Assignment 3: Bayesian Matting

Due: 02/11/2020

The problem this method is trying to solve is the Image Matting problem, i.e. extracting the foreground of image from the background image by estimating the opacity for each pixel of the foreground element. The set of values of these opacities is the alpha matte.

The Bayesian matting approach models this problem by using spatially varying set of gaussians, and assumes a fractional blend of foreground and background colours to produce the final output. It then, uses a maximum likelihood criterion to estimate the opacity, foreground, and the background simultaneously.

The following is the basic matting equation –

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

Where α is the opacity of the pixel, F is foreground pixel and B is background pixel. Using this equation, we come to the following optimising equation –

$$\arg max_{\alpha,F,B}\ P(F, B, \alpha \mid I)$$

In simple words, we want to maximise the probability of getting the foreground, background and the α given the image. On simplifying the equation, using the Baye's rule –

$$\arg max_{\alpha,F,B}\ \frac{P(I \mid F, B, \alpha).P(F, B, \alpha)}{P(I)}$$

Since P(I) is constant, we can ignore the denominator in maximisation problem. Further, we take the logarithm of the equation –

$$arg\ max_{\alpha,F,B} \log\left(P(I \mid F, B, \alpha)\right) + \log\left(P(F, B, \alpha)\right)$$

Now assuming that the foreground, background and the alpha are not statistically coupled, meaning, there are independent, then the second term in the logarithm can simply be decoupled into separate terms using the independence property.

$$arg\ max_{\alpha,F,B} \log\left(P(I \mid F, B, \alpha)\right) + \log\left(P(F)\right) + \log\left(P(B)\right) + \log\left(P(\alpha)\right)$$

Now, using the matting equation, we make as assumption of the first term in the of the above expression as below –

$$P(I \mid F, B, \alpha) = \exp\left(-\frac{||I - (\alpha F + (1 - \alpha)B)||^2}{\sigma^2}\right)$$

So, this equation says that, closer the output of the prediction with the original image, higher the probability, and vice versa.

The rest of the terms are obtained from the known inputs. In this case, this information is provided with the help of trimap. It is an image containing the prior information, about pixels that are certainly in the foreground, pixels that are certainly in the background, and the pixels, that are unknown.

Now we assume that the alpha is constant and the foreground and the background are gaussian PDFs. Using these assumptions, we can calculate closed form solutions for F, B and alpha by solving the equations –

$$\begin{bmatrix} \Sigma_F^{-1} + I\alpha^2/\sigma_C^2 & I\alpha(1-\alpha)/\sigma_C^2 \\ I\alpha(1-\alpha)/\sigma_C^2 & \Sigma_B^{-1} + I(1-\alpha)^2/\sigma_C^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \Sigma_F^{-1}\overline{F} + C\alpha/\sigma_C^2 \\ \Sigma_B^{-1}\overline{B} + C(1-\alpha)/\sigma_C^2 \end{bmatrix}$$

Now, once solving

$$\alpha = \frac{(C-B)\cdot(F-B)}{\|F-B\|^2}$$

for alpha, we feed it back to the first set of equation and solve again for alpha. We perform this process iterative till the values converge.

Now, this process is not done on the entire image, but locally. This makes more sense that taking the entire image for every picture. This is because, the pixels that matter the most, are the pixels that are very close to the unknown pixel. Thus we choose a window size to solve for alpha, foreground and background locally.

In the actual images, it is possible that the chosen window size does not contain sufficient information about the foreground and background to solve for alpha. In this situation, we simply skip the pixel for the moment and expect that on solving for other pixels nearby, we soon are able to gather sufficient information to solve for the pixel we left earlier. In case, this does not happen, even after we solve for all the other pixels in the image, we then need to increase the window size of consideration. Thus, eventually, this algorithm is bound to converge with the increasing window size.

This has certain drawbacks, as larger the window size, the more the number of pixels we are considering for solving for alpha. Practically we observe that the amount of visible smudges on the alpha matte are significantly increasing.

There are several tunable parameters in this algorithm. Some of them are the window size, sigma of the fitten gaussians, clustering algorithm used, number of clusters allowed, minimum number of neighbourhood pixels needed for solving for alpha, and many more. For different images, all these parameters need to be optimized to get the best results. Thus, one set of parameters may not produce the best results for all images. For some images, I have restricted the size of the window, so as to prevent unwanted smudging on the images, and in others, I have changed the variance values so as to get better images.

On the next page, I have attached some of good results using this algorithm. All other images with their scores are submitted along in the zip file.

Predicted mattes

Sum of absolute difference: 11.46

Sum of absolute difference: 29.07

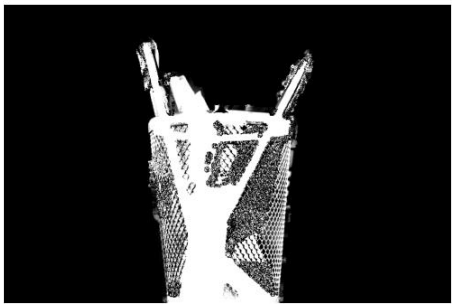
Sum of absolute difference: 14.93

Ground Truth

Predicted mattes

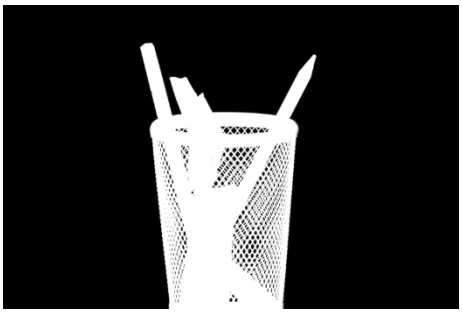
Sum of absolute difference: 18.75

Sum of absolute difference: 10.81



Sum of absolute difference: 11.29

Ground Truth



Note: All other images are attached along with this report in the zip file.

Note:

The score has been calculated using the following methodology. When the algorithm return the value of alpha matte, the matrix is multiplied by 255, so as to form a visual alpha matte. Next, this multiplied matte is saved as a PNG file. On doing this, the pixels are in floating point numbers are rounded off the the nearest pixel values. After this a separate code reads all the alpha matte, along with the ground truth, and then returns the sum of absolute difference.

However, the saving of the alpha matte as an image, leads to some unknown errors. If the alpha matte after multiplication with 255 (without saving as an .png file), is directly compared with the ground truth, error seems to be significantly lower than the other case. The cause of this behaviour is beyond the scope of this assignment, and therefore I have followed the methodology as mentioned in the above paragraph for consistency.

References:

1. Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A Bayesian Approach to Digital Matting. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR 2001)*, Vol. II, 264-271, December 2001

Link: https://grail.cs.washington.edu/projects/digital-matting/image-matting/

2. Youtube Video: https://bit.ly/3jNcbwX

3. Github Repository: https://github.com/MarcoForte/bayesian-matting