CENTRO DE ENSEÑANZA TÉCNICA Y SUPERIOR



Campus Tijuana

Escuela de Ingeniería

Propuesta de Proyecto Final

Avance de Proyecto

Presentan:

25990 - Sevilla Gómez Llanos Hugo

27602 - López Gaxiola Luis Jaime

Equipo: Pipilas

Profesora: Kenia Picos

Tijuana, B.C., 27 de mayo de 2020

**Índice**

[Titulo](#_ttxg48ywlswn) **3**

[Resumen](#_yerwdtvg9zog) **3**

[Descripción del problema](#_ea8qpebhhsv4) **3**

[Objetivos](#_25yd8o33zi0p) **4**

[Marco Teórico](#_s2mb8g23xigr) **5**

[Descripción de Implementación](#_r20vnkllojx) **8**

[Conclusiones y Avance futuro](#_xwjz6rlxs1sz) **9**

[Referencias Bibliográficas](#_qi0rxb56aym1) **10**

# 

# **Titulo**



# **Resumen**

Hacemos borrosa la imagen de automóviles para difuminar el fondo. Encontramos el contorno de la placa del carro (en gris y con canny o threshold), recortar la imagen hacia la placa, aplicamos erosiones y dilataciones para tener los dígitos más claros y después hacemos mejores contrastes, buscar los contornos de los dígitos y les dibujar rectángulos alrededor de cada número, los identificamos e imprimimos en pantalla todo.

# **Descripción del problema**

El mundo se está digitalizando, todos los textos físicos se están pasando a digital, por ejemplo; cheques de pago se comienzan a cobrar por medio de aplicaciones móviles que analizan sus cifras por medio de fotografías. Muchas cosas manuales se han automatizado gracias a las computadoras, por medio de algoritmos de aprendizaje que procesan imágenes se podrán visualizar predicciones sobre lo que se analiza si es correcto o no. Estas tecnologías están cerca de llegar a su apogeo. Nosotros enfocaremos esta visión por computadora hacía fotografías de automóviles para segmentar la imagen y recortar la placa del carro para después identificar el contenido de la misma, transcribiendo a texto su lenguaje alfanumérico.

# **Objetivos**

* Imagen Original suavizada por algún filtro visto en clase para facilitar la detección de bordes.
* Detección de los bordes del carro para cada vez acercarse al objetivo de recortar la placa.
* Disminución de ruido innecesario en la imagen (bordes que no se necesitan).
* Detección de la placa del automóvil
* Recortar una imagen de un automóvil hacia su placa por procesos de contornos
* Detección de bordes de los caracteres de la placa para su identificación.
* Dibujar los bordes del carácter en placa para su identificación.
* Identificación los dígitos de la placa
* Concatenación de la imagen original con el resultado final para ver todo lo que se realizó y cuál fue el resultado obtenido.

**Marco Teórico**

**Reconocimiento de Texto:**

El reconocimiento de texto es una técnica analítica que consigue extraer y analizar la información que se encuentra en documentos de formato imagen, como por ejemplo JPG o PNG. Se puede aplicar a cualquier imagen que contenga texto, independientemente del idioma en el que esté el mismo.

**Project Jupyter:**

El Proyecto Jupyter es una organización sin ánimo de lucro creada para "desarrollar software de código abierto, estándares abiertos y servicios para computación interactiva en docenas de lenguajes de programación". Creado a partir de IPython en 2014 por Fernando Pérez, el proyecto Jupyter soporta entornos de ejecución en varias docenas de lenguajes de programación. El nombre del proyecto Jupyter es una referencia a los tres lenguajes de programación principales soportados por Jupyter, que son Julia, Python y R, y también un homenaje a los cuadernos de Galileo que registran el descubrimiento de los satélites de Júpiter. El proyecto Jupyter ha desarrollado y respaldado los productos de computación interactiva Jupyter Notebook, JupyterHub y JupyterLab, la versión de próxima generación de Jupyter Notebook.

**Python:**

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

**OpenCV:**

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. OpenCV significa Open Computer Vision (Visión Artificial Abierta). Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en una gran cantidad de aplicaciones, y hasta 2020 se la sigue mencionando como la biblioteca más popular de visión artificial. Detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, son sólo algunos ejemplos de aplicaciones de OpenCV.

**Matplotlib:**

Matplotlib es una librería de trazado para el lenguaje de programación Python y su extensión matemática numérica NumPy. Proporciona una API orientada a objetos para incrustar los gráficos en las aplicaciones que utilizan conjuntos de herramientas de GUI de propósito general como Tkinter, wxPython, Qt o GTK+. También existe una interfaz de procedimiento "pylab" basada en una máquina de estados (como OpenGL), diseñada para parecerse mucho a la de MATLAB, aunque se desaconseja su uso. SciPy hace uso de Matplotlib. Matplotlib fue escrito originalmente por John D. Hunter, desde entonces tiene una activa comunidad de desarrollo, y se distribuye bajo una licencia tipo BSD. Michael Droettboom fue nominado como desarrollador principal de Matplotlib poco antes de la muerte de John Hunter en agosto de 2012, y se le unió Thomas Caswell. Matplotlib 2.0.x soporta las versiones 2.7 a 3.6 de Python. El soporte para Python 3 comenzó con Matplotlib 1.2. Matplotlib 1.4 es la última versión que soporta Python 2.6. Matplotlib se ha comprometido a no apoyar a Python 2 más allá de 2020 al firmar la Declaración de Python 3.

**PyTesseract:**

Es una herramienta de reconocimiento óptico de caracteres (OCR) para python. Es decir, reconocerá y "leerá" el texto incrustado en las imágenes. Python-tesseract es un envoltorio para el motor Tesseract-OCR de Google. También es útil como un guión de invocación independiente para tesseract, ya que puede leer todos los tipos de imágenes soportados por las librerías de imágenes Pillow y Leptonica, incluyendo jpeg, png, gif, bmp, tiff, y otras. Además, si se usa como guión, Python-tesseract imprimirá el texto reconocido en lugar de escribirlo en un archivo.

**Math Module:**

Estas funciones realizan varias operaciones aritméticas como calcular el valor superior, inferior o absoluto de un número usando las funciones floor(x), ceil(x), y fabs(x) respectivamente. La función ceil(x) devolverá el entero más pequeño que sea mayor que o igual a x. De manera similar, floor(x) devuelve el entero más grande menor que o igual a x. La función fabs(x) devuelve el valor absoluto de x. También puedes realizar operaciones no triviales cómo calcular el factorial de un número usando factorial(x). Un factorial es el producto de un entero y todos los enteros positivos menores que este. Es usado extensivamente cuando se trabaja con combinaciones y permutaciones. También puede ser usado para calcular el valor de las funciones seno y coseno.

**Pillow:**

Python Imaging Library (abreviada como PIL) (en versiones más recientes conocida como Pillow) es una biblioteca adicional gratuita y de código abierto para el lenguaje de programación Python que añade soporte para abrir, manipular y guardar muchos formatos diferentes de archivos de imagen. Está disponible para Windows, Mac OS X y Linux. La última versión de PIL es 1.1.7, fue publicada en septiembre de 2009 y soporta Python 1.5.2-2.7, con soporte para Python 3 que será publicado "más adelante". El desarrollo parece estar interrumpido, con el último compromiso con el repositorio PIL en 2011. En consecuencia, un proyecto sucesor llamado Pillow ha bifurcado el repositorio PIL y ha añadido soporte para Python 3.x. Esta bifurcación ha sido adoptada como un reemplazo del PIL original en las distribuciones de Linux, incluyendo Debian y Ubuntu (desde el 13.04).

# **Descripción de Implementación**

Nota: hacer copias de imágenes para las funciones de contornos pues son destructivas de la imagen original para visualizar y cambiar los parámetros para llegar a mejores resultados sin tener que volver a correr todo, generar copias de las imágenes.

* Hacer borrosa la imagen del automóvil para difuminar el fondo con blur de opencv.
* Encontrar el contorno de la placa del carro con findcontour de opencv, usando canny.
* Recortar la imagen hacia la placa tomando el contorno de mayor área dados.
* Aplicar erosiones, y dilataciones para tener los dígitos más claros igual de opencv.
* Hacer mejores contrastes con gammas.
* Buscar los contornos de los números con threshold.
* Dibujar rectángulos alrededor de cada número con drawcontours de OpenCV.
* Escribir sobre pantalla los caracteres de la placa con pytesseract.

**Conclusiones y Avances Futuros**

La aplicación de los diferentes filtros vistos en clase fue una base necesaria para la detección de los bordes y por ende por detectar los dígitos, la investigación de las librerías utilizadas fue algo fundamental y algo que ayudó mucho fue la vasta documentación que se encuentra sobre el tema.

OpenCV fue la librería más importante de todo el proyecto porque tiene todas las funciones importantes que se necesitan para buscar todos los contornos, encontrarlos además de sus áreas, cabe recalcar que primera vez que encontramos una función que genera una lista de listas, habíamos escuchado al respecto pero no lo habíamos utilizado como antes.

Finalmente vimos que para la visión por computadora no son necesarias las redes neuronales pues hay una gran diversidad de herramientas y metodologías diferentes para poder llegar al mismo objetivo y con el mismo resultado aunque no sea tan robusto ni tan exacto como una buena red neuronal bien entrenada, aunque puede ser mejor en casos muy específicos.

* Implementar una interfaz gráfica
* Hacer parámetros más robustos o condicionales con diferentes parámetros
* Aplicarlo a una cámara en tiempo real

# 

# **Referencias Bibliográficas**

Abadi, Martin, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … others. (2016). Tensorflow: A system for large-scale machine learning. In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16) (pp. 265–283).

Chollet, F. (2015) keras, GitHub. https://github.com/fchollet/keras

LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database.

Matich, D. J. (2001). Redes Neuronales: Conceptos básicos y aplicaciones. Universidad Tecnológica Nacional, México.

OpenCV Team, 2020. *Opencv*. [online] Opencv.org. Available at: <https://opencv.org/> .

Python Team, 2020. *Pytesseract*. [online] PyPI. Available at: <<https://pypi.org/project/pytesseract/>>.

Python Team, 2020. *Welcome To Tesseract’S Documentation! — Tesseract 0.1.1 Documentation*. [online] Pytesseract.readthedocs.io. Available at: <https://pytesseract.readthedocs.io/en/latest/> .

Hunter, J., 2020. *Overview — Matplotlib 3.1.2 Documentation*. [online] Matplotlib.org. Available at: <https://matplotlib.org/3.1.1/contents.html> .