Flask 기반 OpenCV 이미지 분석 에이전트 구현 가이드

1. 프로젝트 개요

이 문서는 Flask 웹 프레임워크, OpenCV 이미지 처리 라이브러리, 그리고 ChatGPT API를 결합하여 이미지 내의 도형을 분석하고 상세한 설명을 제공하는 웹 에이전트를 구현하는 과정을 단계별로 설명합니다.

1.1 시스템 구성

본 에이전트는 다음과 같은 구성 요소들로 이루어져 있습니다:

- **Flask**: 웹 애플리케이션 프레임워크로 HTTP 요청을 처리
- **OpenCV**: 이미지 처리 및 도형 분석을 위한 컴퓨터 비전 라이브러리
- **ChatGPT API**: 이미지와 OpenCV 분석 결과를 바탕으로 자연어 설명 생성

1.2 작동 방식

사용자가 이미지와 텍스트 프롬프트를 API 엔드포인트로 전송하면, 시스템은 다음과 같은 순서로 처리합니다:

1. 이미지 파일을 수신하고 유효성을 검사
2. OpenCV를 사용하여 이미지 내 도형들을 분석
3. 분석 결과를 구조화된 데이터로 변환
4. 이미지와 OpenCV 결과를 ChatGPT API에 전달
5. ChatGPT의 자연어 설명과 OpenCV 분석 결과를 결합하여 응답

2. 1단계: Flask 애플리케이션 및 의존성 설정

2.1 프로젝트 구성

먼저 프로젝트를 위한 전용 디렉터리를 생성하고 가상 환경을 설정합니다:

```
mkdir flask_opencv_agent
cd flask_opencv_agent
python -m venv venv
```

가상 환경 활성화:

Windows:

```
.\venv\Scripts\activate
```

macOS/Linux:

```
source venv/bin/activate
```

2.2 필요한 라이브러리 설치

가상 환경이 활성화된 상태에서 필요한 패키지들을 설치합니다:

```
pip install Flask opencv-python numpy openai
```

2.3 기본 Flask 애플리케이션 구성

app.py 파일을 생성하고 기본 Flask 구조를 설정합니다:

```
from flask import Flask, request, jsonify
import cv2
import numpy as np
import base64
import json
import io
from openai import OpenAI

# Flask 애플리케이션 초기화
app = Flask(__name__)

# OpenAI 클라이언트 (나중에 초기화)
client = None

@app.route('/')
def home():
    return "Flask-based OpenCV Agent is running!"

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
    print("Flask app is running on http://0.0.0.0:5000")
```

> **테스트:** 애플리케이션을 실행하고 브라우저에서 http://0.0.0.0:5000에 접속하여 정상 작동을 확인합니다.

3. 2단계: OpenCV 이미지 분석 함수 구현

3.1 OpenCV 분석 함수 개발

이미지 내의 도형들을 찾고 분석하는 핵심 함수를 구현합니다:

```
def analyze_image_with_opencv(image_np):
    """
    OpenCV를 사용하여 이미지 내의 도형들을 찾고 외접 사각형 정보를 반환합니다.

    Args:
        image_np (numpy.ndarray): OpenCV로 로드된 이미지 (BGR 포맷)

    Returns:
        list: 각 도형의 정보를 담은 딕셔너리 리스트
    """
    # 그레이스케일 변환
    gray = cv2.cvtColor(image_np, cv2.COLOR_BGR2GRAY)

    # 이진화 처리
    _, thresh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY_INV)

    # 윤곽선 찾기
    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    shapes_data = []
    img_height, img_width = image_np.shape[:2]

    # 가장 큰 윤곽선 찾기 (메인 컨테이너)
    largest_contour_area = 0
    largest_contour_idx = -1
    if contours:
        for i, cnt in enumerate(contours):
            area = cv2.contourArea(cnt)
            if hierarchy[0][i][3] == -1 and area > largest_contour_area:
                largest_contour_area = area
                largest_contour_idx = i

    # 각 윤곽선 분석
    for i, contour in enumerate(contours):
        area = cv2.contourArea(contour)

        # 노이즈 필터링
        if area < 100:
            continue

        # 외접 사각형 계산
        x, y, w, h = cv2.boundingRect(contour)

        # 도형 타입 추정
        shape_type = "unknown"
        aspect_ratio = float(w) / h

        if 0.85 <= aspect_ratio <= 1.15:
            if area > 0:
                circularity = (4 * np.pi * area) / (cv2.arcLength(contour, True)**2)
                if circularity > 0.7:
                    shape_type = "circle"
                else:
                    shape_type = "square"
        elif aspect_ratio < 0.85 or aspect_ratio > 1.15:
            shape_type = "oval" if cv2.isContourConvex(contour) and cv2.arcLength(contour,
True) > (w+h)*1.5 else "rectangle"

        # 중심점 계산
        center_x = x + w // 2
        center_y = y + h // 2

        # 상대적 중심 좌표
        relative_center_x = center_x / img_width
        relative_center_y = center_y / img_height

        # 역할 결정
        if hierarchy.size > 0 and i == largest_contour_idx:
            role = "main_container"
        elif hierarchy.size > 0 and hierarchy[0][i][3] != -1:
            role = "inner_shape"
        else:
            role = "outer_shape"

        shapes_data.append({
            "id": len(shapes_data) + 1,
            "type": shape_type,
            "role": role,
            "bounding_box_pixel": {"x": x, "y": y, "width": w, "height": h},
            "center_coordinates_pixel": {"x": center_x, "y": center_y},
            "center_coordinates_relative": {"x": round(relative_center_x, 4), "y":
round(relative_center_y, 4)},
            "area_pixels": round(area, 2)
        })

    return shapes_data
```

3.2 주요 OpenCV 함수 설명

- `cv2.cvtColor()` : 컬러 이미지를 그레이스케일로 변환
- `cv2.threshold()` : 이진화 처리로 배경과 객체 분리
- `cv2.findContours()` : 윤곽선 검출 및 계층 구조 분석
- `cv2.contourArea()` : 윤곽선의 면적 계산
- `cv2.boundingRect()` : 외접 사각형 좌표 계산

4. 3단계: ChatGPT API 클라이언트 설정

4.1 OpenAI API 키 설정

OpenAI 플랫폼에서 API 키를 발급받고 환경 변수로 설정합니다:

```
import os
from openai import OpenAI

# OpenAI 클라이언트 초기화
try:
    OPENAI_API_KEY = os.environ.get("OPENAI_API_KEY")
    if not OPENAI_API_KEY:
        raise ValueError("OPENAI_API_KEY environment variable not set.")
    openai_client = OpenAI(api_key=OPENAI_API_KEY)
    print("OpenAI client initialized using environment variable.")
except Exception as e:
    app.logger.error(f"Error initializing OpenAI client: {e}")
    openai_client = None

CHATGPT_MODEL = "gpt-4o"  # 비전 지원 모델
```

4.2 환경 변수 설정 방법

Windows (Command Prompt):

```
set OPENAI_API_KEY="your-api-key-here"
```

Windows (PowerShell):

```
$env:OPENAI_API_KEY="your-api-key-here"
```

macOS/Linux:

```
export OPENAI_API_KEY="your-api-key-here"
```

> **보안 주의사항:** API 키는 절대 코드에 직접 하드코딩하지 마세요. 반드시 환경 변수나 보안 저장소를 사용하세요.

5. 4단계: API 엔드포인트 구현

5.1 이미지 업로드 처리

Flask 라우트에서 이미지 파일과 텍스트 프롬프트를 받아 처리하는 로직을 구현합니다:

```
@app.route('/analyze_image', methods=['POST'])
def analyze_image():
    # OpenAI 클라이언트 확인
    if openai_client is None:
        return jsonify({"error": "OpenAI client is not initialized."}), 500

    # 요청 데이터 검증
    if 'image' not in request.files:
        return jsonify({"error": "Image file not found in 'image' field."}), 400

    image_file = request.files['image']
    text_prompt = request.form.get('prompt', "Describe the shapes found in the image.")

    if image_file.filename == '':
        return jsonify({"error": "No selected image file."}), 400

    try:
        # 이미지 디코딩
        image_bytes = image_file.read()
        image_np = np.frombuffer(image_bytes, np.uint8)
        img_decoded = cv2.imdecode(image_np, cv2.IMREAD_COLOR)

        if img_decoded is None:
            return jsonify({"error": "Could not decode image file."}), 400

        # OpenCV 분석 수행
        opencv_analysis_results = analyze_image_with_opencv(img_decoded)

        # 여기서 OpenCV 결과만 반환 (5단계에서 ChatGPT 통합)
        return jsonify({
            "status": "OpenCV analysis successful",
            "opencv_analysis": opencv_analysis_results,
            "received_prompt": text_prompt
        })

    except Exception as e:
        app.logger.error(f"Error during image analysis: {e}")
        return jsonify({"error": f"An error occurred: {str(e)}"}), 500
```

5.2 요청 데이터 처리

- `request.files['image']` : 업로드된 이미지 파일 접근
- `request.form.get('prompt')` : 텍스트 프롬프트 추출
- `np.frombuffer()` : 바이트 데이터를 NumPy 배열로 변환
- `cv2.imdecode()` : 메모리 버퍼에서 이미지 디코딩

6. 5단계: OpenCV와 ChatGPT 완전 통합

6.1 최종 분석 함수 구현

OpenCV 분석 결과와 원본 이미지를 ChatGPT API로 전달하여 종합적인 설명을 생성합니다:

```
@app.route('/analyze_image', methods=['POST'])
def analyze_image():
    if openai_client is None:
        return jsonify({"error": "OpenAI client is not initialized."}), 500

    if 'image' not in request.files:
        return jsonify({"error": "Image file not found."}), 400

    image_file = request.files['image']
    text_prompt = request.form.get('prompt', "Describe the shapes found in the image.")

    if image_file.filename == '':
        return jsonify({"error": "No selected image file."}), 400

    try:
        # 이미지 처리
        image_bytes = image_file.read()
        image_np = np.frombuffer(image_bytes, np.uint8)
        img_decoded = cv2.imdecode(image_np, cv2.IMREAD_COLOR)

        if img_decoded is None:
            return jsonify({"error": "Could not decode image file."}), 400

        # OpenCV 분석
        opencv_analysis_results = analyze_image_with_opencv(img_decoded)

        # ChatGPT용 Base64 인코딩
        _, encoded_image = cv2.imencode('.png', img_decoded)
        base64_image = base64.b64encode(encoded_image.tobytes()).decode('utf-8')

        # ChatGPT 메시지 구성
        messages = [
            {
                "role": "system",
                "content": """당신은 고급 이미지 분석 AI입니다. 기하학적 도형에 대한 상세한 설명을 제공하는 것
이 당신의 임무입니다.
                이미지와 OpenCV 에이전트의 정밀한 기하학적 분석 결과를 함께 받게 됩니다.
                이미지의 시각적 정보와 제공된 OpenCV 데이터를 결합하여 사용자의 요청에 포괄적으로 답변하세요."""
            }
        ]

        # 사용자 메시지 구성
        user_message_content = [
            {"type": "text", "text": f"사용자 요청: {text_prompt}"},
            {"type": "image_url", "image_url": {"url": f"data:image/png;base64,
{base64_image}"}}
        ]

        # OpenCV 분석 결과 추가
        opencv_info_str = "\nOpenCV 분석 결과:\n"
        if opencv_analysis_results:
            for shape in opencv_analysis_results:
                opencv_info_str += (
                    f"- ID: {shape['id']}, 타입: {shape['type']}, 역할: {shape['role']}, "
                    f"경계 상자: (x={shape['bounding_box_pixel']['x']}, y=
{shape['bounding_box_pixel']['y']}, "
                    f"w={shape['bounding_box_pixel']['width']}, h={shape['bounding_box_pixel']
['height']}), "
                    f"중심점: (x={shape['center_coordinates_relative']['x']}, y=
{shape['center_coordinates_relative']['y']}), "
                    f"면적: {shape['area_pixels']} pixels²\n"
                )
        else:
            opencv_info_str += "OpenCV가 도형을 찾지 못했습니다.\n"

        user_message_content.append({"type": "text", "text": opencv_info_str})

        messages.append({
            "role": "user",
            "content": user_message_content
        })

        # ChatGPT API 호출
        chatgpt_response = openai_client.chat.completions.create(
            model=CHATGPT_MODEL,
            messages=messages,
            max_tokens=1000
        )

        # 응답 처리
        if chatgpt_response.choices and chatgpt_response.choices[0].message.content:
            chatgpt_text_response = chatgpt_response.choices[0].message.content

            final_response = {
                "status": "success",
                "opencv_analysis": opencv_analysis_results,
                "chatgpt_explanation": chatgpt_text_response
            }
            return jsonify(final_response)
        else:
            return jsonify({"error": "ChatGPT에서 유효한 응답을 받지 못했습니다."}), 500

    except Exception as e:
        app.logger.error(f"Error during analysis: {e}")
        return jsonify({"error": f"내부 서버 오류: {str(e)}"}), 500
```

6.2 멀티모달 메시지 구성

ChatGPT API의 멀티모달 기능을 활용하여 텍스트와 이미지를 함께 전달합니다:

- **시스템 메시지**: AI의 역할과 행동 방식을 정의
- **사용자 메시지**: 텍스트 프롬프트와 Base64 인코딩된 이미지 포함
- **OpenCV 데이터**: 구조화된 분석 결과를 텍스트로 변환하여 추가

7. 애플리케이션 테스트

7.1 curl을 사용한 테스트

명령줄에서 curl을 사용하여 API를 테스트합니다:

Windows PowerShell:

```
& "C:\Windows\System32\curl.exe" -X POST -F "image=@example.png" -F "prompt=이미지의 도형들을 상세히
분석해주세요." http://127.0.0.1:5000/analyze_image
```

Windows Command Prompt / macOS / Linux:

```
curl -X POST -F "image=@example.png" -F "prompt=이미지의 도형들을 상세히 분석해주세요."
http://127.0.0.1:5000/analyze_image
```

7.2 Python requests를 사용한 클라이언트

Python 스크립트로 API를 테스트하는 방법:

```
import requests
import json
import os

# 설정
API_URL = "http://127.0.0.1:5000/analyze_image"
IMAGE_PATH = "example.png"
TEXT_PROMPT = "이미지의 도형들을 상세히 분석해주세요."

# 이미지 파일 존재 확인
if not os.path.exists(IMAGE_PATH):
    print(f"오류: 이미지 파일을 찾을 수 없습니다: {IMAGE_PATH}")
    exit()

try:
    # 요청 전송
    with open(IMAGE_PATH, 'rb') as img_file:
        files = {
            'image': (os.path.basename(IMAGE_PATH), img_file, 'image/png')
        }
        data = {
            'prompt': TEXT_PROMPT
        }

        print(f"요청 전송 중: {API_URL}")
        response = requests.post(API_URL, files=files, data=data)

        # 응답 처리
        response.raise_for_status()
        response_json = response.json()

        print("\n--- Flask 에이전트 응답 ---")
        print(json.dumps(response_json, indent=2, ensure_ascii=False))

except requests.exceptions.ConnectionError as e:
    print(f"연결 오류: Flask 앱이 {API_URL}에서 실행 중인지 확인하세요.")
except requests.exceptions.HTTPError as e:
    print(f"HTTP 오류: {response.status_code}")
    print(f"상세 내용: {response.text}")
except json.JSONDecodeError:
    print("JSON 디코딩 오류")
    print(f"원본 응답: {response.text}")
except Exception as e:
    print(f"예상치 못한 오류: {e}")
```

7.3 예상 응답 형식

성공적인 요청에 대한 응답 예시:

```
{
  "status": "success",
  "opencv_analysis": [
    {
      "id": 1,
      "type": "square",
      "role": "main_container",
      "bounding_box_pixel": {
        "x": 50,
        "y": 50,
        "width": 200,
        "height": 200
      },
      "center_coordinates_pixel": {
        "x": 150,
        "y": 150
      },
      "center_coordinates_relative": {
        "x": 0.5,
        "y": 0.5
      },
      "area_pixels": 40000.0
    }
  ],
  "chatgpt_explanation": "이미지에는 중앙에 위치한 큰 정사각형이 있습니다. 이 정사각형은 주요 컨테이너 역할을 하
며, 픽셀 좌표 (50, 50)에서 시작하여 200x200 크기를 가집니다. 중심점은 이미지의 정중앙인 (0.5, 0.5) 상대 좌표에
위치합니다."
}
```

8. 결론 및 향후 개선사항

8.1 구현 완료 사항

- Flask 기반 웹 API 서버 구축
- OpenCV를 활용한 이미지 내 도형 분석 기능
- ChatGPT API 통합으로 자연어 설명 생성
- 멀티모달 입력 처리 (이미지 + 텍스트)
- 구조화된 JSON 응답 제공

8.2 향후 개선사항

- **도형 인식 정확도 향상**: 더 정교한 OpenCV 알고리즘 적용
- **웹 인터페이스 개발**: 사용자 친화적인 HTML 폼 제공
- **배포 및 확장성**: 클라우드 플랫폼 배포 및 로드 밸런싱
- **보안 강화**: API 키 관리 및 요청 제한 기능
- **성능 최적화**: 이미지 처리 속도 향상 및 메모리 사용량 최적화

8.3 활용 가능 분야

- 교육용 기하학 도형 분석 도구
- 산업용 품질 검사 시스템
- 의료 영상 분석 보조 도구
- 건축 및 설계 도면 분석

> **참고:** 이 가이드는 기본적인 구현 방법을 제시합니다. 실제 운영 환경에서는 추가적인 보안, 성능, 확장성 고려사항이 필요합니다.