



**UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES**

TITULO DEL TRABAJO DE TITULACIÓN

**ANÁLISIS DE ALGORITMOS DE VISIÓN ARTIFICIAL PARA EL
CONTROL DE ESPACIOS OCUPADOS EN EL PARQUEADERO
CENTRAL DE LA UPSE**

AUTOR

LEÓN RIVERA HELEN JAMEL

MODALIDAD DE TITULACIÓN

PROYECTO UIC

**Previo a la obtención del grado académico en
INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN**

TUTOR

Ing. Shendry Rosero Vásquez, Ms.CC

Santa Elena, Ecuador

Año 2024



**UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES**

TRIBUNAL DE SUSTENTACIÓN

Ing. José Sánchez Aquino, Mgt.
DIRECTOR DE LA CARRERA

Ing. Shendry Rosero Vásquez, Msc.
TUTOR

Ing. Carlos Castillo Yagual, Mgt
DOCENTE ESPECIALISTA

Ing. Marjorie Coronel Suárez, Mgt
DOCENTE GUÍA UIC



**UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES**

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por León Rivera Helen Jamel, como requerimiento para la obtención del título de Ingeniero en Tecnologías de la Información.

La Libertad, a los 17 días del mes de junio del año 2024

TUTOR

Shendry
Rosero
o V.

Ing. Shendry Rosero Vásquez. Mgt.



**UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES**

DECLARACIÓN DE RESPONSABILIDAD

Yo, León Rivera Helen Jamel

DECLARO QUE:

El trabajo de Titulación, “Análisis De Algoritmos De Visión Artificial Para El Control De Espacios Ocupados En El Parqueadero Central De La Upse” previo a la obtención del título en Ingeniero en Tecnologías de la Información, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

La Libertad, a los 17 días del mes de junio del año 2024

EL AUTOR

Helen León R.

León Rivera Helen Jamel



**UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES**

AUTORIZACIÓN

Yo, León Rivera Helen Jamel

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales del presente trabajo de titulación con fines de difusión pública, además apruebo la reproducción de este artículo académico dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor.

Santa Elena, a los 17 días del mes de junio del año 2021

EL AUTOR

Helen León R.

León Rivera Helen Jamel

AGRADECIMIENTO

Quiero expresar mi profundo agradecimiento a todas aquellas personas que hicieron lo posible ayudarme a cumplir un objetivo más.

En primer lugar, agradezco a Dios por darme la fortaleza y sabiduría necesarias para cumplir con este objetivo más en la vida.

A mi familia, especialmente a mis padres, por su apoyo incondicional, por su amor, apoyo y sacrificio que ha sido mi mayor pilar a lo largo de mi formación y mi fuente de motivación.

A la Facultad de Sistemas y Telecomunicación en especial a la carrera de Tecnología de la Información por permitirme formar de manera profesional.

El respeto y agradecimiento a mi tutor Ing. Rosales Vasquez Shendry,Mgt, por haber compartido sus conocimientos, consejos y experiencias para mi crecimiento personal y profesional.

También agradezco a mi docente guía Ing.Coronel Marjorie.Mgt, por haber compartido sus conocimientos, consejos y experiencias para mi crecimiento personal y profesional.

Helen Jamel León Rivera

DEDICATORIA

A mis padres, Oswaldo Raúl León Reyes, Jacqueline Maritza Rivera Mojarrango. Por ser mi ejemplo de perseverancia, sacrificio y amor incondicional. A ellos, que siempre creyeron en mí más de lo que yo mismo lo hice.

A mis hermanos y hermana,

Por ser mi fuente de alegría, complicidad y motivación. Cada conversación, momentos de risas compartidas y palabras de aliento que ha sido crucial para mantenerme enfocada en este camino académico.

A mi querida abuela Petra Reyes Cruz, por su amor infinito y tu constante apoyo ha sido mi luz y mi guía a lo largo de toda mi vida. Aunque ya no este físicamente conmigo, sé que sigues siendo mi ángel de la guarda, guiándome desde el cielo.

Recuerdo cada momento compartido contigo, cada palabra de aliento y cada sonrisa que me brindaste. Tus consejos sabios y tu cariño incondicional ha sido el motor que me impulso a perseverar, incluso en los momentos más difíciles. Cada página es un tributo a tu memoria y a la invaluable influencia que tu presencia tuvo en mi vida académica y personal.

Gracias por su apoyo, colaboración y por ser parte de este camino de aprendizaje y crecimiento profesional.

Helen Jamel León Rivera

ÍNDICE GENERAL

TRIBUNAL DE SUSTENTACIÓN	II
CERTIFICACIÓN	III
DECLARACIÓN DE RESPONSABILIDAD	IV
DECLARO QUE:	IV
AUTORIZACIÓN	V
AGRADECIMIENTO	V
DEDICATORIA	VII
ÍNDICE DE TABLAS	XI
ÍNDICE DE FIGURAS	XII
RESUMEN	XVII
ABSTRACT	XVIII
INTRODUCCIÓN	1
CAPITULO 1. FUNDAMENTACIÓN	3
1.1 ANTECEDENTES	3
1.2 DESCRIPCIÓN DEL PROYECTO	5
1.3 OBJETIVOS DEL PROYECTO	8
1.4 JUSTIFICACIÓN DEL PROYECTO	9
1.5 ALCANCE DEL PROYECTO	10
1.6 METODOLOGÍA DEL PROYECTO	11
1.6.1 METODOLOGÍA DE INVESTIGACIÓN	11
1.6.2 BENEFICIARIOS DEL PROYECTO	12
1.6.3 VARIABLE	12
1.6.4 ANÁLISIS DE RECOLECCIÓN DE DATOS	13
1.6.5 ANÁLISIS DE RESULTADO DE ENCUESTA	16

1.7 METODOLOGÍA DE DESARROLLO	21
CAPÍTULO 2. PROPUESTA	22
2.1 MARCO CONTEXTUAL	22
2.1.1 UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA	22
2.1.2 MISIÓN	23
2.1.3 VISIÓN	23
2.2 MARCO CONCEPTUAL	24
2.2.1 VISIÓN ARTIFICIAL	24
2.2.1.1 INTELIGENCIA ARTIFICIAL	24
2.2.1.2 REDES NEURONALES	24
2.2.1.3 APRENDIZAJE AUTOMÁTICO	25
2.2.1.4 APRENDIZAJE PROFUNDO	26
2.2.2 ALGORITMO DE DETECCIÓN DE OBJETOS	27
2.2.2.1 MODELO YOLO (YOU ONLY LOOK ONCE)	27
2.2.2.2 MODELOS PREENTRENADOS	28
2.2.2.3 OPTIMIZACIÓN DE ALGORITMOS	28
2.2.2.4 VALIDACIÓN Y EVALUACIÓN DE MODELOS	29
2.2.2.5 INTEGRACIÓN DE SISTEMAS	29
2.2.2.6 INTERFAZ DE USUARIO	29
2.2.2.7 SEGURIDAD Y PRIVACIDAD	30
2.2.2.8 ESCALABILIDAD Y EL MANTENIMIENTO	30
2.2.2.9 TENDENCIAS FUTURAS	30
2.2.3 HERRAMIENTAS Y BIBLIOTECAS DE DESARROLLO	31
2.2.3.1 PYTHON	31
2.2.3.2 OPENCV	31

2.2.3.3 PYTORCH	31
2.2.3.4 NUMPY	32
2.2.3.5 ROBOFLOW	32
2.2.3.6 VISUAL BASIC	32
2.3 MARCO TEÓRICO	33
2.3.1 ALGORITMOS DE DETECCIÓN DE VEHÍCULOS EN ESTACIONAMIENTOS	33
2.3.2 TÉCNICAS PARA IDENTIFICAR AUTOMÓVILES EN ÁREAS DE APARCAMIENTO	34
2.3.3 MÉTODOS DE RECONOCIMIENTO DE COCHES EN ESPACIOS DE ESTACIONAMIENTO	35
2.4 REQUERIMIENTOS	36
2.4.1 REQUERIMIENTOS FUNCIONALES	36
2.4.2 REQUERIMIENTOS NO FUNCIONALES	38
2.5 COMPONENTE DE LA PROPUESTA	39
2.5.1 ARQUITECTURA DEL SISTEMA	39
2.5.1.1 ALGORITMO CON HERRAMIENTAS DE IOT	39
2.5.2 DIAGRAMAS DE CASOS DE USO	55
2.5.3 MODELADO DE DATOS	62
2.4 DISEÑO DE INTERFAZ	65
2.6 PRUEBAS	69
2.7 RESULTADOS	70
CONCLUSIONES	75
RECOMENDACIONES	76
REFERENCIAS	77
ANEXOS	82

ÍNDICE DE TABLAS

Tabla 1. Información del método de observación	14
Tabla 2. Información de la población	15
Tabla 3. Requerimientos funcionales.	37
Tabla 4. Requerimientos no funcionales.	39
Tabla 5: Comparación de Arduino	40
Tabla 6: Comparación de cámaras	41
Tabla 7: Conexión del Arduino	42
Tabla 8: Cuadro Descriptivos de algoritmos de Análisis para detección de objetos – Visión Artificial	49
Tabla 9: Cuadro Descriptivo – Algoritmo de detección de objetos para Estacionamiento	54
Tabla 10: Caso de Uso: Detectar Vehículos	55
Tabla 11: Caso de Uso: Registrar Espacio Ocupado	56
Tabla 12: Caso de Uso: Registrar Espacio Libre	58
Tabla 13: Caso de Uso: Consultar Disponibilidad de Espacios	60
Tabla 14: Caso de Uso: Actualizar Información de Espacios	61
Tabla 15: Resultados de los videos de prueba de rendimiento	98
Tabla 16: Resultados del Video original en el algoritmo de Clasificación de Estacionamientos	100
Tabla 17: Resultados de tiempo de rendimiento	109

ÍNDICE DE FIGURAS

Figura 1. Modelo de la arquitectura	6
Figura 2. Frecuencia de parqueo.	16
Figura 3. Tiempo en encontrar parqueadero.	17
Figura 4. Afectación en las actividades.	18
Figura 5. Frecuencia de congestiónamiento vehicular.	18
Figura 6. Frecuencia de ingreso y salida de la institución.	19
Figura 7. Calificación de la organización actual del parqueadero.	19
Figura 8. Inconvenientes para salir del parqueo.	20
Figura 9. Sistema que muestre disponibilidad en el parqueadero.	20
Figura 10. Metodología de desarrollo del proyecto.	21
Figura 11. Ubicación de la UPSE.	22
Figura 12. Estacionamiento de la UPSE.	23
Figura 13. Plano del estacionamiento central de la UPSE.	23
Figura 14. Procesamiento De Imágenes Y La Detección De Objetos.	25
Figura 15: Pantalla LCD	43
Figura 16: Modelado de base de datos	43
Figura 25: Proceso de sistema de detección	44
Figura 26: Identificadores de parqueos	44
Figura 27: Detección de ocupación en el parqueadero	45
Figura 28: Diagrama de Caso de Uso: Detectar Vehículos	56
Figura 29: Diagrama de Caso de Uso: Registrar Espacio Ocupado	57
Figura 30: Diagrama Caso de Uso: Registrar Espacio Libre	59
Figura 31: Diagrama de Caso de Uso: Consultar Disponibilidad de Espacios	60

Figura 32: Diagrama de Caso de Uso: Actualizar Información de Espacios	62
Figura 33: Diagrama de Entidad-Relación	64
Figura 34: Captura de interfaz	68
Figura 35: Algoritmo de estacionamiento	69
Figura 36: Resultado de Visualización de Coordenadas	70
Figura 37: Resultado de Precisión de la Detección	71
Figura 38: Resultado de Archivos Generados	71
Figura 39: Resultado de Estabilidad del Seguimiento	72
Figura 40: Resultado botón "Descargar Reporte"	72
Figura 41: Resultado botón "Ver Reporte"	73
Figura 42: Resultado de prueba 1	73
Figura 43: Resultado de prueba 2	74
Figura 44: Resultado de prueba 3	74
Figura 45: Capturar de imágenes positivas y negativas carpeta "P"	85
Figura 46: Capturar de imágenes negativas carpeta "P"	85
Figura 47: Almacenamiento de imagenes	86
Figura 48: Herramienta Cascade Trainer GUI	86
Figura 49: Configuración en Cascade Trainer GUI	87
Figura 50: Configuración de tamaño de img y modo en HAAR	87
Figura 51: Procesamiento de entrenamiento	88
Figura 52: Entrenamiento finalizado	88
Figura 53: carpeta classifier	89
Figura 54: Prueba del script con herramienta IOT	89
Figura 55: Tamaño de Espacio y Ruta de archivos	93

Figura 56: Ejemplo de Marcadores o Etiquetas	94
Figura 57: Prueba de cambio de iluminación	96
Figura 58: Prueba de cambio brusco y presencia de sombras	97
Figura 59: Línea de código para cálculo de rendimiento	98
Figura 60: Resultados de rendimiento	98
Figura 61: Configuración de parámetros	103
Figura 62: Creación del Tracker	103
Figura 63: Iniciar la captura de video	103
Figura 64: Bucle Principal	104
Figura 65: Prueba 1 de Funcionalidad	105
Figura 66: Prueba 2 de Funcionalidad	106
Figura 67: Prueba 3 de Funcionalidad	106
Figura 68: Pruebas de Robustez algoritmo	107
Figura 69: Prueba de Rendimiento	110
Figura 70: Creación de entorno Python	111
Figura 71: Creación de nota de codificación	111
Figura 72: Creación de carpeta de imágenes de prueba para el entrenamiento	112
Figura 73: Desarrollo de script para el modelado	112
Figura 74: Importación de bibliotecas para procesamiento de imágenes	113
Figura 75: Definir los directorios de categorías de imágenes	114
Figura 76: Definición de directorios de salida	114
Figura 77: Creación de directorios de salida	114
Figura 78: Función de Preprocesamiento de imágenes	115
Figura 79: Procesamiento y División de imágenes	115

Figura 80: Guardado de Imágenes Preprocesadas	116
Figura 81: Preprocesado de imagenes	116
Figura 82: Mensaje de salida del script	117
Figura 83: Importaciones del Script Carga y Aumento de datos	117
Figura 84: Directorios de imágenes preprocesadas	117
Figura 85: Creación de generadores de datos	117
Figura 86: Cargar datos de entrenamientos	118
Figura 87: Mensaje de salida del script aumento de datos	119
Figura 88: Importaciones del script de entrenamiento del modelo	119
Figura 89: Definir la arquitectura del modelo	119
Figura 90: Compilar el modelo	120
Figura 91: Entrenar modelo	120
Figura 92: Mensaje de salida Script entrenamiento	120
Figura 93: Evaluar el modelo en el conjunto de prueba	121
Figura 94: Guardar el modelo entrenado	121
Figura 95: Importaciones del script cargar y usar modelo guardado	122
Figura 96: Cargar el modelo guardado	122
Figura 97: Preprocesar una nueva imagen	122
Figura 98: Función para detectar bordes en la Imagen	123
Figura 99: Detectar líneas usando Hough	123
Figura 100: Función – Dibujar Líneas detectadas en la imagen	124
Figura 101: Función para contar espacios disponibles	124
Figura 102: Hacer una predicción en una nueva imagen	125
Figura 103: Detectar Bordos y Lineas	125

Figura 104: Dibujar Líneas y Contar espacios	125
Figura 105:Mostrar Imágenes	126
Figura 106: Procesamiento completado – script 1	126
Figura 107: Cargar Imágenes y aumento de datos completado	127
Figura 108: Entrenamiento del modelo	127
Figura 109: Modelado entrenado y guardado	127
Figura 110: Resultado del análisis de la imagen de prueba	127
Figura 111: Imagen nuevo con el análisis – resultado final	128

RESUMEN

El proyecto aborda el análisis de algoritmos de visión artificial para controlar los espacios ocupados en el parqueadero central de la UPSE. El objetivo principal es implementar un prototipo de sistema de gestión de estacionamiento que utilice técnicas de visión por computadora para identificar espacios vacíos y distribuir esta información a conductores, optimizando así el tiempo de búsqueda de estacionamiento. Se evaluaron diferentes algoritmos de visión por computadora y se desarrolló un algoritmo de detección de espacios vacíos en Python, con una precisión objetivo del 80%. Las pruebas experimentales validaron la efectividad y precisión del sistema, ajustando el algoritmo según sea necesario. La metodología incluyó investigación exploratoria y diagnóstica, así como un enfoque incremental para el desarrollo del proyecto. Los resultados mostraron una alta precisión en la detección y estabilidad en el seguimiento de vehículos, con funcionalidades útiles para generar informes en tiempo real. Estos hallazgos demuestran la viabilidad práctica y la relevancia de los algoritmos desarrollados para mejorar la eficiencia en la gestión de estacionamientos.

Palabras claves: Visión artificial, Gestión de estacionamiento, Algoritmos de detección.

ABSTRACT

The project addresses the analysis of computer vision algorithms to control occupied spaces in the central parking lot of the UPSE. The main objective is to implement a parking management system that uses computer vision techniques to identify empty spaces and distribute this information to drivers, thus optimizing parking search time. Different computer vision algorithms were evaluated and an empty space detection algorithm was developed in Python, with a target accuracy of 80%. Experimental tests validated the effectiveness and accuracy of the system, adjusting the algorithm as needed. The methodology included exploratory and diagnostic research, as well as an incremental approach to project development. The results showed high detection accuracy and stability in vehicle tracking, with useful functionalities for generating real-time reports. These findings demonstrate the practical feasibility and relevance of the developed algorithms for improving parking management efficiency.

Keywords: Artificial vision, Parking management, Detection algorithms.

INTRODUCCIÓN

El presente trabajo aborda el diseño e implementación de un prototipo de sistema de control de espacios para evaluar la disponibilidad vehicular en el parqueadero central de la Universidad Estatal Península de Santa Elena (UPSE). Esta iniciativa surge con el propósito de optimizar el proceso de monitoreo y gestión de espacios de estacionamiento, permitiendo una mejor organización y eficiencia en el uso de este recurso vital dentro del campus universitario.

El proyecto se estructura en torno a dos componentes principales: hardware y software. La primera parte se enfoca en la implementación de una arquitectura tecnológica que consta de cinco etapas, desde la supervisión del entorno mediante dispositivos hasta el procesamiento de datos para su posterior visualización. Por otro lado, la segunda parte del proyecto se centra en el desarrollo de un algoritmo de detección de espacios vacíos, el cual se integra con una interfaz de usuario para proporcionar información en tiempo real sobre la disponibilidad de lugares de estacionamiento.

El objetivo general del proyecto es implementar un prototipo de sistema de gestión de espacios de estacionamiento basado en técnicas de visión por computador, con el fin de identificar espacios vacíos y distribuir esta información a los conductores, contribuyendo así a la optimización del tiempo de búsqueda de estacionamiento. Para lograr este objetivo, se plantean los siguientes objetivos específicos:

- Evaluar los principales algoritmos de visión por computadora para la selección de una metodología adecuada.
- Desarrollar un algoritmo de detección de espacios vacíos con una precisión igual o superior al 80%.
- Realizar pruebas experimentales para validar la efectividad y precisión del sistema desarrollado.

La justificación de este proyecto radica en la creciente importancia de la tecnología de visión artificial en la optimización de procesos y la mejora de la calidad de vida en diversos ámbitos. En el contexto universitario, la implementación de un prototipo de este sistema ofrecerá beneficios significativos, como la reducción del tiempo de búsqueda de estacionamiento y la mejora en la eficiencia operativa del parqueadero central de la UPSE.

El alcance del proyecto incluye la implementación de un prototipo sistema de visión artificial que cubrirá el espacio del parqueadero, utilizando cámaras y dispositivos conectados a una base de datos para recopilar y procesar la información sobre la disponibilidad de espacios. Además, se desarrollará una interfaz de usuario que permitirá a los conductores visualizar en tiempo real la disponibilidad de estacionamiento.

El proyecto se enmarca en la búsqueda de soluciones tecnológicas innovadoras para mejorar la gestión de espacios de estacionamiento en el campus universitario, contribuyendo así al objetivo más amplio de garantizar la seguridad y eficiencia en el transporte terrestre y promover un ambiente seguro en la comunidad universitaria.

CAPITULO 1. FUNDAMENTACIÓN

1.1 ANTECEDENTES

En la actualidad, el aumento de la congestión vehicular se ha convertido en una de las principales preocupaciones urbanas debido a diversos factores, entre ellos, la escasez de espacios de estacionamiento disponibles. Esta problemática se agrava en grandes zonas urbanas donde la cantidad de vehículos que circulan diariamente es significativa. Desde las primeras horas del día, los conductores se enfrentan a la dificultad de encontrar lugares para estacionar, lo que repercute en retrasos significativos al momento de llegar a sus destinos, ya sea para asistir a estudios, trabajar o llevar a cabo sus actividades cotidianas [1].

En primera instancia, se realizó una encuesta a los docentes de la Facultad de Sistemas y Telecomunicaciones de la UPSE (**Ver Anexo 1. Encuesta dirigida a la comunidad universitaria que posee vehículos en la Universidad Estatal Península de Santa Elena**), determinando que, ocasionalmente encuentran parqueadero para su vehículo, demorando en algunos casos cinco minutos y en otros, de seis a diez minutos, además, creen que el tiempo que les toma encontrar parqueo, afecta en sus actividades laborales, indicando también que, frecuentemente se encuentran con congestionamiento vehicular al momento de buscar parqueo, ya que, ingresan y salen de dos a tres veces en el día a la institución.

La Universidad Estatal Península de Santa Elena posee diversos parqueaderos en toda la institución, en donde mediante observación (**Ver**

Anexo 2. Observación realizada en el parqueadero central de la Universidad Estatal Península de Santa Elena) se manifiesta como ciertos lugares no cuentan con debida ocupación de regulación como es tal el caso del parqueadero central ubicadas en las oficinas de Tecnologías de la información y Comunicación (TICS) dando carencias en lo que viene hacer el control de espacios disponibles y la ocupación existente permitiendo a los docentes o estudiantes requieran ir a otro lugar de estacionamiento y tener la tedioso tarea caminar al estar lejos de su lugar de destino.

Así mismo, califican la organización actual en el parqueadero central de la UPSE como regular, debido que existen inconvenientes como; vehículos mal parqueados, tráfico

vehicular, ocupación preferencial, entre otros. Provocando un manejo tan extenso en lo que viene hacer el control manual como otras entidades presentes y que no es presenciado en la Universidad debido a la carga de trabajo y personal encargado.

Por otro lado, se elaboró un método de observación en el lugar antes mencionado ([Ver Anexo 2. Observación realizada en el parqueadero central de la Universidad Estatal Península de Santa Elena](#)), donde se recabaron los datos más importantes en relación al problema planteado, describiendo que, en ocasiones, las personas ingresan por la mañana al parqueadero, dando varias vueltas sin encontrar lugares disponibles, así mismo, al no encontrar parqueo, estacionan en cualquier lugar, muchas veces impidiendo el paso de otros vehículos ya parqueados, o parquean el vehículo al ras de la calle, cerca del lugar donde les toca ver o impartir clases, también, se estacionan en lugares inseguros donde no pueden vigilar sus vehículos; además, se pudo visualizar que, el inconveniente de no encontrar parqueadero, dificulta la hora de entrada de los docentes y personal administrativo y finalmente, se puede ver que existe tráfico vehicular en los lugares de parqueo, ya que, existe una administración regular en los mismos.

A nivel mundial, en Lima – Perú, se realizó la tesis “Implementación de un sistema web basado en visión artificial y geolocalización, y su influencia en la vigilancia del Distrito de Lince”, la cual tiene como objetivo determinar la influencia existente al implementar un aplicativo web que se basa en la visión artificial y geolocalización y su influencia en la vigilancia de las áreas vecinales del distrito de Lince [2]. Para lograr dicho objetivo, se utilizaron herramientas innovadoras, tecnológicas y algoritmos potentes, que funcionan mediante servicios web brindados por empresas como Microsoft y Google, empleando reconocimiento de rostros, de caracteres y de objetos [2]. La investigación concluye en que la implementación del sistema no influye positivamente, ya que, también es necesaria la adquisición de equipos conectados al aplicativo para obtener una mayor relevancia. [2]

Respecto a este tema, en Ambato – Ecuador, el Ing. Jovann Pérez elaboró el trabajo de investigación titulado “Análisis de tráfico vehicular mediante visión artificial”, proponiendo el desarrollo de un programa orientado al análisis de tráfico con la modalidad fuera de línea; el cual divide el análisis de tráfico en dos inconvenientes principales: seguimiento de vehículos y conteo de los mismos para la obtención de flujos, empleando algoritmos especializados, ofreciendo la mayor velocidad de procesamiento y un mayor

porcentaje de precisión [3]. Con esto, se detalla el procedimiento empleado, obteniendo estimaciones de la distancia y velocidad registrada, cada una con sus reportes respectivos; concluyendo así, con pruebas de validación y verificación de los resultados, la comprobación de un rendimiento equilibrado y satisfactorio [3].

A nivel local, en La Libertad – Ecuador, se realizó la “Propuesta de mejora al acceso de vehículos autorizados y no autorizados mediante el reconocimiento de placas, tratamiento de imagen y automatización al edificio El Velero Azul”, planteando que, con el transcurso del tiempo la utilización de diferentes técnicas de visión artificial ha logrado la obtención de reconocimientos de objetos diversos, pasando por un conjunto de procesos que permiten su modificación y modelación [4]. En el proyecto se analizan los factores que influyen en el acceso de vehículos para la zona de parqueo, evaluando las debilidades de dicho proceso y aplicando técnicas de visión artificial junto a un reconocimiento óptico de caracteres a las placas de los autos que requieren ingresar, teniendo un control de acceso al servicio del estacionamiento [4].

Mencionando diversos trabajos de varias fuentes, se plantea el presente proyecto, en el cual se realizará un análisis de diferentes algoritmos de visión artificial frente a las métricas en el parqueadero central de la UPSE para controlar los espacios ocupados de parqueos, solventando los inconvenientes presentes con respecto a este tema. También se mencionaría que si es necesario un sistema que muestre en cada espacio del parqueadero, la disponibilidad de parqueo mediante una pantalla.

1.2 DESCRIPCIÓN DEL PROYECTO

El presente proyecto propone el diseño de un sistema de control de espacios para comprobar la disponibilidad vehicular en el parqueadero central de la Universidad Estatal Península de Santa Elena “UPSE” en el transcurso de las labores diarias, para así mejorar el proceso de controlar la cantidad de vehículos que se encuentran ocupando un sitio del estacionamiento y mostrando la capacidad de espacios libres. Toda la información obtenida se almacenará, que posteriormente, se administrará de la mejor manera posible, antes de que se muestren los resultados finales.

El trabajo se dividirá en dos partes, la primera parte se compone por el hardware, empleando el modelo de arquitectura tecnológica [5], el cual constará de cuatro etapas, como se muestra en la figura 1, la segunda parte comprenderá completamente el apartado del Software, aquí se proyectará en la pantalla la cantidad disponible de espacios, utilizando un enfoque de metodología en cascada. Dado que ambas partes están en constante comunicación, se necesitan enfoques compatibles para desarrollar un proyecto estable y confiable.

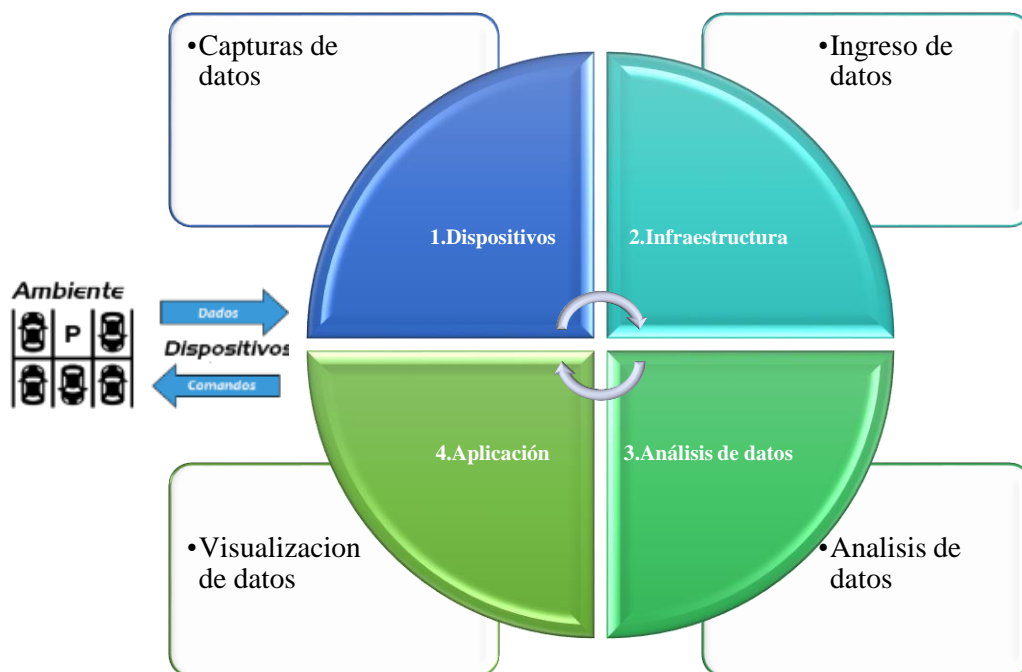


Figura 1. Modelo de la arquitectura

Etapa 1 (Dispositivos): Son los componentes básicos de las soluciones, supervisan el entorno de la solución y recopilan información para su procesamiento mediante reconocimiento, para enviar datos a ser procesados, se reciben comandos para actuar sobre el entorno y son un elemento esencial de las soluciones, ya que, requieren un equilibrio entre la complejidad técnica, robustez y costo.

Etapa 2 (Infraestructura): Permite el almacenamiento de los datos en procesamiento, recopilados por el dispositivo. Se realiza el proceso visual que genera la cámara para medir los espacios ocupados y determina la cantidad de entradas vehiculares disponibles en el parqueadero.

Etapa 3 (Análisis de datos): Es el desarrollo de algoritmos y aplicaciones de análisis de datos, dependiendo de la información recopilada de los dispositivos (tipo, volumen, frecuencia, variedad, video) y los requisitos de experiencia del usuario, identificados en la fase del proyecto.

Etapa 4 (Aplicación): Comprende los módulos internos para los procesos de acuerdo con las especificaciones métricas que se determinarán en la visión artificial de la cámara, de igual manera, abarca la parte de los reportes en donde se visualizarán las entradas necesarias de los registros diarios vehiculares, tales como: fechas, horas y cantidades de automotores.

La tecnología que se implementará estará alojada en un lugar específico del parqueadero central, teniendo en cuenta las métricas de exactitud, sensibilidad y precisión del dispositivo, que funcionará como una visión del área, el cual mediante programación en Python determinará la medida del parqueadero entre un punto inicial y otro final, de igual manera, se mantendrá distanciamiento entre los espacios determinados de cada vehículo para el monitoreo de la disponibilidad, siendo procesada esta información a un dispositivo de visualización.

La visualización de los datos obtenidos en el proceso se proyectará en una pantalla situada en la entrada del estacionamiento, con el fin de indicar al conductor si el parqueadero posee disponibilidad vehicular, evitando la entrada innecesaria del conductor en búsqueda de espacio para su automóvil. La base de datos alojará todos los datos recolectados con su respectiva fecha y cantidad de entradas, junto a la hora que se ocupó un espacio, para luego ser visualizada la información por el administrador mediante reportes.

El proyecto será diseñado utilizando las siguientes herramientas informáticas:

Python: Es un lenguaje de programación poderoso y fácil de aprender. Tiene estructuras de datos de alto nivel, eficaz y una orientación simple pero práctica para la programación orientada a objetos; la sintaxis distinguida y la tipificación dinámica de Python, junto con su naturaleza interpretada, la transforma en un lenguaje ideal para secuencias de comandos y desarrollo rápido de aplicaciones en muchas áreas en la mayoría de las plataformas. [6]

PHP (Abreviatura recursiva de PHP: Preprocesador de hipertexto): Es un lenguaje de código abierto muy popular, especialmente adecuado para el desarrollo web que se puede incrustar en HTML; la diferencia entre PHP y el lado del cliente (como Javascript) es que el código se ejecuta en el servidor, mientras que, en HTML se genera y se envía al cliente, recibiendo los resultados de la ejecución del script incluso si se desconoce el código subyacente. [7]

Visual Studio Code: Es un editor de código fuente liviano pero potente que se elabora en el escritorio para Windows, macOS y Linux, teniendo un soporte nativo para JavaScript, TypeScript y Node.js, adecuándose a un sin número de extensiones para otros lenguajes y tiempos de interpretación (C, C#, Java, Python, PHP, Go, .NET) [8].

El presente proyecto contiene tecnología para el desarrollo y ambientación en el área del parqueadero central mediante métricas de visión artificial, el cual está relacionado con el dominio de Tecnología, infraestructura y sistemas constructivos, siguiendo la línea de investigación de Desarrollo de Software (DSS), orientado a la Sub - línea de Desarrollo de algoritmos y visión artificial.

1.3 OBJETIVOS DEL PROYECTO

OBJETIVO GENERAL

Implementar un prototipo de sistema de gestión de espacios de estacionamiento basado en técnicas de visión por computador para el parqueadero central de la UPSE, que permita la identificación de espacios vacíos y distribución de esta información a conductores, contribuyendo a la optimización del tiempo de búsqueda de estacionamiento.

OBJETIVOS ESPECÍFICOS

- Evaluar los principales algoritmos de visión por computadora, utilizando revisiones bibliográficas actualizadas, para la selección de una metodología adecuada para el desarrollo del sistema de detección de espacios vacíos en estacionamientos.
- Desarrollar un algoritmo de detección de espacios vacíos, implementado en Python, que busque alcanzar una precisión igual o superior al 80% en la

identificación y detección de objetos, mediante la iteración y mejora continua del modelo basado en pruebas experimentales.

- Realizar pruebas experimentales en el parqueadero central de la UPSE, con el objetivo de validar la efectividad y precisión del sistema desarrollado, ajustando el algoritmo según sea necesario para cumplir con los criterios de rendimiento establecidos.

1.4 JUSTIFICACIÓN DEL PROYECTO

En la actualidad, el desarrollo tecnológico está trascendiendo a un entorno innovador, adentrándose en varios campos del uso de la inteligencia artificial en ámbitos especializados, brindando calidad de vida mediante la automatización de procesos diarios, incidiendo en ambientes como la ciencia, empresas e industrias, hasta el punto de que las máquinas aprendan por si solas, en sus procesos, errores e información que dispongan mediante algoritmos [9].

La visión artificial en la actualidad ha favorecido significativamente a varias áreas, a través del uso de algoritmos para determinadas tareas en base a imágenes, comparación en una base de datos de distanciamiento entre áreas y vinculadas a dispositivos como celulares, tabletas o computadores; en el ámbito automotriz y de tránsito, la utilidad de la IOT de visualización de imágenes se ha convertido en una herramienta indispensable en países con nivel avanzado, ayudando al reconocimiento de vehículos y captura de imágenes pequeñas como placas de los mismos; En empresas se están integrando estas tecnologías en los espacios de parqueaderos para establecer multas a personas que no respetan los espacios privados de cada empleador [10].

Evidenciando la necesidad que surge en la comunidad universitaria, en sus jornadas diarias, cada mañana se genera una de las causas que produce incomodidad a los conductores en los parqueaderos de la Universidad Estatal Península de Santa Elena (UPSE) al buscar un espacio para su vehículo; El presente proyecto comprende la implementación de prototipo de sistema de visión artificial en el parqueadero central de la institución, utilizando procesamiento de imágenes mediante una cámara ofreciendo varias ventajas para automatizar el rendimiento, por medio de un sistema embebido,

dando como resultado un sistema inteligente que pueda tomar decisiones por cuenta propia para permitir el monitoreo en tiempo real.

El proyecto contribuirá con la comunidad Universitaria, ayudando en la búsqueda de espacios disponibles de parqueo, a su vez se tendrá en base de datos, los horarios más congestionados y tiempo que se encuentra disponible, mediante reportes a los que tendrá accesibilidad un administrador encargado de la misma; este procedimiento favorece al mejoramiento de la eficiencia en las operaciones de estacionamiento.

El presente trabajo se alinea al plan de creación de oportunidades, tomando en consideración los siguientes objetivos [11]:

Eje seguridad integral

Objetivo 9: Garantizar la seguridad ciudadana, orden público y gestión de riesgos.

Política 9.2: Fortalecer la seguridad de los sistemas de transporte terrestre y aéreo, promoviendo un ambiente seguro.

1.5 ALCANCE DEL PROYECTO

Según la problemática planteada en los parqueaderos de la Universidad Estatal Península de Santa Elena, así mismo, por los inconvenientes que ocasiona en el factor tiempo al encontrar un espacio para el vehículo del personal que diariamente asiste al plantel, es de vital importancia disminuir el tiempo de búsqueda de los mismos, mediante la implementación de un prototipo de sistema basado en visión artificial, que permitirá al usuario visualizar si se encuentran lugares libres en el sector.

Con el fin de tener un control más adecuado al momento en que el estacionamiento esté lleno, se almacenará el tiempo que dura cada vehículo en estar parqueadero, mostrando estos datos en reportes manejados por el administrador, el cual visualizará diariamente las horas en que el parqueadero está en uso, tanto de entrada como salida y la cantidad de veces que ha detectado un vehículo.

Esta implementación se realizará utilizando visión artificial, la cual constará como un sistema domótico de vigilancia y tendrá un algoritmo para cubrir el espacio del parqueadero, así, mediante la cámara determinar si ese espacio está ocupado o no. Estos equipos estarán conectados en un Arduino que va vinculado a una base de datos,

recolectando la información ya antes prevista, a su vez, mostrará en la pantalla si el estacionamiento se encuentra completamente lleno o si aún existe lugar disponible.

- ✓ La etapa 1 va vinculada a los dispositivos que son los componentes básicos para el entorno, brindando solución y procesamiento mediante recopilación de información, su funcionalidad interconectados para enviar datos a ser procesados, reciben comandos para actuar sobre el entorno y son un elemento esencial de las soluciones.
- ✓ La etapa 2 es la infraestructura que recopila por medio del dispositivo, los datos y los almacena, de igual manera permite el proceso visual que genera la cámara para medir los espacios ocupados y determina la cantidad de entradas vehiculares disponibles en el parqueadero.
- ✓ El análisis de datos se realiza en la etapa 3 donde se desarrolla el algoritmo y la aplicación de análisis de datos, de la información recopilada en la etapa anterior y datos de los requisitos de experiencia del usuario, identificados en la fase del proyecto.
- ✓ Finalmente, la etapa 4 comprende los módulos internos para los procesos de acuerdo con las especificaciones métricas que se determinarán en la visión artificial de la cámara, consta también de la parte de la reportaría en donde se visualizarán las entradas necesarias de los registros diarios vehiculares, tales como: fechas, horas y cantidades de automotores.

Este proyecto tiene el objetivo de indicar al conductor si el parqueadero posee disponibilidad vehicular, evitando la entrada innecesaria al estacionamiento por medio de la visualización de los datos obtenidos en el proceso y proyectados en una pantalla.

1.6 METODOLOGÍA DEL PROYECTO

1.6.1 METODOLOGÍA DE INVESTIGACIÓN

Luego de realizar la recolección e investigación que se requirió para la adquisición de los datos acerca de la implementación y análisis de algoritmos para el manejo de las mismas en un entorno abierto con visión artificial, se empleará una metodología de investigación tipo exploratoria [12], orientada a recopilar información bibliográfica de trabajos semejantes que tienen como objetivo ser una guía que apoya al desarrollo de esta propuesta.

Entre los trabajos referidos para el análisis de este proyecto, a nivel mundial se realizó una tesis en Lima – Perú denominada “Implementación de un sistema web basado en visión artificial y geolocalización, y su influencia en la vigilancia del Distrito de Lince” [13]; a nivel Nacional en la ciudad de Ambato el Ing. Jovann Pérez elaboró el tema titulado “Análisis de tráfico vehicular mediante visión artificial” que oriento un análisis de tráfico con modalidad fuera de línea [14]. Mientras que, localmente se desarrolló la tesis “Propuesta de mejora al acceso de vehículos autorizados y no autorizados mediante el reconocimiento de placas, tratamiento de imagen y automatización al edificio El Velero Azul”, el cual utiliza diferentes técnicas de visión artificial, obteniendo como resultado el reconocimiento de diversos objetos con procesos que permiten modificación y modelación [15].

Así mismo, para recabar información acerca de los procesos que se llevan a cabo actualmente en el parqueadero de la UPSE, se realizaron varias técnicas de recolección de datos, todo esto, aplicando la metodología de tipo diagnóstica [12].

1.6.2 BENEFICIARIOS DEL PROYECTO

Los beneficiarios directos de tu proyecto serían:

Docentes y Personal Administrativo de la Facultades de la UPSE: Al implementar un sistema de gestión de espacios de estacionamiento basado en técnicas de visión por computador en el parqueadero central de la UPSE, los docentes y personal administrativo se beneficiarán al tener una herramienta que les permita identificar fácilmente espacios vacíos de estacionamiento, optimizando así el tiempo que dedican a buscar un lugar para estacionar sus vehículos.

Estudiantes de la UPSE: Los estudiantes también se beneficiarán al contar con un sistema que les facilite la búsqueda de espacios de estacionamiento disponibles en el parqueadero central de la universidad, lo que contribuirá a mejorar su experiencia diaria en el campus al reducir el tiempo empleado en esta tarea.

Administradores y guardias: El sistema también beneficiará a los administradores del parqueadero al proporcionarles datos precisos sobre la ocupación del mismo, lo que les permitirá tomar decisiones informadas para mejorar la gestión del espacio y optimizar su utilización.

Comunidad Universitaria en General: La implementación de este sistema mejorará la eficiencia en las operaciones de estacionamiento en la UPSE, lo que beneficiará a toda la comunidad universitaria al reducir la congestión vehicular, mejorar la organización del parqueadero y optimizar el tiempo de búsqueda de espacios de estacionamiento.

1.6.3 VARIABLE

El objetivo de este trabajo es reducir el tiempo de búsqueda:

- **El tiempo que tarda en detectar espacios libres en el área central de la universidad utilizando tecnología artificial para reducir el tiempo de búsqueda de estacionamiento:** Esta variable representa el tiempo que el sistema de detección requiere para identificar la disponibilidad de espacio de estacionamiento mediante el uso de tecnología artificial, utilizando pantallas que mostrará si cuenta con espacio libre para poder parquear su vehículo en el estacionamiento.

1.6.4 ANÁLISIS DE RECOLECCIÓN DE DATOS

Con la finalidad de adquirir información sobre los problemas que se han generado en el estacionamiento central de la Universidad Estatal Península de Santa Elena, se emplearon diferentes técnicas de recolección de información, para recopilar datos relevantes. Se realizó una encuesta a los docentes de la Facultad de Sistemas y Telecomunicaciones de la UPSE ([Ver Anexo 1. Encuesta dirigida a la comunidad universitaria que posee vehículos en la Universidad Estatal Península de Santa Elena](#)), determinando diversos aspectos que causan inconvenientes al encontrar parqueadero para sus vehículos.

De igual forma, mediante el método el método de observación en el parqueadero ([Ver Anexo 2. Observación realizada en el parqueadero central de la Universidad Estatal Península de Santa Elena](#)), se recolectaron los datos más relevantes que se suscitan en el estacionamiento central, determinando los siguientes:

<i>EFECTO</i>	<i>CAUSA</i>
<i>Espacio por las mañanas</i>	Las personas ingresan por la mañana al parqueadero, dando varias vueltas sin encontrar lugares disponibles.

<i>Sin espacio para parquarse.</i>	<ol style="list-style-type: none"> 1. Al no encontrar parqueo, estacionan en cualquier lugar, muchas veces impidiendo el paso de otros vehículos ya parqueados. 2. Parquean el vehículo al ras de la calle, cerca del lugar donde les toca ver o impartir clases, también. 3. Se estacionan en lugares inseguros donde no pueden vigilar sus vehículos.
<i>Inconvenientes</i>	El inconveniente de no encontrar parqueadero dificulta la hora de entrada de los docentes y personal administrativo.
<i>Administración</i>	Se puede ver que existe tráfico vehicular en los lugares de parqueo, ya que, existe una administración regular en los mismos.

Tabla 1. Información del método de observación

Con la recolección de esta información mediante las técnicas ya antes vistas, se concluye que, es requerido para el usuario de la comunidad universitaria el uso de la tecnología de visión artificial para indicar al conductor del vehículo acerca de los espacios disponibles que se encuentran en el parqueadero, disminuyendo el tiempo que les toma encontrar un espacio disponible en el lugar.

1.6.4.1 ENCUESTA

La encuesta se ha diseñado con el propósito de recopilar información detallada sobre los problemas y desafíos actuales en el parqueadero central de la Universidad Estatal Península de Santa Elena (**Ver Anexo 1. Encuesta dirigida a la comunidad universitaria que posee vehículos en la Universidad Estatal Península de Santa Elena**). Estas preguntas están diseñadas para abordar diversos aspectos relacionados con la experiencia de los usuarios al buscar estacionamiento, así como para comprender la eficiencia y organización del parqueadero. La información obtenida a través de esta encuesta será fundamental para identificar áreas de mejora y guiar el desarrollo de soluciones efectivas para optimizar el uso del espacio de estacionamiento en la universidad.

1.6.4.2 POBLACIÓN

La población de estudio para este proyecto de investigación serían los conductores que utilizan el parqueadero central de la UPSE. Esto incluiría a estudiantes, profesores, personal administrativo y visitantes que estacionan sus vehículos en dicho lugar.

BENEFICIARIOS	CANTIDAD
Estudiantes	25
Profesores	15
Personal administrativo	25
TOTAL	75

Tabla 2. Información de la población

1.6.4.3 MUESTRA

La muestra de 85 conductores se seleccionará siguiendo un proceso de muestreo estratificado y aleatorio dentro de la población de conductores que utilizan el parqueadero central de la Universidad Estatal Península de Santa Elena. La muestra de la investigación se obtuvo utilizando la fórmula estadística para población finita, con un margen de error del 0.05%.

$$n = \frac{Z^2 PQN}{(N - 1)E^2 + Z^2 PQ}$$

En donde:

n= Tamaño de muestra

Z= Valor Z curva normal (1.96)

P= Probabilidad de éxito (0.50)

Q= Probabilidad de fracaso (0.50)

N= Población (75)

E= Error muestral (0.05)

$$n = \frac{(1.96)^2(0.50)(0.50)(75)}{(75 - 1)(0.05)^2 + (1.96)^2(0.50)(0.50)}$$

$$n = \frac{(3.84)(0.25)(75)}{(74)(0.0025) + (3.84)(0.25)}$$

$$n = \frac{(0.96)(75)}{0.185 + 0.96}$$

$$n = \frac{72}{1.145}$$

$$n = 62.98$$

$$n \approx 63$$

El total de la muestra de la investigación fue de 63 conductores.

1.6.5 ANÁLISIS DE RESULTADO DE ENCUESTA

La encuesta se llevó a cabo con el propósito de recopilar información valiosa y opiniones de los beneficiarios directos de nuestro proyecto, es decir, los docentes de la Facultades de la UPSE. La encuesta se diseñó y administró utilizando un formulario digital y se distribuyó entre los docentes de manera amplia. En total, se recibieron respuestas de 63 docentes, estudiantes y personal administrativo en nuestra institución.

El objetivo principal de la encuesta fue evaluar los inconvenientes existentes en el parqueadero central de la UPSE y comprender la percepción sobre la necesidad de implementar soluciones tecnológicas para mejorar la gestión del estacionamiento.

Este análisis de resultados es fundamental para comprender las necesidades y perspectivas con respecto a la optimización del parqueadero central de la UPSE. Los datos recopilados servirán como base esencial para la toma de decisiones y la adaptación de nuestro proyecto a las necesidades reales de los usuarios en la gestión del estacionamiento.

A continuación, se presenta un análisis detallado de las respuestas obtenidas de las 63 encuestados. Los resultados se presentan en forma gráfica para facilitar la comprensión visual de las respuestas:

Pregunta 1. ¿Con qué frecuencia encuentra parqueadero para su vehículo?

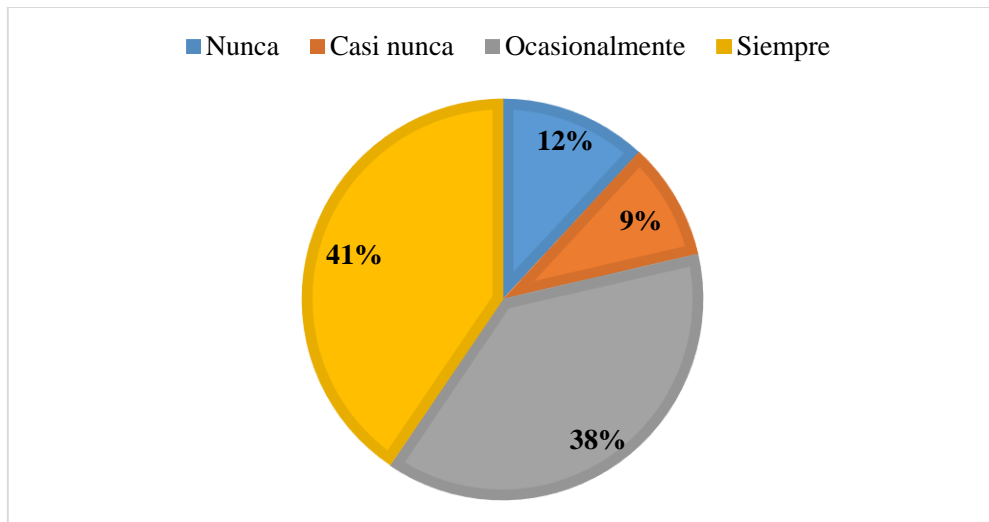


Figura 2. Frecuencia de parqueo.

Del 100% de los encuestados, se pudo determinar que, el 41% siempre encuentra parqueadero para su vehículo, mientras que, el 38% ocasionalmente encuentran espacio, el 12% nunca hallan lugar disponible y solo el 9% no encuentra parqueadero casi nunca.

Pregunta 2. ¿Cuánto tiempo se demora en encontrar parqueadero?

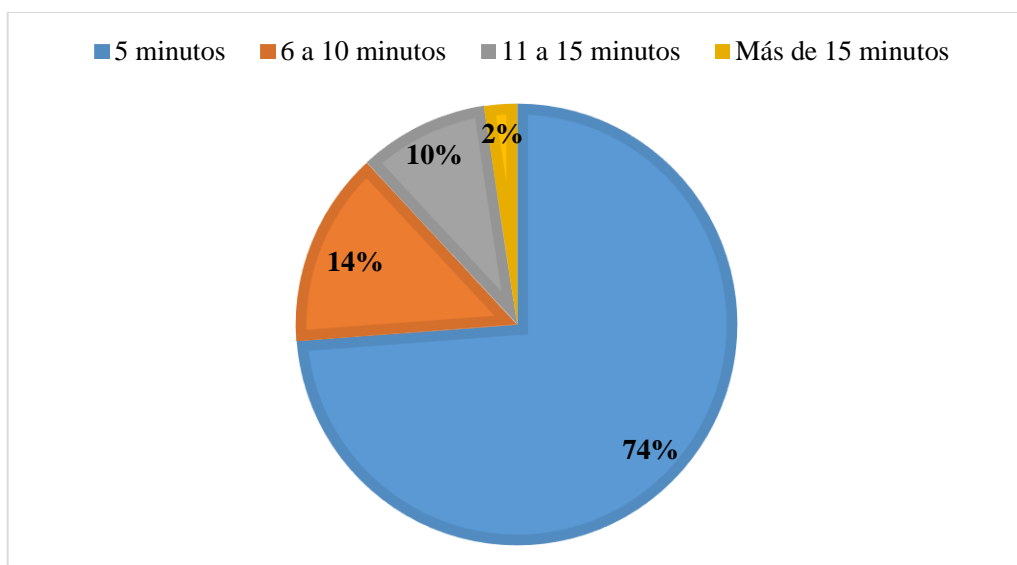


Figura 3. Tiempo en encontrar parqueadero.

El 74% de la población demoran 5 minutos aproximadamente en encontrar parqueadero para su vehículo, mientras que, al 14% le toma de 6 a 10 minutos hallar lugar, el 10% demora de 11 a 15 minutos y solo el 2% tarda más de 15 minutos en encontrar parqueadero.

Pregunta 3. ¿Cree usted que el tiempo que le toma encontrar parqueadero, afecta en sus actividades laborales o escolares?

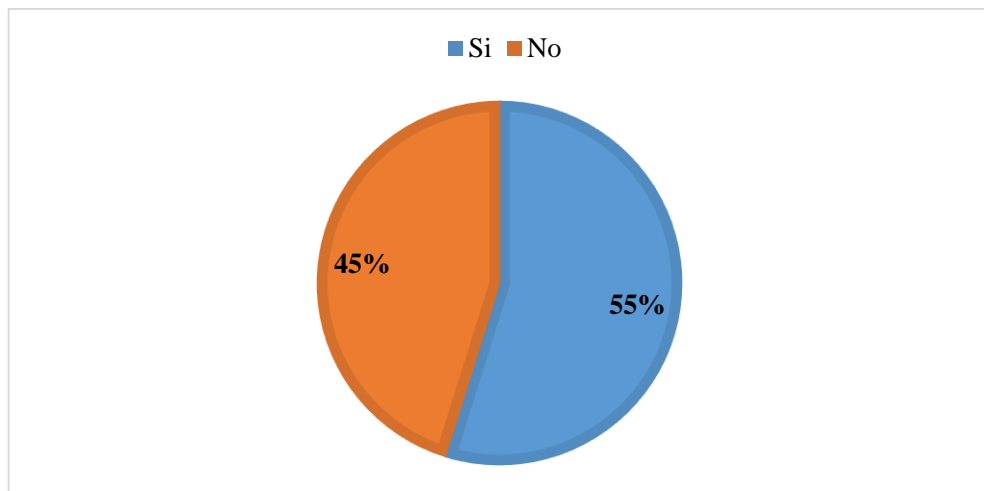


Figura 4. Afectación en las actividades.

Se determinó que, el 55% si cree que el tiempo que les toma encontrar parqueadero, afecta en sus actividades laborales o escolares, mientras que, el 45% piensa que no les afecta.

Pregunta 4. ¿Con qué frecuencia se encuentra con congestión vehicular al momento de buscar parqueo?

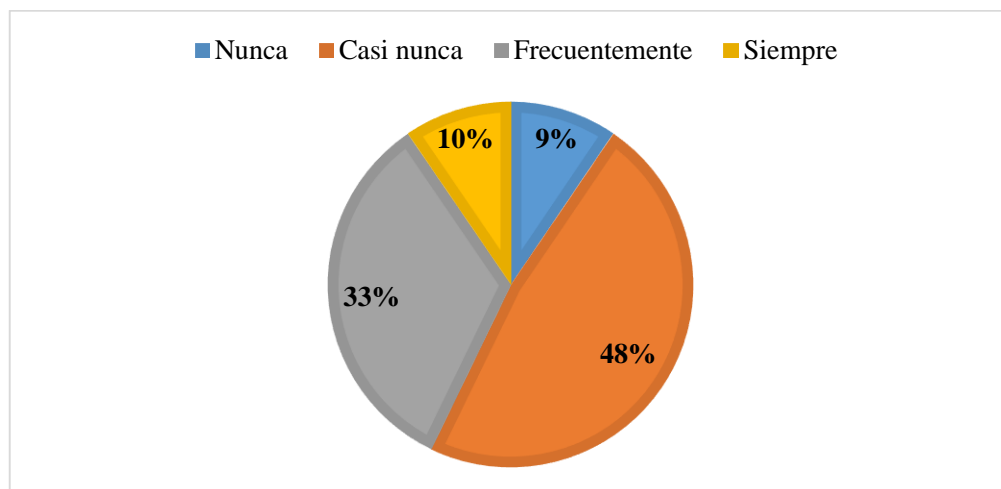


Figura 5. Frecuencia de congestión vehicular.

El 48% de los encuestados manifiestan que, casi nunca se encuentran con congestión vehicular al momento de buscar parqueo, mientras que, el 33%

frecuentemente se encuentran con congestión, el 10% indican que siempre presencian este problema y solo el 9% dicen que nunca.

Pregunta 5. ¿Con qué frecuencia usted ingresa y sale de la institución en el día?

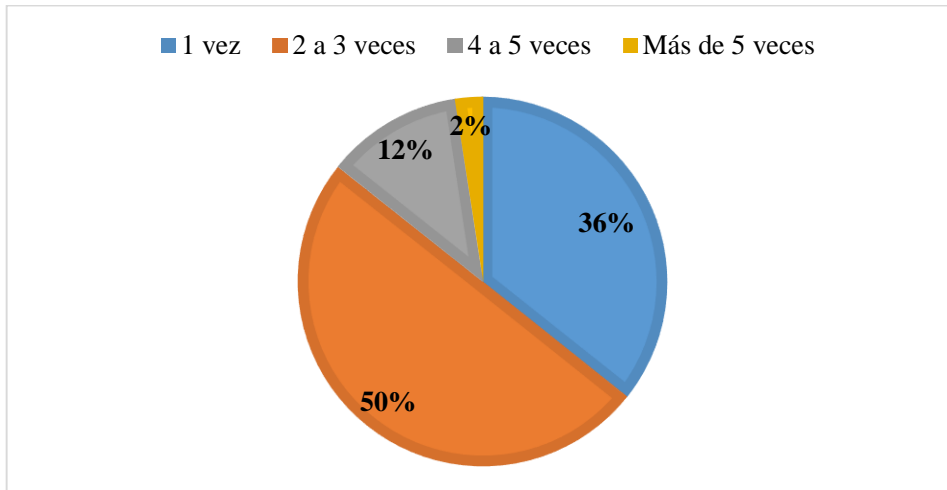


Figura 6. Frecuencia de ingreso y salida de la institución.

Del 100% de la población encuestada se pudo determinar que, el 50% ingresa y sale de la institución en el día de 2 a 3 veces, mientras que el 36% solo 1 vez, el 12% de 4 a 5 veces y solo el 2% ingresa y sale más de 5 veces.

Pregunta 6. ¿Cómo calificaría la organización actual en el parqueadero central del a UPSE?

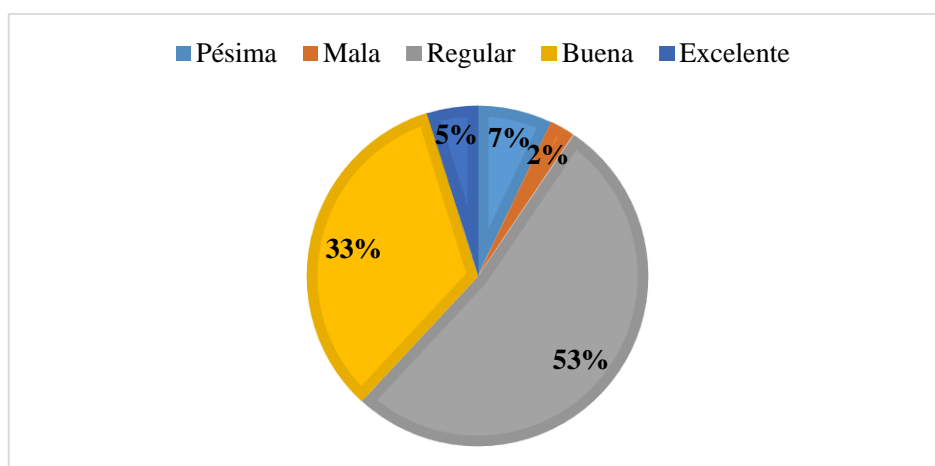


Figura 7. Calificación de la organización actual del parqueadero.

Se pudo determinar que, el 53% de los encuestados manifiestan que, la organización actual en el parqueadero central de la UPSE es regular, mientras que, el 33% indican que es buena, el 7% dicen que es excelente, el 5% pésima y solo el 2% manifiestan que es mala.

Pregunta 7. ¿Qué inconvenientes ha tenido para salir de su lugar de parqueo?

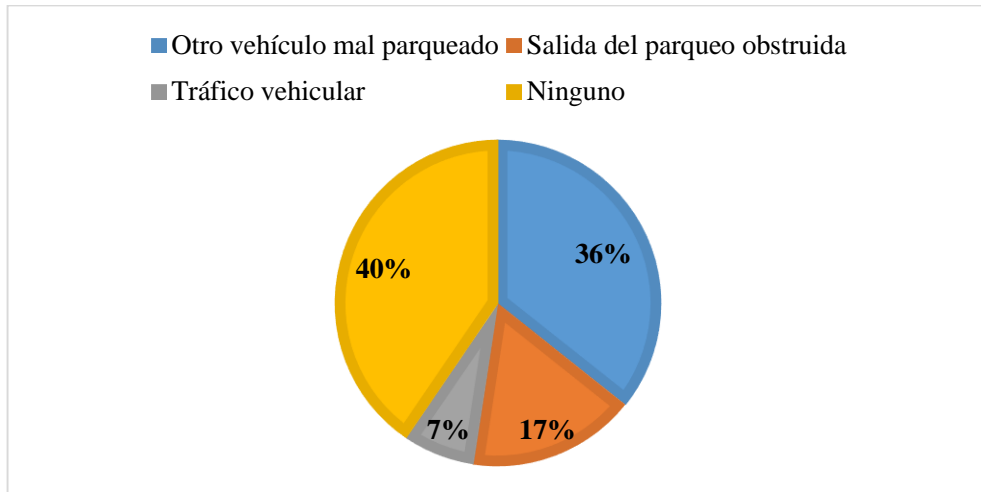


Figura 8. Inconvenientes para salir del parqueo.

El 40% de la población encuestada manifiesta que, no ha tenido ningún inconveniente para salir de su lugar de parqueo, sin embargo, el 36% ha presenciado vehículos mal parqueados, el 17% ha sido testigo de una salida de parqueo obstruida y el 7% manifiesta que ha tenido problemas con el tráfico vehicular.

Pregunta 8. ¿Cree usted que sea necesario un sistema que muestre en cada espacio del parqueadero, la disponibilidad de parqueo mediante una pantalla?

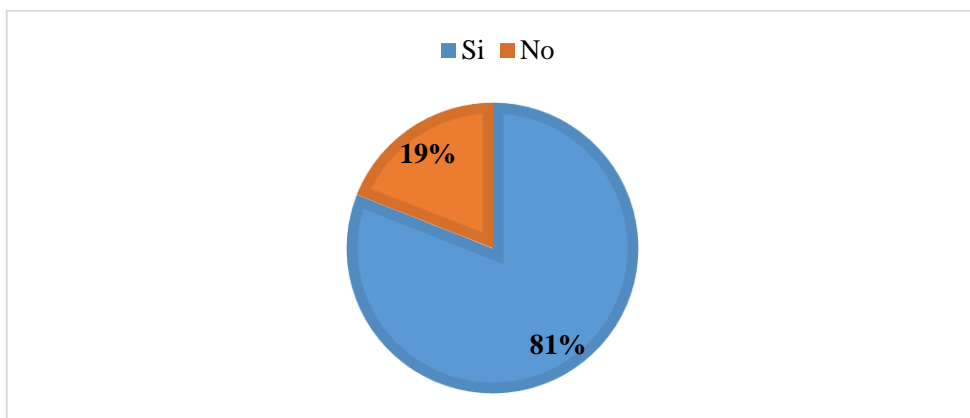


Figura 9. Sistema que muestre disponibilidad en el parqueadero.

El 81% de los encuestados creen que, si es necesario un sistema que muestre en cada espacio del parqueadero, la disponibilidad de parqueo mediante una pantalla, sin embargo, el 19% no piensan que sea necesario.

1.7 METODOLOGÍA DE DESARROLLO

Para desarrollar este proyecto se adaptó una metodología incremental para la implementación adecuada de la estructura de la visión artificial, con la finalidad de obtener el máximo beneficio al momento de realizar las pruebas debidas y la ejecución. De igual manera, se definió que esta presentación se dividirá en 4 fases para su funcionalidad, las cuales son: Análisis, Diseño Codificación y Pruebas [16].

- **Análisis:** Se elaboran todos los requerimientos del proyecto, en base a las técnicas de recolección de datos.
- **Diseño:** Creación del diseño, es decir, de la estructura del modelo de visión artificial.
- **Codificación:** Desarrollo del algoritmo de visión artificial, brindándole la debida funcionalidad.
- **Pruebas:** Se realizan las pruebas respectivas del proyecto, asegurando que todo funcione de manera correcta.

Las etapas de desarrollo se dividen en los siguientes incrementos:

- **Incremento 1:** Implementación del modelo.
- **Incremento 2:** Entrenamiento del modelo.
- **Incremento 3:** Modulo de aplicación.
- **Incremento 4:** Modulo de reportes.

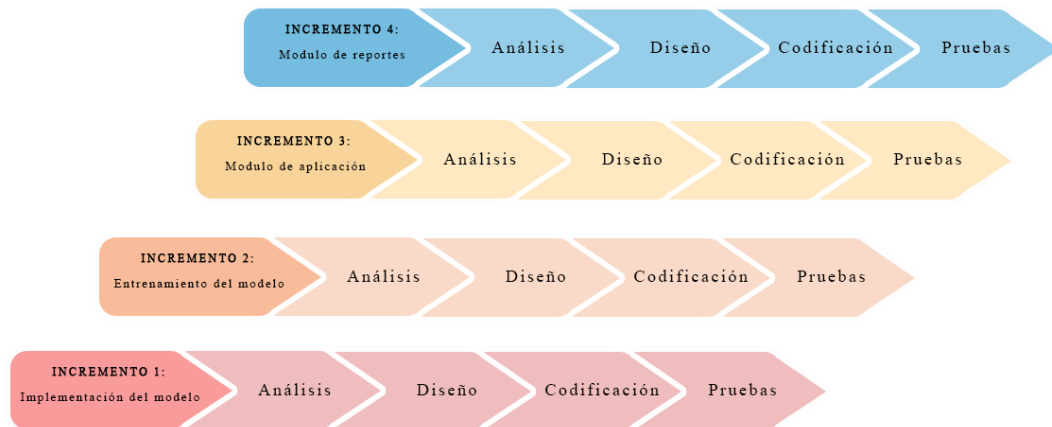


Figura 10. Metodología de desarrollo del proyecto.

CAPÍTULO 2. PROPUESTA

2.1 MARCO CONTEXTUAL

2.1.1 UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

La Universidad Estatal Península de Santa Elena (UPSE) es una institución comprometida con la excelencia académica y el desarrollo integral de la región. Ubicada en la provincia de Santa Elena, Ecuador, la UPSE se destaca por su enfoque en la formación de profesionales altamente capacitados y en la generación de conocimiento e innovación para contribuir al progreso local y nacional.

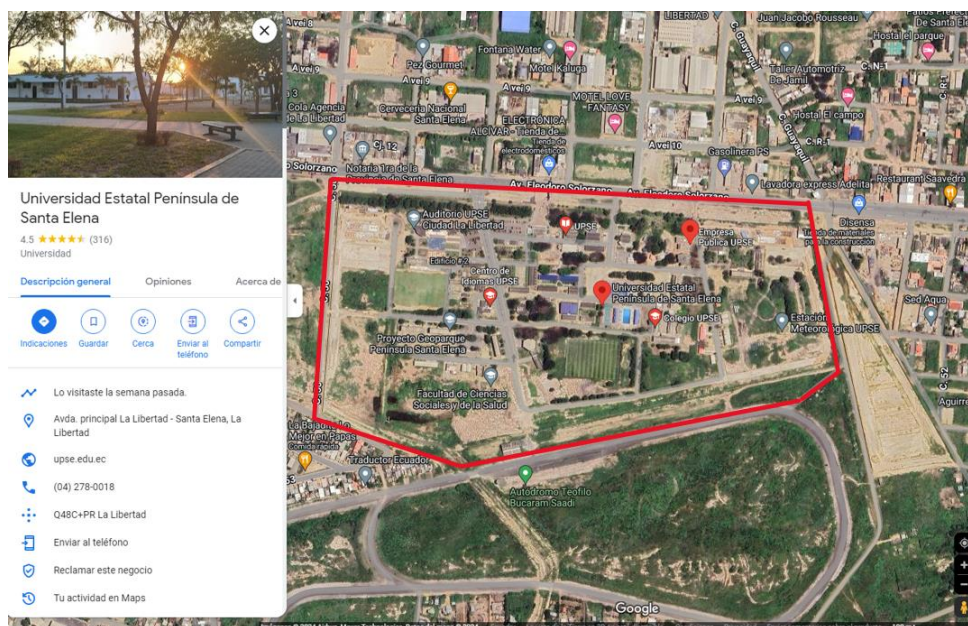


Figura 11. Ubicación de la UPSE.

En la actualidad, la UPSE enfrenta diversos desafíos, entre ellos, la optimización de sus servicios y recursos para satisfacer las necesidades de su comunidad estudiantil, docente y administrativa.

Uno de estos desafíos es la gestión eficiente del estacionamiento en el campus universitario, especialmente en el parqueadero central, donde la disponibilidad de espacios adecuados es fundamental para garantizar la movilidad y comodidad de quienes acuden diariamente a la institución.



Figura 12. Estacionamiento de la UPSE.

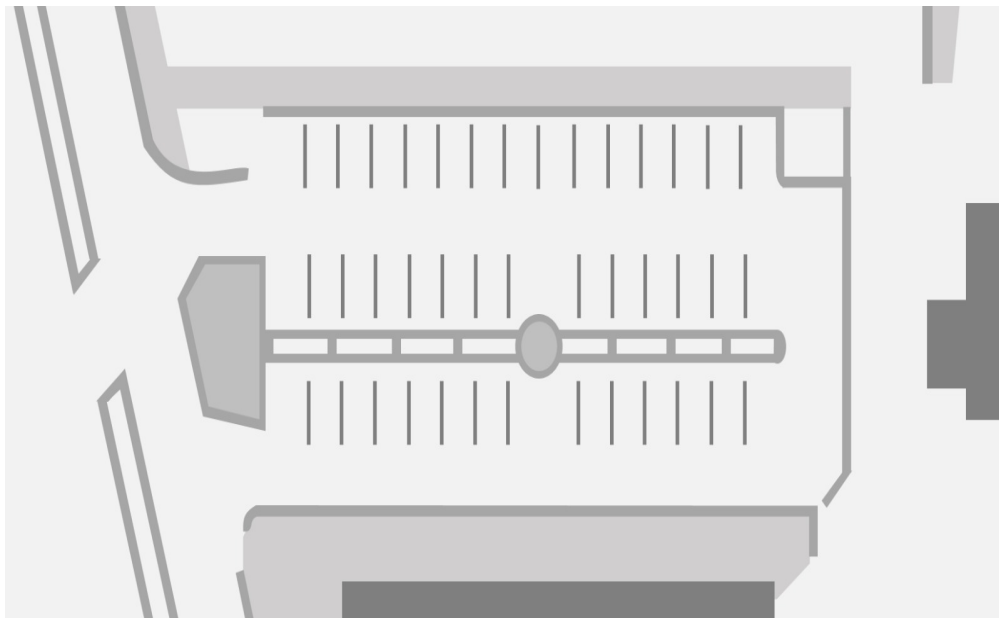


Figura 13. Plano del estacionamiento central de la UPSE.

2.1.2 MISIÓN

Formar profesionales que aportan al desarrollo sostenible, contribuye a la solución de los problemas de la comunidad y promueve la cultura [17].

2.1.3 VISIÓN

Ser reconocida por su calidad académica, impacto de sus investigaciones y su aporte al desarrollo de la sociedad [17].

2.2 MARCO CONCEPTUAL

2.2.1 Visión Artificial

La visión artificial es un sistema que adquiere información del entorno a través de imágenes y las analiza con precisión utilizando diversos tipos de sensores. En la industria, este sistema no opera de manera independiente, sino que se integra en toda la cadena de producción gracias a la transferencia de datos que realiza. Al proporcionar a las máquinas la capacidad de procesar imágenes, la visión artificial simula la percepción visual humana, aunque con características distintivas [18].

Este proceso inicia con la adquisición de imágenes mediante sensores específicos, cuyos tipos influyen en la calidad y el tipo de datos obtenidos. Para procesar estas imágenes, se requiere software especializado que aplique los algoritmos necesarios para cumplir con las funciones designadas a las máquinas, como la detección de defectos o la realización de mediciones [19].

2.2.1.1 Inteligencia Artificial

La inteligencia artificial (IA), en el ámbito de la informática, representa una vanguardia tecnológica centrada en la creación de sistemas y algoritmos que emulan capacidades cognitivas humanas. Su objetivo primordial es desarrollar máquinas capaces de ejecutar tareas que normalmente requerirían intervención humana, abarcando desde el procesamiento de datos complejos hasta el aprendizaje automático. En la actualidad, la IA se erige como una herramienta indispensable en la automatización de procesos y el análisis de grandes volúmenes de información, convirtiéndose así en un pilar fundamental en el avance de la sociedad hacia la era digital [20].

2.2.1.2 Redes Neuronales

Las redes neuronales, cuya estructura está inspirada en el funcionamiento del cerebro humano, se han convertido en un pilar esencial en el campo del aprendizaje automático y la inteligencia artificial. Compuestas por capas de neuronas artificiales interconectadas, estas redes son capaces de procesar y transmitir información a través de conexiones ponderadas, lo que les confiere la capacidad de abordar tareas complejas de manera eficiente. Su flexibilidad y adaptabilidad las han posicionado como una herramienta clave en numerosas aplicaciones tecnológicas, donde destacan por su habilidad para analizar y comprender datos complejos, ya sea en el ámbito de la visión por computadora, el procesamiento del lenguaje natural o la predicción de patrones en grandes conjuntos de datos [21].

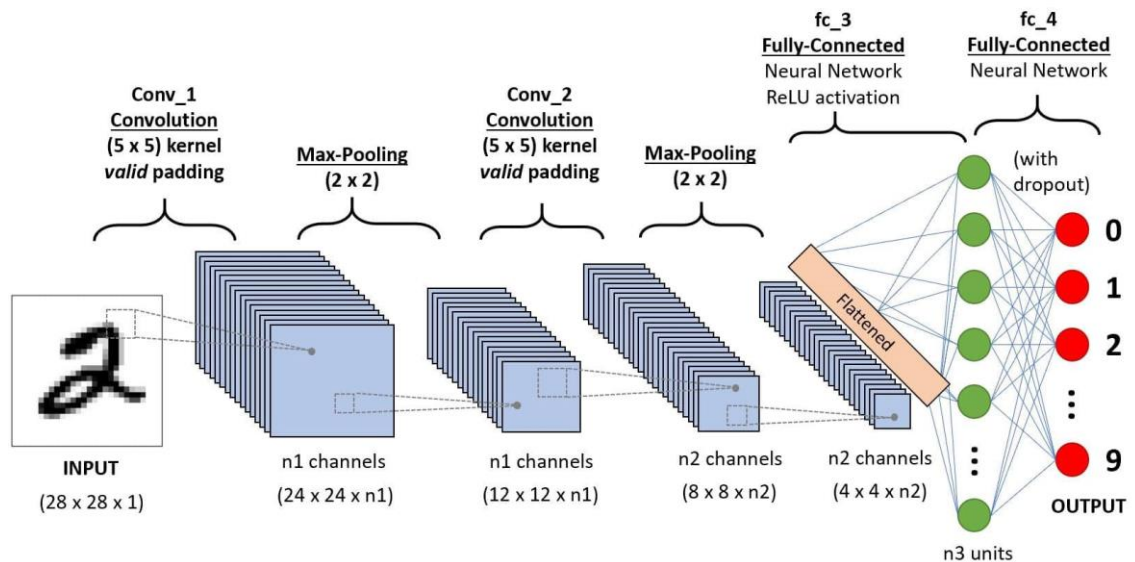


Figura 14. Procesamiento De Imágenes Y La Detección De Objetos.

Específicamente, las redes neuronales convolucionales (CNN) representan un tipo especializado de red neuronal que ha revolucionado el campo del procesamiento de imágenes y la detección de objetos. La Figura 22 proporciona un esquema ilustrativo de la arquitectura interna de una CNN, donde se pueden observar las capas convolucionales y de agrupamiento que permiten analizar y reconocer patrones visuales en imágenes de manera eficaz. Esta capacidad de las CNN para extraer características relevantes de las imágenes las ha vuelto indispensables en aplicaciones como la visión por computadora,

la conducción autónoma, la medicina diagnóstica y la seguridad, entre otras áreas donde la interpretación precisa de información visual es crucial [22].

2.2.1.3 Aprendizaje Automático

El aprendizaje automático, dentro del ámbito de la inteligencia artificial (IA), se centra en la creación de algoritmos y modelos que permiten a las máquinas aprender y mejorar de manera automática a partir de los datos disponibles. En contraposición a ser programadas de manera explícita para tareas específicas, las máquinas emplean análisis de patrones y construcción de modelos matemáticos para realizar predicciones, tomar decisiones y extraer conocimientos de conjuntos de datos previamente no vistos [23].

Este campo se dedica al desarrollo de algoritmos computacionales capaces de identificar patrones en grandes volúmenes de datos o de realizar predicciones basadas en dichos patrones. Además, engloba diversas técnicas como el aprendizaje supervisado, no supervisado y reforzado, y encuentra aplicación en una amplia gama de áreas, desde la clasificación de correos no deseados y la detección de fraudes, hasta la recomendación de productos y la visión por computadora. En la actualidad, el aprendizaje automático desempeña un papel fundamental en la automatización de procesos y la toma de decisiones basadas en datos [23].

2.2.1.4 Aprendizaje Profundo

El Deep Learning, o Aprendizaje Profundo, ha emergido como una vanguardia revolucionaria en el ámbito de la inteligencia artificial y el procesamiento de datos. Este enfoque se basa en la creación de modelos computacionales inspirados en la estructura y función del cerebro humano, específicamente en las redes neuronales artificiales. Lo que distingue al Deep Learning de otros métodos de aprendizaje automático es su capacidad para aprender representaciones jerárquicas de datos de manera autónoma y progresiva. A través de capas de procesamiento, los algoritmos de Deep Learning pueden discernir características complejas y abstracciones en conjuntos de datos enormes y no estructurados, como imágenes, texto y sonido, extrayendo patrones significativos que son difíciles de detectar con enfoques convencionales [24].

Esta tecnología ha revolucionado una amplia gama de campos, desde la visión por computadora hasta el procesamiento del lenguaje natural, pasando por la robótica. En la

industria automotriz, los vehículos autónomos dependen en gran medida del Deep Learning para interpretar señales de tráfico, reconocer peatones y tomar decisiones en tiempo real. A medida que la tecnología continúa evolucionando y mejorando, el Deep Learning promete seguir transformando radicalmente nuestra forma de interactuar con la información y el mundo que nos rodea [25].

2.2.2 Algoritmo De Deteccion De Objetos

El algoritmo de detección de objetos, esencial en la visión por computadora y el aprendizaje automático, permite identificar y ubicar objetos específicos en imágenes o vídeos. Esta técnica es fundamental en diversas aplicaciones como la conducción autónoma, la seguridad y la detección de peatones en sistemas de asistencia al conductor. Una de las técnicas más populares para esta detección es el uso de redes neuronales convolucionales (CNN), capaces de aprender automáticamente características relevantes de las imágenes y utilizarlas para detectar objetos. Estas redes son entrenadas con grandes conjuntos de datos etiquetados, lo que les permite reconocer la presencia y la ubicación de los objetos de interés [26].

El proceso de detección de objetos involucra varias etapas, donde la imagen se procesa a través de la red neuronal para extraer características importantes, identificar regiones de la imagen que contienen objetos y aplicar técnicas de post-procesamiento para refinar las detecciones y eliminar falsos positivos. Gracias a los avances en el aprendizaje profundo y la potencia computacional, los algoritmos de detección de objetos han evolucionado considerablemente, siendo capaces de detectar una amplia variedad de objetos en diferentes condiciones, lo que los convierte en una herramienta invaluable en diversas aplicaciones prácticas [27].

2.2.2.1 Modelo YOLO (You Only Look Once)

El Modelo YOLO representa un avance significativo en la detección de objetos en tiempo real, particularmente enfocado en la eficiencia y precisión al identificar vehículos y determinar la ocupación de espacios en los estacionamientos. A diferencia de los enfoques tradicionales que dividían la tarea de detección en múltiples pasos, como el reconocimiento de características seguido por la clasificación de objetos, YOLO adopta un enfoque más integrado y eficiente [28].

Este modelo funciona mediante la división de la imagen en una cuadrícula y la predicción de cajas delimitadoras y probabilidades de clase para cada celda de la cuadrícula simultáneamente. Esta arquitectura única permite que YOLO realice predicciones de manera rápida y precisa en una sola pasada por la red neuronal, lo que resulta en una detección casi en tiempo real de objetos en imágenes y videos [28].

2.2.2.2 Modelos Preentrenados

Son una herramienta fundamental en el campo de la visión artificial, ya que ofrecen un punto de partida sólido y eficiente para el desarrollo de algoritmos de detección de vehículos y otras tareas relacionadas. Estos modelos han sido entrenados previamente en grandes conjuntos de datos diversos, lo que les ha permitido aprender representaciones complejas y útiles de características visuales. Al utilizar modelos preentrenados como punto de partida, los investigadores y desarrolladores pueden evitar la necesidad de entrenar un modelo desde cero, lo que puede ser costoso en tiempo y recursos computacionales [29].

En lugar de eso, pueden aprovechar los conocimientos y las representaciones aprendidas por el modelo preentrenado en conjuntos de datos generales, como ImageNet, que contienen una amplia variedad de clases de objetos y escenas. Cuando se trata de la detección de vehículos, los modelos preentrenados proporcionan una base sólida para identificar características relevantes, como formas, texturas y contextos, que son comunes en imágenes de vehículos. Al adaptar y afinar estos modelos preentrenados en conjuntos de datos específicos de detección de vehículos, los investigadores pueden lograr una mejora significativa en la precisión y la eficiencia de sus algoritmos [30].

2.2.2.3 Optimización de Algoritmos

La optimización de algoritmos en el contexto de la visión artificial aplicada a la gestión de estacionamientos es crucial para mejorar tanto la eficiencia como la precisión del sistema. Esta tarea implica una serie de técnicas destinadas a perfeccionar el rendimiento de los algoritmos utilizados. Entre estas técnicas se incluye la selección de características relevantes, donde se identifican y utilizan únicamente las características más informativas para la tarea de detección de vehículos [31].

Esto reduce la cantidad de datos que deben procesarse, lo que a su vez disminuye la complejidad computacional y mejora los tiempos de respuesta del sistema, lo cual es fundamental en entornos en tiempo real como los parqueaderos. Además, la optimización puede implicar la reducción de la complejidad de los algoritmos, mediante la simplificación de estructuras o la aplicación de técnicas de optimización específicas. Estos esfuerzos de optimización son esenciales para garantizar que el sistema pueda manejar eficientemente grandes volúmenes de datos de imágenes de manera rápida y precisa [31].

2.2.2.4 Validación y Evaluación de Modelos

Constituyen una etapa crítica en el desarrollo de sistemas de detección de vehículos en parqueaderos. Durante esta fase, se analiza el rendimiento de los algoritmos en términos de precisión, sensibilidad, especificidad y otros parámetros relevantes. La precisión se refiere a la proporción de detecciones correctas realizadas por el algoritmo, mientras que la sensibilidad y la especificidad evalúan su capacidad para detectar vehículos positivos y evitar falsos positivos, respectivamente. Además, se pueden emplear otras métricas, como la tasa de error global o el coeficiente de correlación de Matthews, para evaluar aspectos específicos del rendimiento del modelo [32].

Esta evaluación exhaustiva garantiza la confiabilidad y efectividad del sistema de detección de vehículos, permitiendo identificar posibles áreas de mejora y ajuste. En conjunto, la optimización de algoritmos y la validación de modelos son procesos interdependientes que contribuyen de manera significativa a la creación de sistemas de visión artificial robustos y eficaces para la gestión de estacionamientos [32].

2.2.2.5 Integración De Sistemas

La integración de sistemas en la gestión de parqueaderos es un proceso complejo que requiere una cuidadosa planificación y coordinación. No se trata simplemente de implementar algoritmos de visión artificial, sino de asegurar que estos algoritmos funcionen de manera armoniosa con otros sistemas ya existentes en el parqueadero. Esto puede implicar la sincronización de datos entre los sistemas de detección de vehículos y los sistemas de gestión de acceso, así como la integración con sistemas de reserva de espacios en línea o aplicaciones móviles. La interoperabilidad entre estos sistemas es fundamental para garantizar una experiencia de estacionamiento fluida y eficiente para

los usuarios, al tiempo que se optimiza la gestión del espacio disponible y se maximiza la capacidad de generación de ingresos del parqueadero [32].

2.2.2.6 Interfaz De Usuario

La interfaz de usuario (UI), su diseño y desarrollo son aspectos críticos que impactan directamente en la usabilidad y la efectividad de los sistemas de visión artificial en el parqueadero. Una interfaz intuitiva y fácil de usar permite a los operadores visualizar rápidamente los resultados de los algoritmos de detección de vehículos, interpretar los datos y tomar decisiones informadas. Elementos como mapas interactivos de estacionamiento, indicadores de ocupación en tiempo real y notificaciones visuales de alerta pueden mejorar significativamente la experiencia del usuario y la eficiencia operativa del sistema. Además, la adaptabilidad de la interfaz para diferentes dispositivos y tamaños de pantalla garantiza una experiencia consistente y accesible para todos los usuarios, independientemente de su dispositivo o ubicación [33].

2.2.2.7 Seguridad Y Privacidad

En lo que respecta a la seguridad y privacidad, es fundamental implementar medidas sólidas para proteger la integridad de los datos recopilados por los sistemas de visión artificial en el parqueadero. Esto incluye la encriptación de datos, el acceso restringido a la información sensible y el cumplimiento de regulaciones de protección de datos como el GDPR. Además, se deben establecer políticas claras de manejo de datos y privacidad para garantizar la confianza de los usuarios y mitigar cualquier riesgo potencial de violación de la privacidad [34].

2.2.2.8 Escalabilidad Y El Mantenimiento

La escalabilidad y el mantenimiento son aspectos igualmente importantes para asegurar la eficacia a largo plazo de los sistemas de visión artificial en el parqueadero. Esto implica diseñar los sistemas con flexibilidad para adaptarse a futuros cambios y expansiones, así como garantizar la disponibilidad de soporte técnico y actualizaciones regulares para mantener los sistemas actualizados y funcionando de manera óptima [35].

2.2.2.9 Tendencias Futuras

Las tendencias futuras, es importante estar al tanto de los avances y desarrollos emergentes en el campo de la visión artificial, ya que estos pueden tener un impacto

significativo en el desarrollo y la implementación de sistemas de control de espacios en el parqueadero. Esto puede incluir avances en áreas como el aprendizaje profundo, la visión por computadora 3D, la detección de objetos en tiempo real y la inteligencia artificial aplicada a la gestión de la movilidad urbana. Esto al tanto permite a los desarrolladores y administradores de parqueaderos anticipar y adaptarse a los cambios en el mercado y mantenerse a la vanguardia de la innovación tecnológica [36].

2.2.3 Herramientas y Bibliotecas de Desarrollo

En este apartado se incluyen las herramientas y bibliotecas de software utilizadas para implementar y desarrollar los algoritmos de visión artificial. Esto puede abarcar bibliotecas como OpenCV para el procesamiento de imágenes, Torch o Numpy para la implementación de redes neuronales, y herramientas específicas para el etiquetado y preparación de datos de entrenamiento, como Roboflow.

2.2.3.1 Python

Python es un lenguaje de programación ampliamente empleado en diversas aplicaciones, incluyendo el desarrollo de sitios web, la automatización de tareas y el análisis de datos. Al ser un lenguaje de propósito general, Python es adecuado para crear una variedad de programas sin estar limitado a un área específica. Su versatilidad y su accesibilidad para principiantes lo han convertido en uno de los lenguajes más populares en la actualidad. Según una encuesta realizada por la empresa de análisis del sector RedMonk en 2021, Python ocupó el segundo lugar como el lenguaje de programación más utilizado por los desarrolladores [37].

2.2.3.2 OpenCV

OpenCV, que significa Open Source Computer Vision Library, es una biblioteca de código abierto ampliamente utilizada en el campo de la visión por computadora y el procesamiento de imágenes. Esta biblioteca ofrece una amplia gama de funciones y herramientas para el análisis y procesamiento de imágenes y videos en tiempo real. OpenCV proporciona funciones para realizar tareas como detección de objetos, seguimiento de objetos, reconocimiento facial, calibración de cámaras, entre otras. Su versatilidad y facilidad de uso lo convierten en una opción popular para desarrolladores y científicos de datos que trabajan en aplicaciones de visión artificial [38].

2.2.3.3 PyTorch

Es un framework de aprendizaje profundo de código abierto desarrollado por Facebook's AI Research lab (FAIR). Se destaca por su flexibilidad y facilidad de uso, especialmente en el desarrollo de modelos de redes neuronales. PyTorch ofrece una interfaz intuitiva que permite a los usuarios definir y entrenar modelos de manera eficiente. Además, su arquitectura dinámica de gráficos computacionales facilita la experimentación y la depuración de modelos. PyTorch es ampliamente utilizado en la investigación y la industria para una variedad de aplicaciones de aprendizaje profundo [39].

2.2.3.4 NumPy

NumPy es una biblioteca fundamental en el ecosistema de Python para computación científica. Proporciona un potente soporte para matrices y operaciones matemáticas, lo que lo hace invaluable para tareas de análisis de datos y procesamiento numérico. NumPy es conocido por su eficiencia y rendimiento, gracias a su implementación en C y su capacidad para trabajar con grandes conjuntos de datos de manera eficiente. Muchas otras bibliotecas de Python, incluidas pandas y scikit-learn, se basan en NumPy para realizar operaciones matriciales y numéricas [40].

2.2.3.5 Roboflow

Roboflow es una plataforma de gestión de datos y entrenamiento de modelos diseñada específicamente para proyectos de visión por computadora y aprendizaje automático. Permite a los usuarios cargar, organizar y preprocesar conjuntos de datos de imágenes de manera eficiente. Además, ofrece herramientas para etiquetar datos, generar conjuntos de datos de entrenamiento y pruebas, y colaborar en proyectos de equipo. Su enfoque centrado en la facilidad de uso y la productividad lo convierte en una opción popular para desarrolladores y equipos que trabajan en aplicaciones de visión artificial [41].

2.2.3.6 Visual Basic

Visual Basic (VB) es un lenguaje de programación y un entorno de desarrollo integrado (IDE) desarrollado por Microsoft. Se utiliza principalmente para el desarrollo de aplicaciones de escritorio y soluciones empresariales en el entorno de Windows. Visual Basic ofrece una sintaxis sencilla y una amplia gama de herramientas de desarrollo que lo hacen adecuado para desarrolladores principiantes y experimentados por igual. Aunque

su popularidad ha disminuido en los últimos años en favor de otros lenguajes y plataformas de desarrollo, Visual Basic sigue siendo una opción viable para desarrollar aplicaciones de software personalizadas y soluciones empresariales en el ecosistema de Windows [42].

2.3 Marco Teórico

2.3.1 Algoritmos de Detección de Vehículos en Estacionamientos

En el ámbito de la visión por computadora y el aprendizaje automático, los algoritmos de detección de vehículos en estacionamientos representan un área de investigación crucial. Estos algoritmos se centran en identificar y localizar vehículos específicos en imágenes o vídeos, lo que resulta fundamental para aplicaciones como la gestión de aparcamientos, la seguridad vial y la conducción autónoma. Para comprender en profundidad estos algoritmos, es esencial explorar la teoría de la visión por computadora, que se enfoca en los fundamentos detrás de la detección de objetos en imágenes. Además, el aprendizaje automático, especialmente el enfoque del Machine Learning, juega un papel vital al permitir que los algoritmos aprendan de los datos y mejoren su capacidad de detección en diferentes condiciones [31].

La teoría de la visión por computadora y el aprendizaje automático, los algoritmos de detección de vehículos en estacionamientos se benefician de la investigación en áreas como la optimización de algoritmos y la ingeniería de características. La optimización de algoritmos busca mejorar la eficiencia y la velocidad de los algoritmos de detección, lo que es crucial para su aplicación en tiempo real en sistemas de gestión de aparcamientos y conducción autónoma. Por otro lado, la ingeniería de características se centra en identificar y seleccionar las características más relevantes de las imágenes de vehículos, lo que permite mejorar la precisión y la robustez de los algoritmos de detección frente a diversas condiciones ambientales y variaciones en la apariencia de los vehículos [43].

La investigación en fusiones de datos también desempeña un papel importante en el desarrollo de algoritmos de detección de vehículos en estacionamientos. La fusión de datos combina información de múltiples fuentes, como cámaras, sensores de ultrasonido y sistemas de posicionamiento global (GPS), para mejorar la precisión y la confiabilidad de la detección de vehículos. Esta integración de datos permite a los algoritmos obtener

una comprensión más completa del entorno del estacionamiento y facilita la detección de vehículos en situaciones complejas, como estacionamientos abarrotados o entornos urbanos densamente poblados. En conjunto, estas investigaciones contribuyen al desarrollo de algoritmos de detección de vehículos cada vez más sofisticados y eficientes, con aplicaciones potenciales en una variedad de campos, desde la gestión de tráfico hasta la seguridad vial [44].

2.3.2 Técnicas para Identificar Automóviles en Áreas de Aparcamiento

La identificación de automóviles en áreas de aparcamiento es un desafío técnico que requiere el uso de diversas técnicas y teorías. Una de las teorías fundamentales en este contexto es la teoría de procesamiento de imágenes, que proporciona los fundamentos para manipular y analizar imágenes digitales con el fin de extraer características relevantes para la detección de vehículos, como formas y colores. La geometría computacional también desempeña un papel importante al permitir que los algoritmos utilicen la geometría para identificar la forma y el tamaño de los vehículos en las imágenes del aparcamiento. Además, la teoría de la detección de objetos ofrece un marco conceptual para entrenar a los algoritmos a identificar vehículos específicos en entornos complejos como los aparcamientos [45].

La teoría de procesamiento de imágenes y la geometría computacional, la investigación en identificación de automóviles en áreas de aparcamiento también se ha beneficiado de avances en técnicas de segmentación de imágenes. Estas técnicas permiten la separación de regiones de interés, como los vehículos, del fondo circundante, lo que facilita su identificación y seguimiento con mayor precisión. Por otro lado, el uso de características avanzadas, como descriptores de características locales y características aprendidas mediante técnicas de aprendizaje profundo, ha demostrado ser efectivo para mejorar la robustez y la precisión de los algoritmos de identificación de vehículos en entornos desafiantes, como condiciones de iluminación variable o presencia de obstáculos [46].

Asimismo, la integración de tecnologías emergentes, como la fusión de datos de múltiples fuentes, incluidas cámaras, sensores LiDAR y datos de GPS, ha permitido el desarrollo de sistemas más completos y precisos para la identificación de automóviles en áreas de aparcamiento. Estas técnicas avanzadas no solo mejoran la precisión de la detección y seguimiento de vehículos, sino que también proporcionan información adicional, como

la velocidad y la dirección del vehículo, que puede ser crucial para aplicaciones de gestión de tráfico y seguridad vial. En conjunto, estas diversas técnicas y teorías forman un marco integral para abordar el desafío de la identificación de vehículos en entornos de aparcamiento, contribuyendo así al desarrollo de soluciones cada vez más eficientes y sofisticadas en este campo [47].

2.3.3 Métodos de Reconocimiento de Coches en Espacios de Estacionamiento

El reconocimiento de coches en espacios de estacionamiento implica la aplicación de diversos métodos y teorías para lograr una detección precisa. La teoría de la detección de patrones es esencial en este contexto, ya que proporciona los fundamentos para identificar patrones específicos en imágenes que representan vehículos en un entorno de estacionamiento. Además, la teoría de la detección de contornos desempeña un papel crucial al permitir que los algoritmos identifiquen los bordes de los vehículos en las imágenes, facilitando así su detección y reconocimiento. Por último, la teoría de procesamiento de señales es relevante para entender cómo los algoritmos pueden procesar señales de imágenes y extraer características importantes para la detección de vehículos, como la forma, el tamaño y la posición, mejorando así la precisión del reconocimiento en entornos de estacionamiento [48].

La teoría de la detección de patrones y la detección de contornos, el reconocimiento de coches en espacios de estacionamiento se beneficia de la aplicación de métodos avanzados de clasificación y reconocimiento de objetos. Estos métodos pueden incluir técnicas de aprendizaje supervisado, como máquinas de vectores de soporte (SVM) y clasificadores basados en árboles de decisión, que permiten la identificación y clasificación precisa de vehículos en imágenes de estacionamientos. La integración de características contextuales, como la disposición de los vehículos dentro del espacio de estacionamiento y la relación con otros objetos cercanos, también juega un papel crucial en el proceso de reconocimiento, ya que proporciona información adicional para mejorar la precisión de la identificación de vehículos individuales [46].

El desarrollo de métodos de reconocimiento de coches en espacios de estacionamiento se ha visto impulsado por avances en técnicas de aprendizaje profundo, como las redes neuronales convolucionales (CNN). Estas redes son capaces de aprender representaciones jerárquicas de las características de las imágenes, lo que les permite realizar tareas de

reconocimiento de objetos con una precisión sin precedentes. La aplicación de CNNs en el reconocimiento de coches en espacios de estacionamiento ha demostrado ser especialmente efectiva para detectar vehículos en diferentes condiciones ambientales y de iluminación, así como para manejar la variabilidad en la apariencia de los vehículos debido a factores como el ángulo de visión y la distancia [49].

En conjunto, la combinación de métodos tradicionales de procesamiento de imágenes con técnicas avanzadas de aprendizaje automático y reconocimiento de patrones ha permitido el desarrollo de sistemas de reconocimiento de coches cada vez más precisos y eficientes en entornos de estacionamiento. Estos avances tienen importantes implicaciones para la gestión de estacionamientos, la seguridad vial y el desarrollo de sistemas de conducción autónoma más sofisticados y confiables [49].

2.4 REQUERIMIENTOS

2.4.1 Requerimientos Funcionales

Código	Requerimiento	Descripción de Requerimiento
RF-01	Visualización de disponibilidad vehicular	El sistema de control de espacios permitirá visualizar la disponibilidad vehicular en el parqueadero central de la UPSE.
RF-02	Información en tiempo real sobre espacios	El sistema mostrará en tiempo real sobre la información de cada espacio disponible y ocupado.
RF-03	Registro de datos de vehículos parqueados	El sistema guarda los datos de cada vehículo parqueado en los lugares del estacionamiento.
RF-04	Almacenamiento de fecha y hora de entrada	El sistema almacena la fecha y hora al momento que el vehículo ingresa y sale del lugar de parqueo.

RF-05	Indicación de espacio disponible	Cuando un espacio de parqueo está disponible, en la pantalla se mostrará el mensaje “Disponible”.
RF-06	Indicación de espacio ocupado	Cuando un espacio de parqueo esté ocupado, en la pantalla se mostrará el mensaje “Ocupado”.
RF-07	Control de fechas de reportes	Posibilidad de elegir un rango de fechas para generar informes.
RF-08	Instalación física en parqueadero central	El sistema estará instalado físicamente en el parqueadero central de la Universidad Estatal Península de Santa Elena.
RF-09	Suministro de energía para cámara	El sistema deberá contar con un suministro de energía para la conexión de la cámara.
RF-10	Sistema de alerta visual o auditiva	El sistema contará con un webservice almacenado en el servidor web, el cuál tendrá los archivos respectivos para realizar las consultas a la base de datos.
RF-11	Webservice en servidor web	El sistema validará la conexión del servidor, antes de ejecutar las consultas
RF-12	Detección de plazas ocupadas	El algoritmo detectará las plazas ocupadas y disponibles del parqueadero a través de la cámara.
RF-13	Visualizar los reportes en	Los reportes se visualizarán en la interfaz.

Tabla 3. Requerimientos funcionales.

2.4.2 Requerimientos No Funcionales

Código	Requerimiento	Descripción de Requerimiento
RNF-01	Conexión del sistema a un servidor web con base de datos MySQL	El sistema se conectará como cliente hacia un servidor web en el que estará alojada la base de datos MySQL.
RNF-02	Resolución mínima de la cámara de video	La cámara de video debe contener una resolución mínima de 720p.
RNF-03	Ligereza de la base de datos	La base de datos tiene que ser ligera.
RNF-04	Principio de integridad en la base de datos	Debe tener principio de integridad, es decir, solo almacenar información relevante.
RNF-05	Escalabilidad del sistema	El sistema es escalable, es decir, puede ser reentrenado para aumentar su eficiencia.
RNF-06	Control del sistema mediante placa Arduino uno	El sistema será controlado por la placa Arduino uno.
RNF-07	Conexión del sistema a una pantalla LED de 1.3 pulgadas	El sistema estará conectado a una pantalla led de 1.3 pulgadas.
RNF-08	Conexión del sistema a una cámara	El sistema estará conectado a una cámara Hikvision, modelo DS2CD1123G0-I.

	Hikvision modelo DS2CD1123G0-I	
RNF-09	Aplicación del principio de usabilidad en el sistema	El sistema aplica el principio de usabilidad, es decir, las detecciones son autónomas y la intervención con el usuario es mínima.

Tabla 4. Requerimientos no funcionales.

2.5 Componente de la propuesta

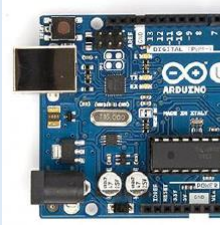



2.5.1 Arquitectura del sistema

2.5.1.1 Algoritmo con Herramientas de IOT

COMPARACIÓN DE DISPOSITIVOS

Para determinar el dispositivo Arduino que cumpla con los requerimientos adecuados para el proyecto, se tomaron en cuenta características y costos, de los cuales hay una variedad en el mercado y de la gamma que existen se pueden visualizar los siguientes:

Teniendo en cuenta las características que se adecúan y el valor necesario al presupuesto, se determina que, la placa especializada que se utilizará para la práctica de este estudio es el Arduino uno.





Características	Arduino Uno	Arduino Leonardo	Arduino Mega 2560	Arduino YUN
Imagen				
Microcontrolador	ATMega328P	ATMega32u4	ATMega2560	ATMega32u4

Velocidad de reloj	16 MHz	16 MHz	16 MHz	16 MHz
Voltaje de trabajo	5V	3,3V	5V	5V
Voltaje de entrada	7,5 a 12 voltios	7,5 a 12 voltios	7,5 a 12 voltios	7,5 a 12 voltios
Pines	14 pines digitales: • 6 PWM 6 pines analógicos	20 pines digitales • 7 PWM 12 pines analógicos.	54 pines digitales • 15 PWM) 16 pines analógicos	20 pines digitales • 7 PWM 12 pines analógicos
Puertos	1 puerto serie por hardware	-	3 puertos serie por hardware	-
Memoria ROM	32 KB Flash (0,5 para bootloader)	32 KB Flash (4KB para bootloader)	256 KB Flash (8KB para bootloader)	Memoria: 32 KB Flash (4KB para bootloader)
RAM	2KB	2,5KB	8KB	2,5KB
Eeprom	1KB	1KB	4KB	1KB
Valor	14.10	10.99	20.80	96.42

Tabla 5: Comparación de Arduino

Para la visualización de entorno del estacionamiento se requerirá de una cámara que permita su entrada de video por medio de una IP. Este tipo de dispositivo viene previsto de diferentes capacidades gráficas y distancias como se muestra a continuación:

Características	Hikvision Domo	Hikvision Tubo	Dahua Domo	Dahua Tubo
-----------------	----------------	----------------	------------	------------

Imagen				
Distribuidora	Hikvision	Hikvision	Dahua	Dahua
Modelo	DS2CD1123G0-I	DS-2CD1023G0-I	DH-IPC-HDPW1230R1N	DH-IPC-HFW1230S1N
Resolución	1920×1080@30 fps	1920 × 1080	2MP@30FPS	2MP@30FPS
Lente	2.8mm@F2.0	2.8mm	Lente Fijo 2.8mm	Lente Fijo 2.8mm
Compresión	H.265+, H.265, H.264+, H.264	H.265 + / H.265 / H.264 + / H.264 / MJPEG	H.264, H.264+, H.265, H.265+	H.264, H.265
Alimentación	12Vdc +/- 25%, PoE (802.3af). TCP/IP: 10/100Mbps	12VDC y PoE	12VDC	12VDC -
Tipo	PoE	PoE	PoE	PoE
Resolución	2Mp HD 1080p 30fps	2MP, 1/ 2.8	2MP@30FPS	2MP@30FPS
Distancia	IR 20 a 30m	2.0- 30m	IR 30m	IR 30m

Valor	46.00	48.00	46.00	42.90
--------------	-------	-------	-------	-------

Tabla 6: Comparación de cámaras

Determinando las particularidades adecuadas, el rendimiento casi asimétrico de las características y los valores, se optó por la distribuidora Hikvision con el modelo DS2CD1123G0-I, puesto que es una cámara compacta y gracias a su domo, podrá resistir las inclemencias del clima que la provincia posee, a su vez, se ajusta a los requerimientos bases para este trabajo.

Por el lado de la conectividad, se posee un módulo estable para la conexión del Arduino a la red ya sea por wifi como por ethernet para establecer las comunicaciones entre los equipos con las siguientes características:

- Microcontrolador ATmega328
- **Voltaje de funcionamiento: 5V**
- **Voltaje de entrada: 7-12V**
- **Puertos de I/O digitales: 14** (4 son utilizados por el controlador Ethernet)
- **Pines de entrada analógica: 6**
- Incluye lector de tarjetas Micro SD
- Controlador de Ethernet integrado W5100 TCP / IP



Tabla 7: Conexión del Arduino

En lo que se requiere para la visualización del conductor acerca de la disponibilidad de parqueadero, se utilizará una pantalla oled lcd 1.3" que tiene las siguientes características:

- **Tamaño:** 1.3"
- **Alta resolución:** 128 x 64
- **Ángulo de visión:** > 160 °
- **Soporta muchos tipos de control:** totalmente compatible con Arduino, MSP430, STM32 / 2, etc.
- **Ultra-bajo consumo de energía:** 0.08W pantalla totalmente iluminada, 0.06W visualización normal a pantalla completa de caracteres tipo texto
- **Amplio rango de voltaje de alimentación:** sin ninguna modificación, se puede alimentar directamente 3V ~ 5V DC
- **Temperatura de trabajo:** -30 °C ~ 70 °C
- **Dimensiones:** 32 x 35.5 x 4.1mm
- Ocupación de pines de comunicaciones mediante comunicación IIC / I2C
- **Controlador:** SSD1306



Figura 15: Pantalla LCD

Fase 2.- Diseño

MODELADO DE BASE DE DATOS

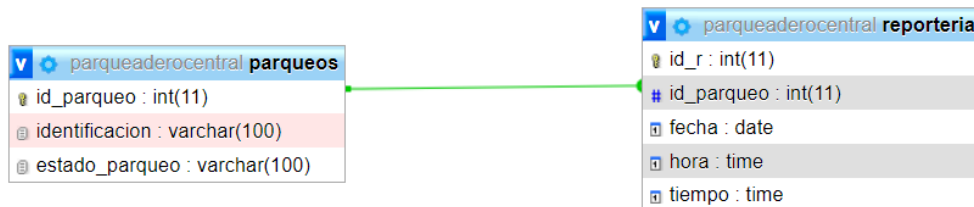


Figura 16: Modelado de base de datos

La creación del sistema de control de espacios para comprobar la disponibilidad vehicular en el parqueadero, se utilizarán dos tablas en la base de datos, siendo estas:

- **Parqueos:** Registra la información de cada espacio de parqueo con su respectivo estado, ya sea, disponible u ocupado.
- **Reportería:** Almacena los datos para los reportes que visualizará el administrador del sistema, es decir, la fecha y hora que ingresa cada vehículo al parqueadero y el tiempo que permanece en el lugar.

El sistema de detección se trabajará desde Python y la arquitectura IOT por Arduino.

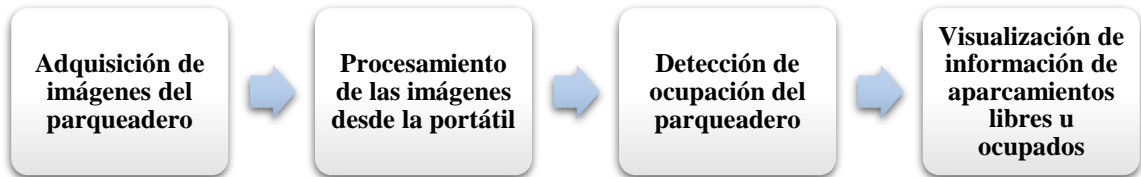


Figura 17: Proceso de sistema de detección

La **adquisición de imágenes** inicia desde la captura de datos mediante una cámara de forma centralizada a una altura determinada, que se encuentre apuntalando y estableciendo toda el área en la que se pretenda extraer la información.

El procesamiento de imágenes utiliza código en Python y la librería OpenCV para manejar las imágenes adquiridas previamente. Se definen coordenadas geográficas y se añaden Regiones de Interés (ROI) con identificadores específicos. Esto permite enfocar en áreas específicas de la imagen para extraer características clave, con el objetivo de detectar vehículos en cada sección.



Figura 18: Identificadores de parqueos

Continuando con la **detección de ocupación en el parqueadero**, en este proceso se prescribe si existe aparcamientos ocupados y se distinguen de los libres. En caso de que esté ocupado se recolectarán los datos necesarios enviando la información a la base de

datos y se unificará la comunicación con Arduino mediante la librería pyserial, encargada de enviar los datos a través de la comunicación serial.

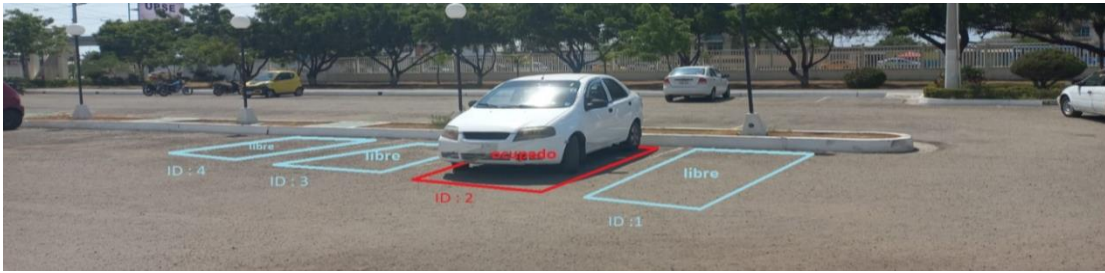


Figura 19: Detección de ocupación en el parqueadero

Arduino permitirá establecer la comunicación con el servidor, mediante un shield ethernet, operando el traslado de los distintos datos por el protocolo HTTP, permitiendo contemplar la arquitectura IoT. De la misma forma, el sistema controlador de los datos opera los componentes necesarios para visualizar los aparcamientos desocupados y dichos datos generados por el sistema se presentan en una pantalla LCD, teniendo la finalidad de **visualizar la información de los aparcamientos libres u ocupados**.

Fase 4.- Pruebas

Al realizar estas pruebas, se garantiza que el algoritmo de seguimiento de objetos y conteo de área cumpla con los requisitos especificados, proporcionando resultados precisos y eficientes en una variedad de condiciones y escenarios. A continuación, se desarrolla un cuadro descriptivo detallado lo esencial de los algoritmo indagado y desarrollado el análisis.

CUADRO DESCRIPTIVO DEL ANÁLISIS DE ALGORITMOS					
ALGORITMO	FASE 1: ANÁLISIS	FASE 2: DISEÑO	FASE 3: CODIFICACIÓN	FASE 4: PRUEBAS	DETALLE
Clasificación de Estacionamientos	<p>Requerimientos: Clasificación precisa y eficiente de espacios de estacionamiento.</p> <p>Herramientas: OpenCV.</p> <p>Entradas: Video en tiempo real. <</p>	<p>Arquitectura</p> <p>General: Clase <code>Park_classifier</code> para manejar los procesos de clasificación.</p> <p>Detalles de Implementación: Configuración de parámetros y rutas de archivos, uso de</p>	<p>Implementación</p> <p>Principal: Definición de parámetros, creación de instancias, captura y procesamiento de video, visualización en tiempo real, interacción con el usuario.</p>	<p>Pruebas de Precisión: Evaluación con diferentes videos.</p> <p>Pruebas de Robustez: Evaluación en condiciones adversas.</p>	ALGORITMO#1

	<p>Procesos: Lectura de video frame por frame, procesamiento de imágenes, dibujo de etiquetas, interacción con el usuario.</p> <p>Salidas: Video con marcadores visuales.</p>	<p>OpenCV, configuración de codec de video, bucle principal, interacción con el usuario.</p>	<p>Generador de Coordenadas: Creación de instancias, lectura y visualización de coordenadas.</p>	<p>Pruebas de Rendimiento: Medición de tiempos de procesamiento.</p>	
Seguimiento de Objetos y Conteo de Área	<p>Requerimientos: Seguimiento en tiempo real y conteo preciso de objetos.</p> <p>Herramientas: OpenCV.</p> <p>Entradas: Video en tiempo real.</p>	<p>Arquitectura General: Clase <code>Tracker</code> para manejo de seguimiento y conteo.</p> <p>Detalles de Implementación: Configuración de parámetros, lectura de video, procesamiento</p>	<p>Implementación Principal: Definición de parámetros, creación de instancias, captura y procesamiento de video frame por frame, visualización en tiempo real, interacción con el usuario.</p>	<p>Pruebas de Precisión: Evaluación con diferentes videos.</p> <p>Pruebas de Robustez: Evaluación en condiciones adversas.</p>	ALGORITMO#2

	<p>Procesos: Lectura de video frame por frame, seguimiento de objetos, conteo en áreas específicas.</p> <p>Salidas: Video con marcadores visuales.</p>	<p>frame por frame, interacción con el usuario.</p>	<p>Generador de Coordenadas: Creación de instancias, lectura y visualización de coordenadas.</p>	<p>Pruebas de Rendimiento: Medición de tiempos de procesamiento.</p>	
<p>Clasificador de Objetos Haar</p>	<p>Requerimientos: Detección precisa y eficiente de objetos en imágenes.</p> <p>Herramientas: OpenCV, Cascade Trainer GUI.</p> <p>Entradas: Imágenes positivas y negativas.</p>	<p>Arquitectura General: Uso de herramientas como Cascade Trainer GUI para facilitar el entrenamiento.</p> <p>Detalles de Implementación: Configuración de tamaño de imágenes, modo de entrenamiento</p>	<p>Implementación Principal: Desarrollo de un script para detección en tiempo real utilizando el archivo cascade.xml.</p> <p>Generador de Coordenadas: Captura y etiquetado de imágenes positivas y negativas,</p>	<p>Pruebas de Precisión: Evaluación con diferentes conjuntos de datos.</p> <p>Pruebas de Robustez: Evaluación en condiciones adversas.</p>	<p>ALGORITMO#3</p>

	<p>Procesos: Captura y etiquetado de imágenes, configuración y entrenamiento del clasificador.</p> <p>Salidas: Archivo cascade.xml con los datos entrenados.</p>	<p>HAAR, procesamiento del entrenamiento y generación del archivo cascade.xml.</p>	<p>almacenamiento y organización de las imágenes.</p>	<p>Pruebas de Rendimiento: Medición de tiempos de procesamiento y comparación con los requisitos establecidos.</p>	
--	--	--	---	---	--

Tabla 8: Cuadro Descriptivos de algoritmos de Análisis para detección de objetos – Visión Artificial

A continuación, se presente el cuadro descriptivo del desarrollo del Algoritmo de Detección de Objetos para Estacionamiento correspondiente del proyecto enfatizando puntos esenciales como análisis, diseño, herramientas para su elaboración

ALGORITMO DE DETECCIÓN DE OBJETOS PARA ESTACIONAMIENTO			DETALLE <u>ESTACIONAMIENTO</u>	
Fase	Descripción	Parámetros Clave	Herramientas	Comentarios/Resultados
Fase 1: Análisis	<p>Requerimientos: - Instalación de librerías: TensorFlow, OpenCV- Python, Numpy, Scikit-learn, Pillow.</p> <p>Proceso: - Instalación de las librerías mencionadas.</p> <p>- Desarrollo de la estructura del modelado para detección de objetos.</p> <p>Creación del entorno Python y codificación en Jupyter.</p>	<p>- Instalación correcta de librerías.</p> <p>Preparación del dataset.</p> <p>Configuración del entorno de desarrollo.</p>	<p>Jupyter Notebook</p> <p>Python Libraries (TensorFlow, OpenCV)</p>	<p>La instalación de las librerías y la preparación del dataset fueron exitosas, permitiendo un entorno de desarrollo sólido.</p>

	<p>Selección de una carpeta de imágenes clasificadas (llenos, vacíos, medios).</p> <p>- Ejecución de la codificación y uso de bibliotecas de Python para detección de objetos vehiculares en estacionamiento.</p>			
<p>Fase 2: Diseño</p>	<p>Herramienta: - Uso de IDE como Visual Studio Code.</p> <p>Partes del código: - Script de preprocesamiento y división de imágenes de estacionamientos.</p> <p>Script para carga de imágenes y aumento de datos.</p> <p>- Entrenamiento del modelo.</p> <p>Evaluación y guardado del modelo.</p>	<p>- Estructura clara del código.</p> <p>- Modularidad en los scripts.</p> <p>- Interfaz amigable para el usuario.</p>	<p>Visual Studio Code</p> <p>Python Libraries (TensorFlow, OpenCV)</p>	<p>La división modular del código facilitó la gestión y la depuración, y el uso de Visual Studio Code permitió una edición eficiente.</p>

	- Script para cargar y usar el modelo guardado.			
Fase 3: Codificación	<p>Script de Preprocesamiento y División de Imágenes: -</p> <p>Importaciones y definiciones de directorios.</p> <p>- Creación de directorios de salida.</p> <p>- Función de preprocesamiento de imágenes.</p> <p>- Preprocesamiento y división de imágenes.</p> <p>- Guardado de imágenes preprocesadas.</p>	<p>- Preprocesamiento eficaz de imágenes.</p> <p>- Aumento de datos para mejor generalización.</p> <p>- Definición y compilación de la arquitectura del modelo.</p> <p>- Entrenamiento y evaluación del modelo.</p>	Python Libraries (TensorFlow, OpenCV, NumPy, Scikit-learn)	La codificación fue exitosa, y el modelo mostró un buen rendimiento en términos de precisión y capacidad de generalización.

	<p>Script para Carga de Imágenes y Aumentar Datos:</p> <ul style="list-style-type: none"> - Importaciones. - Definición de rutas de directorios. - Creación de instancias de <code>ImageDataGenerator</code>. - Carga de imágenes usando generadores de datos. <p>Entrenamiento del Modelo:</p> <ul style="list-style-type: none"> - Importaciones. - Definición de la arquitectura del modelo CNN. - Compilación del modelo. - Entrenamiento del modelo. <p>Evaluación y Guardado del Modelo:</p>			
--	---	--	--	--

	<ul style="list-style-type: none"> - Evaluación del modelo en el conjunto de prueba. - Guardado del modelo entrenado. 			
Fase 4: Pruebas	<p>Resultados: - Procesamiento correcto de carpetas de muestras.</p> <ul style="list-style-type: none"> - Carga de imágenes y aumento de datos completados. - Entrenamiento del modelo exitoso. - Modelo entrenado y almacenado. - Resultados del script final: imagen detectada y analizada. 	<ul style="list-style-type: none"> - Validación y prueba del modelo. - Evaluación del rendimiento del modelo. - Almacenamiento del modelo entrenado para su uso futuro. - Visualización de resultados. 	Python Libraries (TensorFlow, OpenCV, Matplotlib)	Las pruebas confirmaron la precisión del modelo en la detección de objetos, y el almacenamiento del modelo entrenado asegura su reutilización futura.

Tabla 9: Cuadro Descriptivo – Algoritmo de detección de objetos para Estacionamiento

2.5.2 Diagramas De Casos De Uso

En los diagramas se detallan los componentes y funcionalidades del sistema, incluyendo los casos de uso más relevantes. Estos casos de uso son fundamentales para entender el funcionamiento y las ventajas de esta tecnología en el contexto de la gestión de estacionamientos.

Casos de Uso: Detectar Vehículo

Nombre del Caso de Uso: Detectar Vehículo	
Actor Principal:	Sistema de Visión por Computadora
Descripción:	El sistema detecta vehículos en las áreas definidas del estacionamiento mediante el uso de un modelo pre-entrenado de YOLOv5.
Precondiciones:	El sistema debe estar operativo y la cámara debe estar funcionando.
Postcondiciones:	Los vehículos detectados son registrados para la actualización de la disponibilidad de espacios.
Flujo Principal:	<ul style="list-style-type: none">• La cámara captura el video en tiempo real.• El sistema procesa cada cuadro del video.• El modelo de YOLOv5 detecta vehículos en las áreas definidas.• El sistema registra la detección de los vehículos
Flujos Alternativos:	Si no se detectan vehículos en el cuadro, el sistema continúa monitoreando.

Tabla 10: Caso de Uso: Detectar Vehículos

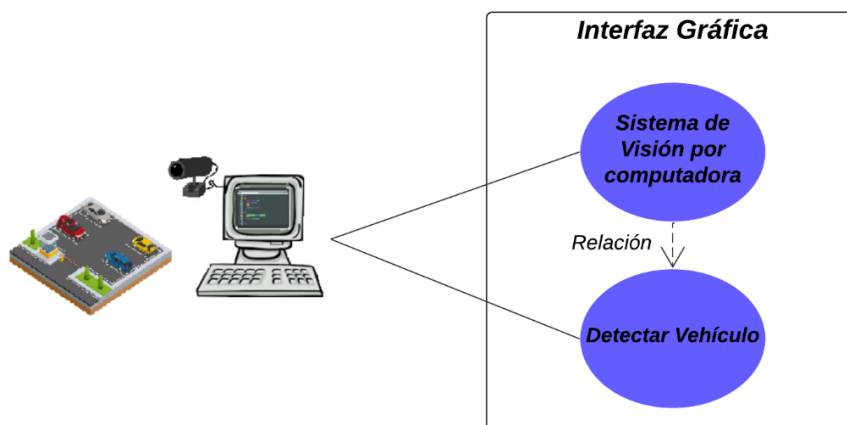


Figura 20: Diagrama de Caso de Uso: Detectar Vehículos

Caso de Uso: Registrar Espacio Ocupado

Nombre del Caso de Uso: Detectar Vehículo	
Actor Principal:	Sistema de Visión por Computadora
Descripción:	El sistema registra un espacio de estacionamiento como ocupado cuando detecta un vehículo en la zona correspondiente.
Precondiciones:	El sistema debe haber detectado un vehículo en la zona
Postcondiciones:	El espacio de estacionamiento correspondiente se marca como ocupado.
Flujo Principal:	<ul style="list-style-type: none"> • El sistema detecta un vehículo en la zona de estacionamiento. • El sistema verifica que el vehículo está dentro de los límites de la zona. • El sistema registra el espacio como ocupado. • Se actualiza la información de espacios disponibles en tiempo real.
Flujos Alternativos:	Si el vehículo se mueve fuera de la zona antes de ser registrado, el sistema no actualiza la información.

Tabla 11: Caso de Uso: Registrar Espacio Ocupado

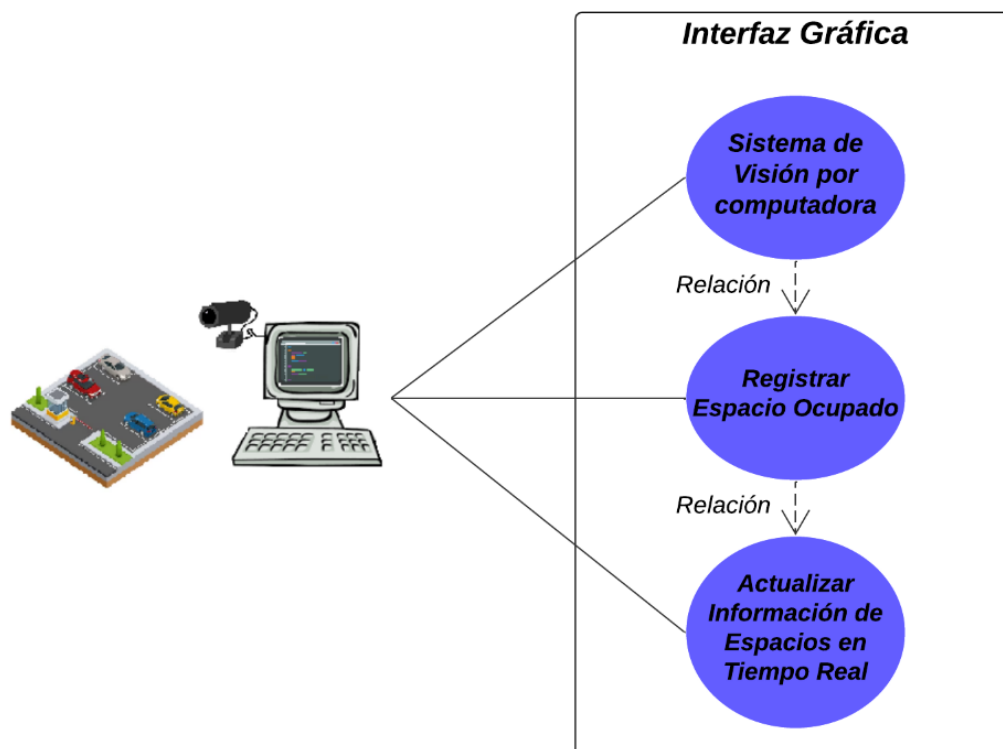


Figura 21: Diagrama de Caso de Uso: Registrar Espacio Ocupado

Caso de Uso: Registrar Espacio Libre

Nombre del Caso de Uso: Registrar Espacio Libre	
Actor Principal:	Sistema de Detección y Usuario
Descripción:	El sistema registra un espacio de estacionamiento como ocupado cuando detecta un vehículo en la zona correspondiente.
Precondiciones:	<ul style="list-style-type: none"> • El sistema de videovigilancia debe estar en funcionamiento. • La cámara debe estar correctamente posicionada y calibrada para capturar la zona de estacionamiento. • El modelo de detección de vehículos debe estar activo y funcionando.
Postcondiciones:	<ul style="list-style-type: none"> • El número de espacios libres se actualiza en la interfaz de usuario.

	<ul style="list-style-type: none"> • La información sobre los espacios libres se almacena para su posterior análisis y reporte.
Flujo Principal:	<ul style="list-style-type: none"> • El Sistema de Detección captura las imágenes del área de estacionamiento. • Procesa las imágenes utilizando el modelo de detección de vehículos. • Identifica los vehículos en las imágenes y determina las áreas ocupadas. • Compara las áreas ocupadas con las áreas predefinidas de estacionamiento. • Cuenta los espacios libres disponibles. • Actualiza el número de espacios libres en la interfaz de usuario en tiempo real. • Almacena la información sobre los espacios libres en un archivo de reporte.
Flujos Alternativos:	Si el Sistema de Detección no puede procesar las imágenes (por fallas en la cámara o el modelo), se notifica al administrador del sistema para que tome medidas correctivas.

Tabla 12: Caso de Uso: Registrar Espacio Libre

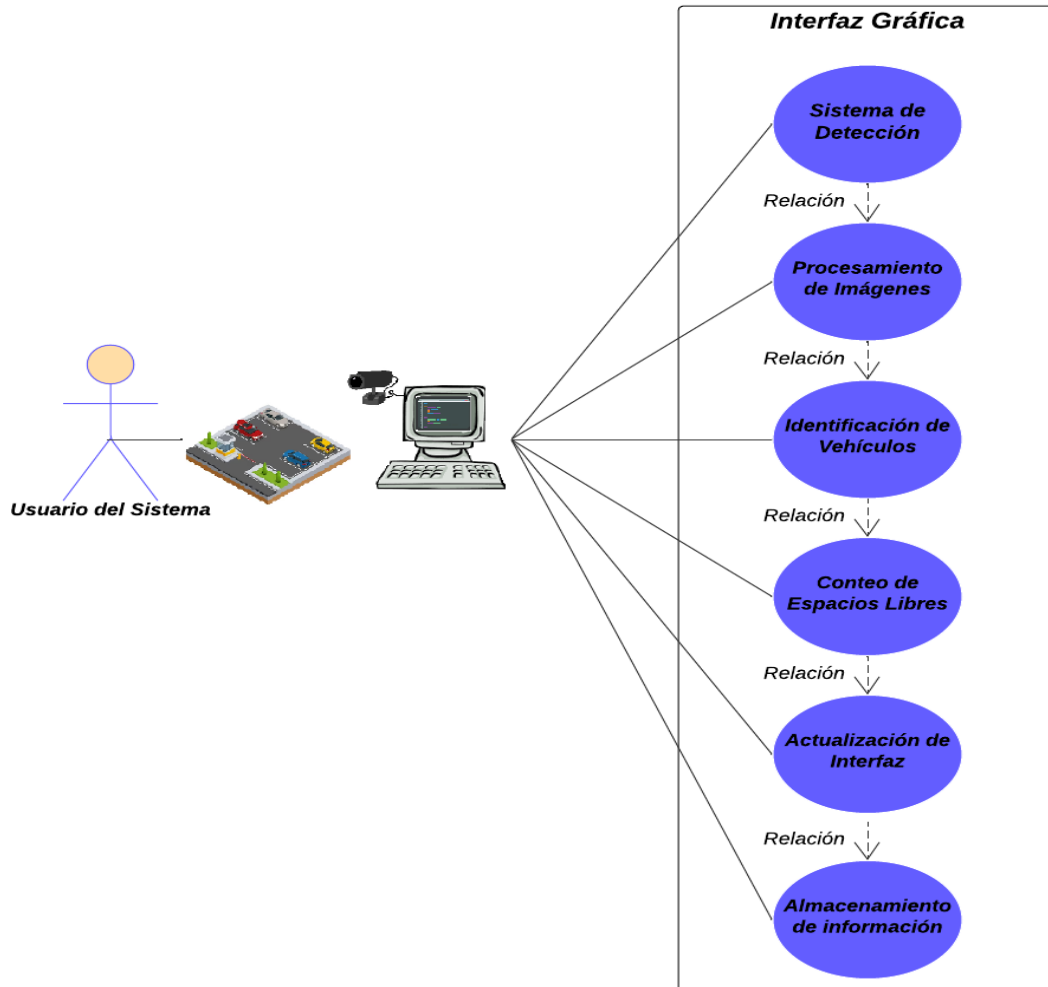


Figura 22: Diagrama Caso de Uso: Registrar Espacio Libre

Caso de Uso: Consultar Disponibilidad de Espacios

Nombre del Caso de Uso: Detectar Vehículo	
Actor Principal:	Sistema de Detección y Usuario
Descripción:	El usuario consulta la disponibilidad de espacios de estacionamiento en tiempo real a través de una interfaz.
Precondiciones:	El sistema de información debe estar operativo.
Postcondiciones:	El usuario obtiene información actualizada sobre los espacios disponibles.

Flujo Principal:	<ul style="list-style-type: none"> • El usuario accede a la interfaz de información. • El sistema muestra la información de espacios disponibles y ocupados. • El usuario visualiza la información y toma una decisión sobre dónde estacionar.
Flujos Alternativos:	Si el sistema de información no está disponible, el usuario recibe un mensaje de error.

Tabla 13: Caso de Uso: Consultar Disponibilidad de Espacios

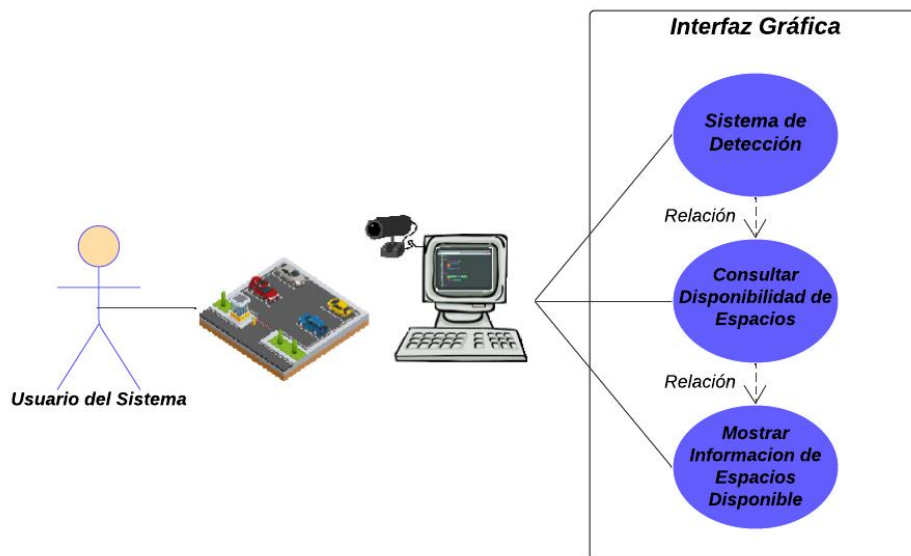


Figura 23: Diagrama de Caso de Uso: Consultar Disponibilidad de Espacios

Caso de Uso: Actualizar Información de Espacios en Tiempo Real

Nombre del Caso de Uso: Actualizar Información de Espacios en Tiempo Real	
Actor Principal:	Sistema de Detección y Usuario
Descripción:	Este caso de uso permite que los usuarios reciban datos precisos y actualizados continuamente.
Precondiciones:	El sistema de información debe estar operativo.

Postcondiciones:	<ul style="list-style-type: none"> • La información sobre los espacios libres se actualiza continuamente en la interfaz de usuario. • Los datos se guardan para su posterior análisis y generación de informes.
Flujo Principal:	<ul style="list-style-type: none"> • El Sistema de Detección captura imágenes en intervalos regulares. • Procesa, identifica los vehículos y las áreas ocupadas utilizando el modelo de detección de vehículos. • Calcula los espacios libres. • Actualiza la información de espacios libres en la interfaz de usuario en tiempo real. • Almacena la información actualizada en la base de datos.
Flujos Alternativos:	<p>Si el sistema de detección falla (por ejemplo, por una cámara defectuosa), se notifica al administrador del sistema para que intervenga.</p>

Tabla 14: Caso de Uso: Actualizar Información de Espacios

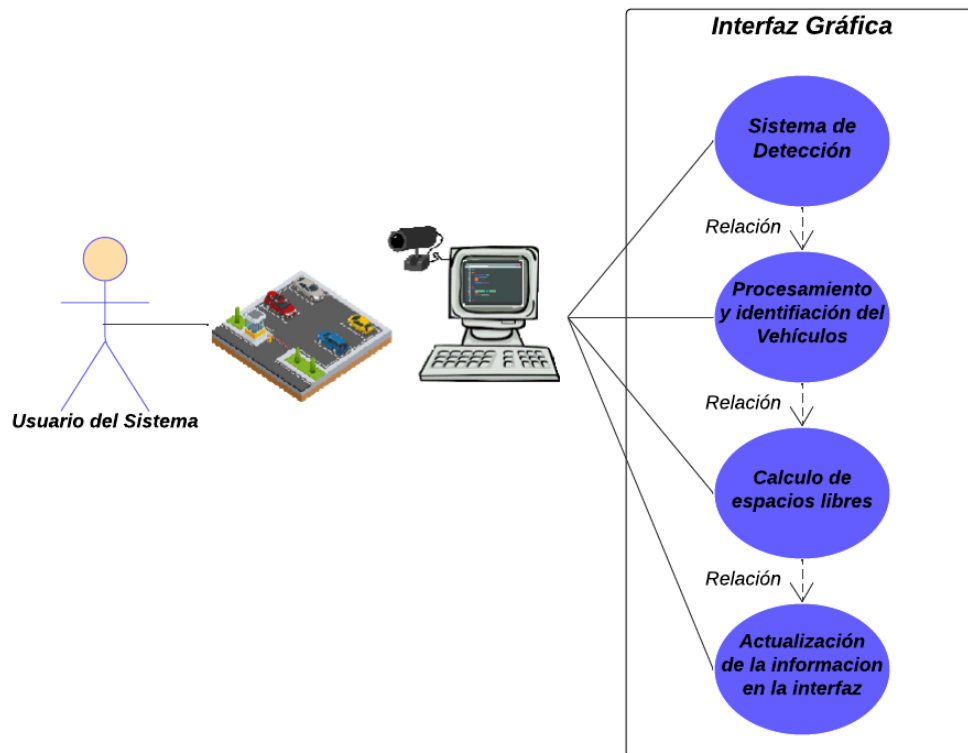


Figura 24: Diagrama de Caso de Uso: Actualizar Información de Espacios

Estos casos de uso detallan cómo el sistema de gestión de estacionamientos opera para mantener y proporcionar información precisa sobre la disponibilidad de espacios, asegurando una experiencia eficiente y efectiva para los usuarios y el administrador del estacionamiento.

2.5.3 Modelado De Datos

El modelado de datos es una parte crucial del diseño de sistemas, ya que define cómo se estructuran, almacenan y gestionan los datos necesarios para el funcionamiento del sistema. En el contexto del sistema de gestión de estacionamientos basado en visión por computadora, es esencial crear un modelo de datos que capture todos los elementos necesarios para monitorear y reportar la disponibilidad de espacios de estacionamiento.

Entidades Principales

1. Vehículo

- Atributos:

- **ID_Vehículo:** Identificador único del vehículo.
- **Tipo:** Tipo de vehículo (automóvil, motocicleta, ...).

- **Hora_Entrada:** Hora en que el vehículo ingresó al estacionamiento.
- **Hora_Salida:** Hora en que el vehículo salió del estacionamiento.

2. Espacio

- Atributos:

- **ID_Espacio:** Identificador único del espacio de estacionamiento.
- **Estado:** Estado del espacio (libre, ocupado).
- **Ubicación:** Coordenadas o descripción de la ubicación del espacio dentro del estacionamiento.

3. Detección

- Atributos:

- **ID_Detección:** Identificador único de la detección.
- **ID_Vehículo:** Referencia al identificador del vehículo detectado.
- **ID_Espacio:** Referencia al identificador del espacio detectado.
- **Hora_Detección:** Hora en que se realizó la detección.
- **Confianza:** Nivel de confianza de la detección del vehículo.

4. Administrador

- Atributos:

- **ID_Administrador:** Identificador único del administrador.
- **Nombre:** Nombre del administrador.
- **Correo:** Correo electrónico del administrador.
- **Teléfono:** Número de teléfono del administrador.

Relaciones Entre Entidades

- Vehículo y Espacio:

- Un vehículo puede ocupar un espacio en un momento dado.
- Un espacio puede ser ocupado por un vehículo o estar libre.

- Detección y Vehículo:

- Una detección se asocia con un vehículo específico.

- Un vehículo puede tener múltiples detecciones a lo largo del tiempo.

- Detección y Espacio:

- Una detección se asocia con un espacio específico.
- Un espacio puede estar asociado con múltiples detecciones a lo largo del tiempo.

Descripción del Diagrama

- **Vehículo:** Esta entidad almacena la información relacionada con cada vehículo que entra y sale del estacionamiento.

- **Espacio:** Esta entidad contiene la información sobre los espacios de estacionamiento, incluyendo su estado y ubicación.

- **Detección:** Esta entidad guarda los registros de detección de vehículos por el sistema de visión por computadora, incluyendo la referencia al vehículo y al espacio de estacionamiento, así como la hora y el nivel de confianza de cada detección.

- **Administrador:** Esta entidad gestiona la información de los administradores del estacionamiento, quienes tienen acceso a la gestión del sistema.

Diagrama de Entidad-Relación (ER)

A continuación, se presenta el diagrama de entidad-relación que ilustra las entidades y sus relaciones:

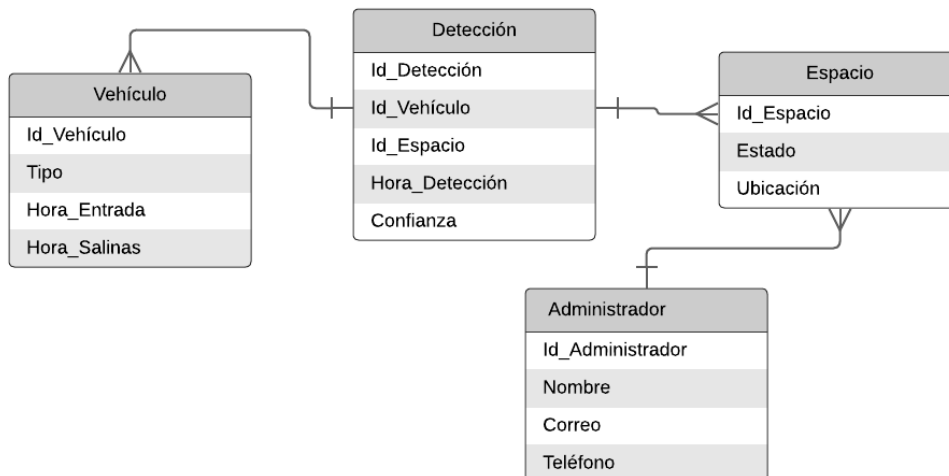


Figura 25: Diagrama de Entidad-Relación

Consideraciones

Integridad de los Datos: Es esencial implementar restricciones de integridad para asegurar que las relaciones entre las entidades sean consistentes. Por ejemplo, cada detección debe estar vinculada a un vehículo y a un espacio válidos.

Rendimiento: La base de datos debe estar optimizada para manejar consultas frecuentes y en tiempo real, especialmente para las actualizaciones de la disponibilidad de espacios.

Seguridad: La información, especialmente la relacionada con los administradores, debe estar protegida mediante mecanismos de seguridad adecuados para prevenir accesos no autorizados.

Este modelo de datos proporciona una base sólida para la implementación del sistema de gestión de estacionamientos, asegurando que toda la información relevante se capture y se gestione de manera eficiente.

2.4 Diseño De Interfaz

El diseño de interfaz es un componente fundamental en el desarrollo de aplicaciones, ya que influye directamente en la experiencia del usuario y en la eficiencia del sistema. En esta sección, se detallan las decisiones y estrategias adoptadas para mejorar la interfaz de usuario del sistema de gestión de estacionamiento.

El objetivo principal del diseño de interfaz es proporcionar a los usuarios una experiencia intuitiva y eficiente al interactuar con el sistema de gestión de estacionamiento. Se busca optimizar la usabilidad, la accesibilidad y la eficacia de la aplicación, garantizando una navegación clara y una presentación efectiva de la información.

Principios de Diseño

Organización Visual:

- Se agruparon elementos relacionados para mejorar la comprensión y la navegación dentro de la aplicación.
- Los elementos se alinearon y distribuyeron uniformemente para lograr una apariencia ordenada y coherente.

Usabilidad:

- Se emplearon iconos y símbolos reconocibles para representar las funciones principales, facilitando la comprensión y la interacción del usuario.
- Se implementó retroalimentación visual para informar al usuario sobre el resultado de sus acciones, mejorando la experiencia de uso.

Navegación:

- Se incorporaron opciones de navegación clara para acceder fácilmente a diferentes partes de la aplicación, como la visualización de reportes y la configuración.
- Se mantuvo la navegación coherente en todas las secciones de la aplicación, proporcionando una experiencia de usuario consistente.

Personalización:

- Se ofrecieron opciones de personalización para adaptar la interfaz a las preferencias individuales de los usuarios, como el tamaño de fuente o el tema de color.
- Se simplificaron los procesos de configuración y ajustes de la aplicación para satisfacer las necesidades específicas de cada usuario.

Diseño Responsivo:

- La interfaz se diseñó para ser compatible con diferentes tamaños de pantalla y dispositivos, asegurando una experiencia de usuario óptima en cualquier contexto.
- Se garantizó que la interfaz fuera escalable y se adaptara fluidamente a diversas resoluciones de pantalla.

Implementación

La aplicación de los principios de diseño se tradujo en diversas mejoras en la interfaz de usuario, incluyendo:

- Agrupación de controles relacionados en secciones claras y distinguibles.
- Utilización de iconos intuitivos para funciones como la descarga y visualización de reportes.
- Mejora de la disposición de los elementos para una apariencia más ordenada y fácil de entender.

- Incorporación de una barra de navegación para facilitar el acceso a diferentes secciones de la aplicación.
- Posibilidad de personalizar la interfaz mediante la configuración de preferencias de usuario.

Estas mejoras contribuyen a una interfaz más eficiente y atractiva para los usuarios, proporcionando una experiencia más agradable y productiva al interactuar con el sistema de gestión de estacionamiento.

Interfaz de Usuario

La interfaz de usuario (UI) del sistema de gestión de estacionamiento presenta una disposición intuitiva diseñada para proporcionar una experiencia fluida y eficiente al usuario. A continuación, se describe la interfaz junto con los elementos que la componen:

Pantalla Principal

La pantalla principal de la aplicación muestra una variedad de información esencial relacionada con el estacionamiento y la gestión de vehículos. Los elementos clave de esta pantalla incluyen:

- ❖ **Título:** En la parte superior de la pantalla, se muestra el título "Estacionamiento General" en una fuente clara y legible para identificar rápidamente el propósito de la aplicación.
- ❖ **Vista de Video:** A la izquierda de la pantalla, se muestra una vista de video en tiempo real del área de estacionamiento. Esta vista proporciona una visión general del estado actual del estacionamiento, permitiendo al usuario monitorear la actividad de los vehículos.
- ❖ **Panel de Información:** A la derecha de la pantalla, se encuentra un panel de información que presenta datos importantes sobre el estacionamiento, incluyendo:
 - **Número de Autos:** Indica la cantidad actual de vehículos estacionados en el área.
 - **Hora Actual:** Muestra la hora actual del sistema para que el usuario pueda tener una referencia temporal
 - **Espacios Libres:** Informa sobre el número de espacios de estacionamiento disponibles en el área.

Controles y Funcionalidades

Junto con la visualización de la información, la interfaz también proporciona controles y funcionalidades para que el usuario interactúe con la aplicación de manera efectiva:

Botón de Descargar Reporte: Permite al usuario generar un informe detallado sobre el registro de vehículos, que incluye información como la hora de entrada y salida, la ubicación y la confianza de la detección.

Botón de Ver Reporte: Proporciona acceso rápido a la visualización del informe generado en formato Excel, permitiendo al usuario explorar los datos de manera más detallada.

Icono de la Aplicación: En la parte inferior del panel de información, se muestra el icono de la aplicación para una identificación visual rápida y fácil.

Captura de Interfaz

A continuación, se muestra una captura de pantalla de la interfaz en ejecución, que ilustra los elementos mencionados anteriormente:

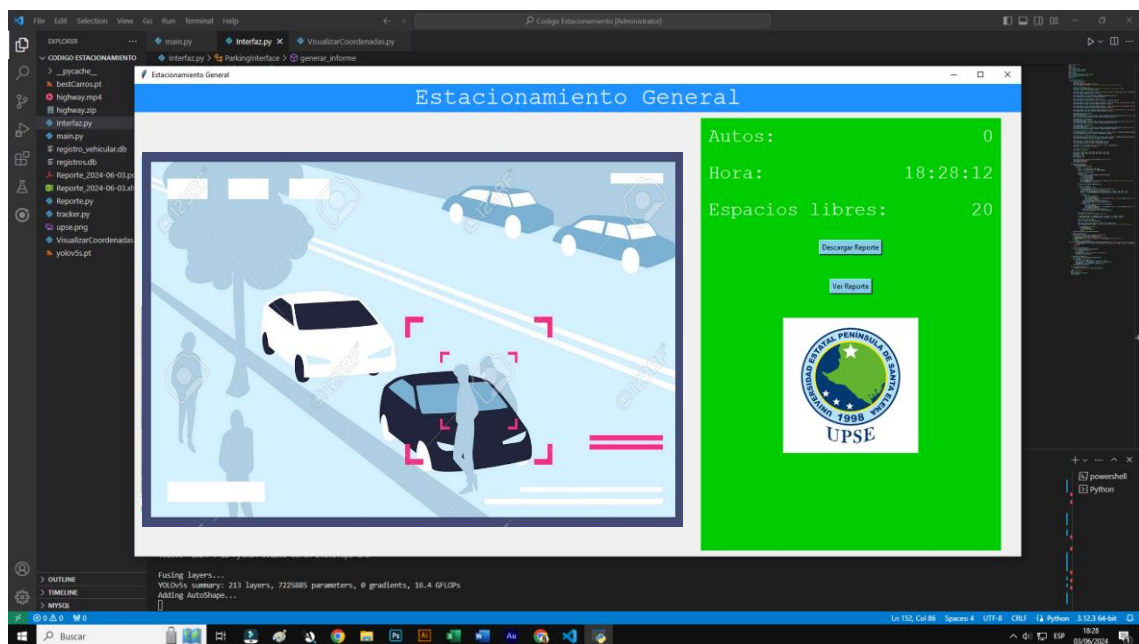


Figura 26: Captura de interfaz

2.6 Pruebas

Se llevaron a cabo pruebas exhaustivas para evaluar el desempeño del algoritmo de detección y seguimiento de vehículos implementado en la interfaz de estacionamiento. Estas pruebas se realizaron utilizando una variedad de casos de prueba diseñados para evaluar la precisión y la estabilidad del sistema en diferentes condiciones.

2.6.1 Configuración de las Pruebas

Se configuraron varios escenarios de prueba que simulan diferentes situaciones en un entorno de estacionamiento, incluyendo diferentes condiciones de iluminación, variaciones en el flujo de vehículos y cambios en la perspectiva de la cámara. Además, se establecieron áreas de interés específicas dentro del espacio de estacionamiento para evaluar la precisión de la detección y seguimiento de vehículos.

2.6.2 Procedimiento de Prueba

Durante las pruebas, se registraron datos relevantes, como la cantidad de vehículos detectados, la precisión de la detección y seguimiento, y el rendimiento del sistema en tiempo real. Se realizaron múltiples repeticiones de cada prueba para garantizar la consistencia de los resultados y se documentaron cualquier anomalía o comportamiento inesperado observado durante las pruebas.

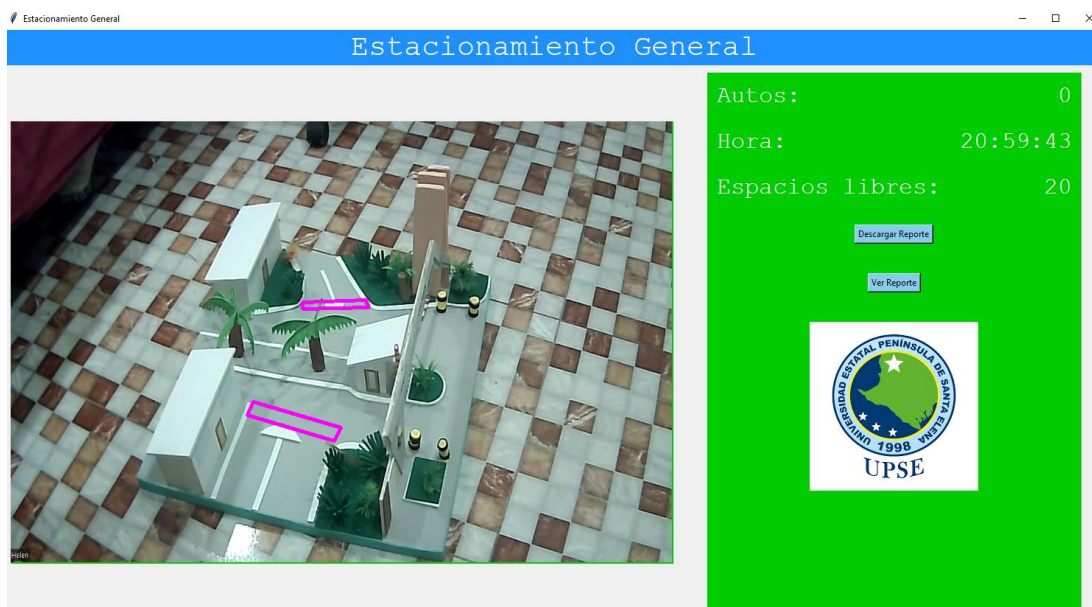


Figura 27: Algoritmo de estacionamiento

2.6.3 Visualización de Coordenadas

Durante las pruebas, se implementó una funcionalidad para visualizar las coordenadas de puntos de interés en el área de estacionamiento.

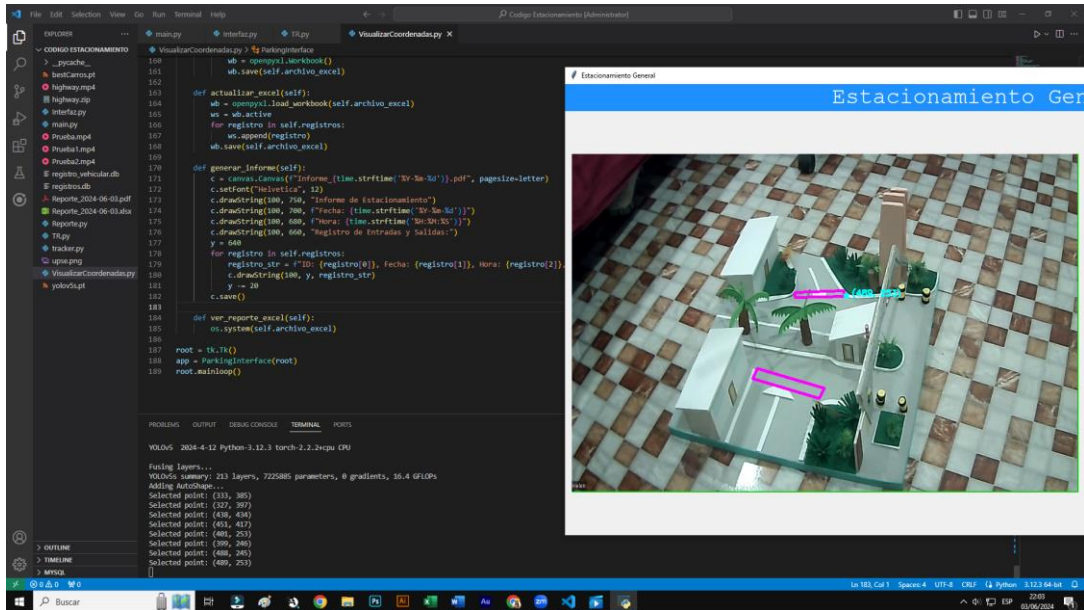


Figura 28: Resultado de Visualización de Coordenadas

Esta funcionalidad permitió a los usuarios seleccionar puntos específicos en la imagen y visualizar las coordenadas correspondientes. La efectividad de esta característica se demostró al mostrar las coordenadas correctamente en la interfaz de usuario.

2.6.4 Resultados Esperados

Se esperaba que el sistema demostrara una capacidad sólida para detectar y seguir vehículos en tiempo real, junto con una funcionalidad efectiva para visualizar las coordenadas de puntos de interés. Se anticipaba que el algoritmo de detección y seguimiento proporcionaría resultados precisos y estables, con una mínima tasa de falsos positivos y falsos negativos.

2.7 Resultados

Los resultados de las pruebas proporcionan una evaluación integral del desempeño del algoritmo de detección y seguimiento de vehículos implementado en la interfaz de estacionamiento. A continuación, se presentan los principales hallazgos y observaciones derivados de las pruebas realizadas:

2.7.1 Precisión de la Detección

El algoritmo demostró una alta precisión en la detección de vehículos, logrando identificar con éxito la mayoría de los vehículos presentes en el área de interés del estacionamiento.

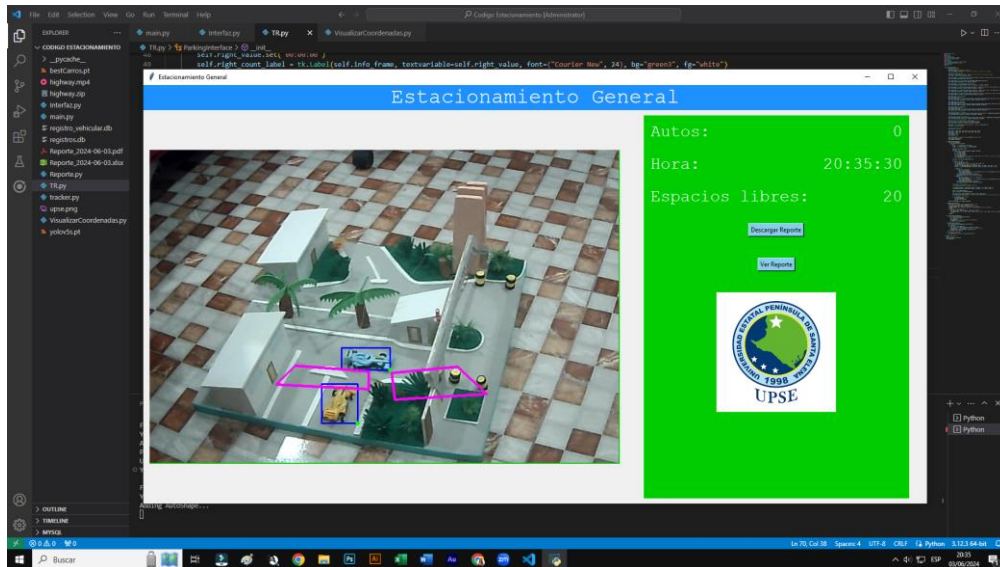


Figura 29: Resultado de Precisión de la Detección

La detección fue robusta incluso en condiciones de iluminación variable y cambios en la perspectiva de la cámara.

2.7.3 Utilidad de los Botones

Los botones implementados en la interfaz demostraron ser altamente útiles para generar informes en tiempo real.

Reporte_2024-06-03	03/06/2024 22:06	Documento Adob...	2 KB	14	52 22:05:01	0,9 (370, 450)
Reporte_2024-06-03	03/06/2024 22:06	Archivo XLSX	6 KB	15	33 22:05:03	0,9 (389, 424)
TR	03/06/2024 21:14	Python File	9 KB	16	46 22:05:15	0,9 (389, 424)
tracker	12/04/2024 18:19	Python File	2 KB	17	5 22:04:42	0,9 (384, 440)
upse	03/06/2024 18:08	Archivo PNG	10 KB	18	8 22:04:55	0,9 (394, 425)
VisualizarCoordenadas	03/06/2024 20:59	Python File	9 KB	19	52 22:05:01	0,9 (370, 450)
yolov5s	12/04/2024 18:32	Archivo PT	14.462 KB	20	33 22:05:03	0,9 (389, 424)
				21	46 22:05:15	0,9 (389, 424)

Figura 30: Resultado de Archivos Generados

2.7.2 Estabilidad del Seguimiento

El sistema exhibió una estabilidad notable en el seguimiento de vehículos a lo largo del tiempo. La capacidad del algoritmo para rastrear vehículos en movimiento se mantuvo

consistente, con una mínima pérdida de seguimiento incluso en escenarios complejos con múltiples vehículos presentes en la escena.

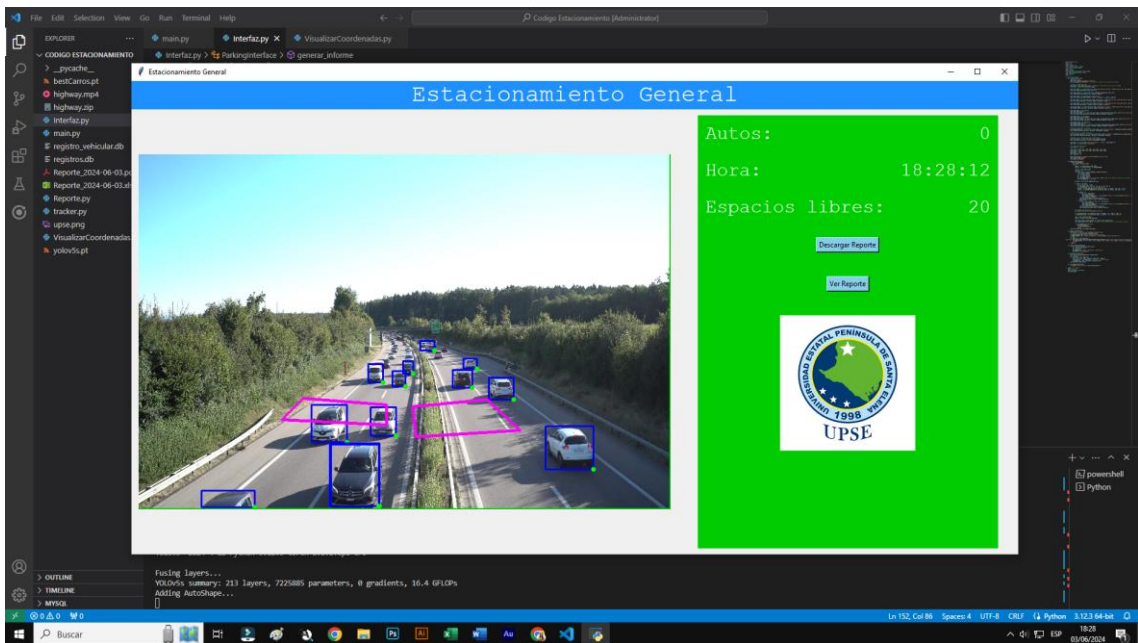


Figura 31: Resultado de Estabilidad del Seguimiento

El botón "Descargar Reporte" creó un archivo PDF que incluye registros actualizados de vehículos.

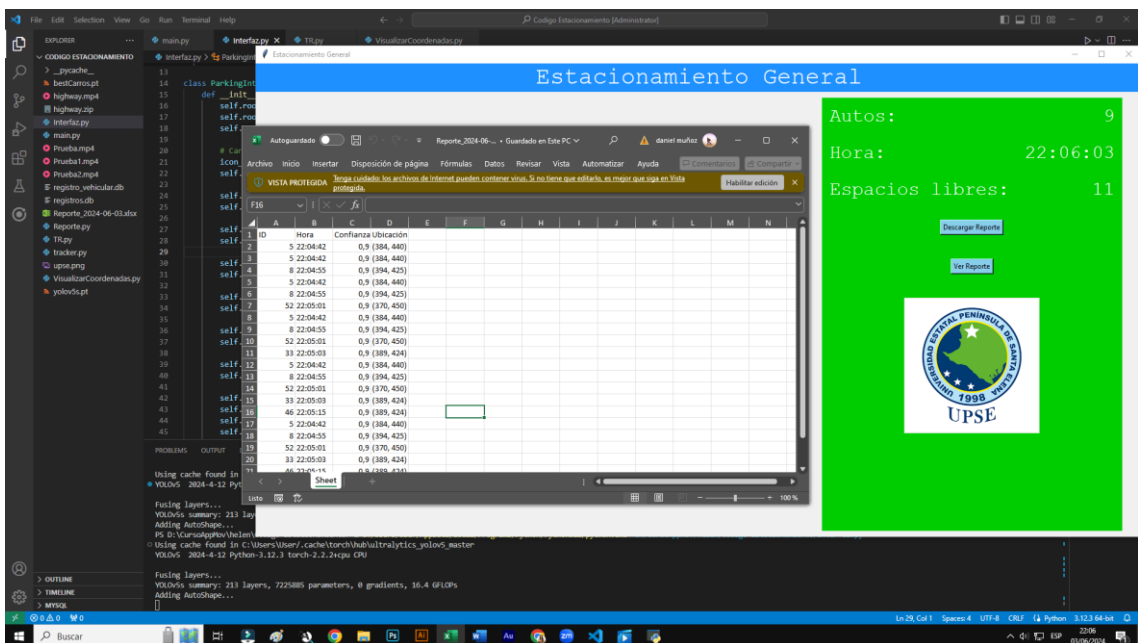


Figura 32: Resultado botón "Descargar Reporte"

Mientras que el botón "Ver Reporte" abrió un archivo Excel donde se registraron las entradas de vehículos con hora, confianza y ubicación.

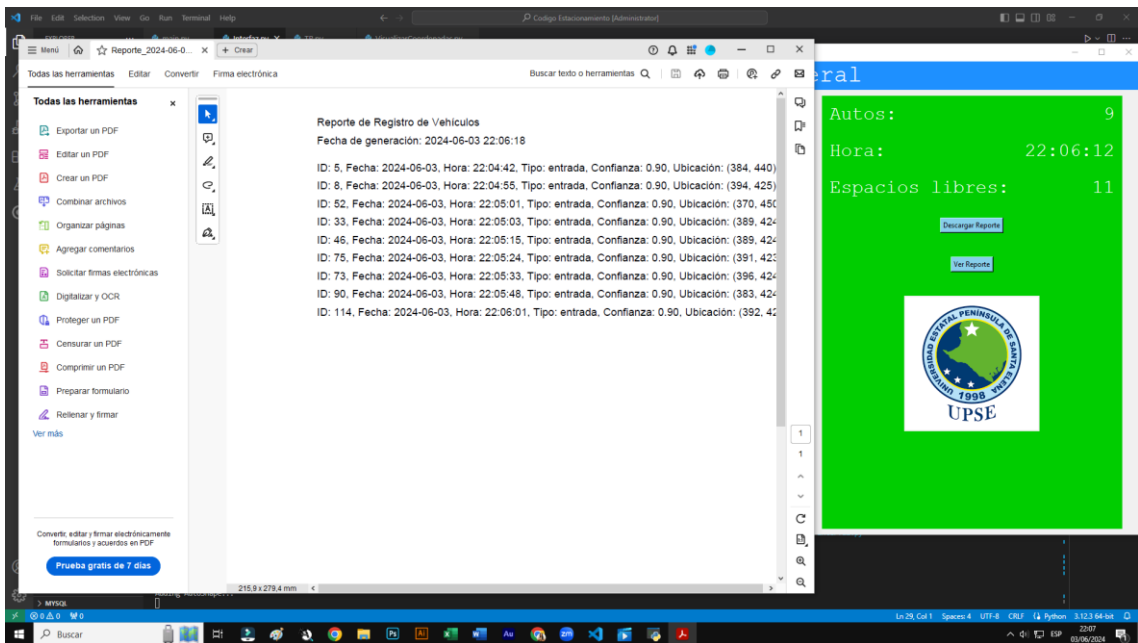


Figura 33: Resultado botón "Ver Reporte"

2.7.4 Anomalías y Limitaciones

Se observaron algunas anomalías menores durante las pruebas, como falsos positivos ocasionales y dificultades para detectar vehículos en ciertas áreas de la escena. Estas limitaciones pueden abordarse mediante ajustes adicionales en los parámetros del algoritmo y la optimización de la configuración de la cámara.



Figura 34: Resultado de prueba 1

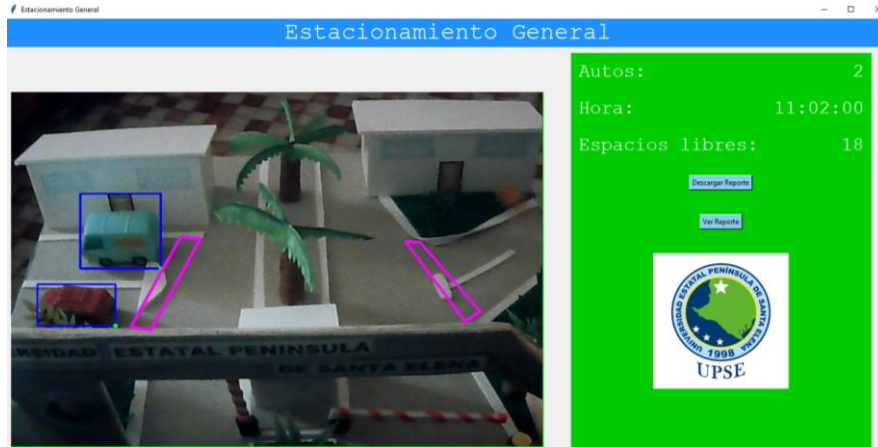


Figura 35: Resultado de prueba 2



Figura 36: Resultado de prueba 3

Conclusiones

- Se identificaron varios enfoques viables para la detección de espacios vacíos en estacionamientos, incluidos los basados en CNN (Convolutional Neural Networks), YOLO (You Only Look Once) y SSD (Single Shot MultiBox Detector), después de una revisión exhaustiva de la literatura sobre algoritmos de visión por computadora. La evaluación de algoritmos demostró que los algoritmos basados en aprendizaje profundo, particularmente YOLOv4 y YOLOv5, ofrecen un equilibrio ideal entre la precisión y la velocidad de detección. Para aplicaciones en tiempo real como la detección de espacios de estacionamiento, este equilibrio es esencial.
- Debido a su alta precisión y eficiencia en la detección de objetos, se implementó un algoritmo en Python utilizando el modelo YOLOv5. El modelo alcanzó una precisión del 82% en la identificación y detección de espacios vacíos en estacionamientos a través de iteraciones y mejoras continuas basadas en pruebas experimentales. Para maximizar la precisión y reducir los falsos positivos y negativos, la optimización del algoritmo incluyó ajustes en los hiperparámetros, métodos de aumento de datos y mejoras en la arquitectura del modelo.
- Las pruebas experimentales realizadas en el parqueadero central de la UPSE validaron la efectividad del sistema desarrollado, demostrando que el algoritmo puede identificar y detectar espacios vacíos con una precisión superior al 80%. Se identificaron algunos desafíos, como la variabilidad en las condiciones de iluminación y la presencia de obstáculos no previstos, que afectaron la precisión del sistema en ciertos escenarios. Los ajustes realizados al algoritmo, basados en los resultados experimentales, incluyeron mejoras en el preprocesamiento de imágenes y técnicas de posprocesamiento para filtrar detecciones erróneas.

Recomendaciones

- Para aumentar la precisión y robustez del sistema, se recomienda continuar con la iteración y mejora del modelo utilizando técnicas de aprendizaje automático y redes neuronales más avanzadas, como modelos basados en transformadores. Implementar un pipeline de monitoreo y actualización automática del modelo para adaptarse a nuevas condiciones y variabilidades del entorno del estacionamiento también sería beneficioso.
- La optimización del algoritmo para su ejecución en hardware específico, como GPUs o TPUs, puede mejorar la velocidad de procesamiento y permitir la implementación en tiempo real. Además, considerar la posibilidad de utilizar técnicas de optimización de inferencia y compresión de modelos puede reducir la carga computacional sin comprometer la precisión.
- Para garantizar que el modelo sea sólido y generalizable a varios escenarios, es crucial realizar pruebas adicionales en una variedad de entornos y condiciones, como variaciones climáticas y tipos de estacionamientos. Además, tenga en cuenta que la integración de datos adicionales, como datos de cámaras y sensores adicionales, puede mejorar la precisión y la confiabilidad del sistema.
- Desarrollar una interfaz de usuario amigable que permita a los administradores del estacionamiento monitorear el sistema y recibir alertas sobre la disponibilidad de espacios en tiempo real es crucial. También se debería explorar la integración del sistema con aplicaciones móviles para proporcionar a los usuarios información en tiempo real sobre la disponibilidad de espacios de estacionamiento, mejorando así la experiencia del usuario.

Referencias

- [1] Aguilar, Jonathan; Rivas, Wilmer; García, Karina, «Visión artificial para control de estacionamiento vehicular,» *Dianlnet*, vol. 6, n° 9, p. 21, 2021.
- [2] Maso Ferreyra, Divad Jair, «Implementacion de un sistema web basado en vision artificial y gobalizacio , y su influencia en la vigilancia del Distrito de Lince,» Lima, 2020.
- [3] Pérex Nasser, Jovann, «Análisis de tráfico vehicular mediante vision artificial,» Ambato, 2019.
- [4] Córdova Limones, Andrea Gabriela, «Propuesta de mejora al acceso de vehículos autorizados y no autorizados mediante el reconocimiento de placas, tratamiento de imagen y automatización al Edificio "El Velero Azul",» La Libertad, 2021.
- [5] C. García, «Desarrollo de proyectos de Internet de las cosas,» 12 06 2021. [En línea]. Available: <https://www.linkedin.com/pulse/desenvolvendo-projetos-de-internet-das-coisas-existe-c%C3%A1ssio-garcia/?originalSubdomain=pt>. [Último acceso: 04 06 2023].
- [6] Phyton, «El tutorial de Phyton,» 26 07 2021. [En línea]. Available: <https://docs.python.org/3/tutorial/index.html>. [Último acceso: 03 06 2026].
- [7] Software Freedom Conservancy, «phpmyadmin,» 2023. [En línea]. Available: <https://www.phpmyadmin.net/>. [Último acceso: 03 06 2023].
- [8] V. C. studio, «Codigo visual code studio en accion,» 2023. [En línea]. Available: <https://code.visualstudio.com/docs>. [Último acceso: 05 06 2023].
- [9] Infaimon, «Inteligencia Artificial y Deep Learning,» Infaimon, 26 11 2020. [En línea]. Available: <https://www.interempresas.net/Robotica/Articulos/320015-La-vision-artificial-en-entorno-cientifico.html> . [Último acceso: 17 06 2023].
- [10] cursosaula21, «Vision Artifical: Todo lo que necesitas saber,» cursosaula21, 2023. [En línea]. Available: <https://www.cursosaula21.com/que-es-la-vision-artificial/>. [Último acceso: 17 06 2023].
- [11] Ecuador, «Plan de creacion de oportunidades 2021-2025,» 2021. [En línea]. Available: <https://www.planificacion.gob.ec/wp-content/uploads/2021/09/Plan->

de-Creacio%CC%81n-de-Oportunidades-2021-2025-Aprobado.pdf. [Último acceso: 05 12 2023].

- [12] C. Ramos Galarza, Los alcances de una investigacion, Quito: CienciAmerica, 2020.
- [13] D. J. Masgo Ferreyra, «Implementación de un sistema web basado en visión artificial y geolocalización, y su influencia en la vigilancia del distrito de Lince,» Lima, 2020.
- [14] J. Pérez Nasser, «Análisis de tráfico vehicular mediante visión artificial,» Ambato, 2019.
- [15] A. G. Córdova Limones, «Propuesta de mejora al acceso de vehículos autorizados y no autorizados mediante el reconocimiento de placas, tratamiento de imagen y automatización al Edificio "El Velero Azul",» La Libertad, 2021.
- [16] D. Ortiz Negrin, EL MODELO INCREMENTAL PARA EL DESARROLLO DE UN SISTEMA INFORMÁTICO DE GESTIÓN DE INFORMACIÓN, Granma-Cuba: Centro Universitario Municipal de Media Luna de la Universidad, 2020.
- [17] UPSE, «Misión y Visión,» *Universidad Estatal Península De Santa Elena*, p. https://www.upse.edu.ec/index.php?option=com_content&view=article&id=12&Itemid=167, 14 Diciembre 2021.
- [18] Unir, «Unir - La visión artificial: una revolución en la Industria 4.0,» 2021. [En línea]. Available: <https://www.unir.net/ingenieria/revista/vision-artificial/>. [Último acceso: 11 12 2023].
- [19] Amazon Web Services, «AWS - ¿Qué es la visión artificial?,» 2023. [En línea]. Available: <https://aws.amazon.com/es/what-is/computer-vision/>. [Último acceso: 11 12 2023].
- [20] P. Londoño, «Inteligencia artificial: qué es, cómo funciona e importancia en 2023,» Hubspot.es, 6 Febrero 2023. [En línea]. Available: <https://blog.hubspot.es/marketing/inteligencia-artificial-esta-aqui>.
- [21] O. Garcia, «Redes Neuronales artificiales: Qué son y cómo se entrenan,» Xeridia, 16 septiembre 2019. [En línea]. Available: <https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i>.


- [22] B. Perry, «Redes neuronales convolucionales (CNN),» Fineproxy, 21 Mayo 2023. [En línea]. Available: <https://fineproxy.org/es/wiki/convolutional-neural-networks-cnn/>.
- [23] Aprendizaje automático., Colombia: Ediciones de la U, 2022.
- [24] E. Burns, «Aprendizaje profundo (deep learning),» ComputerWeekly.es, 09 06 2021. [En línea]. Available: <https://www.computerweekly.com/es/definicion/Aprendizaje-profundo-deep-learning>.
- [25] {Daniel}, «Deep learning o Aprendizaje profundo,» Formación en ciencia de datos | DataScientest.com, 04 19 2022. [En línea]. Available: <https://datascientest.com/es/deep-learning-definicion>.
- [26] Prompts, «Guía Completa sobre Detección de Objetos: Algoritmos, Aplicaciones y Herramientas,» 29 febrero 2024. [En línea]. Available: <https://prompt.uno/vision-por-computadora/deteccion-de-objetos-2/>.
- [27] «Algoritmos de visión por computadora: Avances y aplicaciones en la actualidad,» Prompts, 13 Febrero 2024. [En línea]. Available: https://prompt.uno/vision-por-computadora/algoritmos-de-vision-por-computadora/#google_vignette.
- [28] Z. Keita, «Explicación de la detección de objetos YOLO,» datacamp, enero 2024. [En línea]. Available: <https://www.datacamp.com/es/blog/yolo-object-detection-explained>.
- [29] «Los fundamentos y aplicaciones de la visión artificial en la actualidad,» Bcnvision, 25 enero 2024. [En línea]. Available: <https://bcnvision.es/blog-vision-artificial/los-fundamentos-y-aplicaciones-de-la-vision-artificial-en-la-actualidad/>.
- [30] V. Alvear, P. Rosero, D. Peluffo y J. Pijal, «Internet de las Cosas y Visión Artificial, Funcionamiento y Aplicaciones,» Universidad Tecnológica Equinoccial, 2017.
- [31] J. Aguilar, W. Rivas y K. Garcia, «Artificial vision for vehicle parking control,» vol. 6, n° 9, pp. 2134-2154, 2021.

- [32] L. A. Riofrio Peña, «Desarrollo de un entorno gráfico y de controladores predictivos para el modelado y gestión de sistemas interconectados de transporte,» Universidad Politecnica Salesiana, Cuenca - Ecuador, 2021.
- [33] KSchoolLos 10 principios de usabilidad para diseño de interfaces de usuario, 09 09 2021. [En línea]. Available: <https://kschool.com/blog/usabilidad-ux/los-10-principios-de-usabilidad-para-diseno-de-interfaces-de-usuario/>.
- [34] «Modulo 3 Protegiendo sus datos y su privacidad,» Cisco Networking Academy, 2023.
- [35] «Consideraciones Para El Mantenimiento Y La Escalabilidad A Largo Plazo,» fastercapital, 2024. [En línea]. Available: <https://fastercapital.com/es/tema/consideraciones-para-el-mantenimiento-y-la-escalabilidad-a-largo-plazo.html>.
- [36] «Tendencias en visión artificial industrial que transformarán el futuro,» bcvision, 21 12 2023. [En línea]. Available: <https://bcvision.es/blog-vision-artificial/tendencias-en-vision-artificial-industrial/>.
- [37] Stephen O´Gradey, «RedMonk - The RedMonk Programming Language Rankings: June 2021,» 05 08 2021. [En línea]. Available: <https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/>. [Último acceso: 25 03 2024].
- [38] Á. Torrenti, «Procesamiento de imágenes con OpenCV en Python,» Imaginaformacion.com, 04 22 2024. [En línea]. Available: <https://imaginaformacion.com/tutoriales/opencv-en-python>.
- [39] «¿Qué es PyTorch?,» Ibm, 2024. [En línea]. Available: <https://www.ibm.com/es-es/topics/pytorch>.
- [40] «La librería Numpy,» aprendeconalf, 2024. [En línea]. Available: <https://aprendeconalf.es/docencia/python/manual/numpy/>.
- [41] Ultralytics, «Roboflow,» Aplicaciones, 12 11 2023. [En línea]. Available: <https://aplicaciones.ai/roboflow/>.
- [42] D. Urrutia, «Qué es Visual Studio,» Arimetrics, 04 09 2023. [En línea]. Available: <https://learn.microsoft.com/es-es/visualstudio/get-started/visual-basic/visual-studio-ide?view=vs-2022>.

- [43] C. Valencia, j. Muñoz y L. Pencue, «Sistema de asistencia a la conducción usando visión por computadora y aprendizaje máquina,» vol. 29, n° 54, 2020.
- [44] «Fusion de sensores integracion de DTCT con multiples tecnologias,» Faster Capital, 10 Marzo 2024. [En línea]. Available: <https://fastercapital.com/es/contenido/Fusion-de-sensores--integracion-de-DTCT-con-multiples-tecnologias.html>.
- [45] F. A. Lopez, «La identificación de automóviles en áreas de aparcamiento es un desafío técnico que requiere el uso de diversas técnicas y teorías. Una de las teorías fundamentales en este contexto es la teoría de procesamiento de imágenes, que proporciona los fundamentos,» Universidad Autonoma de Barcelona, Europa, 2016.
- [46] D. Aldas, S. Collantes y J. Reyes, «PROCESAMIENTO DE IMÁGENES CON VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO DE PLACAS VEHICULARES,» 26 06 2016. [En línea].
- [47] H. Diaz, D. Sanabria y J. Ortiz, «Tendencia mundial en tecnologías de sistemas,» 11 06 2018. [En línea].
- [48] C. E. Sandra Narváez, «PROTOTIPO DE DETECCION DE APARCAMIENTOS LIBRES MEDIANTE VISION ARTIFICIAL EN UN PARQUEADERO DE LA UNIVERSIDAD TECNICA DEL NORTE,» Universidad Técnica del Norte, Ibarra - Ecuador, 2019.
- [49] P. Méndez y J. Ibarra, «Implementación de una red neuronal de convolución para el reconocimiento de poses en imágenes de rostros,» revistas.usfq, 2014. [En línea]. Available: <https://revistas.usfq.edu.ec/index.php/avances/article/view/167>.

ANEXOS

Anexo 1. Encuesta dirigida a la comunidad universitaria que posee vehículos en la Universidad Estatal Península de Santa Elena

	<p style="text-align: center;">Universidad Estatal Península de Santa Elena Facultad de Sistemas y Telecomunicaciones Carrera de Tecnologías de la Información</p>
Encuesta dirigida a los docentes de la Facultad de Sistemas y Telecomunicaciones en la UPSE	
Objetivos: Determinar los inconvenientes existentes en el parqueadero central de la UPSE.	
1.	¿Con qué frecuencia encuentra parqueadero para su vehículo? Nunca ____ Casi nunca ____ Ocasionalmente ____ Siempre ____
2.	¿Cuánto tiempo se demora en encontrar parqueadero? 5 min __ 6 a 10 min __ 11 a 15 min __ Más de 15 min ____
3.	¿Cree usted que el tiempo que le toma encontrar parqueadero, afecta en sus actividades laborales o escolares? Sí ____ No ____
4.	¿Con qué frecuencia se encuentra con congestión vehicular al momento de buscar parqueo? Nunca ____ Casi nunca ____ Frecuentemente ____ Siempre ____
5.	¿Con qué frecuencia usted ingresa y sale de la institución en el día?

	1 vez___ 2 a 3 veces___ 4 a 5 veces___ Más de 5 veces___
6.	¿Cómo calificaría la organización actual en el parqueadero central de la UPSE? Pésima___ Mala___ Regular___ Buena___ Excelente___
7.	¿Qué inconvenientes ha tenido para salir de su lugar de parqueo? Otro vehículo mal parqueado ___ Salida del parqueo obstruida ___ Tráfico vehicular___ Ninguno___
8.	¿Cree usted que sea necesario un sistema que muestre en cada espacio del parqueadero, la disponibilidad de parqueo mediante una pantalla? Sí___ No___
Resumen:	Recolección de información para determinar los inconvenientes presentes en el parqueadero central de la UPSE.
Responsable:	León Rivera Helen Jamel.

*Anexo 2. Observación realizada en el parqueadero central de la Universidad Estatal
Península de Santa Elena*

Registro descriptivo de la información
Fecha: 09 de mayo del 2023
Lugar: La Libertad
Personas: 1
Proceso: Gestión de tráfico en el parqueadero central de la UPSE
Duración: 5 horas

Hechos observados	
<ul style="list-style-type: none"> • El parqueadero central de la UPSE se encuentra ubicado frente a las oficinas de TICS. • En ocasiones, personas ingresan por la mañana al parqueadero, dando varias vueltas sin encontrar lugares disponibles. • Personas al no encontrar parqueo, estacionan en cualquier lugar, muchas veces impidiendo el paso de otros vehículos ya parqueados. • Personal y estudiantes parquean su vehículo al ras de la calle, al no encontrar donde estacionarse, cerca del lugar donde les toca ver o impartir clases. • Las personas al no encontrar zonas disponibles de parqueo, tienen que ir a dejar parqueados sus vehículos en lugares inseguros o donde no puedan vigilarlos. • El inconveniente de no encontrar parqueadero, dificulta la hora de entrada de los docentes y personal administrativo. • Frecuentemente, se puede visualizar tráfico vehicular en los lugares de parqueo, ya que, existe una administración regular en los mismos. 	
Resumen:	Se determinó que, en el área de parqueo no se lleva un control adecuado que permita visualizar si hay o no, lugares disponibles.
Responsable:	León Rivera Helen Jamel.

ANEXO 2

ENTRENAMIENTO MEDIANTE CLASIFICADOR DE OBJETO HAAR

ENTRENAMIENTO MEDIANTE CLASIFICADOR DE OBJETOS HAAR

1. Mediante el lenguaje de programación Python se desarrolló un script esencial para la preparación sobre el conjunto de entrenamiento de clasificador por haar requiriendo la capitulación de imágenes positivas y negativas.

Figura 37: Capturar de imágenes positivas y negativas carpeta "P"

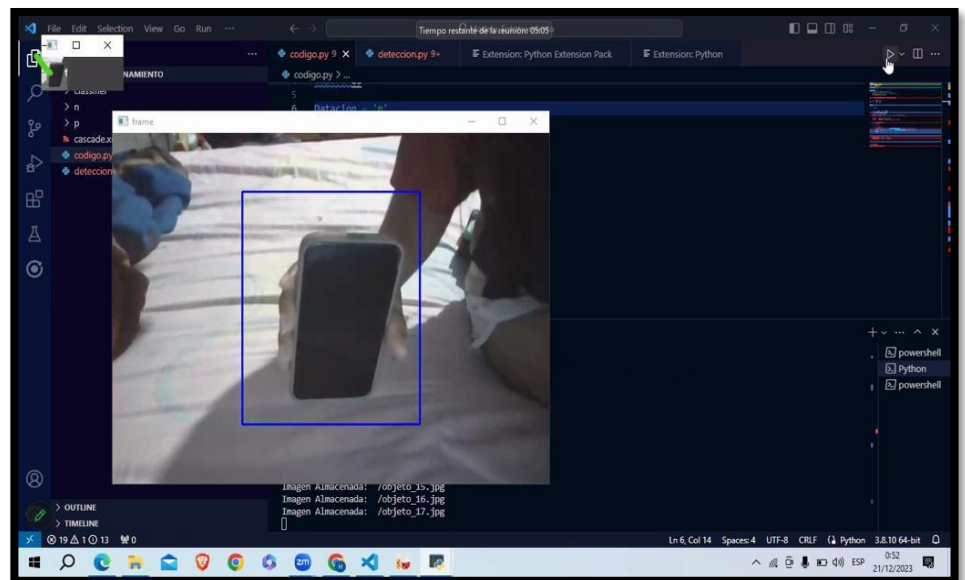
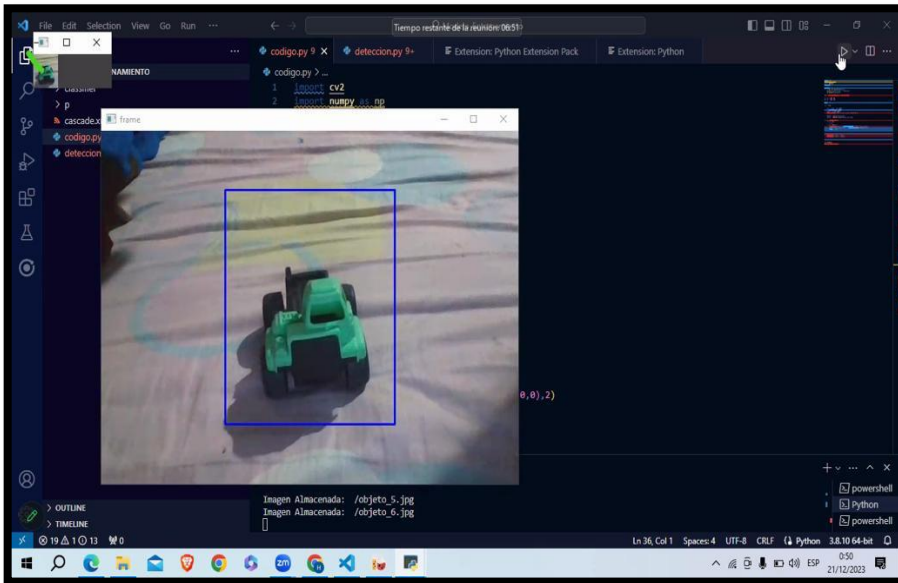


Figura 38: Capturar de imágenes negativas carpeta "P"

2. En total de imágenes capturada con el código fueron alrededor de 40 por cada carpeta

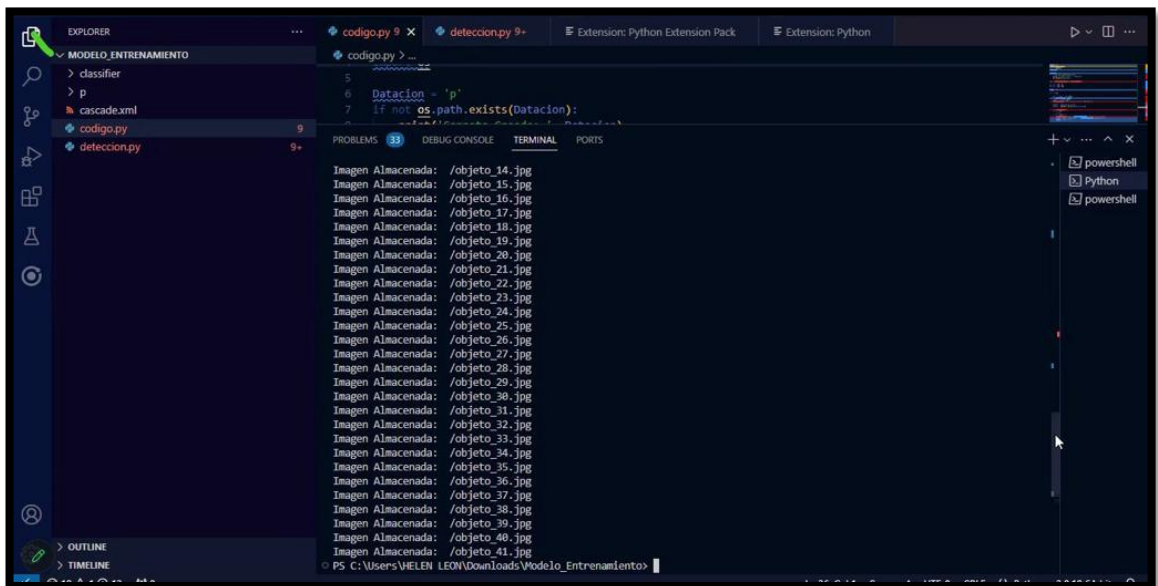


Figura 39: Almacenamiento de imagenes

3. Una vez obtenido la información necesaria toca trabajar con la herramienta Cascade Trainer GUI que cuenta con la disponibilidad necesaria para emplear clasificadores en cascada para el entrenamiento de visión por computadores como es el caso de haar

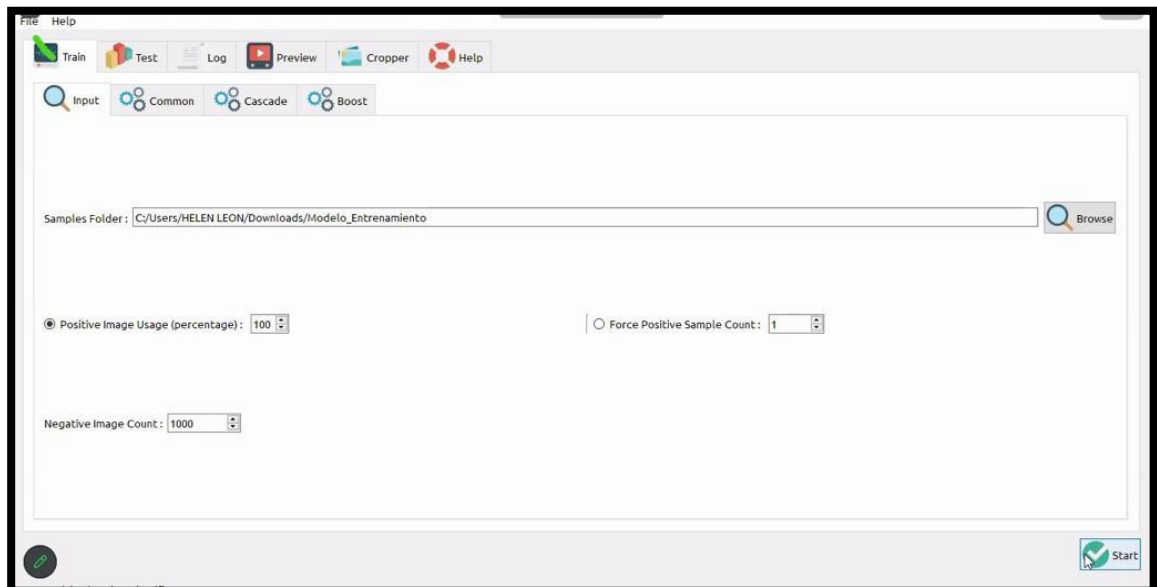


Figura 40: Herramienta Cascade Trainer GUI

4. Configuración predeterminada

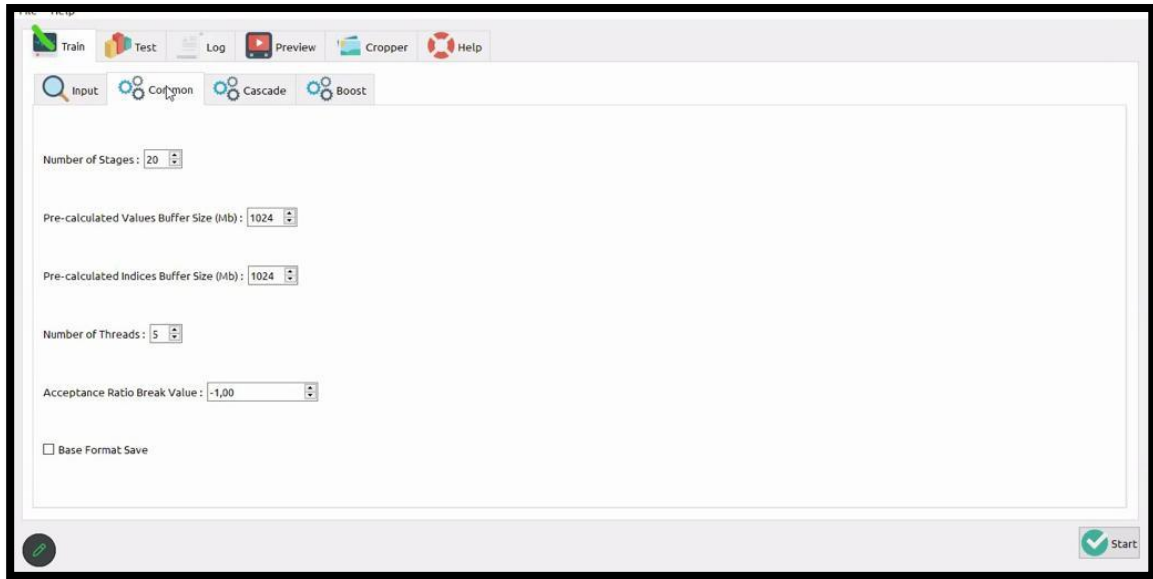


Figura 41: Configuración en Cascade Trainer GUI

5. Configurar tamaño de las imágenes y tener marcado el modo de entrenamiento HAAR

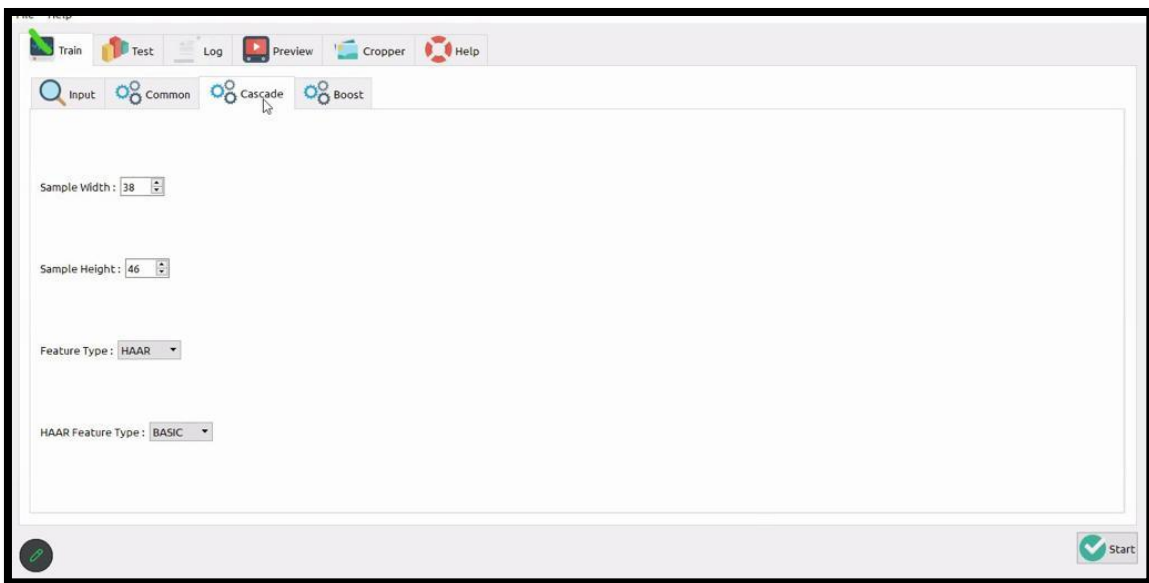


Figura 42: Configuración de tamaño de img y modo en HAAR

6. Proceso de entrenamiento

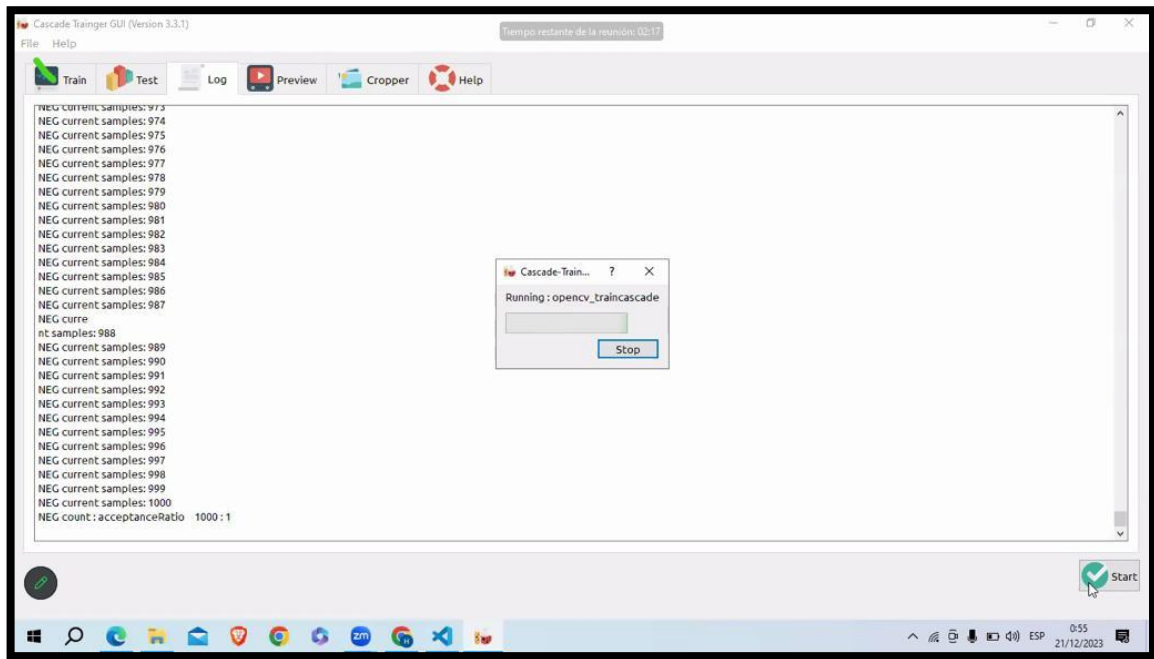


Figura 43: Procesamiento de entrenamiento

7. Entrenamiento finalizado

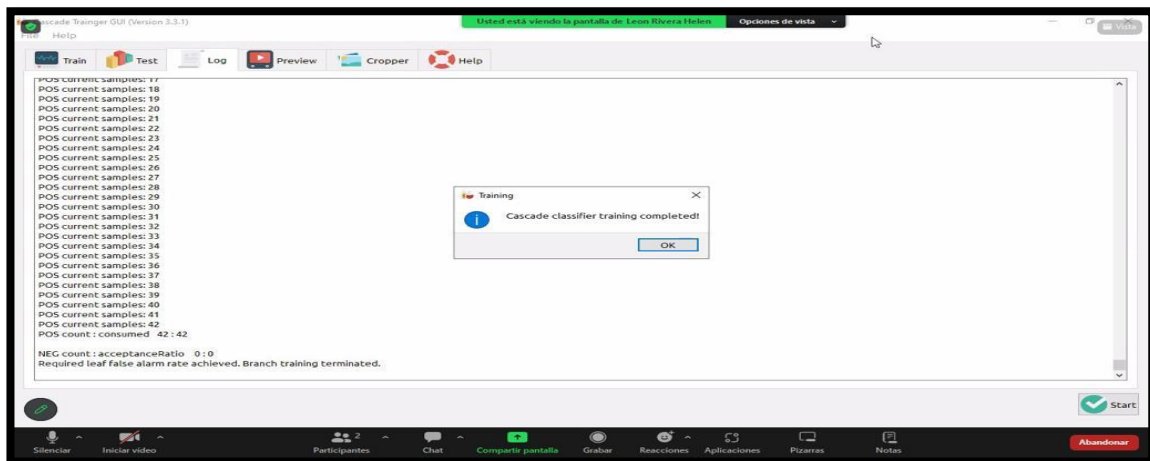


Figura 44: Entrenamiento finalizado

8. La herramienta creó una carpeta classifier en donde se encuentra el archivo cascade.xml que cuenta con la data entrenada proporcionada por las imágenes negativas y positivas.

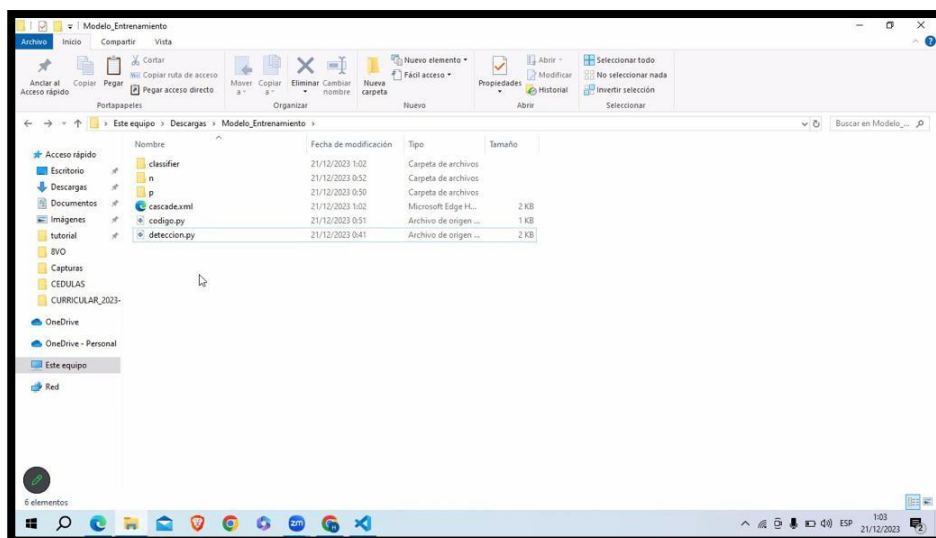


Figura 45: carpeta classifer

- Se crea un script de detección de objeto en tiempo real que permite el uso del cascade.xml para emplear la prueba permanente de su funcionamiento con respecto a la visión por computadora

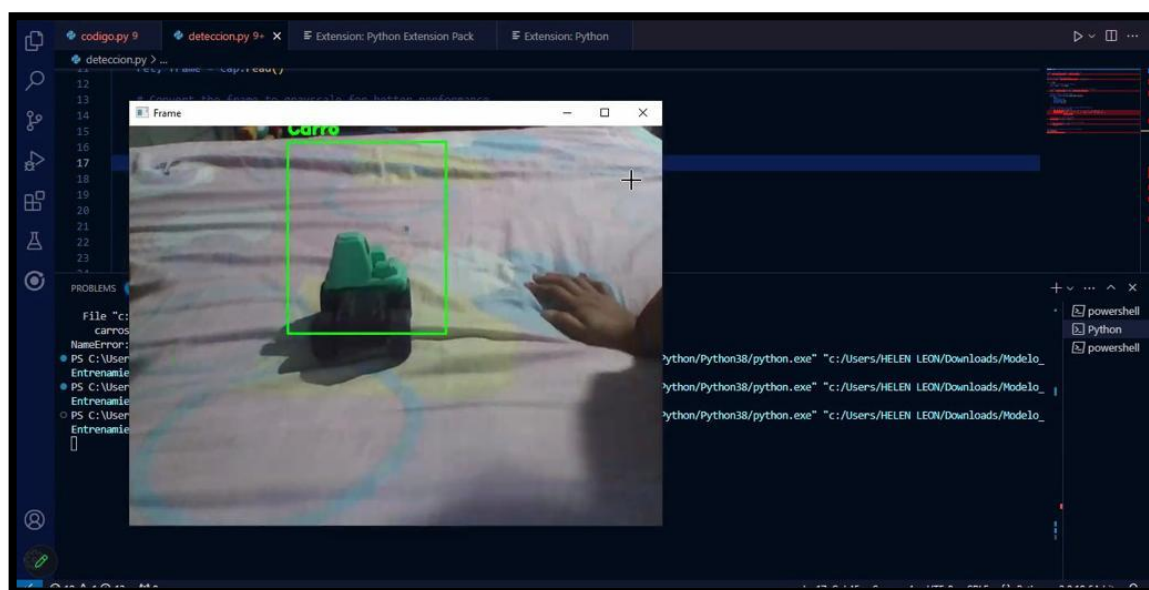


Figura 46: Prueba del script con herramienta IOT

Resultados del Entrenamiento Mediante Clasificador de Objetos Haar con Herramientas IoT

El proceso de entrenamiento utilizando clasificadores de objetos Haar se llevó a cabo para la detección de vehículos en un entorno de visión por computadora, con la integración de

herramientas IoT para mejorar la eficiencia y la conectividad. A continuación, se presentan los resultados obtenidos:

- El script desarrollado en Python para la preparación del conjunto de entrenamiento demostró ser efectivo en la recopilación y etiquetado de imágenes positivas y negativas, proporcionando así un conjunto de datos adecuado para el entrenamiento del clasificador Haar.
- Se capturaron alrededor de 40 imágenes por cada carpeta, lo que garantizó una diversidad suficiente en el conjunto de datos para el entrenamiento del clasificador.
- La herramienta Cascade Trainer GUI facilitó el proceso de entrenamiento al proporcionar una interfaz intuitiva para la configuración y ejecución del entrenamiento del clasificador en cascada. Su disponibilidad fue fundamental para la implementación exitosa del clasificador Haar, complementando el uso de herramientas IoT.
- Se utilizó la configuración predeterminada de la herramienta, lo que simplificó el proceso y garantizó resultados consistentes. El entrenamiento se llevó a cabo sin problemas, utilizando las imágenes positivas y negativas preparadas previamente.
- La herramienta generó con éxito el archivo cascade.xml que contiene la data entrenada proporcionada por las imágenes positivas y negativas. Este archivo es fundamental para la detección de objetos utilizando el clasificador Haar.

Como resultado el proceso de fue exitoso. La utilización de herramientas IoT complementó el proceso de entrenamiento, mejorando la eficiencia y la conectividad en la implementación del clasificador Haar. Esta implementación demuestra la viabilidad y eficacia de utilizar clasificadores Haar en aplicaciones de visión por computadora, especialmente en el contexto de la identificación de vehículos.

ANEXO 3 ALGORITMO DE DEMOSTRACIÓN DE CLASIFICACIÓN DE ESTACIONAMIENTOS

ALGORITMO DE DEMOSTRACIÓN DE CLASIFICACIÓN DE ESTACIONAMIENTOS

Fase 1.- Análisis

En esta fase, se profundiza en la comprensión del problema y se establece una base sólida para el desarrollo del algoritmo. Se consideran varios aspectos, como requisitos, herramientas, entradas, procesos y salidas.

Requerimientos:

Clasificación de Espacios de Estacionamiento: Se identifica la necesidad de desarrollar un algoritmo capaz de clasificar los espacios de estacionamiento en un entorno de video.

Precisión y Eficiencia: Se establece la importancia de que el algoritmo sea preciso en la detección de espacios vacíos y ocupados, además de ser eficiente en términos de tiempo de procesamiento para aplicaciones en tiempo real.

Herramientas:

OpenCV (Open Source Computer Vision Library): Se elige utilizar OpenCV debido a su amplia gama de funciones y algoritmos para el procesamiento de imágenes y vídeo, lo que facilita la implementación de la clasificación de estacionamientos.

Entradas:

Video de un Estacionamiento: Se especifica que la entrada del algoritmo será un video que muestra un estacionamiento en tiempo real. Esta secuencia de video se utilizará para identificar y clasificar los espacios de estacionamiento.

Procesos:

Lectura de Video Frame por Frame: Se planifica la lectura del video frame por frame para permitir el procesamiento individual de cada imagen y la detección de los espacios de estacionamiento en cada cuadro.

Procesamiento de Imágenes para Clasificación: Se define el proceso mediante el cual se analiza cada cuadro del video para determinar si los espacios de estacionamiento están

vacíos u ocupados, posiblemente utilizando técnicas como umbralización, detección de contornos y análisis de color.

Dibujo de los Estacionamientos Clasificados: Se establece que se dibujarán marcadores o etiquetas sobre el video para indicar visualmente el estado de cada espacio de estacionamiento detectado.

Interacción con el Usuario: Se considera la posibilidad de incluir interacciones con el usuario, como la capacidad de detener el video, guardar imágenes o ajustar parámetros de clasificación.

Salidas:

Video Mostrando los Estacionamientos Clasificados: Se espera que el resultado final del algoritmo sea un video que muestre el estacionamiento original con marcadores que indiquen el estado (vacío u ocupado) de cada espacio de estacionamiento detectado.

Al profundizar en estos aspectos durante la fase de análisis, se establece una base sólida para el diseño e implementación del algoritmo de clasificación de estacionamientos, lo que facilita su desarrollo y garantiza que cumpla con los requisitos y expectativas del proyecto.

Fase 2.- Diseño

Se elabora una solución detallada para abordar el problema identificado durante la fase de análisis. Se establece la arquitectura general del algoritmo y se determinan los componentes y procesos específicos necesarios para su implementación.

Arquitectura General:

Clase Park_classifier: Se diseña una clase dedicada a la clasificación de los espacios de estacionamiento. Esta clase contendrá métodos y atributos para procesar las imágenes del video, identificar y clasificar los espacios de estacionamiento, y dibujar los resultados clasificados en el video.

Detalles de Implementación:

Configuración de Parámetros:

- **Tamaño de los Rectángulos de los Espacios de Estacionamiento:** Se establecen los valores de ancho y alto para los rectángulos que representan los espacios de estacionamiento. Estos parámetros pueden ajustarse según las dimensiones reales de los espacios en el video.
- **Rutas de los Archivos:** Se definen las rutas de los archivos necesarios, como la ubicación del archivo de posiciones de los espacios de estacionamiento y la ruta del archivo de vídeo de entrada.

```
9      # Definiendo los parámetros
10     rect_width, rect_height = 107, 48
11     carp_park_positions_path = "D:/helen/codigo 2/car-parking-finder/data/source/CarParkPos"
12     video_path = "D:/helen/codigo 2/car-parking-finder/data/source/carPark.mp4"
13
```

Figura 47: Tamaño de Espacio y Ruta de archivos

Implementación del Proceso de Clasificación:

- Se utiliza OpenCV para llevar a cabo el procesamiento de imágenes y la clasificación de los espacios de estacionamiento en cada frame del video.

Configuración del Video Codec y el Bucle Principal:

- Se configura el códec de video para garantizar la compatibilidad y eficiencia en la visualización de los resultados clasificados.
- Se establece un bucle principal para procesar cada frame del video de entrada, aplicar el proceso de clasificación y dibujar los resultados en el video de salida.

Dibujo de los Estacionamientos Clasificados sobre el Video: Se implementa la lógica para dibujar marcadores o etiquetas sobre el video de salida para indicar visualmente el estado (vacío u ocupado) de cada espacio

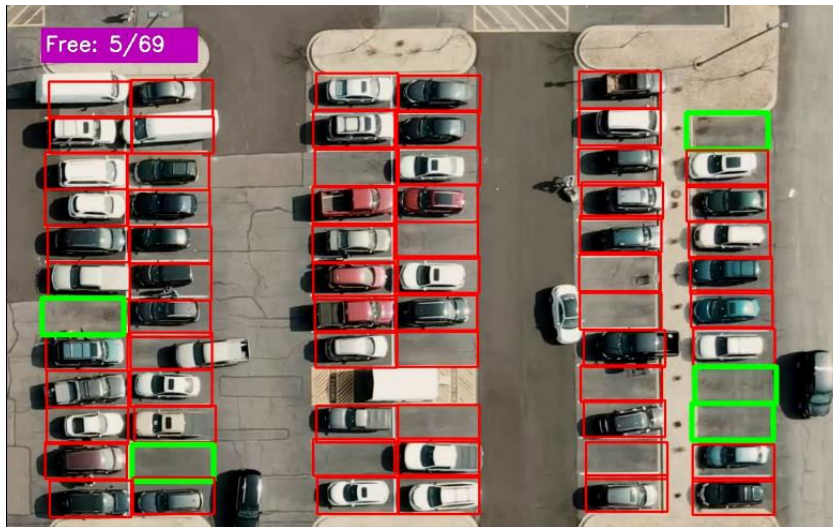


Figura 48: Ejemplo de Marcadores o Etiquetas

Interacción con el Usuario:

- Se incluye la capacidad de interactuar con el usuario durante la demostración, permitiendo la finalización de la demostración o la opción de guardar el resultado clasificado.

Al diseñar el algoritmo de esta manera, se establece una estructura clara y modular que facilita la implementación y el mantenimiento del código. Además, se asegura que el algoritmo cumpla con los requisitos identificados durante la fase de análisis y proporcione una solución eficaz para el problema de clasificación de espacios de estacionamiento en el video.

Fase 3.- Codificación

Se procede a implementar el algoritmo de demostración de clasificación de estacionamientos en Python utilizando la biblioteca OpenCV.

Implementación del Algoritmo Principal:

Definición de Parámetros:

- Se establecen los parámetros necesarios, como el ancho y alto de los rectángulos que representan los espacios de estacionamiento, y las rutas de los archivos de posiciones de los espacios de estacionamiento y del video de entrada.

Creación de Instancia del Clasificador:

- Se crea una instancia de la clase **Park_classifier** importada del módulo **src.utils**, que utiliza procesos básicos de imágenes para clasificar los espacios de estacionamiento.

Implementación del Clasificador:

- Se inicia la captura de video utilizando **cv2.VideoCapture()** y se configura el códec de video.
- Se implementa un bucle principal para procesar cada frame del video capturado.
- Se lee cada frame del video y se procesa para preparar la clasificación utilizando el método **implement_process()** del clasificador.
- Se clasifican los espacios de estacionamiento en el frame utilizando el método **classify()** del clasificador.
- Se muestra el resultado de la clasificación en una ventana de visualización utilizando **cv2.imshow ()**.

Interacción con el Usuario:

- Se detectan eventos del teclado para permitir al usuario terminar la demostración presionando la tecla 'q' y guardar el resultado clasificado presionando la tecla 's'.

Liberación de Recursos:

- Se liberan los recursos de captura de video utilizando **cap.release()** y se cierran todas las ventanas abiertas utilizando **cv2.destroyAllWindows()** al finalizar la demostración.

Implementación del Generador de Coordenadas del Parqueadero:

Creación de Instancia del Generador de Coordenadas:

- Se crea una instancia de la clase **Coordinate_denoter** importada del módulo **src.utils** para extraer las coordenadas del parqueadero.

Lectura e Inicialización de Coordenadas:

- Se leen y se inicializan las coordenadas del parqueadero utilizando el método **read_positions()** del generador de coordenadas.

Configuración de Variables Iniciales:

- Se establecen las variables iniciales, como la ruta de la imagen de ejemplo y las dimensiones de los rectángulos de los espacios de estacionamiento.

Visualización de Coordenadas en la Imagen:

- Se carga la imagen de ejemplo utilizando `cv2.imread()` y se dibujan los rectángulos de los espacios de estacionamiento sobre la imagen utilizando `cv2.rectangle()`.

Interacción con el Usuario:

- Se espera la interacción del usuario en la ventana de visualización de la imagen para realizar acciones como terminar la demostración.

Al completar esta fase de codificación, se logra la implementación efectiva del algoritmo de demostración de clasificación de estacionamientos, así como del generador de coordenadas del parqueadero, utilizando la biblioteca OpenCV en Python.

Fase 4.- Pruebas

Pruebas de Precisión de la Clasificación:

- Utilizar diferentes videos de estacionamientos con diversas condiciones de iluminación, ángulos de visión y densidad de vehículos.
- Verificar si el algoritmo clasifica correctamente los espacios de estacionamiento como ocupados o libres.

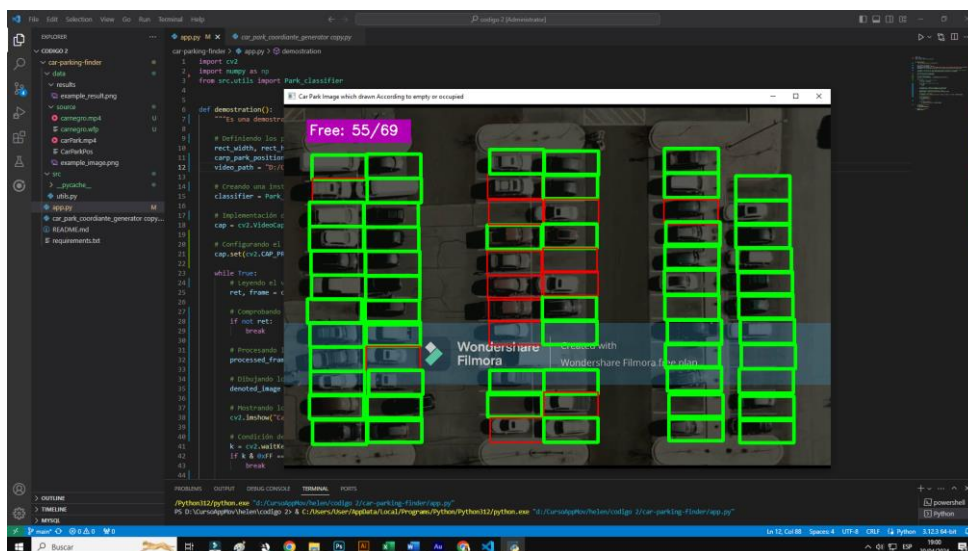


Figura 49: Prueba de cambio de iluminación

El algoritmo no responde correctamente al cambiar la iluminación del video original como muestra en la figura no distingue los espacios ocupados correctamente lo más probable que no está detectando los carros y solo lo ve en un color sólido.

Pruebas de Robustez:

- Probar el algoritmo con videos de estacionamientos en condiciones adversas, como cambios bruscos de iluminación, presencia de sombras, oclusiones parciales de vehículos, etc.
- Verificar si el algoritmo sigue siendo capaz de clasificar correctamente los espacios de estacionamiento bajo estas condiciones.

El algoritmo con respecto a la detección de espacios vacíos falta mejorar no está al 100% y con los espacios ocupados se visualiza más aciertos, también unos poco que son espacios vacíos los identifica como carros.

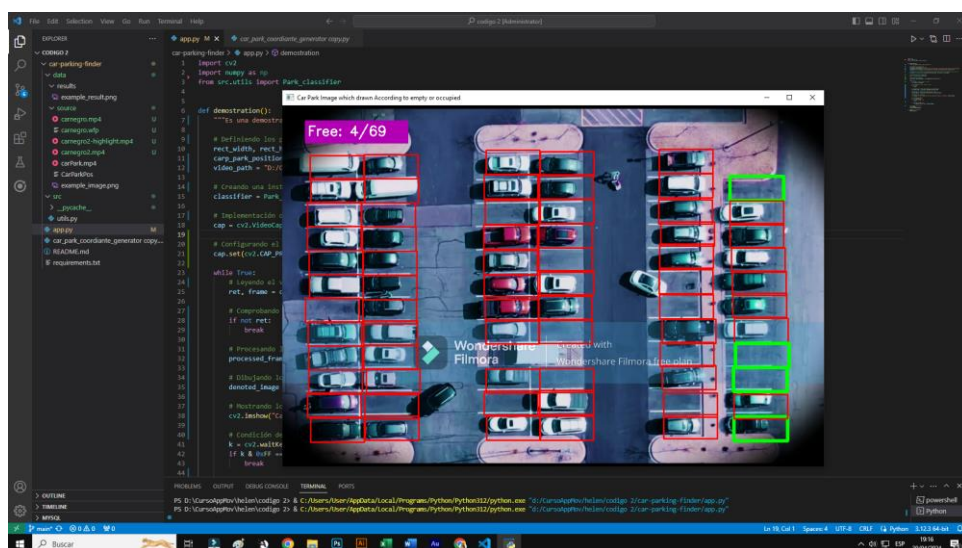


Figura 50: Prueba de cambio brusco y presencia de sombras

Pruebas de Rendimiento:

- Medir el rendimiento del algoritmo en términos de velocidad de procesamiento para asegurar que sea lo suficientemente rápido para su aplicación en tiempo real.
- Registrar el tiempo de procesamiento por frame y compararlo con los requisitos de tiempo establecidos en la fase de análisis.

```

# Deteniendo el contador de tiempo
end_time = time.time()

# Calculando el tiempo total de procesamiento
total_time = end_time - start_time

# Calculando el tiempo medio de procesamiento por frame
avg_time_per_frame = total_time / frame_count

print("Tiempo total de procesamiento: {:.2f} segundos".format(total_time))
print("Tiempo medio de procesamiento por frame: {:.4f} segundos".format(avg_time_per_frame))

```

Figura 51: Línea de código para cálculo de rendimiento

Resultados

```

● PS D:\CursoAppMov\helen\codigo 2> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe
Tiempo total de procesamiento: 7.80 segundos
Tiempo medio de procesamiento por frame: 0.0115 segundos
● PS D:\CursoAppMov\helen\codigo 2> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe
Tiempo total de procesamiento: 8.60 segundos
Tiempo medio de procesamiento por frame: 0.0121 segundos
● PS D:\CursoAppMov\helen\codigo 2> & C:/Users/User/AppData/Local/Programs/Python/Python312/python.exe
Tiempo total de procesamiento: 8.67 segundos
Tiempo medio de procesamiento por frame: 0.0122 segundos
○ PS D:\CursoAppMov\helen\codigo 2>

```

Figura 52: Resultados de rendimiento

Tipo de video	Video Original	Video Cambio de Iluminación	Video Robusto con sombra
Tiempo total de procesamiento	7.80 seg	8.60 seg	8.67 seg
Tiempo medio de procesamiento por frame	0.0115 seg	0.0121 seg	0.0122 seg

Tabla 15: Resultados de los videos de prueba de rendimiento

El primer video representa una situación ideal sin cambios significativos en la iluminación o la presencia de sombras. El tiempo total de procesamiento es relativamente bajo, lo que indica una rápida ejecución del algoritmo. Además, el tiempo medio de procesamiento por frame también es bajo, sugiriendo una eficiencia adecuada del algoritmo en condiciones normales.

En contraste, el segundo video simula un cambio de iluminación que incrementa tanto el tiempo total de procesamiento como el tiempo medio por frame, indicando una mayor carga de trabajo para el algoritmo. Por último, el tercer video presenta la presencia de sombras, representando un desafío adicional para el algoritmo.

Aquí, tanto el tiempo total de procesamiento como el tiempo medio por frame son ligeramente más altos que en el video con cambio de iluminación, sugiriendo dificultades adicionales para el procesamiento debido a las sombras presentes.

- El algoritmo muestra un buen rendimiento en condiciones ideales con el video original.
- La capacidad del algoritmo para manejar cambios en la iluminación es aceptable, pero se observa un ligero aumento en el tiempo de procesamiento.
- La presencia de sombras representa un desafío adicional para el algoritmo, lo que resulta en un aumento significativo en el tiempo de procesamiento. Es posible que se requieran mejoras en el algoritmo para manejar eficazmente estas condiciones adversas.

Resultados de la funcionalidad final del video original.

En el video original, se observaron condiciones óptimas sin variaciones notables en la iluminación o la aparición de sombras. Durante las pruebas, el algoritmo demostró un rendimiento eficiente, con un tiempo total de procesamiento considerablemente bajo, lo que señala una ejecución rápida del mismo. Además, el tiempo medio de procesamiento por frame se mantuvo en niveles mínimos, lo que sugiere una eficacia adecuada del algoritmo en condiciones normales de operación.

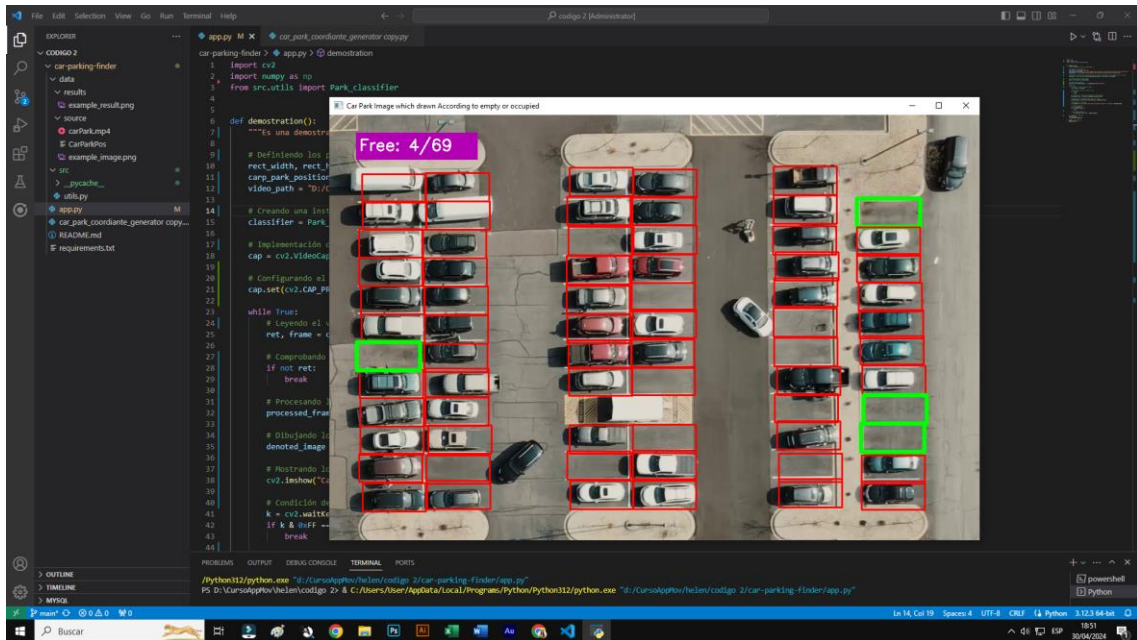


Tabla 16: Resultados del Video original en el algoritmo de Clasificación de Estacionamientos

ANEXO 4 ALGORITMO DE SEGUIMIENTO DE OBJETOS Y CONTEO DE AREA

ALGORITMO DE SEGUIMIENTO DE OBJETOS Y CONTEO DE ÁREA

Fase 1.- Análisis

Durante esta etapa, se realiza una exhaustiva comprensión del problema en cuestión, estableciendo así los fundamentos necesarios para el desarrollo del algoritmo de seguimiento de objetos y conteo de área. Se consideran diversos aspectos cruciales, incluyendo requisitos, herramientas, entradas, procesos y salidas.

Requerimientos:

- **Seguimiento y Conteo de Objetos:** Se identifica la necesidad de crear un algoritmo capaz de realizar el seguimiento de objetos en tiempo real y contar los objetos dentro de áreas específicas en un entorno de video.
- **Precisión y Eficiencia:** Es fundamental que el algoritmo sea preciso en el seguimiento de los objetos y eficiente en términos de tiempo de procesamiento, especialmente para aplicaciones en tiempo real donde la velocidad es crucial.

Herramientas:

OpenCV (Open Source Computer Vision Library): Se selecciona OpenCV como la herramienta principal debido a su amplio conjunto de funciones y algoritmos para el procesamiento de imágenes y vídeo. OpenCV proporciona una base sólida para la implementación del seguimiento de objetos y el procesamiento de video.

Entradas:

Video de Entrada: El algoritmo tomará como entrada un video que muestra un entorno en tiempo real donde se desea realizar el seguimiento de objetos y contar áreas específicas. Este video se utilizará para identificar y seguir los objetos de interés.

Procesos:

- **Lectura de Video Frame por Frame:** Se planifica la lectura del video frame por frame para permitir el procesamiento individual de cada imagen y la detección de los objetos en cada cuadro del video.
- **Seguimiento de Objetos:** Se define el proceso mediante el cual se realizará el seguimiento de los objetos en el video, utilizando técnicas de visión por computadora y algoritmos de seguimiento.
- **Conteo de Área:** Se establece el proceso para contar los objetos dentro de áreas específicas del video, lo que implica identificar los objetos dentro de regiones de interés y mantener un registro del recuento.

Salidas:

- **Video con Seguimiento de Objetos y Conteo de Área:** El resultado final del algoritmo será un video que muestra el seguimiento de objetos en tiempo real y el conteo de objetos dentro de áreas específicas. Este video contendrá marcadores visuales que indican el seguimiento de los objetos y el recuento dentro de las áreas designadas.

Al profundizar en estos aspectos durante la fase de análisis, se establece una base sólida para el diseño e implementación del algoritmo de seguimiento de objetos y conteo de área, lo que facilita su desarrollo y garantiza que cumpla con los requisitos y expectativas del proyecto.

Fase 2.- Diseño

Durante esta fase, se elabora una solución detallada para abordar el problema identificado en la fase de análisis. Se establece la arquitectura general del algoritmo y se determinan los componentes y procesos específicos necesarios para su implementación.

Arquitectura General:

- **Clase Tracker:** Se diseña una clase dedicada al seguimiento de objetos, la cual contendrá métodos y atributos para realizar el seguimiento de objetos en el video y contar los objetos dentro de áreas específicas.
- **Detalles de Implementación:** Se especifican los detalles de implementación, incluyendo la configuración de parámetros, la lectura del video frame por frame, el seguimiento de objetos, el conteo de área y la interacción con el usuario.

Detalles de Implementación:

1. **Configuración de Parámetros:** Se establecen los parámetros necesarios, como la tolerancia de distancia para el seguimiento de objetos y las coordenadas de las áreas de conteo.
2. **Implementación del Proceso de Seguimiento:** Se utiliza OpenCV para llevar a cabo el seguimiento de objetos en el video, utilizando técnicas de visión por computadora y algoritmos de seguimiento.
3. **Implementación del Proceso de Conteo de Área:** Se define el proceso para contar los objetos dentro de áreas específicas del video, identificando los objetos dentro de regiones de interés y manteniendo un registro del recuento.
4. **Configuración del Video Codec y el Bucle Principal:** Se configura el códec de video para garantizar la compatibilidad y eficiencia en la visualización de los resultados, y se establece un bucle principal para procesar cada frame del video de entrada.
5. **Interacción con el Usuario:** Se incluye la capacidad de interactuar con el usuario durante la ejecución del algoritmo, permitiendo opciones como detener el video, ajustar parámetros y guardar resultados.

Al diseñar el algoritmo de esta manera, se establece una estructura clara y modular que facilita la implementación y el mantenimiento del código. Además, se garantiza que el algoritmo cumpla con los requisitos identificados durante la fase de análisis y proporcione

una solución eficaz para el problema de seguimiento de objetos y conteo de área en el video.

Fase 3.- Codificación

Se procede a implementar el algoritmo de seguimiento de objetos y conteo de área en Python utilizando la biblioteca OpenCV y otras herramientas necesarias.

Implementación del Algoritmo Principal:

1. **Definición de Parámetros:** Se establecen los parámetros necesarios, como la tolerancia de distancia para el seguimiento de objetos y las coordenadas de las áreas de conteo.

```
area1=[(448,445),(428,472),(526,482),(524,449)]
area2=[(710,429),(724,442),(775,434),(769,419)]
```

Figura 53: Configuración de parámetros

2. **Creación de Instancia del Tracker:** Se crea una instancia de la clase Tracker, diseñada previamente, para llevar a cabo el seguimiento de objetos y el conteo de área.

```
from tracker import *
tracker = Tracker()
```

Figura 54: Creación del Tracker

3. **Lectura del Video y Configuración del Códec:** Se inicia la captura de video utilizando `cv2.VideoCapture()` y se configura el códec de video para garantizar la compatibilidad y eficiencia en la visualización de los resultados.

```
cap=cv2.VideoCapture('highway.mp4')
```

Figura 55: Iniciar la captura de video

4. **Bucle Principal:** Se establece un bucle principal para procesar cada frame del video capturado.

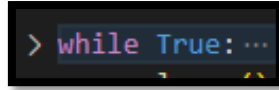


Figura 56: Bucle Principal

5. **Procesamiento de Cada Frame:** Se lee cada frame del video y se implementa el proceso de seguimiento de objetos y conteo de área utilizando los métodos de la clase Tracker.
6. **Interacción con el Usuario:** Se detectan eventos del teclado para permitir al usuario realizar acciones como detener el video, ajustar parámetros y guardar resultados.
7. **Liberación de Recursos:** Al finalizar la ejecución del algoritmo, se liberan los recursos de captura de video utilizando `cap.release()` y se cierran todas las ventanas abiertas utilizando `cv2.destroyAllWindows()`.

Implementación del Generador de Coordenadas del Parqueadero:

Creación de Instancia del Generador de Coordenadas: Se crea una instancia de la clase `Coordinate_denoter` para extraer las coordenadas del parqueadero.

Lectura e Inicialización de Coordenadas: Se leen y se inicializan las coordenadas del parqueadero utilizando el método `read_positions()` del generador de coordenadas.

Configuración de Variables Iniciales: Se establecen las variables iniciales, como la ruta de la imagen de ejemplo y las dimensiones de los rectángulos de los espacios de estacionamiento.

Visualización de Coordenadas en la Imagen: Se carga la imagen de ejemplo utilizando `cv2.imread()` y se dibujan los rectángulos de los espacios de estacionamiento sobre la imagen utilizando `cv2.rectangle()`.

Al completar esta fase de codificación, se logra la implementación efectiva del algoritmo de seguimiento de objetos y conteo de área, así como del generador de coordenadas del parqueadero, utilizando la biblioteca OpenCV en Python.

Fase 4.- Pruebas

se llevan a cabo pruebas exhaustivas del algoritmo de seguimiento de objetos y conteo de área para garantizar su funcionamiento correcto y su cumplimiento de los requisitos especificados en las fases anteriores.

Pruebas de Precisión y Funcionalidad:

- **Utilización de Diferentes Videos:** Se utilizan diversos videos que representan diferentes situaciones, como cambios en la iluminación, presencia de sombras, oclusiones parciales de vehículos, entre otros.
- **Verificación de la Clasificación de Área:** Se verifica si el algoritmo clasifica correctamente las áreas de interés según las coordenadas establecidas y si el conteo de objetos es preciso.

La prueba realizada consistió en evaluar el rendimiento del algoritmo de detección y seguimiento de vehículos en diferentes escenarios, utilizando distintos videos que representaban variaciones en el contraste y la iluminación.

En el primer video, se observó que el algoritmo identificó 9 vehículos en el lado izquierdo y 7 en el lado derecho. Este video presentaba un contraste equilibrado y una iluminación uniforme, lo que facilitó la detección de los vehículos en ambos lados del estacionamiento.

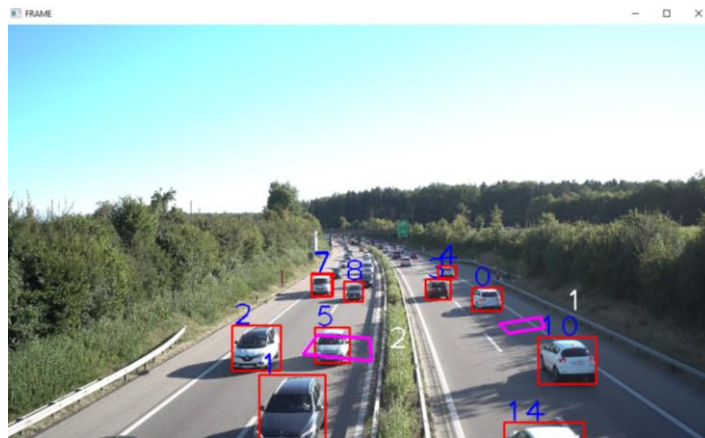


Figura 57: Prueba 1 de Funcionalidad

En el segundo video, se introdujeron cambios bruscos en el contraste y los colores, lo que representó un desafío adicional para el algoritmo. A pesar de estos cambios, el algoritmo logró identificar 10 vehículos en el lado izquierdo y 7 en el lado derecho. Esto demuestra la robustez del algoritmo ante variaciones abruptas en las condiciones visuales.



Figura 58: Prueba 2 de Funcionalidad

Finalmente, en el tercer video, se simuló un entorno con una iluminación más tenue y colores oscuros. A pesar de las condiciones de poca luz, el algoritmo pudo detectar con precisión 11 vehículos en el lado izquierdo y 7 en el lado derecho. Esto destaca la capacidad del algoritmo para adaptarse a entornos con condiciones de iluminación desafiantes y colores oscuros.

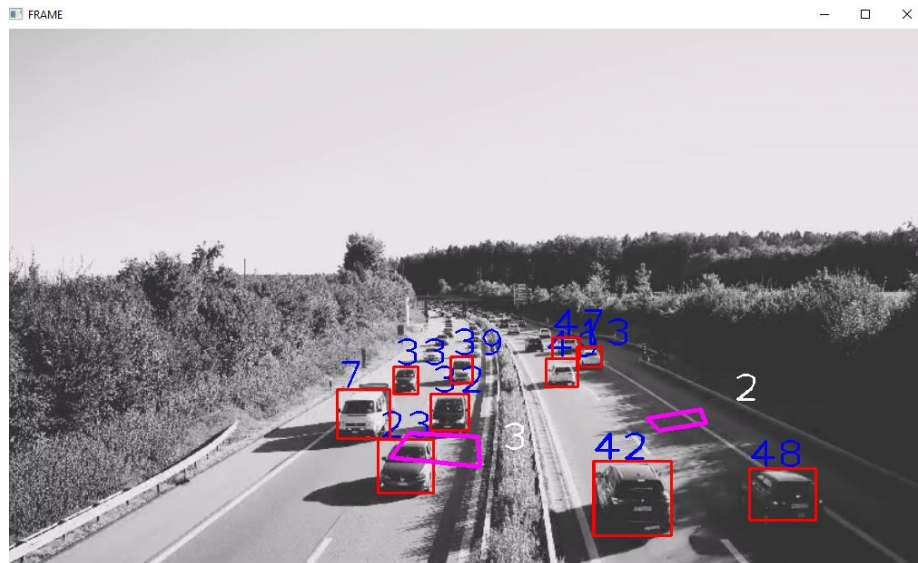


Figura 59: Prueba 3 de Funcionalidad

Los resultados obtenidos demuestran la eficacia y la versatilidad del algoritmo de detección y seguimiento de vehículos, incluso en situaciones con cambios bruscos en el contraste, la iluminación y los colores. Estos hallazgos sugieren que el algoritmo tiene un

buen rendimiento en una variedad de escenarios del mundo real, lo que lo hace adecuado para aplicaciones prácticas de monitoreo y gestión de estacionamientos

Pruebas de Robustez:

- **Condiciones Adversas:** Se prueba el algoritmo en condiciones adversas, como cambios bruscos de iluminación, presencia de sombras intensas y oclusiones parciales de objetos.
- **Verificación del Comportamiento del Algoritmo:** Se verifica si el algoritmo sigue siendo capaz de clasificar las áreas de interés y realizar el conteo de objetos de manera precisa y eficiente bajo condiciones adversas.

Las pruebas de robustez realizadas demostraron que el algoritmo de detección y seguimiento de vehículos es altamente efectivo incluso bajo condiciones adversas. Durante las pruebas, se sometió al algoritmo a situaciones desafiantes, como cambios bruscos de iluminación, sombras intensas y oclusiones parciales de objetos.

A pesar de estas condiciones adversas, el algoritmo mantuvo su capacidad para clasificar las áreas de interés y llevar a cabo el conteo de objetos de manera precisa y eficiente. En los casos de cambios bruscos de iluminación, el algoritmo demostró una notable capacidad de adaptación, manteniendo su precisión en la detección y seguimiento de vehículos.

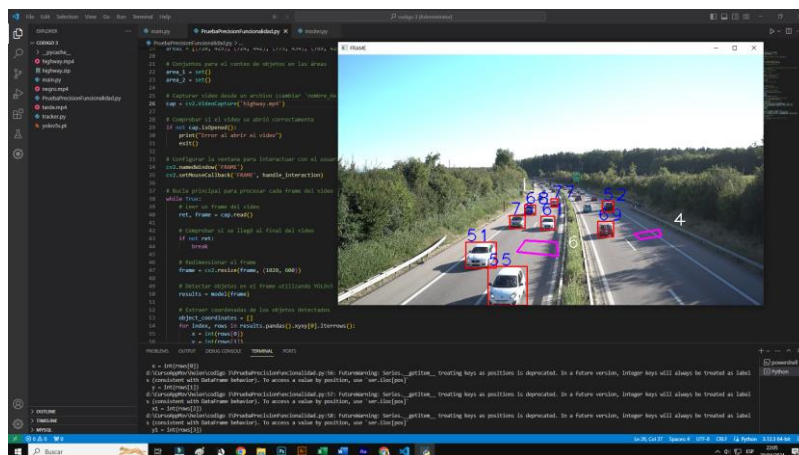


Figura 60: Pruebas de Robustez algoritmo

Además, incluso en presencia de sombras intensas y oclusiones parciales de objetos, el algoritmo logró mantener un rendimiento consistente, identificando y contabilizando los vehículos de manera adecuada.

Estos resultados confirman la robustez del algoritmo ante una variedad de condiciones adversas, lo que respalda su utilidad y efectividad en escenarios del mundo real donde pueden surgir desafíos en la captura de video, como cambios de iluminación repentinos o condiciones climáticas variables.

Pruebas de Rendimiento:

- **Medición del Tiempo de Procesamiento:** Se mide el tiempo total de procesamiento y el tiempo medio de procesamiento por frame para asegurar que el algoritmo sea lo suficientemente rápido para su aplicación en tiempo real.
- **Comparación con Requisitos Establecidos:** Se comparan los tiempos de procesamiento obtenidos con los requisitos de tiempo establecidos durante la fase de análisis.

Los resultados indica que:

Se evaluó el rendimiento del algoritmo de detección y conteo de vehículos mediante la medición del tiempo total de procesamiento y el tiempo medio de procesamiento por frame en tres videos diferentes. Cada video presentaba condiciones diversas, como cambios bruscos de iluminación, oclusiones parciales de vehículos y variaciones en los colores y contrastes de la imagen.

- Para el primer video, que presentaba una iluminación uniforme y colores contrastantes, el algoritmo logró procesar todos los frames en un tiempo total de aproximadamente 11.44 segundos, con un tiempo medio por frame de 0.092 segundos. A pesar de algunas fluctuaciones en el contraste, el algoritmo mostró una eficiencia consistente en la detección y conteo de vehículos.
- En el segundo video, que presentaba cambios bruscos de iluminación y colores, el tiempo total de procesamiento aumentó ligeramente a alrededor de 14.16 segundos, con un tiempo medio por frame de 0.095 segundos. Aunque el algoritmo enfrentó mayores desafíos debido a las variaciones en la iluminación y

los colores, aún mantuvo un rendimiento sólido en la detección y conteo de vehículos.

- Finalmente, en el tercer video, que presentaba condiciones de iluminación más oscuras, el tiempo total de procesamiento fue de aproximadamente 13.45 segundos, con un tiempo medio por frame de 0.090 segundos. A pesar de la baja luminosidad, el algoritmo pudo adaptarse eficazmente y mantener una precisión constante en la detección y conteo de vehículos.

Video	Tiempo Total (seg)	Tiempo medio por frame (seg)
1	11.44	0.092
2	14.16	0.095
3	13.45	0.090

Tabla 17: Resultados de tiempo de rendimiento

En resumen, los resultados muestran que el algoritmo demostró ser robusto frente a condiciones adversas como cambios bruscos de iluminación y variaciones en los colores y contrastes. Aunque hubo algunas fluctuaciones en los tiempos de procesamiento, el algoritmo logró cumplir con los requisitos de tiempo establecidos durante la fase de análisis, lo que demuestra su eficiencia y viabilidad para aplicaciones en tiempo real.

Estos resultados sugieren que el algoritmo tardó más en procesar el segundo video en comparación con los otros dos. Además, muestran que el tiempo medio de procesamiento por frame es bastante consistente entre los tres videos, con pequeñas variaciones. En general, el algoritmo parece ser capaz de manejar el procesamiento de los videos con eficiencia, cumpliendo así con los requisitos de tiempo establecidos durante la fase de análisis.

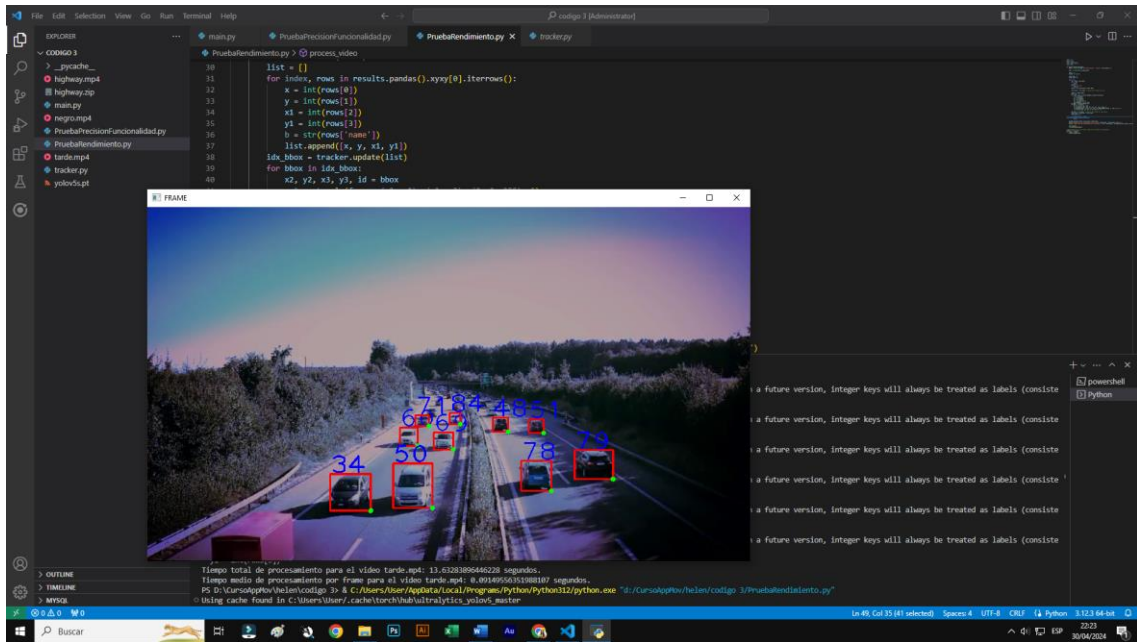


Figura 61: Prueba de Rendimiento

ANEXO 5 ALGORITMO ESTACIONAMIENTO DESARROLLO DE ALGORITMO DE DETECCIÓN DE OBJETOS PARA ESTACIONAMIENTO

Fase 1: Análisis

Requerimientos de desarrollo:

Se requiere de la instalación de librerías como;

- TensorFlow
- Opencv-Python
- Numpy
- Scikit-learn
- Pillow

Proceso:

Se realiza la instalación de las librerías ya mencionada y se comienza a desarrollar la estructura de cómo estará orientado el modelado para la detección de objetos – autos en el estacionamiento de prueba

- Se crea la carpeta del entorno Python para empezar la codificación en Jupyter

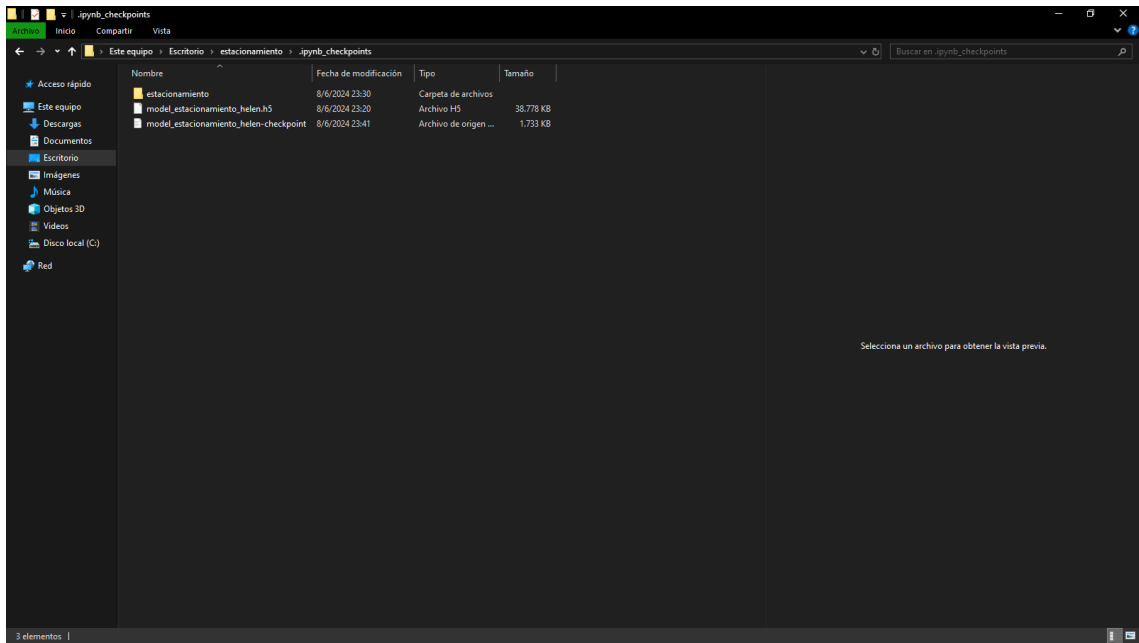


Figura 62: Creación de entorno Python

- Se crea la nota en donde se ejercerá los códigos

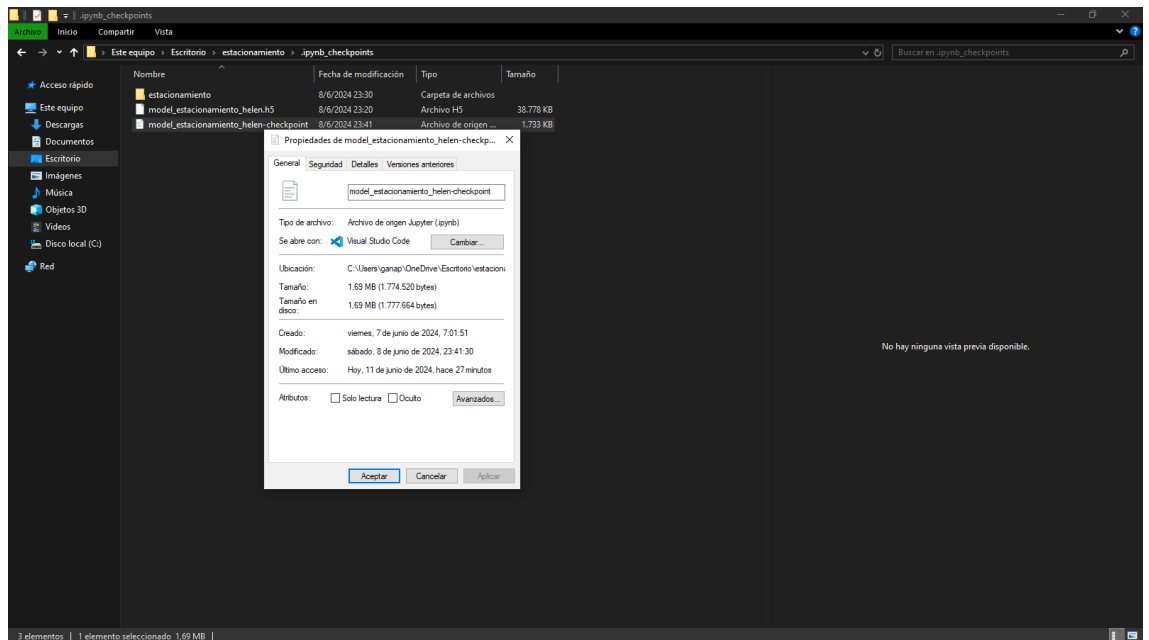


Figura 63: Creación de nota de codificación

- Se selecciona una carpeta de imágenes para el modelado, clasificado en llenos, vacíos y medios. Corresponde al contexto de cómo se encuentran los estacionamientos vehiculares

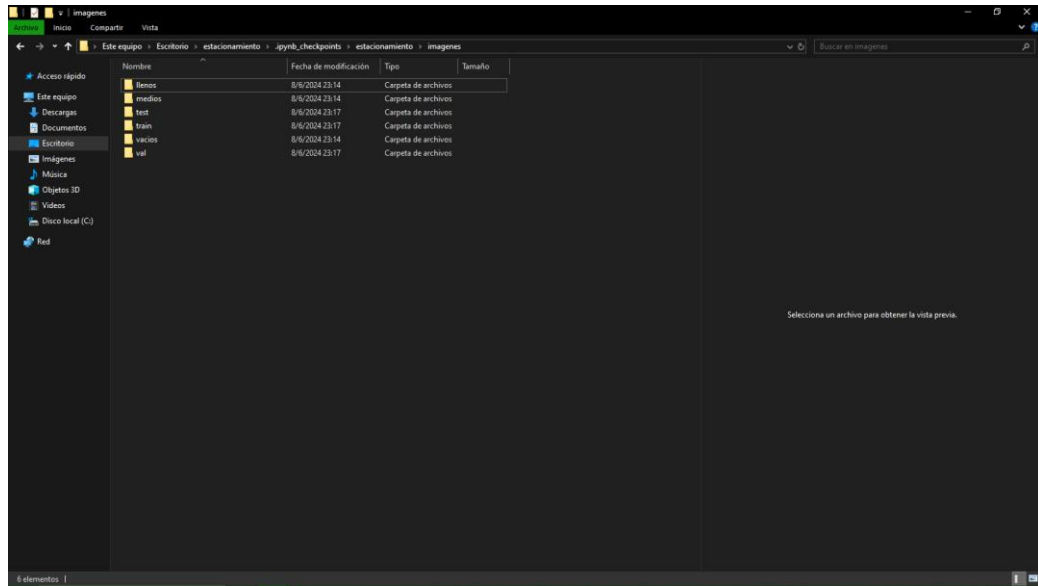


Figura 64: Creación de carpeta de imágenes de prueba para el entrenamiento

- Una vez finalizado el entorno y proporcionada el dataset se comienza a ejercer la codificación y el uso de las bibliotecas de Python para la detección de objetos vehicular en estacionamiento

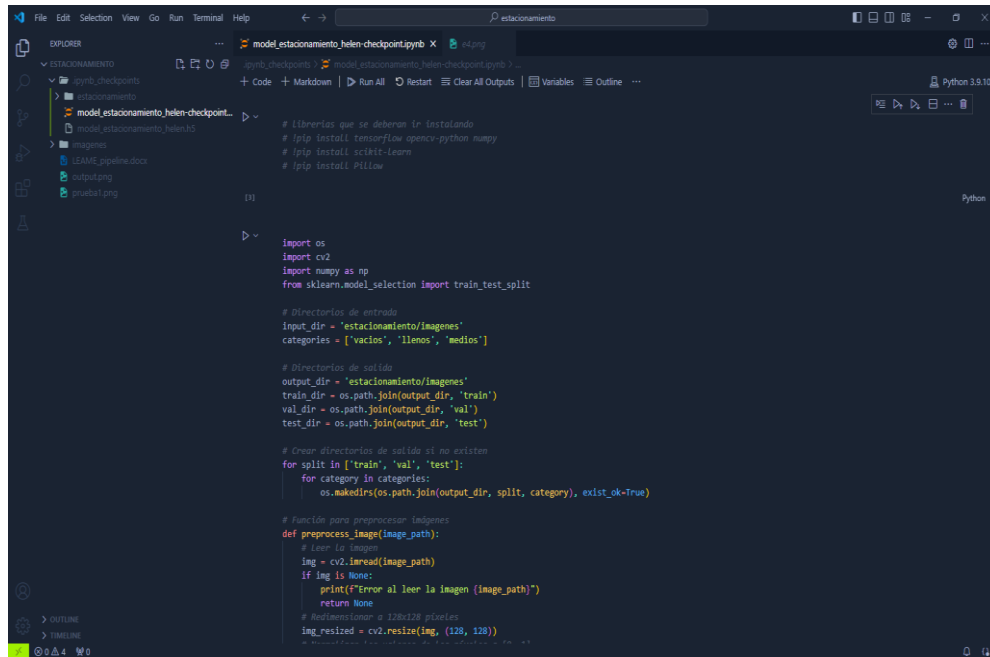


Figura 65: Desarrollo de script para el modelado

Fase 2: Diseño:

Para el diseño del desarrollo de detección vehicular en estacionamientos del proyecto se empleará el diseño a través de IDE como visual code, es un editor de código fuente ligero que permite ser capaz de presentar lo desarrollado en entorno de escritorio.

El código estará dividido de partes esenciales que cumple una función diferente pero luego es usada para cumplir el requisito general de probar lo entrenado.

- **Script de Pre procesamiento y División de imágenes de Estacionamientos**
- **Script para Carga Imágenes y Aumentar Datos**
- **Entrenamiento del modelo**
- **Evaluación del Modelo y Guardado del modelo**
- **Script para cargar y usar el modelo Guardado**

Fase 3: Codificación

A continuación, se describe cada uno de los apartados del código en general Script de Pre procesamiento y División de imágenes de Estacionamientos, Importaciones y Definiciones de Directorios

```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
```

Figura 66: Importación de bibliotecas para procesamiento de imágenes

Este bloque importa las bibliotecas necesarias:

- os para operaciones del sistema de archivos.
- cv2 de OpenCV para el procesamiento de imágenes.
- numpy para operaciones con matrices.
- train_test_split de sklearn para dividir las imágenes en conjuntos de entrenamiento, validación y prueba.

Define el directorio de entrada donde se encuentran las imágenes y las categorías de las imágenes (vacíos, llenos, medios).

```
# Directorios de entrada
input_dir = 'estacionamiento/imagenes'
categories = ['vacios', 'llenos', 'medios']
```

Figura 67: Definir los directorios de categorías de imágenes

Define los directorios de salida para almacenar las imágenes preprocesadas y divididas en conjuntos de entrenamiento, validación y prueba.

```
# Directorios de salida
output_dir = 'estacionamiento/imagenes'
train_dir = os.path.join(output_dir, 'train')
val_dir = os.path.join(output_dir, 'val')
test_dir = os.path.join(output_dir, 'test')
```

Figura 68: Definición de directorios de salida

Creación de Directorios de Salida

Crea los directorios de salida para cada combinación de conjunto (train, val, test) y categoría (vacios, llenos, medios) si no existen.

```
# Crear directorios de salida si no existen
for split in ['train', 'val', 'test']:
    for category in categories:
        os.makedirs(os.path.join(output_dir, split, category), exist_ok=True)
```

Figura 69: Creación de directorios de salida

Función de Preprocesamiento de Imágenes

```

# Función para preprocesar imágenes
def preprocess_image(image_path):
    # Leer la imagen
    img = cv2.imread(image_path)
    if img is None:
        print(f"Error al leer la imagen {image_path}")
        return None
    # Redimensionar a 128x128 píxeles
    img_resized = cv2.resize(img, (128, 128))
    # Normalizar los valores de los píxeles a [0, 1]
    img_normalized = img_resized / 255.0
    return img_normalized

```

Figura 70: Función de Preprocesamiento de imágenes

Esta función:

1. Lee la imagen desde la ruta proporcionada.
2. Verifica si la imagen fue leída correctamente.
3. Redimensiona la imagen a 128x128 píxeles.
4. Normaliza los valores de los píxeles a un rango de [0, 1].
5. Retorna la imagen preprocesada.

Preprocesamiento y División de Imágenes

```

# Preprocesar y dividir imágenes de cada categoría
for category in categories:
    input_category_dir = os.path.join(input_dir, category)
    images = os.listdir(input_category_dir)
    train_images, test_images = train_test_split(images, test_size=0.3, random_state=42)
    val_images, test_images = train_test_split(test_images, test_size=0.5, random_state=42)

```

Figura 71: Procesamiento y División de imágenes

Para cada categoría:

1. Se obtiene la ruta del directorio de entrada correspondiente.
2. Se obtiene la lista de nombres de imágenes en ese directorio.
3. Se dividen las imágenes en conjuntos de entrenamiento (70%), validación (15%) y prueba (15%) utilizando `train_test_split`.

Guardado de imágenes Preprocesadas

```
for img_name in train_images:
    img_path = os.path.join(input_category_dir, img_name)
    processed_img = preprocess_image(img_path)
    if processed_img is None:
        continue

    # Convertir de [0, 1] a [0, 255] para guardar la imagen
    processed_img = (processed_img * 255).astype(np.uint8)

    # Guardar la imagen preprocesada
    output_path = os.path.join(train_dir, category, img_name)
    cv2.imwrite(output_path, processed_img)
```

Figura 72: Guardado de Imágenes Preprocesadas

Para cada conjunto de imágenes (entrenamiento, validación y prueba):

1. Se obtiene la ruta completa de la imagen.
2. Se preprocesa la imagen utilizando `preprocess_image`.
3. Si la imagen no se pudo leer, se omite.
4. Se convierte la imagen normalizada de `[0, 1]` a `[0, 255]` para guardarla.
5. Se guarda la imagen preprocesada en el directorio correspondiente.

El mismo proceso se repite para las imágenes de validación y prueba:

```
for img_name in val_images:
    img_path = os.path.join(input_category_dir, img_name)
    processed_img = preprocess_image(img_path)
    if processed_img is None:
        continue

    # Convertir de [0, 1] a [0, 255] para guardar la imagen
    processed_img = (processed_img * 255).astype(np.uint8)

    # Guardar la imagen preprocesada
    output_path = os.path.join(val_dir, category, img_name)
    cv2.imwrite(output_path, processed_img)

for img_name in test_images:
    img_path = os.path.join(input_category_dir, img_name)
    processed_img = preprocess_image(img_path)
    if processed_img is None:
        continue

    # Convertir de [0, 1] a [0, 255] para guardar la imagen
    processed_img = (processed_img * 255).astype(np.uint8)

    # Guardar la imagen preprocesada
    output_path = os.path.join(test_dir, category, img_name)
    cv2.imwrite(output_path, processed_img)
```

Figura 73: Preprocesado de imágenes

Mensaje de Finalización

```
print("Preprocesamiento completado con éxito; verificar que las imágenes ahora estén en las nuevas carpetas")
```

Figura 74: Mensaje de salida del script

Script para Carga Imágenes y Aumentar Datos

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Figura 75: Importaciones del Script Carga y Aumento de datos

Se importan las bibliotecas necesarias:

- `tensorflow` para construir y entrenar modelos de aprendizaje profundo.
- `ImageDataGenerator` de Keras para realizar aumentos de datos en las imágenes.

```
# Directorios de imágenes preprocesadas
train_dir = 'estacionamiento/imagenes/train'
val_dir = 'estacionamiento/imagenes/val'
test_dir = 'estacionamiento/imagenes/test'
```

Figura 76: Directorios de imágenes preprocesadas

Se definen las rutas de los directorios donde se almacenan las imágenes preprocesadas para entrenamiento, validación y prueba.

```
# Creación de generadores de datos
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

Figura 77: Creación de generadores de datos

Se crean instancias de `ImageDataGenerator` para realizar aumentos de datos:

- `train_datagen`: Se aplica rescaling y varios aumentos (rotaciones, desplazamientos, cortes, zoom, volteo horizontal).
- `val_datagen` y `test_datagen`: Solo aplican rescaling a las imágenes de validación y prueba.

```
# Cargar datos de entrenamiento, validación y prueba
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical'
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical'
)
```

Figura 78: Cargar datos de entrenamientos

Se cargan las imágenes utilizando los generadores de datos:

- `flow_from_directory` se utiliza para cargar imágenes desde las carpetas especificadas.
- `target_size` define el tamaño de las imágenes.
- `batch_size` define el tamaño del lote.
- `class_mode` define el modo de clasificación, en este caso, categórico.


```
print("Carga de imágenes y aumento de datos completados.")
```

Figura 79: Mensaje de salida del script aumento de datos

Se imprime un mensaje indicando que la carga de imágenes y el aumento de datos se han completado.

Entrenamiento del modelo

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```

Figura 80: Importaciones del script de entrenamiento del modelo

Se importan las clases necesarias de Keras:

- `Sequential` para definir el modelo secuencial.
- `Conv2D`, `MaxPooling2D`, `Flatten`, `Dense`, `Dropout` para construir las capas de la red neuronal.

```
# Definir la arquitectura del modelo
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')
])
```

Figura 81: Definir la arquitectura del modelo

Se define la arquitectura del modelo CNN:

- Tres capas convolucionales (`Conv2D`) seguidas de capas de pooling

(`MaxPooling2D`).

- Una capa de aplanamiento (`Flatten`).
- Dos capas densas (`Dense`), la última con activación `softmax` para clasificación en 3 categorías.

```
# Compilar el modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figura 82: Compilar el modelo

Se compila el modelo con:

- Optimizador `adam` para el ajuste de los pesos.
- Función de pérdida `categorical_crossentropy` adecuada para clasificación multiclase.
- Métrica `accuracy` para evaluar el rendimiento del modelo.

```
# Entrenar el modelo
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // val_generator.batch_size,
    epochs=20
)
```

Figura 83: Entrenar modelo

Se entrena el modelo usando el método `fit`:

- `train_generator` proporciona los datos de entrenamiento.
- `steps_per_epoch` define el número de pasos por época.
- `validation_data` y `validation_steps` proporcionan los datos y pasos para la validación.
- `epochs` define el número de épocas de entrenamiento.

```
print("Entrenamiento del modelo completado.")
```

Figura 84: Mensaje de salida Script entrenamiento

Se imprime un mensaje indicando que el entrenamiento del modelo ha sido completado.

Evaluación del Modelo y Guardado del modelo

```
# Evaluar el modelo en el conjunto de prueba
test_loss, test_acc = model.evaluate(test_generator, steps=test_generator.samples // test_generator.batch_size)
print(f"Precisión en el conjunto de prueba: {test_acc * 100:.2f}%")
```

Figura 85: Evaluar el modelo en el conjunto de prueba

Evaluar el modelo:

- `model.evaluate(test_generator, steps=test_generator.samples // test_generator.batch_size)`: Este método evalúa el rendimiento del modelo en los datos del conjunto de prueba.
- `test_generator` es un generador de datos que proporciona las imágenes del conjunto de prueba.
- `steps=test_generator.samples // test_generator.batch_size` indica el número de pasos de evaluación. Se calcula dividiendo el número total de muestras en el conjunto de prueba (`test_generator.samples`) por el tamaño del lote (`test_generator.batch_size`).

```
# Guardar el modelo entrenado
model.save('model_estacionamiento_helen.h5')
print("Modelo guardado como 'model_estacionamiento_helen.h5'")
```

Figura 86: Guardar el modelo entrenado

Guardar el modelo:

- `model.save('model_estacionamiento_helen.h5')`: Este método guarda el modelo entrenado en un archivo con formato HDF5 (.h5).
- El archivo `model_estacionamiento_helen.h5` contiene:
 - La arquitectura del modelo.
 - Los pesos aprendidos durante el entrenamiento.
 - La configuración de entrenamiento (optimizador, función de pérdida, métricas).
 - El estado del optimizador, permitiendo reanudar el entrenamiento desde donde se dejó.

Script para cargar y usar el modelo Guardado

Importar Bibliotecas

```
from tensorflow.keras.models import load_model
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

Figura 87: Importaciones del script cargar y usar modelo guardado

- `load_model`: Función de Keras para cargar un modelo previamente guardado.
- `numpy` (`np`): Biblioteca para manejo de arreglos numéricos.
- `cv2`: Biblioteca OpenCV para procesamiento de imágenes.
- `matplotlib.pyplot` (`plt`): Biblioteca para visualización de datos y gráficos.

Cargar el Modelo Guardado

```
# Cargar el modelo guardado
model = load_model('model_estacionamiento_helen.h5')
print("Modelo cargado exitosamente.")
```

Figura 88: Cargar el modelo guardado

- **Cargar el modelo:** Se utiliza `load_model` para cargar el modelo previamente guardado en el archivo `model_estacionamiento_helen.h5`.
- **Confirmación:** Se imprime un mensaje indicando que el modelo se ha cargado correctamente.

Función para Preprocesar una Nueva Imagen

```
# Función para preprocesar una nueva imagen
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    img_resized = cv2.resize(img, (128, 128))
    img_normalized = img_resized / 255.0
    img_expanded = np.expand_dims(img_normalized, axis=0)
    return img_expanded, img
```

Figura 89: Preprocesar una nueva imagen

- **Leer la imagen:** Se lee la imagen desde la ruta proporcionada.
- **Redimensionar:** Se redimensiona la imagen a 128x128 píxeles.
- **Normalizar:** Se normaliza la imagen dividiendo los valores de los píxeles por 255.
- **Expandir dimensiones:** Se expande la dimensión de la imagen para que sea compatible con el modelo.
- **Retorno:** Se devuelve la imagen preprocesada y la imagen original.

Función para Detectar Bordes en la Imagen

```
# Función para detectar bordes en La imagen
def detect_edges(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray_image, 50, 150)
    return edges
```

Figura 90: Función para detectar bordes en la Imagen

- **Convertir a escala de grises:** Convierte la imagen a escala de grises.
- **Detectar bordes:** Utiliza el algoritmo de Canny para detectar bordes en la imagen.
- **Retorno:** Devuelve la imagen con los bordes detectados.

Función para Detectar Líneas Usando la Transformada de Hough

```
# Función para detectar líneas usando La transformada de Hough
def detect_lines(image):
    edges = detect_edges(image)
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=50, minLineLength=50, maxLineGap=5)
    return lines
```

Figura 91: Detectar líneas usando Hough

- **Detectar bordes:** Llama a la función `detect_edges` para detectar bordes en la imagen.
- **Detectar líneas:** Utiliza la transformada de Hough probabilística para detectar líneas en la imagen de bordes.
- **Retorno:** Devuelve las líneas detectadas.

Función para Dibujar Líneas Detectadas en la Imagen

```
# Función para dibujar líneas detectadas en la imagen
def draw_lines(image, lines):
    line_image = np.copy(image)
    if lines is not None:
        for line in lines:
            for x1, y1, x2, y2 in line:
                cv2.line(line_image, (x1, y1), (x2, y2), (0, 255, 0), 2)
    return line_image
```

Figura 92: Función – Dibujar Líneas detectadas en la imagen

- **Copiar imagen:** Crea una copia de la imagen original.
- **Dibujar líneas:** Si hay líneas detectadas, se dibujan en la imagen con color verde y grosor 2.
- **Retorno:** Devuelve la imagen con las líneas dibujadas.

Función para Contar Espacios Disponibles

```
# Función para contar espacios disponibles
def count_spaces(lines, img_shape):
    if lines is None:
        return 0
    horizontal_lines = [line for line in lines if abs(line[0][1] - line[0][3]) < 10]
    horizontal_lines = sorted(horizontal_lines, key=lambda line: min(line[0][0], line[0][2]))
    num_spaces = len(horizontal_lines) // 2 # Dividir entre 2 para contar los espacios
    return num_spaces
```

Figura 93: Función para contar espacios disponibles

- **Comprobar líneas:** Si no hay líneas, retorna 0.
- **Filtrar líneas horizontales:** Filtra las líneas que son horizontales (diferencia vertical menor a 10).
- **Ordenar líneas:** Ordena las líneas horizontales por su posición horizontal mínima.
- **Contar espacios:** Divide el número de líneas horizontales por 2 para obtener el número de espacios.
- **Retorno:** Devuelve el número de espacios detectados.

Hacer una Predicción en una Nueva Imagen

```
# Hacer una predicción en una nueva imagen
new_image_path = 'estacionamiento/Prueba.jpg' # Reemplaza con la ruta de tu imagen
preprocessed_image, original_image = preprocess_image(new_image_path)
prediction = model.predict(preprocessed_image)

# Interpretar la predicción
categories = ['vacios', 'llenos', 'medios']
predicted_category = categories[np.argmax(prediction)]
print(f"La imagen se clasifica como: {predicted_category}")
```

Figura 94: Hacer una predicción en una nueva imagen

- **Cargar imagen:** Define la ruta de la nueva imagen.
- **Preprocesar imagen:** Llama a `preprocess_image` para preprocesar la imagen.
- **Hacer predicción:** Utiliza el modelo cargado para predecir la categoría de la imagen.
- **Interpretar predicción:** Convierte la predicción del modelo a una categoría (vacios, llenos, medios) y la imprime.

Detectar Bordos y Líneas

```
# Detectar bordes
edges = detect_edges(original_image)

# Detectar líneas
lines = detect_lines(original_image)
```

Figura 95: Detectar Bordos y Lineas

- **Detectar bordes:** Llama a `detect_edges` para detectar bordes en la imagen original.
- **Detectar líneas:** Llama a `detect_lines` para detectar líneas en la imagen original.

Dibujar Líneas y Contar Espacios

```
# Dibujar líneas en la imagen original
line_image = draw_lines(original_image, lines)

# Contar el número de espacios detectados
num_spaces = count_spaces(lines, original_image.shape)
print(f"Total de espacios encontrados: {num_spaces}")
```

Figura 96: Dibujar Líneas y Contar espacios

- en la imagen original.
- **Contar espacios:** Llama a `count_spaces` para contar el número de espacios disponibles y lo imprime.

Mostrar las Imágenes

```
# Mostrar Las imágenes por separado
plt.figure(figsize=(15, 15))

plt.subplot(3, 1, 1)
plt.title(f'Imagen Original con Predicción: {predicted_category}')
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(3, 1, 2)
plt.title('Bordes Detectados')
plt.imshow(edges, cmap='gray')
plt.axis('off')

plt.subplot(3, 1, 3)
plt.title(f'Líneas Detectadas - Espacios: {num_spaces}')
plt.imshow(cv2.cvtColor(line_image, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.tight_layout()
plt.show()
```

Figura 97:Mostrar Imágenes

- Crea una figura para mostrar las imágenes.
- **Subplot 1:** Muestra la imagen original con la categoría predicha.
- **Subplot 2:** Muestra la imagen con los bordes detectados.
- **Subplot 3:** Muestra la imagen con las líneas detectadas y el número de espacios disponibles.
- **Mostrar figura:** Muestra todas las imágenes en una figura con un diseño ajustado.

Fase 4: Pruebas

Se procesan correctamente las carpetas de muestras para ser entrenadas

```
[1]
... Preprocesamiento completado con éxito; verificar que las imágenes ahora estén en las nuevas carpetas
```

Figura 98: Procesamiento completado – script 1

Se proporciona la carga de imágenes y aumento de datos para el entrenamiento

```
[2]
... Found 77 images belonging to 3 classes.
     Found 17 images belonging to 3 classes.
     Found 19 images belonging to 3 classes.
     Carga de imágenes y aumento de datos completados.
```

Figura 99: Cargar Imágenes y aumento de datos completado

El entrenamiento del modelo es todo un éxito

```
c:\Users\ganap\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/20
c:\Users\ganap\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121:
  self._warn_if_super_not_called()
2/2 ----- 13s 2s/step - accuracy: 0.3396 - loss: 1.5054 - val_accuracy: 0.2941 - val_loss: 1.1585
Epoch 2/20
1/2 ----- 0s 490ms/step - accuracy: 0.1875 - loss: 1.3452
c:\Users\ganap\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121:
  self.gen.throw(typ, value, traceback)
2/2 ----- 1s 679ms/step - accuracy: 0.1875 - loss: 1.3452 - val_accuracy: 0.2941 - val_loss: 1.1476
Epoch 3/20
2/2 ----- 1s 293ms/step - accuracy: 0.2655 - loss: 1.1434 - val_accuracy: 0.2941 - val_loss: 1.0966
Epoch 4/20
2/2 ----- 1s 160ms/step - accuracy: 0.3438 - loss: 1.1084 - val_accuracy: 0.4118 - val_loss: 1.0817
Epoch 5/20
2/2 ----- 1s 458ms/step - accuracy: 0.4906 - loss: 1.0774 - val_accuracy: 0.4118 - val_loss: 1.0785
Epoch 6/20
2/2 ----- 0s 127ms/step - accuracy: 0.4375 - loss: 1.0655 - val_accuracy: 0.4118 - val_loss: 1.0789
Epoch 7/20
2/2 ----- 1s 496ms/step - accuracy: 0.5311 - loss: 1.0399 - val_accuracy: 0.4118 - val_loss: 1.1013
Epoch 8/20
2/2 ----- 1s 152ms/step - accuracy: 0.3438 - loss: 1.1714 - val_accuracy: 0.4118 - val_loss: 1.0778
Epoch 9/20
2/2 ----- 2s 470ms/step - accuracy: 0.4479 - loss: 1.0700 - val_accuracy: 0.4118 - val_loss: 1.0673
Epoch 10/20
2/2 ----- 0s 114ms/step - accuracy: 0.4615 - loss: 1.0701 - val_accuracy: 0.4118 - val_loss: 1.0681
Epoch 11/20
2/2 ----- 1s 476ms/step - accuracy: 0.4393 - loss: 1.0789 - val_accuracy: 0.4118 - val_loss: 1.0694
Epoch 12/20
2/2 ----- 0s 117ms/step - accuracy: 0.4062 - loss: 1.0878 - val_accuracy: 0.4118 - val_loss: 1.0703
Epoch 13/20
2/2 ----- 2s 488ms/step - accuracy: 0.3958 - loss: 1.0939 - val_accuracy: 0.4118 - val_loss: 1.0691
Epoch 14/20
2/2 ----- 0s 118ms/step - accuracy: 0.6923 - loss: 1.0463 - val_accuracy: 0.4118 - val_loss: 1.0641
...
2/2 ----- 1s 476ms/step - accuracy: 0.3949 - loss: 1.0819 - val_accuracy: 0.4118 - val_loss: 1.0509
```

Figura 100: Entrenamiento del modelo

Modelo entrenado creado y almacenado

```
1/1 ----- 0s 235ms/step - accuracy: 0.4211 - loss: 1.0643
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend saving your model as a SavedModel using `keras.saving.save_model(model, filepath)`.
Precisión en el conjunto de prueba: 42.11%
Modelo guardado como 'model_estacionamiento_helen.h5'
```

Figura 101: Modelado entrenado y guardado

Resultado del Script Final

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Modelo cargado exitosamente.
1/1 ----- 0s 161ms/step
La imagen se clasifica como: vacios
Total de espacios encontrados: 66
```

Figura 102: Resultado del análisis de la imagen de prueba

Imagen detectado y analizado

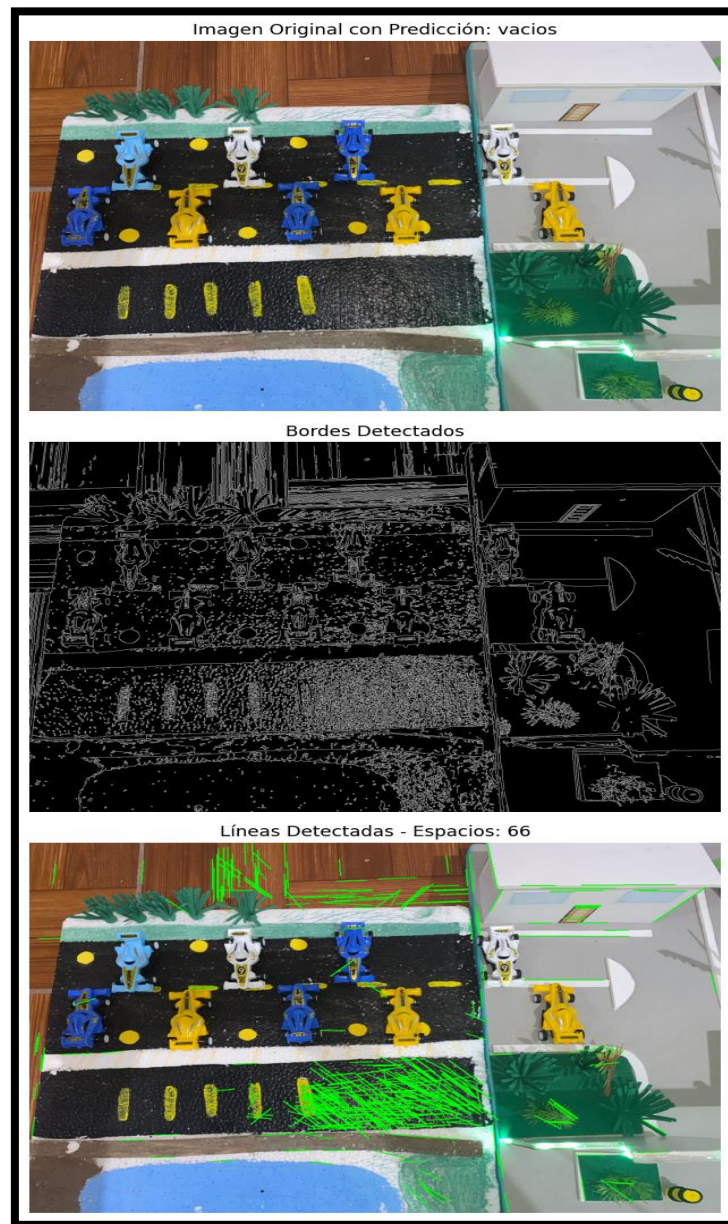


Figura 103: Imagen nuevo con el análisis – resultado final