



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**Desarrollo De Software Experimental
Para El Primer Vehículo Autónomo
Tripulado De La UNAM Utilizando
Métodos Convencionales De Visión Por
Computadora**

TESINA

Que para obtener el título de
Ingeniero Mecatrónico

P R E S E N T A

Alan López Ortiz

DIRECTOR DE TESINA

M.I. Mariano García del Gállego



Ciudad Universitaria, Cd. Mx., 2024

«“The scientific man does not aim at an immediate result. He does not expect that his advanced ideas will be readily taken up. His work is like that of a planter — for the future. His duty is to lay the foundation of those who are to come and point the way.”»

Nikola Tesla

Jurado Asignado

Presidente: **Dr. Adrián Espinosa Bautista**
Vocal: **M.I. Mariano García del Gállego**
Secretario: **Dr. Vicente Borja Ramírez**
Suplente 1: **M.I. Antonio Zepeda Sánchez**
Suplente 2: **Dr. Rogelio Soto Ayala**

Resumen

El proyecto consistió en la creación de un algoritmo para transformar los vehículos eléctricos de la escudería Dzec en vehículos autónomos, dividido en dos subsistemas clave: la detección de carril y la estimación de distancia frente a obstáculos específicos.

La estructura del trabajo abarcó tres áreas clave: en el estado del arte se presentó una visión general de los vehículos autónomos y su presencia en México, así como los conceptos y técnicas utilizadas. En la parte de desarrollo, se describieron las pruebas realizadas para implementar el algoritmo, y en los resultados se detallaron los hallazgos obtenidos.

Para lograr esto, se realizaron diversas pruebas con diferentes algoritmos con los que se probó la capacidad de la visión por computadora con métodos convencionales en aplicaciones autónomas.

Dedicatoria

Es un honor para mí dedicar este trabajo a mi familia, quienes siempre han sido mi apoyo incondicional a lo largo de mi vida, ya que sin ellos y su amor nada de esto sería posible, me han enseñado la importancia del trabajo duro y la perseverancia.

Les agradezco por creer en mí, por inspirarme y motivarme a seguir mis sueños y metas académicas. Espero que este logro les haga sentir tan orgullosos como yo me siento agradecido.

Me gustaría que esta culminación tangible de todo este trabajo sea un testimonio de mi gratitud y amor hacia ustedes.

Con cariño y agradecimiento

Alan López Ortiz.

Agradecimientos

Me gustaría expresar mi más sincero agradecimiento a mis padres Rosalba Ortiz y Walter López por darme su amor incondicional, apoyo y un ejemplo de perseverancia y dedicación. Gracias por creer en mí, por inspirarme, motivarme y por estar siempre a mi lado en cada paso del camino. Toda la sabiduría, paciencia y sacrificio han sido un pilar fundamental para mi éxito y nunca podré agradecerles por todo lo que han hecho por mí. Este logro también es suyo.

A mi cómplice y amigo, mi hermano Bruno López. Gracias por apoyarme en los momentos difíciles y por celebrar todos mis logros conmigo.

A mi tía Carmen Ortiz, gracias por tu paciencia y amor, pero, sobre todo, por apoyarme siempre.

A mi abuelita Elena Leyva, gracias por todo. Siempre recordaré las enseñanzas que me diste y la ayuda que me brindaste en cada etapa de mi vida.

Quiero expresar mi más sincero agradecimiento al Maestro Mariano García del Gállego por su compromiso y dedicación en la supervisión de esta tesis. Sus valiosas sugerencias y sabiduría han sido fundamentales para alcanzar los objetivos propuestos, así como su invaluable papel en este importante hito de mi carrera.

Quiero expresar agradecimientos a mis sinodales por su tiempo, atención y contribuciones. Sus comentarios y sugerencias han enriquecido enormemente mi trabajo y han sido fundamentales para su mejora. Agradezco sinceramente su dedicación y compromiso con mi crecimiento académico.

Gracias a mi amigo y mentor, Enrique Munive que, durante todo este proceso, su orientación ha sido valiosa, demostrando constantemente su experiencia y paciencia. Agradezco su dedicación al enseñarme y proporcionarme las herramientas esenciales para alcanzar mis metas académicas. Su apoyo incondicional y motivación han sido fundamentales para seguir adelante. Su experiencia ha sido un regalo invaluable que siempre valoraré.

Al equipo Dzec con quienes he forjado una valiosa amistad. Su dedicación y colaboración han sido fundamentales en el éxito de este proyecto. Estoy agradecido por su apoyo constante y su contribución significativa.

Agradezco también a mis amigos Ángel, Carlos y Diego con los que hice un gran equipo de trabajo durante toda la carrera y que me han apoyado tanto en lo académico como en lo personal.

Y gracias a todos los amigos, familiares y mentores cuyos nombres llenarían esta página, que de una u otra manera marcaron mi historia y todo ello me trajo hasta aquí.

Finalmente quisiera agradecer a la Universidad Nacional Autónoma de México por brindarme las oportunidades y los recursos que necesitaba para mi formación como ingeniero. Gracias por motivarme a explorar nuevas ideas y enfoques. Este logro es también el resultado del compromiso y la excelencia de la máxima casa de estudios.

Contenido

Resumen	III
Dedicatoria	IV
Agradecimientos	v
1. Introducción	1
1.1. Antecedentes	2
1.1.1. Antecedentes de los vehículos de la escudería Dzec	2
1.1.2. Movilidad eléctrica	6
1.1.3. Conceptos	7
1.1.4. Herramientas para la visión por computadora	12
1.2. Estado del Arte	19
1.2.1. Los vehículos autónomos	19
1.2.2. Los vehículos autónomos en México	20
1.3. Definición del problema	21
1.4. Objetivo general	22

1.4.1. Objetivos específicos	22
1.5. Alcances	22
1.5.1. Diseño e implementación de un sistema de detección de carriles y obstáculos	23
1.5.2. Estimación de distancias en tiempo real	23
1.5.3. Adaptación de algoritmos para ambientes reales	23
1.5.4. Desarrollo de hardware	23
1.5.5. Mejoras a futuro	23
1.6. Requerimientos	24
1.7. Justificación	24
2. Metodología	25
2.1. Planeación y selección de herramientas	25
2.2. Desarrollo del sistema en componentes básicos	26
2.3. Integración y pruebas conjuntas de subsistemas	26
2.4. Desarrollo de prototipos de hardware y soportes	27
2.5. Calibración del sistema y ajuste de parámetros	27
2.6. Pruebas finales y validación del sistema	28
2.7. Propuesta de mejoras y trabajo a futuro	28
3. Desarrollo	29
3.1. Apertura de imagen y vídeo	30
3.2. Prueba 1: Detección de carril	30

3.3. Prueba 2: Detección del centro de carril en video	31
3.4. Pruebas de detección de color	31
3.5. Primer prueba de distancia	31
3.6. Primer prueba completa de software	32
3.7. Primer prueba de campo	32
3.8. Segunda Prueba de software	32
3.9. Pruebas de detección de distancia	34
3.10. Segunda Prueba de campo	36
3.11. Prueba final de detección de distancia	36
3.12. Diseño de piezas para el hardware	37
3.13. Prueba Final	38
4. Resultados	40
4.1. Apertura de imagen y video	40
4.2. Prueba 1: Detección de carril	41
4.3. Prueba 2: Detección del centro de carril en video	43
4.4. Pruebas de detección de color	44
4.5. Primer prueba de distancia	47
4.6. Primer prueba completa de software	49
4.7. Primer prueba de campo	50
4.8. Segunda Prueba de software	54
4.9. Pruebas de detección de distancia	55

<i>CONTENIDO</i>	x
4.10. Segunda Prueba de campo	59
4.11. Prueba final de detección de distancia	63
4.12. Diseño de piezas para el hardware	65
4.13. Prueba Final	69
5. Conclusiones	87
5.1. Trabajo a futuro	92
5.1.1. Prueba de comunicación serial	92
5.1.2. Diseño de soportes	94
5.1.3. Sensor LiDar	94
Referencias	96

Capítulo 1

Introducción

Este proyecto desarrolló un sistema autónomo para un vehículo, enfocado en la detección de carriles, la estimación de distancias y la identificación de obstáculos en tiempo real, empleando técnicas tradicionales de visión por computadora. El trabajo se enmarca en el ámbito de la movilidad autónoma y busca aplicar métodos convencionales de visión por computadora para evaluar su efectividad en la navegación segura de vehículos en entornos controlados.

Se inició el proyecto con la selección de herramientas principales, priorizando aquellas que ofrecieran flexibilidad y acceso a librerías de código abierto adecuadas para el procesamiento de imágenes, sin requerir una interfaz gráfica compleja.

El proyecto implementó subsistemas de detección de carriles y color mediante diversas técnicas. Estos subsistemas fueron evaluados tanto en entornos de laboratorio como en condiciones reales, mostrando que, aunque los métodos tradicionales de visión por computadora son en gran medida efectivos, presentan ciertas limitaciones en situaciones más complejas.

Durante el desarrollo, se refinaron los algoritmos para reducir errores y mejorar la precisión del sistema en la estimación de distancias y la alineación en el carril. Para obtener medidas más exactas, se aplicaron regresiones lineales que permitieron mejorar la precisión del sistema en diversas condiciones de prueba.

La estructura del hardware fue diseñada para optimizar la estabilidad del sistema en movimiento. Se fabricaron soportes mediante impresión 3D para sostener las cámaras, y se diseñó una canastilla modular destinada a alojar la electrónica y las baterías, permitiendo

una adaptación versátil a distintos vehículos de prueba.

Finalmente, el proyecto concluye con una serie de recomendaciones y sugerencias de mejora para futuras iteraciones del sistema en condiciones reales de conducción autónoma.

1.1. Antecedentes

1.1.1. Antecedentes de los vehículos de la escudería Dzec

Dzec es un equipo multidisciplinario conformado por diversas carreras de la UNAM como lo son ingeniería mecánica, eléctrica, mecatrónica y química, así como administración y contabilidad. Que busca innovar y desarrollar tecnologías eficientes aplicadas a la movilidad eléctrica.

Con una fundación el día 6 de junio de 2017, el capitán M.I. Enrique Gabriel Munive Roldan y el asesor M.I. Mariano García del Gállego, realizan el primer reclutamiento [1].

El 14 de agosto del mismo año se realiza el primer boceto de Kalani, el primer vehículo diseñado por el equipo, el cual fue a competir en noviembre de 2017 en Shell Eco-Marathon sede Brasil, siendo el primer vehículo extranjero en esta competencia.

Shell Eco-Marathon contiene dos categorías en sus vehículos las cuales son: Prototipo y Urbano. Un vehículo prototipo es uno regularmente de 3 llantas, pequeño y liviano que se utiliza para la prueba de las nuevas tecnologías. Mientras que el vehículo urbano consta de todas las características de un vehículo que se puede encontrar en la calle, como lo son luces, espejos retrovisores, mecanismo limpia parabrisas y claxon. Así como la implementación obligatoria de las cuatro llantas y unas dimensiones más grandes a comparación del prototipo [2].

Kalani es un vehículo prototipo manufacturado en acero, carrocería de fibra de vidrio, con un peso de 95 kg, tres llantas de bicicleta rodada veinte, dirección de piñón cremallera, un motor en la llanta trasera tipo brushless y un banco de baterías de 48 [V] y 10 [A]. Sus dimensiones son 120 centímetros de ancho, 90 centímetros de alto y 220 centímetros de largo [3].



Figura 1.1: Kalani

El siguiente año, el equipo llevo a Kalani a Sonoma California en la misma competencia.

En 2019 se fabricó Mictlán, el primer vehículo urbano construido en la UNAM el cual compitió ese mismo año en Shell Eco-Marathon sede Sonoma, California.

Mictlán, con unas dimensiones de 240 centímetros de largo, 125 centímetros de ancho y 115 centímetros de alto, cuenta con una carrocería de fibra de vidrio, chasis de aluminio y baterías de litio, alcanza una velocidad de 90 kilómetros por hora, que se restringen por medio de electrónica y debido al reglamento de la competencia a 40 kilómetros por hora.



Figura 1.2: Mictlán

El vehículo pesa 150 kg, el banco de baterías da 48[V] y 10[A]. Cuenta con claxon, limpiaparabrisas, cajuela, espejos retrovisores, dirección de piñón cremallera, tracción trasera y luces como direccionales, freno y diurna en la defensa delantera como trasera [4].

Durante la emergencia sanitaria el equipo permaneció trabajando de manera remota sin embargo esto atraso los planes de manufacturar nuevos vehículos debido al confinamiento, en 2021 se participo en un cuestionario donde se respondieron preguntas relacionadas a la competencia.

En 2022 se empezó con el mantenimiento de los vehículos ya que al estar tanto tiempo guardados hubo varios desperfectos, motivo por el cual fue hasta abril de 2023 que Mictlán regresa a las competencias presenciales en Shell Eco-Marathon sede Indianapolis [5].

A finales de 2022 y principios de 2023, se inició el diseño y manufactura de Axolotl, un nuevo vehículo prototipo pensado para competir en la categoría de celda de combustión de hidrógeno. El vehículo tiene unas dimensiones de 120 centímetros de ancho, 110 centímetros de alto y 250 centímetros de largo, dirección tipo brazo de pivote. Durante todo 2023, el

vehículo se preparó para competir en el Shell Eco-marathon sede Brasil. Con un chasis de aluminio, carrocería de fibra de vidrio y un peso de 80 kg, el vehículo obtuvo el segundo lugar en la competencia, además del reconocimiento de ser el primer vehículo mexicano en dicha categoría [6].



Figura 1.3: Axolotl

En paralelo con la manufactura de Mictlán, se diseñó un vehículo tipo prototipo llamado Xipe esta manufacturado con un chasis de aluminio y una carrocería de fibra de vidrio. Este vehículo cuenta con unas dimensiones de 165 centímetros de largo, 90 centímetros de alto y 110 centímetros de ancho, una dirección de piñón cremallera, y por primera vez en la escudería, se implementó una suspensión tipo basculante. Este vehículo compitió en Shell Eco-marathon sede Brasil 2024 [7].



Figura 1.4: Xipe

1.1.2. Movilidad eléctrica

En los últimos años, la movilidad eléctrica se ha convertido en una de las opciones para disminuir los efectos de la contaminación atmosférica y del cambio climático. Con ello, se están buscando incentivos para cambiar los vehículos de combustión interna por eléctricos. Los vehículos eléctricos son todos aquellos que están impulsados por un motor eléctrico alimentado por una batería que se carga por medio de la red eléctrica. Al hablar de este tipo de vehículos, se suele pensar inmediatamente en los automóviles; sin embargo, también existen camiones, bicicletas, motocicletas y últimamente, patines tipo scooter.

El auto eléctrico se ha convertido en uno de los bienes más populares en los últimos años, vendiéndose con la idea de ser un coche con cero emisiones. Esto es cierto hasta cierto punto, ya que no se suelen considerar en estas declaraciones las emisiones que se generan durante la manufactura de dichos vehículos y la necesidad de cerrar el ciclo de vida del vehículo con el correcto desecho de las baterías. Además, la manera en que se recargan las baterías de estos vehículos a menudo proviene de centrales eléctricas que aún utilizan combustibles fósiles para

operar.

Esta tecnología, que es nueva en cuanto a su implementación pero no en su creación, implica costos mayores que los vehículos convencionales de combustión. Sin embargo, presenta ventajas a largo plazo, como la reducción de las emisiones de CO₂. Actualmente, existe una amplia variedad de vehículos eléctricos en el mundo; no obstante, pocos de estos se encuentran a la venta en México [1].

1.1.3. Conceptos

Visión artificial o visión por computadora

La visión artificial es una rama científica que se compone de un conjunto de métodos que posibilitan la adquisición, el procesamiento y el análisis de imágenes. El propósito central radica en capacitar a una computadora para que pueda obtener información valiosa y responder a cuestionamientos relacionados con el contenido de dichas imágenes. Para lograr este objetivo, se han empleado técnicas provenientes de diversas áreas, como la ingeniería y la informática, las cuales se caracterizan por su base matemática. Esto tradicionalmente limitaba su acceso a especialistas en el campo. Sin embargo, bibliotecas como OpenCV [8] han facilitado la tarea al ocultar la complejidad de los algoritmos en los que se fundamenta [9].

Esta disciplina ha permitido el diseño de soluciones automatizadas e inteligentes que permiten a los equipos capturar imágenes del mundo real, procesarlas y analizar la información, visto desde otra perspectiva la visión por computadora permite a las computadoras “ver” Lo cual ha hecho posible el diseño de maquinaria industrial, vehículos, sistemas de traducción, cámaras de seguridad, que son capaces de tomar decisiones inteligentes con base a su entorno [10].

- OpenCV es una biblioteca de uso libre especializada en el procesamiento y análisis de imágenes, originalmente fue desarrollada por Intel. Nació en enero de 1999 y hasta la fecha se sigue coronando como la biblioteca más popular de visión artificial, con ella se puede lograr: detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, etc. Su popularidad se debe a que es libre para propósitos comerciales y de investigación, es multiplataforma y existe para diversas arquitecturas

de hardware. Es muy utilizada por grupos de investigación, organismos gubernamentales y empresas desde el ámbito tecnológico como Google y Microsoft hasta el ámbito automotriz como Honda [9].

- Numpy [11] es una biblioteca de Python que da soporte para crear vectores y matrices multidimensionales, a su vez contiene las funciones matemáticas para operar con ellas.

Los métodos convencionales de la visión por computadora se centran en la aplicación de técnicas de segmentación y filtrado sin recurrir a redes neuronales. Estas técnicas implican dividir la imagen en regiones de interés que poseen las características que el usuario busca, como la detección de bordes y la implementación de filtros [12].

Imagen Digital

Una imagen digital es una representación bidimensional de una imagen a partir de una matriz numérica, frecuentemente en binario. Dependiendo de si la resolución de la imagen es estática o dinámica, puede tratarse de una imagen matricial ¹ o de un gráfico vectorial. Al ser representaciones, las imágenes contienen datos con los que se puede trabajar y obtener información [13].

La unidad mínima de trabajo en una imagen es el píxel que representa la intensidad de luz en cierto punto de la imagen. Poca luz nos da como resultado píxeles oscuros mientras que la presencia de mas luz nos entrega píxeles mas claros. La resolución viene dada por el número total de filas y columnas de píxeles, una resolución alta conlleva un mayor número de píxeles.

Las imágenes digitales se clasifican en imágenes binarias, en escala de grises y a color. Las binarias están representadas usando ceros y unos en donde el cero es el negro y el uno es el blanco. En escala de grises, la intensidad de la luz o nivel de gris es representado usando números enteros. La intensidad de la luz varia del negro al gris para finalmente terminar en el blanco. El número de los distintos niveles de gris en una imagen esta dado por una potencia de dos:

$$L = 2^B$$

En donde B es el numero de bits en la representación binaria de los niveles de brillo.

¹También llamada mapa de bits

- Cuando $B > 1$ hablamos de una imagen a escala de grises.
- Cuando $B = 1$ hablamos de una imagen binaria [14].

En cuanto a las imágenes a color existen diferentes espacios de color de los cuales se profundiza a continuación.

Espacios de color

Un modelo o espacio de color es la manera en la que un color está definido, nos permite analizar toda la información que esta presente dentro de una imagen [13].

Los modelos de color que fueron utilizados en el procesamiento digital de imágenes en este trabajo de grado son el RGB, HSV, HLS, YCrCb y Escala de Grises.

Físicamente cada píxel en una pantalla es representado por la cantidad de intensidad que se le aplique para cada color, y la pantalla completa se describe como una matriz, esta puede transformarse de varias maneras digitalmente, modificando sus valores para así obtener otros modelos de color. Matemáticamente a esto se le denomina como transformación de matrices.

El modelo RGB esta conformado por tres matrices que representan la intensidad de color en la imagen de los colores rojo, verde y azul que al combinarse puede generar una gama de colores bastante amplia.

$$RGB = \left[R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ r_{21} & r_{22} & r_{23} & \cdots & r_{2n} \\ r_{31} & r_{32} & r_{33} & \cdots & r_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & r_{m3} & \cdots & r_{mn} \end{bmatrix} \quad G = \begin{bmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{m1} & g_{m2} & g_{m3} & \cdots & g_{mn} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \cdots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \cdots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & b_{m3} & \cdots & b_{mn} \end{bmatrix} \right]$$

Figura 1.5: Estructura de una imagen de forma matricial en el espacio de color RGB

Este modelo y el modelo de escala de grises están embebidos dentro del cubo unitario de color.

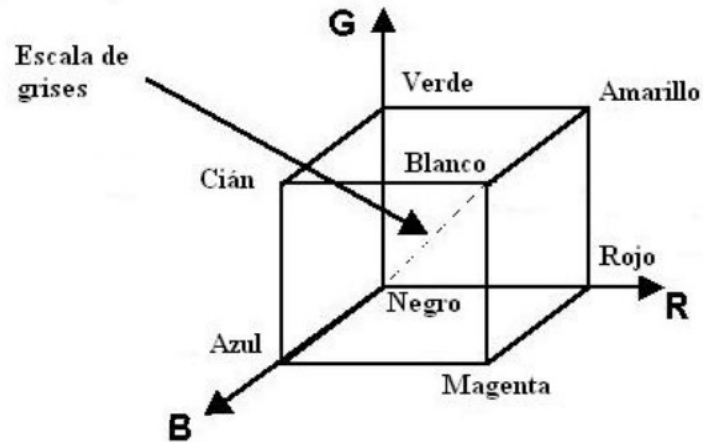


Figura 1.6: Cubo unitario de color

Los ejes están conformados por los colores rojo, verde y azul. El cian está situado en el vértice del cubo en donde el color azul y verde tienen su máximo valor de intensidad y el rojo es cero. Es por esto que los colores en este espacio de color se representan como coordenadas de valores (R,G,B) . El color negro está posicionado en el origen del sistema, mientras que el blanco está en el vértice opuesto. La escala de grises se encuentra en la diagonal que va del negro al blanco [13].

El modelo HSV es representado por tres componentes que son: El matiz (H), la saturación (S) y el brillo (V), de igual manera tiene una representación gráfica en donde estas tres magnitudes son representadas en un cilindro que contiene la información necesaria para encontrar el color. El matiz está representado por un ángulo que recorre la base del cilindro, mientras que la saturación y el brillo se encuentran en el eje horizontal y vertical respectivamente.

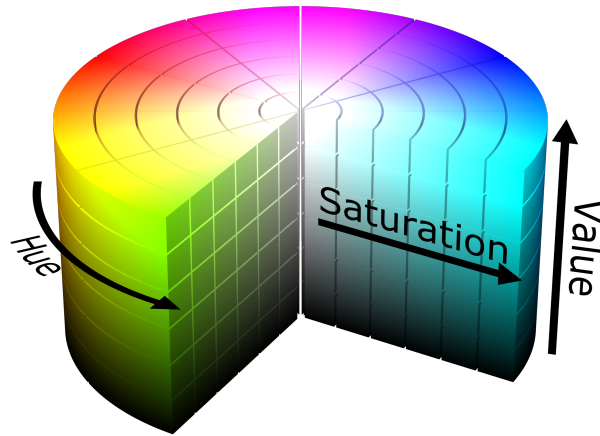


Figura 1.7: Interpretación gráfica HSV

El matiz es el valor del ángulo del matiz respecto al eje del cilindro, maneja valores de 0° a 360° donde cada ángulo representa un color diferente.

La saturación indica la escala de grises en el espacio de color, los valores varían entre el 0 % y 100 %, al aumentar este parámetro se obtiene un color mas vivo y al disminuirlo el color empezará a tender al blanco.

Finalmente el brillo es el grado de luminosidad en la imagen, los valores varían entre el 0 % y 100 %, por este lado los valores mas cercanos al 0 el color estará en una tonalidad mas oscura.

En OpenCV el valor máximo esta limitado a 255 debido a que los píxeles se trabajan en un formato llamado uint8 el cual utiliza 8 bits.

El modelo HLS es muy similar al HSV con la diferencia que la saturación en HSL va del color puro gris medio y en HSV del color puro al blanco, el tono en HSV va desde el negro al color y en HSL va desde el negro al blanco, en nuestro sistemas HSL se utiliza para detectar cambios en la saturación y luminiscencia (L) [15].

El modelo YCrCb se trata de una codificación no lineal del modelo RGB, el parámetro Y indica la luminiscencia, mientras que los dos restantes indican el tono del color, Cr nos muestra la ubicación entre el rojo y el verde mientras que Cb entre el azul y amarillo [16].

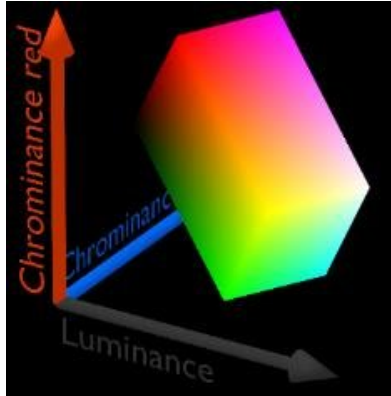


Figura 1.8: Interpretación gráfica del modelo YCrCb

1.1.4. Herramientas para la visión por computadora

Las imágenes digitales son procesadas para interpretar y entender su contenido, los pasos fundamentales para este proceso son:

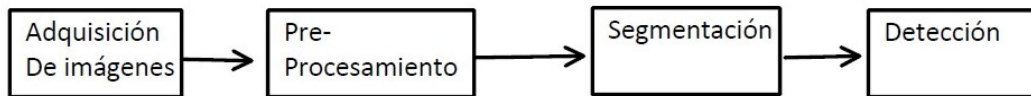


Figura 1.9: Pasos fundamentales en el procesamiento de imágenes digitales

- La adquisición de imagen es el proceso por el cual se capturan imágenes del mundo real ya sea en formato de imagen estática o de video para su futuro análisis.
- El pre-procesamiento se hace para mejorar la visibilidad de los puntos de interés y eliminar ruido en la imagen.
- La segmentación se utiliza para subdividir una imagen en sus regiones fundamentales para localizar objetos o bordes. Esto se hace de cierta manera en la que la división de la imagen sea significativa para la futura aplicación.
- La detección identifica la presencia de una característica u objeto detectado en la imagen [14].

Filtros

Los filtros son una etapa importante en el procesamiento de imágenes, son herramientas que toman una imagen de entrada y generan una imagen de salida, con la finalidad de destacar las áreas en las que se requerirá aplicar técnicas específicas de análisis como lo son el suavizado, eliminación de ruido y detección de bordes [9].

Los filtros lineales son aquellos que, utilizan un kernel para determinar el nivel de luminosidad de los píxeles en la imagen de destino, emplean una mezcla lineal de los valores correspondientes en la imagen original [9]. Como respuesta a este tipo de filtros, se tiene que la suma de dos señales es igual a la suma de las respuestas individuales. Es decir que cada píxel obtenido a la salida es una suma ponderada de una cierta cantidad de píxeles de entrada dependiendo del kernel [17].

Matemáticamente representan una convolución entre un kernel y la imagen.

El kernel en la visión artificial es una matriz que se coloca en la posición de cada píxel en la imagen para aplicar el filtro. Esta matriz se desplaza píxel por píxel para que en cada posición se pueda realizar la operación específica [9].

La convolución es un operador matemático que transforma dos funciones $f(x)$ y $g(x)$ en una tercera función que representa la magnitud en la que se superponen $f(x)$ y una versión trasladada e invertida de $g(x)$. Este operador se define como la integral del producto de ambas funciones después de desplazar una de ellas una distancia x [18]:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x - \alpha)d\alpha$$

La convolución nos proporciona un medio para trabajar con entradas de tamaño variable.

Un filtro no lineal es aquel cuyo resultado no se puede obtener mediante una operación de convolución lineal con un kernel fijo. En lugar de eso, los filtros no lineales utilizan una combinación no lineal de píxeles vecinos para obtener un resultado.

Los filtros lineales son matemáticamente más sencillos y eficientes de calcular en comparación con los filtros no lineales. Sin embargo, pueden no ser capaces de preservar bien los bordes y detalles finos en una imagen, ya que aplican la misma operación a todos los píxeles sin tener en cuenta su contexto o intensidad.

La diferencia clave entre un filtro lineal y un filtro no lineal radica en la naturaleza de su operación y en el rendimiento de cada uno [17].

- **Median:** Es un filtro no lineal que se utiliza para reducir el ruido en una imagen, su objetivo es eliminar los valores atípicos o ruidosos presentes en los píxeles de una imagen. El filtro toma una ventana de píxeles y coloca esta ventana sobre cada píxel de la imagen. Luego, ordena los valores de los píxeles dentro de la ventana y reemplaza el valor del píxel central por la mediana de los valores ordenados. Al utilizar la mediana en lugar del promedio, el filtro de mediana preserva mejor los bordes y los detalles de la imagen.
- **Bilateral:** Es un filtro no lineal utilizado para reducir el ruido en una imagen, toma en cuenta la similitud de intensidad entre los píxeles vecinos, los parámetros que utiliza es una ventana de píxeles y el segundo un parámetro de desviación estándar que controla la similitud entre píxeles. Al aplicar el filtro bilateral, se asegura de que los píxeles con intensidades similares tengan un mayor impacto en el resultado final, lo que ayuda a reducir el ruido mientras se conservan los bordes y detalles.
- **Gaussiano:** Es un filtro lineal utilizado para suavizar una imagen, la convolución utiliza un kernel que tiene una distribución de valores basada en la función gaussiana y al aplicarse, los píxeles en una imagen se mezclan con sus vecinos circundantes lo que resulta en el difuminado de la imagen, esto es muy útil para reducir el ruido de alta frecuencia.

[19], [17]

Detección de bordes y momentos

En la detección de carriles, se necesitaba obtener la detección de bordes, que como su nombre lo indica, busca los bordes de una imagen. Para esto, se utilizaron las siguientes técnicas en las diferentes pruebas realizadas:

- **Sobel:** es un operador de convolución utilizado para detectar bordes en una imagen. Se compone de dos kernels separados para detectar cambios en la intensidad horizontal

y vertical. Estos se aplican en la imagen para calcular aproximaciones de las derivadas parciales en ambas direcciones. Finalmente las aproximaciones se combinan para obtener la magnitud del gradiente que indica la presencia de bordes en la imagen [19].

- Canny: Es un detector de bordes que tiene como objetivo principal encontrar las regiones en una imagen donde hay cambios abruptos de la intensidad. lo que generalmente indica la presencia de un borde [20].
- findContours es una función utilizada por OpenCv que sirve para identificar y extraer los contornos cerrados que rodean áreas o figuras distintas presentes en la imagen binaria. Los contornos se representan como una lista de coordenadas que forman una curva cerrada [9].
- Momentos: son características numéricas que se utilizan para descubrir la forma, distribución y orientación de los objetos presentes en una imagen, estos se calculan a partir de la intensidad de los píxeles en una imagen. Existen diferentes tipos de momentos, y los más comunes son: Momentos de área (m_{00} , m_{01} , m_{10}) estos están relacionados con el área y el centroide de la región de interés, m_{00} representa el área, m_{01} y m_{10} se utilizan para calcular las coordenadas de centroide. Momentos centrales (m_{20} , m_{02} , m_{11}) se utilizan para describir la orientación y forma del área de interés con respecto a su centroide y Momentos de Hu que son utilizados para el reconocimiento de formas [21].

Transformación de perspectiva

Es una técnica que se utiliza para corregir la perspectiva de una imagen o para realizar ajustes de posición y orientación en la imagen.

La transformación de perspectiva se aplica cuando se desea convertir una imagen desde un sistema de coordenadas tridimensional a un sistema de coordenadas bidimensional. En situaciones en las que la cámara no está perfectamente alineada con la escena o cuando se toma una fotografía desde un ángulo no ortogonal, los objetos pueden aparecer distorsionados debido a la perspectiva. La transformación de perspectiva se utiliza para corregir estas distorsiones y obtener una vista rectificada de la escena.

Matemáticamente, la transformación de perspectiva se representa mediante una matriz de perspectiva. Esta matriz se calcula a partir de puntos de correspondencia entre la imagen

original y la imagen deseada. Los puntos de correspondencia son puntos clave en ambas imágenes que representan las mismas ubicaciones en el mundo real.

Una vez que se calcula la matriz homográfica, se puede aplicar para transformar cada píxel de la imagen original a su correspondiente posición en la imagen rectificadas. Esto permite corregir las distorsiones de perspectiva y alinear correctamente los objetos en la imagen final [22]².

Histograma

Es una representación gráfica de la distribución de valores en una imagen, muestra la frecuencia de aparición de diferentes niveles de la intensidad de los píxeles. El eje horizontal representa los niveles de intensidad posibles que van desde el 0 a 255, mientras que el eje vertical muestra la frecuencia de píxeles en la imagen. Mientras mas grande sea el número de píxeles que se encuentran a la derecha, la imagen será mas luminosa [9].

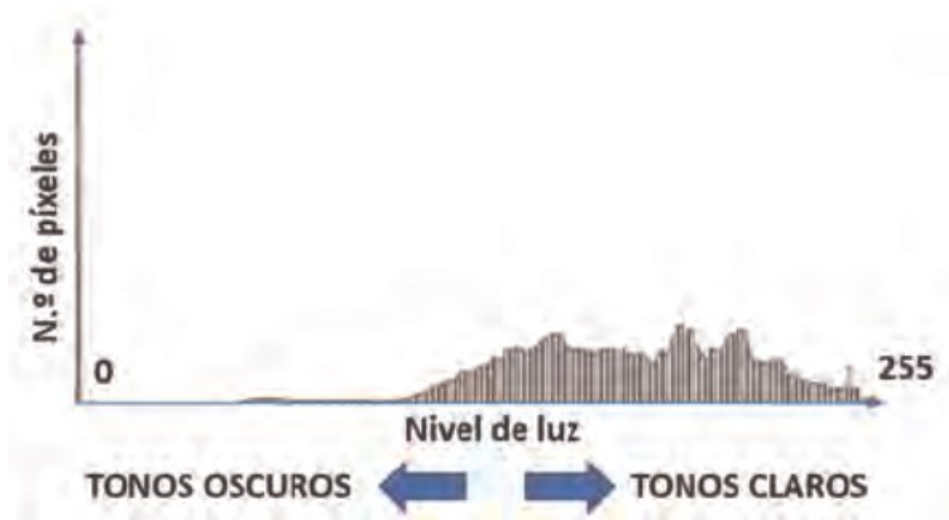


Figura 1.10: Representación gráfica de un Histograma

²Definición tomada de la clase Temas Selectos de Programación: Un enfoque al procesamiento de imágenes y video impartida en el semestre 2021-1 por el Mtro Gersain Barrón Velázquez en la Facultad de Ingeniería de la UNAM.

Visión Estéreo

Es una técnica de visión por computadora que utiliza dos cámaras para obtener información de profundidad de un punto a partir de dos imágenes tomadas desde diferentes puntos de vista [23]. El principio básico detrás de esta técnica se basa en la disparidad o diferencia de posición aparente de un objeto visto desde dos puntos de vista distintos, nuestros ojos pueden percibir esta disparidad y utilizan esta información para construir la imagen tridimensional que vemos, esta técnica básicamente imita este proceso utilizando cámaras. Los pasos a seguir en esta técnica son:

- **Adquisición de imágenes:** Se requiere al menos dos cámaras para capturar las imágenes. Estas cámaras deben estar colocadas en posiciones ligeramente diferentes por ejemplo, a la distancia aproximada entre nuestros ojos y deben estar calibradas para asegurar que las imágenes tengan la misma escala y se correspondan correctamente.
- **Rectificación de imágenes:** Antes de realizar la correspondencia estéreo, se pueden rectificar las imágenes para alinear las líneas epi-polares. Esto simplifica el proceso de correspondencia, ya que los puntos correspondientes en las imágenes se encuentran a lo largo de líneas horizontales.
- **Correspondencia estéreo:** El paso clave en la visión estéreo es encontrar los puntos correspondientes entre las dos imágenes. Esto implica buscar un punto en la primera imagen que coincida con el mismo punto en la segunda imagen. La disparidad entre estos puntos correspondientes se calcula como la diferencia de sus coordenadas en las dos imágenes.
- **Cálculo de profundidad:** Una vez que se obtiene la disparidad entre los puntos correspondientes, se puede utilizar la geometría estéreo para calcular la profundidad o distancia de los objetos en la escena. La disparidad está inversamente relacionada con la profundidad; es decir, cuanto mayor sea la disparidad, más cercano está el objeto.
- **Reconstrucción 3D:** Finalmente, se puede usar la información de profundidad calculada para generar un modelo tridimensional de la escena. La reconstrucción 3D puede expresarse en coordenadas de mundo real o coordenadas relativas a una de las cámaras.

StereoBM es una librería de OpenCV que se utiliza para realizar la correspondencia estéreo en imágenes, lo que significa que calcula la disparidad entre píxeles entre dos imágenes para crear un mapa de disparidad y poder obtener una percepción 3D [24].

Triangulación Estéreo

Esta técnica se basa en el concepto de paralaje y triangulación para estimar la distancia. Es un proceso similar a cómo el ojo humano percibe la profundidad. Se utiliza, por ejemplo, para medir la distancia entre cuerpos celestes. El nombre de esta técnica se debe a que se realiza una triangulación entre el objeto de interés y las cámaras, como se puede apreciar en la siguiente imagen:

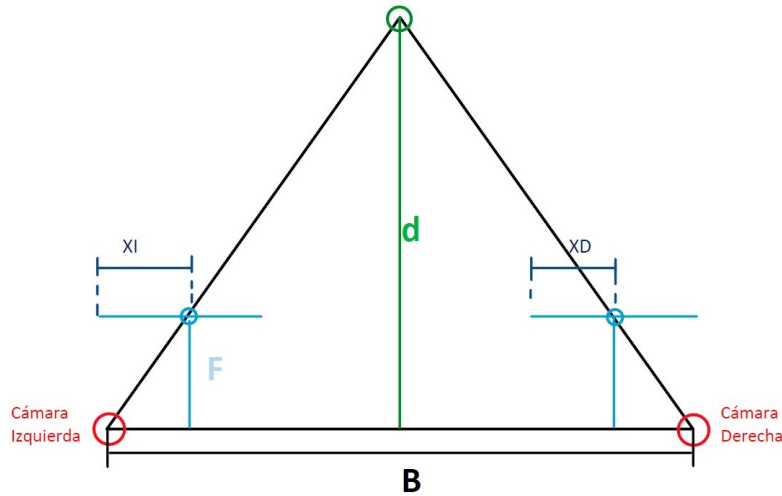


Figura 1.11: Triangulación Estéreo

Las cámaras capturan la imagen del objeto de interés separadas por una distancia B , y el objeto se registra en las imágenes con cierto desfase XI y XD . La distancia al objeto se determina mediante la siguiente ecuación:

$$d = \frac{B * F}{XI - XD}$$

Donde:

- F es la distancia focal de las cámaras.
- B es la distancia entre los centros de las cámaras.
- $XI - XD$ es la disparidad para el objeto de interés [25].

Ventana deslizante

Es una técnica que consiste en seleccionar un subconjunto de tamaño fijo, llamado "ventana", de un conjunto grande de datos y mover esta ventana gradualmente a lo largo del conjunto de datos. Este desplazamiento se realiza de elemento en elemento, y en cada paso se realiza alguna operación en los elementos dentro de la ventana. En el contexto de la visión por computadora, este enfoque se utiliza principalmente para la detección de objetos. Para este trabajo en específico, se emplea para la detección de carriles [26].

1.2. Estado del Arte

1.2.1. Los vehículos autónomos

Los vehículos autónomos son los que tienen la capacidad de controlarse por sí mismos, es decir sin la intervención humana. Poseen la tecnología que les permite reconocer su entorno y de dirigirse al destino que el conductor desee. En la conducción de estos vehículos participan distintos elementos, el más importante es el ordenador el cual le da la capacidad de respetar las normas de tránsito lo cual incluye respetar las señalizaciones, cediendo el paso o bien detenerse al detectar un peatón. Posee los mapas de las ubicaciones registradas y finalmente se encarga de controlar la velocidad y las vueltas que realiza el vehículo. Una pieza más es la capacidad para detectar vehículos u otros objetos del entorno, esto para medir la distancia que los separa de dicho obstáculo. Esto con la finalidad de poder actuar ante cualquier eventualidad. Dispone de un sistema GPS que indica con precisión la ubicación del vehículo así como la dirección en la que se desplaza. Cuenta con una unidad de medición inercial que determina la velocidad, radares auxiliares para la detección de objetos y de las líneas de carretera. Se espera que la generalización de los vehículos autónomos pueda hacer más eficiente el flujo del tránsito en las urbes más pobladas, así como reducir los accidentes automovilísticos [27].

La arquitectura de los vehículos autónomos tiene diferentes etapas que integran el funcionamiento eficaz del sistema, tales son: la adquisición de los datos, percepción del entorno, fusión de sensores, comunicación, planificación de rutas, control y actuación. Esto se logra definir por los distintos niveles de autonomía que se han establecido por la Sociedad de Ingenieros Automotrices (SAE International).

- Nivel 0: Una persona realiza la conducción completa del vehículo.
- Nivel 1: El vehículo cuenta con una función automatizada.
- Nivel 2: El vehículo controla su aceleración y dirección, el conductor es capaz de controlar el vehículo en cualquier momento.
- Nivel 3: El vehículo puede detectar su entorno y desarrollar la mayor parte de sus funciones, sin embargo, requiere de supervisión.
- Nivel 4: El vehículo es capaz de realizar una conducción autónoma bajo condiciones específicas.
- Nivel 5: El vehículo es capaz de desempeñar todas las funciones de manera autónoma sin importar las condiciones o el entorno.

Empresas como Tesla han logrado tener un nivel 3 de autonomía y los vehículos de Waymo han logrado realizar pruebas de nivel 4 [28].

Tesla actualmente cuenta con ocho cámaras y un potente sistema de procesamiento de 360°, con un alcance de 250 metros, así como una computadora que procesa toda la información recibida por las cámaras que ejecuta una red neuronal para proporcionar una vista en todas direcciones de manera simultánea. Según la página de Tesla "Tesla Vision, desarrollado sobre la base de una red neuronal profunda, deconstruye el entorno del auto con mayores niveles de confiabilidad que los que se pueden lograr con las técnicas clásicas de procesamiento de la visión"[29]

1.2.2. Los vehículos autónomos en México

En el año 2019, el equipo de electro-movilidad del Tecnológico de Monterrey conformado por alumnos y profesores de los campus Toluca, Puebla, Monterrey y CDMX realizaron una investigación y desarrollo de sistemas de generación de vehículos autónomos y eléctricos, este trabajo les permitió la semi automatización de un vehículo de prueba, así como el desarrollo de algoritmos y simuladores. Muchos de estos diseños se basan en la inteligencia artificial y en el Machine Learning. Se logró hacer trabajos de simuladores que ayudaban a probar y optimizar sus vehículos en circuitos cerrados, esto para poder testear los avances y después proceder a construirlos y probarlos de forma física [30].

En el año 2023, el grupo estudiantil VantTec, conformado por 25 alumnos y bajo la asesoría del profesor Hernán Castañeda, presentó en el campus Monterrey un vehículo tipo golf que avanza, con un volante que responde al software y recaba información del entorno. Este vehículo cuenta con diversos sensores que proporcionan datos sobre el entorno, la aceleración y la velocidad. Además, incorpora un sensor LiDAR que mapea el entorno del vehículo. Como característica adicional, el vehículo puede ser manejado mediante un control de videojuegos [31].

En el año 2024, en el campus Puebla, un grupo de profesores y estudiantes presentó unas plataformas eléctricas autónomas destinadas a transportar pasajeros. Con la colaboración de la Universidad de Wuppertal en Alemania, lograron desarrollar un vehículo capaz de recorrer distancias cortas de manera autónoma. Se planea que este vehículo siga una ruta fija entre los edificios dentro del campus [32].

1.3. Definición del problema

El problema consiste en diseñar un algoritmo que convierta un vehículo eléctrico en uno autónomo y pueda ser implementado en cualquier vehículo de la escudería Dzec. Así como determinar el alcance de la visión por computadora con métodos convencionales. Debido al tamaño del trabajo y a los recursos necesarios para el procesamiento de imágenes, es recomendable dividir el problema en secciones o subsistemas que posteriormente se probarán en conjunto. Dichos subsistemas son:

- Detección de los carriles de una carretera.
- Detección de objetos específicos.
- Estimación de distancia.

Como el principal objetivo de este trabajo es la programación, el problema mecánico y eléctrico/electrónico pasará a segundo plano, no obstante, será importante obtener información cercana a la realidad, por lo que se realizarán pruebas de campo con el objetivo de probar el algoritmo en un entorno real y con las implicaciones que eso conlleva. Para dichas pruebas es necesario utilizar los vehículos eléctricos con los que cuenta el equipo Dzec, así como la manufactura de algunas piezas para la adecuación del hardware necesario para el

correcto funcionamiento del software. El problema de programación es el tema principal de este trabajo y por ende es el que se realizará con más detalle, pero conforme avance el trabajo se irá especificando con más detalle los problemas que se presentaron con la parte mecánica y eléctrica/electrónica.

1.4. Objetivo general

Desarrollar un software experimental que permita la conversión de los vehículos eléctricos de la escudería Dzec en vehículos autónomos y evaluar las capacidades de la visión por computadora en aplicaciones autónomas.

1.4.1. Objetivos específicos

- Lograr que el software detecte el camino en una carretera.
- Que el software detecte círculos de color azul y estime la distancia que hay entre la cámara y el obstáculo.
- Probar diversos métodos convencionales de visión por computadora para determinar los alcances de la disciplina.
- Que el sistema autónomo pueda desmontarse y montarse en diversos vehículos eléctricos.

1.5. Alcances

Diseñar el algoritmo que permita la detección precisa del camino, además de estimar la distancia a obstáculos específicos. Este algoritmo debe poder adaptarse a diversos vehículos eléctricos sin mayor problema, garantizando su utilidad en diferentes vehículos y condiciones.

1.5.1. Diseño e implementación de un sistema de detección de carriles y obstáculos

- Diseñar un algoritmo de visión por computadora que permitan al vehículo detectar carriles, en diferentes condiciones. Para representar los resultados visualmente, el carril derecho se resaltará en color rojo y el izquierdo en color azul. Esta diferenciación de colores permite identificar y verificar la correcta detección de los carriles.
- Incorporar un sistema de detección de obstáculos basado en el color.

1.5.2. Estimación de distancias en tiempo real

- Diseñar e implementar un algoritmo para estimar la distancia entre el vehículo y los obstáculos utilizando cámaras presentes en el día a día, utilizando diferentes técnicas con el objetivo de encontrar la que presente menor error de estimación.

1.5.3. Adaptación de algoritmos para ambientes reales

- Que el algoritmo sea capaz de adaptarse a cambios de iluminación, curvas pronunciadas y carriles estrechos.
- Validar que el sistema pueda adaptarse a entornos reales con distintas configuraciones de cámaras.

1.5.4. Desarrollo de hardware

- Diseñar y fabricar soportes que permitan que la cámara permanezca estable en entornos reales, así como una canastilla en donde la electrónica se mantenga segura durante la conducción autónoma.

1.5.5. Mejoras a futuro

- Proponer dependiendo de los resultados, las mejoras en la detección, así como incorporar sensores adicionales para un sistema de percepción robusto y adaptable.

1.6. Requerimientos

- Uso de Python y OpenCV para implementar algoritmos de visión y procesamiento de imágenes sin interfaz gráfica con la intención de migrar a un sistema computacional monoplaca.
- Cámaras de alta resolución con especificaciones actuales.
- Soportes de cámara y una canastilla para la electrónica, en materiales adecuados para soportar el peso y el uso en pruebas de campo.
- Realización de pruebas en entornos de iluminación controlada y en campo abierto con cambios de luz para evaluar la respuesta del sistema.
- Uso de calibraciones específicas para ajustar el ancho del carril, estimaciones de distancia, y tolerancias a variaciones en la iluminación y la geometría del camino.

1.7. Justificación

Los vehículos eléctricos son la transición hacia donde se dirige la industria automotriz, sin embargo, una nueva tendencia que ha empezado a surgir ha sido que los vehículos sean cada vez más autosuficiencia en cuanto a la hora de manejarlos. Se empezó con la creación e implementación de transmisiones automáticas, pasando por la asistencia de software a la hora de manejar o estacionarse y en la actualidad se ha empezado a innovar con la creación de vehículos que sean capaces de operarse a sí mismos. Esta automatización promete traer múltiples beneficios a la sociedad, así como facilitarle un poco más la vida al ser humano. La motivación es que actualmente no hay muchos desarrollos de este tipo en México, por lo que es importante el realizar desarrollos de tecnología, no solo por la necesidad actual de la movilidad sino también por el desarrollo, investigación e implementación de tecnología, así como el conocimiento que se adquiere a la hora de llevar a cabo este tipo de proyectos que a la hora de realizar la sinergia con la teoría complementan de manera enorme nuestra formación como profesionistas.

Capítulo 2

Metodología

El proyecto se desarrolló con un enfoque iterativo, mediante pruebas y ajustes sucesivos en cada etapa, sustentado en una metodología experimental de prueba y error que permitió identificar, adaptar y refinar los componentes de software y hardware conforme avanzaba el desarrollo. Esta metodología incluyó una fase de análisis detallado de resultados y propuestas de mejora para el futuro, con el objetivo de evolucionar el proyecto hacia aplicaciones de autonomía mas robustas.

2.1. Planeación y selección de herramientas

El objetivo del proyecto se centra en desarrollar un sistema de visión por computadora para la detección de carriles y obstáculos en un vehículo autónomo. Para lograrlo, se seleccionaron cuidadosamente las herramientas de programación y visión por computadora, optando por Python y la biblioteca OpenCV. Estas herramientas ofrecen flexibilidad en el desarrollo y pruebas del sistema, y además minimizan la necesidad de interfaces complejas, permitiendo un enfoque más ágil y adaptativo en la implementación del proyecto así como la futura migración del sistema a una monoplaca.

2.2. Desarrollo del sistema en componentes básicos

- Para cada subsistema básico, se llevaron a cabo desarrollos y pruebas individuales de funcionalidad, enfocándose en tareas como la apertura de imágenes, la detección de carriles y bordes, la identificación de colores y el cálculo de distancias. Este enfoque permitió evaluar y ajustar el desempeño de cada componente por separado antes de integrarlos en el sistema completo.
- Se implementaron y probaron diversos algoritmos para cada subsistema, iterando en técnicas de procesamiento de imagen hasta alcanzar los resultados esperados. El trabajo incluyó desde métodos básicos, como la binarización de imágenes, hasta procesos avanzados, como la detección de contornos y la identificación de colores en espacios de color específicos, como HSV. Esta iteración continua permitió optimizar cada subsistema, adaptándolo a los requisitos de detección y precisión necesarios para el proyecto.

2.3. Integración y pruebas conjuntas de subsistemas

- Tras validar los subsistemas de manera individual, se realizó su integración en un sistema unificado que permite el procesamiento simultáneo de detección de carriles, identificación de color y cálculo de distancia. Esta fusión de algoritmos facilitó la coordinación de los distintos módulos, logrando un sistema autónomo capaz de analizar múltiples parámetros en tiempo real para mejorar la precisión y eficacia en entornos de prueba.
- Las primeras pruebas de campo se centraron en evaluar el rendimiento del sistema en entornos controlados, donde se identificaron áreas de mejora en los algoritmos de detección y ajustes necesarios en las tolerancias para la estimación de distancias y la identificación de color bajo condiciones de iluminación variable. Estos ensayos iniciales permitieron realizar las primeras calibraciones y optimizar la respuesta del sistema ante diferentes escenarios.

2.4. Desarrollo de prototipos de hardware y soportes

En la fase de diseño y fabricación de soportes de cámara, se crearon piezas específicas para las cámaras y la canastilla de la electrónica. Estos soportes fueron ajustados iterativamente según los resultados obtenidos en las pruebas de campo, con el fin de optimizar su funcionalidad y estabilidad. Durante la evaluación en condiciones reales, se realizaron modificaciones adicionales para mejorar la estabilidad del sistema en pruebas dinámicas, así como para evaluar la resistencia de los materiales frente al uso continuo y condiciones de manejo en movimiento.

2.5. Calibración del sistema y ajuste de parámetros

- Durante la calibración de cámaras, se llevaron a cabo pruebas específicas para ajustar los parámetros de las mismas, con el objetivo de asegurar una estimación precisa de la distancia. Estos ajustes permitieron optimizar el sistema de visión, reduciendo errores en la detección y medición de distancias, aspectos esenciales para el funcionamiento autónomo en tiempo real.
- Se realizaron ajustes en el algoritmo de detección de carriles mediante la calibración de los parámetros en función del ancho real de los carriles en el campo de pruebas y bajo condiciones de iluminación variable. Estos ajustes permitieron mejorar la precisión en la detección, asegurando que el sistema pudiera identificar correctamente los carriles incluso en escenarios de luz cambiantes.
- Se optimizaron los rangos de color y se implementaron filtros para mejorar la precisión en la detección de objetos y bordes. Los ajustes incluyeron la modificación de umbrales en el espacio de color HSV, lo cual permitió una identificación más exacta en diferentes condiciones de iluminación y contribuyó a la robustez del sistema en el reconocimiento de elementos en el entorno.

2.6. Pruebas finales y validación del sistema

- Durante esta fase, el sistema fue evaluado en entornos con condiciones cercanas a las del mundo real, enfrentando variaciones de luz, curvas pronunciadas y carriles estrechos. Estas pruebas permitieron identificar limitaciones específicas en la detección de carriles y en la precisión de la estimación de distancias, evidenciando áreas de mejora para el desempeño en escenarios más complejos y desafiantes.
- En esta etapa, se registraron minuciosamente los resultados de cada prueba, cuantificando y documentando los errores observados. Este análisis detallado sirvió para fundamentar propuestas de mejora y para validar el desempeño alcanzado, estableciendo así una base sólida para futuras optimizaciones y desarrollos del sistema.

2.7. Propuesta de mejoras y trabajo a futuro

Con base en los resultados obtenidos en las pruebas finales, se identificaron áreas clave para mejorar el rendimiento del sistema, destacando la necesidad de adaptar el algoritmo a condiciones variables de iluminación y considerar la integración de sensores adicionales. Estas mejoras permitirán optimizar la precisión y robustez del sistema en futuros desarrollos, ampliando las capacidades del sistema para enfrentar condiciones y entornos más complejos.

Capítulo 3

Desarrollo

En este capítulo, se describe todo el proceso llevado a cabo para la realización del proyecto. En las próximas secciones se abordarán de manera cronológica las pruebas realizadas para la resolución del problema, mostrando la secuencia en la que el software fue evolucionando para obtener los mejores resultados posibles utilizando métodos convencionales de visión por computadora.

Se tomó como base los vehículos eléctricos de la escudería Dzec debido a la versatilidad de sus prototipos, que, gracias a su tamaño y peso, permiten una manipulación sencilla por una o dos personas. No obstante, al concluir el proyecto, el software desarrollado podrá adaptarse y aplicarse a otros tipos de vehículos o prototipos.

El primer paso consistió en determinar el lenguaje de programación más adecuado para abordar el problema. Previamente, durante la clase de Temas Selectos de Programación I, se habían desarrollado programas de visión por computadora utilizando EmguCV y Visual Studio con C#. Este software se orienta hacia la programación orientada a objetos y requiere la creación de una interfaz gráfica para la interacción con el usuario, lo que lo hace interactivo hasta cierto punto. Sin embargo, la implementación del código en este entorno suele ser extensa.

En la materia de Diseño Mecatrónico, se volvió a trabajar con visión por computadora, pero esta vez utilizando el lenguaje Python con OpenCV. En Python, no es necesario desarrollar una interfaz gráfica y el código resulta más simplificado.

Dado que en el proyecto no se requería una interfaz gráfica, se descarto el uso de C#. Se decantó por utilizar Python debido a que la implementación del código es menos extensa, pero principalmente por querer utilizar software de uso libre que pueda ser migrado a alguna tarjeta de desarrollo y ahorrar espacio a la hora de migrar el sistema.

El siguiente paso fue la instalación del lenguaje de programación y las librerías necesarias mediante el uso del administrador de paquetes pip.

Luego, se procedió a la creación de algoritmos destinados a acercarse al cumplimiento de los objetivos ya establecidos.

3.1. Apertura de imagen y vídeo

Con ayuda de Open CV se puede generar un algoritmo sencillo capaz de abrir imágenes. Un video es un proceso en el que gracias a la adición de un bucle while la imagen se actualiza en tiempo real. Abriendo imagen por imagen hasta que dicho bucle se interrumpe.

Estas pruebas se realizaron utilizando la cámara web integrada en la computadora, ya que el objetivo era únicamente familiarizarse con el código básico de OpenCV, sin necesidad de emplear cámaras de mayor calidad. Si bien el uso de una cámara de mayor resolución podría ofrecer una mejora en la calidad de imagen, este factor depende del tipo de cámara seleccionada y no era crítico para los fines exploratorios de estas pruebas.

3.2. Prueba 1: Detección de carril

Utilizando el algoritmo antes mencionado para la apertura de una imagen obtenida de la web, se generó un nuevo algoritmo para la detección de caminos, este algoritmo comienza con la apertura de la imagen dentro del entorno. A continuación, se procede a la conversión de esta imagen a un espacio de color en escala de grises. Posteriormente, se aplica un filtrado bilateral para refinar aún más la imagen. Después de este paso, se lleva a cabo un proceso de binarización para segmentar la imagen. Finalmente, se realiza la detección de bordes canny en la imagen procesada.

3.3. Prueba 2: Detección del centro de carril en video

El algoritmo utilizado para la apertura de un video obtenido de la web y la detección de carriles se ha condensado en una nueva versión al incorporar la búsqueda del centro del carril. Esta nueva parte del algoritmo consiste en identificar los contornos más grandes presentes en la imagen. Se espera que este contorno más grande corresponda al espacio inferior de la imagen que representa el carril. Luego, se calcula el centro del carril con la ayuda de los momentos de área. Para visualizar tanto los bordes como el centro del carril en la imagen procesada, se hace uso de las herramientas proporcionadas por OpenCV, como line (líneas) y circle (círculos), que permiten dibujar estos elementos. Los videos utilizados para esta prueba son pregrabados obtenidos de la web.

3.4. Pruebas de detección de color

El algoritmo utilizado para la detección de color necesita que se le especifique un rango de colores en el espacio de color HSV.¹ Luego, se crea una máscara binaria que resalta los objetos que poseen el color deseado. Esta máscara se utiliza posteriormente para segmentar la imagen y llevar a cabo una operación específica, en este caso, se emplea para identificar los contornos del objeto que posee dicho color. Finalmente, se dibuja el contorno del objeto detectado en la imagen procesada.

Para esta prueba se utilizó la cámara web integrada en la computadora.

3.5. Primer prueba de distancia

Este algoritmo incorpora la función de detección de color y agrega información del entorno, como el ancho del objeto en milímetros y la distancia focal de la cámara a utilizar. Además, se incluye el procedimiento previamente descrito en este capítulo para encontrar el centro de un objeto. Con todos estos datos y la obtención del contorno del objeto, se aplica la siguiente fórmula para estimar la distancia.

$$distancia = \frac{tamañoObjeto * distanciaFocal}{contornoObjeto}$$

¹Este rango de color es único para cada color

En esta prueba se utilizaron la configuración de cámaras de un Samsung A52s y un Sony Xperia XZ1.

3.6. Primer prueba completa de software

En este algoritmo, se integran todos los subsistemas mencionados anteriormente, trabajando de manera conjunta. Para empezar a llevar un registro de las pruebas, se implementó la función de guardar los videos con el procesamiento de imágenes en formato mp4, así como la captura de los resultados obtenidos en la consola, que incluyen las distancias estimadas y la dirección estimada en caso de ser necesario dar una vuelta.

En esta prueba de software se reutilizaron los videos pregrabados de las pruebas anteriores.

3.7. Primer prueba de campo

El algoritmo utilizado para esta prueba es el mismo mencionado anteriormente en la sección "Primer prueba completa de software". La prueba se realizó en el laboratorio de manufactura ubicado en el edificio O de la facultad de ingeniería y los resultados de esta prueba se presentan en el siguiente capítulo.

Se repitió la configuración de cámaras antes descrita en la sección anterior.

3.8. Segunda Prueba de software

Este algoritmo esta dividido en diferentes funciones, cada una de ellas realiza una tarea en específico:

- La función `combinacion`, realiza una combinación de canales de color para resaltar ciertos aspectos de la imagen, fusionando la información de crominancia con la imagen en escala de grises.
- La función `calcula_puntos_destino`, toma las dimensiones de la imagen de entrada y los

parámetros de re-dimensionamiento y aspecto, y calcula las coordenadas de los puntos de destino para la transformación de perspectiva. Estos puntos de destino se utilizan en la función `birdseye` para obtener una vista de pájaro de la imagen.

- La función `birdseye`, utiliza la función `cv2.getPerspectiveTransform` para obtener la matriz de transformación de perspectiva a partir de los puntos de origen (`src_points`) y destino (`dst_points`). Luego, se utiliza la función `cv2.warpPerspective` para aplicar la transformación de perspectiva a la imagen original (`imagen`) utilizando la matriz de transformación `M`. La imagen transformada resultante se devuelve como resultado de la función.
- La función `ROI`, toma una imagen y una lista de puntos que representan un polígono, y luego dibuja ese polígono en la imagen utilizando la biblioteca OpenCV. Resaltará las regiones de interés ² en el video mediante la superposición de un polígono.
- La función `sobel`, realiza el procesamiento de la imagen utilizando el operador Sobel para resaltar los bordes en los canales `S` y `L` de la imagen `HLS`. Luego, se aplican umbrales para binarizar los resultados y finalmente se combina las binarizaciones de ambos canales en una sola matriz binaria.
- La función `líneas_carril`, realiza el proceso de encontrar los carriles en la imagen utilizando ventanas deslizantes y ajustando polinomios de segundo grado a los puntos encontrados. Luego, se crean imágenes binarias para cada carril y se ajustan los colores de los carriles encontrados antes de combinarlos en una sola imagen. Se puede agregar un paso adicional para llenar el espacio entre los carriles con un área verde. Por último, los carriles encontrados y el área verde se combinan en una sola imagen.
- La función `curvatura`, calcula el radio de curvatura de la carretera en metros a partir de los coeficientes de los polinomios ajustados para los carriles izquierdo y derecho.
- La función `posicion_coche_con_respecto_al_centro`, calcula la posición del coche con respecto al centro de los carriles en centímetros utilizando los polinomios ajustados para los carriles izquierdo y derecho y la información sobre las dimensiones de la imagen.

Este código está diseñado para detectar carriles y proporcionar información sobre la curvatura de la carretera y la posición del vehículo con respecto a los carriles.

²ROI, por sus siglas en inglés "Region Of Interest"

En este punto del proceso de programación, se optó por dividir los programas en dos partes distintas. Por un lado, se enfocó en la programación de las funciones necesarias para cumplir con el objetivo del proyecto y por el otro, se dedicó a la aplicación de estas funciones en el programa principal. Esta decisión se tomó con el propósito de mantener un mayor orden y facilitar la visualización de los programas.

En esta prueba de software nuevamente se utilizaron los videos pregrabados de las pruebas anteriores.

3.9. Pruebas de detección de distancia

Como indica el nombre de la sección, se llevaron a cabo específicamente dos pruebas utilizando dos enfoques diferentes. El primero se basó en el uso de la Visión Estéreo, mientras que el segundo se centró en la triangulación estéreo.

En cuanto a la Visión Estéreo, en primer lugar se desarrolló un obturador dual que permitía la visualización en tiempo real de las imágenes de dos cámaras y la capacidad de guardar las imágenes en directorios específicos al presionar la tecla 's'. El programa podía detenerse al presionar la tecla 'Esc', lo que liberaba los recursos de las cámaras y cerraba las ventanas de visualización.

El siguiente programa se encargó de llevar a cabo la calibración de las cámaras utilizando la librería OpenCV. Este proceso incluyó la calibración estéreo y la rectificación estéreo de un sistema de dos cámaras mediante un tablero de ajedrez. Además, calculaba y mostraba la distancia focal en milímetros para ambas cámaras, y guardaba los mapas de rectificación en un archivo XML.

Finalmente, se implementó el último programa, que se encargaba de realizar la rectificación estéreo de las imágenes provenientes de dos cámaras, calcular el mapa de disparidad y estimar la profundidad en tiempo real mediante la triangulación estéreo. Luego, mostraba las imágenes rectificadas y la profundidad estimada en ventanas separadas.

Con respecto a la triangulación estéreo y continuando con el nuevo proceso de programación el algoritmo se dividió en las siguientes funciones:

- La función `Filtro_HSV` aplica un filtro de color HSV para detectar objetos en un rango

de colores en específico, como resultado tenemos una máscara binaria que resalta los píxeles que caen en el rango especificado.

- *Círculos* es una función que con ayuda de la máscara binaria de la función anterior encuentra la figura circular mas grande de la imagen. Primero, encuentra los contornos en la máscara binaria y selecciona el contorno mas grande asumiendo que es el círculo deseado, luego calcula el centro y el radio del círculo y devuelve las coordenadas del centro.
- La función *Profundidad*, esta función calcula la distancia en centímetros entre el círculo detectado utilizando la triangulación estéreo, toma las coordenadas del círculo, la distancia entre las cámaras y especificaciones de la cámara como distancia focal y ángulo de visión para calcular la distancia horizontal entre las coordenadas x del mismo círculo captado en la cámara derecha e izquierda. para finalmente realizar una corrección en la profundidad para una estimación mas precisa. Esta corrección se logra tras realizar pruebas midiendo la distancia real contra la distancia que marcaba el programa para posteriormente realizar una regresión logarítmica.

La regresión logarítmica se realizo con ayuda del programa Excel, el procedimiento es el siguiente:

- Transforma los datos originales de x convirtiéndolos a $\ln(x)$ Por lo tanto, las tuplas quedan como $(\ln(x), y)$.
- Realiza una regresión lineal utilizando $\ln(x)$ como la variable independiente, y como la variable dependiente.
- La ecuación de la línea recta con estos datos queda como

$$y = a \ln(x) + b$$

Donde a es la pendiente y b la intersección.

Excel calcula los valores de a y b para la mejor línea de ajuste de los datos transformados, dando como resultado una línea de tendencia logarítmica.

Este código esta diseñado para procesar videos en tiempo real capturados por dos cámaras, realizar un seguimiento de objetos de interés, como círculos, y calcular su profundidad tridimensional mediante la técnica de triangulación estéreo.

Se repitió la configuración de cámaras utilizada en las secciones pasadas.

3.10. Segunda Prueba de campo

Para esta prueba, se realizó una evaluación de las cámaras, las cuales incluyeron la cámara principal de un Samsung A52s, un Xperia XZ y cámaras web Full HD. Estas cámaras se seleccionaron por la facilidad con la que hoy en día se cuenta con un teléfono y una cámara con especificaciones actuales, se probaron utilizando el algoritmo de estimación de distancia con la técnica de triangulación estéreo, bajo condiciones de luz solar favorable a las 3 pm, con el fin de determinar el desempeño en la detección de color y la estimación de distancia.

Del mismo modo, con las mismas cámaras, se evaluó el desempeño del algoritmo de detección de caminos en las mismas condiciones. Para este algoritmo se requiere realizar una calibración mediante un programa que también fue desarrollado para este trabajo. Este programa activa la cámara que se utilizará para la detección de caminos y permite seleccionar manualmente el área de interés donde se desea ubicar el camino. El objetivo de este proceso es proporcionar al programa el ancho del carril que se detectará posteriormente. La detección de puntos debe llevarse a cabo de la siguiente manera:

Se debe seleccionar como primer punto la esquina superior izquierda, seguido por la esquina superior derecha como segundo punto, la esquina inferior izquierda como tercer punto, y finalmente la esquina inferior derecha como cuarto punto. Una vez obtenidos estos puntos, el programa devuelve las coordenadas correspondientes para posteriormente iniciar el proceso de detección de carriles.

Esta prueba se llevó a cabo para determinar los algoritmos y técnicas más adecuados para la realización de una prueba final.

3.11. Prueba final de detección de distancia

Para esta prueba se utilizó un nuevo enfoque el cual implica aprovechar la percepción a favor del sistema, con la finalidad de disminuir el error a la hora de la estimación de distancia. El obstáculo se colocó a una distancia medida con un flexómetro, y el programa calculó el área en píxeles cuadrados. Posteriormente, se movió el objeto a diferentes distancias de la

cámara para obtener sus respectivas áreas en píxeles cuadrados. Con los datos recopilados, se generó una gráfica que mostraba una forma similar a una función de potencia. Se aplicó una regresión potencial a estos datos para obtener la función que relaciona el área del objeto con la distancia a la que se encuentra. Esta función se utilizó para realizar una nueva estimación de distancia, cuyos resultados se detallarán más adelante.

La regresión potencial se realizó nuevamente con ayuda de Excel:

- Toma los datos de las tuplas (x,y) y aplica el logaritmo a ambos elementos de las tuplas, transformándolos a $(\log(x),\log(y))$.
- Con los datos transformados, Excel realiza una regresión lineal para encontrar la línea recta que se ajusta mejor a estos puntos transformados.
- Esta ecuación lineal quedaría como: $\log(y)=\log(a)+b\log(x)$. Donde $\log(a)$ es la intersección y b es la pendiente.
- Una vez realizada la regresión lineal, Excel convierte los resultados a la forma de la ecuación potencial $y = ax^b$.
- La constante a se obtiene como: $10^{\log(a)}$.

Para esta técnica solo se requiere el uso de una cámara para funcionar adecuadamente.

3.12. Diseño de piezas para el hardware

Se realizaron dos diseños para el soporte de las cámaras. El primero fue un soporte que pudiera acomodar la cámara en cualquiera de los vehículos, con la finalidad de que la cámara pudiera ver la parte posterior del vehículo y que el lente estuviera en posición horizontal, proporcionando así una vista similar a la del conductor.

El primer diseño incluyó una esfera, aprovechando los tres grados de libertad que brinda esta figura. El segundo diseño, nuevamente con tres grados de libertad para facilitar el acomodo de la cámara, se inspiró en un nudo universal. Ambos diseños fueron impresos en 3D.

El segundo diseño consistió en una canastilla destinada a contener una computadora de 14 pulgadas con el software, diseñada para ser removible, así como poder transportar la computadora de manera segura en los vehículos prototipo. Esta canastilla también está diseñada para llevar las baterías del vehículo, así como la electrónica necesaria para su funcionamiento.

3.13. Prueba Final

En esta prueba se evaluará el desempeño en conjunto de los algoritmos de estimación de distancia con el método de la percepción y el algoritmo de detección de carriles con ventanas deslizantes, utilizando la cámara web full HD que ya se ha descrito con anterioridad y las cámaras de un teléfono Samsung A52s, respectivamente. Bajo condiciones de luz solar al medio día con un clima nublado. El objetivo es conjuntar un algoritmo completo que pueda ser considerado para liderar el sistema de un vehículo autónomo. Esta prueba también se realiza con el vehículo en movimiento para evaluar los datos obtenidos en un entorno real.

El algoritmo que se utilizó se dividió en las siguientes funciones, algunas de ellas ya habían sido utilizadas anteriormente, por lo que su descripción se omitirá, como: `Combinacion`, `calcula_puntos_destino`, `birdseye`, `ROI`, `sobel`, `lineas_carril`, esto en cuanto a la detección del carril.

La nueva función es `puntos_origen` que da formato a los cuatro puntos de origen obtenidos por la calibración, con el fin de realizar la transformación de perspectiva en la función `birdseye`.

En cuanto a las funciones de estimación de distancia, se utilizaron `Filtro_HSV` y `Círculos`. La nueva función es `calcular_distancia` que calcula la distancia basándose en el área de un objeto detectado, ajustando la relación entre el área del objeto y la distancia a la cámara utilizando la siguiente fórmula:

$$distancia = b \left(\frac{area}{a0} \right)^n$$

donde:

- b es el coeficiente de calibración.

- area es el área actual del objeto detectado.
- a0 es una constante de referencia empleada para la calibración.
- n es la potencia a la que se eleva el área.

Esta fórmula se obtiene mediante la calibración, siendo, como se puede observar, una función potencial de la forma: $y = ax^b$.

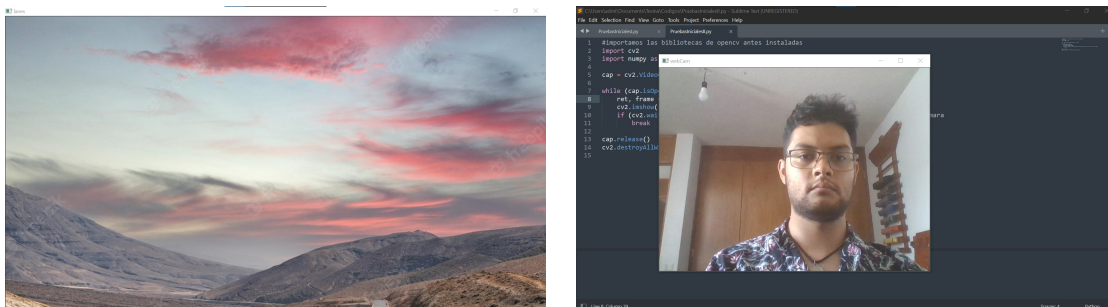
Capítulo 4

Resultados

En este capítulo, se describen los resultados obtenidos durante todo el proceso llevado a cabo para la realización del proyecto. En las próximas secciones se abordarán de manera cronológica los resultados obtenidos durante las pruebas realizadas para la resolución del problema, destacando como la metodología iterativa permitió alcanzar los objetivos planteados.

4.1. Apertura de imagen y video

En la figura 4.1 a, se observa una imagen de un paisaje abierta en python y OpenCV. En la figura 4.1 b, se observa como es que se abrió la cámara de la computadora en modo de video en tiempo real.



(a)

(b)

Figura 4.1: (a) Apertura de imagen (b) Apertura de vídeo

Estas dos pruebas sencillas se realizaron para familiarizarse con el entorno de Open CV y python. Estos resultados son el preámbulo para la realización de tareas mas complejas en el entorno y con las librerías.

4.2. Prueba 1: Detección de carril

En la figura 4.2 se observa la imagen original de una carretera junto con su procesamiento de imagen. En este procesamiento se pueden apreciar claramente los bordes blancos de la carretera, así como las líneas amarillas que marcan la división de carriles. Además, se detecta un espectro de luz que no nos interesa.

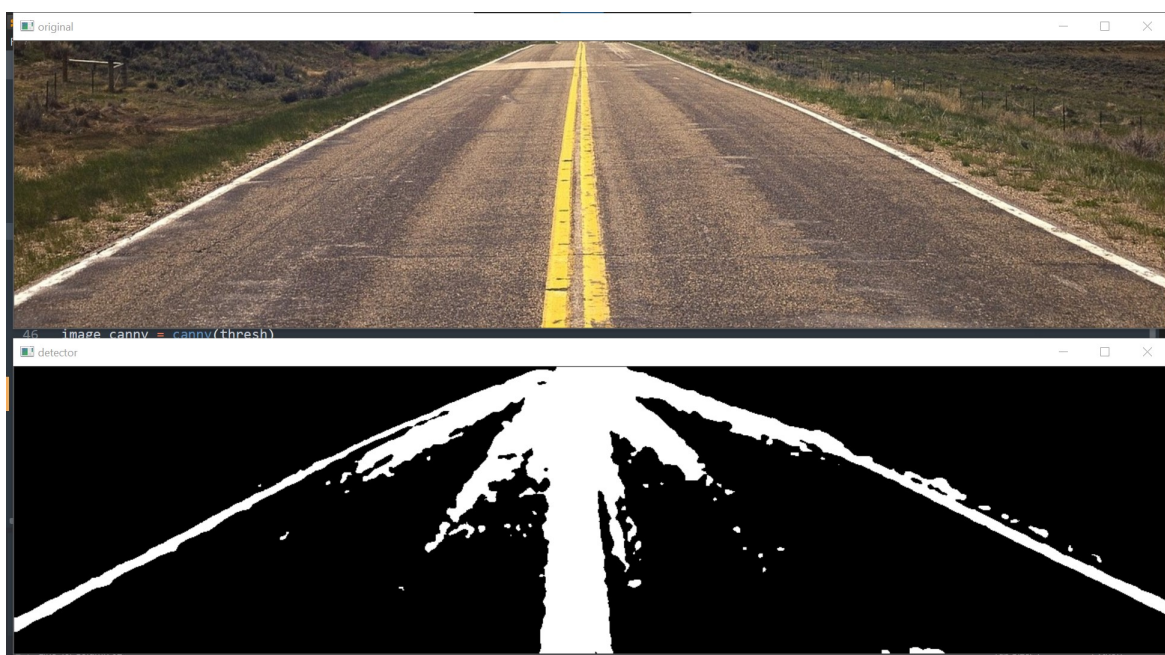


Figura 4.2: Detección de carril con aberraciones

La secuencia que se explicó en el capítulo anterior es la que generalmente se aplica a las imágenes cuando se busca realizar una detección de objetos, en nuestro caso, específicamente de carriles. Algunos detalles que se observaron durante esta primera prueba incluyen el hecho de que el filtro median, en combinación con el filtrado bilateral, muestra un rendimiento superior en ciertas imágenes en comparación con el filtro gaussiano. Sin embargo, como se menciona previamente, existe un espectro de luz no deseado que afecta negativamente la detección de carriles. Por lo tanto, se implemento un procesamiento diferente, en el cual se

crea una máscara basada en el color deseado, en este caso, amarillo y se combina con otra máscara que contiene exclusivamente el color blanco. Este enfoque proporcionó el resultado que se puede apreciar en la figura 4.3. Aquí en primera instancia se colocó el pequeño pedazo de código que se utilizó para el procesamiento antes descrito. Mediante este proceso, se eliminó el espectro de luz, así como, se enfocó de manera más precisa la detección de los bordes de la carretera y las líneas centrales, que son las áreas de interés principal.

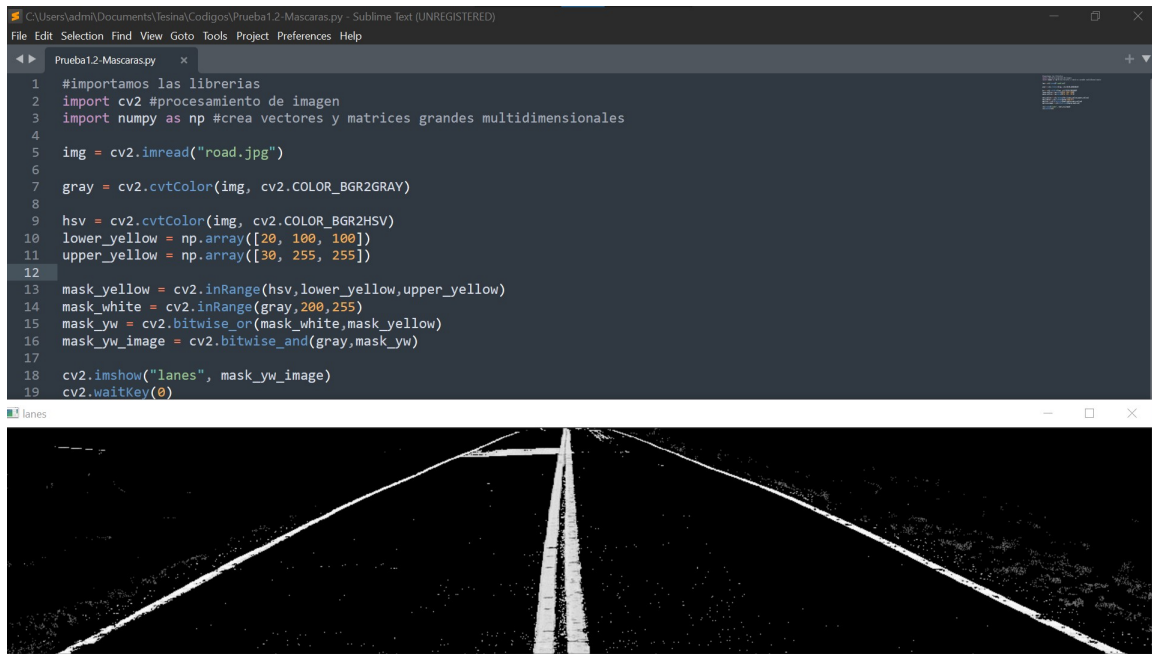


Figura 4.3: Detección de carril sin aberraciones

4.3. Prueba 2: Detección del centro de carril en video

En la figura 4.4 se puede apreciar los bordes difusos de la carretera y los bordes de las barandillas protectoras del carril marcados en color verde, a su vez que logra detectar bien el centro del carril marcado en color negro.



Figura 4.4: Prueba2-Detector2

El resultado de esta prueba fue medianamente satisfactorio ya que la detección de bordes no se realizó de la manera esperada, sin embargo la detección del centro del carril fue bastante buena, por lo que esta forma continuará presente en las siguientes pruebas.

4.4. Pruebas de detección de color

En esta figura 4.5 se puede observar unas manchas de color blanco que corresponden al espectro de color blanco, esto en un fondo negro.

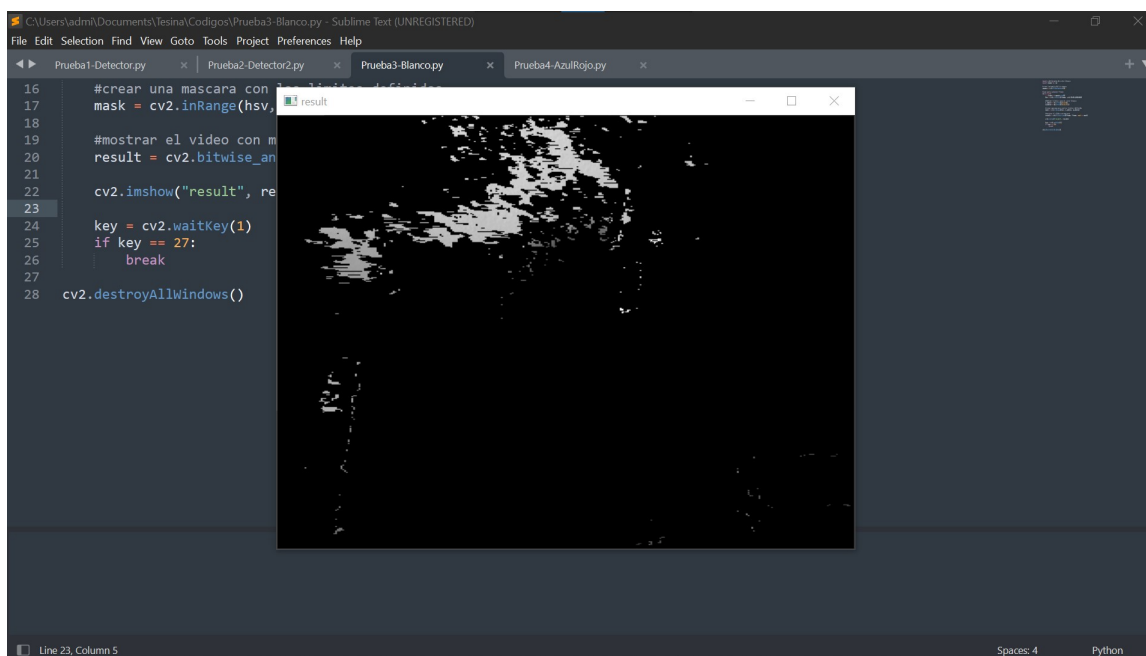
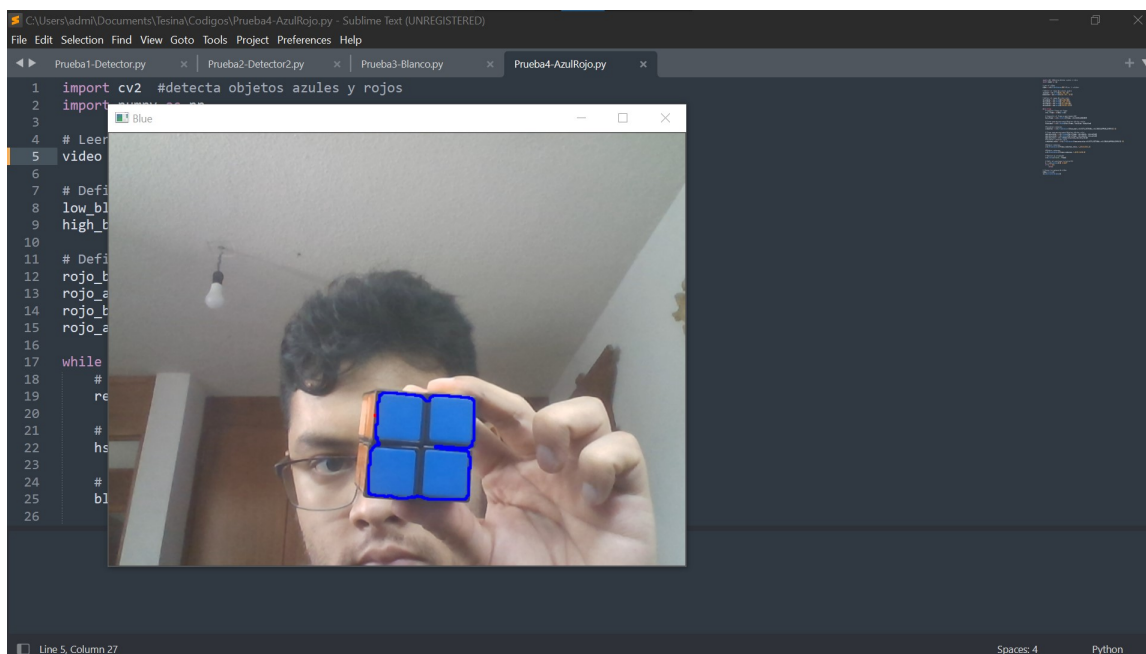


Figura 4.5: Prueba3-Blanco

Esta prueba se llevó a cabo con el propósito de aplicar los conceptos teóricos relacionados con la obtención de un color. Esto implica la correcta determinación del umbral de color según el modelo de color y la creación de una máscara binaria que resalte los píxeles que cumplen con dicho umbral. Aunque los resultados de esta prueba pueden parecer inexactos o un tanto ambiguos, sirvieron como una introducción para la ejecución de una prueba subsiguiente que será de gran utilidad en nuestra tarea principal: la detección de los colores, en especial el color azul.

En la figura 4.6 se puede observar de manera óptima cómo se logra detectar el color azul y cómo se resalta con cuadrados del mismo color en el resultado. Es importante destacar que esta detección excluye otros colores presentes, como el café, blanco y amarillo, que también son visibles en la imagen.

**Figura 4.6:** Prueba4-Azul

En la figura 4.6 se puede apreciar que se obtuvieron resultados bastante satisfactorios lo cual deja un buen preámbulo para poder empezar a combinar las pruebas y darles la aplicación marcada en los objetivos específicos.

Llegado a este punto, se continuaron realizando varias pruebas hasta que se logró desglosar el desarrollo del programa en diferentes etapas. Sin embargo, surgió un desafío cuando se intentó hacer funcionar todos los programas en conjunto. Inicialmente, se consideró la posibilidad de alternar el encendido y apagado de la cámara o los programas como una forma de unir todo el proceso en uno solo. Sin embargo, esta idea resultó ser incorrecta, ya que al realizar el cambio, el proceso se reiniciaba por completo, lo que habría significado que el vehículo avanzara a ciegas en ciertos momentos o incluso que no avanzara en absoluto debido a las pérdidas momentáneas de conexión entre la lógica y la comunicación.

La solución a este problema consistió en crear un programa único que incorporara diferentes filtros y líneas de código específicas para cada situación. Esto se menciona porque, al principio de este subproceso, se contempló la posibilidad de realizar todos los procesos con un mismo filtro o los mismos parámetros para todas las funciones. Sin embargo, resultó ser imposible debido a las diferencias inherentes en la naturaleza de cada función y en los resultados deseados para cada una. Por ejemplo, como se ha mencionado previamente en este trabajo,

la detección de colores requiere trabajar en el espacio de color HSV, mientras que la detección del camino es más efectiva cuando se realiza en escala de grises. La idea de aplicar los mismos parámetros en todas las funciones resultó inviable debido a estas diferencias fundamentales.

Los subsistemas que se mencionan son:

- Detección de carril.
- Detección del centro del carril.
- Detección de objetos de color.
- Detección de dirección.

En las pruebas ya mencionadas, se obtuvo la preparación inicial para los primeros cuatro subsistemas. Las pruebas posteriores se centraron en la afinación del programa, la implementación de la comunicación y la detección de la curvatura del camino. Estos pasos fueron esenciales para poder determinar la dirección en la que el vehículo debía girar el volante.

4.5. Primer prueba de distancia

En esta figura 4.7 se observa un objeto pequeño de color azul que se encuentra a una cierta distancia de la cámara, junto con la estimación de la distancia calculada por el código en la consola. También se nota que la detección de color está identificando otros objetos que entran en el rango de color azul, aunque no necesariamente son de dicho color.

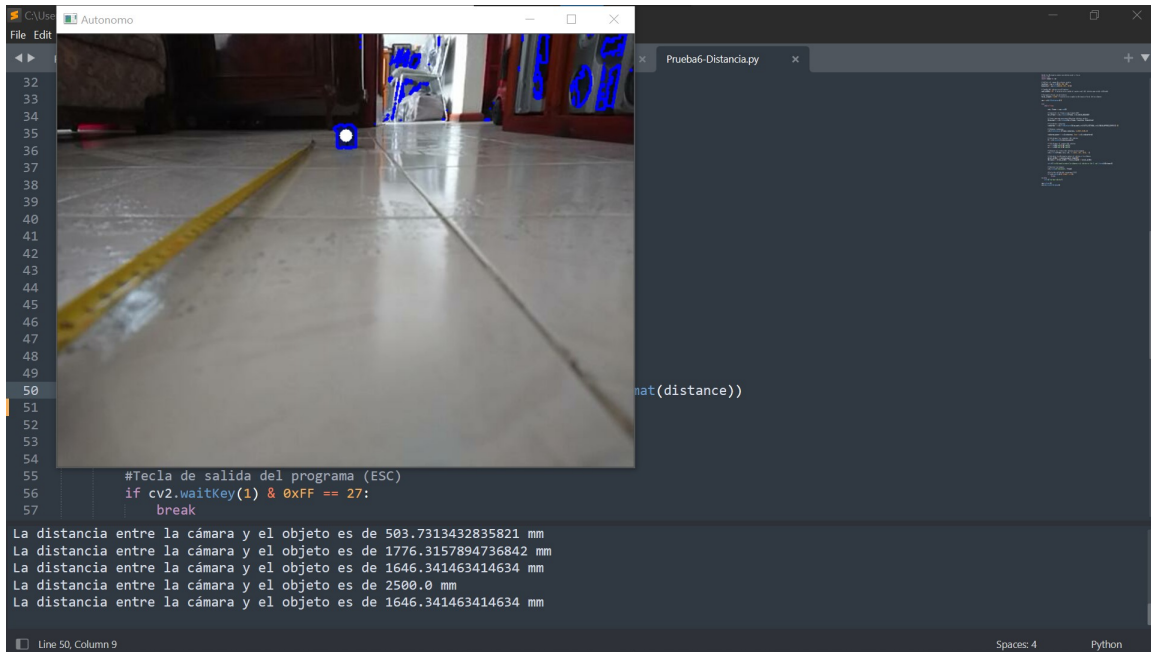


Figura 4.7: Prueba de distancia

La distancia a la que se colocó el objeto fue de un metro, y se aplicó la fórmula de valor experimental porcentual que puede apreciarse a continuación:

$$E_{ex} = \frac{|ValorTeorico - ValorExperimental|}{ValorTeorico} \times 100 \%$$

Se tomaron dos valores de estimación, el más bajo que fue de 503.7 y el más alto de 2500. Al aplicar la fórmula mencionada previamente, se obtuvieron valores de error experimental significativamente altos:

$$E_{ex} = \frac{|1000 - 503.7|}{1000} \times 100 \% = 49.63 \%$$

$$E_{ex} = \frac{|1000 - 2500|}{1000} \times 100 \% = 150 \%$$

Dada la magnitud de estos errores, se tomó la decisión de realizar una nueva prueba para medir la distancia. Esto se debió a la importancia de obtener mediciones precisas en el contexto de un vehículo y las implicaciones de seguridad asociadas.

Se comenzó a trabajar con la técnica de visión estéreo. Se utilizaron dos cámaras de teléfono celular: un Samsung A52s y un Sony Xperia XZ1, esto debido a la practicidad de tener los dispositivos a la mano. Con ayuda del software iVcam se conectaron ambos teléfonos celulares a la computadora para realizar las pruebas. Esta configuración de cámaras es en conjunto es decir, ambas cámaras actúan sobre el sistema a la par, algo como lo que nuestros ojos hacen al ver un objeto.

A pesar de que los errores no variaron significativamente en comparación con las pruebas anteriores, se decidieron agregar líneas de código al programa que sería sometido a condiciones de trabajo reales para evaluar si su comportamiento cambiaba al definir las posiciones de las cámaras y observar cómo se desempeñaba en situaciones de trabajo real.

Finalmente, se incorporaron líneas de código que incluyeron el tamaño del objeto en milímetros, la distancia entre las cámaras y la escala de disparidad, así como una calibración de las cámaras utilizando la función StereoBM de OpenCV.

4.6. Primer prueba completa de software

En la figura 4.8, se puede observar cómo el sistema logra detectar con éxito el centro del camino y el color azul, así como realizar una detección parcial del borde del camino. Sin embargo, se detecta el color verde en áreas donde no debería estar presente debido a la presencia de matices de azul que se encuentran dentro del rango de colores definido para el verde y el azul.



Figura 4.8: Prueba7-Completo

A pesar de este pequeño fallo, se tomó la decisión de avanzar hacia la primera prueba de campo con el propósito de evaluar el rendimiento del programa en situaciones del mundo real, dado que todas las pruebas anteriores se habían realizado utilizando videos pregrabados.

4.7. Primer prueba de campo

En la figura 4.9 se observa un espacio cerrado rodeado de numerosos objetos y máquinas, una línea de color amarillo y el techo de color blanco. Se aprecia que el sistema detecta el color blanco y lo marca en azul claro, pero no logra detectar el camino.



Figura 4.9: Prueba de Campo I

En ese mismo momento, se realizaron modificaciones en el código para habilitar la detección de los bordes amarillos, y los resultados se encuentran ilustrados en la figura 4.10. Estas modificaciones en el programa fueron mínimas, como la inclusión del rango de color amarillo y las máscaras que definen el umbral para llevar a cabo la detección del color amarillo. Por lo tanto, estos cambios no se consideran como un nuevo programa independiente.

La prueba arrojó un resultado mejor en comparación con la anterior, como se puede observar en la figura 4.10. El algoritmo logra detectar el camino de color amarillo y lo marca, aunque

se detectan otros objetos del mismo color y es importante destacar que el centro marcado no coincide precisamente con el centro del camino.

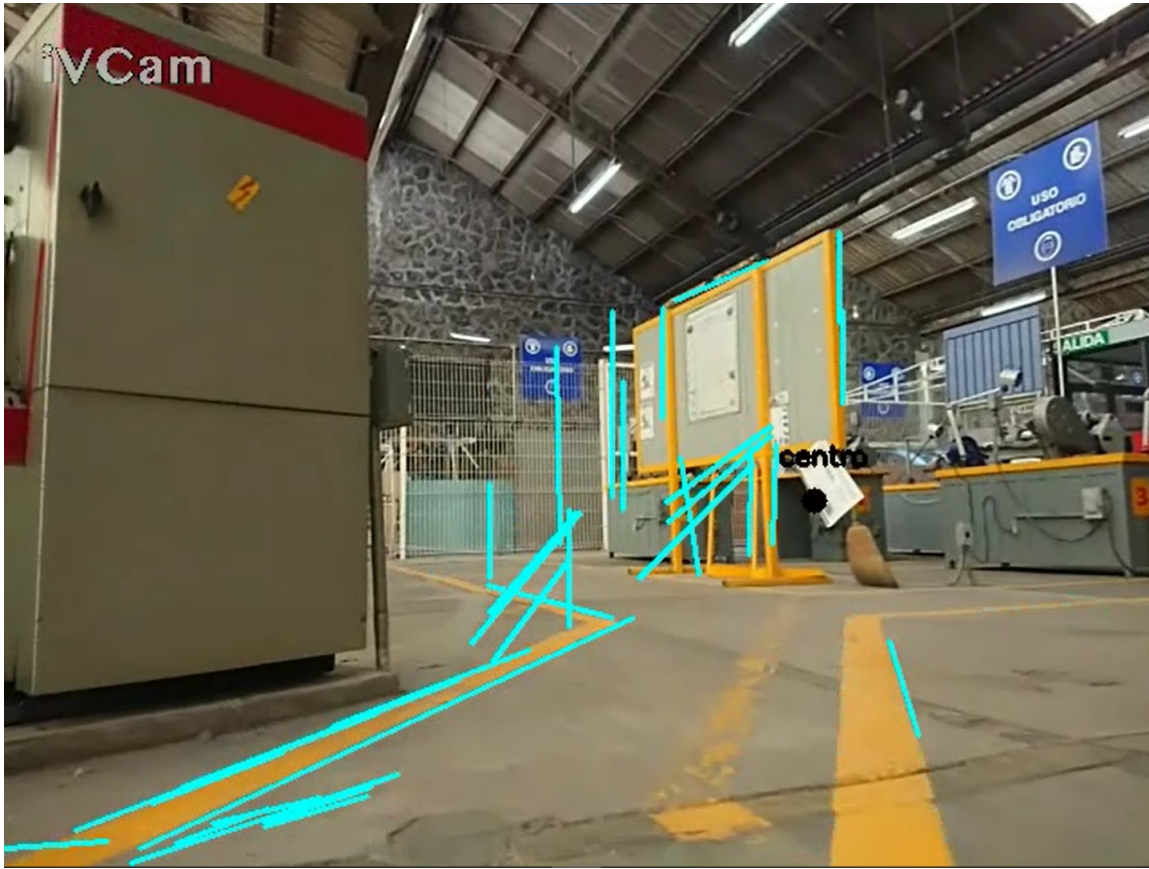


Figura 4.10: Prueba de Campo II

Esta parte de la prueba no proporciona una conclusión definitiva, ya que la detección del camino se lleva a cabo de manera deficiente al igual que la detección del centro.

Esta prueba también incluyó la detección de colores. En la figura 4.11, se puede apreciar que en lo que respecta al color azul y verde, el sistema tiende a confundir tonalidades de azul con verde, tal como se observó anteriormente en la prueba de software. Por otro lado, la detección del color amarillo se realizó de manera efectiva.

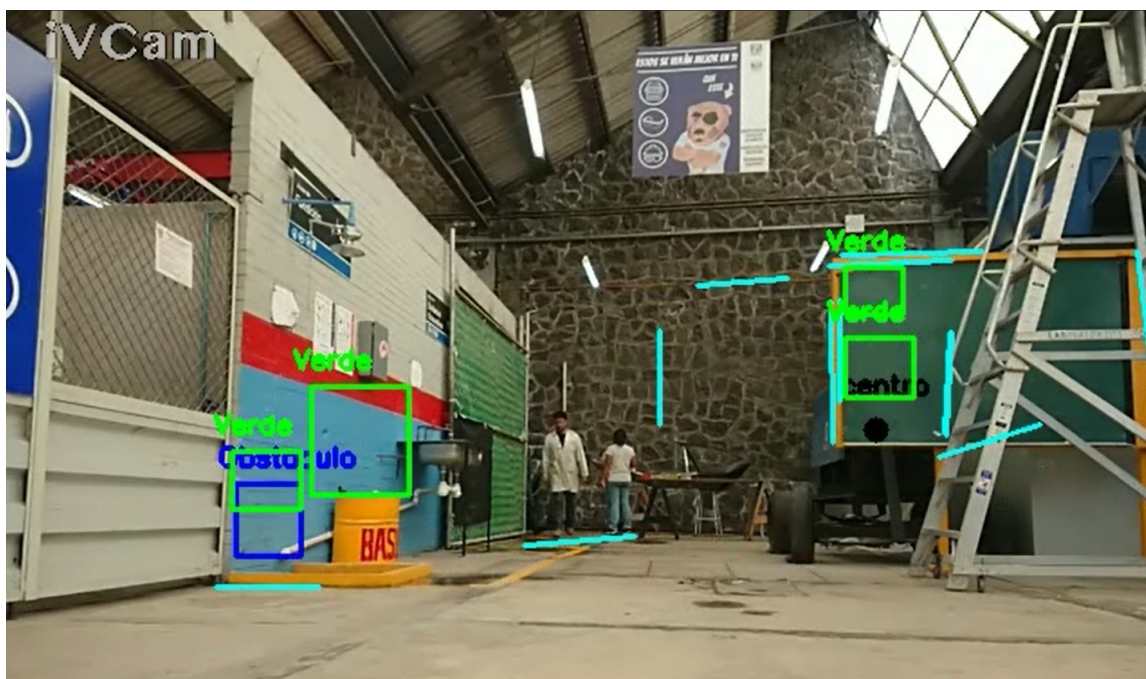


Figura 4.11: Detección de color - Campo

En este sentido, tampoco se observaron cambios significativos al pasar del entorno virtual al mundo real.

Como se puede apreciar en la figura 4.12, esta muestra capturas de pantalla de los archivos obtenidos durante las pruebas. Estos archivos son las estimaciones de distancia entre la computadora y objetos de color azul y la dirección hacia donde dar la vuelta si es que se presenta una curva.

```

1 #Completo 11
2 #Importamos las librerías
3 import cv2
4 import serial
5
6 # Salva
7 archivo
8 fps = 2
9 tamaño
10 video_s
11 archivo
12 video_s
13 archivo
14 video_s
15 # Tamañ
16 real_wi
17 # Dist
18 baselin
19 dispari
20 # Escal
21 dispari
22 # Confi
23 cap_lef
24 cap_rig
25 cap_lef
26 cap_rig
27 cap_lef
28 cap_rig
29 cap_rig
30 # Calib
31 stereo
32 stereo
33 stereo
34 stereo
35 stereo
36 stereo
37 stereo
38 stereo

```

```

La dirección de la curva es de: 30.439724380643128°
derecha
La dirección de la curva es de: 30.48278062007951°
derecha
La dirección de la curva es de: 6.731237122814964°
derecha
La dirección de la curva es de: 7.765166018425333°
derecha
La dirección de la curva es de: 9.08524391139104°
derecha
La dirección de la curva es de: 9.08524391139104°
derecha
La dirección de la curva es de: 6.745579647482819°
derecha
La dirección de la curva es de: 5.799217629098416°
derecha
La dirección de la curva es de: 6.745579647482819°
derecha
La dirección de la curva es de: 6.745579647482819°
derecha
La dirección de la curva es de: 7.9406670436299995°
derecha
La dirección de la curva es de: 7.937776688148746°
derecha
La dirección de la curva es de: 7.8996916142652305°
derecha
La dirección de la curva es de: 7.648636548317627°
derecha
La dirección de la curva es de: 8.16847862404549°
derecha

```

```

Distancia: -507741.11 mm
Distancia: -494825.38 mm
Distancia: -26182.84 mm
Distancia: -26048.54 mm
Distancia: -26271.58 mm
Distancia: -27118.96 mm
Distancia: -27196.90 mm
Distancia: -24287.56 mm
Distancia: -22830.61 mm
Distancia: -21449.59 mm
Distancia: -21413.94 mm
Distancia: -20302.99 mm
Distancia: -25716.68 mm
Distancia: -33074.57 mm
Distancia: -37306.06 mm
Distancia: -41053.31 mm
Distancia: -38332.58 mm
Distancia: -36745.84 mm
Distancia: -42908.66 mm
Distancia: -40768.63 mm
Distancia: -35567.46 mm
Distancia: -37442.62 mm
Distancia: -38229.31 mm
Distancia: -33327.67 mm
Distancia: -31097.28 mm
Distancia: -30824.87 mm
Distancia: -29645.34 mm
Distancia: -26503.29 mm
Distancia: -25173.81 mm

```

Figura 4.12: Documentos generados tras la prueba de campo

En relación a la medición de distancia, la calibración resultó poco exitosa y arrojó distancias negativas. De manera similar a lo observado con la detección de colores, el traslado al entorno del mundo real no tuvo un impacto significativo en la ejecución del programa. En cuanto a la funcionalidad de guardar los videos generados por el procesamiento y los datos obtenidos para su posterior envío a través de comunicación serial, esta información fue almacenada con éxito en la computadora.

A través de estas pruebas, se hizo evidente que la iluminación juega un papel crucial en la detección de colores. Hasta este punto, la detección de color había sido el primer paso para percibir el entorno en todo el sistema. Sin embargo, al enfocarse en la detección de carriles a través de los colores de las líneas del mismo, surgió un problema importante: los rangos de color en el mundo real resultaron ser diferentes a los rangos de un entorno ideal esto se debe a que la iluminación varía según el lugar, la posición y la hora en que se realice el experimento, mientras que los videos pregrabados siempre mantienen los mismos parámetros.

4.8. Segunda Prueba de software

En la figura 4.13 se muestra una carretera con poca curvatura, donde los bordes del carril central se resaltan en colores rojo y azul. En la esquina superior izquierda, se puede observar la curvatura estimada de la carretera en metros y la posición estimada del vehículo en relación al centro del carril en centímetros.



Figura 4.13: Segunda Prueba de software

Los resultados obtenidos son bastante satisfactorios, ya que el programa logra un reconocimiento efectivo del carril central de la carretera según los alcances del proyecto. Con respecto a la información sobre la posición del vehículo y la estimación de la curvatura, se planea mejorar o eliminar este dato, dependiendo de los resultados que se obtengan en las próximas pruebas de campo.

4.9. Pruebas de detección de distancia

Después de realizar pruebas iniciales relacionadas con la medición de la distancia, se identificó que para lograr un desempeño satisfactorio en la visión estéreo, era esencial contar con una calibración precisa de las cámaras. Esta técnica busca replicar la percepción de profundidad y distancia que posee el ojo humano.

Iniciando con el proceso descrito en el capítulo anterior, se comenzó capturando fotografías de un tablero de ajedrez utilizando ambas cámaras 4.14.

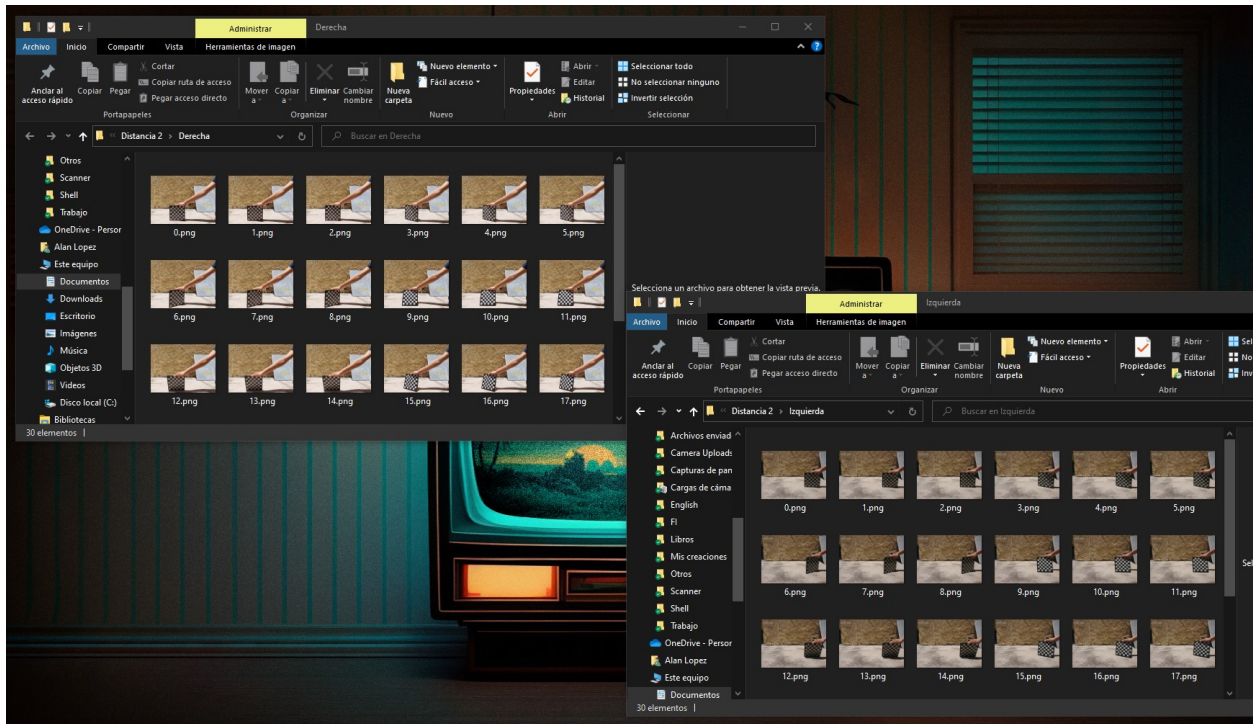


Figura 4.14: Imágenes Obtenidas para calibración

Luego, se procedió a llevar a cabo la calibración, lo que resultó en la generación de un archivo XML que contenía los resultados de dicha calibración. Finalmente, se realizó la prueba de profundidad utilizando dos imágenes capturadas con el mismo programa obturador y a una distancia de 80 cm. Los resultados de esta prueba se pueden observar en la figura:

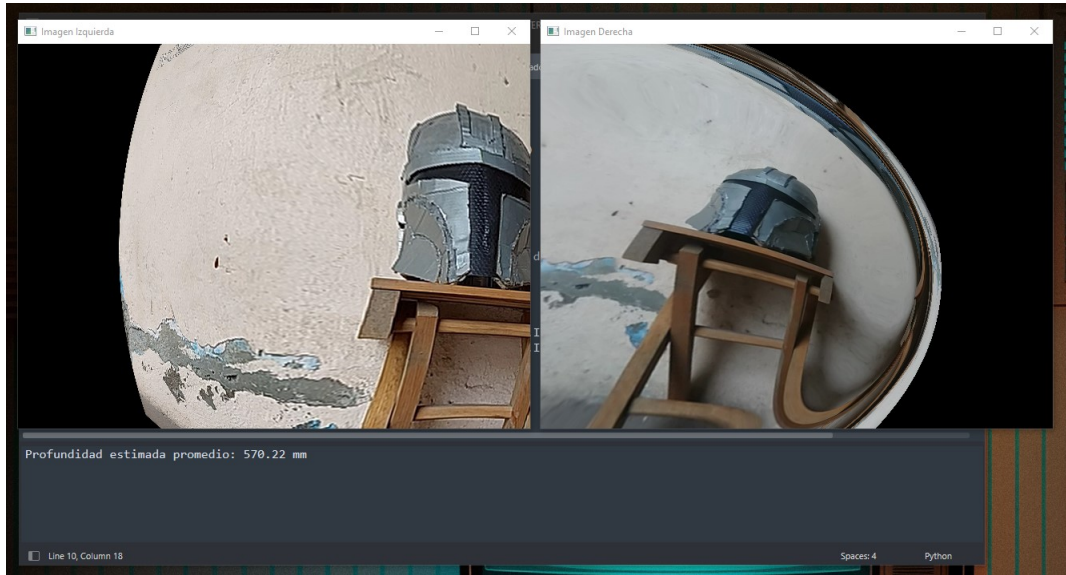


Figura 4.15: Resultado de la estimación de distancia

A pesar de haber intentado realizar la calibración, como se puede apreciar en la figura 4.15, las imágenes resultantes después de pasar por el proceso de calibración presentan distorsiones significativas. Esto se traduce en una estimación incorrecta de 57 cm con un error experimental de:

$$E_{ex} = \frac{|80 - 57|}{80} \times 100\% = 28.75\%$$

Agregando que al mover el objeto este error aumentaba significativamente.

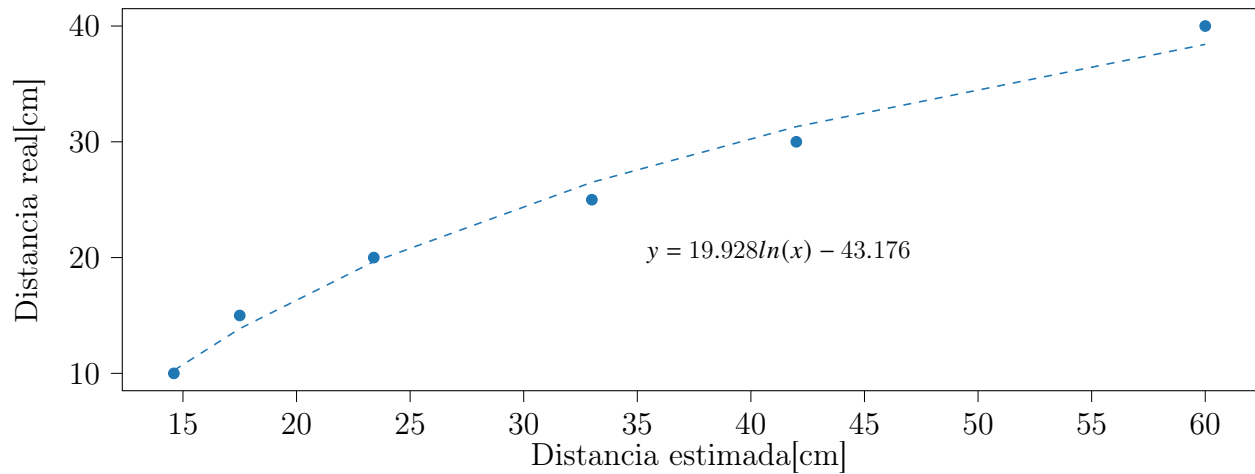
Dados los resultados en la prueba de Visión Estéreo, se exploró el uso de una técnica de triangulación estéreo en el programa, esto debido a que es un error significativamente alto el que se obtuvo con la técnica pasada.

Con este enfoque se lograron estimaciones que se acercaban mucho más al valor real, como se puede observar en la siguiente tabla:

Tabla 4.1: Datos ocupados para la regresión logarítmica

Valor estimado [cm]	Valor real [cm]
14.60	10.00
17.50	15.00
23.42	20.00
33.00	25.00
42.00	30.00
60.00	40.00

Al hacer una graficación de estos datos, se observó que el experimento mostraba una tendencia hacia una gráfica logarítmica. Por lo tanto, se tomó la decisión de llevar a cabo una regresión logarítmica para corregir el error y obtener así una estimación más cercana al valor real. Esta regresión se puede apreciar en la figura:

**Figura 4.16:** Regresión Logarítmica

La ecuación obtenida tras la graficación se anexo al código y con ello se obtuvieron los siguientes resultados:

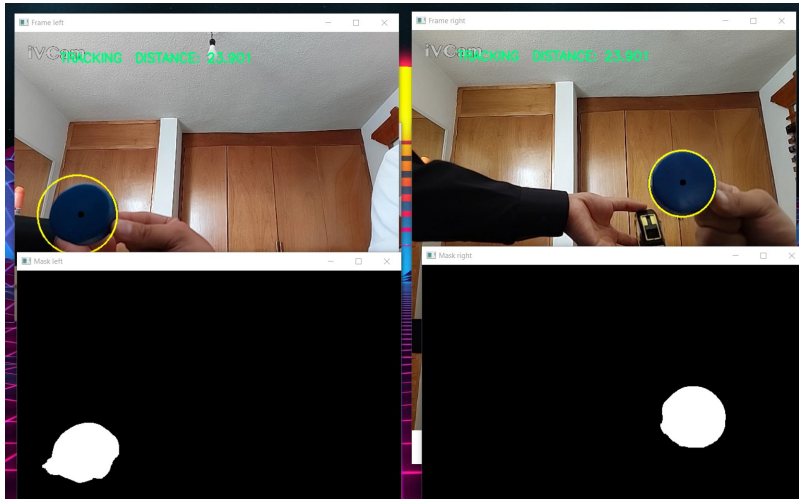


Figura 4.17: Estimación de distancia con triangulación estéreo



Figura 4.18: Distancia del objeto a la cámara

Como se puede observar en las figuras 4.17 y 4.18 la distancia a la que se encontraba el objeto era de 20 cm, y la distancia obtenida mediante la estimación en el software fue de 23.9 cm,

$$E_{ex} = \frac{|20 - 23.9|}{20} \times 100 \% = 19.5 \%$$

A pesar de que el error fue significativo, la diferencia clave con respecto a la prueba anterior fue que en este caso se observó un error constante de alrededor de 3 a 4 cm. Esto contrasta con la prueba anterior, donde el error aumentaba a medida que se alejaba el objeto, sin embargo se seguirá iterando para obtener un mejor resultado.

4.10. Segunda Prueba de campo

Con la realización de algoritmos pasados se inicio el proceso para la elección de las cámaras finales para el proyecto. Como ya se menciono anteriormente en este trabajo, la primer propuesta que se hizo fue la utilización de celulares, se empezó utilizando un Samsung A52s con una cámara de 64 MP(F1.8) OIS y un Xperia XZ con una cámara de 19 MP(F2.0) sin embargo el algoritmo de la detección de profundidad en las dos técnicas anteriores necesitaba que se utilizaran cámaras con especificaciones idénticas, por lo que se opto por utilizar dos Samsung A52s, con lo que se obtuvo un buen resultado en cuanto a la debido a la calidad de la imagen, sin embargo se hizo una prueba mas con dos cámaras web con una resolución Full HD con las cuales se obtuvieron los siguientes resultados:

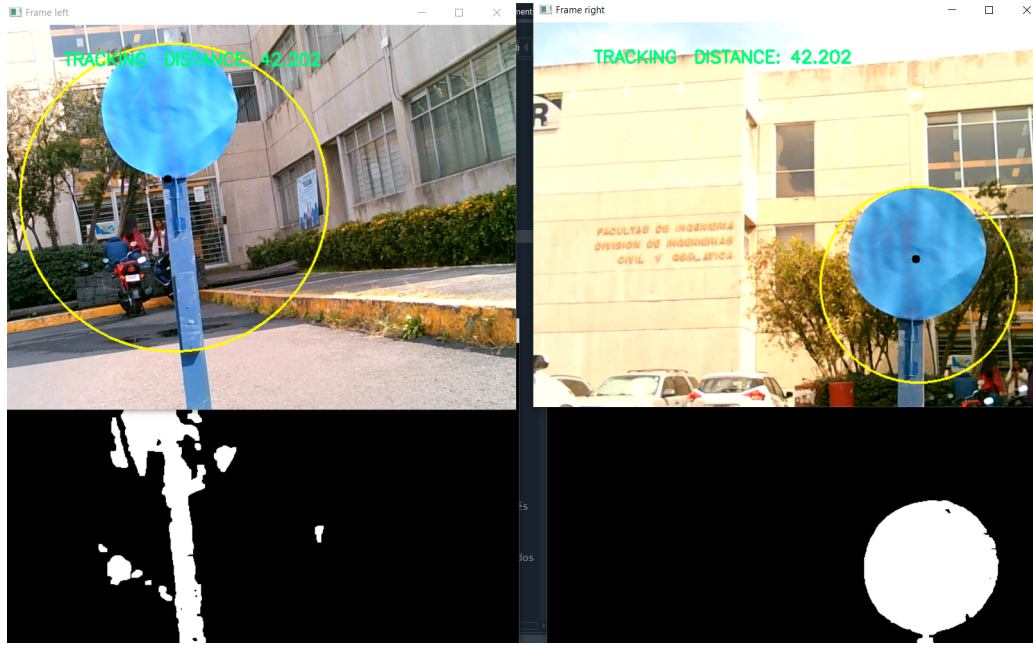


Figura 4.19: Resultado de la utilización de cámaras web

La detección de colores se hizo de buena manera por lo que se optó por tener estas dos cámaras web para este subsistema esto debido a que las cámaras web son más pequeñas que los celulares y realmente la calidad de la imagen no refleja un cambio significativo en este sistema. Por otro lado para el subsistema de detección de carriles, se hizo la prueba con una de las cámaras web antes utilizada y un A52s se obtuvo el siguiente resultado:



Figura 4.20: (a) Resultado de la utilización del A52s (b) Resultado de la utilización de la cámara web

El resultado que podemos ver aquí es que el foco de la imagen obtenida con la cámara web es algo inferior al obtenido con el A52s, así como la calidad de la imagen. Cabe destacar que ambas imágenes se tomaron con las cámaras en la misma posición y con un par de minutos de diferencia por lo que se optara por la utilización del celular para este subsistema, ya que al estar en la misma posición y otorgarnos un rango de imagen mas amplio es algo bastante mas favorecedor, aunado a que el reconocimiento del camino fue mas preciso con el celular.

En cuanto a los datos de curvatura y posición del coche, el primero resulta aceptable, ya que en un carril recto la curvatura se aproxima a cero. Sin embargo, el dato de la posición del vehículo muestra un error considerable respecto al centro, lo que sugiere su eliminación en futuras pruebas. La curvatura se descarta porque, para los objetivos actuales del proyecto, es irrelevante dado que el software no se implementará en un vehículo completamente autónomo. Por su parte, la posición se omite debido al margen de error significativo, dejándose ambos aspectos para trabajo futuro.

Con respecto a la detección de carriles hasta el momento de esta prueba se han utilizado métodos tradicionales para realizar dicha detección, esto quiere decir que aplicamos filtros y técnicas propias de la visión por computadora, con estas técnicas se logro hacer una detección de carril medianamente decente, que depende en su totalidad de los puntos desde donde se quiere realizar la medición. los resultados de esto se podría decir que son correctos según los alcances del proyecto, hasta el punto en el que el vehículo se encuentra con curvas demasiado pronunciadas, con curvas ligeras de carreteras la detección tiene un ligero desfase pero que no afecta en si a la visión del carril. Pero cuando la curva se pronuncia demasiado la detección se pierde.

Este algoritmo se probó en el vehículo prototipo Xipe. Se preparó el camino dibujando los carriles con cinta canela, con un tamaño de 140 cm entre carriles. Con este camino, la detección del carril se realizó de manera óptima. Sin embargo, debido al tamaño del carril y al foco de la cámara, cuando el vehículo se aproximaba a las curvas, el camino se perdía y, por lo tanto, la detección fallaba. Los resultados de esta prueba se pueden apreciar en la siguiente figura:



(a) Deteccion de Carril



(b) Perdida del carril

Figura 4.21: Prueba del software en Xipe

Con este resultado en mente, para la prueba final se buscará un camino con carriles del tamaño de un vehículo convencional, es decir, un camino que se puede encontrar en las carreteras convencionales.

4.11. Prueba final de detección de distancia

Para la realización de la calibración se obtuvieron las siguientes duplas:

Tabla 4.2: Datos obtenidos para la calibración

Área [Píxeles al cuadrado]	Valor real [cm]
21444.4673	133
16764.7456	150
11728.0579	180
10252.7997	200
6156.67	250
4192.2291	300
2342.4543	400
1513.4187	500

Una vez obtenidas las duplas se observó que la gráfica que se formaba tenía forma de una potencia por lo que se optó por realizar una regresión.

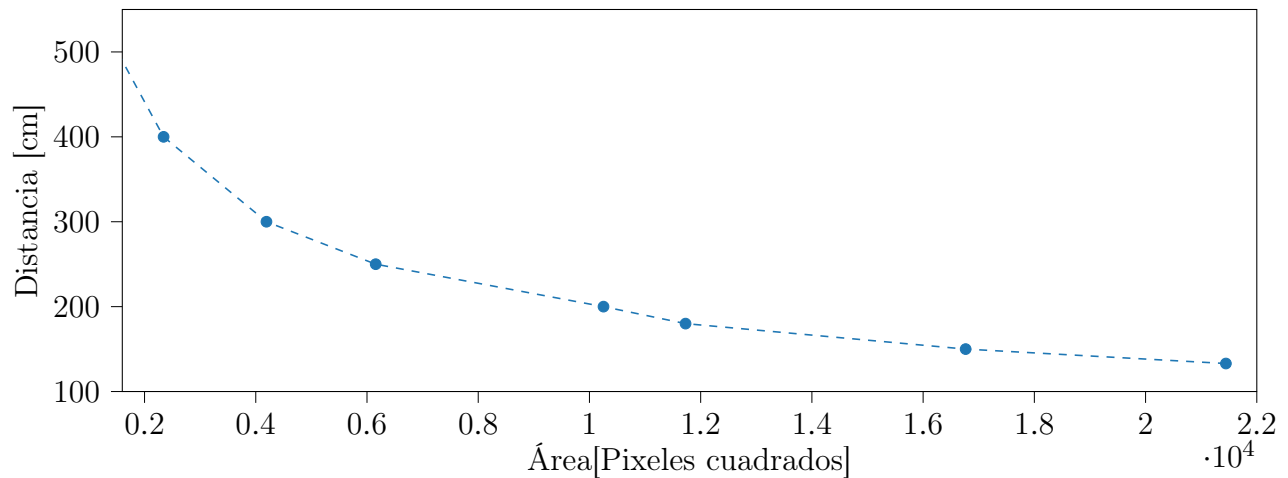


Figura 4.22: Regresión: $y = 18921x^{-0.496}$

Con esta última prueba, se obtuvieron los siguientes datos de la estimación de distancia:

Tabla 4.3: Datos experimentales

Valor Real [cm]	Valor Estimado [cm]	Porcentaje de Error [%]
147	148.83	1.24
200	196.3	1.85
424	420.55	0.81
300	297	1.00
980	1000	2.04

Como se puede observar en la tabla, el error experimental disminuyó considerablemente en comparación con la prueba anterior utilizando la técnica de triangulación. Finalmente, la figura 4.37 confirma que a una distancia medida con el flexómetro de 147 cm, el programa estimó una distancia de 148.83 cm.



(a) Distancia Estimada



(b) Distancia Medida

Figura 4.23: Comparación de la distancia real con la distancia estimada

Una ventaja de esta técnica de percepción visual en comparación con la triangulación estéreo es que solo requiere el uso de una cámara para funcionar adecuadamente, simplificando el diseño y reduciendo la complejidad del sistema. Este factor, junto con el error experimental

obtenido, es lo que inclinó la decisión hacia el uso de esta técnica para la prueba final sobre las otras dos alternativas evaluadas previamente.

4.12. Diseño de piezas para el hardware

Para el primer diseño, que se puede observar en la siguiente figura, se utilizaron dos partes prácticamente iguales. A los lados, se encuentra el espacio para colocar una tuerca y un tornillo de $1/8$, y en la parte media, una media circunferencia para sujetar los espejos retrovisores y una esfera.

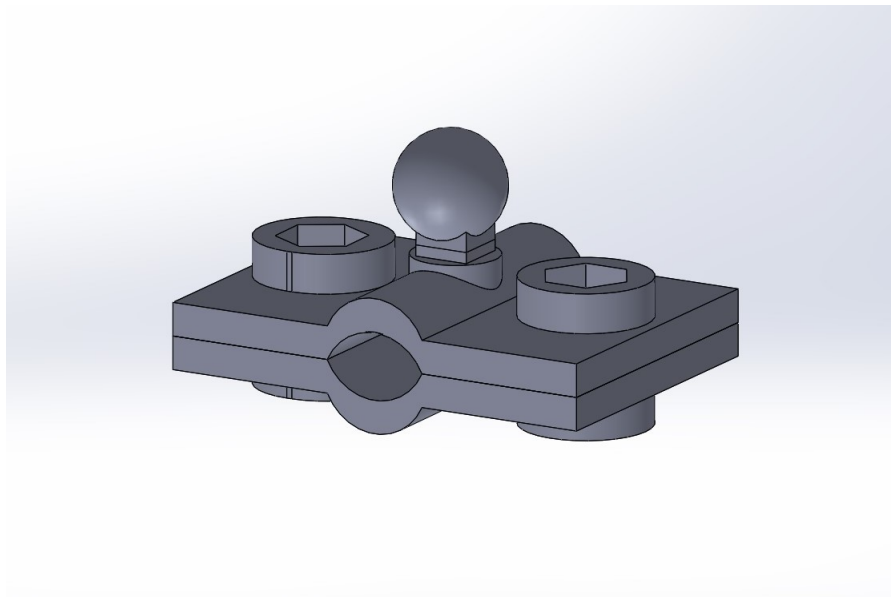


Figura 4.24: Primer diseño porta-cámaras

El inconveniente surgió cuando la esfera no logró sostener la cámara debido a su peso. Para completar este diseño, es necesario implementar algún método para sujetar la cámara con la esfera, como una rosca o un tornillo. Sin embargo, debido al pequeño tamaño del diseño, esto resulta complicado a la hora de manufacturar.

En el segundo diseño, que consta de tres piezas que en conjunto forman un nudo universal, se utilizaron tornillos de $1/8$ en las juntas.

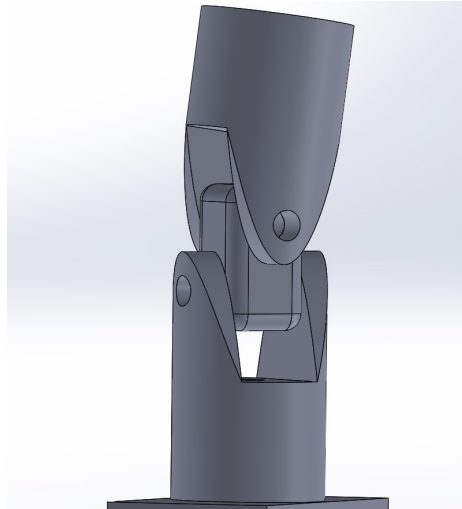


Figura 4.25: Diseño final porta-cámaras

Este diseño cumplió mejor su función, permitiendo realizar algunas de las pruebas con la cámara montada en el vehículo. No obstante, debido a la forma en la que fueron manufacturadas mediante impresión 3D, las piezas sufrieron rupturas en las juntas por la presión aplicada para mantener la cámara en posición.



Figura 4.26: Ruptura del porta-cámaras

Finalmente para la canastilla se realizo el siguiente diseño:

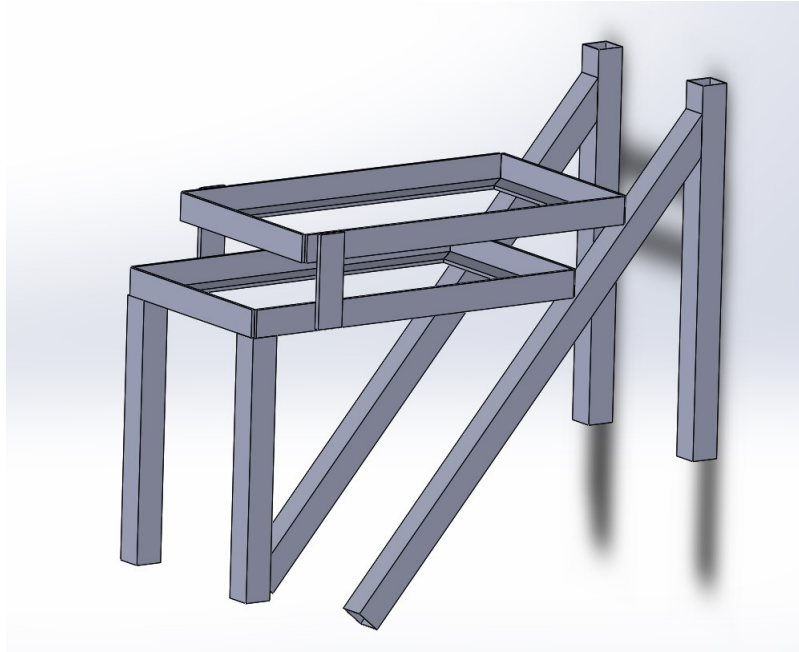


Figura 4.27: Primer diseño para la canastilla

El diseño original de la canastilla consideraba su fabricación con perfiles de 1 pulgada y ángulos de la misma medida, ambos en acero, con la electrónica y las baterías ubicadas en el nivel superior, y la computadora en un segundo nivel. No obstante, durante el proceso de manufactura, el diseño sufrió algunos cambios. Se optó por utilizar únicamente ángulos de 1 pulgada y se añadió un nivel adicional para las baterías, evitando compartir el espacio con la electrónica, como se había planteado inicialmente. Además, se incorporó un soporte más alto a modo de cabecera para mejorar el soporte en los vehículos y facilitar el desmontaje. Estos cambios pueden observarse en las siguientes imágenes:



Figura 4.28: Diseño final de canastilla

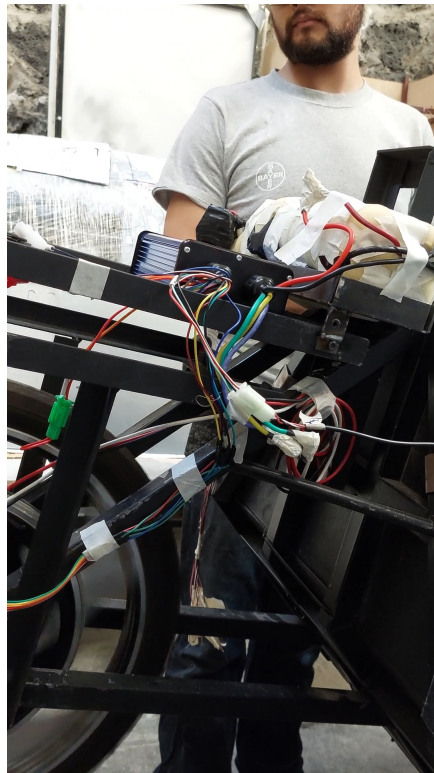


Figura 4.29: Diseño final de canastilla

4.13. Prueba Final

Se inició el proceso de calibración del sistema autónomo, el cual es necesario tanto para el sistema de estimación de distancia como para la detección del camino, con la siguiente configuración de cámaras dispuestas en el vehículo Xipe, la cual ya se menciona en el capítulo anterior.



Figura 4.30: Disposición de cámaras para la prueba final

En lo referente a la estimación de distancia, se obtuvieron las siguientes tuplas:

Tabla 4.4: Datos obtenidos para la calibración

Área [Píxeles al cuadrado]	Valor real [cm]
12800	367
10500	377
10130	387
9600	398
9100	409
8700	420
7600	452

Al realizar la regresión potencial, se obtuvo la siguiente ecuación, que describe la relación entre el área del objeto en píxeles al cuadrado y la distancia del mismo:

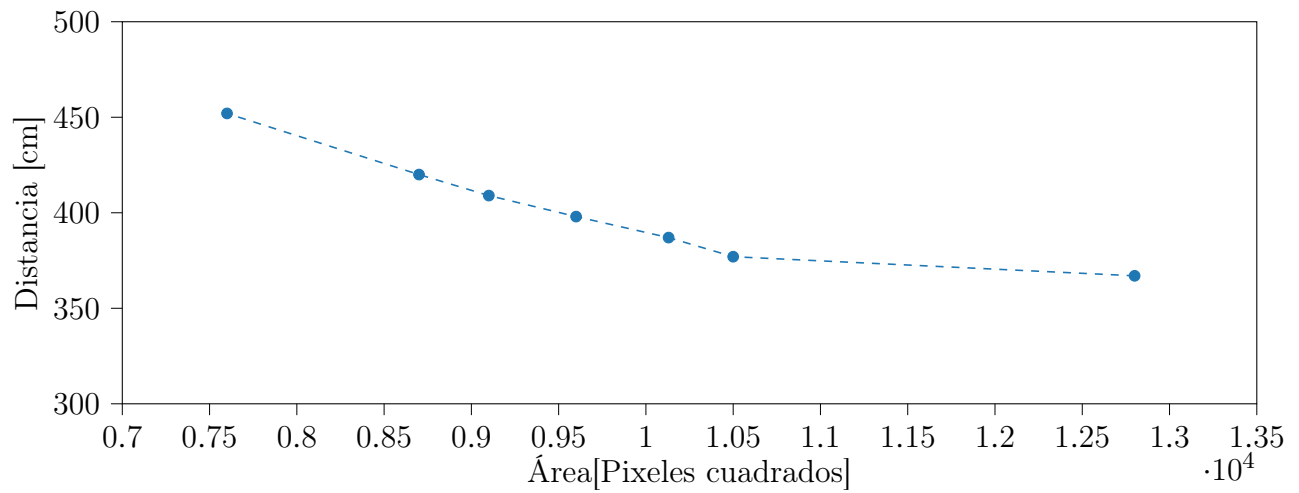


Figura 4.31: Regresión: $y = 17896x^{-0.414}$

Posteriormente, se llevó a cabo la calibración para la detección del carril, como se describió previamente dando como resultado la siguiente región de interés.

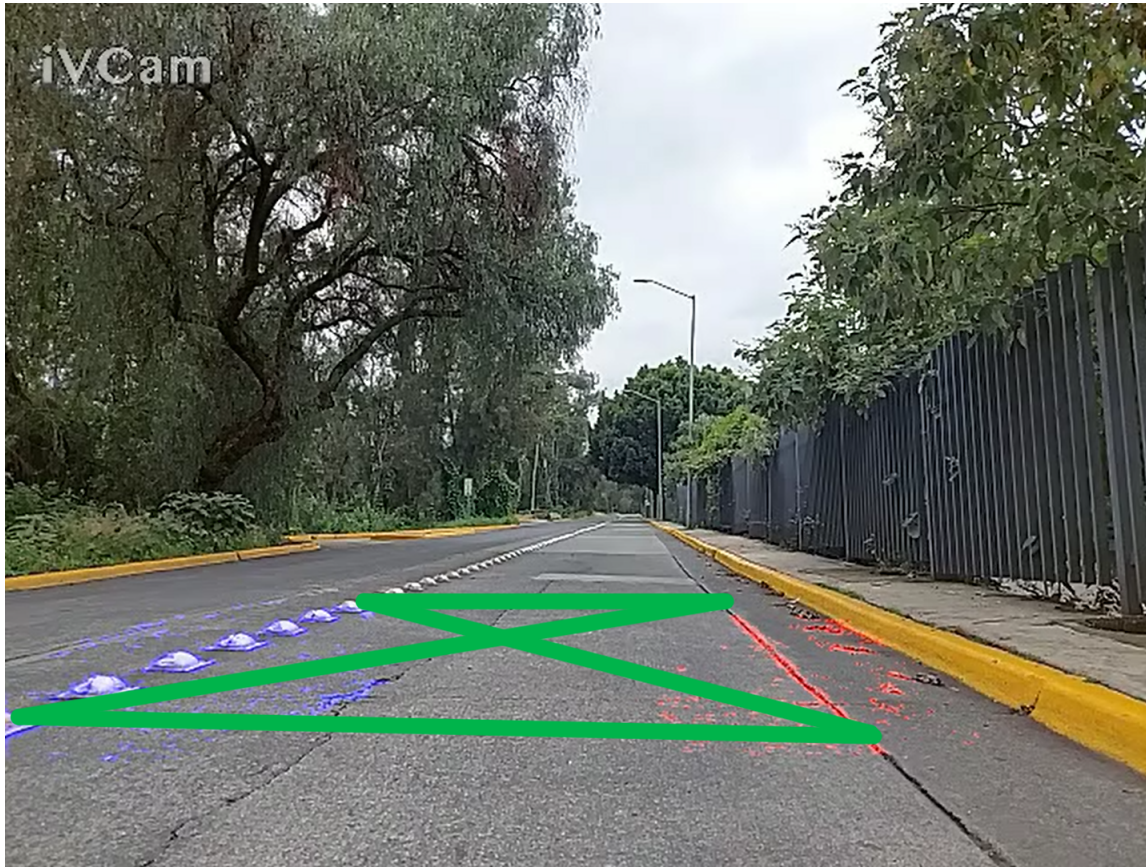


Figura 4.32: Región de interés obtenida por la calibración

Con los programas ya calibrados, se dio inicio a la prueba. En la imagen 4.33 se puede observar claramente la detección de la línea formada por las esferas en el camino, así como la línea marcada en el asfalto.



Figura 4.33: Detección inicial del camino

Este primer frame muestra que la calibración se realizó correctamente, permitiendo que el vehículo comenzara a moverse. Una vez en marcha, se observa que, gracias a la calibración y al posicionar el vehículo más al centro del carril, la línea azul continúa detectando la formación de esferas en el camino, mientras que la línea roja ha cambiado para seguir la línea amarilla que delimita la banqueta 4.34.



Figura 4.34: Ajuste de carril

Con esto, se puede observar que el programa detecta las líneas y las interpreta como parte del carril, de acuerdo con el ancho del mismo obtenido durante la calibración. Este comportamiento se repite incluso con líneas que, para nosotros o para el usuario, no representan un carril propiamente dicho. Un ejemplo de esta situación se puede apreciar en la siguiente figura:



Figura 4.35: Interpretación de carril

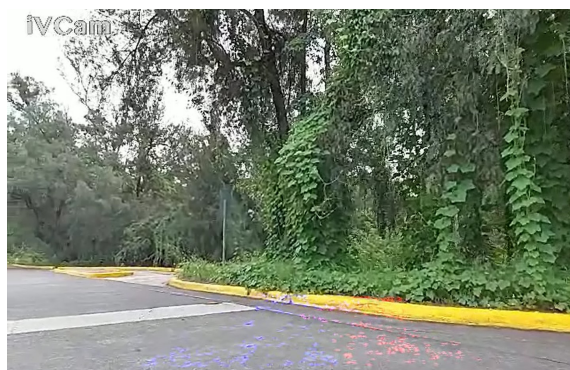
Este comportamiento ocurre porque la calibración instruye al programa a identificar líneas que coincidan con el ancho de carril previamente especificado. El software aún no es capaz de distinguir si se trata de un carril convencional o no; simplemente detecta las líneas y, si cumplen con el tamaño estipulado, las resaltarán en los colores azul y rojo, como se observa en la figura anterior 4.35. Otro ejemplo de este comportamiento se puede ver en la siguiente figura:



Figura 4.36: Detección de cajón de estacionamiento

En esta imagen se puede observar un espacio de estacionamiento que tiene el ancho de un carril, y a pesar de no ser un carril convencional, el software detecta las líneas y las marca con los colores azul y rojo.

Continuando con la prueba, en la imagen 4.37, se muestra un cambio de carril, donde la detección se pierde momentáneamente, pero tras unos instantes, el sistema logra recuperarla.



(a) Pérdida de detección



(b) Detección recuperada

Figura 4.37: Cambio de carril

La calibración permite que el sistema encuentre el carril de acuerdo con la anchura especificada durante el proceso y aplicando las técnicas de detección previamente establecidas, garantiza que el software identifique las líneas dentro del rango calibrado, proporcionando una detección adecuada del carril.

Con respecto a las curvas, también se probó el software en varios escenarios. En la primera curva, que se puede apreciar en la siguiente figura, se observa cómo la línea azul detecta la línea amarilla que corresponde a la separación entre la banqueta y el asfalto, mientras que la línea roja identifica la que divide los dos carriles de la carretera. Cabe destacar que el algoritmo logra detectar esta última línea a pesar de no estar pintada con un color específico o tener cierto grosor, gracias al análisis detallado que realiza el software en combinación con la calibración de la anchura del carril.



Figura 4.38: Detección de curva

Aun cuando la curva alcanzó su punto más pronunciado, la detección se realizó de manera adecuada, manteniendo la misma configuración de la detección 4.39. El algoritmo continuó respetando la calibración inicial y fue capaz de seguir identificando correctamente las líneas que delimitaban el carril, a pesar de la aparición de un vehículo en escena 4.40 , demostrando así su capacidad para adaptarse a las variaciones del camino y entorno.



Figura 4.39: Detección de curva pronunciada



Figura 4.40: Detección de curva pronunciada con vehículo

Cuando se llevó a cabo la prueba en una carretera con carriles pintados de color blanco y con un grosor definido, se pudo apreciar que la detección se realizó de manera correcta ejemplo de esto es la siguiente imagen en donde se destaca cómo la línea azul es detectada de manera continua, tal como está pintada en la carretera, mientras que la línea roja detecta únicamente la sección pintada sobre el asfalto. Este comportamiento demuestra la precisión del algoritmo para identificar las líneas delimitadoras en escenarios más estructurados y definidos.



Figura 4.41: Detección de carril con líneas de grosor definido

Cuando la prueba se trasladó a una entrada de un solo carril, la detección nuevamente fue precisa. En la siguiente imagen se puede apreciar cómo el software detectó de manera correcta los bordes de las banquetas, lo que demuestra la capacidad del algoritmo para adaptarse a distintos tipos de vías, manteniendo la exactitud en la identificación de los límites del camino.



Figura 4.42: Detección de entrada

En esta figura 4.43a se puede observar cómo el software, gracias a la calibración, detectó una línea en el asfalto y el borde de la banqueta, identificándolos como un carril. Este comportamiento se debe nuevamente a que la calibración determina el ancho del carril, y el software, con estos parámetros, realiza la detección de líneas. Si las líneas cumplen con los criterios establecidos, el software las interpreta y las arroja como si fueran parte de un carril, independientemente de si lo son en realidad. Un resultado similar se puede observar en la figura 4.43b. La diferencia en este caso es que una de las dos líneas sí estaba pintada, lo que facilitó su detección por parte del software.

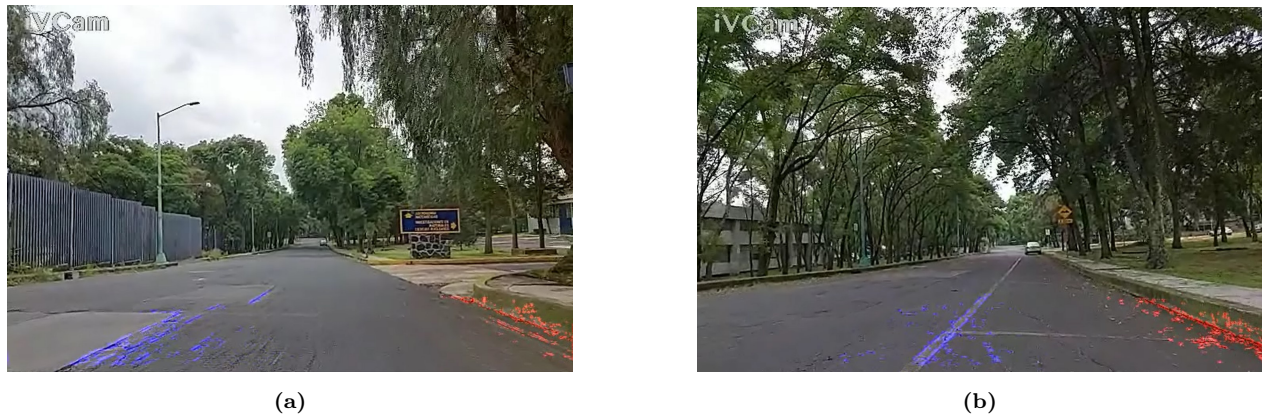


Figura 4.43: Identificación de líneas como carril

En el siguiente ejemplo, ambas líneas estaban pintadas, lo que permitió que el algoritmo detectara el carril de manera más precisa y continua, resaltando correctamente los bordes según los parámetros de calibración establecidos.



Figura 4.44: Detección de carril

Un aspecto a destacar de estos últimos resultados es que, sin importar el color o el ancho

de las líneas, el software realiza de manera adecuada la detección.

Sin embargo, en la figura que se presenta a continuación, se puede observar que, cuando el carril es significativamente más pequeño en comparación con el tamaño definido en la calibración, el software no logra detectar el carril correctamente. Esto provoca un empalme entre los colores que corresponden a las líneas azul y roja, comprometiendo la precisión de la detección en curvas cerradas o carriles estrechos.



Figura 4.45: Empalme en la detección debido a un carril estrecho

A pesar de esta limitación, el software logra detectar y seguir curvas de manera adecuada siempre y cuando el ancho del carril, definido por la calibración, sea el adecuado o incluso un poco más grande. Esto se puede observar claramente en la siguiente imagen, donde el algoritmo sigue correctamente la trayectoria del carril durante una curva, manteniendo la detección de las líneas en su lugar conforme a los parámetros establecidos.

**Figura 4.46:** Seguimiento de curva

Pasando a la parte de la estimación de distancia, que se llevó a cabo simultáneamente con la detección de caminos, los resultados obtenidos fueron los siguientes:

**Figura 4.47:** Prueba de distancia estática

La estimación de distancia fue de 537 cm y la medición con el flexómetro fue de 530 cm.

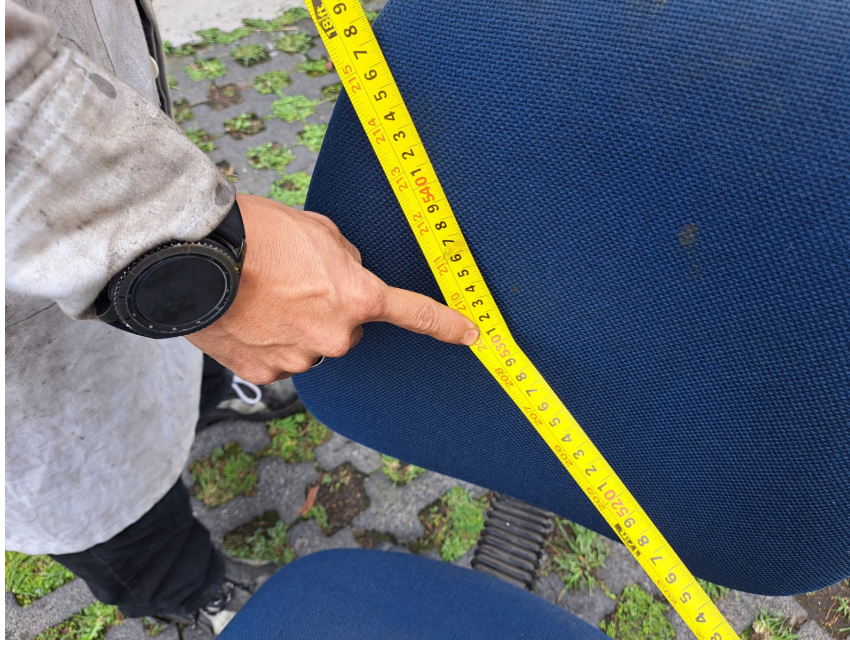


Figura 4.48: Medida entre la cámara y el obstáculo

El porcentaje de error obtenido fue de:

$$E_{ex} = \frac{|530 - 537|}{530} \times 100 \% = 1.32 \%$$

Como se observó anteriormente en las pruebas estáticas referentes a este programa, el porcentaje de error es bastante pequeño. El propósito de esta nueva prueba era evaluar la estimación de distancia con el vehículo en movimiento, y los resultados obtenidos fueron los siguientes:

Tabla 4.5: Datos experimentales

Valor Real [cm]	Valor Estimado [cm]	Porcentaje de Error [%]
840	785.6	6.47
590	564.4	4.33
215	234.7	9.16



Figura 4.49: Estimación de distancia con porcentaje de error de 4.33 %

El obstáculo se colocó frente al vehículo, y fue este último el que se acercó al obstáculo para obtener los resultados reflejados en la tabla. Como se puede observar, los porcentajes de error en esta prueba varían entre el 4 % y el 9 %.

Tabla 4.6: Datos experimentales

Valor Real [cm]	Valor Estimado [cm]	Porcentaje de Error [%]
810	798.1	1.46
587	584.1	0.49
380	382.1	0.55
280	294.5	5.1



Figura 4.50: Estimación de distancia con porcentaje de error de 0.55 %

En esta siguiente iteración, se observan porcentajes de error que van desde el 0.49 % hasta el 5 %, mostrando un incremento considerable cuando el obstáculo está más cerca de la cámara. Esto se debe a que la calibración se realizó con medidas comprendidas entre los 367 y 452 centímetros, como se puede apreciar en la tabla al inicio de esta prueba. Por lo tanto, las distancias inferiores a este rango no son bien compensadas por la ecuación obtenida.

Tabla 4.7: Datos experimentales

Valor Real [cm]	Valor Estimado [cm]	Porcentaje de Error [%]
450	444.8	1.15
400	415.4	3.85
304	316.2	4.01
217	240.8	10.96



Figura 4.51: Estimación de distancia con porcentaje de error de 1.15 %

Este comportamiento se repite, con porcentajes de error que van desde el 1 % hasta el 10 %, lo que confirma que las distancias fuera del rango de calibración presentan un mayor margen de error en la estimación de la distancia.

Aunque el uso de un flexómetro implica ciertos errores de paralelismo y existen métodos de medición más precisos, estos no son necesarios para los objetivos de este proyecto. La precisión en centésimas resulta irrelevante, dado que, al migrar el sistema a un vehículo con funciones automatizadas, acciones como el frenado o el giro del volante se realizarán dentro de un rango específico, según lo determine el diseñador. De forma similar, en la programación, no se redondearon los datos para el despliegue gráfico, ya que este se diseñó únicamente para mostrar un resultado tangible que pudiera documentarse en el desarrollo del proyecto.

Capítulo 5

Conclusiones

El proyecto alcanzó avances importantes en la implementación de sistemas de visión por computadora para la movilidad autónoma. A lo largo de las pruebas, los algoritmos de detección de carriles, colores y estimación de distancias fueron refinados continuamente, aunque surgieron desafíos que requirieron soluciones iterativas.

Las primeras pruebas resultaron útiles para familiarizarse con el entorno y trabajar de manera efectiva con las bibliotecas utilizadas.

Con respecto a la detección de carriles y el centro de carril, uno de los primeros objetivos fue mejorar la precisión en la detección de estos elementos. Inicialmente, los resultados mostraron aberraciones causadas por espectros de luz no deseados. Sin embargo, estos problemas fueron corregidos al aplicar máscaras de color específicas (blanco y amarillo), lo que permitió una mejor identificación de los bordes de la carretera. A pesar de que inicialmente se abordó el problema desde la perspectiva de la detección de color, esto resultó ser un error, lo que ayudó a comprender que la manera óptima de realizar la detección del camino era mediante la segmentación del mismo. En cuanto a la detección del centro del carril, aunque el centro fue identificado con precisión, los bordes de la carretera no siempre fueron detectados con la calidad esperada, lo que hizo que este apartado fuera perdiendo relevancia a medida que avanzaba el proyecto.

En cuanto a la detección de color, las pruebas resultaron útiles para familiarizarse con el entorno de visión artificial. La detección del color azul fue exitosa, pero surgieron dificultades al distinguir ciertos tonos de azul y verde, lo que generó confusiones en algunas pruebas. Este

desafío también se trasladó a las pruebas de campo, donde las variaciones en la iluminación del entorno afectaron el rendimiento de los algoritmos de detección de color. Esto evidenció la necesidad de ajustar los rangos de color según las condiciones ambientales para mejorar la precisión del sistema.

En lo referente a la estimación de distancia, las pruebas iniciales con visión estéreo presentaron errores considerables debido al uso de cámaras distintas y problemas en la calibración. Estos obstáculos llevaron a la exploración de técnicas adicionales, como la triangulación estéreo y una mejor calibración de cámaras. A medida que el proyecto avanzó, los errores en la estimación de distancia disminuyeron significativamente con la implementación de regresiones logarítmicas y potenciales, así como mejoras en los algoritmos. Aunque los errores no se eliminaron por completo, los resultados fueron lo suficientemente precisos para continuar con el desarrollo del proyecto.

Cuando el proyecto se llevó a cabo en las primeras pruebas de campo, se evidenciaron algunos problemas de detección, como la pérdida del carril en curvas estrechas. Sin embargo, el sistema demostró un desempeño aceptable en condiciones reales. A pesar de los resultados positivos, las diferencias entre los entornos controlados del laboratorio y las condiciones del mundo real fueron significativas. Esto indicó la necesidad de continuar refinando los algoritmos y ajustarlos para enfrentar situaciones más complejas, donde factores como la variabilidad en iluminación y el tipo de carretera puedan afectar la precisión del sistema.

En lo referente al hardware y diseño de piezas, el montaje de cámaras en el vehículo presentó varios desafíos. Las piezas fabricadas mediante impresión 3D mostraron vulnerabilidad, ya que sufrieron rupturas bajo presión. Por otro lado, se realizaron ajustes al diseño original de la canastilla, mejorando la distribución de la electrónica y las baterías, lo que aumentó la estabilidad y funcionalidad del sistema, optimizando su desempeño en pruebas de campo.

Algunos aspectos a resaltar:

- Se lograron avances considerables en la implementación de algoritmos de visión por computadora, destacando la detección de colores, bordes de carril y la medición de distancias.
- Las pruebas realizadas permitieron identificar y corregir errores en tiempo real, lo que condujo a una mejora continua y progresiva del rendimiento del sistema.
- La combinación de técnicas tradicionales de visión por computadora con enfoques de

calibración y regresión resultó efectiva, incrementando la precisión tanto en la detección de carriles como en la medición de distancias.

Mientras que las áreas de mejora identificadas son:

- La dependencia del sistema en las condiciones de iluminación sigue siendo un desafío importante. Los cambios en el entorno, como variaciones de luz y color, afectaron significativamente el rendimiento del algoritmo, lo que indica la necesidad de aumentar su adaptabilidad a distintas condiciones.
- La detección de curvas estrechas en carreteras sigue mostrando limitaciones en precisión. El sistema actual aún requiere optimizaciones para mejorar el seguimiento del carril en este tipo de escenarios complejos.
- A pesar de las mejoras en la medición de distancias, aún persiste un margen de error que podría ser crítico para aplicaciones en vehículos autónomos en situaciones reales.

El haber realizado múltiples pruebas permitió que desarrollara un método de programación más limpio y estructurado, facilitando la búsqueda, modificación y reutilización de fragmentos de código. Todas estas pruebas condujeron a la prueba final.

En cuanto a esta última prueba, el enfoque principal fue calibrar y probar el software encargado de la visión por computadora, con especial énfasis en la detección de carriles y el cálculo de distancias mediante el uso de cámaras. Esto permitió validar la funcionalidad del sistema bajo condiciones más cercanas a las del mundo real.

Se calibraron manualmente los parámetros del ancho del carril para asegurar una detección precisa en diversas condiciones, incluidas curvas y diferentes tipos de caminos. Se emplearon múltiples cámaras, como las de teléfonos móviles y cámaras web, para evaluar el desempeño del software tanto en entornos reales como controlados. La detección demostró ser robusta bajo condiciones de buena iluminación y carriles bien definidos; sin embargo, presentó dificultades en caminos estrechos o mal señalizados, lo que sugiere la necesidad de ajustes adicionales para mejorar su adaptabilidad en escenarios más complejos.

El subsistema de estimación de distancia utilizó algoritmos de percepción visual para calcular la distancia entre el vehículo y los obstáculos cercanos. Las pruebas indicaron que los errores en la estimación disminuían cuando el obstáculo se encontraba a mayores distancias,

lo cual fue atribuido a la calibración realizada en ese rango. Se identificó que una mejora significativa podría lograrse mediante la calibración con un mayor número de muestras. Aunque el margen de error aumentaba a medida que el obstáculo se acercaba, los resultados obtenidos fueron satisfactorios, con un rango de error entre el 0.49 % y el 10 %.

Los aspectos a destacar de la última prueba son:

- Calibración exitosa del sistema de detección de caminos: La calibración del sistema permitió que la cámara del vehículo detectara correctamente tanto los carriles pintados como los obstáculos simulados, como las esferas en el camino. Esto demostró la capacidad del software para identificar los elementos del entorno y alinearse con las especificaciones predefinidas.
- Detección de carriles: El sistema tuvo un buen desempeño al identificar líneas y carriles en diversas condiciones, como curvas pronunciadas, carreteras con líneas de grosor definido, e incluso en espacios de estacionamiento. La detección fue precisa, respetando las líneas previamente calibradas, sin importar las variaciones de color o grosor.
- Estimación de distancia: El software también mostró una capacidad destacable para estimar distancias, con márgenes de error relativamente bajos, que variaron entre el 0.49 % y el 10 % según la proximidad del obstáculo. En las pruebas estáticas, el error fue aún menor (1.32 %), lo que indica una alta precisión del sistema en situaciones controladas.

Mientras que las limitaciones encontradas durante la prueba fueron:

- Problemas con carriles estrechos: El sistema presentó dificultades para detectar carriles demasiado angostos o en curvas cerradas. En estos escenarios, el empalme de colores en la detección afectó la precisión del software, lo que indica que la calibración inicial no fue suficiente para cubrir variaciones importantes en el ancho del carril.
- Margen de error en distancias cortas: Aunque el sistema tuvo un buen rendimiento en el rango de calibración (367-452 cm), los porcentajes de error aumentaron significativamente en distancias menores. Esto sugiere que la calibración debería ajustarse para mejorar la precisión en distancias más cortas.
- Dificultades con carriles no convencionales: El software interpretó cualquier línea que coincidiera con el ancho calibrado como un carril, lo que resultó en detecciones incorrec-

tas en lugares donde no había líneas definidas. Esta limitación resalta la necesidad de mejorar la capacidad del sistema para diferenciar entre carriles reales y otros elementos de la vía.

El proyecto logró un progreso notable en la integración de visión por computadora para vehículos autónomos, alcanzando la detección efectiva de carriles, colores y la medición de distancias en la mayoría de los casos. No obstante, aún se necesitan mejoras en los algoritmos y ajustes en el hardware para asegurar un rendimiento óptimo en escenarios del mundo real. Los resultados obtenidos sientan las bases para investigaciones futuras, con un enfoque en incrementar la precisión en situaciones complejas y mejorar la robustez del sistema frente a variaciones en las condiciones ambientales.

El objetivo principal de desarrollar un sistema autónomo para la detección de carriles y estimación de distancias fue cumplido, mostrando un alto grado de precisión en la mayoría de las pruebas. Sin embargo, las limitaciones en la adaptación a cambios en el ancho de los carriles y la estimación de distancias fuera del rango calibrado sugieren que se requieren refinamientos adicionales. Aunque el sistema es eficaz en condiciones controladas, necesita mejorar su capacidad de adaptación para escenarios más desafiantes.

Este trabajo representa un punto de partida prometedor para la creación de sistemas autónomos en vehículos eléctricos, pero exigirá futuras mejoras, sobre todo en la calibración y la habilidad para diferenciar entre carriles reales y otros elementos similares del entorno.

Un punto crucial en este trabajo es el alcance de los métodos convencionales de visión por computadora en aplicaciones autónomas. Aunque se logró con éxito el desarrollo de un sistema autónomo para la detección de carriles, obstáculos y la estimación de distancias con una buena precisión en la mayoría de los escenarios, también se evidenciaron limitaciones inherentes a estos enfoques tradicionales. Como se ha señalado a lo largo del proyecto, los mayores desafíos surgieron en escenarios donde la visión por computadora dependía exclusivamente de calibraciones manuales y enfoques estáticos, lo que redujo su efectividad en entornos más complejos.

La necesidad de realizar calibraciones manuales, enfocarse en un tipo específico de obstáculo y la incapacidad para ajustar dinámicamente el ancho del carril limitaron significativamente el rendimiento en condiciones fuera del rango calibrado. Estos factores restringen el uso del software en aplicaciones autónomas más avanzadas, como en vehículos de nivel 3 o superior, donde se requiere una mayor flexibilidad y adaptabilidad a las variaciones del entorno.

En su estado actual, el software esta muy cerca de posicionar a un vehículo dentro de las características de un nivel 2 de autonomía, pero no más allá.

Una solución a estos desafíos sería la incorporación de redes neuronales convolucionales, las cuales tienen el potencial de mejorar significativamente la capacidad del sistema para adaptarse a distintos tipos de entornos y obstáculos, así como para realizar una calibración automática y dinámica en tiempo real. Esto permitiría que el software fuera más robusto y flexible, y podría facilitar su transición a niveles más avanzados de autonomía.

Un mérito destacado del proyecto radica en el esfuerzo por minimizar el uso de bibliotecas preexistentes para aplicaciones de visión por computadora a excepción de OpenCV, con el fin de adaptar el desarrollo desde su inicio en lugar de intentar ajustar soluciones ya creadas. Esto se refleja en la técnica de visión estereo y percepción visual: mientras que en la primera se emplearon parámetros predefinidos sin obtener un error significativamente bajo, en la segunda se logró una precisión superior al desarrollarse de acuerdo con los alcances del proyecto.

OpenCV, cuenta con una amplia variedad de funciones y parámetros debido a su longevidad, fue empleado en este caso no como una solución completa, sino como una caja de herramientas cuya efectividad depende del conocimiento aplicado en su uso para alcanzar resultados acordes con los alcances del proyecto.

5.1. Trabajo a futuro

Finalmente, se plantean las áreas que requieren ajustes, como la mejora en la manufactura de soportes, también se proponen ideas para el desarrollo futuro del proyecto, como la integración de un sensor LiDAR para mejorar la detección de obstáculos, así como los aspectos necesarios a implementar en el software para poder considerar a un vehículo de nivel dos que lo tenga incorporado.

5.1.1. Prueba de comunicación serial

Se realizaron pruebas para testear la comunicación serial entre Python y un microcontrolador, con el fin de preparar y tener un antecedente para la implementación futura del proyecto. Se espera que, bajo ciertas circunstancias determinadas por el procesamiento de

video, los motores sean capaces de responder de manera adecuada gracias a los datos que hay que recabar como la curvatura del carril, la posición del coche y la distancia.

En la figura 5.2, se dispone del circuito que se implementó para llevar a cabo la prueba, el cual incluye un módulo L298N, un Arduino Uno, una fuente de voltaje de 12V y un motor DC.

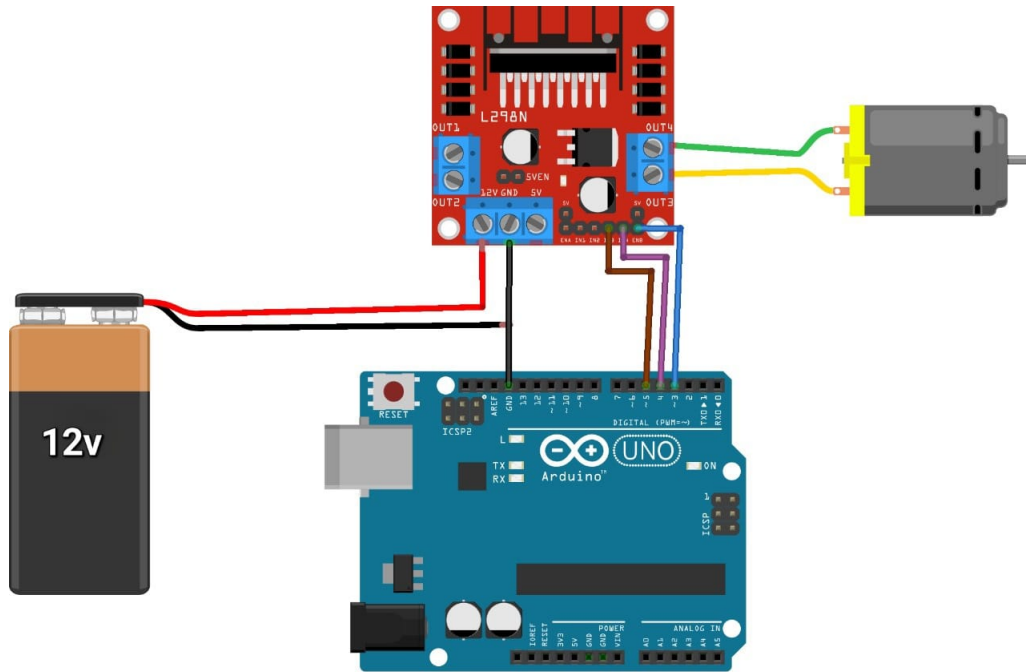


Figura 5.1: Circuito implementado para la prueba

Para esta prueba, el programa desarrollado en Python establece la comunicación serial y a través de la consola, proporciona instrucciones al usuario y solicita la dirección de giro, ya sea en sentido horario o antihorario, así como el porcentaje de velocidad de giro, como se puede observar en la siguiente figura:

```
Instrucciones:
- Ingresa 'CW' o 'CCW' para la dirección.
- Ingresa un valor de 0-100 para el porcentaje de velocidad.

Ingresa la dirección: CW

Ingresa el porcentaje de velocidad: 100
Motor configurado para girar CW al 100% de velocidad.
```

Figura 5.2: Consola del programa

Por otro lado, el programa de Arduino recibe la dirección de giro y la velocidad en forma de una cadena de caracteres para posteriormente, activar el motor con las características enviadas por Python.

A futuro, se plantea utilizar esta prueba como base para la implementación de la electrónica que comunicará el software con el hardware y así poder implementar un vehículo autónomo. Para esto, se tendrán que modificar los programas para que puedan comunicarse entre sí de manera libre y sin intervención.

También se deberá verificar que la trama logre comunicar distancias mayores, ya que esta prueba se realizó a nivel de escritorio, por lo que las conexiones fueron relativamente pequeñas. Una vez puesto en un vehículo, las conexiones tenderán a tener interferencias y ruido, lo cual no es viable para un proyecto de esta magnitud. Por lo tanto, también se plantea implementar electrónica con amplificadores operacionales para las señales.

Así como realizar la adecuada selección de sensores como lo son de proximidad/ultrasónicos, finales de carrera y GPS.

5.1.2. Diseño de soportes

Como se mencionó en el capítulo anterior, el soporte diseñado para mantener la posición de las cámaras sufrió rupturas debido al esfuerzo al que fue sometido, lo que evidenció la necesidad de realizar un rediseño o utilizar materiales más resistentes para futuras pruebas. Además, se considera la posibilidad de cambiar el método de manufactura, ya que la impresión 3D no parece ser la mejor opción para piezas que estarán sometidas a esfuerzos considerables.

5.1.3. Sensor LiDar

Se inició la implementación de un sensor LiDAR, un sistema de detección remota que utiliza rayos láser para medir distancias y movimientos en tiempo real. Este sensor se emplea en la conducción autónoma y en otros campos de la ingeniería debido a sus bajos requisitos de potencia y procesamiento, lo que permite una detección y seguimiento más rápidos [33].

Siguiendo los mismos principios del radar y sonar, es decir, la emisión de ondas de energía para detectar objetos, la diferencia radica en que el LiDAR utiliza luz reflejada, lo que

incrementa la precisión y resolución en comparación con el radar, que emplea microondas, o el sonar, que utiliza ondas de sonido.

El LiDAR emite pulsos de luz láser que viajan a la velocidad de la luz, rebotan en los objetos y regresan al sensor. El sensor mide el tiempo que tarda cada pulso en regresar y calcula la distancia recorrida. Al repetirse este proceso a gran velocidad, se recopilan miles de puntos individuales que se procesan en tiempo real; este conjunto de puntos es conocido como "nube de puntos"[34].

Debido a que el sensor provenía de China y todo el software y la documentación estaban en chino, resultó algo complicado entender su funcionamiento. Sin embargo, tras algo de experimentación y lectura, se logró encender el sensor y obtener un escaneo de un área. Los resultados fueron satisfactorios, obteniendo una nube de puntos y la medición de las distancias.

En el futuro, se plantea utilizar este sensor LiDAR para complementar la estimación de distancia ya propuesta. Para ello, será necesario obtener la trama enviada por el dispositivo y desarrollar un programa que permita leer y aprovechar la información contenida en dicha trama.

Referencias

- [1] E. Munive. “Desarrollo de electrónica empleada en la movilidad de vehículos eléctricos”. Universidad Nacional Autónoma de México, 2020.
- [2] *Shell Eco-marathon*. Acceso: 2024-10-25. URL: <https://www.shell.com.mx/energia-y-innovacion/shell-eco-marathon.html> (visitado 19-03-2023).
- [3] *Vehículo prototipo eléctrico*. Acceso: 2024-10-25. URL: <https://sites.google.com/view/escuderiadzec/que-hacemos/veh%C3%ADculo-prototipo-el%C3%A9ctrico?authuser=0> (visitado 19-03-2024).
- [4] *MICTLÁN, VEHÍCULO ELÉCTRICO TOTALMENTE HECHO EN LA UNAM*. Acceso: 2024-10-25. URL: https://www.dgcs.unam.mx/boletin/bdboletin/2020_227.html (visitado 13-03-2020).
- [5] *Mictlán, primer vehículo eléctrico creado en la UNAM, compite en EE. UU.* Acceso: 2024-10-25. URL: <https://www.gaceta.unam.mx/tag/equipo-dzec/> (visitado 27-03-2023).
- [6] *Shell Eco-marathon terá categoria de hidrogênio pela primeira vez no Brasil*. Acceso: 2024-10-25. URL: <https://www.shell.com.br/imprensa/press-releases-2023/shell-eco-marathon-tera-categoria-de-hidrogenio-pela-primeira-vez-no-brasil.html> (visitado 30-08-2023).
- [7] IIUNAM. *Entrevista DZEC*. Video en línea. Acceso: 2024-10-25. 2024. URL: <https://www.youtube.com/watch?v=6vnGdBZV18k>.
- [8] *OpenCV*. Acceso: 2023-03-06. URL: <https://opencv.org/> (visitado 06-03-2023).
- [9] T. Domínguez. *Visión Artificial: Aplicaciones prácticas con OpenCV Python*. Primera. Marcombo, 2021, pág. 389.
- [10] *Visión por computador: qué es, objetivos y aplicaciones*. Acceso: 2023-03-06. URL: <https://www.edsrobotics.com/blog/vision-computador-que-es/> (visitado 06-03-2023).

- [11] *Numpy*. Acceso: 2023-03-19. URL: <https://web.archive.org/web/20120120095953/http://numpy.scipy.org/> (visitado 19-03-2023).
- [12] V. Teddy, F. Sanz y F. Arteaga. “Sistema de visión por computadora para la detección de objetos esféricos a través de la transformada de Hough”. En: *Revista INGENIERÍA UC* 15 (2008), págs. 77-87. ISSN: 1316-6832. URL: <https://www.redalyc.org/pdf/707/70712315009.pdf>.
- [13] M. Alonso. “Espacios de Color RGB, HSI y sus Generalizaciones a n-Dimensiones”. Instituto Nacional de Astrofísica, Óptica y Electrónica, Agosto de 2009.
- [14] S. Anand y L. Priya. *A Guide for Machine Vision in Quality Control*. Primera. CRC Press, 2020, pág. 179.
- [15] P. García y S. González. “Reconocimiento de color y posición con un sensor Kinect para aplicación de robótica móvil”. Pontificia Universidad Javeriana, 2016.
- [16] G. Molinero. “Segmentación de imágenes en color basada en el crecimiento de regiones”. Universidad de Sevilla, 2010.
- [17] R. Szeliski. *Computer Vision: Algorithms and Applications*. Primera. Springer, 2010, pág. 979.
- [18] R. Gonzalez y R. Woods. *Digital Image Processing*. Segunda. Addison-Wesley Publishing Company, 1993.
- [19] *Image filtering-OpenCV*. Acceso: 2023-03-19. URL: https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html#ga564869aa33e58769b4469101aac458f9 (visitado 19-03-2023).
- [20] *Feature detection-OpenCV*. Acceso: 2023-03-19. URL: https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga04723e007ed88dddf11d9ba04e2232de (visitado 19-03-2023).
- [21] *Moments Class Reference-OpenCV*. Acceso: 2023-03-19. URL: https://docs.opencv.org/4.x/d8/d23/classcv_1_1Moments.html (visitado 19-03-2023).
- [22] *Temas Selectos de Programación: Un enfoque al procesamiento de imágenes y video*. Acceso: 2023-04-05. URL: https://www.ingenieria.unam.mx/dimei/docs/TS20202/TS_Programacion_GBarron_Vis.pdf (visitado 05-04-2023).
- [23] M. Duke. “Integration of stereo vision system calibration and kinematic calibration for an autonomous kiwifruit harvesting system”. En: *International Journal of Intelligent Robotics and Applications* 7 (2023), págs. 350-369. ISSN: 2366-5971. DOI: 10.1007/s41315-022-00263-x.

- [24] *Camera Calibration and 3D Reconstruction-OpenCV*. Acceso: 2023-03-19. URL: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html (visitado 19-03-2023).
- [25] Akhil Valsaraj et al. “Stereo Vision System Implemented on FPGA”. En: *Procedia Technology* 24 (2016), págs. 1105-1112. DOI: 10.1016/j.protcy.2016.05.243.
- [26] *How to Use the Sliding Window Technique – Algorithm Example and Solution*. Acceso: 2024-02-02. URL: <https://www.freecodecamp.org/news/sliding-window-technique/#:~:text=The%20sliding%20window%20technique%20is,in%20a%20step%2Dwise%20fashion> (visitado 02-02-2024).
- [27] *¿Qué es un coche autónomo?* Acceso: 2023-03-06. URL: <https://www.mapfre.com.mx/particulares/seguros-auto/noticias/que-es-coche-autonomo/> (visitado 06-03-2023).
- [28] R. Riquelme. “Vehículos autónomos requieren talento para llegar a México: Lyft”. En: *El Economista* (2022).
- [29] *Autopilot*. Acceso: 2024-10-25. URL: https://www.tesla.com/es_MX/autopilot.
- [30] *Coches autónomos mexicanos: una realidad en el futuro cercano*. Acceso: 2023-03-06. URL: <https://transferencia.tec.mx/2019/12/09/coches-autonomos-mexicanos-una-realidad-en-el-futuro-cercano/> (visitado 06-03-2023).
- [31] *En el marco del 80 aniversario del Tec, VantTec presentó el vehículo autónomo en el campus Monterrey*. Acceso: 2024-06-25. URL: <https://conecta.tec.mx/es/noticias/monterrey/educacion/hacia-la-innovacion-jovenes-mexicanos-desarrollan-vehiculo-autonomo> (visitado 25-06-2024).
- [32] *El proyecto AMR Tec es un vehículo eléctrico autónomo que busca cambiar el rumbo de la movilidad humana e industrial*. Acceso: 2024-06-25. URL: <https://conecta.tec.mx/es/noticias/puebla/educacion/revolucion-de-vehiculos-alumnos-y-profesores-crean-carro-autonomo> (visitado 25-06-2024).
- [33] F. Farzadpour. “Modeling Solid-State LiDAR Sensor for Optimization of Area Coverage Deployment”. En: *ASME International Conference on Advanced Intelligent Mechatronics (AIM)* (2023), págs. 1182-7. DOI: 10.1109/AIM46323.2023.10196227.
- [34] *¿Qué es LiDAR?* Acceso: 2024-02-02. URL: <https://www.ibm.com/mx-es/topics/lidar> (visitado 02-02-2024).