Manual Técnico

Descripción general del sistema de información desarrollado

El Programa de Detección es un software basado en el lenguaje Python que utiliza las instrucciones ejecutables de YOLOv5. Este programa ha sido desarrollado para monitorear y analizar el tráfico vehicular y peatonal, empleando pesos específicos obtenidos a través del entrenamiento de una red neuronal diseñada para esta tarea. Además, se ha implementado una interfaz gráfica intuitiva que facilita su uso, permitiendo realizar inferencias tanto en imágenes estáticas como en videos en streaming.

Requerimientos técnicos a nivel de hardware y software para instalar y operar el software desarrollado.

A nivel de Software Yolov5 necesita Python>=3.7.0, la ejecución de Yolov5 necesita diferentes librerías para su correcto funcionamiento tales como: gitpython>=3.1.30, matplotlib>=3.3, numpy>=1.18.5, opencv-python>=4.1.1, Pillow>=7.1.2, psutil, PyYAML>=5.3.1, requests>=2.23.0, scipy>=1.4.1, thop>=0.1.1, torch>=1.7.0, torchvision>=0.8.1, tqdm>=4.64.0, ultralytics>=8.0.100, pandas>=1.1.4, seaborn>=0.11.0, setuptools>=65.5.1

Si hablamos de Hardware Yolov5 puede ser ejecutado en CPUs, pero la inferencia se realiza en un tiempo grande, por lo que es preferible realizarlo con GPU de tipo NVIDIA

Entorno y/o Lenguaje de desarrollo utilizado

El software se ha desarrollado utilizando Python como lenguaje principal. Este programa de detección se enfoca en el análisis de imágenes que ofrece un conjunto de funcionalidades para la detección y clasificación de estos artefactos. La implementación se realiza mediante algoritmos y técnicas de procesamiento de imágenes, así como métodos de aprendizaje automático. El programa también cuenta con una interfaz gráfica de usuario (GUI) que facilita la interacción con los usuarios y permite la visualización de los resultados obtenidos.

Descripción de los diferentes programas (módulos) que hacen parte del sistema de información y su interrelación (teniendo en cuenta que un programa de computador puede estar compuesto de uno o varios módulos que cumplen funciones específicas)

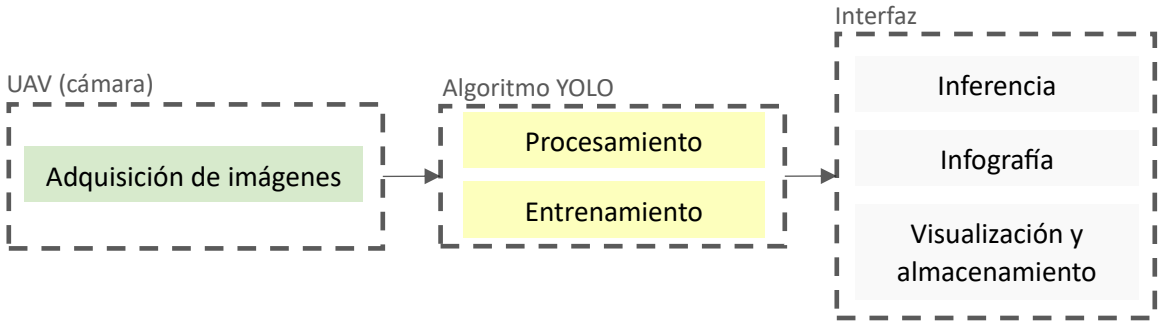

Figura 1. Propuesta de herramienta para detección de vehículos y peatones basados en algoritmo YOLO

La Figura 1 ilustra los distintos módulos utilizados en el sistema de detección. En primer lugar, las unidades aéreas no tripuladas capturan imágenes y videos, los cuales se utilizan para crear el conjunto de datos necesario para el testeo del algoritmo. A continuación, el módulo de entrenamiento utiliza estos datos para obtener los pesos que se emplearán en la inferencia realizada por el sistema.

La interfaz gráfica utiliza los pesos previamente obtenidos para llevar a cabo la detección de objetos en nuevos videos e imágenes, esta misma genera una infografía que muestra el porcentaje de vehículos y peatones detectados.

Con esta configuración modular, el sistema de detección logra un flujo eficiente y efectivo, permitiendo la adquisición de datos, el entrenamiento del algoritmo, la inferencia en tiempo real y la presentación visual de los resultados obtenidos.

Motores de bases de datos sobre los que fue desarrollado el sistema (en caso de que este basado en archivos planos se debe realizar una descripción de la forma como operan).

Para el desarrollo y prueba de este sistema, se utilizó la base de datos VisDrone2019, la cual es de acceso público y fue recolectada por el equipo AISKYEYE del Laboratorio de Aprendizaje Automático y Minería de Datos en la Universidad de Tianjin, China.

Esta base de datos consta de 288 videos, que representan 261,908 cuadros (frames), y 10,208 imágenes, las cuales fueron tomadas con diversas cámaras montadas en drones en diferentes escenarios urbanos y rurales en 14 ciudades de China. Estas imágenes capturan diferentes objetos como automóviles, peatones, triciclos, furgonetas, autobuses, bicicletas, entre otros.

La base de datos se dividió en 4 subconjuntos según la tarea a realizar: Detección de Objetos en Imágenes, Detección de Objetos en Videos, Seguimiento de Objetos Únicos y Seguimiento de Múltiples Objetos. Para este sistema, se utilizó únicamente el conjunto de datos de Detección de Objetos, el cual cuenta con 24,198 archivos para entrenamiento, 2,846 para validación y 6,635 para pruebas.

Diccionario de datos (descripción de las tablas, campos, tipos y tamaño de campos que la conforman, identificando llaves primarias y secundarias si es del caso)

Para la validación del modelo propuesto para el sistema, se utilizó la técnica de validación cruzada, que implica la separación de los datos en conjuntos destinados al entrenamiento y a la validación. Posteriormente, el modelo con mejor rendimiento se prueba con nuevos datos de prueba para verificar su desempeño.

Para entrenar y validar el algoritmo YOLO, es necesario construir la siguiente estructura de datos:

```
├── VisDrone2019
│   ├── test
│   │   ├── images
│   │   └── labels
│   ├── train
│   │   ├── images
│   │   └── labels
│   ├── valid
│   │   ├── images
│   │   └── labels
```

Figura 2. Estructura de datos requeridas

Donde cada instancia es una matriz $X \in [0,255]^{HxWxC}$ y una matriz de etiquetas $Y \in R^{Nx5}$ donde HxW es la resolución de la imagen, C son los canales de la imagen (en este caso, el espacio de color RGB) y N es el número de objetos detectados en la imagen.

Principales casos de uso

La herramienta se puede para utilizar para 1) Realizar detección en imágenes, 2) Realizar detección en videos locales 3) Realizar detección sobre videos en streaming, 4) Obtener infografía de cada una de las detecciones

Figura 3. Ventana para la detección sobre imágenes

Figura 4. Ventana para la detección sobre videos locales

Figura 5. Ventana para la detección sobre videos en streaming

Cada caso de uso es capaz de mostrar una infografía la cual proporciona información relevante para el usuario

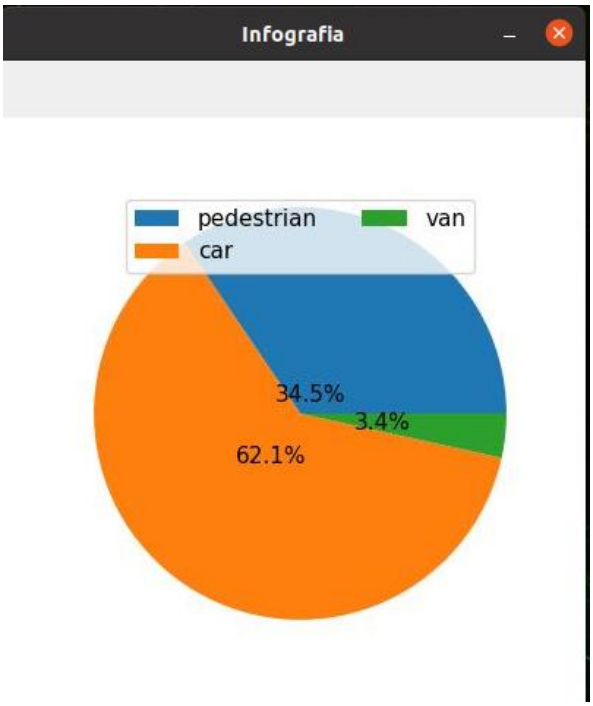

Figura 6. Infografía realizada en imágenes

Se deben anexar los programas fuente como parte de los entregables.

Todo lo relacionado con la detección e interfaz grafica puede ser encontrado en el siguiente repositorio: https://github.com/liturriago/Programa-Deteccion-OOPART-GCPDS.git