**Real-time Auto License Plate Recognition with Jetson Nano**

**1. Introduction**

**1.1 Project Overview**

This project aims to develop a real-time auto license plate recognition system using the Jetson Nano. The system will capture license plate information from live video feeds, process it, and display or store the recognized data. This project integrates computer vision, deep learning, and embedded system development for effective image recognition.

**1.2 Objectives**

* Implement real-time video capture from a camera.
* Detect license plates using a machine learning algorithm.
* Extract text from detected plates using Optical Character Recognition (OCR).
* Store or display the recognized license plate information.

**1.3 Applications**

* Automated vehicle entry systems.
* Traffic monitoring and law enforcement.
* Parking management solutions.

**2. Setup and Requirements**

**2.1 Hardware Requirements**

* **Jetson Nano**: Main computing unit.
* **Camera**: USB camera or compatible camera module (e.g., Raspberry Pi Camera).
* **Power Supply**: Appropriate power adapter for the Jetson Nano.
* **Storage**: MicroSD card (at least 16GB) for OS and software installation.

**2.2 Software Requirements**

* **Operating System**: Ubuntu-based Jetson Linux.
* **Required Libraries**:
  + OpenCV: For video processing and image manipulation.
  + Tesseract: For text recognition from images.
  + PyTesseract: Python wrapper for Tesseract.

**Installation Steps**

1. **Update the System**:

**Bash code:**

sudo apt-get update

sudo apt-get upgrade

1. **Install OpenCV**:

**Bash code:**

sudo apt-get install python3-opencv

1. **Install Tesseract**:

**Bash code:**

sudo apt-get install tesseract-ocr

1. **Install PyTesseract**:

**Bash code:**

pip install pytesseract

**2.3 Additional Libraries**

* If using deep learning models (like YOLO), install additional libraries (e.g., TensorFlow or PyTorch).

**3. Implementation Details**

**3.1 Environment Setup**

**3.1.1 Jetson Nano Configuration**

* Follow the official NVIDIA instructions to flash the Jetson Nano with the latest image.
* Ensure network connectivity for package installations.

**3.1.2 Setting Up the Camera**

* Connect the camera and verify its functionality with a simple script:

**Python Code:**

import cv2

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if ret:

cv2.imshow('Camera Feed', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**3.2 Real-time Video Processing**

**3.2.1 Video Capture Script**

**Python Code:**

import cv2

# Initialize video capture

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if not ret:

break

cv2.imshow('Live Feed', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**3.3 License Plate Detection**

**3.3.1 Detection Algorithm**

* Use Haar Cascade or a deep learning model for license plate detection. Here’s an example using Haar Cascade:

**Python Code:**

# Load Haar Cascade model

plate\_cascade = cv2.CascadeClassifier('haarcascade\_russian\_plate\_number.xml')

# Detect plates

plates = plate\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

for (x, y, w, h) in plates:

cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

**3.4 Optical Character Recognition (OCR)**

**3.4.1 Tesseract Configuration**

* Crop the detected plate region and use Tesseract for text extraction:

**Python Code:**

import pytesseract

# Crop and recognize

plate\_image = gray[y:y + h, x:x + w]

plate\_text = pytesseract.image\_to\_string(plate\_image, config='--psm 8')

**3.5 Integration**

**3.5.1 Full Pipeline Code**

Combine the detection and recognition into a single script:

**Python Code:**

import cv2

import pytesseract

# Load Haar Cascade

plate\_cascade = cv2.CascadeClassifier('haarcascade\_russian\_plate\_number.xml')

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

plates = plate\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

for (x, y, w, h) in plates:

cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

plate\_image = gray[y:y + h, x:x + w]

plate\_text = pytesseract.image\_to\_string(plate\_image, config='--psm 8')

cv2.putText(frame, plate\_text.strip(), (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.imshow('License Plate Recognition', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**4. Testing and Validation**

**4.1 Test Scenarios**

* Different lighting conditions (day/night).
* Various license plate designs and orientations.
* Speed of vehicles (if applicable).

**4.2 Performance Metrics**

* **Accuracy**: Percentage of correctly recognized plates.
* **Processing Speed**: Frames per second (FPS) during detection and recognition.

**5. Results and Discussion**

**5.1 Output Analysis**

* Provide screenshots or video clips demonstrating the system in action.
* Summary of successful detections and any missed plates.

**5.2 Challenges and Solutions**

* Discuss any issues encountered (e.g., false positives, detection speed).
* Solutions implemented to address these challenges.

**6. Conclusion**

**6.1 Summary**

* Recap the project goals, methods used, and results achieved.
* Highlight the importance of real-time recognition in practical applications.

**6.2 Future Work**

* Suggestions for improvements (e.g., using deep learning for detection).
* Potential features to add (e.g., database integration for tracking).