1) ابتدا با استفاده ازتکه کد اول پنجره ی تیره را از عکس کراپ میکنیم :

1. image = cv2.imread(image_path)

تصویر ورودی را از مسیر داده شده بارگذاری می‌کند.

2. gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

تصویر رنگی را به تصویر خاکستری تبدیل می‌کند تا پردازش آن ساده‌تر شود.

3. _, thresh = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY_INV)

تصویر خاکستری را باینری می‌کند؛ یعنی پیکسل‌های با مقدار کمتر از ۵۰ را سفید و بقیه را سیاه می‌کند تا نواحی تاریک بهتر مشخص شوند.

4. contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

کانتورهای موجود در تصویر باینری را پیدا می‌کند که برای شناسایی نواحی تاریک مورد استفاده قرار می‌گیرند.

5. largest_contour = max(contours, key=cv2.contourArea)

بزرگترین کانتور از بین کانتورهای پیدا شده را انتخاب می‌کند، که نشان‌دهنده بزرگترین ناحیه تاریک است.

6. x, y, w, h = cv2.boundingRect(largest_contour)

یک مستطیل محدودکننده، دور بزرگترین کانتور رسم می‌کند و مختصات آن را به دست می‌آورد.

7. dark_window = image[y:y+h, x:x+w]

ناحیه شناسایی شده را از تصویر اصلی برش می‌دهد.

8. cv2.imwrite(cropped_image_path, dark_window)

تصویر برش داده شده را در فایل جدید ذخیره می‌کند.

خروجی :

2) با استفاده ازتکه کد دوم پنجره سفید داخل پنجره تیره که همان پلاک است را خارج میکنیم :

توضیح کد مشابه با حالت قبل است با این تفاوت که در اینجا به دنبال پنجره سفید هستیم .

خروجی :

3) با استفاده ازتکه کد سوم کنتراست را افزایش میدهیم :

1. image_src = cv2.imread(image_file)

تصویر ورودی را از مسیر داده شده بارگذاری می‌کند.

2. در تابع read_this، بسته به این که gray_scale باشد یا نه، تصویر به خاکستری یا RGB تبدیل می‌شود.

3. r_image, g_image, b_image = cv2.split(image_src)

در صورتی که تصویر رنگی باشد، آن را به سه کانال رنگی قرمز، سبز و آبی تقسیم می‌کند.

4. r_image_eq = cv2.equalizeHist(r_image)

در این خط (و خطوط مشابه برای کانال‌های سبز و آبی)، هیستوگرام هر کانال رنگی را برابر می‌کند تا کنتراست آن را افزایش دهد.

5. image_eq = cv2.merge((r_image_eq, g_image_eq, b_image_eq))

کانال‌های رنگی با کنتراست افزایش‌یافته را دوباره با هم ترکیب می‌کند و تصویر رنگی بهبودیافته را می‌سازد.

6. cv2.imwrite("increasedContrastCar.jpg", image_eq)

تصویر با کنتراست بهبودیافته را با نام increasedContrastCar.jpg ذخیره می‌کند.

خروجی:

4) با استفاده ازتکه کد چهارم ، متن پلاک را استخراج میکنیم :

1. image = Image.open(image_path)

تصویر با نام increasedContrastCar.jpg را بارگذاری می‌کند و آماده پردازش می‌سازد.

2. 'custom_config = r'--oem 3 --psm 6

تنظیمات اختصاصی برای pytesseract تعریف می‌کند:

- --oem 3 نشان‌دهنده استفاده از بهترین مدل OCR (ترکیبی از مدل‌های عصبی و استاندارد) است.
- --psm 6 حالت شناسایی صفحه را مشخص می‌کند، که برای بلوک‌های متنی مناسب است.

3. text = pytesseract.image_to_string(image, config=custom_config)

با استفاده از تنظیمات تعریف‌شده، OCR را بر روی تصویر اجرا می‌کند و متن شناسایی‌شده را به‌عنوان رشته‌ای متنی برمی‌گرداند.

خروجی :

```
print(text)
```

```
Requirement already satisfied: pytesseract in /usr/local/lib/python3.10/dist-packages (0.3.13)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from pytesseract) (24.1)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from pytesseract) (10.4.0)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tesseract-ocr is already the newest version (4.1.1-2.1build1).
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
MH2OEE 7602
```