به نام خدا

ترم ۹۹۲

عنوان درس : مبانی بینایی کامپیوتر

نام استاد : دکتر شاه بهرامی

عنوان پروژه: document scanner with OpenCV

اسامی افراد و شماره دانشجویی :

سحر فخریه کاشان ۹۶۰۱۲۲۶۸۰۰۱۲

نگین بنای شاهانی ۹۶۰۱۲۲۶۸۰۰۳۱

دلیل انتخاب این پروژه :

ما همیشه دوست داشتیم با یادگیری مباحث جدید خود را به چالش بکشیم ، طراحی برنامه ی document scanner یکی از آن موارد جالب و چالش برانگیز برای گروه ما است. با توجه به مباحث تدریس شده در چند جلسه ی اخیر ما به مبحث image processing علاقه پیدا کردیم و بعد از مطالعه ی چند مقاله که در رفرنس آمده تصمیم به انتخاب این پروژه گرفتیم.

کاربرد:

با توجه به شرایط قرنطینه و تعطیلی مراکز آموزشی نیاز به فرستادن سند (document) نسبت به گذشته بیشتر احساس می شود. به همین علت تصمیم گرفتیم که یک برنامه برای اسکن کردن تصاویر و تبدیل آن ها به فرمت های مختلف که کاربر بتواند از بین آن ها انتخاب کند(به عنوان مثال Warp Gray ، Adaptive Threshold، Warp Prespective و ....) طراحی کنیم. و در نهایت کاربر بتواند فرمت انتخابی خود را به صورت pdf در خروجی بگیرد.

مراحل انجام پروژه:

۱. باز کردن file browser و گرفتن تصویر
۲. تصویر اصلی را کپی میکنیم و نگه میداریم و خروجی میدیم
۳. تبدیل تصویر ورودی به تصویر سطح خاکستری و خروجی
۴. مات کردن تصویر با یکی از دو تابع زیر و خروجی آن
۵. یافتن گوشه های تصویر و خروجی
۶. کپی کردن تصویر edge
۷. پیدا کردن بزرگ ترین مربع یا ذوزنقه
۸. تبدیل و تغییر شکل آن
۹. اعمال فیلتر های threshholding
۱۰. نمایش خروجی و تبدیل عکس به pdf

کار های انجام شده:

۱. یادگیری opencv پایتون در حد مورد نیاز
۲. تحقیق درمورد توابع لازم
۳. طراحی گرافیک پروژه (با استفاده از Qt Designer)
۴. باز کردن file browser
۵. نوشتن کد اصلی
۶. اضافه کردن کد به گرافیک و انجام تغییرات نهایی
۷. یافتن تابع برای تبدیل عکس به pdf
۸. پیدا کردن بزرگ ترین مربع یا ذوزنقه در تصویر
۹. تبدیل و تغییر شکل آن

و یافتن توابع زیر:

۱۰.گرفتن تصویر
۱۱.کپی گرفتن از تصویر
۱۲.تبدیل تصویر ورودی به تصویر سطح خاکستری
۱۳.مات کردن تصویر با یکی از دو تابع زیر
۱۴.یافتن گوشه های تصویر (edge)
۱۵.یافتن بزرگ ترین منحنی و شکل بسته ۴ ضلعی در تصویر
۱۶.تابعی برای تغیر شکل تصویر و دادن پرسپکیو به ۴ ضلعی یافته شده
۱۷.فیلتر های threshholding
۱۸.نشان دادن تصویر
۱۹.خروجی به صورت pdf تصویر

کتاب خانه های مورد استفاده:

Numpy برای تغییر شکل و پرسپکتیو ۴ ضلعی

PyQt5 برای گرافیک

PIL , img2pdf برای تبدیل عکس به پی دی اف

کدها و توضیح مراحل:

۱. باز کردن file browser و گرفتن تصویر

```
class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        super(MainWindow, self).__init__(parent)
        path = QFileDialog.getOpenFileName(self, 'Open a file', '',
                                           'All Files (*.*)')
        if path != ('', ''):
            global g
            g = path[0]
app = QtWidgets.QApplication(sys.argv)
window = MainWindow()

image = cv2.imread(g)
```

با استفاده از کتاب خانه PyQt5 یک file browser میاورد و اگر مسیری وجود داشت این مسیر را به متغیر g، global میدهد.

سپس متغیر g که دارای آدرس عکس هست را با استفاده از image = cv2.imread(g) میخوانیم و در متغیر image ذخیره میکنیم.

---

۲. تصویر اصلی را کپی میکنیم و نگه میداریم و خروجی میدیم

```
orig = image.copy()
```

۳. تبدیل تصویر ورودی به تصویر سطح خاکستری و خروجی

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

برای تبدیل کردن فضای رنگی از تابع "cv2.cvtcolor(input_image , flag)" استفاده میشود که flag نوع تبدیل را تعیین میکند.

برای تبدیل BGR به Gray از فلگ "cv2.COLOR_BGR2GRAY" استفاده میشود.

۴. مات کردن تصویر با یکی از دو تابع زیر و خروجی آن

```
blurred = cv2.medianBlur(gray, 5)
```

از این فیلتر برای گرفتن نویز تصاویر و سیگنال‌ها استفاده می‌شود. نویزگیری تصاویر معمولاً پیش‌زمینه‌ای برای دیگر تغییرات و شناسایی‌ها بر روی تصاویر است. مثلاً در این پروژه برای تشخیص گوشه ابتدا باید با استفاده از یکی از فیلترها مانند فیلتر میانه نویز تصویر را گرفتیم.

۵. یافتن گوشه های تصویر و خروجی و کپی کردن تصویر edge

```
edged = cv2.Canny(blurred, 0, 50)
orig_edged = edged.copy()
```

آشکارکننده لبه Canny روشی است برای استخراج اطلاعات ساختاری مفید از اشیا بصری و به طرز چشمگیری میزان داده‌هایی که باید پردازش شوند را کاهش می‌دهد. این روش به‌طور گسترده در سیستم‌های بینایی کامپیوتر استفاده شده‌است. Canny دریافت که لازمه‌های اعمال آشکارسازی لبه روی سیستم‌های بینایی گوناگون نسبتاً یکسان است. به این ترتیب، راهکار آشکارسازی لبه برای مشخص کردن این لازمه‌ها می‌تواند در رنج وسیعی از وضعیت‌ها پیاده‌سازی شود. معیارهای کلی برای لبه یابی شامل موارد زیر می‌شوند:

۱. آشکارسازی لبه با میزان خطای کم، بدین معنی که آشکارسازی باید تا جایی که ممکن است بیشترین لبه‌های نشان داده شده در تصویر را به درستی بدست آورد.
۲. نقطه لبه آشکار شده توسط عملگر باید به‌طور دقیق در مرکز لبه متمرکز شده باشد.
۳. لبه داده شده در تصویر باید یکبار مشخص شود و در صورت امکان، نویز تصویر نباید لبه‌های نادرست ایجاد کند.

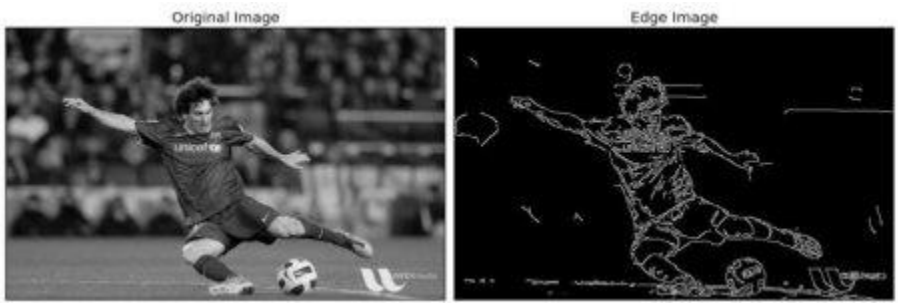
برای تحقق این لازمه‌ها Canny از حساب تغییرات – روشی که تابعی را برای بهینه‌سازی یک تابعی داده شده پیدا می‌کند – استفاده کرد. تابع بهینه در آشکارساز Canny به وسیله جمع ۴ مؤلفه نمایی توصیف می‌شود، اما می‌تواند با مشتق اول یک گاوسی تقریب زده شود.

در بین متدهای لبه یابی توسعه یافته تا کنون، الگوریتم آشکارساز لبه Canny یکی از روش‌های تعریف شده به‌طور دقیقی است که آشکارسازی خوب و قابل اعتمادی را فراهم می‌کند. به دلیل بهینه بودن آن برای تحقق ۳ معیار لبه یابی وسادگی فرایند پیاده‌سازی، این روش یکی از محبوبترین الگوریتم‌ها برای آشکارسازی لبه است.

Canny edge detector

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('messi5.jpg',0)
edges = cv2.Canny(img,100,200)

plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

See the result below:



---

۶. پیدا کردن بزرگ ترین مربع یا ذوزنقه

```
# find the contours in the edged image, keeping only the
# largest ones, and initialize the screen contour
(contours, _) = cv2.findContours(edged, cv2.RETR_LIST,
cv2.CHAIN_APPROX_NONE)
contours = sorted(contours, key=cv2.contourArea, reverse=True)

# get approximate contour
for c in contours:
    p = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.02 * p, True)

    if len(approx) == 4:
        target = approx
        break
```

فرض بر این است که هر بار فقط یک برگه را می خواهیم اسکن کنیم و برگه ها همگی مستطیل شکل (۴ گوشه) هستند.

ابتدا با استفاده از تابع findContours خطوط را در یک تصویر باینری پیدا می کند که در این پروژه این تصویر همان تصویر باینری شده گوشه های توسط تابع canny است. با این کار contours لیستی مرتب از خطوط را می دهد.

سپس این خطوط را با تابع sorted مرتب میکنیم به این ترتیب contours[۱] بزرگترین خط contours[۲] دومین بزرگترین خط و به ترتیب بقیه خطوط را می دهد.

سپس داخل حلقه for با استفاده از تابع arcLength یک طول منحنی یا یک محیط کانتور بسته را محاسبه می کند.

پس از آن که محیط Contour های مشخص شده را پیدا کردیم ، برحسب آنها یک چند گوشه را با استفاده از دستور cv2.approxPolyDP مشخص می کنیم.

در نهایت اگر اندازه approx برابر ۴ شود یعنی شکل ما یک ۴ ضلعی است و همان شکلی است که ما میخواهیم و از طرفی چون خطوط را از قبل با تابع sorted مرتب سازی کرده بودیم میدانیم که این ۴ ضلعی بزرگ ترین ۴ ضلعی موجود در تصویر است.

---

۷. تبدیل و تغییر شکل آن

```
def rectify(h):
    h = h.reshape((4, 2))
    hnew = np.zeros((4, 2), dtype=np.float32)

    add = h.sum(1)
    hnew[0] = h[np.argmin(add)]
    hnew[2] = h[np.argmax(add)]

    diff = np.diff(h, axis=1)
    hnew[1] = h[np.argmin(diff)]
    hnew[3] = h[np.argmax(diff)]

    return hnew



# mapping target points to 800x800 quadrilateral
approx = rectify(target)
pts2 = np.float32([[0, 0], [800, 0], [800, 800], [0, 800]])

M = cv2.getPerspectiveTransform(approx, pts2)
dst = cv2.warpPerspective(orig, M, (800, 800))

imgWarpColored = cv2.warpPerspective(orig, M, (800, 800))
cv2.drawContours(image, [target], -1, (0, 255, 0), 2)
dst = cv2.cvtColor(dst, cv2.COLOR_BGR2GRAY)
```

در این مرحله میخوایم به ۴ ضلعی خود پرسپکتیو بدهیم تا به شکل مستطیل دلخواهمان تبدیل شود

برای استفاده از تبدیل پرسپکتیو به مختصات اولیه و نهایی ۴ نقطه از تصویر احتیاج داریم. مختصات نهایی را خودمان بدست آوردیم. در این برنامه، مرحله قبلی صرفاً جهت پیدا کردن مختصات اولیه آن ۴ نقطه انجام شد و این مرحله اصلی ترین مرحله است.

برای این کار می توانیم از getPerspectiveTransform استفاده کنیم. به این صورت که مختصات چهار گوشه قسمتی از تصویر که می خواهیم تبدیل روی آن انجام شود را مشخص می کنیم (که همان متغیر approx است که با استفاده از تابع rectify آن را به دست آورده ایم) و همچنین مشخص می کنیم که می خواهیم این چهار نقطه روی چه نقاطی در تصویر جدید نگاشت شوند(pts2).

تابع getPerspectiveTransform یک ماتریس ۳ * ۳ بر می گرداند که درواقع اگر هر کدام از چهار نقطه تصویر اصلی به فرم [xi, yi, 1] در این ماتریس ضرب شوند حاصل، نقاط مورد نظر روی تصویر جدید که به فرم [ti*x'i, ti* y'i, ti] هستند خواهد بود(مختصات تصویر جدید x' و y' می باشد و ti عدد ثابت است).

حال برای اینکه تصویر جدید را ایجاد کنیم از warpPerspective استفاده می کنیم که به عنوان ورودی عکس اولیه و ماتریس تبدیل و اندازه عکس جدید را می گیرد و عکس جدید را برمی گرداند.

در نهایت ناحیه چند ضلعی مورد نظر را در یک تصویر ماسک با استفاده از تابع drawContours رسم میکنیم.

برای تبدیل کردن فضای رنگی از تابع "cv2.cvtcolor(input_image , flag)" استفاده میشود که flag نوع تبدیل را تعیین میکند.

برای تبدیل BGR به Gray از فلگ "cv2.COLOR_BGR2GRAY" استفاده میشود.

۸. اعمال فیلتر های threshholding

```
# using thresholding on warped image to get scanned effect (If
Required)
ret2, th4 = cv2.threshold(dst, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
```

```
# other thresholding methods

ret, thresh1 = cv2.threshold(dst, 127, 255, cv2.THRESH_BINARY_INV)
ret, thresh2 = cv2.threshold(dst, 127, 255, cv2.THRESH_TOZERO)
```

Image Thresholding

موارد مورد استفاده Simple thresholding, Adaptive thresholding, Otsu's thresholding etc.

Simple thresholding

در اینجا ، موضوع ساده است. اگر مقدار پیکسل از مقدار آستانه بزرگتر باشد ، یک مقدار به آن اختصاص می یابد (ممکن است سفید باشد) ، در غیر این صورت مقدار دیگری به آن اختصاص می یابد (ممکن است سیاه باشد). تابع cv2.threshold از کتابخانه ی openCV مورد استفاده قرار میگیرد. اولین آرگومان ، تصویر منبع است که باید تصویری در مقیاس grayscale باشد. آرگومان دوم مقدار آستانه (threshold) است که برای دسته بندی مقادیر پیکسل استفاده می شود. آرگومان سوم maxVal می باشد. گفته بودیم "مقدار پیکسل بزرگتر یا کوچتر از آستانه باشد مقداری به آن تعلق میگیرد" در واقع maxVal تعیین میکند که تا چه حد مقدار پیکسل میتواند از مقدار آستانه بزرگتر یا کوچکتر باشد. OpenCV انواع مختلفی از آستانه گذاری را ارائه می دهد و توسط آرگومان چهارم تابع مورد نظر تعیین می شود. انواع مختلف آن عبارتند از:

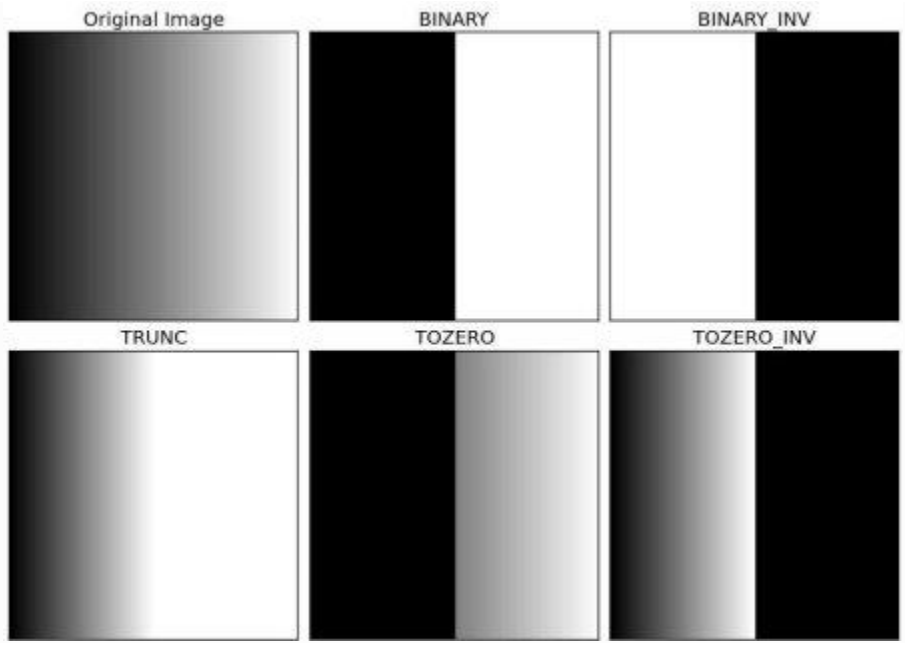
- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

در نهایت دو خروجی به دست می آید. اولین خروجی retval می باشد و دومین خروجی تصویر آستانه یا همان thresholded image است.

- Code :

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('gradient.png',0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)

titles = ['Original
Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in xrange(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

نتیجه ی نهایی خروجی ای مانند تصویر زیر خواهد بود که از میان آن ها به احتمال زیاد از Binary، Binary_inv و ToZero استفاده خواهیم کرد.



Otsu’s Binarization

در Simple thresholding ، بیان کردیم که پارامتر دومی به نام retVal وجود دارد.به دنبال استفاده از Otsu’s Binarization ، به این پارامتر نیاز پیدا خواهیم کرد.

در آستانه ی global ، از یک مقدار دلخواه برای مقدار آستانه استفاده می کردیم .اما برای فهمیدن این که آیا مقدار آستانه ای که انتخاب شده مناسب است ، از آزمون و خطا بهره می بردیم. حال یک تصویر دو حالته یا bimodal را در نظر بگیرید (به عبارت ساده ، تصویر دو حالته تصویری است که هیستوگرام آن دو قله دارد).

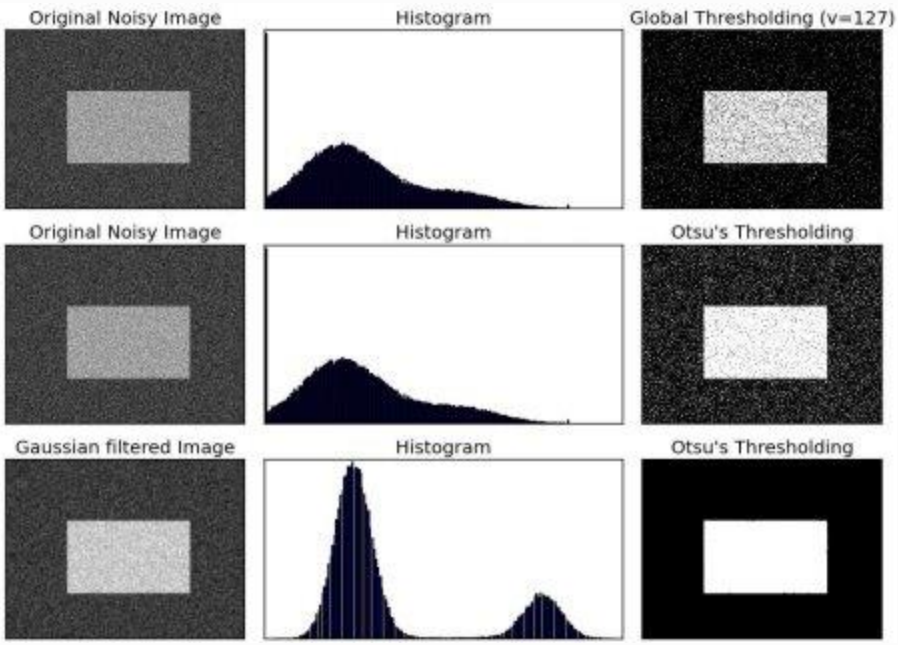
برای این تصویر ، می توانیم مقداری را در وسط آن قله ها به عنوان مقدار آستانه در نظر بگیریم . این همان کاری است که Otsu’s Binarization انجام می دهد. بنابراین به طور خودکار مقدار آستانه را از هیستوگرام تصویر برای یک تصویر دو حالته محاسبه می کند. (برای تصاویری که دو بعدی نیستند ، Binarization دقیق نیست.)

برای این کار ، از تابع cv2.threshold () استفاده می کنیم ، اما یک flag اضافی هم با نام cv2.THRESH_OTSU برای تابع می فرستیم . برای مقدار آستانه ، ابتدا صفر را به تابع میدهیم. سپس الگوریتم مقدار آستانه ی بهینه را پیدا می کند و آن را به عنوان خروجی دوم ، retVal برمی گرداند. اگر از آستانه Otsuاستفاده نشود ، retVal همان مقدار آستانه ای است که استفاده کرده اید.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('noisy2.png',0)

# global thresholding
ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

# Otsu's thresholding
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Otsu's thresholding after Gaussian filtering
blur = cv2.GaussianBlur(img,(5,5),0)
ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# plot all the images and their histograms
images = [img, 0, th1,
          img, 0, th2,
```

```
          blur, 0, th3]
titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',
          'Original Noisy Image','Histogram',"Otsu's Thresholding",
          'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]

for i in xrange(3):
    plt.subplot(3,3,i*3+1),plt.imshow(images[i*3],'gray')
    plt.title(titles[i*3]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+2),plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+3),plt.imshow(images[i*3+2],'gray')
    plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])
plt.show()
```

Result :



---

۹. نمایش خروجی و تبدیل عکس به pdf

```
def dblackAndWhitePDF(self):
cv2.imwrite("q.png", cv2.resize(th4, (600, 600)))
img_path = os.path.join("q.png")

# storing pdf path
pdf_path = os.path.join('blackAndWhite.pdf')

# opening image
image = Image.open(img_path)
```

```
# converting into chunks using img2pdf
pdf_bytes = img2pdf.convert(image.filename)

# opening or creating pdf file
file = open(pdf_path, "wb")

# writing pdf files with chunks
file.write(pdf_bytes)

# closing image file
image.close()

# closing pdf file
file.close()

# output
msg = QMessageBox()
msg.setWindowTitle("Success message")
msg.setText("blackAndWhite.pdf Successfully made pdf file")
x = msg.exec_()  # this will show our messagebox


def dgrayPDF(self):
cv2.imwrite("q.png", cv2.resize(thresh2, (600, 600)))
img_path = os.path.join("q.png")
pdf_path = os.path.join("gray.pdf")
image = Image.open(img_path)
pdf_bytes = img2pdf.convert(image.filename)
file = open(pdf_path, "wb")
file.write(pdf_bytes)
image.close()
file.close()
# output
msg = QMessageBox()
msg.setWindowTitle("Success message")
msg.setText("gray.pdf Successfully made pdf file")
x = msg.exec_()  # this will show our messagebox

def dbinaryPDF(self):
cv2.imwrite("q.png", cv2.resize(thresh1, (600, 600)))
img_path = os.path.join("q.png")
pdf_path = os.path.join("binary.pdf")
image = Image.open(img_path)
pdf_bytes = img2pdf.convert(image.filename)
file = open(pdf_path, "wb")
file.write(pdf_bytes)
image.close()
file.close()
# output
msg = QMessageBox()
msg.setWindowTitle("Success message")
```

```
msg.setText("binary.pdf Successfully made pdf file")
x = msg.exec_()  # this will show our messagebox

def doriginalScanPDF(self):
cv2.imwrite("q.png", cv2.resize(thresh2, (600, 600)))
img_path = os.path.join("q.png")
pdf_path = os.path.join("originalScan.pdf")
image = Image.open(img_path)
pdf_bytes = img2pdf.convert(image.filename)
file = open(pdf_path, "wb")
file.write(pdf_bytes)
image.close()
file.close()
# output
msg = QMessageBox()
msg.setWindowTitle("Success message")
msg.setText("originalScan.pdf Successfully made pdf file")
x = msg.exec_()  # this will show our messagebox


def dwrapGrayPDF(self):
cv2.imwrite("q.png", cv2.resize(dst, (600, 600)))
img_path = os.path.join("q.png")
pdf_path = os.path.join("wrapGray.pdf")
image = Image.open(img_path)
pdf_bytes = img2pdf.convert(image.filename)
file = open(pdf_path, "wb")
file.write(pdf_bytes)
image.close()
file.close()
# output
msg = QMessageBox()
msg.setWindowTitle("Success message")
msg.setText("wrapGray.pdf Successfully made pdf file")
x = msg.exec_()  # this will show our messagebox
```

در تمام این توابع یک کار برای عکس های مختلف انجام میشود

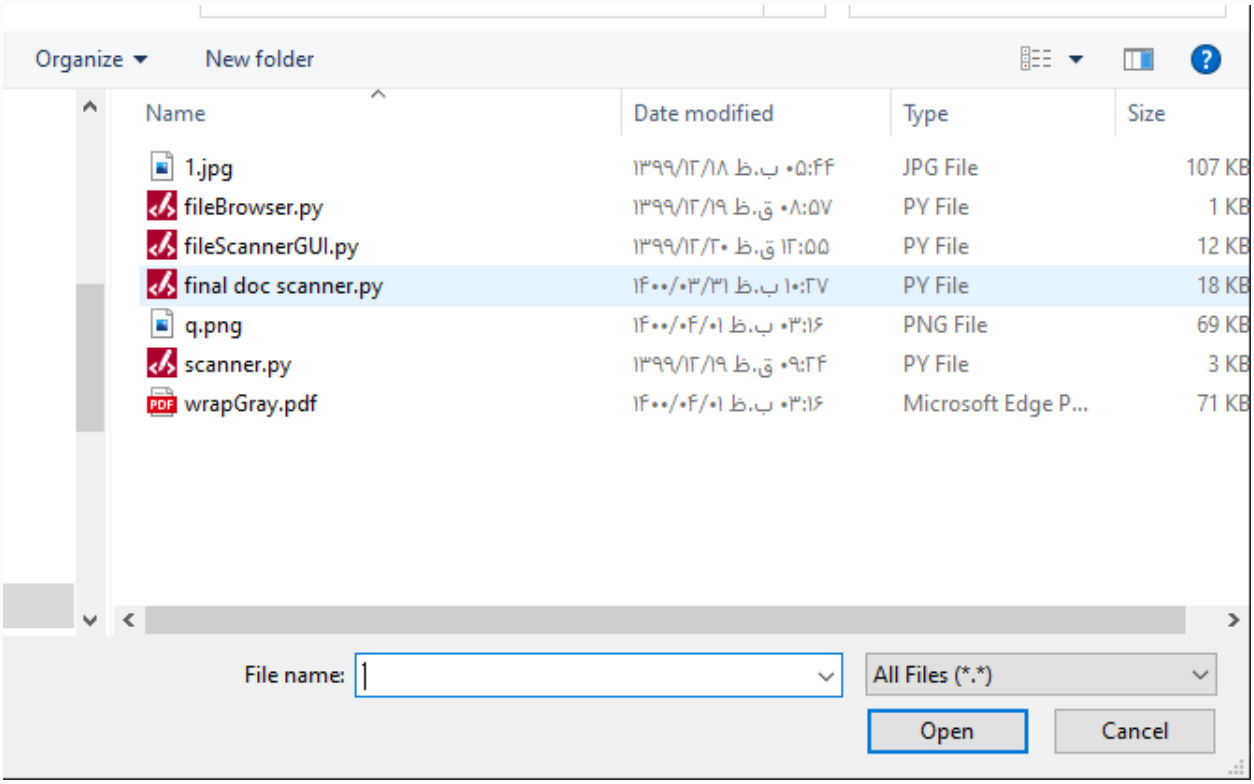
در ابتدا عکسی که ساختیم را با پسوند png. ذخیره میکنیم سپس فایلی با پسوند pdf. ایجاد میکنیم در ادامه عکس ذخیره شده مورد نظر را با تابع (img2pdf.convert(image.filename تبدیل به pdf میکنیم.

در نهایت pdf ایجاد شده را داخل فایل پسوند pdf. که ایجاد کردیم وارد میکنیم(مینویسیم).

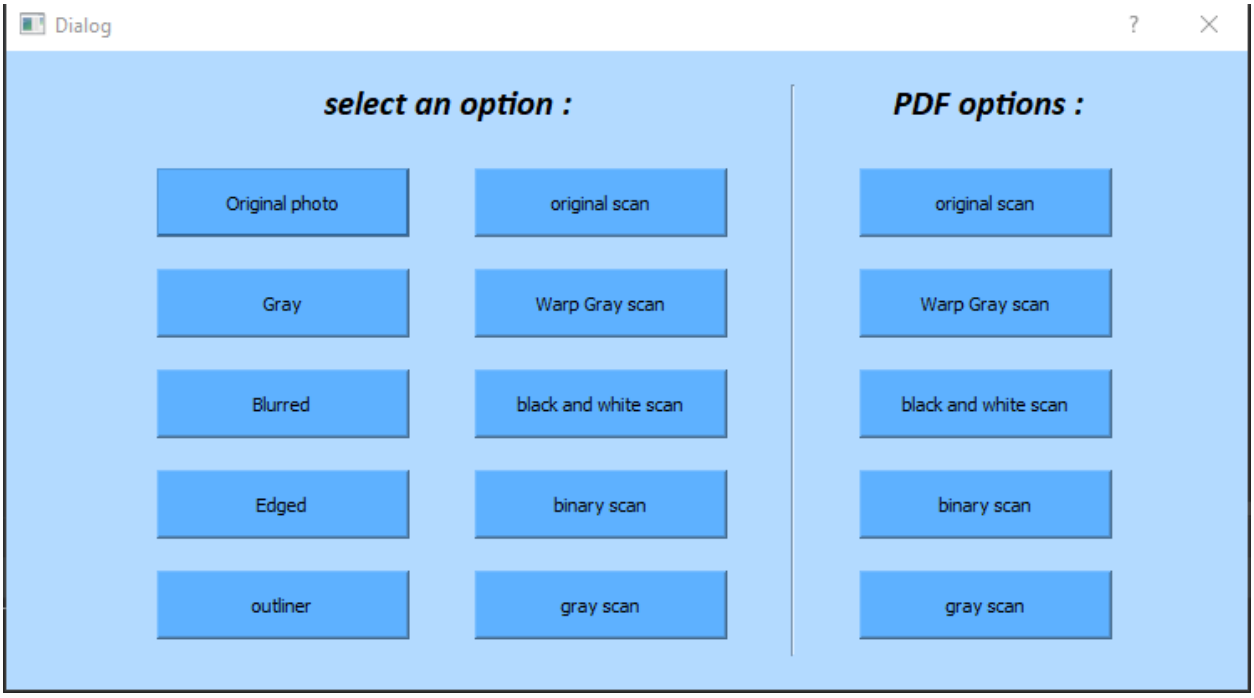
در ادامه برای این که کاربر متوجه شود که عملیات موفقیت آمیز بوده است با استفاده از PyQt5 یک جعبه Success message به کاربر نشان داده میشود.

نمایی از برنامه و ورودی و خروجی ها :

ابتدا با باز کردن یک file browser فایل عکس مورد نظر خود را به عنوان ورودی به برنامه میدهیم:

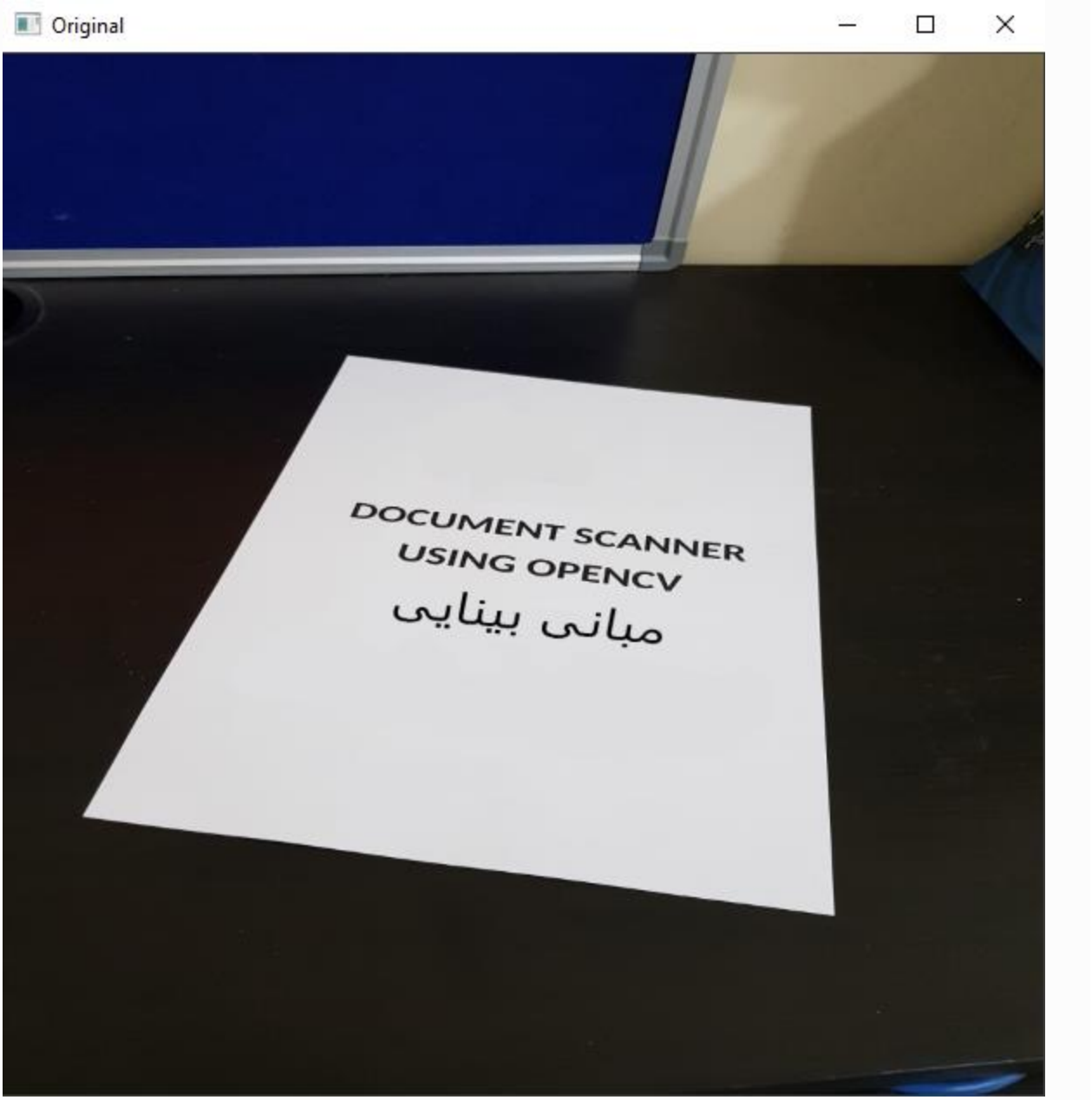


پس از انتخاب فایل پنجره زیر باز خواهد شد و ما میتوانیم با انتخاب هر گزینه خروجی های مورد نظرمان را ببینیم که این خروجی ها به صورت عکس یا پی دی اف هستند.

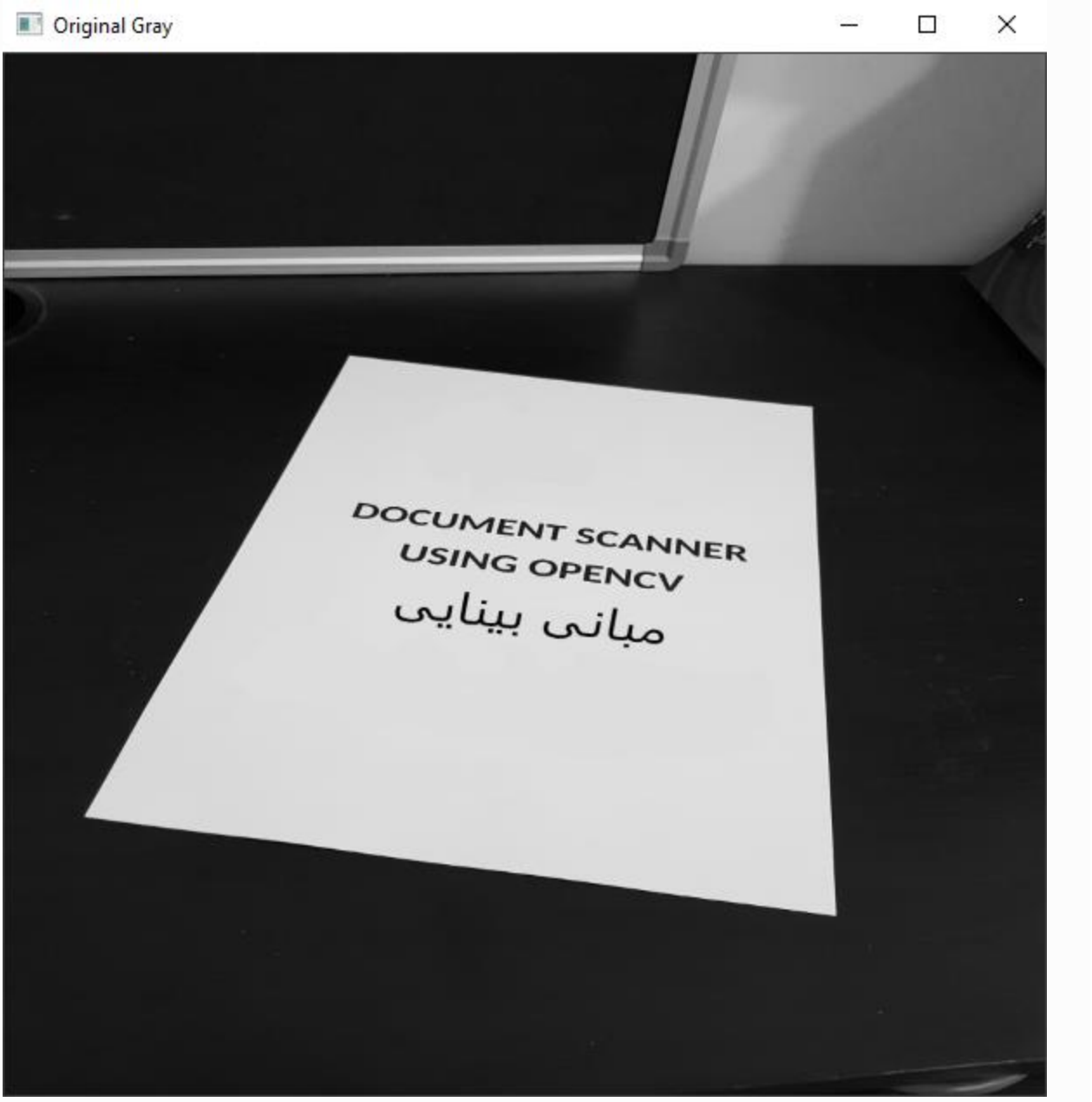


در نهایت عکس هایی از ورودی و خروجی های برنامه :

Original photo



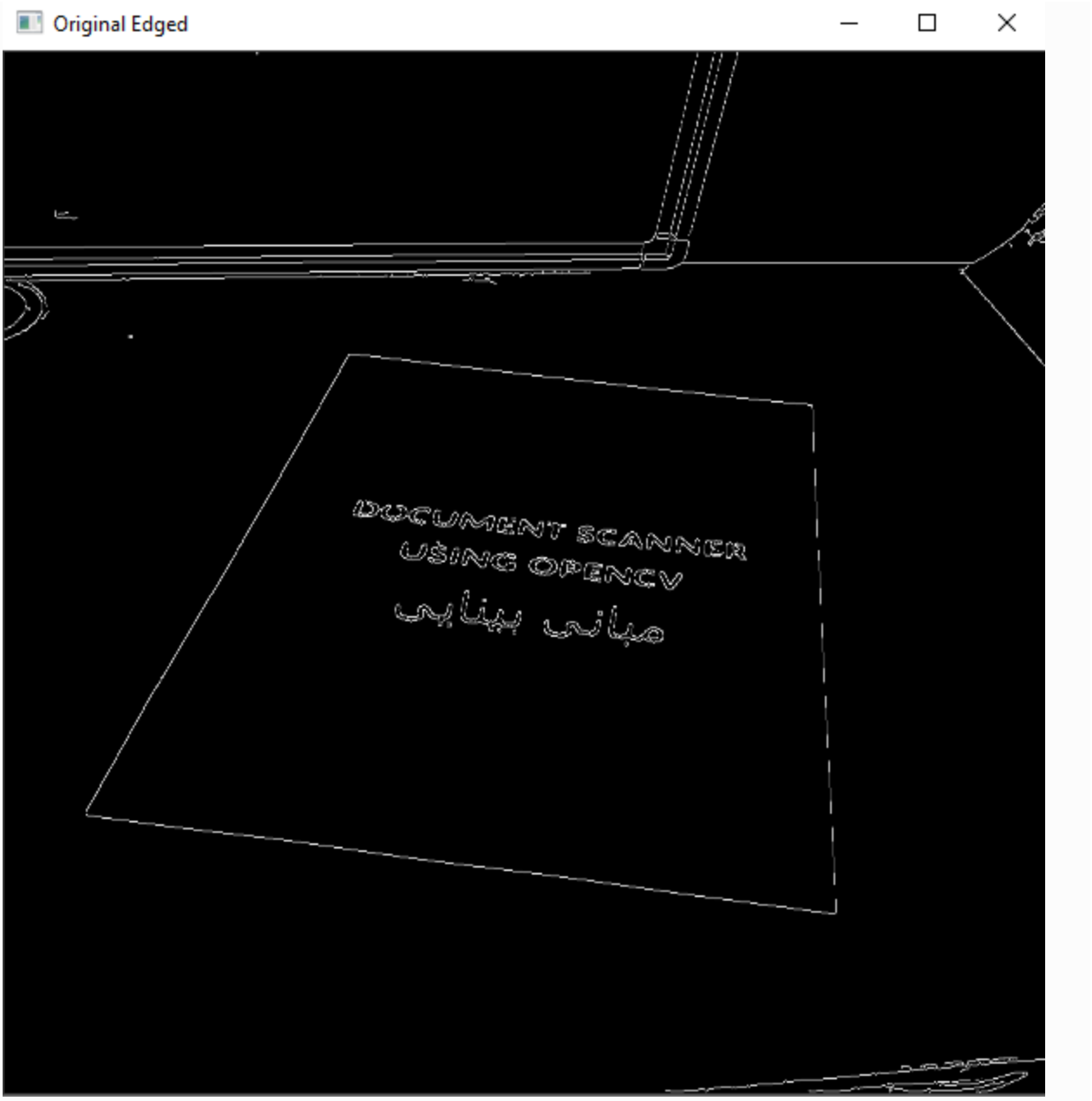
Gray



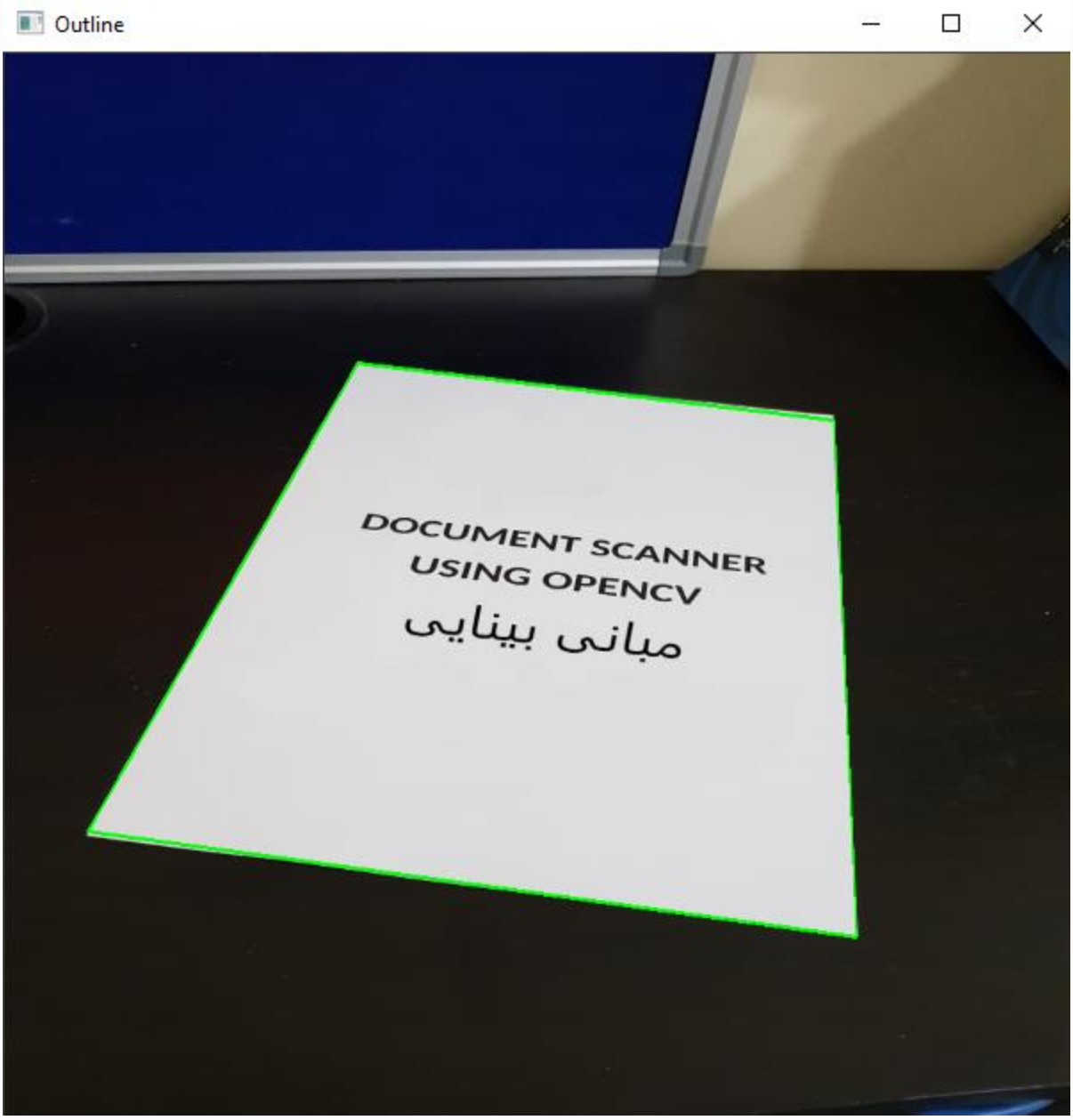
Blurred


Original Blurred
DOCUMENT SCANNER
USING OPENCV
مبانی بینایی

Edged


Original Edged
DOCUMENT SCANNER
USING OPENCV
مبانی بینایی

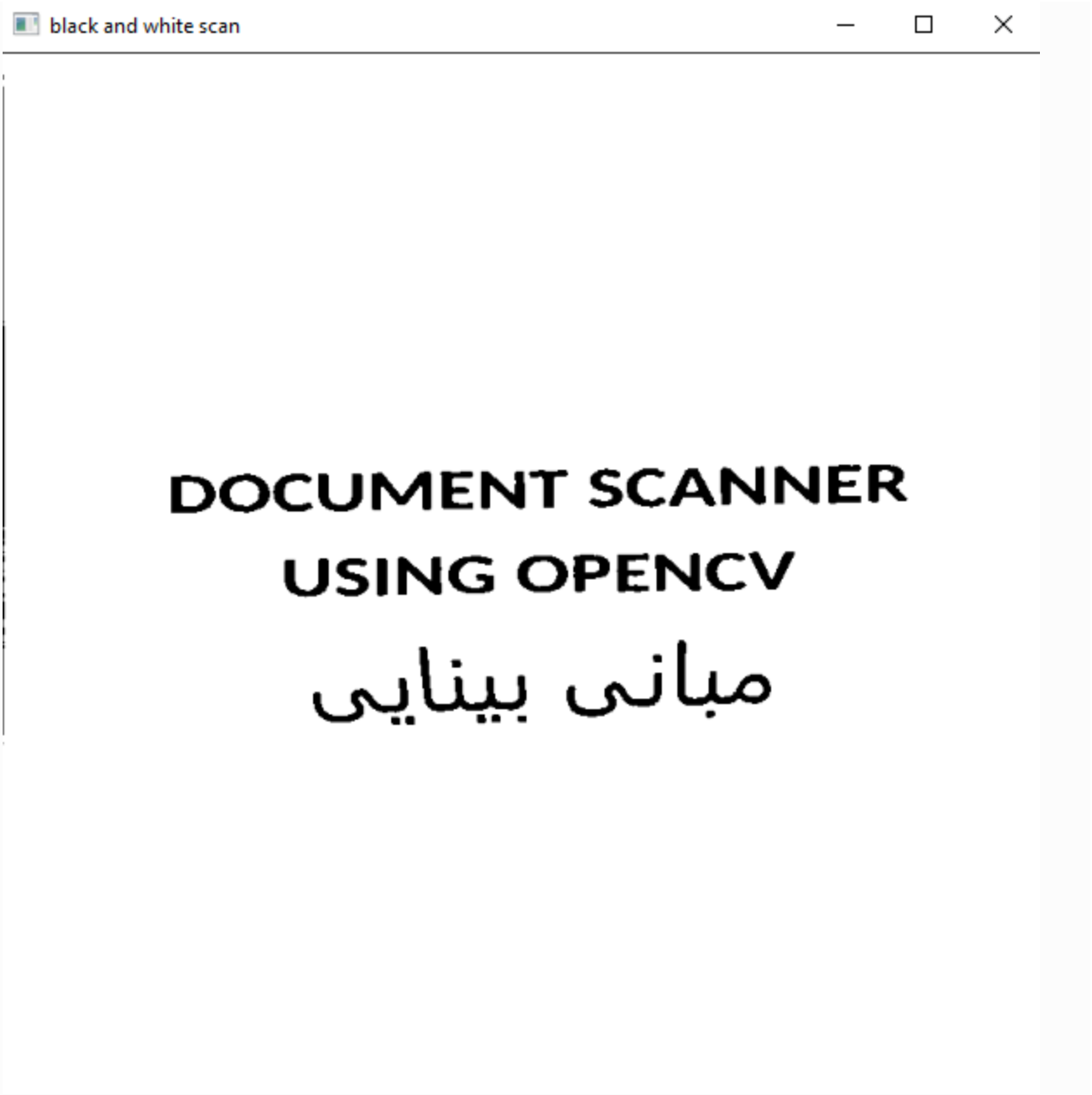
Outliner



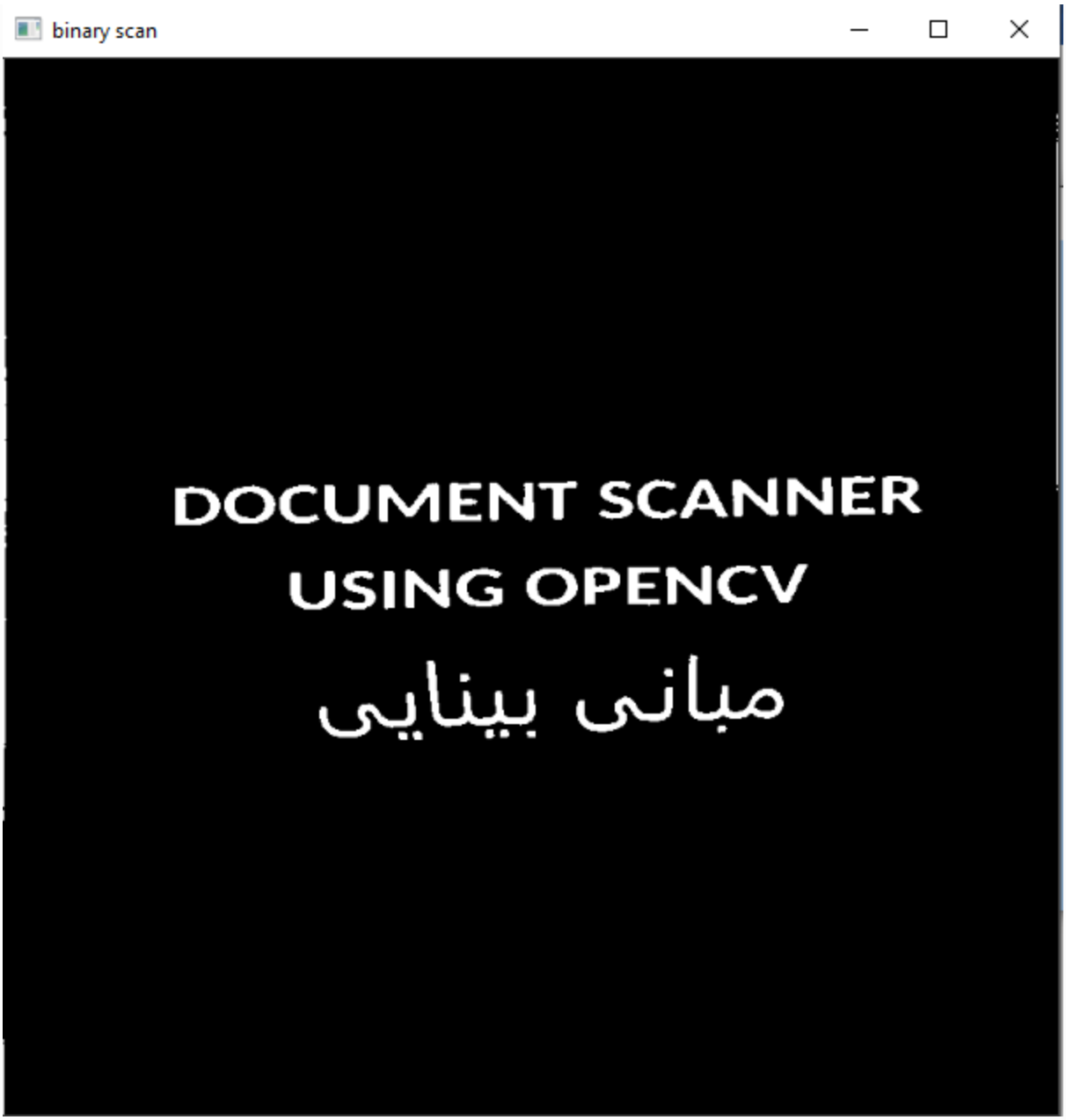
Original scan

Wrap gray scan

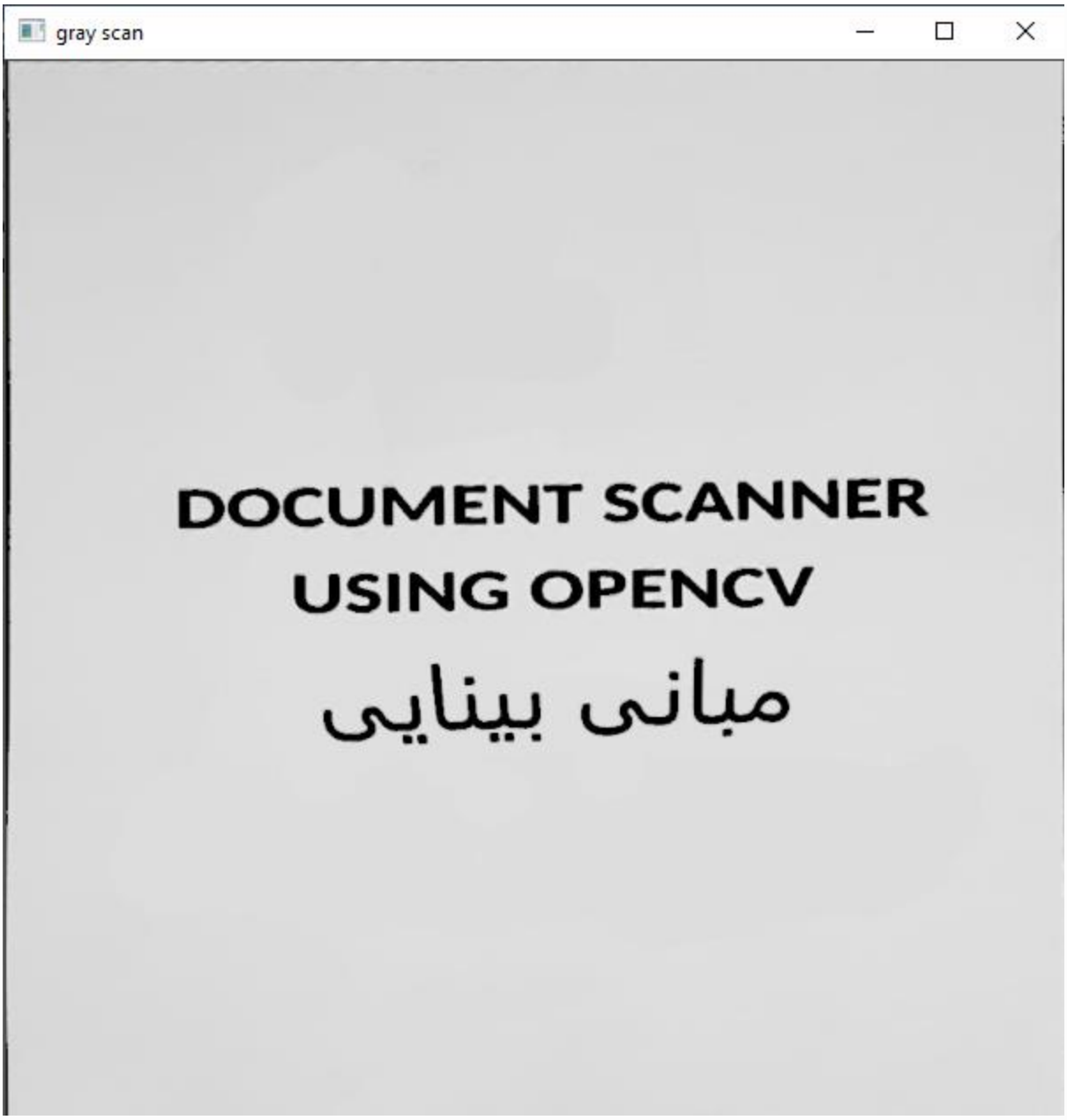
Black and white scan


black and white scan
DOCUMENT SCANNER
USING OPENCV
مبانی بینایی

Binary scan



Gray scan



پی دی اف ها مانند عکس های قبل هستند برای مثل پی دی اف خروجی wrap gray گذاشته شده است.

منابع:

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiJ09ri5ZnvAhXP-aQKHWPGDJ8QFjACegQIFBAD&url=https%3A%2F%2Firjmets.com%2Frootaccess%2Fforms%2Fuploads%2Fdocument-scanner-application-using-python.pdf&usg=AOvVaw369gyU9DXosNliy0-w9lWn

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.1860&rep=rep1&type=pdf

https://ieeexplore.ieee.org/abstract/document/5563693/

https://search.proquest.com/openview/4c51200b092bd6bf2b6dc2e2713338c8/1.pdf?pq-origsite=gscholar&cbl=646415

https://link.springer.com/article/10.1007/s10032-005-0010-9

https://www.scientific.net/AMM.644-650.4477

https://link.springer.com/content/pdf/10.1007/978-1-84628-726-8_2.pdf

اسکن مدارک با استفاده از OpenCV و Python - پروژه های میکروکنترولر و اندروید (microdroidprj.ir)

تبدیل Affine و Perspective در OpenCV – بینایی کامپیوتر (class.vision)

با تشکر