

문서 스캐너 및 자동 기울기 보정 프로젝트 계획서

21113969 컴퓨터공학전공 정솔주

과제명	국문	문서 스캐너 및 자동 기울기 보정
	영문	Document scanner and auto-slope calibration

1. 프로젝트 문제 정의

1-1. 책 인식 불가 문제

: 책의 제본선 부분을 기준으로 명확하지 않은 명암도 때문에 책의 경계선을 인식하지 못함

: 이 문제는 책에서 종이 형태로 된 문서 이미지 데이터로 대상을 변경하였음

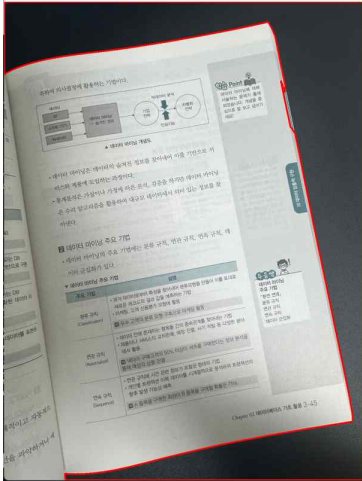


그림 1 책 경계선 인식 불가

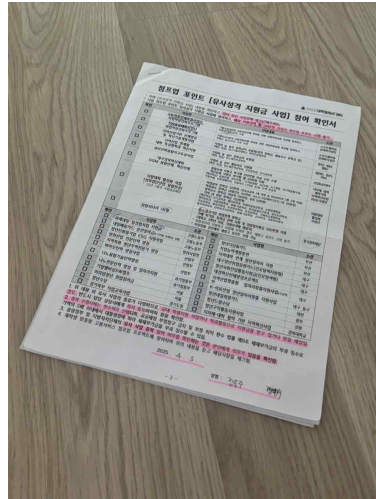


그림 2 문서 이미지 데이터로 변경

1-2. 컨투어 검출 어려

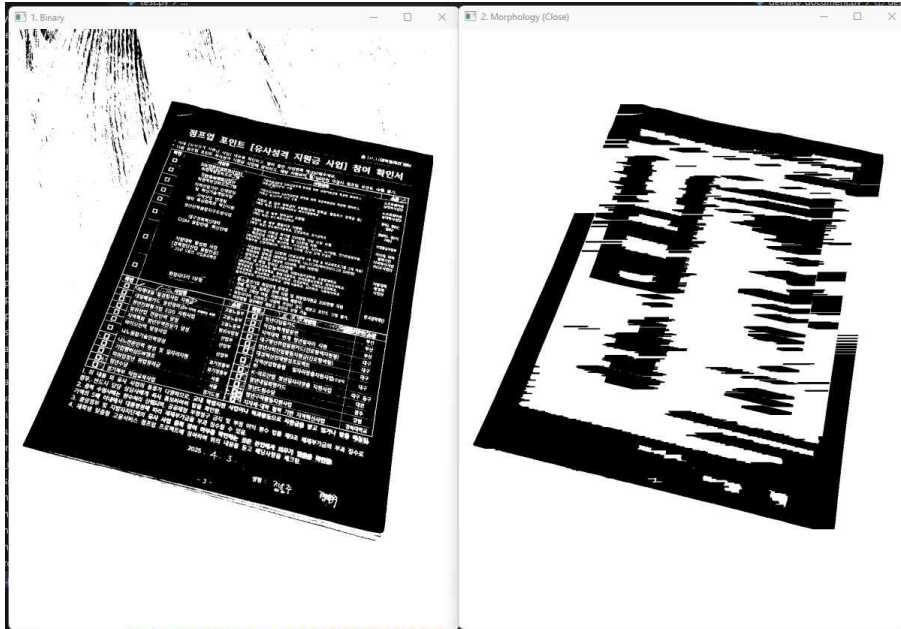
: 이미지 데이터의 꼭짓점은 인식하지 못해서 투시 변환이 이루어지지 않는 문제



그림 3 코드 수정 전 컨투어 추출 어려

2. 문제 원인 분석 (컨투어 추출 에러)

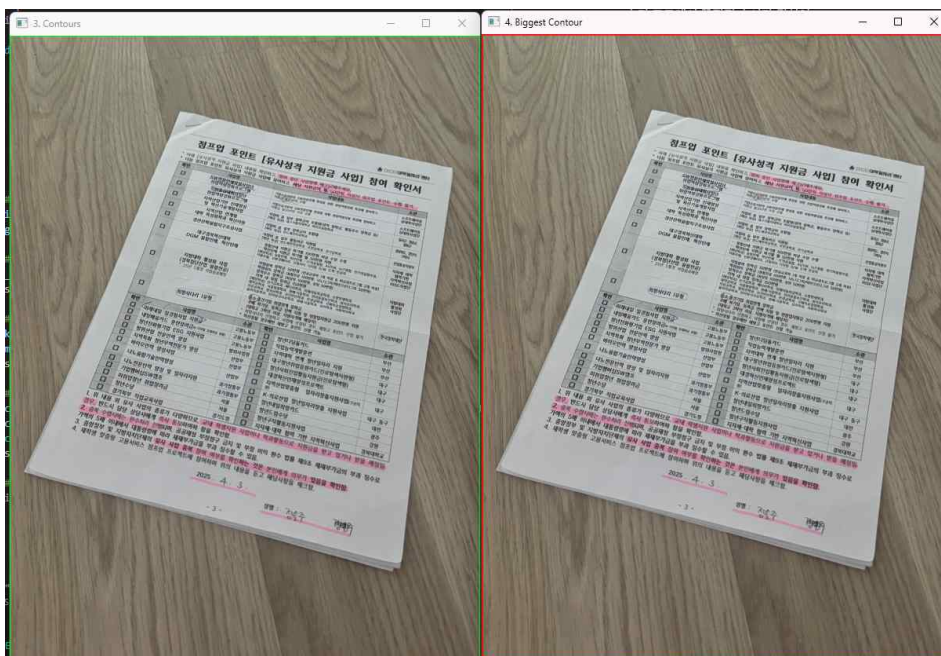
2-1. 원인 분석 과정



- 모폴로지 연산을 적용하였을 경우, 문서가 세로 방향 부분이 없어진 현상이 나타남

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (50, 3))
```

- 수정 전 코드를 보면 가로가 길고, 세로가 짧은 타원형 커널(50, 3)로 인해 가로로만 강하게 연결되고 세로 방향은 무시되는 현상이 발생



- (50, 3) 크기의 타원형 때문에 외곽선을 추출하여도 외곽선을 찾을 수 없는 현상이 발생함

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
```

- 이후 (3,3)으로 모폴로지 연산의 범위를 수정하여 적용해봄



그림 8 수정된 모폴로지 연산 적용 결과

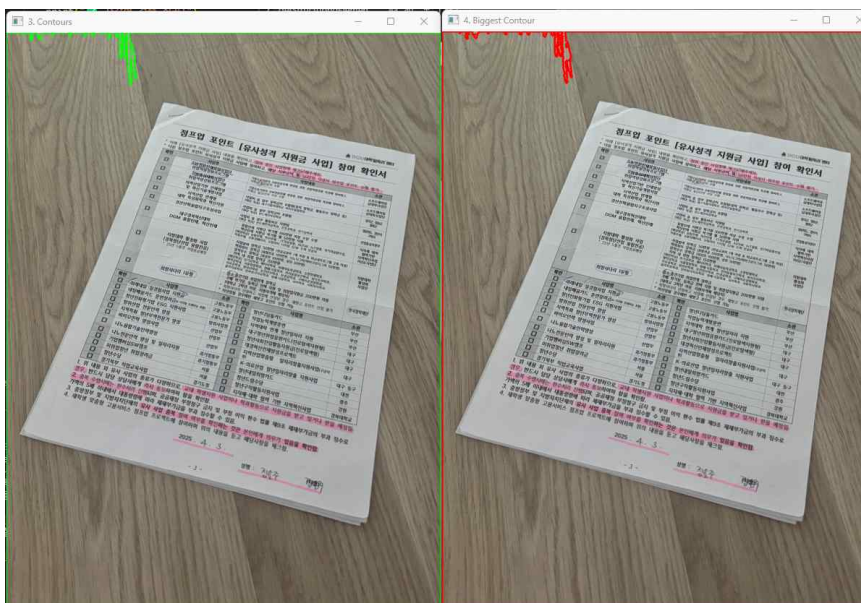


그림 9 모폴로지 연산 후 외곽선 추출 결과

- 수정된 모폴로지 연산을 적용한 결과로 정상적으로 이미지의 배경과 문서가 분리된 것을 확인할 수 있으나, 이미지에서는 여전히 문서의 외곽선을 추출할 수 없음

- 외곽선이 추출되지 않는 이유는 이진화, 모폴로지 연산을 적용하였을 때 문서 영역은 검은색, 배경은 흰색으로 처리되었는데, 이는 OpenCV의 컨투어 검출 함수가 흰색 영역에서 외곽선을 찾는 동작 방식과 이진화 결과의 픽셀값 배치가 일치하지 않아서 문서의 외곽선이 검출되지 않음

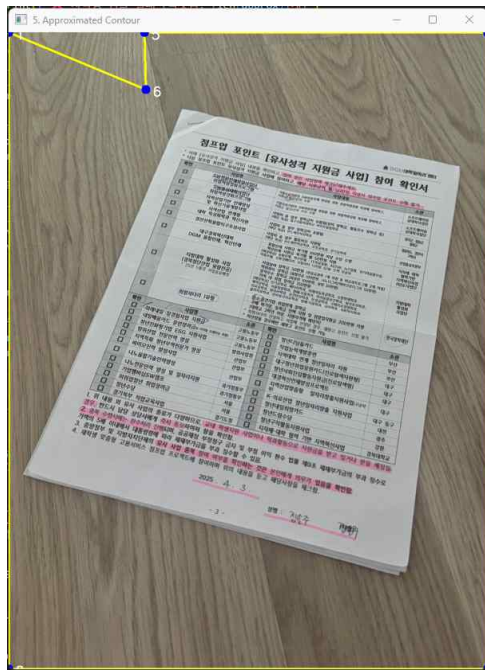


그림 10 꼭짓점 추출

근사화된 꼭짓점 좌표:

점 1:	(0, 0)
점 2:	(0, 1439)
점 3:	(1079, 1439)
점 4:	(1079, 0)
점 5:	(306, 0)
점 6:	(309, 129)

그림 11 근사화된 꼭짓점 좌표

- 수정된 모폴로지 연산을 적용한 후 여러 개의 꼭짓점이 검출되는데, 이는 꼭짓점 추출 과정에서 사용된 Epsilon(2%로 설정) 값이 작아서 미세한 굴곡까지 꼭짓점으로 인식되었음

2-2. 원인 정리

- 컨투어 검출 함수의 동작 방식에 적합하지 않은 이진화
: 흰색 영역에서 외곽선을 찾는 OpenCV 컨투어 검출 함수의 동작 방식과 문서 영역이 검은색, 배경이 흰색으로 처리된 이진화 결과의 픽셀값 배치가 맞지 않아서 문서의 외곽선이 검출되지 않음
- 낮은 Epsilon 값
: Epsilon 값을 2%로 설정하여 미세한 굴곡까지 꼭짓점으로 인식함

3. 문제 해결 방법

3-1. 전처리 강화

- 이진화 문제를 해결하기 위해 적용함

3-1-1. CLAHE(적응적 히스토그램 평활화)

: 이미지를 여러 작은 타일(8x8)로 나누고, 각 타일별로 히스토그램 평활화를 적용하는 방식

- 어두운 영역의 디테일을 보존하고, 그림자와 반사를 제거하기 위해 사용

3-1-2. OTSU 방식의 전역 이진화

: 이미지 전체를 하나의 임계값으로 자동 계산한 후, 모든 픽셀에 동일한 임계값을 적용해 이진화함

- 문서와 배경을 명확히 분리하기 위해 사용

```
def preprocess_image(image):
    """이미지 전처리"""
    # Gray 스케일 변환
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # 적응적 히스토그램 평활화 (대비 향상)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    gray = clahe.apply(gray)

    # 블러 처리
    gray = cv2.GaussianBlur(gray, (5, 5), 0)

    # OTSU 방식의 전역 이진화
    _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    return gray, binary
```

그림 12 CLAHE, OTSU 방식의 전역 이진화 적용한 코드

3-2. 점 근사화

3-2-1. Epsilon 수치 조절

: 원본 컨투어와 근사화된 컨투어 사이의 최대 허용 거리인 Epsilon 수치를 조절함

: Epsilon을 3%로 수정하여 직선화된 형태로 더 적은 꼭짓점(4개)을 검출

```
def find_rectangle_contour(contours):
    """사각형 외곽선 찾기"""
    for i, c in enumerate(contours):
        area = cv2.contourArea(c)
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.03 * peri, True)

        print(f"외곽선 {i}: 면적={area:.0f}, 꼭짓점={len(approx)}")

        if len(approx) == 4:
            print("사각형 문서 영역을 찾았습니다")
            return approx

    return None
```

그림 13 Epsilon을 3%로 조절

4. 개발 과정

4-1. 데이터 수집

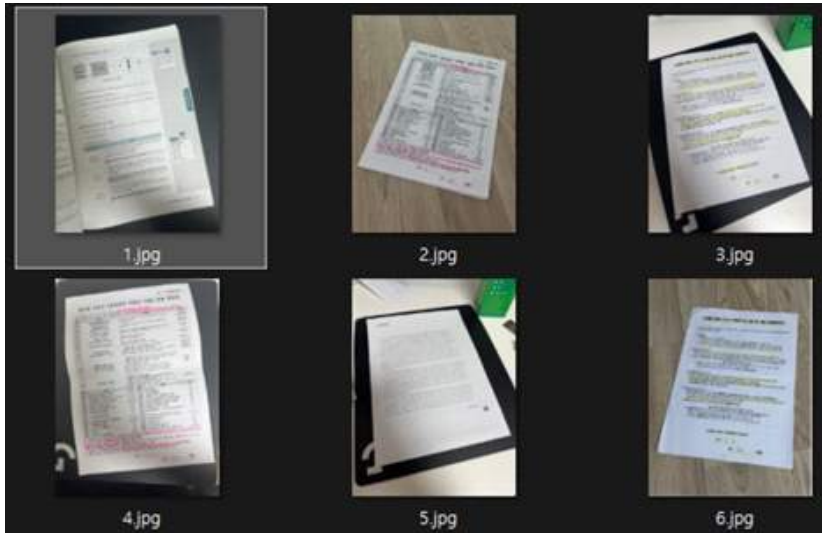


그림 14 수집한 데이터 형식

- 책의 제본선 부분을 인식하지 못하는 문제로 종이로된 문서 형식으로 데이터를 변경
- 여러 배경으로 문서를 직접 찍어서 이미지 데이터를 수집

4-2. 함수 및 알고리즘

- 투시 변환에 사용된 단계적 과정

원본 이미지 → 그레이스케일 → CLAHE → 블러 → OTSU 방식의 전역 이진화 → Canny 에지 → 모폴로지 닫힘(3x3) → 컨투어 검출 → 면적 필터링 → 사각형 근사화 → 최종 반환

4-2-1. CLAHE 적용

- 이미지를 8x8 타일로 분할 후 각 타일별로 히스토그램을 균일화
- 지역적 대비를 개선하기 위해 사용

4-2-2. OTSU 방식의 전역 이진화

- 문서, 배경을 명확하게 분리하기 위해 사용

```
def preprocess_image(image):
    """이미지 전처리"""
    # Gray 스케일 변환
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # 적응적 히스토그램 평활화 (대비 향상)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    gray = clahe.apply(gray)

    # 블러 처리
    gray = cv2.GaussianBlur(gray, (5, 5), 0)

    # OTSU 방식의 전역 이진화
    _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    return gray, binary
```

그림 15 CLAHE, OTSU 방식의 전역 이진화 적용한 코드

4-2-3. Canny 에지 검출

- 임계값 30, 100을 이용해서 약한 에지는 무시하고, 강한 에지만 남김
- 문서의 경계선을 추출하기 위해 적용

4-2-4. 모폴로지 닫힘 연산 적용

- 3x3 크기의 커널로 모폴로지 닫힘 연산을 적용
- 끊어진 외곽선을 연결하기 위해 사용

```
def detect_edges(binary):
    """에지 검출"""

    # Canny 에지 검출
    edged = cv2.Canny(binary, 30, 100)

    kernel = np.ones((3,3), np.uint8)
    edged = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel) # 모폴로지 닫힘 연산

    return edged
```

그림 16 Canny 에지 검출, 모폴로지 닫힘 연산 적용

4-2-5. 컨투어 검출

- 모든 외곽선을 검출함

```
def find_contours(edged):
    """외곽선 찾기"""
    cnts, _ = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    return cnts
```

그림 17 컨투어 검출

4-2-6. 컨투어 필터링

- 면적, 볼록성, 위치를 기반으로 문서 외곽선만 선택함

```
def filter_document_contours(cnts, img_shape, min_area_ratio=0.05):  
    """문서 외곽선 필터링"""  
    img_area = img_shape[0] * img_shape[1]  
    min_area = img_area * min_area_ratio  
  
    # 면적순으로 정렬  
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:10]  
  
    valid_contours = []  
    for c in cnts:  
        area = cv2.contourArea(c)  
        if area >= min_area:  
            valid_contours.append(c)  
  
    return valid_contours
```

그림 18 컨투어 필터링

4-2-7. 사각형 근사화

- 2%에서 3%로 Epsilon 값을 수정하여 4개의 꼭짓점을 검출하도록 설정

```
def find_rectangle_contour(contours):  
    """사각형 외곽선 찾기"""  
    for i, c in enumerate(contours):  
        area = cv2.contourArea(c)  
        peri = cv2.arcLength(c, True)  
        approx = cv2.approxPolyDP(c, 0.03 * peri, True)  
  
        print(f"외곽선 {i}: 면적={area:.0f}, 꼭짓점={len(approx)}")  
  
        if len(approx) == 4:  
            print("사각형 문서 영역을 찾았습니다")  
            return approx  
  
    return None
```

그림 19 사각형 근사화

4-2-8. 투시 변환

- cv2.warpPerspective 함수를 사용하여 투시 변환 적용

```
M = cv2.getPerspectiveTransform(rect, dst)  
warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight),  
                             flags=cv2.INTER_LANCZOS4)  
  
# 화질 개선 샤프닝  
warped = cv2.detailEnhance(warped, sigma_s=10, sigma_r=0.15)  
  
return warped
```

그림 20 투시 변환

5. 프로젝트 결과

- 정상적으로 투시 변환 적용

: 기울어진 각도의 문서를 정면에서 보는 것과 동일하게 투시 변화가 적용됨

- 문서에만 적용 가능한 투시 변환

: 책의 제본선 부분을 인식하지 못하기 때문에 문서에서만 적용이 가능함

- OCR 적용

: 이전 결과물보다 한국어 인식률이 더 높아졌으나, 여전히 인식을 하지 못하는 부분이 존재함

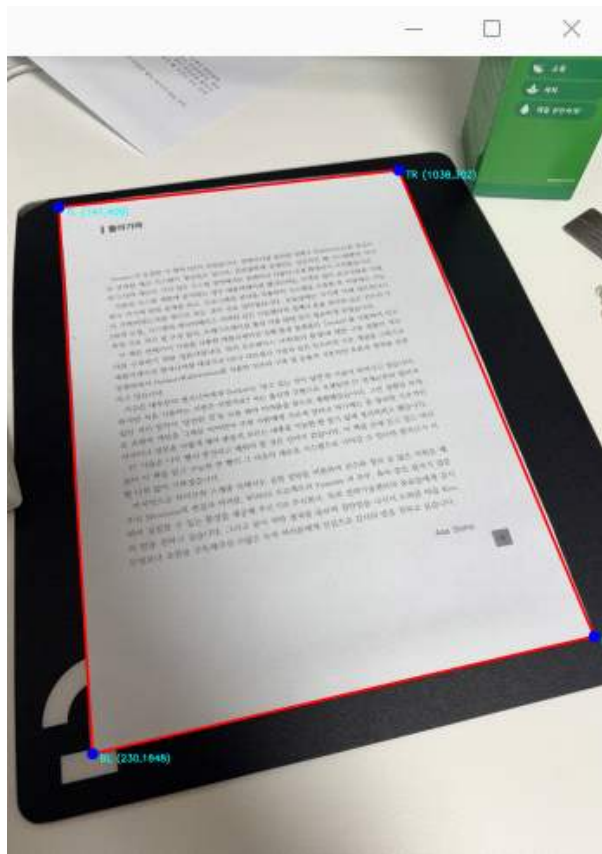


그림 21 문서의 꼭짓점 정상적으로 추출됨

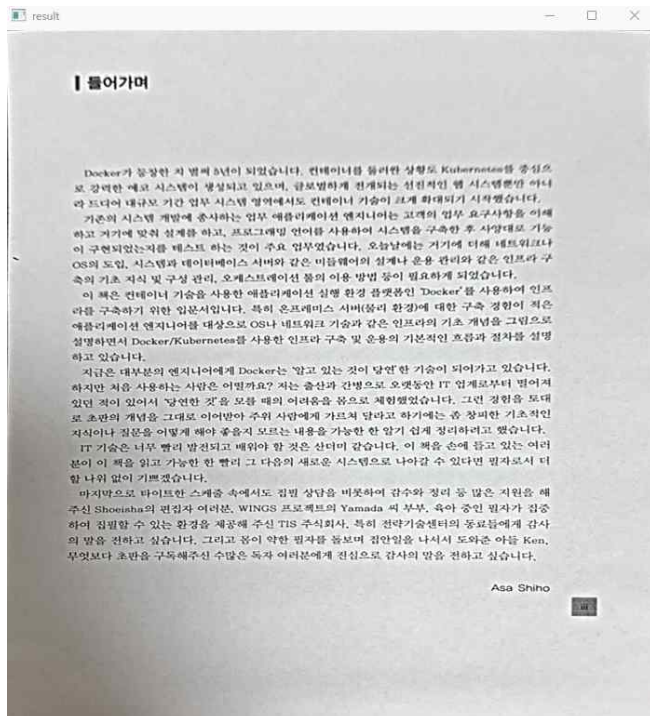


그림 22 투시 변환이 정상적으로 적용된 모습

```

output > ㉔ scanned_result_ocr.txt
1 | 들어가며
2
3 Docker7) WR 지 벌써 6년이 되었습니다. 컨테이너를 올린 ADA Kubernetett 244
4 로 강력한 예보 AAO) BASAL 있으며, 글로벌하게 전개되는 선진적인 Ae 아디
5 라 드디어 대규모 FRE 업무 시스템 영역에서도 컨테이너 기술이 Atm 확대되기 시작였습니다.
6 ASO 시스템 AVION 총사하는 업무 애플리케이션 엔지니어는 고객의 임무 요구사항을 이해
7 하고 거기에 맞춰 설계를 하고, 프로그래밍 언어를 사용하여 시스템을 PB FAVRE 기능
8 이 구현되었는지를 테스트 하는 것이 주요 업무였습니다, 오늘날에는 거기에 더해 네트워크다
9 3 OSS) 도입. AAW 데이터베이스 서버와 같은 미들웨어의 설계나 운용 관리와 같은 Cet
10 측의 기초 지식 및 구성 관리, 오케스트레이션 불의 이용 방법 Sol 필요하게 되었습니다.
11
12 이 책은 컨테이너 기술을 사용한 애플리케이션 실행 환경 TAWA Docker' 를 사용하여 인프
13 라를 구축하기 위한 입문서입니다. 특히 온프레미스 서버(물리 환경)에 대한 구축 DAO 적은
14 애플리케이션 엔지니어를 대상으로 05나 네트워크 기술과 같은 인프라의 기초 개념을 그림으로
15 설명하면서 Docker/Kubernetes st 사용한 인프라 구축 및 운용의 기본적인 흐름과 절차를 설명
16 하고 있습니다.
17
18 지금은 대부분의 엔지니어에게 000<@는 “알고 있는 것이 당연”한 기술이 되어가고 있습니다.
19 하지만 처음 사용하는 사람은 어떨까요? 저는 출산과 간병으로 오랫동안 IT 업계로부터 떨어져
20 적이 있어서 당연한 wv 모를 때의 ASA 몸으로 체험했었습니다. 그런 경험을 토대
21 as 개념을 그대로 이어받아 주위 사람에게 가르쳐 달라고 하기에는 좀 창피한 기초적인
22 이나 질문을 어떻게 해야 좋을지 모르는 내용을 가능한 한 알기 쉽게 정리하려고 했습니다.
23 “lene 너무 빨리 발전되고 배워야 할 것은 산더미 같습니다. 이 책을 손에 들고 있는 여려
24
25 [지막으로 타이트한 스케줄 속에서도 집필 상담을 비롯하여 감수와 정리 등 많은 지원을 해
26 1 St hae) 편집자 여러분. WINGS 프로젝트의 Yamada 씨 부부, 육아 Se1 필자가 WS
27
28 고 싶습니다. 그리고 몸이 약한 필자를 돌보며 집안일을 나서서 도와준 아들 Ken.
29 구독해주신 수많은 독자 여러분에게 진심으로 감사의 말을 전하고 싶습니다.
30
31 Asa Shiho

```

그림 23 OCR을 통해 이미지에서 텍스트 추출 결과값

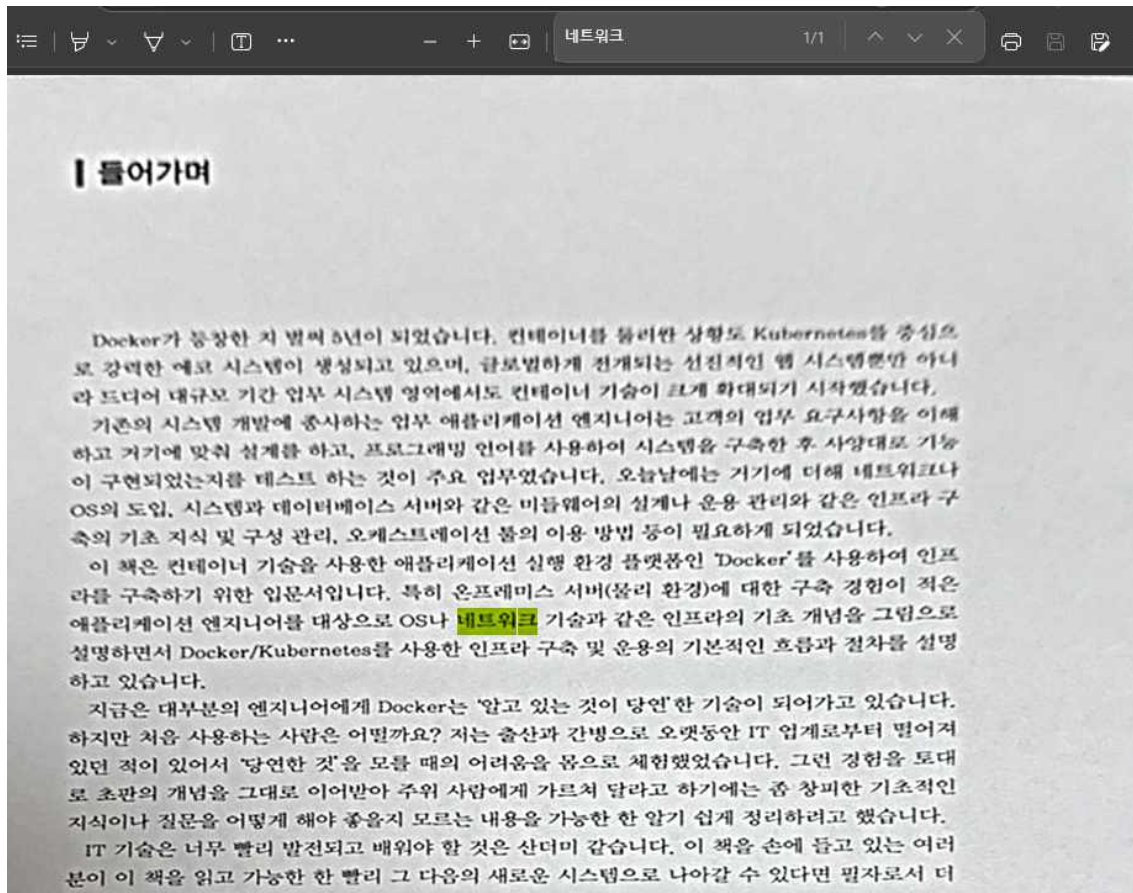


그림 24 텍스트 검색이 가능한 모습

GitHub 주소

<https://github.com/grovince/ComputerVision.git>