Assignment 1 — Foreground Segmentation and Poisson Blending

Computer Graphics – Spring 2024

Submission 06.07.2023 at 23:55

1 Introduction

In this assignment, you will be implementing the GrabCut algorithm to perform foreground segmentation on an input image. You will then be required to replace the background of the segmented object using Poisson blending, which is a powerful image editing technique that seamlessly blends the foreground object with a new background while preserving the object's texture and color.

Please read the complete document carefully before you start coding!

2 GrabCut

An accessible description of the implementation [3] can be found in the following link. We strongly recommend you to read this short paper and implement the algorithm accordingly.

All the required information is located in the short paper. Still, if you would like to get additional information you can read the original grabcut algorithm [2] in the following link. You only need to implement the iterative procedure described in Sec.3.1 and Sec.3.2 to get a binary segmentation of the image.

In the downloaded zip folder, you can find a Python file named grabcut.py. Here are instructions for students on how to fill each unimplemented method. In all method replace the the TODO statement in the code with your implementation.

2.1 init_GMMs(img, mask, n_components=5):

In this method you should initialize and return two GMMs models one for the background and one for the foreground, each of them should consist of n_components. The initialization should use kmeans. You could use your own implementation of GMM or use any existing one, however you should pay attention to the differences between off-the-shelf models and the use of it in the grabcut algorithm.

2.2 update_GMMs(img, mask, bgdGMM, fgdGMM):

In this method, you need to update the Gaussian Mixture Models (GMMs) for the foreground and background pixels based on the current mask. This step involves calculating the mean, covariance matrix and weights of each GMM component for the foreground and background pixels in the input image. You can use the cv2.calcCovarMatrix() (or np.conv) function to calculate the covariance matrix and mean values, respectively.

After calculating the mean and covariance matrix, you need to update the GMMs using the formulae given in the GrabCut algorithm. You can refer to the following links for more information on this step: cv2.calcCovarMatrix.

2.3 calculate_mincut(img, mask, bgdGMM, fgdGMM):

In this method you should build a graph based on the existing mask and the energy terms defined in the grabcut algorithm. Then a mincut should be used. The method should return the vertices (i.e. pixels) in each segment and the energy term corresponding to the cut. You are allowed to use any graph optimization library, for example igraph.

Equation (1) in the short paper describes how to determine the N-Links weights. Use the following definition of β (equation 5 in the original paper):

$$\beta = \frac{1}{2 \cdot < (z_m - z_n)^2 >} \tag{1}$$

Where $< \cdot >$ denotes the expected value in the input image, i.e. the mean squared distance between each pixel in the image and all its 8 neighbors.

2.4 update_mask(img, mask, mincut_sets):

In this method, you need to update the current mask based on the mincut and return a new mask.

2.5 check_convergence(energy):

In this method, you need to check whether the energy value has converged to a stable minimum or not. You can use a threshold value to determine whether the energy has converged or not. If the energy has converged, you can return True, otherwise return False.

2.6 cal_metric(mask, gt_mask):

In this method you will evaluate your segmentation results. Given two binary images, the method will return a tuple of the accuracy (the number of pixels that are correctly labeled divided by the total number of pixels in the image) and the Jaccard similarity (the intersection over the union of your predicted foreground region with the ground truth).

2.7 Running the code

After implementing all the methods, you can run the code to see the GrabCut algorithm in action. The code loads an example image and defines a bounding box around the object of interest. You can play with the arguments of this code to load different images or your own image and bounding box area. The GrabCut algorithm is then applied to the image and bounding box to extract the object from the background. The final mask is then applied to the input image to show the result. You can adjust the number of iterations and the bounding box in the grabcut() function to see the effect on the result.

3 Poisson Blending [1]

Poisson Blending will allow you to blend source and target images. As we have seen, this involves the computation of the Laplacian matrix and finding the boundary of the binary mask (use the gradients for this). In the poisson_blend.py you should:

1. calculate the Laplacian operator
2. Solve the Poisson equation
3. Blend the source and target images using the Poisson result
4. Save the blended image

Your source and target images might no be of the same sizes, but you should only support a case where the source mask is smaller from the target image size. Pay attention to the matrices size, you should use sparse matrices in your code.

4 Submission instructions

The zip file you have downloaded includes several directories you will use throughout the assignment:

- imgs/ - images of objects
- seg_GT/ - ground truth binary images
- bg/ - background images
- bboxes/ - rectangles as txt file for each img in the imgs/ folder.
- grabcut.py, poisson_blend.py - python code to be filled with your implementation.

Img name	Accuracy	Jaccard
banana1.bmp	xx	yy

Table 1: Results table

You are required to submit a short summary of your work (in pdf format) along with your implementations of grabcut.py and poisson_blend.py (using python of versions 3.7-3.9). In your report you are expected to analyze failure cases and explain them. You are allowed to use off-the-shelfs: kmeans, GMM and graph optimization tools, but any other implementation unless stated otherwise in the question should be fully implemented (i.e. using cv2 blending will result in a grade of 0). In addition, our grader is fully aware of the open-source implementations.

GrabCut The short summary should include both Qualitative and Quantitative evaluations of your results, i.e. show the results on each of the images found in the imgs/ directory, compared to the ground truth. For the metrics please include a table similar to Table 1. Each row of the table will include the results for specific img, and the table should include two columns, one for each metric. In addition, you should analyze the effect of blur (low, high and without) on the results, the number of GMMs of grabcut, different initialization of the rectangles for grabcut. This analysis can be presented on 2-3 images. The code itself should not take longer than 1 minute per image to converge (it should be much less) - please indicate the average time per image in your report.

Poisson blending For each of the images in imgs/ folder select one of the background images found in the bg/ folder. Your report should include the result of applying Poisson blending between the foreground object detected by your grabcut algorithm and the corresponding background images. In addition, you should show for two images from the imgs/ folder and a single background image of your selection, what happens if the mask is not tight around the object.

References

[1] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 313–318.

[2] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "" GrabCut" interactive foreground extraction using iterated graph cuts". In: *ACM transactions on graphics (TOG)* 23.3 (2004), pp. 309–314.

[3] Justin F Talbot and Xiaoqian Xu. "Implementing grabcut". In: *Brigham Young University* 3 (2006).