Principles of Computer Vision for AI Assignment

University of Malta

Faculty of Information & Communication Technology

Bachelor of Science in Information Technology (Honours) (Artificial Intelligence) 2nd Year

ARI2129 – Principles of Computer Vision for AI

Mr Dylan Seychell

|  |  |
| --- | --- |
| Jake Sant | Francesca Maria Mizzi |
| 117699M | 1118201L |

Table of Contents

[Part 1 – Object Blending 3](#_Toc75623515)

[Implementation 3](#_Toc75623516)

[Evaluation 3](#_Toc75623517)

[Part 2 – Image Inpainting 3](#_Toc75623518)

[Implementation 3](#_Toc75623519)

[Evaluation 3](#_Toc75623520)

[References 5](#_Toc75623521)

# Part 1 – Object Blending

The first part of the assignment consisted of creating a variety of functions in order to manipulate images, the overall goal being to implement a successful object blending function. The first stage consisted of 4 functions which were used in order to extract an object from the image using the provided tasks and then blend the extracted object into a new image. The second stage consisted of 2 functions which removed the green screen of an image and replaced it with a new background.

## Implementation

In order to implement the functions described below, the libraries *numpy* and *OpenCV* were used.

### Stage 1

As described earlier, the first stage of part 1 of the assignment involved extracting an object from an image using the provided masks and blending it into a new image. This was done through the use of 4 functions, ExtractObject(), ApplyFilter(), ObjectBlender() and CompareResults().

In order to perform the above functions, the relevant masks and scenes must first be imported. In stage 1, we made use of two different scenes, a photo of a small ornament elephant and a photo of the same elephant but with a small ornament glass, as seen below.



Figure 2 - Small Ornament Elephant with Small Ornament Glass

Figure 1 - Small Ornament Elephant

A picture containing graphical user interface

Description automatically generatedThe first function implemented is the ExtractObject() function. This function multiplied Figure 2 with the given mask and then performed the bitwise\_not() function from the OpenCV library in order to extract the ornamental glass from the image as seen in figure 3.

Figure 3 - Extracted Ornamental Glass

A picture containing graphical user interface

Description automatically generatedA picture containing graphical user interface

Description automatically generatedThe second function implemented is the ApplyFilter() function. This function has the functionality to apply three filters to the newly extracted image, these filters being Gaussian Blur filter, Histogram Equalization Filter or Median blur. Based on the parameter provided, the function will return the extracted image with the chosen filter.

A picture containing graphical user interface

Description automatically generated

Figure 5 - Extracted Image with Histogram Equalization Filter

Figure 4 - Extracted Image with Gaussian Blur

Figure 6 - Extracted Image with Median Blur

The third function is the ObjectBlender() function which takes the newly filtered image and blends it into a new image, in our case Figure 1. This is done through the OpenCV function addWeighted() using the parameters alpha = 0.65, beta = 1 – alpha and gamma = -60.



Figure 7 - New photo after Object Blending with Median Blur

Figure 8 - New photo after Object Blending with Histogram Equalization Filter

Text

Description automatically generatedThe final function is the CompareResults() function which compares the newly blended photo with the original and returns either the Sum of Squared Distance Error (SSD) or the Mean Squared Error (MSE) based off of the chosen parameter as seen in Figure 9.

Figure 9 - Error results for each filter using both SSD and MSE

### Stage 2

The second stage of the first part of the assignment involved removing the green screen of an image and replacing it with a new background. The original image chosen was that of a brown boot and the new backgrounds added were of some chairs outside, stairs in Valletta and a photo of the night sky as seen below.

Figure 10 - Original Image of Brown Boot

Figure 11 - Background image of chairs outside



Figure 13 - Background image of the night sky

Figure 12 - Background image of Valletta

A person's foot in the air at night

Description automatically generated with low confidenceThe first function implemented was RemoveGreen() which removed the green screen from the original image. This was done by defining the upper and lower bounds for the green in the image and turning all the pixels within that range black as seen in Figure 14.

Figure 14 - The original photo of the shoe without the green screen

A picture containing building, outdoor, way, arch

Description automatically generatedThe second function is the NewBackground() function which replaces the black pixels with the chosen background image. This is done by first calling the method RemoveGreen() defined earlier to remove the green screen and then layering the new image on top of all the black pixels in the image, resulting in the images seen below.

A pair of boots on a bench

Description automatically generated with medium confidence

Figure 15 - Green screen replaced by chairs outside

Figure 16 - Green screen replaced by Valletta street

A picture containing outdoor, nature, mountain

Description automatically generated

Figure 17 - Green screen replaced by the night sky

## Evaluation

Comparison of two images is performed using a function that calculates the error score based on a specified metric. The two metrics used are Sum of Squared Distance (SSD) and Mean Squared Error (MSE).

The blended image is compared with the image with two objects (S2) using both error metrics. A lower error value signifies that the blended image resembled the actual image S2 more.

|  |  |  |
| --- | --- | --- |
| **Filter** | **SSD** | **MSE** |
| None | 185244013 | 1329 |
| Gaussian | 185525091 | 1333 |
| Histogram Equalisation | 186486483 | 2616 |
| Median Blur | 185400136 | 1325 |

The image with the best SSD score used the object without applying any sort of filtering. The image with the best (i.e. lowest) MSE score used the object with median blur filtering. The image which did not use filtering had a very similar MSE score however the median blur image surpassed it slightly.

# Part 2 – Image Inpainting

The second part of the assignment consisted of using off-the-shelf inpainting functions to replicate the results of paper [1] as well as using the same function on new images from the COTS dataset.

## Implementation

In order to implement the second part of the assignment, the libraries *numpy* and *OpenCV* were used.

### Part A

The first part of the image inpainting section involved replicating the results of paper [1] using off-the-shelf functions. The images used were of statues, glasses, books, footwear, mugs and tech. In order to perform inpainting on the images, two algorithms were used, NS and Telea.

Table

Description automatically generated with medium confidenceAfter performing inpainting on all the images, the results were compared using the CompareResults() function described in Part 1, returning the following results.

Figure 18 - Table holding the error metrics for the original 6 images

### Part B

The second part of the inpainting section involved implementing the same inpainting algorithms but on a new set of images.

Text

Description automatically generated with medium confidenceAfter inpainting and calculating the same error calculations as before, the following results are returned.

Figure 19 - Table holding the error metrics for the new 6 images

## Evaluation

As with the tasks in Part 1, the SSD and MSE error metrics are used in evaluation.

6 different sets of images are used in task A. Using both the Telea and NS inpainting functions in the OpenCV library, an object is removed from the scene with two objects S2. The inpainted images are then compared with actual image with one object S1. The results are displayed in figure X.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Telea** | | **NS** | |
| **Set** | **SSD** | **MSE** | **SSD** | **MSE** |
| Statues | 115180351 | 455 | 115914723 | 511 |
| Shot Glasses | 33363637 | 59 | 33704007 | 59 |
| Academic Books | 59162597 | 327 | 61476765 | 383 |
| Footwear | 44000379 | 68 | 46650657 | 90 |
| Mugs | 38383500 | 75 | 40576761 | 84 |
| Technology | 38996685 | 118 | 40382590 | 145 |

The highlighted records are the lowest error score for each image set. A lower score indicates that that algorithm was better at inpainting an object out of an image.

Using the Telea technique provided the best scores for both metrics and hence resembled S1 greater than when using the NS technique.

Task B once again uses inpainting algorithms, this time over 6 sets of images with complex backgrounds obtained from the COTS Dataset. The results are displayed in figure Y.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Telea** | | **NS** | |
| **Set** | **SSD** | **MSE** | **SSD** | **MSE** |
| Food | 214050543 | 3573 | 214916338 | 3664 |
| Statues | 79180499 | 1738 | 79105387 | 1668 |
| Souvenirs | 73850406 | 2996 | 74686615 | 3077 |
| Academic Books | 93337891 | 3547 | 93754563 | 3476 |
| Cups | 125109291 | 902 | 125000973 | 899 |
| Electronics | 58630734 | 511 | 58714712 | 457 |

# References

[1] D. Seychell and C. J. Debono, “An Approach for Objective Quality Assessment of Image Inpainting Results,” in *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*, Jun. 2020, pp. 226–231.

[2] D. Seychell and C. J. Debono, “Monoscopic Inpainting Approach Using Depth Information,” in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, Apr. 2016, pp. 1–5.