Servicio Social, Trimestre: 25-I, 2025

Construcción de una base de datos de videos aéreos y su análisis vía herramientas de IA

Martínez Buenrostro Jorge Rafael.

Universidad Autónoma Metropolitana
Unidad Iztapalapa, México
*molap*96@*gmail.com*

Resumen: Este documento describe el proyecto cuya meta es contar con una caracterización de los grupos de humanos que se desplazan juntos. Identificando las características estadísticas de los mismos.

1. Introducción

1.1. Descripción general del proyecto

La simulación de una red de comunicaciones en donde intervienen dispositivos personales de comunucación requiere contar con modelos que representen fielmente los patrones de movimiento de las personas. De otra manera, la utilidad de las conclusiones que se puedan obtener de esa simulación es limitada.
Para avanzar hacia la definición de un modelo de movilidad humana grupal, se propone la construcción de una base de datos de videos aéreos (capturados por un dron) y su análisis vía herramientas de IA. Esto nos permitirá determinar algunas características de la movilidad de interés.

1.2. Objetivos y propósitos

El objetivo principal del proyecto es contar con una caracterización de los grupos de humanos que se desplazan juntos. Identificando las características estadísticas de los grupos de humanos que se desplazan juntos.
Los propósitos del proyecto son:

- Construir una base de datos de videos aéreos de grupos humanos.
- Usar un modelo de IA que permita identificar y caracterizar los grupos humanos en los videos.
- Analizar los patrones de movimiento y las interacciones entre los grupos humanos.

1.3. Alcance del sistema

El sistema se enfoca en la recolección de datos aéreos de grupos humanos y su análisis utilizando herramientas de IA. El alcance incluye:

- Captura de videos aéreos mediante un dron en diferentes escenarios y condiciones.
- Almacenamiento y organización de los videos en una base de datos estructurada.
- Procesamiento de los videos para la detección y seguimiento de individuos y grupos humanos.
- Extracción de características relevantes sobre la movilidad y las interacciones grupales.
- Generación de reportes y visualizaciones de los resultados obtenidos.

No se considera dentro del alcance el desarrollo de hardware de drones ni la implementación de modelos de IA desde cero; se utilizarán herramientas y modelos existentes.

2. Requisitos

2.1. Requisitos del sistema

- Base de datos PostgreSQL 17.5.1
- Versión mínima de Python 3.13.3
- Dependencias principales:
  - torch
  - torchvision
  - numpy
  - pandas
  - scikit-learn
  - matplotlib
  - seaborn
- Dependencias procesamiento de imágenes/video:
  - ultralytics
- Dependencias de tracking:
  - deep-sort-realtime
- Dependencias de multimedia:
  - ffmpeg-python
- Dependencias de base de datos:
  - sqlalchemy
  - psycopg2-binary
- Dependencias de variables de entorno:
  - python-dotenv
- Dependencias de barras de progreso:
  - tqdm

2.2. Instrucciones de instalación

1. Instalar PostgreSQL 17.5.1 desde la página oficial de PostgreSQL.
2. Clonar el repositorio del proyecto desde Github el proyecto se encuentra dentro de la carpeta `Implementación`.
3. Crear un entorno virtual en Python:

```
python -m venv .venv
source .venv/bin/activate   # Linux/Mac
.\.venv\Scripts\activate    # Windows
```

4. Instalar las dependencias del proyecto:

```
pip install -r requirements.txt
```

5. Configurar variable de entorno:

```
cp .env.example .env
nano .env  # Editar valores
```

6. Verificar instalación:

```
python tests/check_requirements/
```

Una vez descargadas las dependencias necesarias. Si es la primera vez que se usará el programa el siguiente paso es crear las tablas de la base de datos utilizando los modelos ya creados.

1. Se tiene que abrir el manejar de la base de datos, en el caso de PostgreSQL se llama pgAdmin, para este proyecto se recomienda usar la versión cuatro.
2. Una vez abierto se tiene que crear una base de datos llamada `VideoData`. El nombre de la base de datos, el usuario y la contraseña se pueden modificar en la variable de entorno `DB_CONNECTION_STRING`, se encuentra dentro del archivo `.env`.
3. Con la base de datos creada lo siguiente es ejecutar el archivo `creationOfModels.py`, se encuentra dentro del paquete `database` del proyecto.
4. Al terminar la ejecución ya deben estar las mismas tablas que modelos dentro de la base de datos.

3. Arquitectura

3.1. Diagrama de la estructura del proyecto

```
Proyecto/
├── .venv/
├── src/
│   ├── converted_videos/
│   ├── database/
│   │   ├── connection.py
│   │   ├── creationOfModels.py
│   │   ├── db_crud.py
│   │   └── models/
│   ├── features/
│   │   ├── detect_tracking.py
│   │   ├── handler.py
│   │   ├── reconstruct_video.py
│   │   ├── video_functions.py
│   │   └── video_to_frames_concurrent.py
│   ├── frames/
│   ├── menus/
│   │   └── main_menu.py
│   ├── main.py
│   └── yolo8n.pt
├── tests/
│   └── check_requirements.py
├── .env
└── requirements.txt
```

Figura 1: Diagrama de la estructura del proyecto

3.2. Descripción de los componentes

El orden en el que se presentan los componentes será de acuerdo al diagrama de la *Figura.* 1

src

Contiene el código fuente principal de la aplicación, donde se organizan los módulos y paquetes necesarios para su funcionamiento.

database Este paquete contiene todo lo necesario para la concexión con la base de datos. Así como la creación de las tablas de acuerdo al modelo establecido

connection.py - Script que carga las variables de entorno para poder obtener el `connection string` para realizar la conexión con la base de datos.

creationOfModels.py - Este script crea una conexión con la base de datos para poder crear las tablas con base en los modelos creados.

db_crud.py - Script que contiene los métodos CRUD para la base de datos.

models.py - Script que contiene la descripción de los modelos usados en el proyecto. Estos modelos se usan para la creación de las tablas de la base de datos

features Este paquete contiene los scripts que manipulan los videos.

handler.py - Este archivo es una clase de Python. Tiene dos atributos: el primero guarda la ruta en la que se guardan los videos que se analizarán; mientras que la segunda guarda la ruta en la que se guardarán los resultados que se generen durante la ejecución del programa.
La clase contiene varios métodos. El primero se llama `configure_requirements`, su función es instalar las dependecias necesarias para el funcionamiento del programa. También contiene métodos para obtener y asignar los valores de las rutas antes mencionada.
Los demás métodos manejan la respuesta de los menús para poder direccionar al usuario de forma correcta.

video_functions.py - Esta clase contiene métodos abstractos que llaman a otros scripts de Python para poder realizar el manejo, análisis y creación de videos.

video_to_frames_concurrent.py - Script que con base en un directorio convierte todos los videos `.mp4` a frames de forma concurrente. El script crea una carpeta por cada video dentro del directorio, tiene el mismo nombre que el video convertido.
La variable `sampling_rate` puede ser ajustada de acuerdo a la necesidad. Al empezar el análisis de un video se guardan sus metadatos en la tabla `VideoMetadata` de la base de datos.

frames Esta carpeta contiene carpetas que contienen los frames de cada video convertido. Cada carpeta tiene el mismo nombre que el video del cual se generaron los frames. Dentro de cada carpeta se guardan los frames en formato `.jpg`.

menus Este paquete contiene los scripts que crean los menus y mensajes que se ven en la terminal.

main.py Este script contiene los menus que se ven en la terminal así como algunos de los mensajes que se crean durante la interacción con el usuario.

tests

Directorio dedicado a las pruebas automatizadas, que incluye tanto pruebas unitarias como de integración para asegurar la calidad del código.

check_requirements.py Este script revisa las dependecias contenidas en el archivo `requirements.txt` para poder determinar que dependencias hacen falta en el sistema. Antes de la instalación de las dependencias faltantes se actualiza el `pip` para evitar errores.

ėnv

Este archivo contiene las variables de entorno necesarias para el programa.

requirements.txt

Archivo que lista las dependencias del proyecto, especificando las versiones de los paquetes necesarios para su correcto funcionamiento.

3.3. Flujo de datos

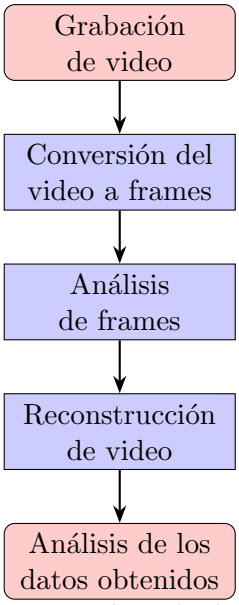

Figura 2: Flujo de datos

4. Configuración

4.1. Archivos de configuración

4.2. Parámetros ajustables