## Defect detection with computer vision

Initial requirements:

* Web-camera
* Python
* OpenCV library

Task 1:

Dynamically recognize arrangement of simple objects in relation to their initial position. Indicate when a book is rotated at angle greater than some specified value.

To solve the problem the following approach is proposed:

1. Recognize objects using basic library operators
2. Recognize angle of object’s tilt
3. Capture video stream from web-camera
4. Create graphical output indicating expected changes

We are going to use two basic libraries to create computer-vision script: NumPy and OpenCV.

**NumPy** is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

**OpenCV** (Open Source Computer Vision Library) is an open-source library that includes several hundreds of computer vision algorithms. Originally developed by Intel. The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

**IMutils** is also required to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges etc.

All needed packages are installed with pip manager:

pip install numpy

pip install opencv-python

pip install imutils

### Recognizing object on image using basic CV operators

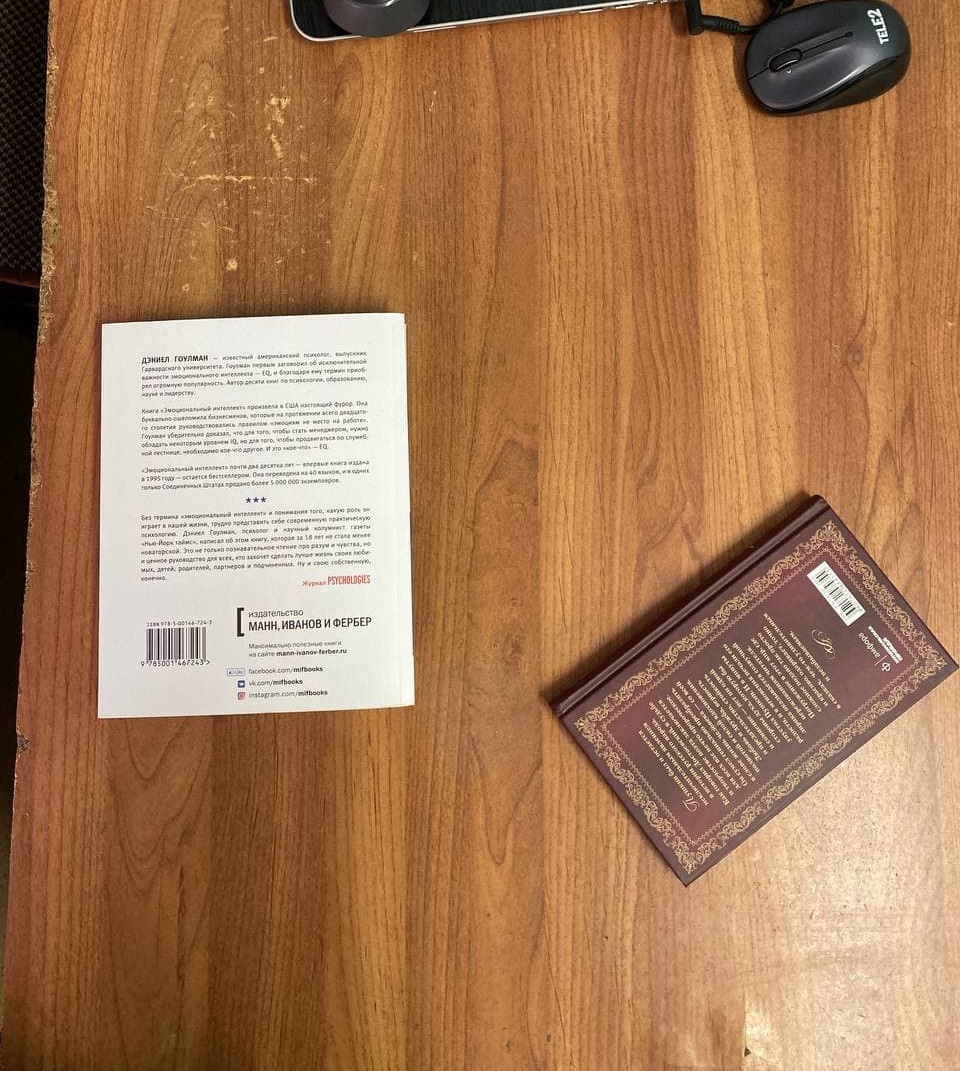


Figure 1. Original static image

First we read image from static file, convert it to grayscale and blur to get rid of high-frequenced noises (figure 2):

gray = cv.cvtColor(image, cv.COLOR\_BGR2GRAY)

gray = cv.GaussianBlur(gray, (15, 15), 0)

cv.imwrite("output/gray.jpg", gray)



Figure 2. Blurred image

Sobel operator (figure 3) showed more recognizable edges than Laplace operator (figure 4).



Figure 3. Result of Sobel operator



Figure 4. Result of Laplace operator

Get edged image using Canny operator:

edged = cv.Canny(gray, 10, 250)

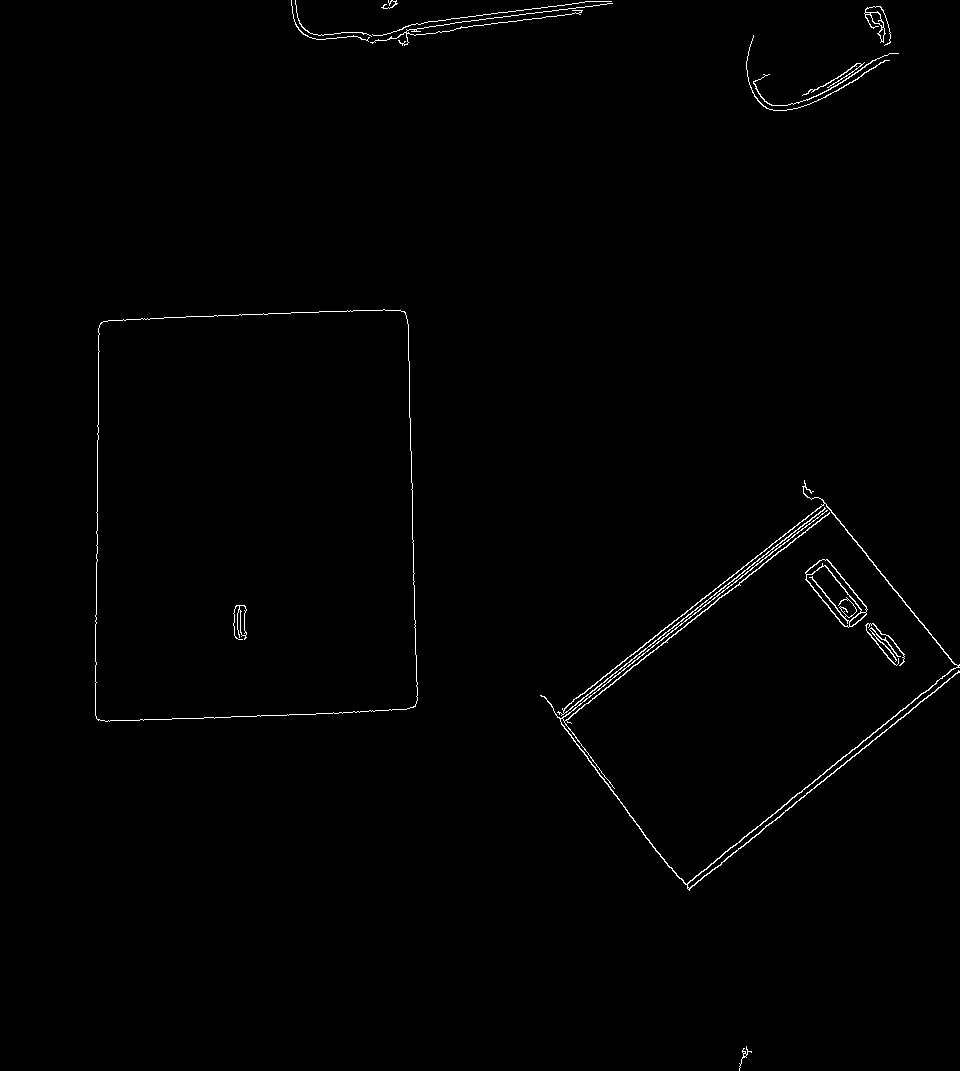


Figure 5. Result of Canny operator

Make obtained contours closed:

kernel = cv.getStructuringElement(cv.MORPH\_RECT, (7, 7))

closed = cv.morphologyEx(edged, cv.MORPH\_CLOSE, kernel)

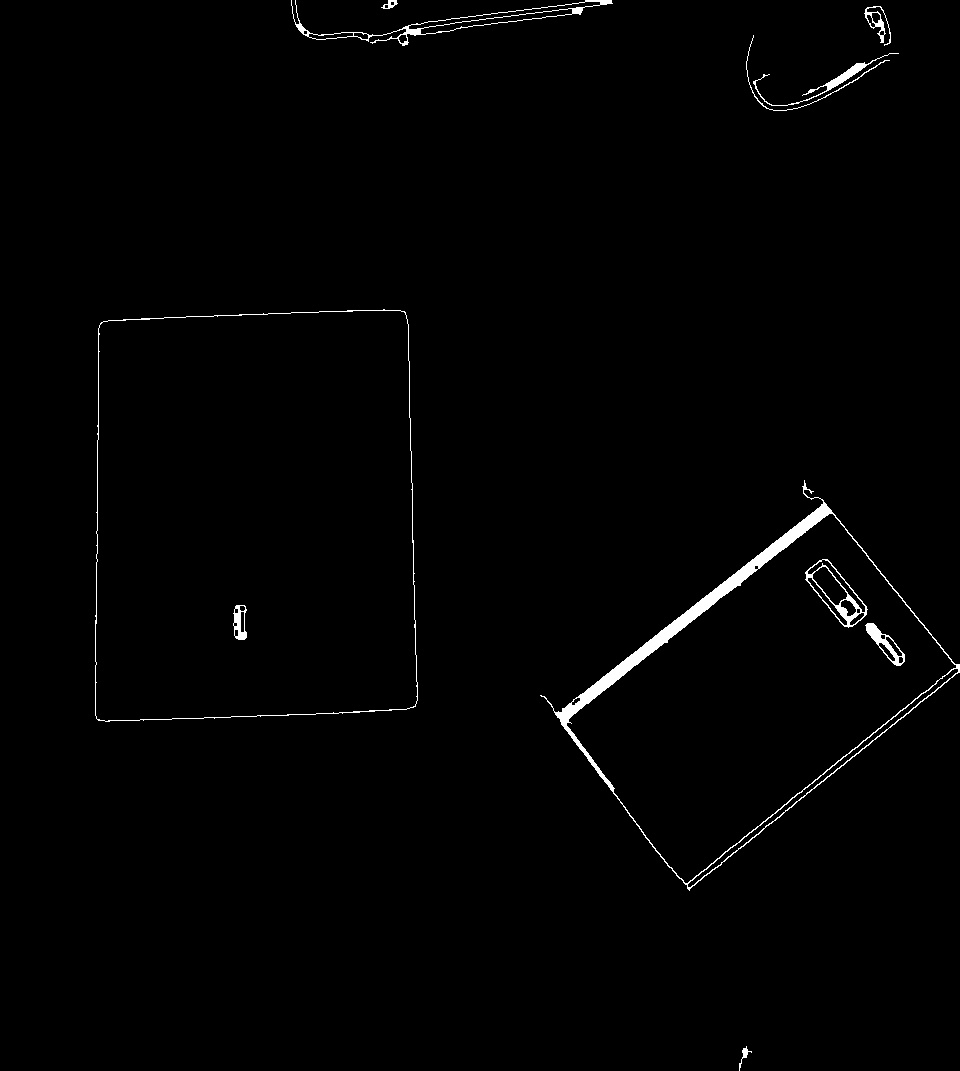


Figure 6

Find separate contours with IMutils:

cnts = cv.findContours(closed.copy(), cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_SIMPLE)

cnts = imutils.grab\_contours(cnts)

Then approximate each of found contours to closed polygon, approximate central line for ones that have 4 vertices, and calculate its tilt angle.

for c in cnts:

    peri = cv.arcLength(c, True)

    approx = cv.approxPolyDP(c, 0.1 \* peri, True)

    if (len(approx) in [4]):

        cv.drawContours(image, [approx], -1, (0, 255, 0), 4)

        rows,cols = image.shape[:2]

        [vx,vy,x,y] = cv.fitLine(c, cv.DIST\_L2, 0, 0.0001, 0.0001)

        lefty = int((-x\*vy/vx) + y)

        righty = int(((cols-x)\*vy/vx)+y)

        line\_start = (cols-1,righty)

        line\_end = (0,lefty)

        cv.line(image, line\_start, line\_end, (0,255,0), 2)

        angle\_pi = (righty - lefty)/(cols-1)

        angle = int(np.arctan(angle\_pi)\*180/np.pi)

        text\_start = (int(x[0]),int(y[0]))

        rect = np.array( [[[text\_start[0]-10,text\_start[1]],[text\_start[0]+50,text\_start[1]],[text\_start[0]+50,text\_start[1]-30],[text\_start[0]-10,text\_start[1]-30]]], dtype=np.int32 )

        cv.fillPoly(image, rect, (255, 255, 255), cv.LINE\_4)

        cv.putText(image, str(angle), text\_start, cv.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, (0,0,255))

Central line and value of its angle are put to output image using OpenCV (figure 7).

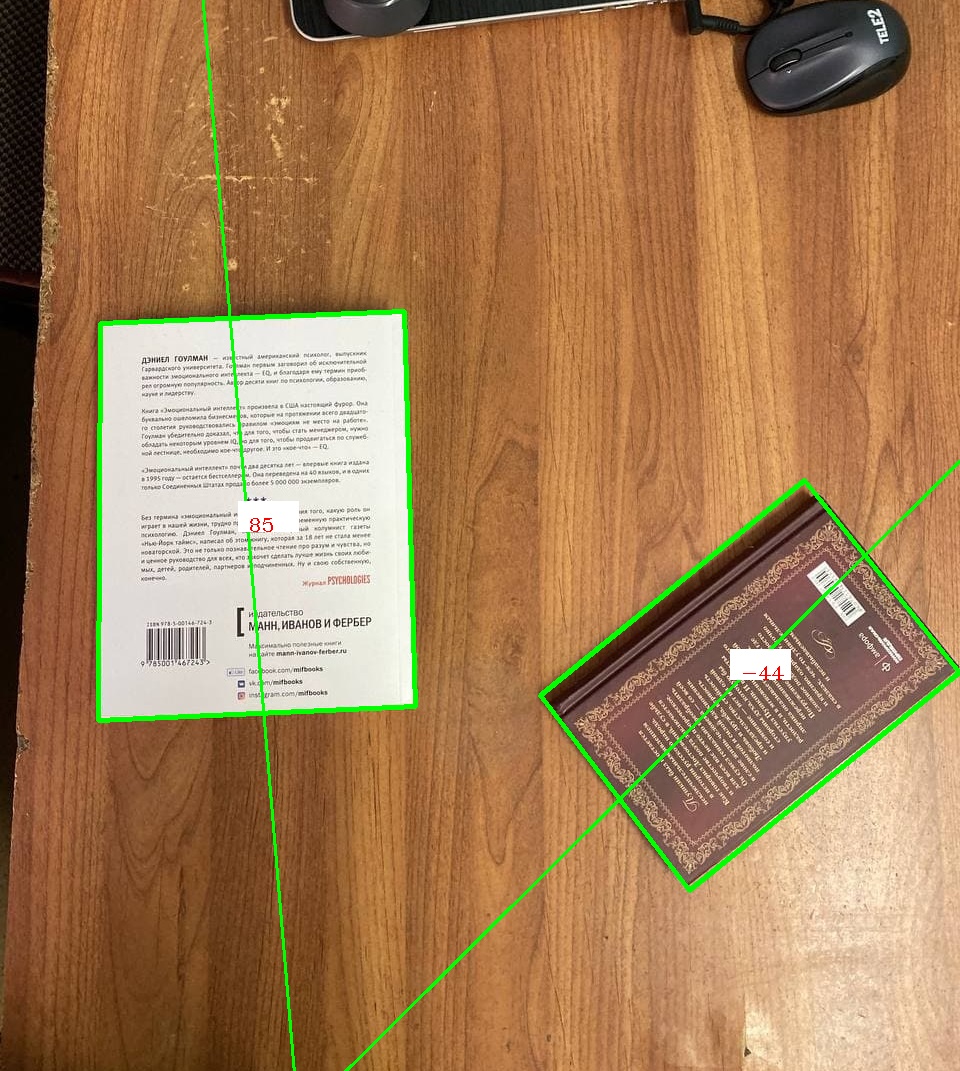


Figure 7. Output image