Document Scanner

The python script prepared using openCV and numpy functions can be used to convert the image of a document into a cleaner and sharpened version.
We first resize the image, apply gaussian blur on it and use a canny edge detection algorithm to identify the edges present in the image. The image is accentuated via dilation and white noises are removed via erosion. After this, the contours are detected and the largest contour is used for applying Warp perspective to the image to remove the skewness present in the image and for the final effect, we have used adaptive thresholding coupled with bitwise not operation.

1.RESIZE

We have used the function cv2.resize to convert all the images to dimension 480X640.

2. GRAY IMAGE

Function used- cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
A digital image is composed of groups of three pixels with colors of red, green and blue (RGB). Each channel also contains a luminance value to determine how light or dark the color is. To get a grayscale image, the color information from each channel is removed, leaving only the luminance values, and that is why the image becomes a pattern of light and dark areas devoid of color, essentially a black and white image.
The way it is done is: The 3 channels of the colored image(pixel values for Red,Green and Blue) are separated,
The values in the channel corresponding to red are scaled by 0.299
The values in the channel corresponding to green are scaled by 0.587
The values in the channel corresponding to blue are scaled by 0.114
The values of these 3 channels are summed up to form the pixel values of the final gray image which has a single channel only.

3.BLUR IMAGE

It is implemented with the help of the cv2.gaussianblur() function.We should specify the width and height of the kernel which should be positive and odd. We also should specify the standard deviation in the X and Y directions, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as equal to sigmaX. If both are given as zeros, they are calculated from the kernel size. Gaussian filtering is highly effective in removing Gaussian noise from the image.
It is a method of blurring an image through the use of a Gaussian function.
If the Gaussian distribution is superimposed over a group of pixels in an image, it is apparent looking at the graph, that if we took a weighted average of the pixel's values and the height of the curve at that point, the pixels in the center of the group would contribute most significantly to the resulting value. This is, in

essence, how Gaussian blur works. While the kernel can technically be an arbitrary size, we should scale σ in proportion to the kernel size. If we have a large kernel radius, but a small sigma, then all of the new pixels we're introducing with our larger radius aren't really affecting the computation.
The Gaussian Kernel is first defined which is then used for convolution with the original image to give the output. We have used a 5x5 Gaussian Kernel.
A Gaussian Kernel of size (2k+1,2k+1) is given by

$$h_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-k)^2+(j-k)^2}{2\sigma^2}\right), 0 \le i, j < 2k+1$$

4. CANNY IMAGE DETECTOR

cv2.Canny(imgBlur,100,100)

As we had studied in one assignment, Canny Edge Detection requires the following steps.

1. Apply Gaussian Filter by using gaussian kernel to smooth the image in order to remove the noise.
2. Find the intensity gradients of the image.
3. Apply Non Maximum Suppression.
4. Apply double threshold to determine potential edges
5. Track edges by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Canny algorithm uses 4 filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operators return a value for the first derivative in the horizontal direction (dx) and the vertical direction (dy).

Two 3×3 kernels are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. From this the gradient magnitude and direction can be determined.

$$\theta = tan^{-1}(dy/dx)$$

$$d = \sqrt{dx^2 + dy^2}$$

The edge direction angle is rounded to one of four directions representing vertical, horizontal and the two diagonals (0°, 45°, 90° and 135°).

An edge direction falling in each color region will be rounded off to a specific angle value, for instance θ in [22.5°,67.5°] or [-157.5°, -112.5°] maps to 45°and θ in [0°, 22.5°] or [157.5°, 180°] maps to 0°.

5. NON-MAXIMUM SUPPRESSION

After computing our gradient magnitude representation, the edges themselves are still quite noisy and blurred, but in reality there should only be *one* edge response for a given region, not a whole clump of pixels reporting themselves as edges.

To remedy this, we can apply edge thinning using non-maxima suppression. To apply non-maxima suppression we need to examine the gradient magnitude d and orientation **θ** at each pixel in the image and:

- Compare the current pixel to the neighborhood surrounding it
- Determine in which direction the orientation is pointing:
  a) If it's pointing towards the north or south, then examine the north and south magnitude
  b) If the orientation is pointing towards the east or west, then examine the east and west pixels
- If the center pixel magnitude is *greater than* both the pixels it is being compared to, then preserve the magnitude; otherwise, set it to zero.

6. DOUBLE THRESHOLD

Even after applying non-maxima suppression, we need to remove regions of an image that are not technically edges, but still respond as edges after computing the gradient magnitude and applying non-maximum suppression.

To ignore these regions of an image, we need to define two thresholds: T_{upper} and T_{lower}.

If $d < T_{lower}$, *not an edge*

If d>T_{upper} , *it is an edge*

If T_{lower}< d <T_{upper}: If the particular pixel is connected to a strong edge, then mark the pixel as an edge, else discard it.

7.DILATION

Kernel of size 5x5 is convolved with the image and the pixel value in the original image is 1 if at least a single pixel is 1 in its kernel.

8.EROSION

After convolution with a 5x5 kernel, the pixel value in the original image will be 1 if all values in its kernel are 1.

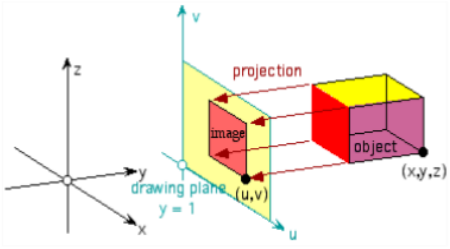
9. CONTOUR DETECTION

We use cv2.findContours to detect contours after applying the Canny algorithm .The function takes 3 parameters, the image in which want to detect the contours, the mode for contour retrieval- used to specify whether we only want outer contours and discard the inner contours or we want all the contours(with/without the hierarchy) and the method to approximate contours - to specify whether we want to store all the boundary points of contours or remove the redundant ones. Lines joining the pixels of same intensity along the image boundaries give rise to contours.

After that we find the largest contour from the contours obtained and this is treated like the outline of the document that is present in the image. We use the coordinates of the largest contour as the outline of our output image.

10. WARP PERSPECTIVE

Perspective transformations- Map point in three spaces to the drawing plane.A projection involves two coordinate systems. A point in the coordinate system of an object to be drawn is given by X=(x, y, z) and the corresponding point on the drawing plane is P=(u, v). x and y correspond to width and depth and z corresponds to height. On the drawing plane, we let u be the horizontal variable and v the vertical.



Projecting an object to the drawing plane

We can measure the distances between pairs of points in the usual way using the Euclidean metric.If X=(x1,x2,x3) and P=(u1,u2) and so on, then:

$$dist(X_1, X_2) = \{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2\}^{1/2}$$
$$dist(P_1, P_2) = \{(u_1 - u_2)^2 + (v_1 - v_2)^2\}^{1/2}$$

The projection from X to P is parallel projection if all sets of parallel lines in the object are mapped to parallel lines on the drawing. Such a mapping is given by transformation, which is of the form f(X)=T+AX where T is a fixed vector in the plane and A is a constant matrix. Parallel projection has ratios preserved. That is if X (1, 2, 3, 4) are collinear points in the object, then the ratio of distances is preserved under parallel projection

$$\frac{dist(X_1, X_2)}{dist(X_3, X_4)} = \frac{dist(f(X_1, X_2)}{dist(f(X_3, X_4)}$$

So in our project what we have done- A perspective transform matrix is obtained by using cv2.getPerspectiveTransform function which returns a 3x3 matrix which is used to map the coordinates of the contour to the border coordinates of the image.

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Where i=0,1,2,3
(x_i , y_i) = Coordinates of largest Contour and (x_i' , y_i') = Border Coordinates of Image
Then when we pass this Map Matrix(M) to the Warp perspective function , the output image (dst(x,y)) is calculated from the initial image (src) via the given transformation.

$$\texttt{dst}(x, y) = \texttt{src}\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right)$$

11. ADAPTIVE THRESHOLDING

In thresholding, we convert an image from color or grayscale into a binary image, i.e., one that is simply black and white.
Adaptive Thresholding: Adaptive threshold computes threshold for a small neighborhood (5x5, 7x7 etc)at a time by using simple gaussian mean and then applies thresholding to that area. The process is repeated for all other neighborhoods until the entire image is processed.
We are using cv2.adaptiveThreshold(source, maxVal, adaptiveMethod, thresholdType, blocksize, constant).
Parameters:
-> source: Input Image array(Single-channel, 8-bit or floating-point)
-> maxVal: Maximum value that can be assigned to a pixel.
-> adaptiveMethod: Adaptive method decides how threshold value is calculated.

12.BITWISE NOT:

The bitwise not function flips the values of the pixels of the image, All pixels with value greater than 0 are set to 0 and pixels with value 0 are set to 255. We have implemented this using cv2.bitwise_not function.