6.815/6.865 Digital & Computational Photography

Problem Set 3: Matting and Morphing

Due Friday, April 4 7:00pm

Matting

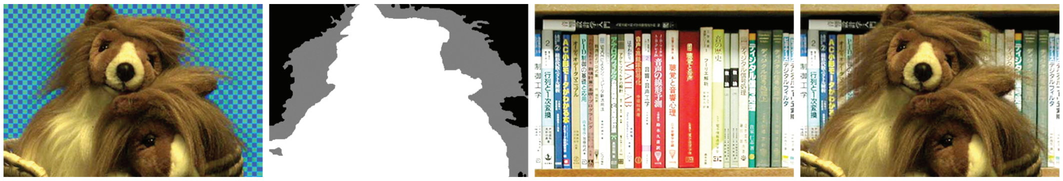
Matting refers to the process of separating the foreground and background elements of an image, generally for the purpose of compositing. A variety of methods have been proposed to solve this difficult and underconstrained task. In this assignment, you will be implementing a simplified version of Chuang et al.'s Bayesian technique from CVPR 2001: `http://grail.cs.washington.edu/projects/digital-matting/image-matting/`.

The user provides an input image and a corresponding *trimap* which coarsely classifies each pixel as foreground, background, and unknown. The known information from the foreground and background pixels is then used to estimate the color and transparency of the unknown pixels. This is done as follows:

- Model the color statistics of foreground and background using three-dimensional Gaussian distributions in RGB space. Note that Chuang et al. model both the foreground and background using multiple Gaussians each (a Gaussian Mixture Model) to create local estimates of the likelihoods. For this assignment, assume that the distributions are spatially invariant. In other words, you only need to compute two Gaussians—one for the foreground pixels, and another for the background pixels.
- For each unknown pixel, compute a *maximum a posteriori* estimate of the foreground color, background color, and alpha channel. This is achieved using an iterative optimization which is described in the paper and lecture notes.

Problem 1 (6.815/6882)

Implement the simplified matting technique described above and apply it to the provided images. Specifically, you should compute a matte for `toy.jpg` using the trimap `trimap.png` and composite your result with `bookshelf.jpg`. The provided image, along with the desired result, is shown below.



One thing you should notice about *toy.jpg* is that the foreground and background are both fairly uniform in color. This is why the simplified technique works well, even though it would be too simplified for more general cases.

Place your implementation in a script called `matting.m`. It doesn't have to be a neatly abstracted function or anything. If you'd like, you can even hardcode the image dimensions. The results shown on the previous page were achieved using an initial α of 0.5 for all unknown pixels, $\sigma_C = 0.01$, and 20 iterations. It's not a very fast algorithm—maybe a few minutes for the given parameters—so don't be alarmed if your code seems slow at first.

In your writeup: Paste images of your alpha matte and composite. That's all.

Extra credit: In case you have lots of spare time over spring break and want some extra points: implement the full technique in the Chuang paper and show your results for images with more variable color distributions. Compare the results to the simplified method.

Morphing

In this part of the assignment you will implement a triangulation based warping method, and then use it to perform morphing. You can use your technique to achieve a variety of very cool (but often very tacky) special effects.

Let's say we're given a small set of input (*warped!*) pixel coordinates $p_1, \ldots, p_n$ and their corresponding output (*unwarped!*) pixel coordinates $\tilde{p}_1, \ldots, \tilde{p}_n$[1]. Suppose we define a triangular mesh over the input coordinates $p_1, \ldots, p_n$. We can apply this same triangulation to the output pixel coordinates and produce a triangle-to-triangle correspondence between the two images. We can then use barycentric coordinates to define a pixel-to-pixel mapping between each corresponding triangle.

For any point p, located in a triangle with vertices p_0, p_1, p_2, the barycentric coordinates $\lambda_0, \lambda_1, \lambda_2$ are defined such that:

$$p = \lambda_0 p_0 + \lambda_1 p_1 + \lambda_2 p_2 \tag{1}$$

$$\text{and } \lambda_0 + \lambda_1 + \lambda_2 = 1. \tag{2}$$

If we replace the vertices p_0, p_1, p_2, with the vertices, $\tilde{p}_0, \tilde{p}_1, \tilde{p}_2$, from the corresponding triangle in the output mesh and use the same barycentric coordinates will we get the corresponding pixel $\tilde{p}$ in the output image:

$$\tilde{p} = \lambda_0 \tilde{p}_0 + \lambda_1 \tilde{p}_1 + \lambda_2 \tilde{p}_2.$$

---

[1] As discussed in lecture, you want to go from warped to unwarped coordinates to produce good results.

Given a point p, and vertices p_0, p_1, p_2, the barycentric coordinates can be solved for using equations (1) and (2). There are 3 unknowns, $\lambda_0, \lambda_1, \lambda_2$, and 3 equations (equation (1) holds for both the x and y coordinates).

Problem 2 (6.815/6.865)

Write a script `checkerwarp.m` that loads the provided `checkerboard.png` image and generates the warped image according to the following displacements:

$$\begin{aligned}(32,32) &\rightarrow (88,58)\\(224,32) &\rightarrow (229,82)\\(224,224) &\rightarrow (165,212)\\(32,224) &\rightarrow (61,178)\end{aligned}$$

You can use the MATLAB `delaunay` function in order to triangulate your mesh, and `tsearch` to search the triangulation. Remember to add feature points that keep the corners of the images unwarped. Use bilinear interpolation (e.g. MATLAB `interp2` function) to lookup the color at fractional pixel locations.

You'll be using the same code for the next two problems, so it might be beneficial for you to write a general warping function. This isn't required, though.

In your writeup: Paste your results for both warping from input coordinates to output coordinates, and warping from output to input coordinates. Describe, in a few sentences, any problems you've found with either method.

Problem 3 (6.815/6.865)

Load the images `face1.jpg` and `face2.jpg` and their corresponding feature points from the provided `facepts.mat` file (loading the MAT file creates two matrices `facepts1` and `facepts2` in your MATLAB workspace). Compute the result of warping the feature points for `face1` to their corresponding feature points for `face2`. Put your code in a script called `facewarp.m`. Again, remember to add correspondences that keep the corners of the image fixed.

In your writeup: Display the warped `face1` image that you generated.

Problem 4 (6.815/6.865)

Finally, with image warping completed, we're ready to perform morphing. Let's say you want to compute an image that's partially between the two faces (say, $0 \leq t \leq 1$ where 0 corresponds to `face1.jpg` and 1 corresponds to `face2.jpg`). Here's how you do it:

- Linearly interpolate the feature points from both images to produce a set of intermediate warping positions: `(1-t)*facepts1 + t*facepts2`.
- Warp both images to the intermediate positions.

- Linearly blend the two images from the previous step (again using $(1 - t)$ and t as blending weights.

The interpolation and blending should be weighted so that you can smoothly go from one image to another. Write this code in a script called `morph.m`.

In your writeup: Show a morphing sequence of maybe five or six images that goes from `face1.jpg` to `face2.jpg` using evenly spaced values of t between 0 and 1. Also, show a morphing sequence using your own example images.

Bells & whistles: Combine your matting and morphing. Performing morphing on images you have composited objects into.

Problem 5 (6.865 only)

This problem is for 6.865 only, and unlike the previous problems, it's going to be all reading and writing. Your job is to review a recent publication in computational photography. This will hopefully give you some more insight into the review process that these papers go through, and perhaps help you come up with ideas about your final project (and in case you were concerned, we don't expect your final project to be publication-quality, but it's still helpful to keep the review criteria in mind).

You should select one paper from the provided `papers.html` file. If there's another relevant paper that you're particularly interested in reading, ask us about it and we'll probably be okay with it (unless it's something that you're already supposed to read for one of the problem sets).

You should use the SIGGRAPH review form, available at the following URL: `http://www.siggraph.org/s2005/main.php?f=cfp&p=papers&s=reviewform`. You may omit the numerical ratings from questions 5 and 6. For question 8, try to come up with at least one outstanding question that you think the authors didn't address in the paper. You can skip question 9. You should put your review in your PDF file along with everything else. Try to be thorough! Probably a paragraph or two for each question, and significantly more for question 7.

One final note: don't be compelled to like a paper because it was already published. Every paper has some shortcomings. Maybe the method is very elegant and general but doesn't end up producing very impressive results. Maybe the results are spectacular, but the method itself has some mathematical holes. Maybe everything seems great, but there just aren't enough details to reproduce the results. It's up to you as a reviewer to weigh the strengths and weaknesses of each paper.

Submission

Like the previous assignment, you should assemble a ZIP file that is named after your Athena login. Make sure this file contains:

- A PDF file with your results and (if you're in 6.865) your review.
- Your MATLAB code:

  - `matting.m`
  - `checkerwarp.m`
  - `facewarp.m`
  - `morph.m`
- Any images (other than the provided ones) that might be necessary to run your code.

All submissions are due on the Stellar website by April 4 at 7pm.