**Sincronización de tráfico de vehículos**

**por medio de inteligencia artificial**

**Almirón, Yoel Maximiliano**

**UTN - Facultad Regional La Plata,**

**Buenos Aires, Argentina**

**yalmiron@alu.frlp.utn.edu.ar**

**Abstract**

*Este trabajo presenta un desarrollo para la sincronización entre los vehículos en una zona urbana mediante una técnica de cascada, buscando facilitar el tránsito entre las zonas urbanas de una ciudad. Con la coordinación de los semáforos actuales, se puede percibir la utilidad de la “onda verde”, aunque en situaciones no es del todo justa.*

*En la solución planteada proponemos una demostración que, mediante una técnica de inteligencia artificial, se puede lograr una mejor sincronización, más justa y más segura, evitando la acumulación de tiempo innecesario en distintas zonas. (Abstract)*

**Sincronización, vehicular, Haar Cascade**

**Introducción**

Existen diversas ciudades, pueblos en el mundo que buscan un paradigma de “Ciudad Inteligente”, este término plantea optimizar objetos cotidianos o adjuntar servicios más eficientes y funcionales. En este desarrollo se aspira a la utilización de un algoritmo para hacer uso de los recursos hardware para simplificar la sincronización vehicular, implementando el concepto de seguimiento de imágenes con la técnica de inteligencia artificial más conocida como Haar Cascade.

El tránsito vehicular de los pueblos, ciudades, son un factor creciente que genera desconcierto, accidentes y retrasos temporales en distintas situaciones, sectores de distintas ciudades. Existen muchas personas que suelen tener disconformidades por estar expuestos a sufrir este tipo de desorden a diario. Al ser perjudicados gran cantidad de personas por variantes que no son controladas. Para evitar esto, se optó por implementar el concepto de Onda verde que permite facilitar el tránsito, sincronización de los semáforos. Pero esto no contempla todas las posibles eventualidades, una suele ser que estén pocos vehículos esperando en una avenida y sin ningún peatón o vehículo cercano a las calles que interceptan a la misma. Esto en algunas zonas podría ser peligroso, además de no ser equitativo para el resto de vehículos.

**Tecnica Haar Cascade**

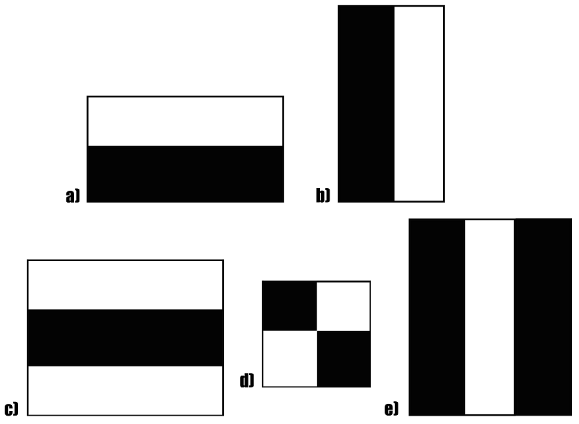
Para mejorar la detección de vehículos, se plantea una solución tendiente a perfeccionar su sincronización. Para esto, se propone el uso de la técnica cascada que se basa en la concatenación de varios clasificadores débiles, cada uno analizando una porción diferente de una imagen o frame en el caso de un video.

Esta técnica hace el uso de características haar que atraviesa continuamente desde la parte superior izquierda de una imagen hasta la parte inferior derecha para buscar la característica particular.

La función haar atravesara pixel por pixel la imagen. Esté algoritmo a implementar se puede explicar en cuatro etapas:

* Cálculo de características de Haar.
* Creación de imágenes integrales.
* Adaboost.
* Implementación de clasificadores.

El primer paso es recopilar las características de Haar. La misma consiste esencialmente en cálculos que se realizan en regiones rectangulares adyacentes en una ubicación específica en una ventana de detección. El cálculo implica sumar las intensidades existentes de píxeles en cada región y calcular las diferencias entre las sumas.

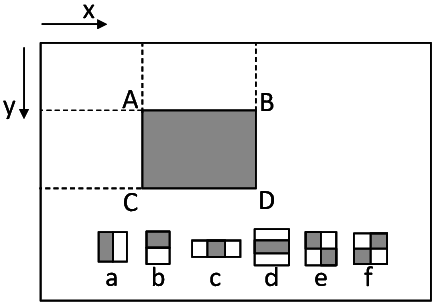


Luego realizaremos la creación de imágenes integrales. Es conocida como tabla de área sumada. El algoritmo para calcular de forma rápida y eficiente realiza una suma de los valores en un subconjunto de rectángulos de una cuadrícula. Este concepto fue introducido por Viola y Jones, la imagen integral se construye de la siguiente manera:



donde la función ii(x,y) es la imagen integral en la ubicación de los píxeles (x,y) y (x’.y’) es la imagen original. Usando la imagen integral calculamos la suma de cualquier área rectangular,es extremadamente eficiente.

Ilustración de cómo funciona una imagen integral.



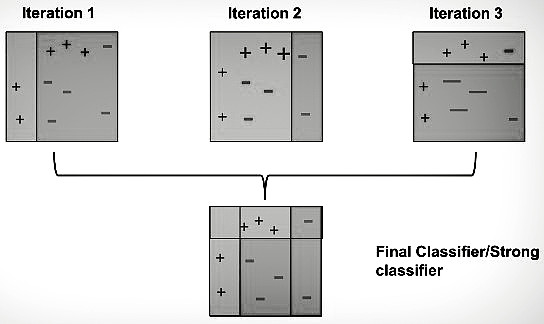
La suma de pixeles en una región de un rectangulo es posible calcular de la siguiente manera:



Las características se definen como la diferencia de intensidad (ponderada) entre dos y cuatro rectángulos. El objetivo es averiguar la suma de todos los píxeles de la imagen que se encuentran en el área más oscura de la función haar y la suma de todos los píxeles de la imagen que se encuentran en el área más clara de dicha función.

El entrenamiento Adaboost, esencialmente elige las mejores funciones y entrena a los clasificadores para usarlas. Utiliza una combinación de “clasificadores débiles” para crear un “clasificador fuerte” que el algoritmo puede usar para detectar objetos.

Para esto,se crean moviendo una ventana sobre la imagen de entrada y calculando las características de Haar para cada subsección de la imagen. Esta diferencia se compara con un umbral aprendido que separa los no objetos de los objetos. Debido a que estos son "clasificadores débiles", se necesita una gran cantidad de características de Haar para que la precisión forme un clasificador fuerte.

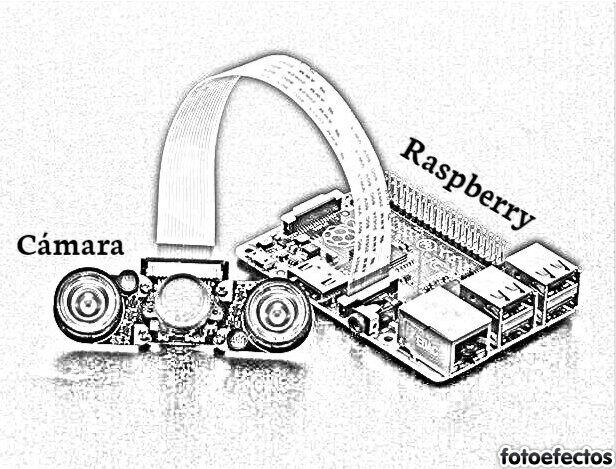


Finalmente en el último paso combina estos clasificadores débiles en fuertes utilizando la clasificación en cascada.

Por último, implementamos la cascada. Esto se hace por medio de OpenCV que nos proporciona un método de entrenamiento o puede ser incluso modelos previamente entrenados, que se pueden leer mediante el método cascadeclassifier::load. El modelo a utilizar está basado en el entrenamiento que realizó Andrew Sobral.

**Desarrollo**

Para mejorar la sincronización, se propone el uso de un elemento de hardware (Raspberry Pi), está misma no deja de ser un computador portátil de forma reducida, más una cámara con visión nocturna.

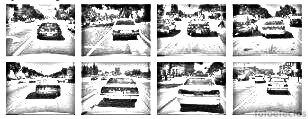


La Raspberry Pi 4B utilizada para el desarrollo fue conectada con la cámara Catda C1130, dicho componente consta de una nitidez de 5 megapíxeles , además de un sensor de visión nocturna. Esté sensor permite detectar objetos en distintos momentos del día y en diferentes condiciones.

Se instaló la interfaz de Raspberry Pi OS 32 bit, está misma cuenta con muchas herramientas,entre ellas nos provee la posibilidad de tener Python + IDE para realizar la codificación.

Luego a través de un algoritmo con la técnica de Haar cascade podríamos generar un conteo de vehículos para luego aplicar un algoritmo de prioridad y ser más equitativo para el resto de vehículo.

En este desarrollo se optó por utilizar un Haar cascade construido por Andrew Sobral, el cual está compuesto de un entrenamiento con diferentes vehículos en distintas situaciones.



Para la realización de la codificación, se eligió el lenguaje de programación Python 2.3, el tipo abstracto de datos utilizado fue OpenCv que cuenta con un motor de inferencia Intel Back-end (OpenVino), esto nos permite el manejo de imágenes en tiempo real. Además de brindarnos operaciones para ajustar la detección de video.

Esto se desarrolló con el fin de evitar la ausencia de la onda verde en las avenidas, se podría aplicar dicho desarrollo mencionado.

Vale aclarar que se utilizó la librería cv2 de openCV para poder importar datos en tiempo real sea una cámara o un video.

import cv2

face\_cascade = cv2.CascadeClassifier('Cascada a importar.xml')

//cap = cv2.VideoCapture(1)

cap = cv2.VideoCapture('video1.avi')

while true:

imag= cap.read()

gray = cv2.cvtColor(imag, cv2.COLOR\_BGR2GRAY)

autos = face\_cascade.detectMultiScale(gray, 1.1, 4)

for (x, y, w, h) in faces:

cv2.rectangle(imag, (x, y),(x+w, y+h), (255,0,0))

cv2.inshow('imag',imag)

k = cv2.waitkey(30)

if k==27:

break

cap.release()

Al ejecutar el código obtuvimos una ventana con una respuesta, reconocimiento de vehículos en tiempo real.



ventana de simulación.

**Conclusión**

En este trabajo se ha desarrollado una posible solución para la sincronización de vehículos, utilizando un algoritmo de prioridad con el método de cascada. Se propone así una posible mejora a aplicar dicho conceptos en las ciudades actuales donde el tránsito es abundante y los accidentes son habituales. También existe provecho en reducir la emanación de gases contaminantes para el ambiente, esto dado a que se pudo comprobar que el parque automotor incluye un numeroso y las emisiones procedentes de los escapes de estos vehículos contienen monóxido de carbono, hidrocarburos y óxidos de nitrógeno que son liberadores en gran cantidad y derivan a la atmósfera.

Por último, es importante destacar que el prototipo de dicho algoritmo funcionó de manera correcta. Se puede inferir que , al aplicar el modelo a una escala mayor, se debería aplicar otro algoritmo para ser más justo hacia los peatones, debido que el algoritmo implementado no lo tiene en cuenta.

Para la continuación de este proyecto se seguirá profundizando sobre el sistema implementado, también se seguirá investigando situaciones más puntuales y justas de tal manera de incorporar eventos con peatones.

**Referencias**

[1]. Face Detection with Haar Cascade

([Haar cascade](https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08) )

[2].OpenCV ([OpenCV Library](https://opencv.org/releases/))

[3].Classification Haar Cascade ([classification](https://scikit-image.org/docs/dev/auto_examples/applications/plot_haar_extraction_selection_classification.html))

[4].Mastering OpenCV 4 with Python ([Mastering\_OpenCV\_4\_with\_Python)](https://openlibrary.org/works/OL25243321W/Mastering_OpenCV_4_with_Python)

[5]. David H. Eberly ([David Documentation](https://www.geometrictools.com/Documentation/Documentation.html))

[6].Detection with Haar Cascade ([Haar Detection](https://towardsdatascience.com/face-detection-with-haar-cascade-part-ii-50120fd1700))