# Image Processing Problem statement using Python-OpenCv

- Niraj Chawale(1744787)

# Problem Statements

1. **Blackened**-Remove the dotted pattern of the table header such that the header background is white and the contents (text) are black.
2. **Low Resolution-**Improve the image sharpness and contrast of the text.
3. **Noise**-Reduce the noise in the image while maintaining the quality.
4. **Perspective-**Fix the perspective of the image such that it is in a front-facing view.
5. **Rotation**-Rotate image by given angle.
6. **Skew**-Deskew the image such that the skew angle is zero and the image is no longer slanted.
7. **Tempered Lines-** Join the broken lines of table and to make them of equal size.
8. **Watermark**- Remove watermark from the image

## General Instructions

All the codes are done using **Python3.8.1** and **OpenCV2.4.1.2**

For all the problem statement there is only **one executable file**. You need to just **run that specific file** and nothing more.

It will find *"source_path"* find all "**.jpg*" images and perform the given task and store the result images in folder name given by *"destination_path".*

# Files to Run

| **Problem statement** | **File to Run** |
|---|---|
| Blackened | Blackened_remove.py |
| Low Resolution | Resolution_Improvement.py |
| Noise | Noise_removal.py |
| Perspective | prespective_correction.py |
| Rotation | Rotation.py |
| Skew | Deskewing.py |
| Tempered Lines | - |
| Watermark | Watermark_removal.py |

## Dependencies

We need to pip install following libraries for the execution purpose.

- OpenCV2
- glob
- os
- Numpy
- Imutils

## Problem Statement 1: **Blackened**

Image is taken as a **BINARY image**.

We use errosion and dilation property.

**First we errode** the image with kernel of size (20,20) pixels. This remove all the text and boarder if any. And what remains is the big black box.

Then **after dilating** the image with same kernel size of (20,20) the original size of box is obtained. Then it is removed with the help of simple if statement.

## Problem Statement 2: **Low Resolution**

We convert given image into standard A4 size image i.e. width 8.27inch and height=11.69inch with given DPI condition of 300.

And then use standard noise removal code as given in problem statement 3.

**NOTE: The folder name is chaged to “LowResolution” to directly access through OS commands.**

## Problem Statement 3: **Noise**

The main problem is that we have paper-salt noise which can be removed by simple average filter on 1. white spots on black and 2. black spots on white bg.

If we apply average or median filter it will blur the edges of text so we use simple **bilateral filter** that will keep edges sharp.

The noise level is small so **kernel of 2,2** is used

We first convert **image to black bg** and remove white noise by **morphological closing** technique.

Then we revert **image to white bg** and remove black noise by **morphological closing** technique.

## Problem Statement 4: **Perspective**

We created a custom code that finds affine transformed rectangle around given text and then use warpPerspective feature of cv2 to correct the perspective.

## Problem Statement 5: **Rotation**

We use OpenCv library **imultis** for this

It rotates the given matrix by the angle value provided keeping anchor point as (0,0). i.e. images are rotated at with respect to top left pixel.

## Problem Statement 6: **Skew**

We use CV2 property to create rectangle around given set of points i.e. **cv2.minAreaRect**

Cv2.minArea creates rectangle around the given co-ordinates and also calculates the angle at which the major axis of image lies.

Once we get the angle we can use warp feature of CV2 to rotate the image at anchor point (center of image) by given angle value.

## Problem Statement 7: **Tempered Lines**

NOT SOLVED

## Problem Statement 8: **Watermark**

In case of watermark  we have watermark whose pixel value lies in between black and white i.e. in between 0-255.

So we multiply pixel value by 2 (alpha) and then subtract it by 100 (beta). Beta is average pixel value of watermark region. This process push watermark region pixel value below zero i.e. it makes it white keeping all the text content on higher scale. Which intact the text data and remove the watermark from the image.