HTTPS://GITHUB.COM/ANUSHKAWAKANKAR/POISSON-MATTING

POISSON MATTING BY GIRL POWER

Team Members:
Sagrika Nagar 20171204 (ECE)
Gowri Lekshmy 20171053 (ECD)
Anushka Wakankar 20171112 (CSE)

Mentor : Prathyakshun Rajashankar

Presentation Outline

- Introduction to the Problem Statement
- Our approach
- Problems faced
- Final results

How do we change the backgrounds of images?

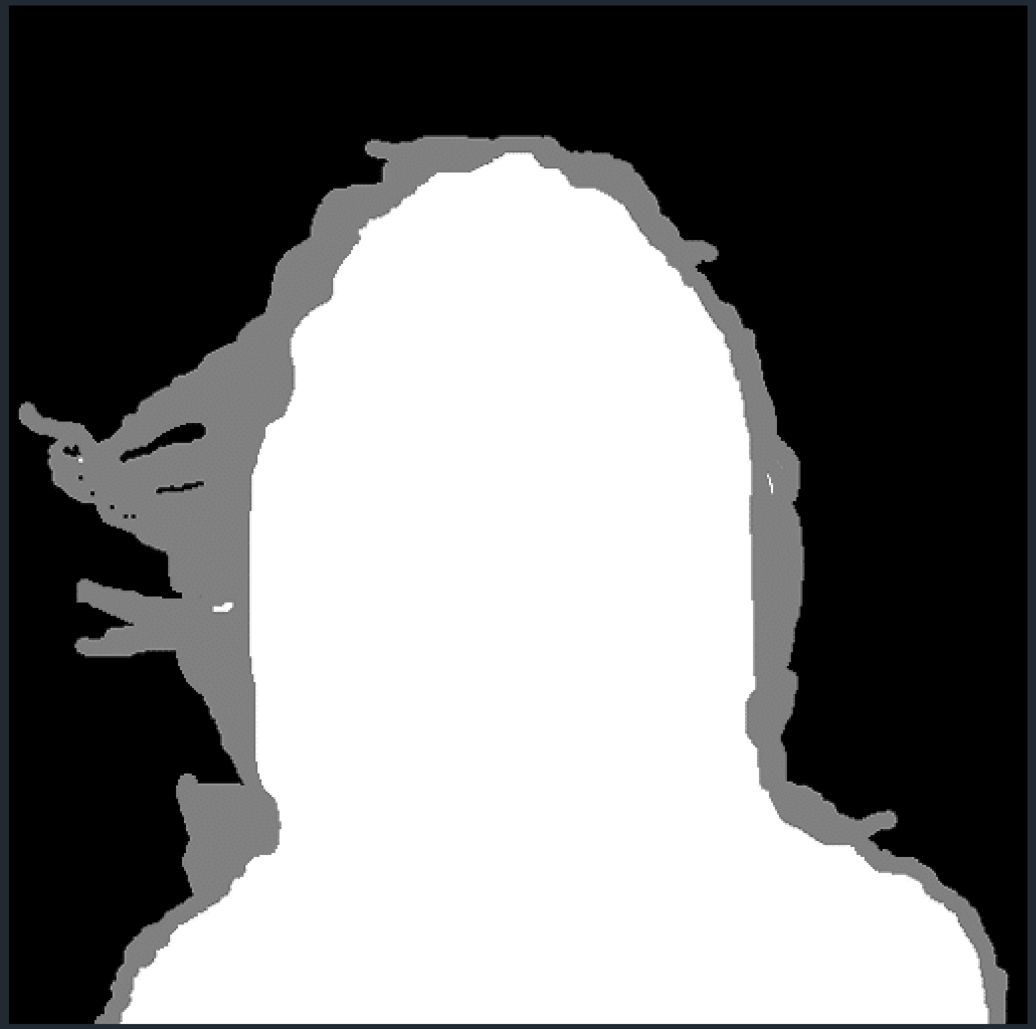
WE HAVE BEEN GIVEN AN IMAGE AND ITS TRIMAP. NOW WE NEED TO CHANGE THE BACKGROUND OF THE IMAGE. BUT FIRSTLY, WHAT IS A TRIMAP?

THE INPUTS

ORIGINAL

TRIMAP

How to generate an alpha matte?



TRIMAP

ALPHA MATTE

Estimating FG, BG, unknown regions and alpha

- Obtain the grayscale image from the original coloured image
- Inpaint the image to get an approximate fg and bg and then calculate their difference.
- Calculate an estimated alpha matte by giving a weight of 1 to foreground regions, 0 to background regions, and 0.5 to the unknown regions.

Generating the gradient matte

- Using the approx difference, and the second derivative of the intensities of the grayscale image, we calculate alpha gradient by the formula

$$\Delta\alpha = \mathrm{div}(\nabla I)/(F - B)$$

where Δ is the laplacian operator, and div is the divergence operator

Generating the alpha matte

- Next, we generate the alpha matte by solving the laplacian

$$(\nabla^2 u)_{ij} = \frac{1}{\Delta x^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}) = g_{ij}$$

- To solve this, we use the Gauss-Seidel equation

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right), \quad i = 1, 2, \ldots, n.$$

Now we have an alpha matte

WHY NOT JUST USE CHROMA-KEYING?

We now know which pixels are background and which ones are foreground. Why not just replace the ones in the background with a different image?

THIS IS WHAT HAPPENS...

Instead, we use this equation

$$I = \alpha F + (1-\alpha)B$$

where F is the foreground image, and B is the background image

USING THE PREVIOUS EQUATION

Comparing the input and output

Some more examples

Some more examples

Some more examples

Some more examples

Using Local Poisson Matting

- To refine our results further, we can use Local Poisson matting instead of global matting.
- The user would have to manually select a problem area. Applying the above operations on the region selected results in a better localized matte, and hence a better image.
- The altered equation for local poisson matting is

$$\nabla \alpha = A(\nabla I - D)$$

where $A = 1/(F - B)$ and $D = [\alpha \nabla F + (1-\alpha)\nabla B]$.

After Local Poisson Matting

Alpha matte after local matting

Alpha matte after global matting

Boosting Brush

- This is a refining method, which is used when the resulting image is smoother or sharper than expected.
- A boosting brush is applied to increase or decrease A (from the local matting equation) directly
- The modification is as follows -

$$A'_p = [1 + \lambda \exp(-\frac{\|p - p_0\|^2}{2\sigma^2})] \cdot A_p$$

where p0 is the coordinate of the brush center, σ and λ are user defined parameters to control the size and strength of the effect

High Pass Filtering

- To obtain a clearer alpha matte structure, we can apply High Pass Filtering.
- We use the following formula-

$$D = K * \nabla I$$

where K = = N(p; p0,σ^2) is a Gaussian filter centered at p0 and ∗ is the convolution operator.

The new alpha matte is now -

$$\nabla \alpha = A(\nabla I - K * \nabla I)$$

Alpha Matte after refining

After Refining

Before Refining

Other Refining Methods

Some other methods which can be used to refine the images are -

- Diffusion filtering
- Clone brush
- Channel selection

The user can use these methods depending on the results desired and the inputs given.

Distribution of work

ANUSHKA

- Taking inputs and making preliminary masks
- Chroma keying
- Initial local matting operations

SAGRIKA

- Reconstructing alpha matte using Poisson equations
- Alpha blending
- Local matting by user selection

GOWRI

- Generation of estimate alpha matte
- Optimization of the code
- Trials with refinement

THANK YOU!