Libraries Used:

- **Scipy**: For obtaining the value of pi up to a certain digits.
- **Numpy**: For operating and modifying array .
- **OpenCV** : For image reading,writing and processing

PI Filter

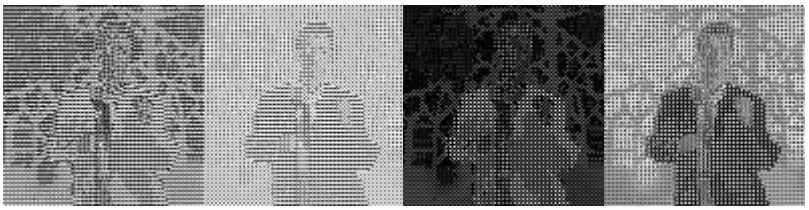
Steps:

1. Compute the size of the image.
2. Generate the digits of pi up to the total number of pixels in the image +1 (as scipy gives rounded values).
3. The intensity values of the *i-th* pixel correspond to **digit × 10**.
4. Compare the actual digits of pi with the extracted digits from the image.
5. Store the mismatched digits in the image in a array called **distorted_digits** .
6. Multiply each digit by 10, take the floor of all values, sort it in decreasing order and reshape them into a **2×2 matrix**, which serves as the required **PI filter**.

Recovering image

Steps:

1. Appling OR, AND and XOR filter to each pixel and get the recover image



Distorted Or And XOR

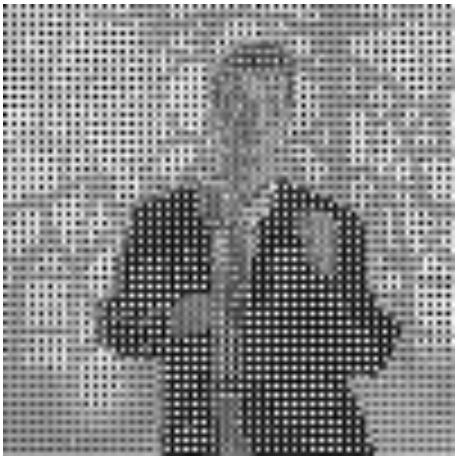
2. From here we can found that image computed after xor is better for further computation.

Template matching

1. Convert the collage intro gray scale image.
2. Iterate through each pixels of the collage.
3. Compute the difference of the image block start from the given pixel in the collage and the recover image we get after filter.
4. Return the coordinate with least difference (required (x,y)).

Password generation

- Restore image matched with the image start from coordinate (100,100).



Matched image from collage **restored image**

Required password is : **floor((100+100)*pi) = 628**

Secure document obtained by using password: