Assignment 1 — Foreground Segmentation and Poisson Blending

1. **Introduction:**

On this assignment we implemented foreground segmentation using GrabCut algorithm and used Poisson Blending technique to blend in cut objects with given images.   
In most cases, our results were good with high precision using the GrabCut algorithm. The main issues were caused by similar color palettes between the background and foreground, as well as initial masks that included too much of the background.

When using the Poisson Blending algorithm, we found that the best results occurred when the subject's color palette was similar to the target image.

1. **GrabCut Algorithm:**Our implementation of the GrabCut algorithm uses the KMeans and GMM algorithms as implemented in sklearn. KMeans is used to set initial values for the GMM to achieve low variance Gaussian components. Additionally, we use numpy for most GMM model calculations and the igraph library for the mincut algorithm.   
     
   We achieved an average convergence time of \*\*\* minutes for the given images, with an average accuracy score of \*\*\*\*. While most images produced good cut results in a reasonable time, a few did not converge as quickly and achieved poor results.
   1. **Algorithm results:**

Table 1 - GrabCut results

|  |  |
| --- | --- |
| Image name | Result |
| banana1.jpg |  |
| banana2.jpg |  |
| book.jpg |  |
| bush.jpg |  |
| cross.jpg |  |
| flower.jpg |  |
| fullmoon.jpg |  |
| grave.jpg |  |
| llama.jpg |  |
| memorial.jpg |  |
| sheep.jpg |  |
| stone2.jpg |  |
| teddy.jpg |  |

As can be seen in the table, most images had good results with both Accuracy and Jaccard value above 95%.

* 1. **Failure cases:**The GrabCut algorithm tends to produce unsatisfactory results when the subject in the given image is too close in color to the background as can be seen in the example image 'banana1.jpg'.  
     Also, when the initial bounding box around the subject includes a large portion of the background, the time until convergence increases. This issue is especially critical when the object's shape is not well contained within a small rectangle.
  2. **Effects of Blur:**we compared the effects of different blur intensities on the GrabCut algorithm. In our test we used two types of blur kernels: and of 1's. We also used gaussian blur filter which produced similar results to the bigger mask, so it is not displayed here. We found that \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*. This is compatible with logic since \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*. Actual findings are presented in *Table 1.*

Table 2 - Effects of blur

|  |  |  |  |
| --- | --- | --- | --- |
| Image name | Blur | Accuracy | Iteration until converge |
| Image1 | No blur |  |  |
| Low blur |  |  |
| High blur |  |  |
| Image 2 | No blur |  |  |
| Low blur |  |  |
| High blur |  |  |
| Image 3 | No blur |  |  |
| Low blur |  |  |
| High blur |  |  |

* 1. **Effects of GMM components count:**we compared the effects of GMM components count on the GrabCut algorithm. We found that \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*. Actual findings are presented in *Table 2.*

Table 3 - Effects of GMM components count

|  |  |  |  |
| --- | --- | --- | --- |
| Image name | Components | Accuracy | Iteration until converge |
| Image1 | 2 |  |  |
| 5 |  |  |
| 10 |  |  |
| Image 2 | 2 |  |  |
| 5 |  |  |
| 10 |  |  |
| Image 3 | 2 |  |  |
| 5 |  |  |
| 10 |  |  |

* 1. **Effects of Bounding box initialization:**we compared the effects of initial bounding box on the GrabCut algorithm. We found that a tighter box around the object produce better results and improves the time until convergence. Also, the accuracy increases for same amount of iterations. Actual findings are presented in *Table 3.*

Table 4 - Effects of box initialization

|  |  |  |  |
| --- | --- | --- | --- |
| Image name | Box | Accuracy | Iteration until converge |
| Image1 | Tight box |  |  |
| Loose box |  |  |
| Image 2 | Tight box |  |  |
| Loose box |  |  |
| Image 3 | Tight box |  |  |
| Loose box |  |  |

1. **Poisson blending:**

Our implementation of the Poisson Blending algorithm follows the formula discussed in class. We compute the Laplacian operator (matrix A) and the current state vector (vector b), and then solve the Poisson equation using the scipylibrary. Our results vary in quality; some appear quite realistic and plausible, while others seem mismatched with both the source and target images.

* 1. **Blending results:**To present the blending outcomes, we utilized the true masks of the provided images and blended them with the background images. We noticed varying levels of success: results were generally better when using the grass mountains background but less satisfactory when using the table background.  
     Upon observing the outcomes, it becomes apparent that pasting an image onto a light background can cause it to appear unclear or even overexposed. Additionally, as seen in the bush image, the color of the pasted subject seem to be blended with the background color**.**

Table 6 - Poisson Blending results

|  |  |  |
| --- | --- | --- |
| Target image | Source image | result |
| Grass mountains | Bush |  |
| Llama |  |
| Sheep |  |
| Grave |  |
| table | Banana1 |  |
| Teddy |  |
| Banana2 |  |
| Book |  |
| wall | Cross |  |
| Stone2 |  |
| Flower |  |
| Fullmoon |  |
| Memorial |  |

* 1. **Effects of mask tightness:**

We compared the effect of a tight mask around the object vs a simple box by running the algorithm on the sheep image as source and the grass mountains as target. Once using the tight true mask and once with the box for the GrabCut algorithm.  
we found that when using a box around the object, a slight tint is caused in the subject but the image looks a bit better.

|  |  |  |
| --- | --- | --- |
| Target, source | Mask type | result |
| Grass mountains, sheep | True mask |  |
| Box mask |  |
| Table, banana1 | True mask |  |
| Box mask |  |

1. **Appendices:**