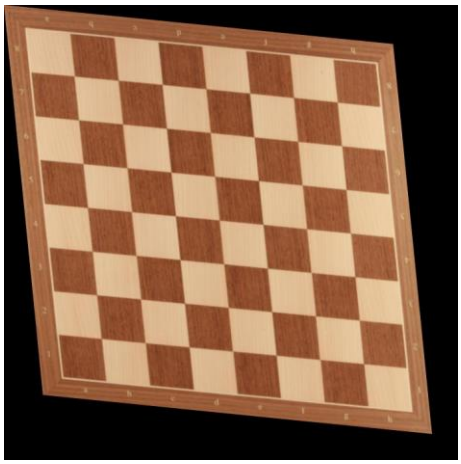
Reflection Essay

I. Affine Transformation:

Given distorted image, distorted.jpg has planar axial shear distortion. Coordinates of chessboard corners in the distorted image in clockwise direction starting from top left corner as origin are [A, B, C, D] = [(0,0), (600,-60),(660,-660),(60,-600)]. These coordiantes have been calculated using crop feature and basic math on dimensions.

The coordinates of the chessboard in the original.jpg(expected after distortion) are [A', B', C', D'] = [(0,0), (600,0),(600,-600),(0,-600)] starting from top left corner as origin in the clockwise direction.

Distorted Image: distorted.jpg (700x700)



Manual:

- To perform un-distortion we have to find the transformation matrix H manually in this case.
- Given the coordinates of a point in an image (x_s, y_s) and the coordinates in the image after transformation (x_d, y_d), the relation between these coordinates is:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = H * \begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix}, \text{ where } H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{pmatrix}.$$

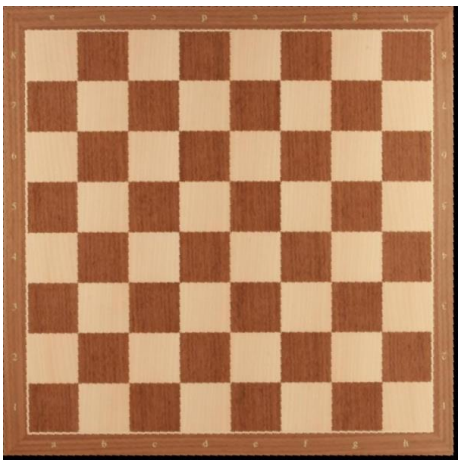
- Here, in our case source coordinates of any point (x_s, y_s) would be the coordinates in the distorted.jpg and the corresponding point in original.jpg would be (x_d, y_d).
- We would require 6 equations to find H i.e. 3 different coordinates in source image $[(x_{s1}, y_{s1}), (x_{s2}, y_{s2}), (x_{s3}, y_{s3})]$and their corresponding points in the desired image $[(x_{d1}, y_{d1}), (x_{d2}, y_{d2}), (x_{d3}, y_{d3})]$.
- We can find the transformation matrix (H) by solving the below shown linear equations (in matrix form):

$$\begin{pmatrix} x_{s1} & y_{s1} & 1 \\ x_{s2} & y_{s2} & 1 \\ x_{s2} & y_{s2} & 1 \end{pmatrix} \cdot \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \end{pmatrix} = \begin{pmatrix} x_{d1} \\ x_{d2} \\ x_{d3} \end{pmatrix} \;\&\; \begin{pmatrix} x_{s1} & y_{s1} & 1 \\ x_{s2} & y_{s2} & 1 \\ x_{s2} & y_{s2} & 1 \end{pmatrix} \cdot \begin{pmatrix} h_{21} \\ h_{22} \\ h_{23} \end{pmatrix} = \begin{pmatrix} y_{d1} \\ y_{d2} \\ y_{d3} \end{pmatrix}$$

- Using cv2.warpAffine() we can reconstruct the original image from distorted image with the transformation matrix as H and flags set to cv2.WARP_INVERSE_MAP.

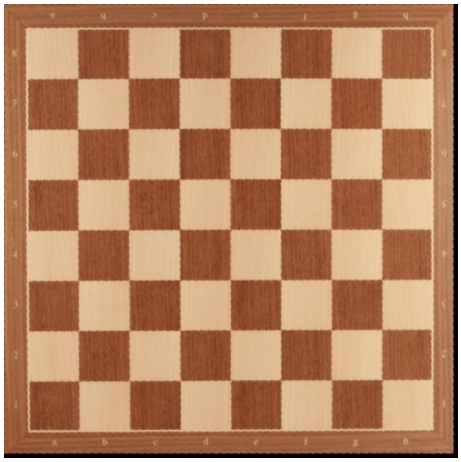
This flag means that the transformation matrix is from source to desired image (dst(x,y)=src($h_{11}x + h_{12}y + h_{13}, h_{21}x + h_{22}y + h_{23}$)) otherwise inversion of H matrix is done and then the image transformation.

- Output:



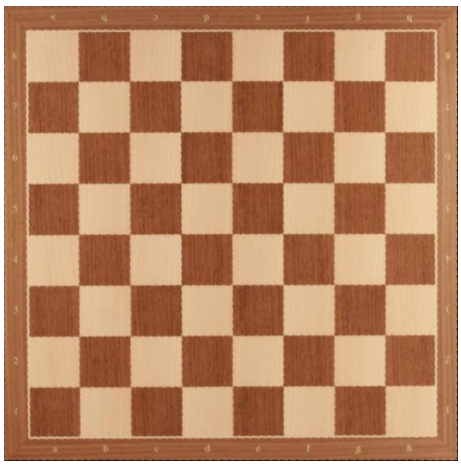
API: To perform un-distortion, we have used cv2.getAffineTransform() to get the transformation matrix H. Then again using cv2.warpAffine() we get the Original image.

Output:



Observation: As you can observe the original image we got using both manual and API methods are same. We have used 3 corresponding points as the function cv2.getAffineTransform() takes 3 corresponding points of type float32.

Also, it is observed that chessboard image is of size 595x595 pixels (approx.) which is shown below (no extra part in the image at the bottom and the right side of the chessboard).



II. Perspective Transformation:

III. Document Scanner:

The goal of this task is to create a document scanner similar to cam scanner apps. It also needs to be automated to detect the contours and perform perspective transformation. Following are the steps which accomplishes the desired task.

Step 1: Resize the image. (This is because contour detection doesn't seem to be working well with larger images.)

Step 2: Convert to grayscale.

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

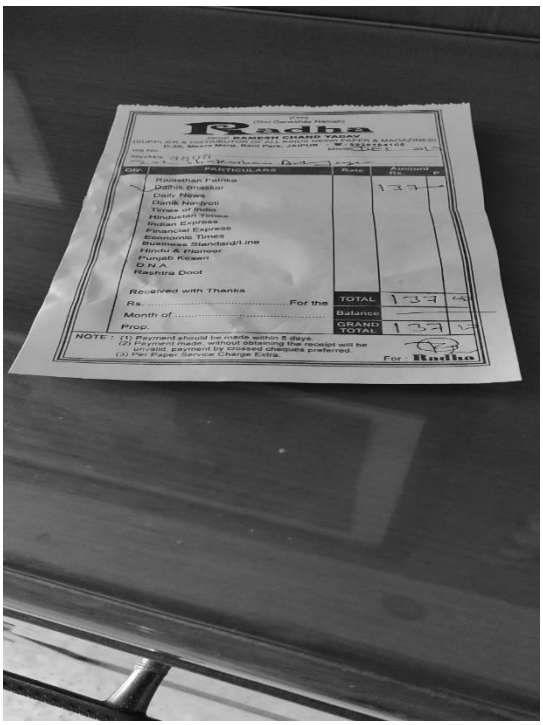


Figure 1 Grayscale Image

Step 3: Add Gaussian Blur.

This is done to reduce noise in image which decreases the performance of edge detection.

```
cv2.GaussianBlur(src, ksize, sigmaX)
```

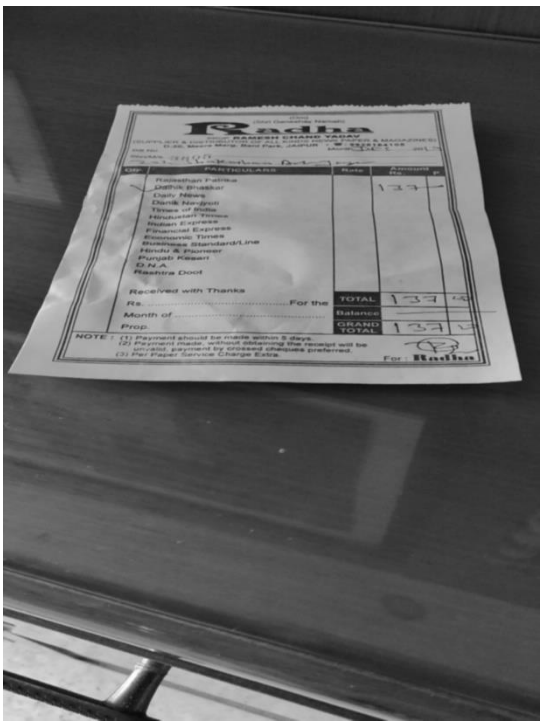


Figure 2 Gaussian Blur

Step 4: Do Canny edge detection to detect the edges.

```
cv2.Canny(src, threshold1, threshold2)
```

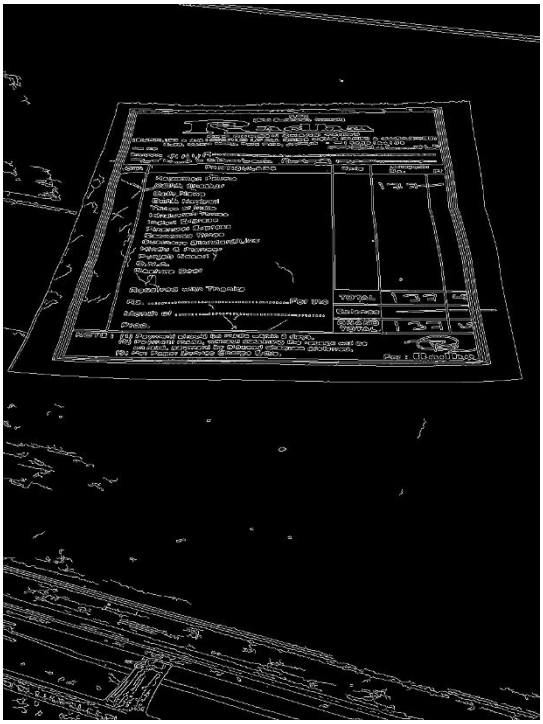


Figure 3 Edge Detection

Step 5: Find contours

```
cv2.findContours(src, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

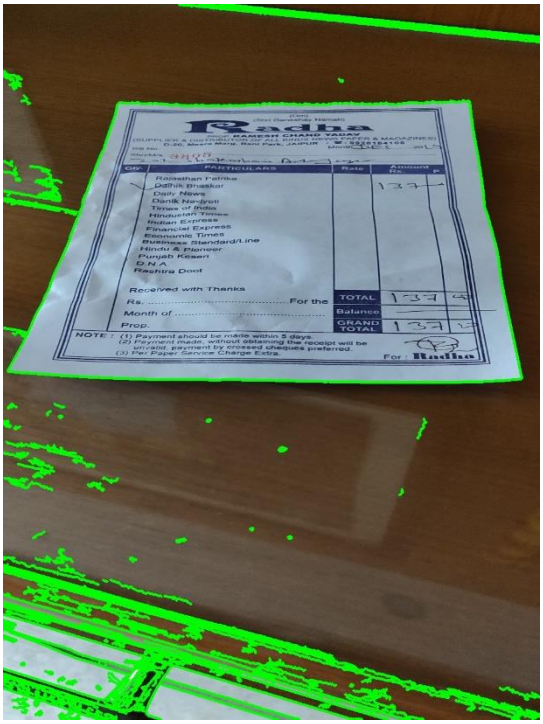


Figure 4 Contours

Step 6: Sort the contours based on area.

```
sorted(contours, key=cv2.contourArea, reverse=True)
```

Step 7: Get the largest quadrilateral.

This is done using `cv2.approxPolyDP` to get minimum points to represent the contour. If the minimum number points is 4 (quadrilateral) the we chose that contour.

Step 8: Reorder the corner points so that they correspond to the order of points of the target.
Step 9: Do perspective transformation.

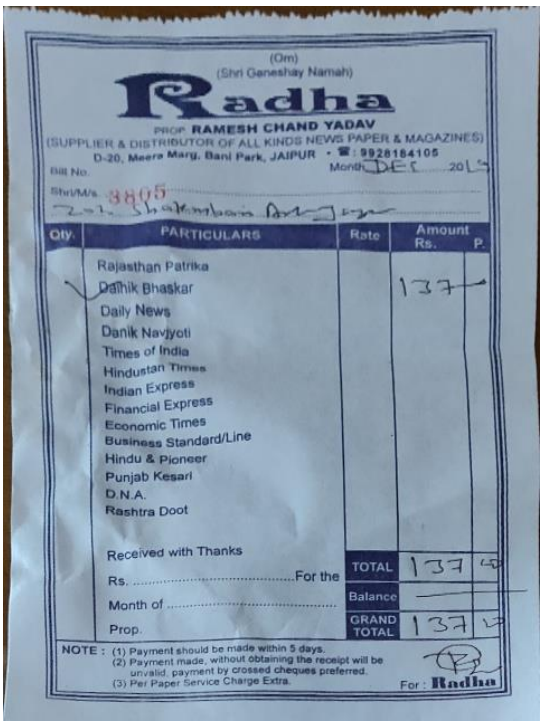


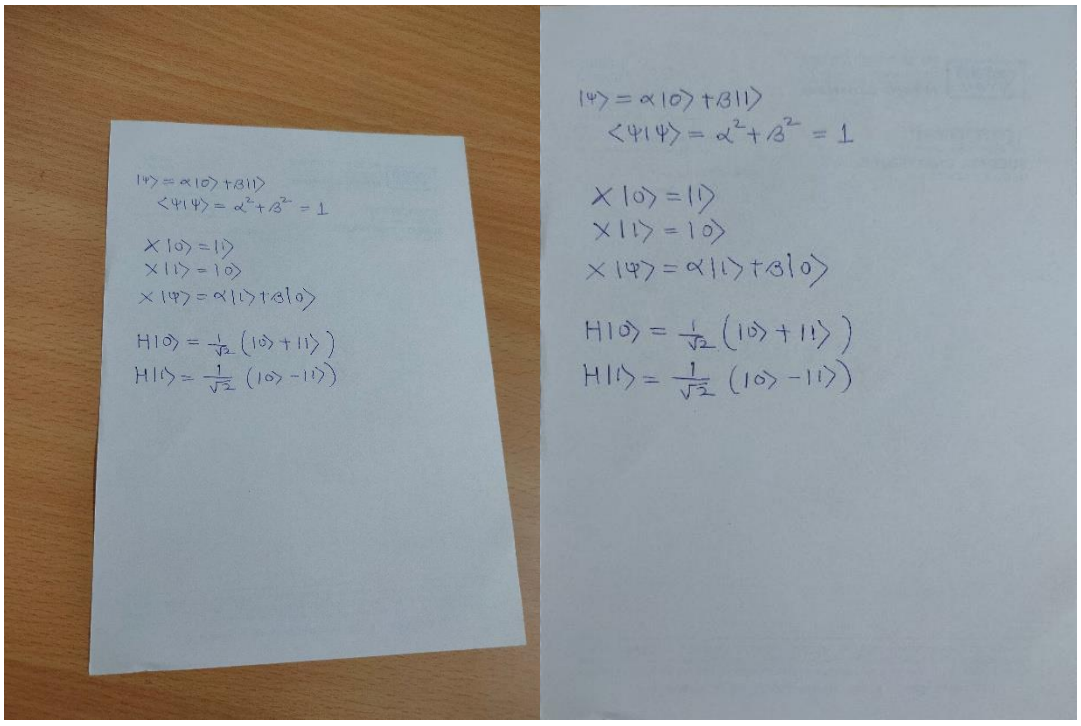
Figure 5 Final Image

Observations:

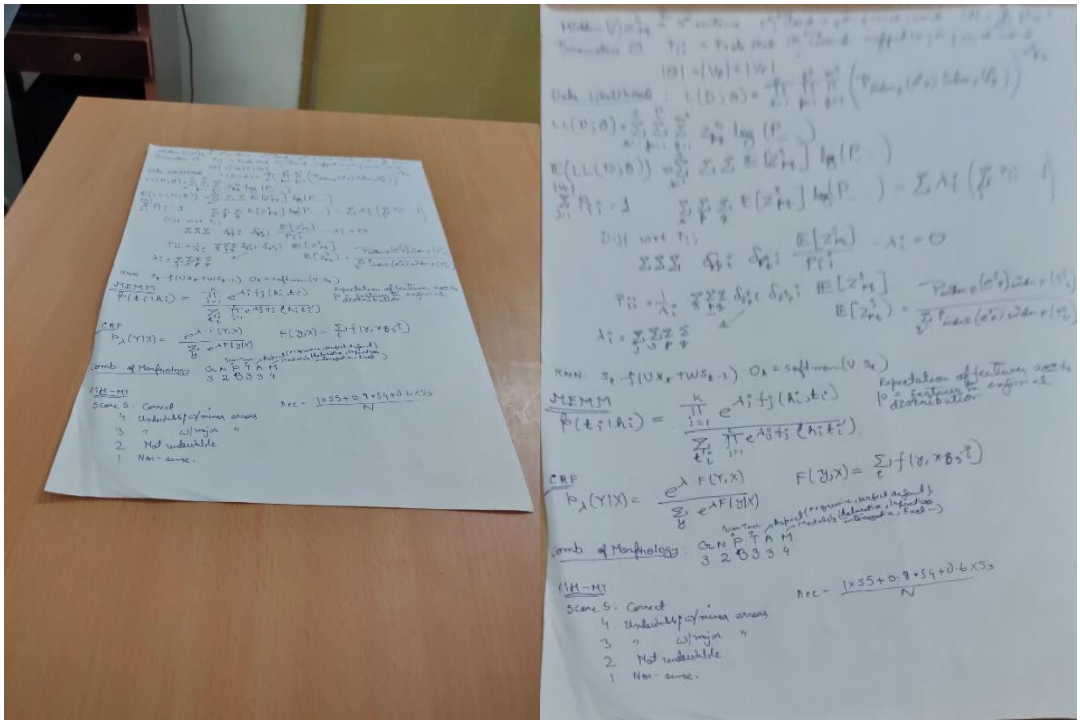
The crucial step for automatic document detection is the canny edge detection step. This step is very susceptible to noise in the image. It was observed that if there is very clear gap between the edges of the document other stuffs in the image. If there is even a little gap between the edge detected by canny edge detection algorithm, then the contour finding step is unable to find the proper contour and hence incorrect output is generated.

Output:

Document 1:



Document 2:



Document 3: