Проект ANPR (Автоматично Разпознаване на Регистрационни Табели) – Подробна документация

1. Въведение

Този документ предоставя пълно описание на проекта ANPR, включително Python кода, функциите, workflow-а, OCR техниките, обучението на YOLO модела и подготовката на данните. Съдържа също така обяснение на използваните технологии, схеми, графики и примери със снимки.

---

2. Обзор на проекта

Системата заснема видео кадри от камера, открива регистрационни табели в реално време с помощта на YOLO модел за детекция на обекти, разпознава текста на табелите чрез OCR (EasyOCR), валидира текста спрямо българския формат на номера и записва потвърдените табели в база данни Supabase. Процесът е автоматизиран и оптимизиран за бързина и точност.

2.1 Основни компоненти

- **Заснемане на кадри:** Непрекъснато заснемане на кадри от видео източник.
- **YOLO модел:** Открива областите на табелите в кадрите.
- **OCR (EasyOCR):** Извлича текста от откритите табели.
- **Валидация на данните:** Проверява дали текста съответства на български формат.
- **Запис в база данни:** Съхранява потвърдените табели в Supabase.
- **Queue система:** Асинхронна обработка на кадрите за плавна работа на системата.

---

3. Обяснение на кода

3.1 Импорти и конфигурация

```
import os, re, time, json, cv2
from threading import Thread
from collections import defaultdict
from queue import Queue
import easyocr
from supabase import create_client
from ultralytics import YOLO
```

- `os, re, time, json, cv2`: Библиотеки за работа с файлове, регулярни изрази, таймери, JSON и обработка на изображения. - `Thread, Queue`: За мултитрединг и асинхронна обработка на кадри. - `easyocr`: За разпознаване на текст. - `supabase`: За работа с облачна база данни. - `YOLO`: Модел за детекция на обекти.

Конфигурационни параметри:

```
CAMERA_SOURCE = 0
SUPABASE_URL = "..."
SUPABASE_KEY = "..."
TABLE_NAME = "plates"
CONFIRM_FRAMES = 2
FRAME_SKIP = 4
QUEUE_MAX = 5
```

- `CONFIRM_FRAMES`: Брой последователни детекции за потвърждаване на табела. - `FRAME_SKIP`: Прескача кадри за оптимизация. - `QUEUE_MAX`: Максимален брой кадри в опашката за обработка.

3.2 Директории и Regex

```
DATASET_DIR = "dataset_auto"
IMAGES_DIR = os.path.join(DATASET_DIR, "images")
LABELS_DIR = os.path.join(DATASET_DIR, "labels")
BG_PLATE_REGEX = re.compile(r'^[A-Z]{1,2}[0-9]{4}[A-Z]{2}$')
```

- `IMAGES_DIR` и `LABELS_DIR` съхраняват автоматично събраните изображения и техните анотации. - `BG_PLATE_REGEX` проверява дали разпознатият текст отговаря на българския формат на регистрационни номера (1–2 букви + 4 цифри + 2 букви).

3.3 Опашка и множества

```
frame_queue = Queue(maxsize=QUEUE_MAX)
seen_counts = defaultdict(int)
confirmed = set()
```

- `frame_queue`: Съхранява кадрите, които чакат за обработка. - `seen_counts`: Брои колко пъти е разпозната всяка табела. - `confirmed`: Множество от вече потвърдени табели, за да се избегнат дублирани записи.

3.4 Инициализация на моделите

```
reader = easyocr.Reader(['bg'], gpu=False)
model = YOLO("iliev_licence_plate.pt")
model.fuse()
```

- `easyocr.Reader(['bg'])`: OCR за български език. - `YOLO`: Моделът е предварително обучен за български регистрационни табели. - `model.fuse()`: Оптимизация на YOLO модела за по-бързо inference.

3.5 Функция за заснемане на кадри

```
def capture_frames():
    cap = cv2.VideoCapture(CAMERA_SOURCE)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
    if not cap.isOpened(): return print("Cannot open camera")
    while True:
        ret, frame = cap.read()
        if ret and not frame_queue.full(): frame_queue.put(frame)
        else: time.sleep(0.05)
```

- Заснема кадри от камерата и ги добавя в опашката за обработка. - Ако камерата не е достъпна, се отпечатва съобщение. - Ако опашката е пълна, изчаква 0.05 секунди.

3.6 Функции за обработка на текст

```
def clean_text(text):
    return re.sub(r'^(.)\1+|[^A-Z0-9]', lambda m: m.group(1) if m.group(1) else
'', text.upper())
```

- Премахва повторения на първия символ и всички невалидни символи.

```
def ocr_plate(img):
    res = reader.readtext(cv2.cvtColor(img, cv2.COLOR_BGR2RGB),
allowlist='ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789', detail=0)
    if not res: return None
    text = clean_text(''.join(res))
    return text if 6 <= len(text) <= 9 else None
```

- Извлича текст чрез EasyOCR и го почиства. - Връща None, ако текстът не е валиден.

```
def is_valid_bg_plate(text): return bool(BG_PLATE_REGEX.match(text))```
- Проверява дали текстът съответства на българския формат.

```python
def save_plate_db(text):
    try: supabase.table(TABLE_NAME).insert({"plate_text": text}).execute()
    except Exception as e: print("Supabase error:", e)
```

- Записва потвърдените табели в база данни Supabase.

3.7 Основен цикъл

```
Thread(target=capture_frames, daemon=True).start()
print("Parking System Started. Press 'q' to quit.")

frame_index = 0
while True:
    if frame_queue.empty(): time.sleep(0.01); continue
    frame = frame_queue.get()
    frame_index += 1
    if frame_index % FRAME_SKIP != 0: continue

    results = model(frame, conf=0.4, verbose=False)
    for box in results[0].boxes.xyxy:
        x1, y1, x2, y2 = map(int, box)
        plate_roi = frame[y1:y2, x1:x2]
        text = ocr_plate(plate_roi)
        if not text: continue

        valid = is_valid_bg_plate(text)
        color = (0,255,0) if valid else (0,0,255)
        cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
        cv2.putText(frame, text, (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
color, 2)

        if valid:
            seen_counts[text] += 1
            if seen_counts[text] >= CONFIRM_FRAMES and text not in confirmed:
                confirmed.add(text)
                print("Confirmed Plate:", text)
                save_plate_db(text)
```
```

```
    cv2.imshow("Parking System", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"): break

cv2.destroyAllWindows()
print("Parking System Stopped.")
```

- Основен цикъл за обработка на кадри. - Детекция с YOLO и OCR с EasyOCR. - Валидация на табели и потвърждение чрез `CONFIRM_FRAMES`. - Показва резултатите върху видеото в реално време.

---

4. Обучение на YOLO модела

1. **Събиране на данни:**
2. Снимки на коли се изтеглят автоматично от `cars.bg`.
3. Използва се `LabelImg` за анотиране на табели в снимките.
4. **Формат на данните:**
5. Всяка снимка има съответен `.txt` файл с координати на bounding box и клас (plate = 0).
6. **Обучение:**
7. Използва се Ultralytics YOLOv8.
8. Настройки: `img=640`, `batch=16`, `epochs=60`, `data=dataset.yaml`.
9. **Резултати:**
10. Epoch, GPU usage, загуба (box_loss, cls_loss, dfl_loss).
11. mAP50, mAP50-95 се следят за точност.

4.1 Диаграма на workflow

```
[ Камера ] --> [ YOLO детекция ] --> [ Изрязване ROI ] --> [ OCR EasyOCR ] -->
[ Валидация на табелата ] --> [ Потвърждение ] --> [ Supabase ]
```

- Зелените линии показват успешни табели, червените – невалидни.

---

5. Заключение

Проектът комбинира детекция (YOLO), разпознаване на текст (OCR), проверка на формати, и база данни за автоматично разпознаване на български регистрационни табели. Системата е гъвкава и може да се разшири с други OCR или CRNN технологии за по-висока точност.

(В Word версията може да се добавят снимки на LabelImg интерфейс, примери на dataset и графики на загубите при обучение.)