Open CV를 이용한 자동 문서 인식

장다솜°
이화여자대학교
somdah@ewhain.net

Automatic Document Scanning Using Open CV

Dasom Jang°
Ewha Womans University

<목차>

1. 목적

문서 사진을 입력하고 문서를 스캔한 것과 같은 영상을 출력한다.

input : color document image output : scanned image (binary, grayscale)

2. 소스코드 및 수행과정

2.1 영상 입력 및 크기 조절

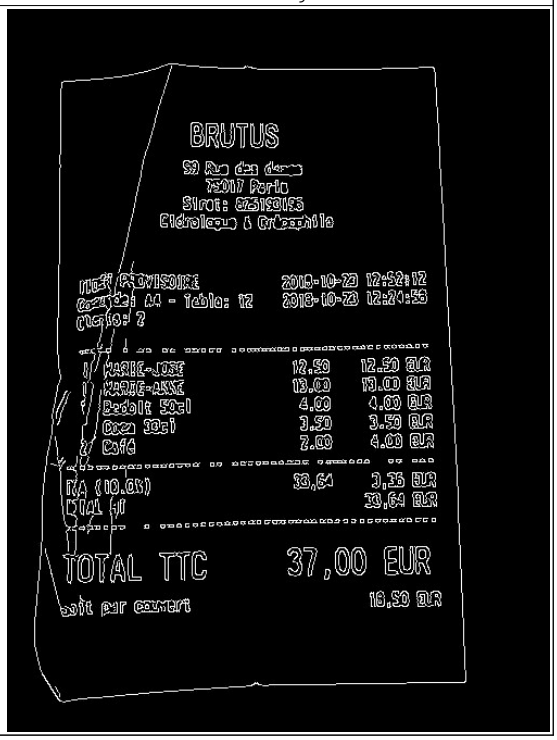
영상을 입력받는다. 영상 크기가 너무 크거나 작으면, 잘못된 결과가 나온다는 것을 발견하고, 영상 크기를 1000으로 제한하였다.

```
# Read the image
img=cv2.imread('input.jpeg')

# Image resizing if needed
if(img.shape[1]>1000 or img.shape[0]>1000):
        if (img.shape[1]>1000):
                r=1000.0 / img.shape[1]
                dim=(1000, int(img.shape[0] * r))
                img=cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
        elif (img.shape[0]>1000):
                r=1000.0 / img.shape[0]
                dim=(int(img.shape[1] * r),1000)
                img=cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

if(img.shape[1]<500 or img.shape[0]<500):
        if (img.shape[1]<500):
                r=1000.0 / img.shape[1]
                dim=(500, int(img.shape[0] * r))
                img=cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
        elif (img.shape[0]<500):
                r=500 / img.shape[0]
                dim=(int(img.shape[1] * r),500)
                img=cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
```

2.2 Edge detection을 통한 문서 영역 찾기

Edge detecting 과정을 수월하게 하기 위해 color 영상을 grayscale로 전환한다. 이후 잡음을 없애기 위해 GaussianBlur 함수를 통해 영상을 흐리게 만들어 준 다음, Canny 함수를 써서 영상의 Edge를 추출해낸다.

```
# Convert to grayscale and find edges
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
blur=cv2.GaussianBlur(gray,(5,5),0)
edge=cv2.Canny(blur,50,150)
#cv2.imwrite('canny.jpg',edge)
```

input	canny
	

<그림1>

findContours 함수를 이용하여 문서 영역을 찾는다. findContours 함수는 edge를 기준으로 하여 영상의 영역을 분류한 후, 각 영역에 대한 좌표 값을 반환한다. 그 이후, contourArea 함수를 통해 영역의 넓이를 계산한 후, 넓이가 가장 큰 영역을 문서 영역으로 지정한다.

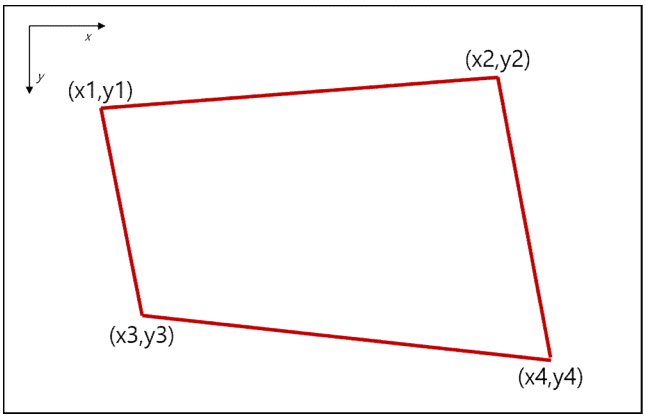
```
#Find and draw contours
contours,_=cv2.findContours(edge.copy(),
                            cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(img,contours,-1,[0,255,0],2)
#cv2.imshow('Contours',img)
#cv2.imwrite('Contours.jpg',img)
#Find the part of the document(maximum area) in the image
n=len(contours)
max_area=0
pos=0

for i in contours:
        area=cv2.contourArea(i)
        if area>max_area:
                max_area=area
                pos=i
```

input	contours
	

<그림2>

2.3 문서 영역을 사각형 틀에 맞추기

문서의 꼭짓점의 좌표를 구한다. 문서의 꼭짓점의 좌표는 contours(문서 영역의 경계)에 해당하는 x좌표와 y좌표의 차로 구할 수 있다. 아래 그림을 예로 들어 보자. 표시된 부분이 문서의 테두리라고 하면, 표시된 영역의 x,y 좌표 값의 합(x+y)는 문서의 좌측 상단 꼭짓점(x1,y1)에서 가장 작고, 우측 하단 꼭짓점(x2,y2)에서 가장 크다. 표시된 영역의 y좌표 값에서 x 좌표 값을 뺀 값(y-x)은 문서의 우측 상단 꼭짓점(x2,y2)에서 가장 작고, 좌측 하단 꼭짓점(x3,y3)에서 가장 크다. 이를 바탕으로 문서의 꼭짓점을 찾는 findVertices 함수를 만들 수 있다. 이 함수는 문서의 대략적인 크기와, 문서의 꼭짓점의 좌표를 반환한다.



<그림3>

```
def findVertices(pos):
        pts=[]
        n=len(pos)

        for i in range(n):
                pts.append(list(pos[i][0]))
                # list of contour points {(x,y)}
        sums={}
        diffs={}
        for i in pts: # for every contour points
                x=i[0] # x value (column)
                y=i[1] # y value (row)
                sum=x+y
                diff=y-x
                sums[sum]=i
                diffs[diff]=i
```

```
        for i in pts: # for every contour points
                x=i[0] # x value (column)
                y=i[1] # y value (row)
                sum=x+y
                diff=y-x
                sums[sum]=i
                diffs[diff]=i

        sums=sorted(sums.items())
        diffs=sorted(diffs.items())
        n=len(sums)

        rect=[sums[0][1], diffs[0][1], diffs[n-1][1], sums[n-1][1]]
        #     top-left     top-right   bottom-left   bottom-right

        h1=np.sqrt((rect[0][0]-rect[2][0])**2 + (rect[0][1]-rect[2][1])**2)
                                                    #height of left side
        h2=np.sqrt((rect[1][0]-rect[3][0])**2 + (rect[1][1]-rect[3][1])**2)
                                                    #height of right side
        h=max(h1,h2)

        w1=np.sqrt((rect[0][0]-rect[1][0])**2 + (rect[0][1]-rect[1][1])**2)
                                                    #width of upper side
        w2=np.sqrt((rect[2][0]-rect[3][0])**2 + (rect[2][1]-rect[3][1])**2)
                                                    #width of lower side
        w=max(w1,w2)

        return int(w),int(h),rect
```

```
# Find the vertices and the size of the object
peri=cv2.arcLength(pos,True)
approx=cv2.approxPolyDP(pos,0.02*peri,True)
size=img.shape
w,h,arr=findVertices(approx)
```

문서의 꼭짓점의 좌표와, 문서의 크기를 바탕으로 perspective matrix를 계산한다. 이때, getPerspective Transform 함수를 이용하면 쉽게 구할 수 있다. warpPerspective 함수를 이용하면, Perspective matrix를 이용

하여, 원본 문서 영상을 사각형의 틀에 맞게 변형한 영상을 생성할 수 있다.

```
# Find the perspective matrix
pts2=np.float32([[0,0],[w,0],[0,h],[w,h]])
pts1=np.float32(arr)
M=cv2.getPerspectiveTransform(pts1,pts2)

# Adjust the image into a rectangular form with perspective transformation
fitted=cv2.warpPerspective(gray,M,(w,h))
#cv2.imshow('FITTED',fitted)
```

input	fitted
	

<그림 4>

2.4 영상 보정

영상을 더 선명하고 가독성 있게 만드는 과정이다. 결과를 Grayscale로 출력하느냐, binary 로 출력하느냐에 따라 그 과정이 다르다.

문서의 글자는 까맣게, 바탕은 하얗게 출력하는 binary output을 생성하기 위하여, adaptiveThreshold 함수를 사용하였다. Adaptive thresholding 이후의, binary output이 출력되긴 했지만, 글자의 경계가 다소 거칠게 표현되어 Gaussian blurring을 통해 좀 더 부드럽게 만들어 주었다.

```
#Rivision process
if(1):  # binary output
        # Make the document clear with adaptive thresholding
        # and make letters smoother by Gaussian blurring
        output=cv2.adaptiveThreshold(fitted, 255,
                                     cv2.ADAPTIVE_THRESH_MEAN_C,
                                     cv2.THRESH_BINARY, 7, 12)
        output=cv2.GaussianBlur(output,(3,3),0)
```

fitted	binary output
	

<그림 5>

영상을 더욱 선명하게 만들어 주는 Unsharp Masking 과정을 통해 grayscale output을 생성하였다.

```
else : # grayscale output
        # using unsharp masking
        k=0.7
        blurred=cv2.GaussianBlur(fitted,(5,5),0)
        output=cv2.addWeighted(fitted, 1/(1-k), blurred, -k/(1-k), 0)
```

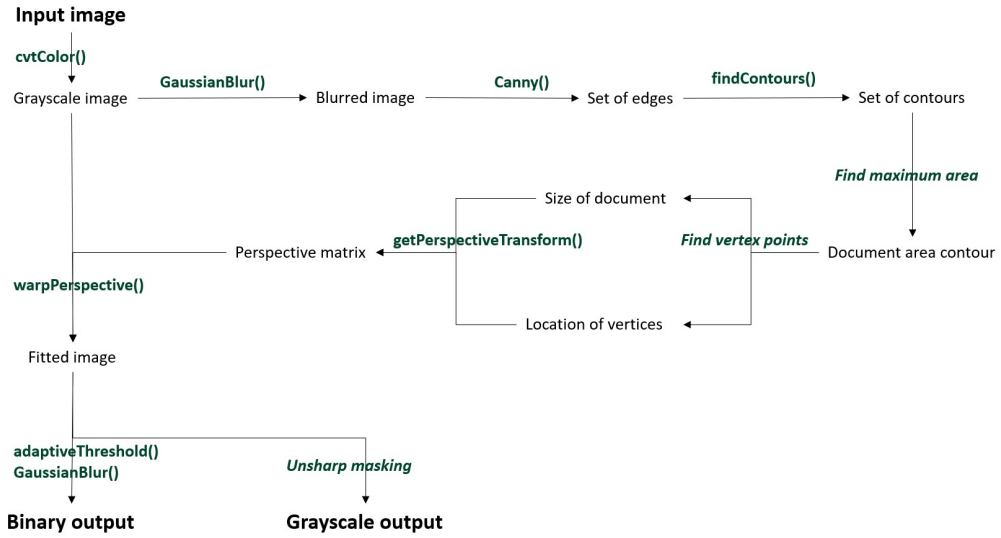
fitted	grayscale output
	

<그림 6>

2.5 결과 출력

```
output = cv2.resize(output,(w,h),interpolation = cv2.INTER_AREA)

#Show the final output image (the scanned document)
cv2.imshow('OUTPUT',output)

#Save the final output image (the scanned document) and finish
cv2.imwrite('output.jpg',output)
```

이 모든 과정을 요약하자면 다음과 같다.



<그림 7>

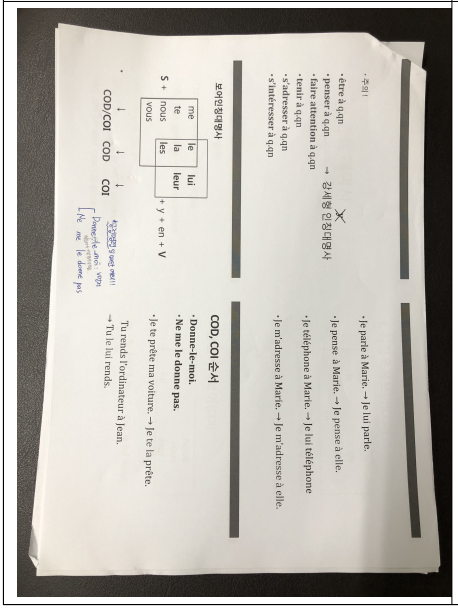
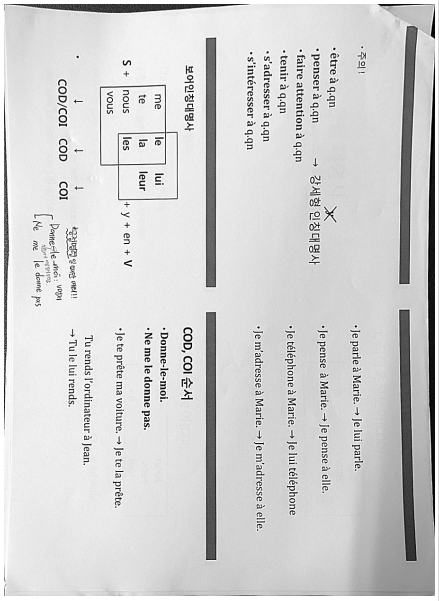
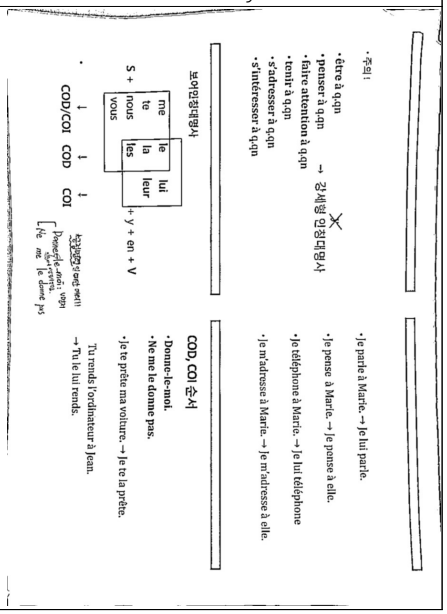
3. 결과

input	grayscale	binary
		

<그림8>

input	grayscale	binary
		

<그림9>

input	grayscale	binary
		

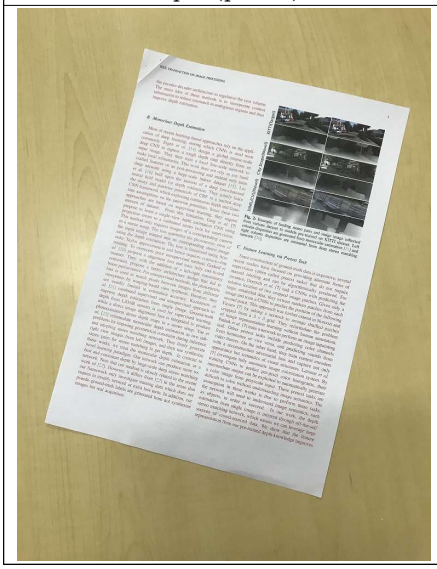
<그림10>

input	grayscale	binary
		

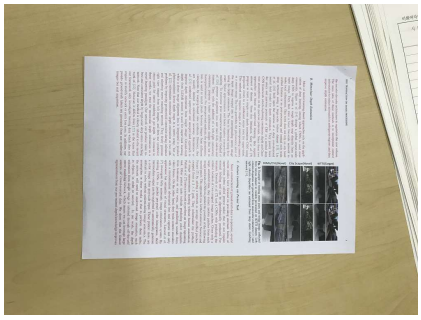
<그림11>

input (pic00)	grayscale	binary
		

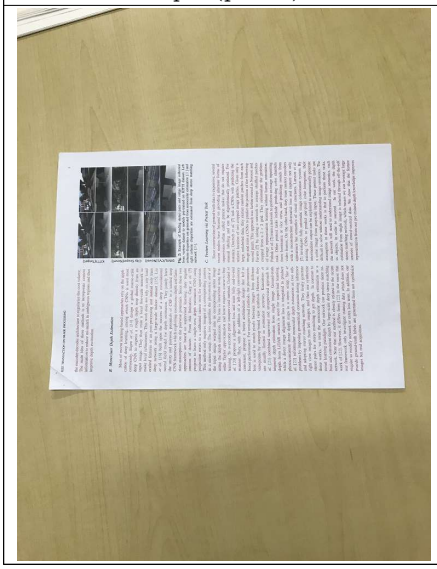
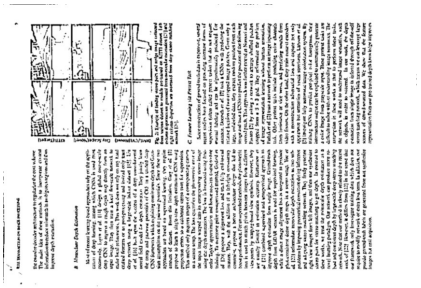
<그림12>

input(pic01)	grayscale	binary
		

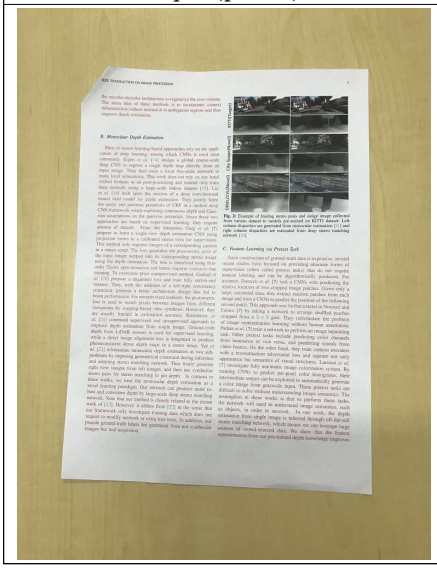
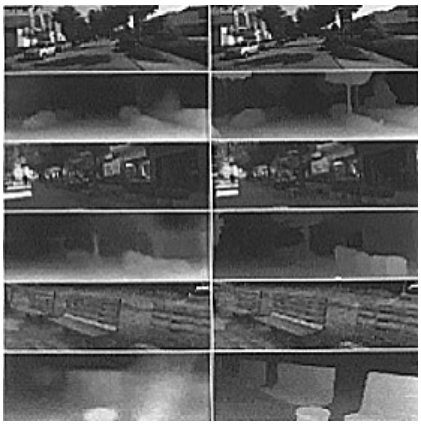
<그림13>

input(pic02)	grayscale	binary
		

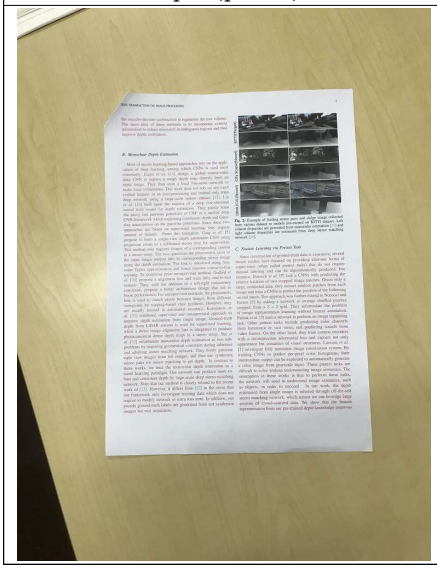
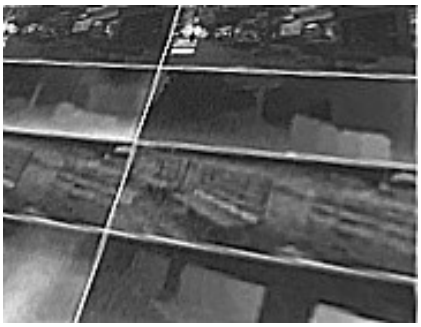
<그림14>

input(pic03)	grayscale	binary
		

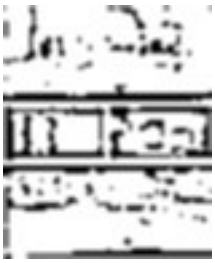
<그림15>

input(pic04)	grayscale	binary
		

<그림16>

input(pic05)	grayscale	binary
		

<그림17>

input(pic06)	grayscale	binary
		

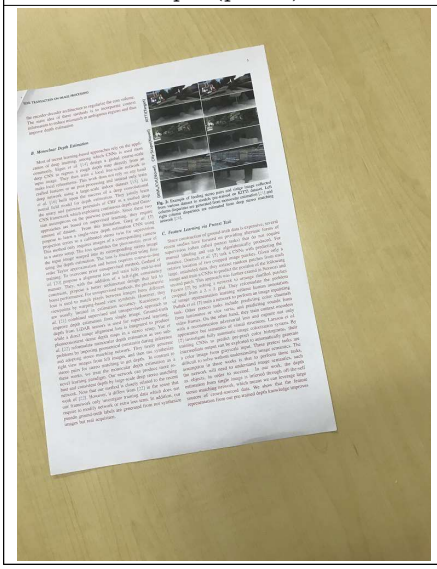
<그림18>

input(pic07)	grayscale	binary
		

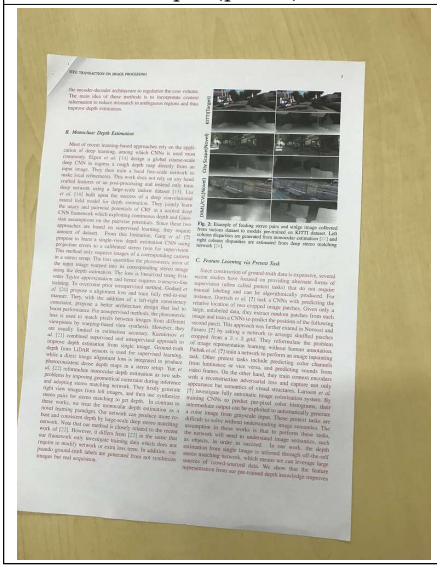
<그림19>

input(pic08)	grayscale	binary
		

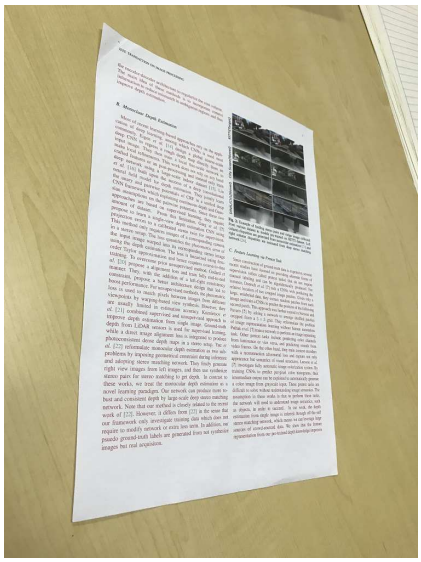
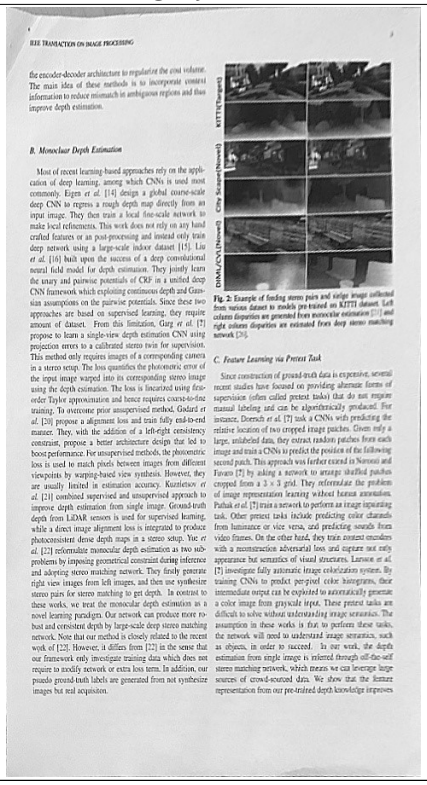
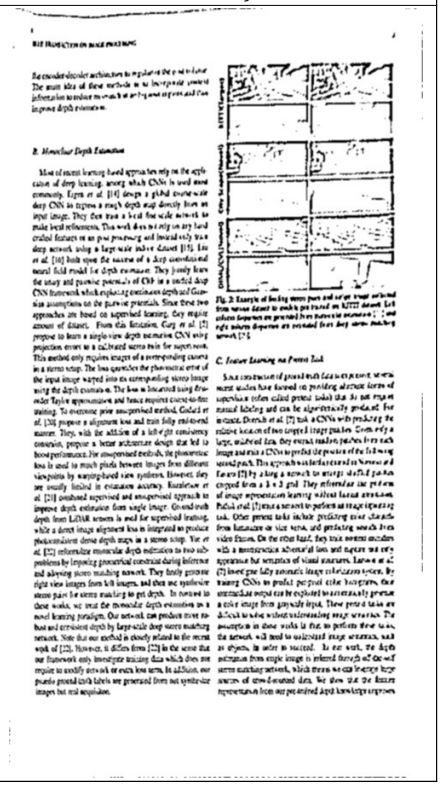
<그림20>

input(pic09)	grayscale	binary
		

<그림21>

input(pic10)	grayscale	binary
	Traceback (most recent call last): File "C:₩Users₩OWNER₩Downloads₩Dataset (document scanning)₩Dataset (document scanning)₩document_scanning.py", line 101, in <module> w,h,arr=findVertices(approx) File "C:₩Users₩OWNER₩Downloads₩Dataset (document scanning)₩Dataset (document scanning)₩document_scanning.py", line 30, in findVertices rect=[sums[0][1], diffs[0][1], diffs[n-1][1], sums[n-1][1]] IndexError: list index out of range >>>	

<그림22>

input(pic11)	grayscale	binary
		

<그림23>

4. 결과 분석

Binary output의 경우, 문서의 어두운 부분은 검은색으로, 밝은 부분은 흰 색으로 나타내어지는 것이 적절한 결과이다. 그러나 adjustive thresholding을 사용한 결과 그 경계부분만 검은 색으로 표시되었다. Optimum global thresholding을 사용하면 더욱 알맞은 결과가 나올 것으로 예상된다

findContour 함수에 의해 문서의 경계가 정확히 인식되지 않은 경우가 많았다. 어떤 경우는 그림 속의 표와 같이 사각형 영역을 문서로 인식했으며(그림16, 그림18, 그림19) 전혀 인식이 되지 않는 경우가 많았다. 그림21, 22와같이 한쪽 모서리가 잘린 경우는 findContour에 의해 문서를 인식할 수 없다. 그러나 문서의 외곽이 전부 사진 안에 담겨 있는 경우도 문서 영역을 인식하지 못하는 경우가 많았다. Canny 함수를 이용한 Edge 검출 결과를 확인해보니, 영상의 edge가 정확이 인식되지 않았기 때문이었다. Canny 함수는 입력 영상을 흑백으로 변환한 후, 사용되기 때문에, 컬러 영상에서 경계가 확인 되더라도 흑백 영상에서 경계가 명확하지 않다면 edge로 인식할 수 없다.

5. 사용한 함수

	함수	출처
1	cv.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst])	OpenCV
2	cv.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]])	OpenCV
3	cv.Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]])	OpenCV
4	cv.cvtColor(src, code[, dst[, dstCn]])	OpenCV
5	cv.drawContours(image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]]])	OpenCV
6	cv.findContours(image, mode, method[, contours[, hierarchy[, offset]]])	OpenCV
7	cv.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]])	OpenCV
8	cv.getPerspectiveTransform(src, dst[, solveMethod])	OpenCV
9	cv.imread(filename[, flags])	OpenCV
10	cv.imshow(winname, mat)	OpenCV
11	cv.imwrite(filename, img[, params])	OpenCV
12	cv.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])	OpenCV
13	cv.warpPerspective(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])	OpenCV
14	findVertices(contourPoints)	Ebey Abraham

참고문헌

Ebey Abraham, “Untitled”, 2017, https://pastebin.com/2zLk75ze

OpenCV-Python Study documentation!, “이미지 임계처리”, https://opencv-python.readthedocs.io/en/latest/doc/09.imageThresholding/imageThresholding.html

OpenCV-Python Study documentation!, “이미지의 기하학적 변형”, https://opencv-python.readthedocs.io/en/latest/doc/10.imageTransformation/imageTransformation.html

후니의 컴퓨터,2018, “[OpenCV-python] adaptive Threshold”, https://hoony-gunputer.tistory.com/99

데이터사이언스스쿨, “이미지 컨투어”, 2018, https://datascienceschool.net/view-notebook/f9f8983941254a34bf0fee42c66c5539/