Automatic Number Plate Recognition (ANPR) System

EURON

Automatic Number Plate Recognition (ANPR) System

Automatic Number Plate Recognition (ANPR) is a computer vision-based system that detects, extracts, and recognizes vehicle license plate numbers from images or video streams.

1. High-Level Architecture

The ANPR system consists of multiple components working together in real time. The key components are:

1. **Data Acquisition** (Cameras & Sensors)
2. **Edge Processing (Preprocessing & Initial Image Processing)**
3. **Data Ingestion (Streaming Pipeline)**
4. **Computer Vision-based License Plate Detection**
5. **OCR-based License Plate Recognition**
6. **Post-Processing & Validation**
7. **Data Storage & Retrieval**
8. **Business Logic (Alerts, Analytics, Search, etc.)**
9. **Visualization & Reporting**
10. **Integration with External Systems (Law Enforcement, Toll Systems, Parking Management, etc.)**

---

2. Detailed Component-wise Architecture Flow

Each component is elaborated below in a highly detailed manner.

---

1. Data Acquisition (Cameras & Sensors)

- **Input Sources:**
  - CCTV Cameras (Traffic, Parking Lots, Toll Booths, Entry Gates)
  - Mobile Cameras (Law Enforcement Officers' Handheld Devices)
  - Drones (For surveillance)
  - IR/Night Vision Cameras (For low-light conditions)
- **Key Considerations:**
  - **Resolution & Frame Rate:** Minimum 1080p at 30FPS for optimal clarity.
  - **Angle of Capture:** Must ensure that plates are visible from different angles.
  - **Weather-Proofing:** Cameras should handle glare, rain, fog, and night-time visibility.
  - **Edge AI Cameras:** On-camera initial processing for speed optimization.

2. Edge Processing (Preprocessing & Initial Image Processing)

- **Hardware Considerations:**
  - NVIDIA Jetson Nano, Jetson Xavier, Intel Movidius, Google Coral for **on-device AI inference**.
  - FPGA-based acceleration for real-time applications.
- **Tasks Performed:**
  - **Noise Reduction:** Gaussian Blur, Median Blur to reduce pixel noise.
  - **Frame Selection:** Selecting the clearest frame from video streams.
  - **ROI Extraction:** Identifying Regions of Interest (ROI) using basic edge detection before sending to main processing units.
- **Edge Decision:**
  - If license plate is detected **with high confidence**, process locally.
  - Otherwise, **stream to cloud or edge server** for further analysis.

---

3. Data Ingestion (Streaming Pipeline)

- **Technologies Used:**
  - **Kafka or Pulsar:** Real-time streaming pipeline for event-driven processing.
  - **Apache NiFi:** For ingesting images and video frames into processing pipelines.
  - **RTSP Streaming:** Direct ingestion of live streams.
- **Flow:**
  1. **Raw video streams → Frame extraction → Kafka topics → AI models**
  2. **High-traffic areas:** Edge AI filtering reduces unnecessary frames before ingestion.
  3. **Multiple sources (CCTV, mobile, etc.) → Merged into unified pipeline.**

---

4. Computer Vision-based License Plate Detection

- **Goal:** Detect license plates in an image.
- **Models Used:**
  - YOLOv8 (Ultralytics) – **Fast & accurate object detection.**
  - SSD (Single Shot Detector) – **Good for mobile and edge inference.**
  - Faster R-CNN – **High accuracy but computationally heavy.**
  - OpenCV’s Haar Cascades – **Lightweight but less accurate.**
- **Processing Pipeline:**
  1. Convert image to grayscale.
  2. Apply Adaptive Thresholding to enhance edges.
  3. Apply Edge Detection (Canny) + Contour Detection.

4. Use **CNN-based models (YOLO, Faster R-CNN) to detect the bounding box of the license plate**.
5. Crop and forward **detected plate** to the next stage.

- **Challenges Handled:**
  - Handling different plate sizes and orientations (Perspective Transformations)
  - Night-time recognition (IR-assisted capture)
  - Motion blur correction

---

5. OCR-based License Plate Recognition

- **Goal:** Convert cropped license plate image into text.
- **Approaches:**
  - **Tesseract OCR (Open Source, Traditional)**
  - **Deep Learning-based OCR (CRNN + CTC Loss)**
  - **CNN-based OCR (Custom trained CNN models like EAST + LSTM)**
  - **Google Vision API / AWS Rekognition OCR for cloud-based OCR solutions**
- **Processing Pipeline:**
  1. **Preprocessing:** Denoising, Resizing, Contrast Enhancement
  2. **Segmentation:** Segment each character using Connected Components or Deep Learning.
  3. **Character Recognition:** Use trained CNN/LSTM model for text recognition.
  4. **Post-processing:** String correction using Spell Check & N-Gram Matching.

---

6. Post-Processing & Validation

- **Error Correction:**
  - Dictionary-based correction to fix misrecognized characters.
  - Cross-referencing against a vehicle registration database for verification.
- **Duplicate Frame Removal:**
  - If the same number is detected multiple times in a short time, keep only the best-quality recognition result.

---

7. Data Storage & Retrieval

- **Databases Used:**
  - **MongoDB** (NoSQL, flexible storage for images & metadata)
  - **PostgreSQL** (for structured records like vehicle registrations)
  - **Elasticsearch** (for fast search and retrieval of plates)
- **Schema Design Example:**

```
{
  "plate_number": "MH12AB1234",
  "timestamp": "2025-02-10T14:23:11Z",
  "camera_id": "CAM_001",
  "location": "Toll Plaza - Mumbai",
  "image_url": "s3://anpr-data/plate1.jpg",
  "confidence": 92.3
}
```

---

8. Business Logic (Alerts, Analytics, Search, etc.)

- **Alert System:**
    - If a plate matches **stolen vehicles, blacklisted cars**, send real-time alert to law enforcement.
    - If a car enters **restricted areas**, trigger a notification.
- **Analytics & Dashboards:**
    - Number of vehicles per hour, per location.
    - Most common plate numbers.
    - Traffic heatmaps.

---

9. Visualization & Reporting

- **Tools Used:**
    - **Grafana, Kibana, Tableau, Power BI** for visualization.
    - **Flask/Django + React.js** for real-time monitoring dashboard.

---

10. Integration with External Systems

- **Integration Points:**
    - **Law Enforcement Database:** Check for stolen vehicles.
    - **Toll Management:** Automatic toll deduction.
    - **Parking Management Systems:** Entry/Exit time tracking.
- **APIs & Protocols Used:**
    - RESTful APIs for integration with external systems.
    - WebSocket for real-time alerts.
    - MQTT for IoT-based camera alerts.

Technology Stack

Component	Technology
Edge AI	NVIDIA Jetson, Intel Movidius
Object Detection	YOLOv8, Faster R-CNN
OCR	CRNN, Tesseract, AWS Rekognition
Streaming	Kafka, Pulsar, Apache NiFi
Storage	MongoDB, Elasticsearch, PostgreSQL
Visualization	Kibana, Grafana, Tableau
API Backend	Flask, FastAPI, Django
Frontend	React.js, Angular

Final Workflow Summary

1. **Capture Image/Video** from CCTV or Mobile Cameras.
2. **Edge Preprocessing** filters & extracts plates.
3. **Streaming Pipeline** ingests frames into Kafka/NiFi.
4. **License Plate Detection Model** extracts plates.
5. **OCR Model Recognizes Text** from plates.
6. **Post-processing & Validation** enhances accuracy.
7. **Store Data in NoSQL/SQL Databases**.
8. **Trigger Alerts, Search, & Visualize Data**.