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

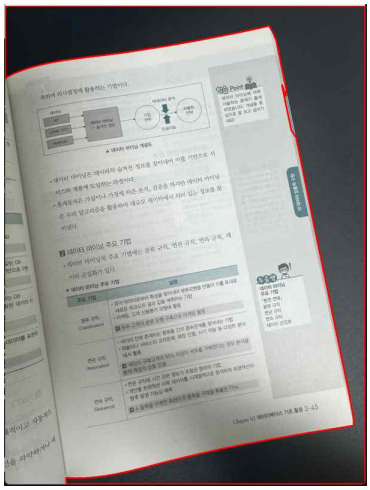


그림 1 책 경계선 인식 불가

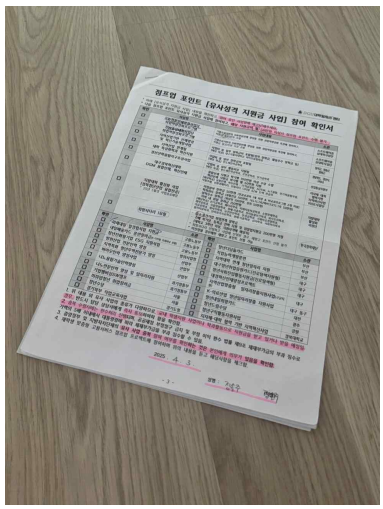


그림 2 문서 이미지 데이터로 변경

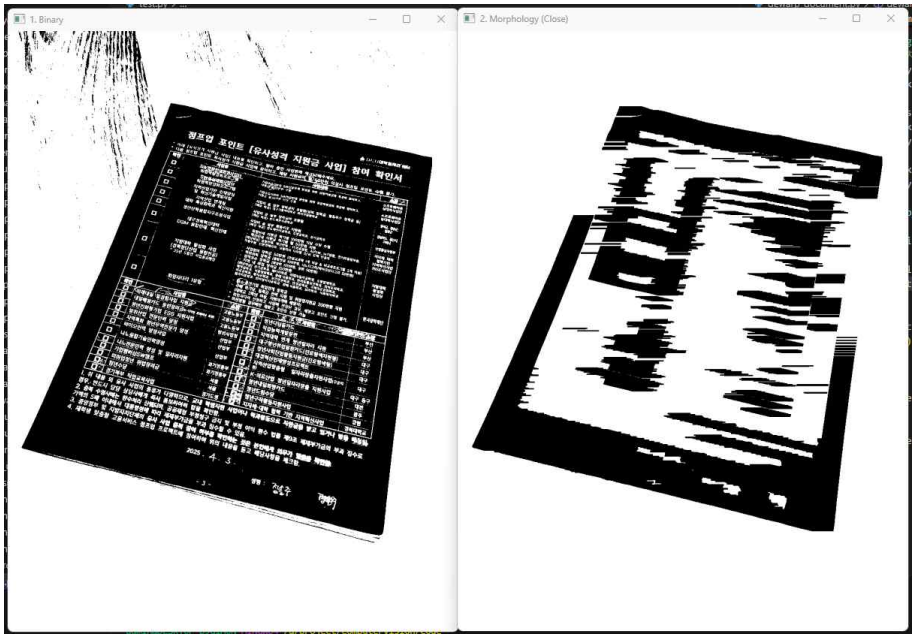
1-2. 컨투어 검출 에러

: 이미지 데이터의 꼭짓점은 인식하지 못해서 투시 변환이 이루어지지 않는 문제

그림 3 코드 수정 전 컨투어 추출 에러

2. 문제 원인 분석 (컨투어 추출 에러)

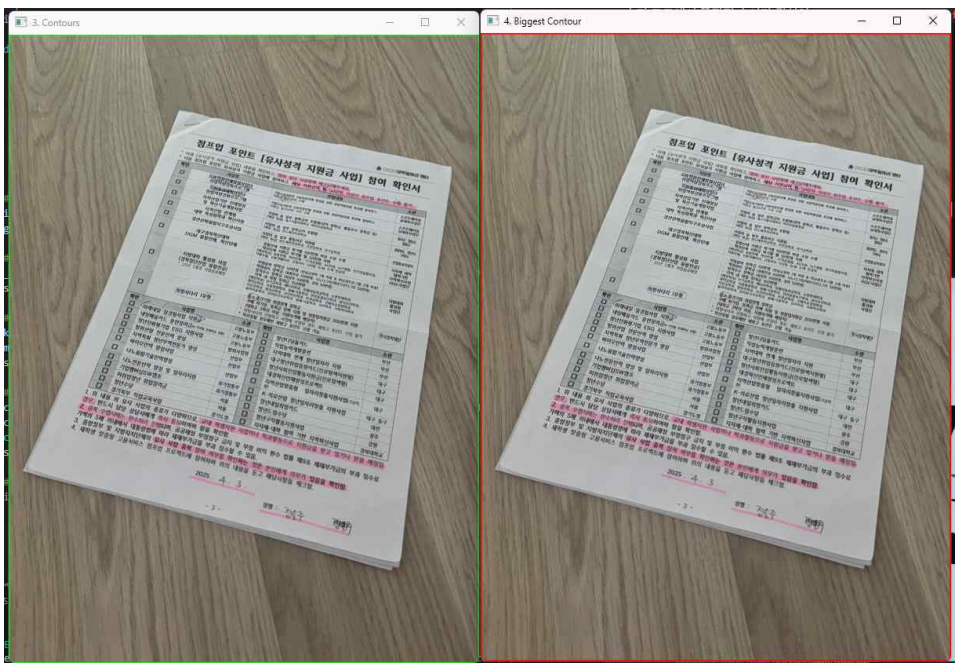
2-1. 원인 분석 과정



- 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

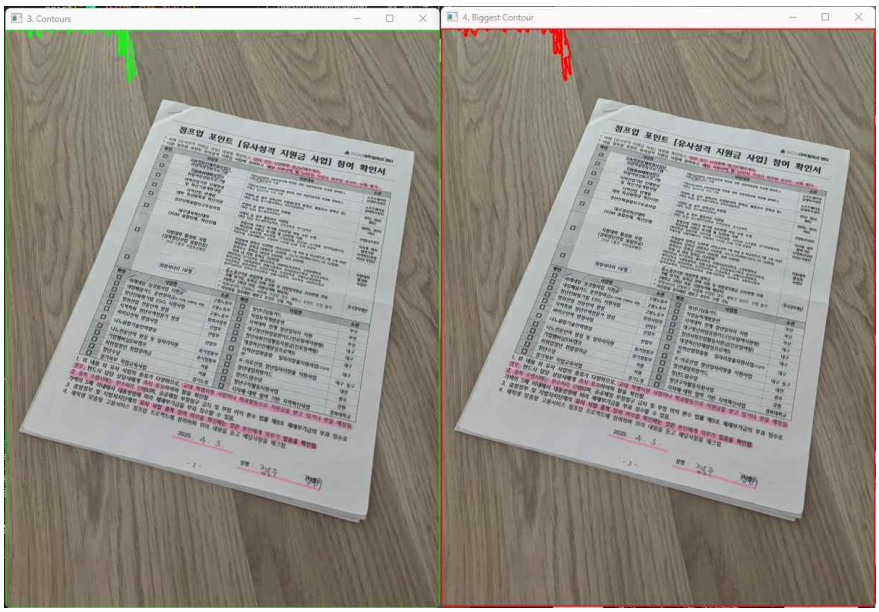


그림 9 모폴로지 연산 후 외곽선 추출 결과

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

- 외곽선이 추출되지 않는 이유로는 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

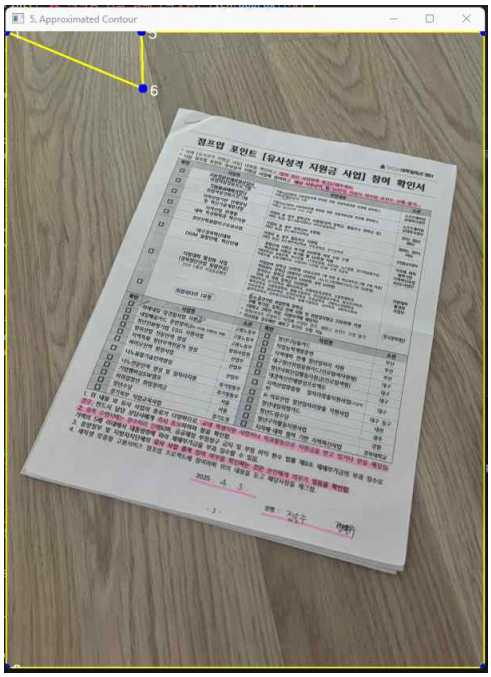


그림 10 꼭짓점 추출

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

```python
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

```python
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

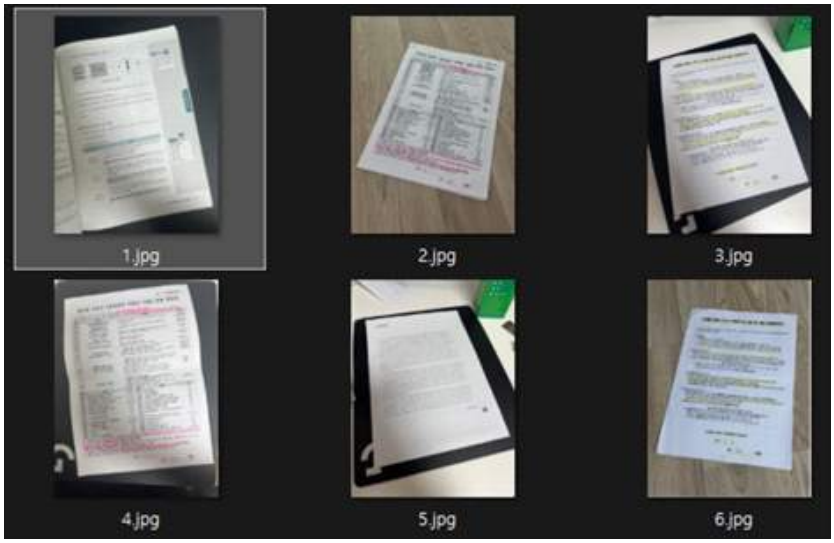


그림 14 수집한 데이터 형식

- 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

```python
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

```python
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

```python
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

```python
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

```python
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

```python
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

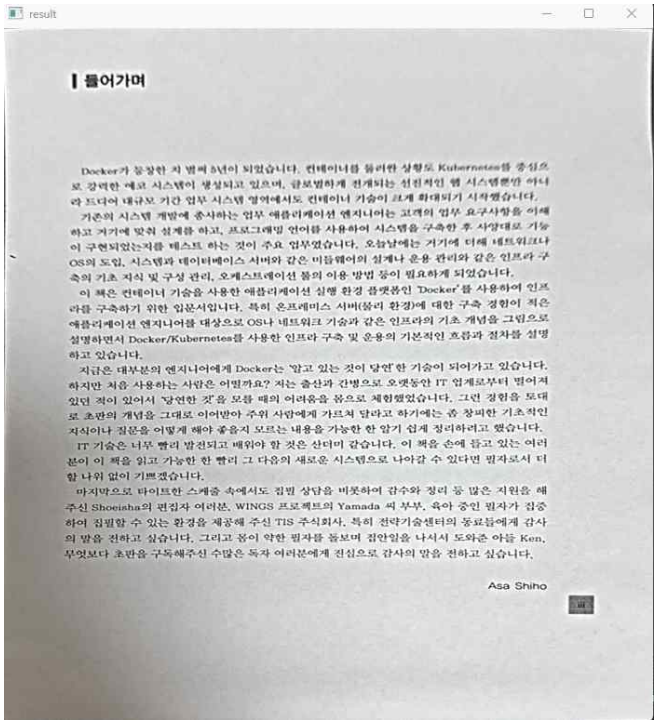


그림 22 투시 변환이 정상적으로 적용된 모습

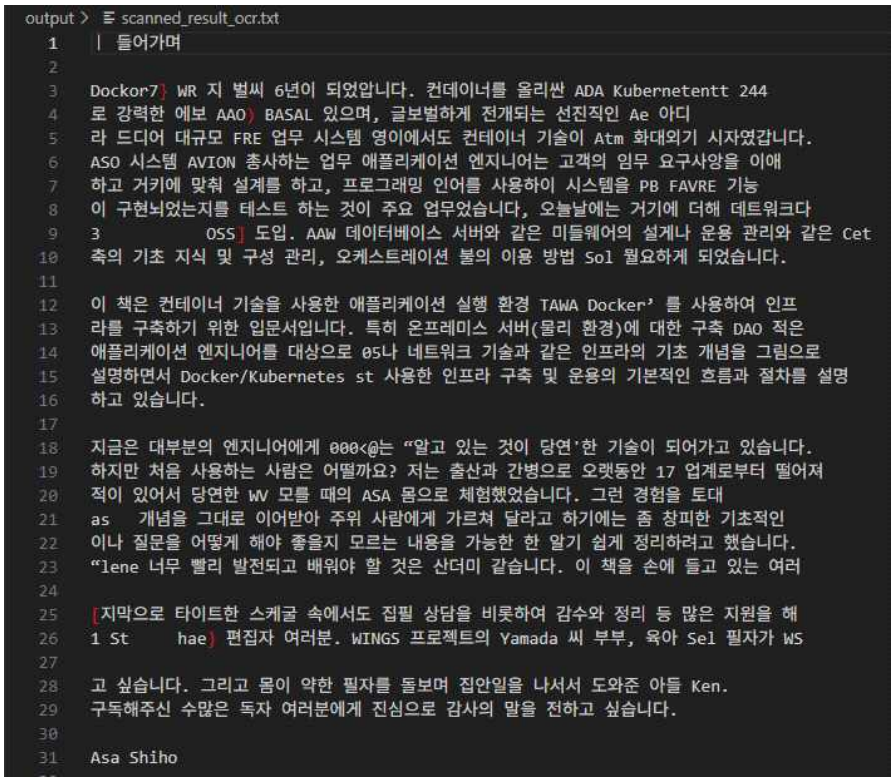


그림 23 OCR을 통해 이미지에서 텍스트 추출 결과값

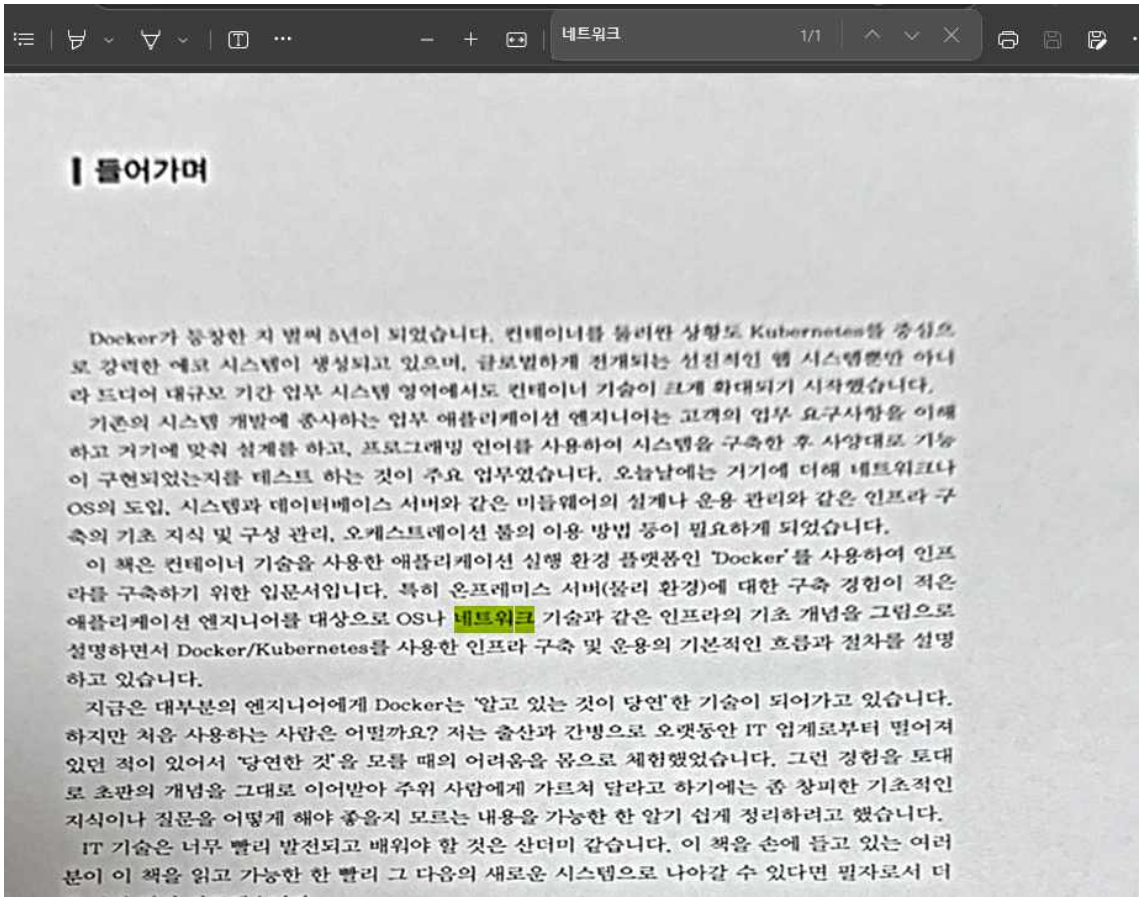


그림 24 텍스트 검색이 가능한 모습

GitHub 주소

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