Introduction

As a part of lab 2, we were given a task to compute Gaze Fixation Density Maps / Wooding maps from the gaze fixation points available in the MexCulture142 dataset. Fig 1 shows the main function that we should have to follow when computing the saliency map.

$$S(I,m) = Ae^{-\left(\frac{(x-x_{0_m})^2}{2\sigma_x^2} + \frac{(y-y_{0_m})^2}{2\sigma_y^2}\right)}$$

Fig-01

Wooding's Approach: In Wooding's method, the Gaussian spread (σ) is fixed at an angle (α) of 2 degrees. This is based on the imitation of the human eye's fovea, which covers an area of 1.5 to 2 degrees of the retina's diameter. The formula for calculating σ is:

According to my research I will be using 1 as a degree.
σ = R.D.tan(α)

Methodology

According to the given data most important thing is calculating sigma because it gives more affect to create gaussian distribution and to calculate the sigma for each image the ratio is computed.

Ratio = if the image width *screen height /screen_height is equal or less that screen width the ratio for sigma is image height/screen height. If not, the ration will be image width/ screen with.

```
if (image_width*screen_height)/screen_height<=screen_width:
    ratio =image_height/screen_height
else:
    ratio =image_width/screen_width
sigma =ratio *R *D * tan_alpha
```

Fig2

Saliency map = After iterating all the images according to fig1 equation **Saliency map** has been computed for that all the corresponding fixation file with there names are computed.

```
x0, y0 = fixation_point
x, y = np.meshgrid(np.arange(image_width), np.arange(image_height))
gaussian = np.exp(-((x - x0) ** 2 + (y - y0) ** 2) / (2 * sigma ** 2))
saliency_map += gaussian
```

Fig3

Saliency normalization
Most importantly, Saliency map has been to normalize and it has been done by dividing form maximum value of the Saliency map.

Evaluation matrix

1. PCC / **Pearson Correlation Coefficient**
    - *High Positive PCC*: It indicates a strong positive linear relationship between the two images. In the context of comparing images, this suggests that the two images are highly correlated, meaning they are similar in terms of intensity or color distribution.
    - *High Negative PCC*: It indicates a strong negative linear relationship between the two datasets. In the context of comparing images, this suggests that the two images are strongly negatively correlated, meaning they are dissimilar in terms of intensity or color distribution but have an inverse relationship.
    - *PCC Near Zero*: It suggests a weak or no linear relationship between the two datasets. In the context of image comparison, this means that the two images are not strongly correlated, indicating differences in intensity or color distribution.
2. MSE **(Mean Squared Error):**
    - Low MSE: A low MSE value indicates that the two datasets are very similar. In the context of comparing images, a low MSE means that the two images are nearly identical pixel by pixel.
    - High MSE: A high MSE value indicates that the two datasets are dissimilar. In image comparison, a high MSE suggests that there are significant differences between the two images.

Result

After computing saliency map using colored map is printing using some technique like following.

Alpha = transparency or blending factor when overlaying the colorized saliency map onto the original image. It determines how much of the colorized saliency map is blended with the original image. A value of 0.7 for alpha means that 70% of the colorized saliency map will be visible, and the remaining 30% will show the original image underneath.

The cv2.addWeighted function is used for linearly blending two images together. In this case, it combines the colorized saliency map and the original image using the specified alpha value.
And the gray image multiplying the saliency map by 255, it can be scaled to values 8-bit range so that they can be properly displayed as grayscale values or applied as colors using a colormap.

```python
alpha = 0.7
overlay = cv2.addWeighted(np.array(image), alpha, colormap, 1 - alpha, 0)

# Save the resulting color-mapped image to the output folder
output_image_path = os.path.join(output_folder, os.path.basename(image_filename))
cv2.imwrite(output_image_path, overlay)

# Save the resulting grayscale saliency map to the gray output folder
output_image_path_gray = os.path.join(output_folder_gray, os.path.basename(image_filename))
cv2.imwrite(output_image_path_gray, (saliency_map * 255).astype(np.uint8))
```

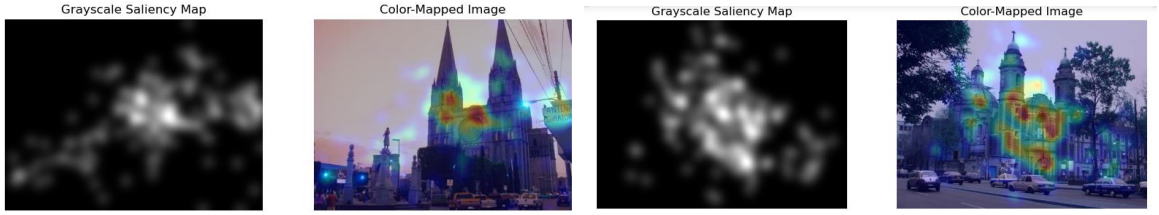

Fig4

Some worst result

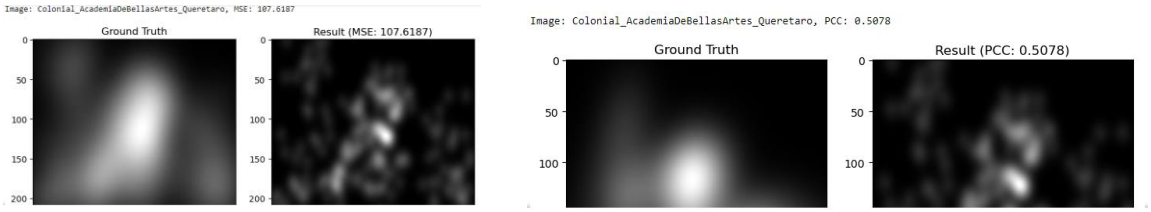

Fig5

Some Good results

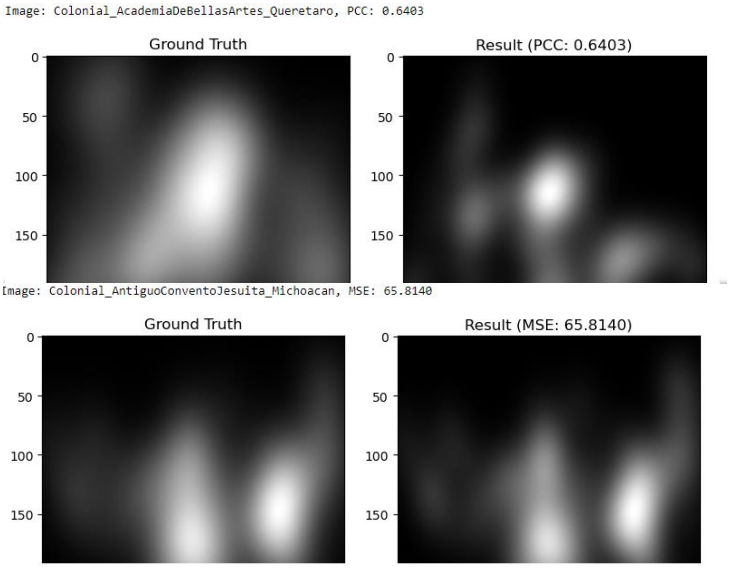

Fig6

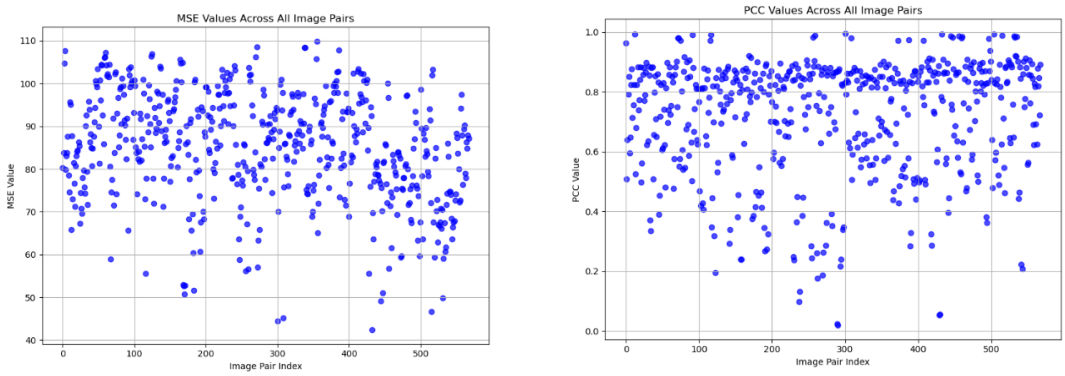

Fig7

(Graphical representation of comparison)

Discussion

According to the given data I tried to compute Gaze Fixation Density Maps / Wooding maps and compare with given ground truth using MSE and PCC. Most importantly I have noticed the given ground truth and the result that I computed are not well match in fact I have discovered large MSE values and lower PCC values from the dataset.

Reference

https://en.wikipedia.org/wiki/Saliency_map