Rapport sur le Code de OCR

Vue d'ensemble:

Ce rapport couvre un script Python conçu pour traiter une image d'un document, extraire les informations pertinentes à l'aide de l'OCR (Reconnaissance Optique de Caractères), et identifier des détails clés tels que le numéro d'identité, le groupe sanguin et la date de naissance. Le script utilise plusieurs bibliothèques, notamment OpenCV, imutils, pytesseract, PIL et re pour diverses tâches.

Décomposition du Code:

Importations et Chargement de l'Image:

```python
import cv2 as cv
import numpy as np
from imutils.perspective import four_point_transform
import pytesseract
import matplotlib.pyplot as plt
from PIL import Image
import re
import json
```

Le script commence par importer les bibliothèques nécessaires :

cv2 : Pour les tâches de traitement d'image.
numpy : Pour les opérations numériques.
imutils : Pour les transformations de perspective.
pytesseract : Pour l'OCR.
matplotlib : Pour l'affichage des images (bien que non utilisé dans le script final).
PIL : Pour la manipulation des images.
re et json : Pour les opérations regex et la gestion de JSON.

```
img = cv.imread("nin.jpeg")
WIDTH, HEIGHT, CHANNEL = img.shape
```

L'image "nin.jpeg" est chargée, et ses dimensions sont récupérées.

Détection de Document:

Fonction : scan_detection

```
def scan_detection(image):
    global document_contour
    document_contour = np.array([[0, 0], [WIDTH, 0], [WIDTH, HEIGHT], [0, HEIGHT]])

    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    blur = cv.GaussianBlur(gray, (5, 5), 0)
    _, threshold = cv.threshold(blur, 50, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)

    contours, _ = cv.findContours(threshold, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv.contourArea, reverse=True)

    max_area = 0
    for contour in contours:
        area = cv.contourArea(contour)
        if area > 1000:
            peri = cv.arcLength(contour, True)
            approx = cv.approxPolyDP(contour, 0.015 * peri, True)
            if area > max_area and len(approx) == 4:
                document_contour = approx
                max_area = area

    cv.drawContours(image, [document_contour], -1, (0, 255, 0), 3)
```

-Convertit l'image en niveaux de gris.

-Applique un flou gaussien pour réduire le bruit.

-Utilise le seuillage d'Otsu pour créer une image binaire.

-Trouve les contours dans l'image binaire et identifie le plus grand contour qui approximativement est un quadrilatère, supposé être les bords du document.

Application de la Détection de Document:

```
scan_detection(img)
warped = four_point_transform(img, document_contour.reshape(4, 2))
```

Le contour détecté est utilisé pour déformer l'image afin d'obtenir une perspective de haut en bas du document.

Traitement de l'Image pour l'OCR:

Fonction : image_processing

```
def image_processing(image):
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    _, threshold = cv.threshold(gray, 80, 255, cv.THRESH_BINARY)
    return threshold
```

Convertit l'image déformée en niveaux de gris.
Applique un seuillage binaire pour préparer l'image pour l'OCR.

OCR et Extraction d'Informations:

```
cv.imshow('Cropped Document', processed)
cv.waitKey(0)
cv.destroyAllWindows()


ocr_text = pytesseract.image_to_string(processed)
print(ocr_text)
```

Affiche l'image traitée.
Utilise Tesseract pour extraire le texte de l'image traitée.

Fonction : extract_information

```python
def extract_information(text, doc_type):
    if doc_type == "national_id":
        id_pattern = re.compile(r'\b\d{18}\b')
        blood_type_pattern = re.compile(r'\s*([ABO]{1,2}[+-])')
        dob_pattern = re.compile(r'\b\d{4}\.\d{2}\.\d{2}\b')

        id_match = id_pattern.search(text)
        blood_type_matches = blood_type_pattern.findall(text)
        dob_matches = dob_pattern.findall(text)

        id_number = id_match.group() if id_match else ''
        blood_type = blood_type_matches[-1] if blood_type_matches else ''
        dob = dob_matches[-1] if dob_matches else ''

        return {
            "ID": id_number,
            "blood_type": blood_type,
            "dob": dob
        }
    elif doc_type == "driving_license":
        id_pattern = re.compile(r'\b\d{18}\b')
        dob_pattern = re.compile(r'\b\d{4}\.\d{2}\.\d{2}\b')

        id_match = id_pattern.search(text)
        dob_matches = dob_pattern.findall(text)

        id_number = id_match.group() if id_match else ''
        dob = dob_matches[-1] if dob_matches else ''
        return {
            "ID": id_number,
            "dob": dob
        }
```

Utilise des modèles regex pour extraire le numéro d'identité, le groupe sanguin et la date de naissance en fonction du type de document.

Conclusion:

Le script traite efficacement une image pour détecter et extraire un document, le prépare pour l'OCR, et extrait des informations clés à l'aide de modèles regex. L'utilisation d'OpenCV pour le traitement d'image, de pytesseract pour l'OCR et de regex pour l'extraction d'informations fournit une solution robuste pour la numérisation et l'extraction de données de documents.