Circulo();
    tarea = 1;
  }
  if (figura == 7)
  {
    Elipse();
    tarea = 1;
  }
}
```

```
void NameLCD()
{
  if (figura == 0)
  {
    nameFigura = "No indentificado";
  }
  if (figura == 1)
  {
    nameFigura = "Triangulo       ";
    cont = cont + 1;
  }
  if (figura == 2)
  {
    nameFigura = "Cuadrado        ";
    cont = cont + 1;
  }
  if (figura == 3)
  {
    nameFigura = "Rectangulo      ";
    cont = cont + 1;
  }
  if (figura == 4)
  {
    nameFigura = "Pentagono       ";
    cont = cont + 1;
  }
  if (figura == 5)
  {
    nameFigura = "Estrella        ";
    cont = cont + 1;
  }
  if (figura == 6)
  {
    nameFigura = "Circulo         ";
    cont = cont + 1;
  }
  if (figura == 7)
  {
    nameFigura = "Elipse          ";
    cont = cont + 1;
  }
}
////////// FIN ////////////////////
```