TECNOLÓGICO NACIONAL DE MÉXICO
CAMPUS CULIACÁN

Opción:
TITULACIÓN INTEGRAL
TESIS PROFESIONAL

“Generación y análisis de secuencias de video para la estimación de la velocidad de un vehículo mediante técnicas de visión artificial y aprendizaje máquina”

ALUMNO:
Mendoza Meza Elmer Josue

CARRERA:
Ingeniería en Tecnologias de la Información y Comunicaciones.

No. DE CONTROL:
16170711

Director:
Gloria Ekaterine Peralta Peñuñuri

Culiacán, Sinaloa, Noviembre de 2021.

Hoja de autorización

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Culiacán

"2021, Año de la Independencia"

DIVISION DE ESTUDIOS PROFESIONALES
ASUNTO: **Autorización de Impresión**

Culiacán, Sinaloa, 05/11/2021 12:15

C. ELMER JOSUÉ MENDOZA MEZA
PASANTE(S) DE LA CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
P R E S E N T E.

De acuerdo con fallo emitido por la Comisión Revisora Integrada por:

		Cédula Profesional
Presidente:	Maestría en Ciencias en Ciencias de la Computación **Gloria Ekaterine Peralta Peñuñuri**	6038275
Secretario:	Doctorado en Ciencias en Ingeniería Eléctrica Opción en Sistemas Computacionales **Héctor Rodríguez Rangel**	9972835
Vocal:	Licenciado en Informática **Nora Esmeralda Cancela García**	3372654

Y considerando que reúne los requisitos establecidos en la Opción: **TITULACIÓN INTEGRAL: TESIS PROFESIONAL "GENERACIÓN Y ANÁLISIS DE SECUENCIAS DE VIDEO PARA LA ESTIMACIÓN DE LA VELOCIDAD DE UN VEHÍCULO MEDIANTE TÉCNICAS DE VISIÓN ARTIFICIAL Y APRENDIZAJE MÁQUINA".**, damos a usted nuestra autorización para que proceda a imprimir su trabajo profesional

Sin otro particular, quedo de usted.

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE CULIACÁN
DEPARTAMENTO DE DIVISIÓN DE ESTUDIOS PROFESIONALES

A T E N T A M E N T E
Excelencia en Educación Tecnológica®

M.C. CARLOS RAFAEL LIZÁRRAGA ARREOLA
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

AGRADECIMIENTOS

A mis padres, por el gran esfuerzo que han dedicado y por todo el tiempo que estuvieron siempre presente conmigo, nunca me dejaron caer, siempre fueron responsables para que yo fuera una persona de bien y llevara conmigo siempre buenos valores, me han enseñado que todo lo que te propongas se puede lograr, que nunca hay que desistir por un sueño, me han dado todo su cariño incondicionalmente y eso me ha motivado en concluir mis estudios, formándome para ser un gran profesional.

A mis hermanos que siempre estuvieron presentes también en la trayectoria de mis estudios y de mi vida, apoyándome cada que lo necesite, me dieron buenos consejos para salir adelante y me enseñaron a ser alguien responsable.

A mis grandes amigos que siempre me brindaron su apoyo y la lealtad de saber que siempre estarán para lo que necesite.

A mis profesores que me fueron guiando en todo el proceso de mi carrera profesional, gracias a ellos es que estoy culminando una etapa más.

Generación y análisis de secuencias de video para la estimación de la velocidad de un vehículo mediante técnicas de visión artificial y aprendizaje máquina.

RESUMEN

Actualmente la demanda del uso automovilístico en las ciudades se incrementó demasiado, lo que ha provocado un aumento en accidentes de tránsito. Para ayudar a contribuir a reducir esta problemática se hará el uso de la inteligencia artificial (IA), de la cual a través de técnicas de machine learning y filtro de Kalman se diseñarán soluciones a través de tomas de secuencias (Videos) que brindaran información confiable que son las cantidades de vehículos que transitan por las áreas que deseen estudiar, para ello es necesario tener estudios de transito vial. El propósito es desarrollar un sistema capaz de brindar soluciones automatizadas de una manera eficiente para ayudar a la toma de decisiones en mejoras viales.

El sistema será capaz de clasificar, medir la velocidad y obtener la trayectoria de los vehículos proporcionando datos de suma importancia para el personal de vialidad y se podrán tomar mejores decisiones para un mejor flujo vial y probablemente tener una tasa más chica de accidentes.

CONTENIDO

1 INTRODUCCIÓN *8*

1.1. Antecedentes 9

1.2 Descripción del problema 10

1.3 Hipótesis o pregunta de investigación 10

1.4 Objetivo general 10

1.5 Objetivos especificos 10

1.6 Justificación 11

1.7 Alcance 11

2 FUNDAMENTO TEÓRICO *12*

2.1 Python 12

2.2 OpenCV 12

2.3 OpenCV-Python 13

2.4 YOLO 13

2.5 Tensorflow 13

2.6 Inteligencia Artificial 13

2.7 Visión Artificial 14

2.8 Machine Learning 14

2.9 Filtro de Kalman 15

3 PROCEDIMIENTO *16*

3.1 Estudio de la locación. 16

3.2 Evaluación del equipo 17

3.3 Toma de secuencias 19

3.4 Limpieza de los videos 20

3.5 Implementación de las técnicas de Machine Learning 21

4 RESULTADOS *33*

5 CONCLUSIONES *37*

6 COMPETENCIAS DESARROLLADAS *39*

7 REFERENCIAS *40*

Lista de figuras

Figura 0: Esquema del procedimiento explicado. 16
Figura 1: Locación elegida para toma de secuencias. 17
Figura 2: Radar de velocidad 17
Figura 3: Smartphone 18
Figura 4: Tripie 18
Figuta 5: Video 1 Figura 6: Video 2 19
Figura 7: Video 3 Figura 8: Video 4 19
figura 9: Archivo .txt de datos requeridos por el sistema. 20
Figura 10: Código Machine Learning 22
Figura 11: Método Tracker 22
Figura 12: Código de redimensión de imagen 23
Figura 13: Código de retorno de seguimientos. 23
Figura 14: Código de limitadores. 24
Figura 15: Código de obtención de información del tiempo. 24
Figura 16: Código de dibujo de caja en el vehículo. 24
Figura 17: Código para guardar video de salida. 25
Figura 18: Código para guardar CSV. 25
Figura 19: Código donde se llama el método para verificar coincidencias. 26
Figura 20: Código de detección del vehículo. 26
Figura 21: Código de agregar nuevo vehículo al seguimiento. 27
Figura 22: Código de mantener vehículo perdido en lista. 27
Figura 23: Código para eliminar vehículos y guardar los ID buenos. 28
Figura 24: Código de respaldo de ID eliminados. 28
Figura 25: Código de retorno de listas. 28
Figura 26: Código para garantizar y guardas las detecciones de vehículos. 29
Figura 27: Código para garantizar que ya se tenga la imagen. 29
Figura 28: Código para detener acción de no guardar nada. 30
Figura 29: Código para obtener recta, ángulo y dimensiones. 30
Figura 30: Código para generar recta. 30
Figura 31: Código para sacar las áreas de entrada y descenso. 31
Figura 32: Código para agregar carril. 31
Figura 33: Código para guardar parámetros. 32
Figura 34: Código para guardar CSV. 32
Figura 35: Ejecución Por Terminal 33
Figura 36: Detección De Auto Punto A 34
Figura 37: Detección De Auto Punto B 34
Figura 38: Trayectoria 35
Figura 39: Todas las clases que se pueden detectar. 35
Figura 40: Contador de clases. 36
Figura 41: Resultado Final. 36

Lista de Graficas

Grafica 1: Objetos Detectados 36

Lista de Diagramas

Diagrama de flujo 1: Metodo YOLO *21*

1 INTRODUCCIÓN

La inteligencia y la visión artificial son sistemas muy avanzados capaces de lograr la detección de objetos, así como su clasificación, entre otras tareas; aportando grandes soluciones de distintas áreas, gracias al gran procesamiento de datos que pueden manejar y brindan aportaciones muy eficientes.

Para emular la vista humana, las máquinas necesitan adquirir, procesar, analizar y entender imágenes. El enorme crecimiento en el logro de este hito, se hizo gracias al proceso de aprendizaje interactivo, hecho posible con las redes neuronales. El proceso de aprendizaje comienza con un conjunto de datos recopilados con información que ayuda a la máquina a aprender un tema específico. Si el objetivo es identificar vídeos de gatos como lo fue para Google en el 2012, el conjunto de datos utilizado por las redes neuronales necesita tener imágenes y videos con gatos, así como ejemplos sin gatos. (Visión Artificial, 2020).

El objetivo del desarrollo del proyecto es detectar los vehículos por medio de secuencias de video a través de técnicas de machine learning y aprendizaje máquina con el fin de poder estimar su velocidad y saber qué tipo de vehículo es el que mayormente está transitando. Para esta tarea se requiere conocimientos de inteligencia artificial (IA) y del uso de herramientas (software).

1.1. Antecedentes

En este apartado se presentarán los antecedentes de los distintos avances de la visión artificial, mencionando algunas opiniones de diferentes autores.

Las primeras investigaciones relacionadas con la visión artificial surgieron en la década de los años 50. La facilidad con la que los seres humanos podemos conocer todo lo que contiene nuestro entorno, hizo creer que un ordenador podría interpretar las imágenes que recibe del exterior de una manera relativamente fácil (Sucar & Gómez, 2011).

A pesar de estos resultados y del entusiasmo inicial, la visión artificial no resultó tan fácil como se esperaba, y se acabó demostrando que es un problema muy complejo. En los años siguientes lo único que se consiguieron en las investigaciones fueron fracasos y frustraciones por no obtener buenos resultados. Esto se debía principalmente a que no existía una tecnología tan avanzada como para poder realizar procesamientos mejores. Sin embargo, en los años siguientes se desarrollaron algoritmos muy importantes como los detectores de bordes de Roberts (1965), Prewitt (1970) o Sobel (1970) que se siguen utilizando hoy en día. Por aquel entonces, estos algoritmos eran aplicables solo para unos casos muy concretos, estaban muy limitados. Por todo esto, en los años 70 se puede apreciar una disminución en las investigaciones y desarrollos sobre la visión artificial, fallando las predicciones que se hicieron a mediados de siglo y comenzando lo que se conoce como el “Invierno de la Inteligencia Artificial” (Ferro, 2018).

1.2 Descripción del problema

La contaminación ambiental y los accidentes automovilísticos han estado en crecimiento cada vez más conforme pasa el tiempo, su agraviamiento en gran escala se debe a la gran demanda de uso de vehículos por la sociedad, que por lo tanto parcialmente genera una mayor contaminación. Estos problemas incrementan el número de accidentes provocando más tráfico, embotellamientos en la ciudad y en lo peor de los casos muertes.

1.3 Hipótesis o pregunta de investigación

Las nuevas tecnologías serán capaces de formular mejores soluciones y más rápidas que los humanos, siendo capaz de identificar una gran cantidad de objetos de una manera más precisa. Cumpliendo con el objetivo de este proyecto, el sistema será capaz de clasificar, medir la velocidad y obtener la trayectoria de los vehículos proporcionando datos de suma importancia para el personal de vialidad y se podrán tomar mejores decisiones para un mejor flujo vial y una tasa más chica de accidentes.

1.4 Objetivo general

Generación y análisis de secuencias de video para la estimación de la velocidad y la clasificación de un vehículo mediante técnicas de visión artificial y aprendizaje máquina.

1.5 Objetivos especificos

- Diseñar e implementar un sistema que permita identificar y clasificar un vehículo.
- Diseñar e implementar un sistema que determine la trayectoria de un vehículo.
- Diseñar, implementar y entrenar una red neuronal que permita determinar la velocidad del vehículo.
- Diseñar e implementar una herramienta que a partir de una secuencia de video determine la velocidad de un vehículo.

1.6 Justificación

La idea del proyecto fue creada ya que, a través del tiempo, el número de habitantes y vehículos en circulación se ha incrementado, aumentando los tiempos de traslado, una mayor contaminación y un gran número de accidentes, lo cual se vuelve un caos actualmente para todas las personas que manejan un automóvil y también un problema para el medio ambiente. Se busca apoyar en la solución estos problemas, con el fin de apoyar a los expertos en el área a tomar mejores decisiones. La solución que se propone es desarrollar un DataSet que servirá para alimentar a una red neuronal para que aprenda a clasificar los objetos (en este caso los automóviles), y a estimar su velocidad para así generar un análisis vehicular para que los expertos en el área tengan una información más exacta de lo que ocurre diariamente y tomen una mejor decisión para mejorar los problemas viales que hay actualmente.

1.7 Alcance

Para un analisis mas concreto de la prediccion del numero de vehiculos que transita por un punto especifico se pretende entrenar la red neuronal con una capacidad de 5 mil video aproximadamente, el cual el sistema sera capaz de detectar con una mejor precisión y con una mayor veracidad. El sistema ayudara a personal de transito vehicular a conocer de mejor la cantidad de vehiculos que a distintas horas transitan, con el fin de implementar medidas mas seguras en las areas seleccionada para reducir accidentes que puedan ocurrir constantemente.

Este sistema podra verse reflejado en un sistema de PC y sera exclusica para expertos en el area vial.

2 FUNDAMENTO TEÓRICO

En el presente proyecto se centrará en la detección de vehículos por medio de técnicas de visión artificial y aprendizaje maquina que se aplicaran en un sistema, el cual podrá ser utilizado por personal capacitado de viabilidad publica para tener un resultado concreto de los automóviles q transitan en distintas horas. Para comprender de una manera mas concreta este trabajo fue necesario definir los siguientes conceptos que brindaran una mejor comprensión.

2.1 Python

Python es un lenguaje de programación interpretado cuya principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación. Además, posee una serie de características que lo hacen muy particular y que, sin duda, le aportan muchas ventajas y están en la raíz de su uso tan extendido. (ESIC, 2021).

Es un lenguaje de programación relativamente fácil de aprender, y las herramientas necesarias están disponibles para todos de forma gratuita. Esto hace que sea accesible para casi todo el mundo. Si dispones de tiempo para aprender, conseguirás crear esos proyectos que tienes en mente. (Aula21, 2020).

2.2 OpenCV

OpenCV (Open Source Computer Vision) es una librería software open-source de visión artificial y machine learning.

OpenCV provee una infraestructura para aplicaciones de visión artificial.

La librería tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial para usar.

Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios.OpenCV está escrito en C++, tiene interfaces en C++, C, Python, Java y MATLAB interfaces y funciona en Windows, Linux, Android y Mac OS. OX

2.3 OpenCV-Python

OpenCV-Python es una biblioteca de enlaces Python diseñados para resolver problemas de visión por computadora.

OpenCV-Python hace uso de Numpy, que es una biblioteca altamente optimizada para operaciones numéricas con una sintaxis de estilo MATLAB. Todas las estructuras de matrices OpenCV se convierten a y desde matrices Numpy. Esto también facilita la integración con otras bibliotecas que usan Numpy, como SciPy y Matplotlib.

2.4 YOLO

Es un tipo de técnica muy novedosa (2016), acrónimo de “You Only Look Once” y que es la más rápida del momento, permitiendo su uso en video en tiempo real.

Esta técnica utiliza un tipo de red Neuronal Convolucional llamada Darknet para la clasificación de imágenes y le añade la parte de la detección, es decir un “cuadradito” con las posiciones x e y, alto y ancho del objeto encontrado. (YOLO, 2020).

2.5 Tensorflow

Es una plataforma de código abierto diseñado para el aprendizaje automático. Tiene una gran capacidad de herramientas muy flexibles que permiten que los demás puedan innovar cada vez más utilizando el aprendizaje automático y crear nuevas aplicaciones. Capaz de construir y entrenar redes neuronales que sirven para descifrar patrones.

2.6 Inteligencia Artificial

En términos sencillos, inteligencia artificial (IA) se refiere a los sistemas o las máquinas que imitan la inteligencia humana para realizar tareas y que tienen la capacidad de mejorar iterativamente a partir de la información que recopilan.

La IA se trata mucho más sobre el proceso y la capacidad de pensamiento superpoderado y el análisis de datos que sobre cualquier formato o función en particular. Aunque la IA muestra imágenes de robots de aspecto humano de alto funcionamiento que se apoderan del mundo, la IA no pretende reemplazar a los humanos. Su objetivo es mejorar significativamente las capacidades y contribuciones humanas. (IA, 2019).

2.7 Visión Artificial

La visión artificial es una disciplina científica que incluye métodos para adquirir, procesar y analizar imágenes del mundo real con el fin de producir información que pueda ser tratada por una máquina. La principal finalidad de la visión artificial es dotar a la máquina de “ojos” para ver lo que ocurre en el mundo real, y así poder tomar decisiones para automatizar cualquier proceso.

Una manera simple de comprender este sistema es basarnos en nuestros propios sentidos. Los humanos usamos nuestros ojos para comprender el mundo que nos rodea, y la visión artificial trata de producir ese mismo efecto en máquinas. Éstas podrán percibir y entender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación.

La comprensión en los dispositivos se consigue gracias a una descomposición de la imagen en pequeños fragmentos (píxeles) y en su posterior estudio. (Contaval, 2018).

2.8 Machine Learning

Machine learning es una forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita. Sin embargo, machine learning no es un proceso sencillo. Conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en datos. Un modelo de machine learning es la salida de información que se genera cuando entrena su algoritmo de machine learning con datos. Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida. (IBM, s.f.).

2.9 Filtro de Kalman

El Filtro de Kalman es uno de los algoritmos de estimación más importantes y comunes. El filtro de Kalman produce estimaciones de variables ocultas basadas en mediciones inexactas e inciertas. Además, el filtro de Kalman proporciona una predicción del estado futuro del sistema, basado en estimaciones pasadas.

Hoy en día, el filtro Kalman se utiliza en objetivos de seguimiento (radar), sistemas de localización y navegación, sistemas de control, gráficos por computadora y mucho más. (Alex Becker, 2018).

3 PROCEDIMIENTO

En esta etapa se describirán paso por paso el proceso que se llevo a cabo para poder llegar al objetivo, las etapas son las siguientes:

- Estudiar la locación
- Evaluación del equipo
- Toma de secuencias
- Limpieza de videos
- Implementación de las técnicas de Machine Learning

A continuación, en la figura [0] un esquema explicado del procedimiento a llevar:

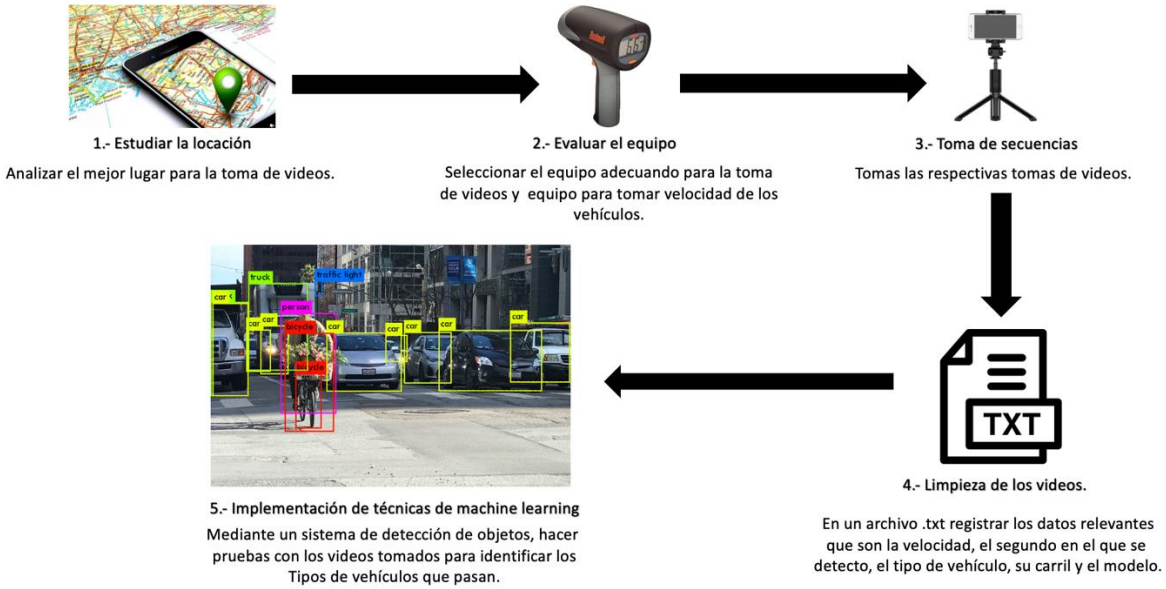

Figura 0: Esquema del procedimiento explicado.

3.1 Estudio de la locación.

Al estar analizando varios puntos factibles para la toma de videos se determinó en concreto un lugar en específico para ello, ya que permitió obtener una gran cantidad de datos por su excelente ubicación y la gran cantidad de vehículos que transitaban a todas horas del día.

La ubicación que se eligió para las pruebas se muestra en la siguiente figura [1]:

Boulevard Calzada de las torres, frente a la plaza comercial Paseo San Isidro en Culiacán Sinaloa.



Figura 1: Locación elegida para toma de secuencias.

3.2 Evaluación del equipo

Una vez tenido ya elegida la locación, se evaluaron los equipos necesarios que fueron de gran ayuda para la toma de videos, estos equipos son un Smartphone con cámara de video, un radar de velocidad y un tripie.

En las figuras [2,3 y 4] se muestran los equipos seleccionados.



Figura 2: Radar de velocidad

Se decidió usar esta herramienta para obtener la velocidad exacta en la que los vehículos pasaban.



Figura 3: Smartphone

Para la toma de videos se utilizó un celular de alta gama con una cámara de excelente calidad para tener una mejor representación de la transición de video.



Figura 4: Tripie

La elección de utilizar un tripie fue para ayudarnos a mantener el smartphone en una posición fija y así obtener una mejor calibración de los videos.

3.3 Toma de secuencias

Al tener todo el equipo listo se procedió a recurrir a la locación elegida para comenzar con la toma de los videos,los cuales en un siguiente proceso se prepararon en un sistema para interpretarlos y obtener datos importantes como su velocidad, clasificación, trayectoria etc. En las figuras de [5,6,7 y 8] se muestran evidencias de los videos tomados en la locación elegida.



Figuta 5: Video 1



Figura 6: Video 2

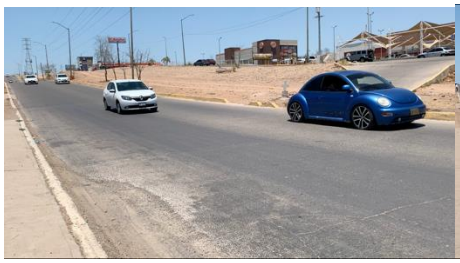

Figura 7: Video 3



Figura 8: Video 4

Como se puede observar en las distintas imágenes de diferentes videos tomados, se hizo una prueba de campo en donde se permaneció en el lugar por varias horas y varias semanas con el fin de obtener una gran de datos que en este caso son los vehículos.

La forma en que se logró hacer esta tarea fue de la siguiente manera:

Se procedió a colocar el Smartphone en el tripie colocándolo a una última de un metro y medio aproximadamente para obtener una imagen amplia del punto deseado, una vez de tener listo los dispositivos se comenzó a tomar la grabación del video que consistió en tomar el radar y cada que pasara un vehículo apuntarle con el radar para obtener su velocidad y la cual al saberla se pronunciaba fuerte para que el video grabara en voz la velocidad en la que pasó cada vehículo.

Para obtener una buena cantidad de datos por semana se estableció una meta de obtener alrededor de 500 y 1000 datos para poder entrenar de manera eficiente la red neuronal.

3.4 Limpieza de los videos

Para esta actividad se generó un archivo .txt con información relevante al video la cual constó de asignar el segundo exacto en que pasaba un vehículo por cierto punto, así como la marcar a la velocidad en la que paso, que tipo de vehículo era y por cual carril del boulevard estaba pasando.

En la figura [9] se muestra un ejemplo del archivo .txt.

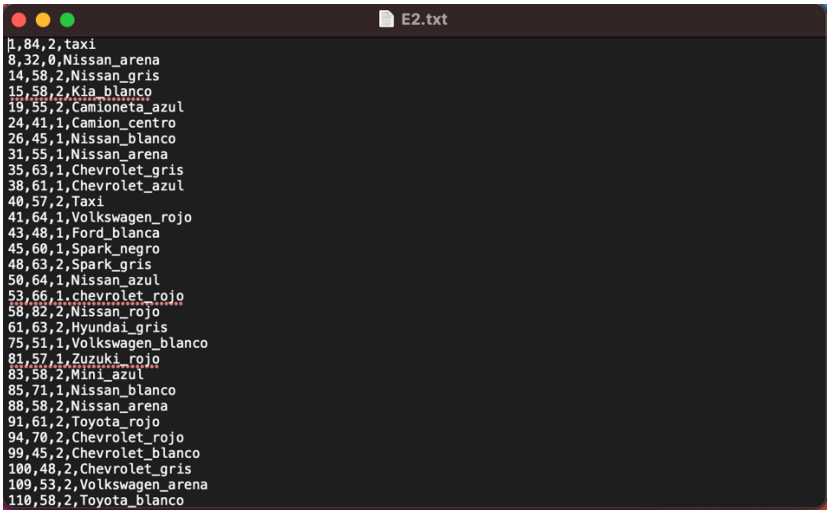

figura 9: Archivo .txt de datos requeridos por el sistema.

Para la elaboración del archivo .txt, se pueden observar en la figura [9] los distintos parámetros ya mencionados, en el primer parámetro se asignaron los segundos, el en seguro parámetro se asignó la velocidad del vehículo, en el tercer parámetro se asignó el carril por donde pasó y el cuarto parámetro es el modelo del vehículo.

3.5 Implementación de las técnicas de Machine Learning

Para llevar a cabo del desarrollo de esta técnica se utilizó YOLO, que es una técnica de detección de objetos en tiempo real. Fue útil para la detección de los vehículos, así como su clase, es decir, si es un carro, un autobús, una bicicleta etc.

En el diagrama de flujo [1] de explica a grandes rasgos la funcion del metodo YOLO:

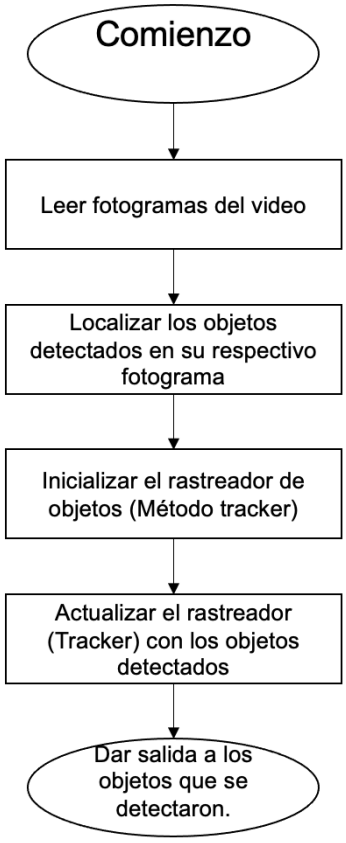

Diagrama de flujo 1: Metodo YOLO

El código desarrollado se muestra en la figura [10] :

```
        if FLAGS.framework == 'tflite':
            interpreter.set_tensor(input_details[0]['index'], image_data)
            interpreter.invoke()
            pred = [interpreter.get_tensor(output_details[i]['index']) for i in range(len(output_details))]
            if FLAGS.model == 'yolov3' and FLAGS.tiny == True:
                boxes, pred_conf = filter_boxes(pred[1], pred[0], score_threshold=0.25,
                                                input_shape=tf.constant([input_size, input_size]))
            else:
                boxes, pred_conf = filter_boxes(pred[0], pred[1], score_threshold=0.25,
                                                input_shape=tf.constant([input_size, input_size]))
        else:
            batch_data = tf.constant(image_data)
            pred_bbox = infer(batch_data)
            for key, value in pred_bbox.items():
                boxes = value[:, :, 0:4]
                pred_conf = value[:, :, 4:]

        boxes, scores, classes, valid_detections = tf.image.combined_non_max_suppression(
            boxes=tf.reshape(boxes, (tf.shape(boxes)[0], -1, 1, 4)),
            scores=tf.reshape(
                pred_conf, (tf.shape(pred_conf)[0], -1, tf.shape(pred_conf)[-1])),
            max_output_size_per_class=50,
            max_total_size=50,
            iou_threshold=FLAGS.iou,
            score_threshold=FLAGS.score
        )
```

Figura 10: Código Machine Learning

Este código permite realizar la detección de los vehículos diciendo que tipo de clase es.

Para realizar el seguimiento del vehículo se generó un método para hacer este trabajo, el cual se encarga de hacer un match de las predicciones anteriores y la predicción actual.

El método se puede observar en la figura [11]:

```
def gen_trackers(prediction, max_age, min_hits, tracker_list, track_id_list,
    x_limits, g_vars, cut_img=None, out_csv=False
    ):

    bboxes, scores, classes = prediction
    x_box =[]

    if len(tracker_list) > 0:
        for trk in tracker_list:
            x_box.append(trk.box)


    matched, unmatched_dets, unmatched_trks = assign_detections_to_trackers(
            x_box, bboxes, iou_thrd = 0.3
        )
```

Figura 11: Método Tracker

Al tener ya la predicción de YOLO, se separan en 3 variables

- Boxes
- Scores
- Classes

Las cuales sirven para saber qué tan cierta es la clase del tipo de vehículo y asignar que clase es para cada uno.

Para proseguir con la detección y seguimiento del vehículo se realizó el siguiente código mostrado en la figura [12]:

```
pred_bbox = [boxes.numpy(), scores.numpy(), classes.numpy(), valid_detections.numpy()]
pred_bbox = TLI_utils.post_prediction(pred_bbox, frame_size,
    cut_img= FLAGS.c_img, score= FLAGS.score)
```

Figura 12: Código de redimensión de imagen

Aquí lo que se hace es que redimensiona la imagen a 416 x 416, YOLO regresa los boxes con cada clase y las vuelve a redimensionar al tamaño original de la imagen.

Por cada detección que se realiza envía al método principal gen_trackers, donde ese método regresa todos los seguimientos que se van registrando, así como todos sus ID (Vehículos detectados) y también todos los ID que son válidos.

Para realizar esta operación se realizó el siguiente código mostrado en la figura [13]:

```
tracker_list, track_id_list, good_tracker_list = gen_trackers(pred_bbox,
    FLAGS.max_age, FLAGS.min_hits, tracker_list, track_id_list,
    FLAGS.x_lim, g_vars, cut_img=FLAGS.c_img, out_csv=False
)
```

Figura 13: Código de retorno de seguimientos.

Para obtener una precisión de la posición del vehículo es importante establecer limitadores del punto de inicio al punto final.

Para esta acción se elaboró el siguiente código mostrado en la figura [14]:

```
result = np.asarray(frame)
result = cv2.cvtColor(result, cv2.COLOR_RGB2BGR)
TLI_utils.draw_limits_area(result, FLAGS.x_lim, FLAGS.c_img)
```

Figura 14: Código de limitadores.

Así como se dibujaron los limitadores, obtener información relevante como es el tiempo, los frames que se han procesado y el frame actual, para lograr esta tarea se elaboró el siguiente código mostrado en la figura [15]:

Figura 15: Código de obtención de información del tiempo.

```
if FLAGS.time:
    fps = 1.0 / (time.time() - start_time)
    seconds = time.time() - start_time_r

    TLI_utils.set_time_str(result, seconds, pre_txt="Precess Time: ")
    seconds = TLI_utils.get_seconds_from_fps(g_vars["f_count"], g_vars["fps"])
    TLI_utils.set_time_str(result, seconds, pos=(30, 60),
        pre_txt="Real Time: ")
    TLI_utils.set_text(result, "FPS: {:.3f}, Current Frame: {}"
        .format(g_vars["fps"], g_vars["f_count"]), pos=(30, 90)
    )
    TLI_utils.set_text(result, "FPS P: {}".format(fps), pos=(30, 120))
```

Para realizar la selección del vehículo o en otras palabras para dibujar la caja o rectángulo que aparece en el vehículo se elaboró el siguiente código mostrado en la figura [16]:

```
for trk in good_tracker_list:
    TLI_utils.draw_tracker(result, trk)
```

Figura 16: Código de dibujo de caja en el vehículo.

Además de dibujar la caja también dibuja la recta generada a partir de ese tracker, es decir, la línea que se dibuja cada que se detecta un vehículo.

Una vez dibujada las cajas se debe almacenar en un video de salida las cajas de los vehículos, así como su seguimiento, para ello se elaboró el siguiente código mostrado en la figura [17]:

```
if FLAGS.output:
    out.write(result)

if cv2.waitKey(1) & 0xFF == ord('q'): break
g_vars["f_count"] += 1
```

Figura 17: Código para guardar video de salida.

Se puede observar también que se puede detener el proceso si así se requiere solo oprimiendo la letra “q”.

El siguiente paso es almacenar la información en un archivo CSV con el de veces que una clase o tipo de vehículo paso por el video. Para esto se realizó de la siguiente manera, mostrado en la figura [18]:

```
if FLAGS.count:
    print ('clases', classes_names)
    print ('contador', count_clases)
    pass
```

Figura 18: Código para guardar CSV.

Teniendo recabada toda la información de las predicciones que YOLO arrojó, entonces se procedió a validar que esa información sea precisa, principalmente se desarrolló un código para verificar si hay coincidencias de la predicción actual con la anterior, es decir, que si pasaron 5 vehículos se verifique que fue verdadero.

Para ello se elaboró el siguiente código mostrado en la figura [19]

```
matched, unmatched_dets, unmatched_trks = assign_detections_to_trackers(
        x_box, bboxes, iou_thrd = 0.3
    )
```

Figura 19: Código donde se llama el método para verificar coincidencias.

Para validar estas acciones del match se elaboró el siguiente código mostrado en la figura [20]:

```
if matched.size >0:
    for trk_idx, det_idx in matched:
        z = bboxes[det_idx]
        z = np.expand_dims(z, axis=0).T
        tmp_trk= tracker_list[trk_idx]
        tmp_trk.score = scores[det_idx]
        tmp_trk.kalman_filter(z)
        xx = tmp_trk.x_state.T[0].tolist()
        xx =[xx[0], xx[2], xx[4], xx[6]]
        x_box[trk_idx] = xx
        tmp_trk.add_box(xx)
        tmp_trk.hits += 1
        tmp_trk.no_losses = 0
```

Figura 20: Código de detección del vehículo.

La acción realizada en este código es simple pero muy funcional, lo que se está haciendo es que, en dado caso que se detecta el mismo vehículo es decir que coincida con el de la predicción, entonces se agrega al seguimiento que se está haciendo.

Sin embargo, puede que la predicción del YOLO haya fallado y no sea muy precisa por lo que es normal que no haya detectado algún vehículo por lo que se realizó un código que, si en la predicción eran 5 vehículos, pero en realidad son 6 lo que hace es que agrega este nuevo vehículo al seguimiento. Esto método se muestra en la figura [21]:

```
if len(unmatched_dets)>0:
    for idx in unmatched_dets:
        z = bboxes[idx]
        z = np.expand_dims(z, axis=0).T
        tmp_trk = Tracker(x_limits=x_limits, g_vars=g_vars) # Create a new tracker
        tmp_trk.class_id = classes[idx]
        tmp_trk.score = scores[idx]
        x = np.array([[z[0], 0, z[1], 0, z[2], 0, z[3], 0]]).T
        tmp_trk.x_state = x
        tmp_trk.predict_only()
        xx = tmp_trk.x_state
        xx = xx.T[0].tolist()
        xx =[xx[0], xx[2], xx[4], xx[6]]
        tmp_trk.add_box(xx)
        tmp_trk.id = track_id_list.popleft() # assign an ID for the tracker
        tracker_list.append(tmp_trk)
        x_box.append(xx)
```

Figura 21: Código de agregar nuevo vehículo al seguimiento.

Por otra parte, también se consideró la falta de algún vehículo es decir que en la predicción eran 5 carros, pero al hacer el match solo se detectaron 4, aun así, el aunque 1 vehículo se haya perdido en el proceso se va a seguir agregando a la lista. Para este proceso se realizó el siguiente código mostrado en la figura [22]:

```
if len(unmatched_trks)>0:
    for trk_idx in unmatched_trks:
        tmp_trk = tracker_list[trk_idx]
        tmp_trk.no_losses += 1
        tmp_trk.predict_only()
        xx = tmp_trk.x_state
        xx = xx.T[0].tolist()
        xx =[xx[0], xx[2], xx[4], xx[6]]
        tmp_trk.add_box(xx)
        x_box[trk_idx] = xx
```

Figura 22: Código de mantener vehículo perdido en lista.

Ahora teniendo en cuenta que un vehículo que se perdió en la secuencia no es detectado tras varias pruebas superando así su capacidad máxima de veces perdidas, lo que procede es eliminarlo y a la vez guardar todos los ID que fueron buenos.

Para este proceso se elaboró el siguiente código mostrado en la figura [23]:

```
good_tracker_list =[]
for trk in tracker_list:
    if ((trk.hits >= min_hits) and (trk.no_losses <=max_age)):
        good_tracker_list.append(trk)
deleted_tracks = filter(lambda x: x.no_losses >max_age, tracker_list)
```

Figura 23: Código para eliminar vehículos y guardar los ID buenos.

De igual manera lo que se hace después de haber eliminado esos ID (Vehículos) es que respalda los ID eliminados para en un seguimiento posterior volver a utilizarlos y así no dejar ID en blanco. Para esto se realizó el siguiente código mostrado en la figura [24]

```
for trk in deleted_tracks:
        track_id_list.append(trk.id)
```

Figura 24: Código de respaldo de ID eliminados.

Una vez se hayan realizado los procesos del match se retornan la lista con la información que se pidió, que es, todos los seguimientos, los ID, y ID que si me sirven. Para esta acción se realizó el siguiente código mostrado en la figura [25]

```
tracker_list = [x for x in tracker_list if x.no_losses<=max_age]
return tracker_list, track_id_list, good_tracker_list
```

Figura 25: Código de retorno de listas.

Al realizar todo el proceso del mach donde se valida que las predicciones sean ciertas, necesitamos ir guardando todas las detecciones correctas que se realizaron. Para esto se creó el siguiente código mostrado en la figura [26]:

```
if trk.limits_ind[0] is not None and \
        trk.limits_ind[1] is None \
    and not trk.saved[0] and img is not None:
    straight1 = trk.straights[trk.limits_ind[0]]
    angle1 = trk.angles[trk.limits_ind[0]]

    if len(straight1) != 2:
        return

    img_w = np.copy(img)
    utils.draw_triangule(straight1, img_w)
    utils.draw_all_boxes(img_w, [trk.box], box_color=(255, 255, 255))
    utils.draw_limits_area(img_w, trk.x_limits, trk.g_vars["c_img"])
    trk.imgs.append(img_w)
    trk.saved[0] = True
```

Figura 26: Código para garantizar y guardas las detecciones de vehículos.

Prácticamente lo que se realiza es que garantiza que se hizo la primera detección de un vehículo, lo valida y guarda esa imagen.

De igual forma se realizó un proceso mostrado en la figura [27], para garantizar que si se haya guardado correctamente la imagen

```
if trk.limits_ind[0] is None or \
        trk.limits_ind[1] is None \
    or trk.saved[1]:
    return
```

Figura 27: Código para garantizar que ya se tenga la imagen.

En dado caso que la detección no esté en el segundo indicado, no se ejecuta nada. Para esto se elaboró el siguiente código mostrado en la figura [28]

```
if len(trk.g_vars["seconds"]) == 0:
    return
```

Figura 28: Código para detener acción de no guardar nada.

Para poder obtener la recta del seguimiento, ángulo y las dimensiones del punto que inicio al punto de entrada y del punto que inicio al punto salida, se realizó el siguiente código mostrado en la figura [29]

```
straight1 = trk.straights[trk.limits_ind[0]]
angle1 = trk.angles[trk.limits_ind[0]]
dim1 = trk.dimensions[trk.limits_ind[0]]

if len(straight1) != 2:
    return
#punto de salida
straight_all = trk.straights[-1]
angle_all = trk.angles[-1]
dim_all = trk.dimensions[-1]
```

Figura 29: Código para obtener recta, ángulo y dimensiones.

El código para generar la reta desde el punto inicial al vehículo al punto final se muestra en la figura [30]

```
straight2 = np.array([np.array([trk.x_limits[0], y_r]), straight_all[-1]])
angle2 = np.copy(angle_all)
dim2 = []
dim2.append(straight2[1][0] - straight2[0][0])
dim2.append(straight2[1][1] - straight2[0][1])
dim2.append(math.sqrt(dim2[0]**2 + dim2[1]**2))
```

Figura 30: Código para generar recta.

Por otra parte, se necesitó conocer el área en la que el vehículo entro y el área en la que descendió. Para esto se elaboró el siguiente código mostrado en la figura [31]:

```
if angle1[0] == 0 or angle1[1] == 0 or \
    angle2[0] == 0 or angle2[1] == 0:
    return

area1 = trk.areas[trk.limits_ind[0]]
area2 = trk.areas[-1]

top, left, bottom, right = trk.boxes[trk.limits_ind[0]]
perimeter1 = [right-left, bottom-top]

top, left, bottom, right = trk.boxes[-1]
perimeter2 = [right-left, bottom-top]

counter = (len(trk.areas) - trk.limits_ind[0]) - 1
time =  counter / trk.g_vars["fps"]
```

Figura 31: Código para sacar las áreas de entrada y descenso.

Fue necesario conocer la posición por la cual pasaba el vehículo por lo que se desarrolló un código para saber sobre que carril estaba pasando dicho vehículo y lo agrega a un formato. Para esto se elaboró el siguiente código mostrado en la figura [32]

```
secs = trk.g_vars["seconds"][0]
speed_lane = ""
if len(secs) >= 3:
    speed_lane = "{},{}".format(secs[1], secs[2])
else:
    speed_lane = "{}".format(secs[1])
```

Figura 32: Código para agregar carril.

Una vez que se obtuvieron todos los parámetros requeridos se guardaron. En la figura [33] se puede observar el código desarrollado.

```
line = "{},{},{},{},{},{},{},{},{},{},{},{},{:.15f},{:.15f},{:.15f},{:.15f},{:.15f},{},{},{}\n".format(
        ",".join(map(lambda x: "{:.15f}".format(x), angle1)),
        ",".join(map(lambda x: "{:.15f}".format(x), angle2)),
        ",".join(map(str, dim1)),
        ",".join(map(str, dim2)),
        area1,
        area2,
        ",".join(map(lambda x: "{:.15f}".format(x), angle_all)),
        ",".join(map(str, dim_all)),

        ",".join(map(lambda x: ",".join(map(str,x)), straight_all)),

        ",".join(map(str, trk.x_limits)),

        ",".join(map(str, perimeter1)),
        ",".join(map(str, perimeter2)),
        trk.g_vars["fps"],
        trk.g_vars["width"],
        trk.g_vars["height"],
        time,
        trk.score,
        trk.class_id,
        speed_lane,
        "{}_{}".format(trk.g_vars["file_name"], trk.g_vars["f_count"])
    )
```

Figura 33: Código para guardar parámetros.

Y posteriormente guardar los datos en un archivo CSV.

El código para guardar los datos en un CSV se muestra en la figura [34]

```
if save_csv:
    cv2.imwrite(img_path, img_r)
    file_name = "{}/{}.csv".format(
        trk.g_vars["root_path"],
        trk.g_vars["file_name"]
    )
    with open(file_name, "a+") as f:
        f.write(line)

return (line, img_r, img_path)
```

Figura 34: Código para guardar CSV.

4 RESULTADOS

Los resultados que se obtuvieron fueron los esperados, pues se cumplieron los objetivos establecidos en el comienzo del proyecto que fue la detección, clasificación y conteo de los autos. Gracias a las técnicas de machine learning y el filtro de Kalman se pudo obtener todos los requisitos que se tenían previstos con una muy buena veracidad de datos, teniendo como resultado un sistema casi exacto.

Para iniciar la ejecución del sistema se realiza a través de la terminal de la PC con el siguiente comando:

python3 detect_video.py --weights model_data/yolov3-416 --model yolov3 --video resources/E2.MOV --x_lim 500 --x_lim 1200 –count

En la figura [35] se muestra la acción de ejecución del sistema.

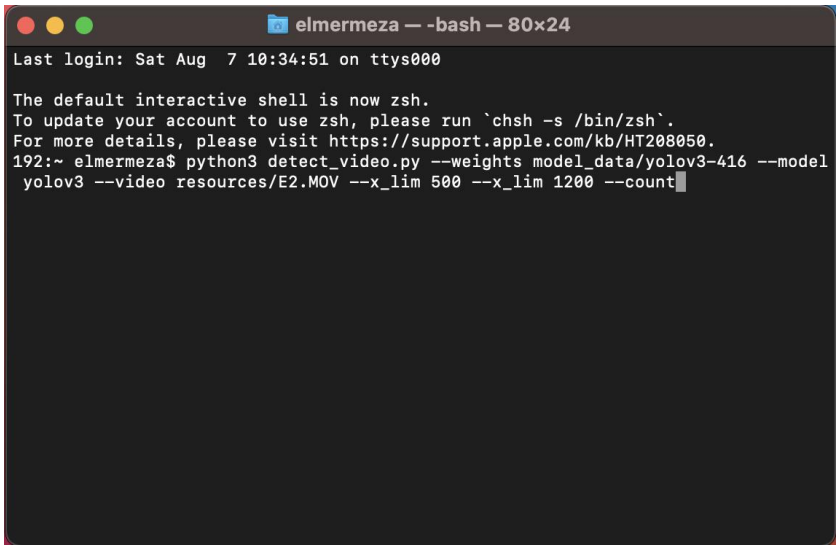

Figura 35: Ejecución Por Terminal

Una vez iniciado el proceso los resultados son los siguientes:

En las figuras [36 y 37] se muestra la detección de un vehículo desde un punto A (que es el inicio de entrada del vehículo) hasta el punto B (el final de la detección del vehículo).

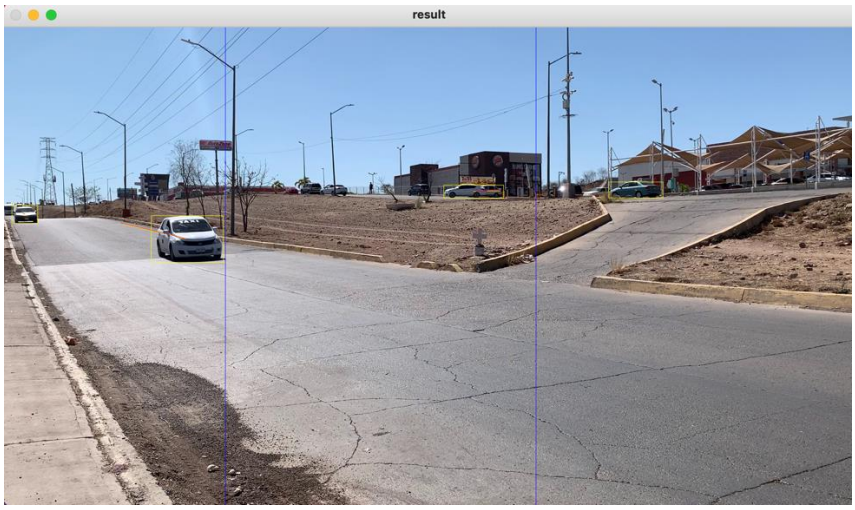

Figura 36: Detección De Auto Punto A

Como se puede observar en la imagen el sistema capta cuando un vehículo se aproxima a los márgenes establecidos.

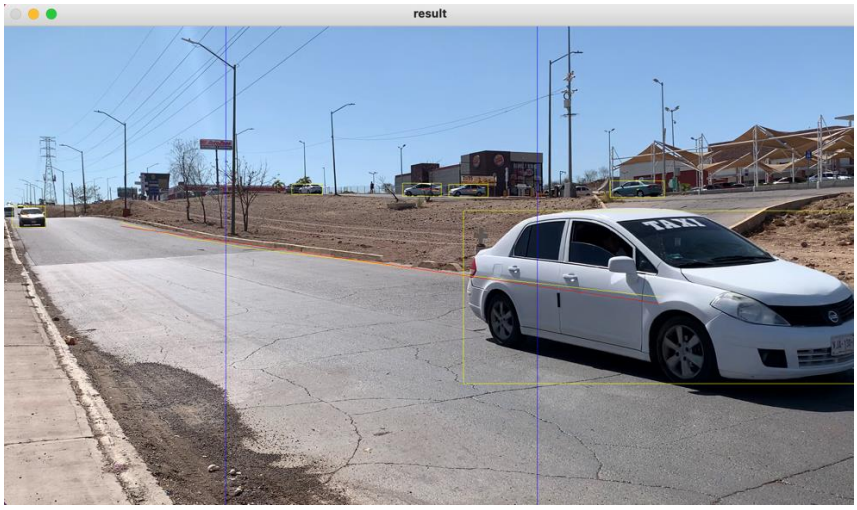

Figura 37: Detección De Auto Punto B

En la toma de la figura 38, el sistema termina de determinar la detección total del vehículo pues este inicia en un punto A y al llegar a un punto B, que es de extremo a extremo, valida que un vehículo ha pasado.

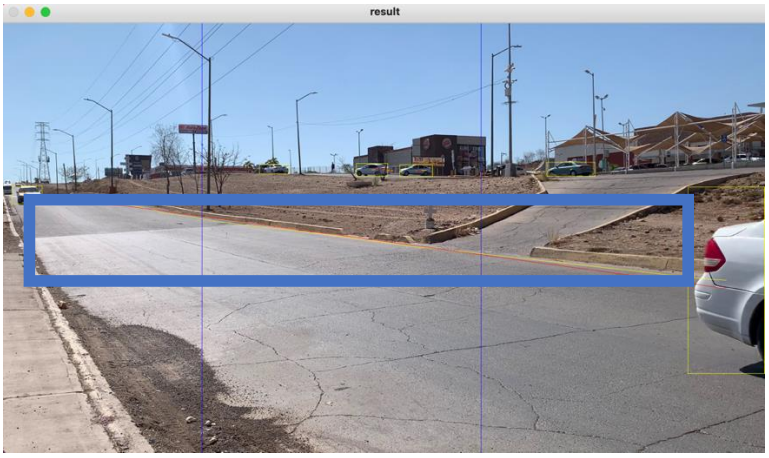

Figura 38: Trayectoria

Al haber pasado el vehículo ya por el punto A y B podemos observar en la ilustración [40] en las líneas roja y amarilla la trayectoria que se marca del vehículo.

De misma forma cuando se realizó este proceso se obtuvo la clasificación del vehículo, así como el total de cada clase de vehículo que paso.

Después de terminar todo el proceso en cada video como resultado el sistema arroja una lista con todas las clases posibles y un contador por cada clase, es decir te indica cuántos vehículos de cada clase fueron detectados.

En la figura [39] se pueden observar todas las clases que detecta el sistema, y en la figura [40] muestra el contador de cada clase.

```
clases ['person\n', 'bicycle\n', 'car\n', 'motorbike\n', 'aeroplane\n', 'bus\n', 'train\n', 'truck\n', 'boat\n', 'traffic-light\n', 'fire-hydrant\n', 'stop-sign\n', 'parking-meter\n', 'bench\n', 'bird\n',
 'cat\n', 'dog\n', 'horse\n', 'sheep\n', 'cow\n', 'elephant\n', 'bear\n', 'zebra\n', 'giraffe\n', 'backpack\n', 'umbrella\n', 'handbag\n', 'tie\n', 'suitcase\n', 'frisbee\n', 'skis\n', 'snowboard\n', 'spo
rts-ball\n', 'kite\n', 'baseball-bat\n', 'baseball-glove\n', 'skateboard\n', 'surfboard\n', 'tennis-racket\n', 'bottle\n', 'wine-glass\n', 'cup\n', 'fork\n', 'knife\n', 'spoon\n', 'bowl\n', 'banana\n', 'a
pple\n', 'sandwich\n', 'orange\n', 'broccoli\n', 'carrot\n', 'hot-dog\n', 'pizza\n', 'donut\n', 'cake\n', 'chair\n', 'sofa\n', 'pottedplant\n', 'bed\n', 'diningtable\n', 'toilet\n', 'tvmonitor\n', 'laptop
\n', 'mouse\n', 'remote\n', 'keyboard\n', 'cell-phone\n', 'microwave\n', 'oven\n', 'toaster\n', 'sink\n', 'refrigerator\n', 'book\n', 'clock\n', 'vase\n', 'scissors\n', 'teddy-bear\n', 'hair-drier\n', 'to
othbrush\n']
```

Figura 39: Todas las clases que se pueden detectar.

```
contador [2, 1, 40, 0, 0, 1, 0, 1, 0, 0, 0
```

5 CONCLUSIONES

Al terminar el desarrollo de actual proyecto se pudo comprender la gran cantidad de aportaciones que existen sobre este tema, facilitando un poco el continuo desarrollo para mejoras del sistema, pero teniendo su grado de dificultad ya que son temas que requieren de mucha comprensión y concentración para poder entender los conceptos para el desarrollo de nuevas tareas a futuro.

Es muy importante tener conocimientos suficientes de los sistemas que se van a utilizar, pues se necesita de un buen manejo de ellos, ya que aplican áreas de lógica, matemáticas, desarrollo de código, entre otras, se menciona que es importante porque ayudará a realizar las actividades requeridas de una manera más eficiente.

En lo que transcurre el tiempo de desarrollo del proyecto se familiariza con los temas lo que lo vuelve más interesante y por ello querer explotar más sobre la tecnología que está aplicando, por lo que se vuelve a un entorno más factible de realizar.

La cantidad de problemas de tecnología son en realidad muchísimos, de los cuales varios siguen en desarrollo por la razón de que se está tratando de lograr que los sistemas computarizados lleguen a comprender lo más aproximado a lo que es el pensamiento humano, es decir que sean capaces de dar soluciones más rápidas de distintos problemas que actualmente existen, como el problema de este proyecto, se logró que la computadora lograra captar lo que el ojo humano ve y dándole lógica para realizar diferentes tareas.

Actualmente se han logrado avances muy importantes en el área de la tecnología, pero se quedan cortos, pues existe una gran cantidad de problemas por resolver aun, pero que la comunidad tecnológica poco a poco ira solucionando.

Al finalizar la etapa del actual proyecto, se logro de una manera satisfactoria la clasificación de los vehículos del cual se pueden obtener los datos relevantes que son su tipo de clase, ejemplo, arrojo datos de que pasaron 20 carros, 3 motocicletas, 1 persona, etc. De esta forma es fácil dar un resultado final de las cantidades de vehículos que pasaron en algún transcurso de hora y lugar decidido para realizar las pruebas.

6 COMPETENCIAS DESARROLLADAS

Al finalizar la elaboración de este proyecto me quedó un gran aprendizaje, además que desarrollé competencias de un ámbito profesional aprovechando al máximo toda la información que me brindaban.

Las competencias desarrolladas son:

- **Autoconfianza**, es una de las competencias que más me quedo marcada porque me hizo creer en mí mismo, que todo lo que se propone lo puedes realizar con mucho esfuerzo y dedicación a lo que haces.
- El **compromiso individual** que fue una clave muy importante porque primero más que nada debí comprometerme yo mismo dedicarle el cien por ciento a este proyecto.
- La **organización** fue fundamental ya que me ayudo a separar los tiempos correctos en lo que tenía que realizar las actividades.
- La **tolerancia a estrés** en tiempos críticos me ayudo en mantener la calma en todo momento con actividades complejas y me ayudo a poder razonar y pensar soluciones más rápidas.
- La **habilidad** de aprender nuevas técnicas de trabajo, así como también de comprender temas diferentes y aplicarlos en un mismo objetivo.
- La **innovación** a un proyecto ya planificado, aplicando mejoras en cuanto a su funcionalidad y haciéndolo un sistema más completo.

7 REFERENCIAS

Bibliografía

Aula21. (2020). *Aula21*. Obtenido de Aula21: https://www.cursosaula21.com/que-es-la-vision-artificial/

Aula21. (2020). *cursoAula21*. Obtenido de Aula21: https://www.cursosaula21.com/que-es-python/

Becker, A. (2018). *Kalmanfilter*. Obtenido de Mirada General: https://www.kalmanfilter.net/ES/default_es.aspx

Contaval. (2016). *Contaval*. Obtenido de Contaval: https://www.contaval.es/que-es-la-vision-artificial-y-para-que-sirve/

GRACIA, L. (9 de 10 de 2013). *UN POCO DE JAVA*. Obtenido de https://unpocodejava.com/2013/10/09/que-es-opencv/

GRACIA, L. (9 de 10 de 2013). *UN POCO DE JAVA* . Obtenido de https://unpocodejava.com/2013/10/09/que-es-opencv/

IBM. (2020). *ibm*. Obtenido de IBM: https://www.ibm.com/mx-es/analytics/machine-learning

Oracle. (2021). *Oracle*. Obtenido de Oracle: https://www.oracle.com/mx/artificial-intelligence/what-is-ai/

S.A. (2015). *Tensorflow*. Obtenido de Tensorflow: https://www.tensorflow.org/?hl=es-419

S.A. (2020). *aprendemachinelearning*. Obtenido de Aprende Machine Learning: https://www.aprendemachinelearning.com/deteccion-de-objetos-con-python-yolo-keras-tutorial/

S.A. (2021). *ESIC*. Obtenido de ESIC: https://www.esic.edu/rethink/tecnologia/para-que-sirve-python