Instituto Tecnológico de Culiacán

Tópicos de Inteligencia Artificial

Sistema de Detección de Matriculas

Unidad: 4

Maestro: Zuriel Dathan Mora Felix

Alumno: Miguel Angel Barraza Leon

Carrera: Ingeniería en Sistemas Computacionales

Numero Control: 20170601

Culiacán, Sinaloa.

30/11/25

Documentación del Sistema de Detección de Matrículas

Objetivo General

Desarrollar un sistema automatizado capaz de detectar con precisión las matrículas de los vehículos a partir de transmisiones de video o imágenes en tiempo real y asociarlas con sus respectivos propietarios registrados en una base de datos.

Descripción del Problema

La gestión manual del control de acceso vehicular y la identificación de propietarios es un proceso lento y propenso a errores humanos. En escenarios como estacionamientos, zonas residenciales o puntos de control de seguridad, es crucial contar con una herramienta que permita identificar rápidamente si un vehículo está autorizado o quién es su propietario, sin necesidad de intervención manual constante.

Justificación

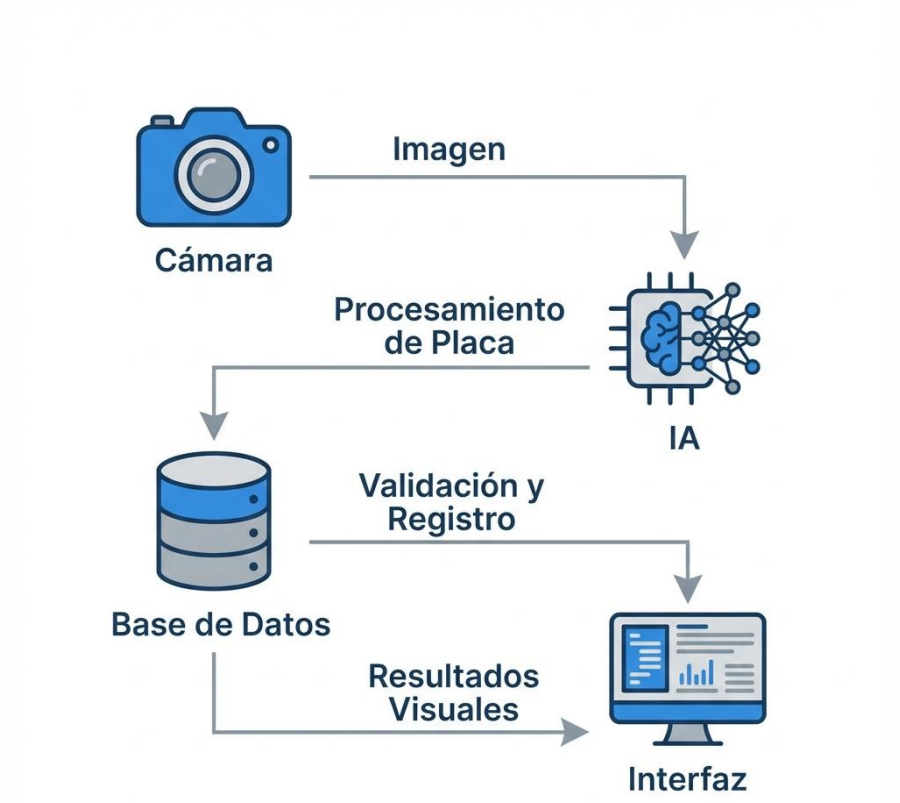
Este proyecto es vital para modernizar la seguridad y la gestión vehicular. Su implementación permite:

- **Automatización**: Reducción de la carga de trabajo del personal de seguridad.
- **Precisión**: Minimización de errores en la lectura de placas.
- **Rapidez**: Identificación instantánea de propietarios para la toma de decisiones.
- **Registro Histórico**: Almacenamiento de evidencias (imágenes y horas) de los accesos.

Componentes Clave del Proyecto

El sistema se compone de cuatro pilares fundamentales:

1. **Base de Datos**: Almacena la información relacional de propietarios, vehículos y el historial de detecciones.
2. **Modelo de Visión Artificial**: Utiliza Inteligencia Artificial para localizar la matrícula en la imagen y extraer el texto (OCR).
3. **Sistema de Vinculación**: Lógica de negocio que cruza los datos detectados con la información almacenada.
4. **Documentación**: Guías completas para usuarios finales y desarrolladores.



Detalles de Implementación y Requisitos Específicos

Construcción de la Base de Datos

Sistema de Gestión de Bases de Datos (DBMS):

Se seleccionó **SQLite** por su ligereza, facilidad de implementación (sin servidor) y eficiencia para aplicaciones de escritorio independientes. Es ideal para este tipo de despliegues donde la portabilidad es clave.

Diseño del Esquema:

El esquema relacional consta de las siguientes tablas principales:

propietarios:

- id (PK): Identificador único.
- nombre: Nombre completo del propietario.
- telefono: Información de contacto.
- direccion: Dirección física.

vehiculos:

- id (PK): Identificador único.
- placa(Unique): Número de matrícula (clave de búsqueda principal).
- marca, modelo, anio, color: Detalles del vehículo.
- propietario_id (FK): Enlace a la tabla propietarios.

lecturas:

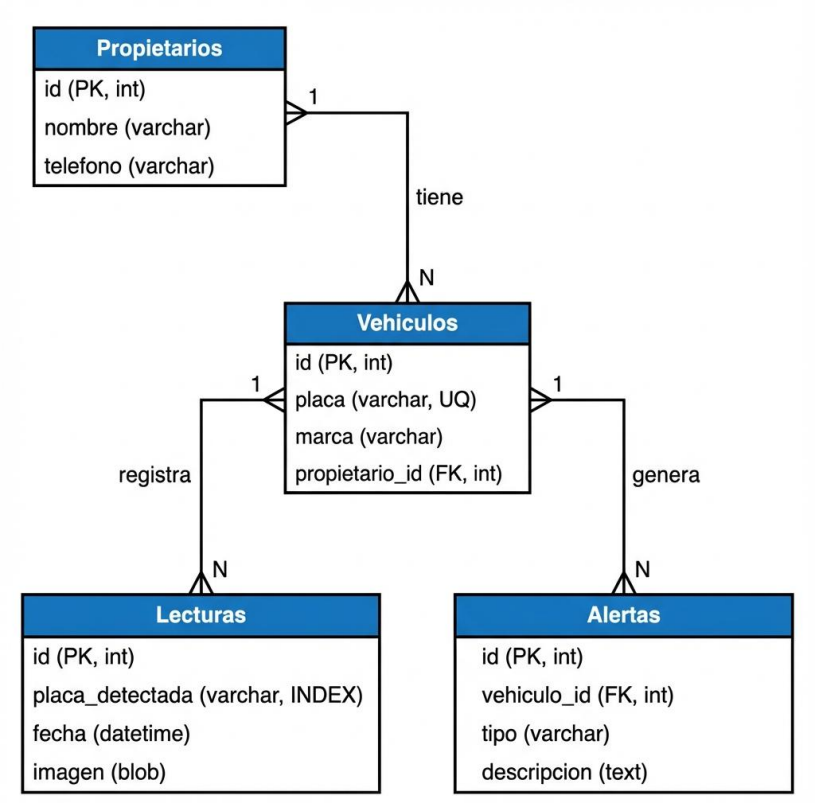
- Tabla transaccional principal donde se registran todos los eventos de detección.
- Campos: fecha, placa_detectada, imagen_ruta (evidencia fotográfica) y coincidencia (nivel de confianza del OCR).

Nota: Aquí se almacenan tanto los vehículos reconocidos como los desconocidos (con vehiculo_id nulo) para mantener un historial completo de accesos.

alertas:

Tabla reservada para notificaciones específicas sobre vehículos registrados (ej. "Vehículo robado", "Acceso denegado").

Estado actual: Esta tabla está definida en el esquema para futura expansión. Actualmente, el sistema gestiona las alertas de "vehículo desconocido" directamente a través de la tabla lecturas para simplificar el flujo de datos.



Implementación:

La base de datos se inicializa mediante el script database/schema.sql y se gestiona a través de database/db.py, asegurando la integridad referencial mediante Claves Foráneas (Foreign Keys).

Modelo de Visión Artificial

Marco de Aprendizaje Automático:

Se utiliza una arquitectura híbrida:

Detección de Objetos:**YOLOv8 (Ultralytics)**. Es el estado del arte en detección en tiempo real, permitiendo localizar la región de la placa con alta precisión. https://huggingface.co/Koushim/yolov8-license-plate-detection/tree/main

Reconocimiento Óptico de Caracteres (OCR): **EasyOCR**. Una librería robusta basada en PyTorch para leer el texto dentro de la región recortada por YOLO.

Procesamiento de Datos:

1. **Entrada**: Frame de video o imagen.
2. **Detección**: YOLO predice la "Bounding Box" de la matrícula.
3. **Preprocesamiento (RO**: Se recorta la imagen de la placa, se convierte a escala de grises y se aplica umbralización/contraste para facilitar la lectura (utils/image_utils.py).
4. **Reconocimiento**: EasyOCR extrae el texto alfanumérico.

Sistema de Vinculación

Lógica de Enlace:

El núcleo del sistema (services/captura_service.py) orquesta el flujo:

1. Recibe el texto de la matrícula detectada.
2. Limpia la cadena (elimina espacios, caracteres especiales).
3. Consulta la tabla vehiculos buscando una coincidencia exacta.
4. Si existe, recupera los datos del propietario asociado.

Manejo de Errores:

Si la placa no existe en la BD, el sistema la marca como "Desconocido" pero registra la lectura para seguridad.

Se manejan excepciones de conexión a la BD y errores de lectura de archivos de video para evitar cierres inesperados.

Manual de Usuario (Documentación para el Usuario Final)

Instalación y Ejecución

1. Asegúrese de tener la carpeta del proyecto en su equipo.
2. Ejecute el archivo principal haciendo doble clic o desde la terminal:

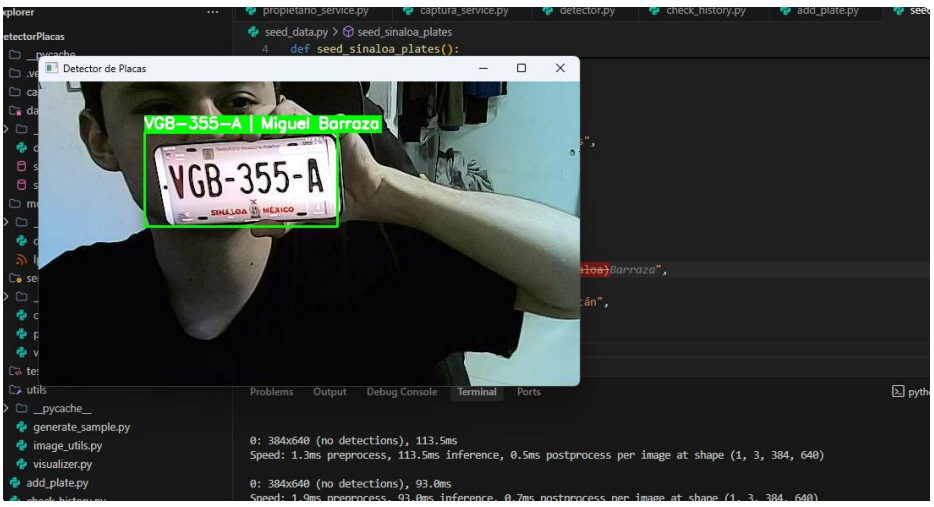
   python main.py

Guía de Uso

Al iniciar, verá el siguiente menú principal:

1. **Iniciar Detección**:

- Seleccione esta opción para abrir la cámara.
- El sistema mostrará en pantalla un recuadro verde sobre las placas detectadas.
- Si la placa está registrada, mostrará el nombre del dueño. Si no, dirá "Desconocido".
- Presione la tecla **'q'** para salir de la cámara.



2. **Registrar Nuevo Vehículo**:

- Siga las instrucciones en pantalla para ingresar Nombre, Teléfono, Dirección, Placa, Marca, etc.
- Esto "dará de alta" al vehículo en el sistema de confianza.

```
(.venv) PS C:\Users\Migue\OneDrive\Escritorio\DetectorPlacas> python main.py
Base de datos inicializada correctamente.

--- SISTEMA DE DETECCIÓN DE MATRÍCULAS ---
1. Iniciar Detección (Cámara/Video)
2. Registrar Nuevo Vehículo
3. Ver Propietarios Registrados
4. Salir
Seleccione una opción: 2

--- Registro de Vehículo ---
Nombre del Propietario: Carlos
Teléfono: 678593403
Dirección: Barrancos
Placa (ej. ABC-123): VGB-356-B
Marca: Nissan
Modelo: Versa
Año: 2022
Color: Rojo
Vehículo registrado exitosamente.

--- SISTEMA DE DETECCIÓN DE MATRÍCULAS ---
1. Iniciar Detección (Cámara/Video)
2. Registrar Nuevo Vehículo
3. Ver Propietarios Registrados
4. Salir
Seleccione una opción:
```

3. **Ver Propietarios**:

- Muestra una lista rápida de todos los vehículos actualmente registrados en la base de datos

```
--- SISTEMA DE DETECCIÓN DE MATRÍCULAS ---
1. Iniciar Detección (Cámara/Video)
2. Registrar Nuevo Vehículo
3. Ver Propietarios Registrados
4. Salir
Seleccione una opción: 3

--- Vehículos Registrados ---
Propietario: Juan Pérez (Sinaloa) | Placa: VSA-1234 | Vehículo: Toyota
Propietario: María López (Sinaloa) | Placa: VLX-5678 | Vehículo: Nissan
Propietario: Carlos Ramos (Sinaloa) | Placa: VRG-9012 | Vehículo: Ford
Propietario: Usuario Solicitado | Placa: VPM-45-32 | Vehículo: Chevrolet
Propietario: Miguel Barraza | Placa: VGB-355-A | Vehículo: Chevrolet
Propietario: Carlos  | Placa: VGB-356-B | Vehículo: Nissan
```

Solución de Problemas (Troubleshooting)

La cámara no abre: Verifique que ninguna otra aplicación (Zoom, Teams) esté usando la cámara. Intente cambiar el índice de la cámara (0 o 1) en el menú.

No detecta placas: Asegúrese de que haya buena iluminación. Si la placa está muy sucia o en un ángulo extremo, el modelo podría no reconocerla.

Lentitud: El reconocimiento de texto es un proceso pesado. Si su computadora es antigua, el video puede verse con retraso

Documentación Técnica y de Instalación

Requisitos del Sistema

Sistema Operativo: Windows 10/11, Linux o macOS.

Python: Versión 3.8 o superior.

Dependencias: Listadas en `requirements.txt` (opencv-python, ultralytics, easyocr, etc.).

Proceso de Instalación para Desarrolladores

1. **Clonar el repositorio**:

   git clone https://github.com/mabi2002/DetectorPlacas.git

   cd DetectorPlacas

2. **Crear entorno virtual**:

   python -m venv .venv

   source .venv/bin/activate # En Windows: .venv\Scripts\activate

3. **Instalar librerías**:

   pip install -r requirements.txt

Configuración Inicial:

El sistema creará automáticamente la base de datos sistema_placas.db en la primera ejecución.

Arquitectura del Código

El proyecto sigue una estructura modular para facilitar el mantenimiento:

- main.py: Punto de entrada y gestión del menú CLI.
- database/: Manejo de conexiones SQL y scripts de creación de tablas.
- model/: Contiene la clase LicensePlateDetector que encapsula la lógica de IA (YOLO + EasyOCR).
- services/: Capa de lógica de negocio (VehiculoService, CapturaService) que conecta la interfaz con la base de datos y los modelos.
- utils/: Funciones auxiliares para procesamiento de imágenes y visualización.

Pruebas y Validación

Para garantizar la robustez del sistema, se han implementado scripts de pruebas automatizadas y herramientas de generación de datos sintéticos.

Pruebas Unitarias e Integración

El proyecto incluye un conjunto de pruebas en la carpeta tests/ que validan:

1. Conexión y creación de tablas en la base de datos.
2. Registro correcto de propietarios y vehículos.
3. Búsqueda eficiente de placas (existentes e inexistentes).

Ejecución de pruebas:

python -m tests.test_integration

```
(.venv) PS C:\Users\Migue\OneDrive\Escritorio\DetectorPlacas> python tests/test_integration.py
Base de datos inicializada correctamente.
Propietario registrado con ID: 1
.Vehículo registrado con ID: 1
.Búsqueda de placa exitosa.
.Manejo de placa inexistente correcto.
.
----------------------------------------------------------------------
Ran 4 tests in 0.065s

OK
(.venv) PS C:\Users\Migue\OneDrive\Escritorio\DetectorPlacas>
```

Generación de Datos de Prueba

Si no dispone de un vehículo real para probar, puede generar imágenes de matrículas sintéticas con la utilidad incluida:

python utils/generate_sample.py

Esto creará imágenes en la carpeta ejemplos/ que puede pasar al detector para validar el reconocimiento OCR.