JULIUS ELINSON NOVEMBER 14, 2011

CS 155 FINAL PROJECT PROPOSAL

In the image processing assignment, we implemented a means of compositing images that required a mask to indicate interpolation values for each pixel location. In a recent paper by Farbman et al., they propose a novel approach for image cloning that removes the need for a mask [1]. This is not by any means the first attempt at simple image cloning, but rather offers a faster and perhaps simpler approach than a previous common approach that requires solving a Poisson equation. This alternative approach uses mean-value coordinates to determine interpolant values. They then offer a number of optimizations and further applications of their algorithm. I propose to implement the core framework they describe and a selection of their optimizations and extensions.

In particular, I plan to implement aspects of this paper in three parts: a user interface, interpolation algorithm, and then add optimizations and extensions. These three aspects will correspond to the primary weekly milestones and will be roughly equally weighted, as indicated in my rubric and schedule in Figure (1).

The user interface will first need to support two windows open at time: one for the image source and one for the image destination, to foster a simple user experience. I will leverage the existing image class from the image processing assignment as the main data structure and perhaps some of the same control functionality to interface between OpenGL and the image class framework. The interface must be then able to support the user selecting the vertices of the bounding polygon in the source image. While this is a simple front-end addition, it will then require determining which pixels in the source image are enclosed in the given polygon. The user will then be able to specify the target location via a single coordinate, which will correspond to first (and last) vertex identified in the source patch. Similarly, once this location is given, the pixels that will be in the cloned area must then be determined in a way that pairs them with the corresponding pixel in the source image.

Once the source and destination pixels have all been identified, the patch will then be interpolated according to the algorithm Farbman et al. describe. The first step of this is to determine the mean-value coordinates for each pixel in the source patch. The paper includes an explicit formula for this computation. Then intensity differences between the boundaries in the target and source patch are calculated and stored. Finally, for each pixel in the target patch, the mean-value interpolant is computed, which uses the previously computed mean-value coordinates, weighted by the boundary differences at each boundary vertex. This formula is also given explicitly in the paper. Once this value is calculated, it is added to the intensity of the target image pixel to yield the pixel in the final image.

The final component will include one optimization to the algorithm run-time and further application of image cloning. While the paper describes two powerful improvements in performance, I've chosen to attempt to implement only a single one given the level of difficulty of each. The paper describes a process of hierarchical boundary sampling. This essentially eliminates the need to use all of the boundary points when computing mean-value coordinates. It does this by creating layers of boundaries by effectively downsampling with each successive layer. Then for a given point, starting with the coarsest level of boundary points, if it is sufficiently far from a boundary point and the difference in angle between it and two pairs of consecutive boundary points, then no finer level of boundary points is needed. Otherwise, it proceeds to use a more detailed boundary level. This optimization essentially eliminates the need to use all the boundary points when distant ones contribute similarly and with little weight.

Of the numerous extensions Farbman et al., I intend to attempt to perform image cloning on video footage. While they describe a GUI that performs this in real-time, I will modestly attempt to take in a sequence of images from a decomposed video file, perform image cloning on each and then reassemble the stills into a final movie. The paper describes a method of retaining previous mean-value interpolants and then using those with time-decaying weights and the current value to form a smooth sequence. The location of the cloned patch will be static throughout the image sequence.

Should any of these features be too difficult to implement, I can alternatively try implementing selective boundary suppression. This essentially removes smudging by allowing the user to specify boundary vertices not to use when computing the interpolants. Changing the computation shouldn't be too difficult, but allowing the user to select the vertex to ignore might be challenging. Furthermore, another alternative will be to compare the run-time of my program with the results they report in the paper and give a brief analysis of any difference between the two.

One anticipated risk is the need for a large number of boundary points of the image patch. Rather than require the user to painstakingly enter many discrete points, I would like to be able to both sample linearly between adjacent vertices and allow for smooth tracing through the GLUT motion function. I anticipate the former to be easier to implement and will try that approach first. To allow tracing, I will model the function off the motion function used in Assignment 3. Another challenge I expect to encounter is efficiently handling the input/out when cloning over a number of images in an automated fashion. While it should be no more difficult than handling a single image, I have little experience working with a sequence of inputs and outputs. Resolving this should only be a matter of diligence and being careful when designing the framework to allow it to load and save multiple pairs of consecutively. Lastly, I anticipate some difficulty in simply setting up a functional interface with two windows that integrates easily with the image class. To resolve this, I have already begun reading over some of the provided code from the image processing assignment and researching techniques for managing multiple windows in GLUT. It should be an easy blockade to hurdle, but all further progress is contingent on a completely functional framework for handling opening, viewing and saving two images at once.

References

[1] Zeev Farbman, Gil Hoffer, Yaron Lipman, Daniel Cohen-Or, and Dani Lischinski. Coordinates for instant image cloning. *ACM Trans. Graph.*, 28:67:1–67:9, July 2009.

Component	**Points**	**Due Date**
PART I - User Interface		
OpenGL interface with two windows that independently load, display and save .bmp images	**5**	11/22
Ability to discretely specify polygon outline of patch to clone via the mouse	**5**	11/22
Determine which points are in the patch given boundary vertices	**10**	11/22
Linearly sample additional boundary points between user-specified vertices	**10**	11/29
Allow user to continuously trace polygon boundary	5	11/29
PART II - Interpolation		
Compute mean-value coordinates in each pixel in the source patch	**10**	12/1
Compute difference in boundary intensities	**5**	12/1
Compute mean-value interpolant for each pixel and the resulting image	**10**	12/1
PART III - Optimizations & Extensions		
Create the hierarchy of boundary points	**5**	12/8
Primitive cloning over a sequence of images, without using weighted average of time-decaying interpolant values	**5**	12/8
Incorporate the hierarchical boundary into mean-value interpolant calculation	7	12/12
Smooth cloning over a sequence of images using weighted average of previous interpolant values	7	12/12
Selective boundary suppression	5	12/12
Artwork of video and/or image cloning	5	12/12
Test program run-time performance and compare with CPU implementation results reported in paper	5	12/12

Figure 1: Suggested project rubric and schedule. Note that bolded points are required and total to 65. The rest of the points may be chosen among the remaining features.