[CS 766] Project Proposal

Luke Swanson

Bob Effinger

**Introduction & Motivation:**

In some image editing situations, it is useful to combine elements from multiple pictures into a single composite image. These situations may range from stitching many whole images into a single panorama, to selectively placing small pieces of the images into a composite. Our project will focus on the latter situation. Selective combination of photographs presents several challenges. Firstly, one must find some way to select only the desired elements from a source image(s) and remove all unwanted portions. Second, the picture elements must be combined. However, simply placing the selected elements together over a common background may not give desirable results. Separate images, taken under differing conditions, may exhibit drastically different lighting and color characteristics. We will attempt to solve this problem of selective image compositing through the creation of a simple desktop application. This tool will allow users to select image elements, combine the elements with a background image, and automatically match the color characteristics of the individual pieces.

Creating composite images from several elements is a common task in certain styles of photography, as well as modern filmmaking. For example, when shooting scenes for a movie, it is often impractical to have all the desired elements together in one location. The script may call for an event to take place in a location that does not physically exist, or that would be dangerous or otherwise impractical to film in. In certain film genres, such as science fiction, there are often requirements for characters, vehicles, or environments that must be entirely computer-generated. As such, the post-production team on a film project will often be given a collection of disparate images, or plates, to be combined into a final frame. To achieve an acceptable result, they must isolate the desired elements from each plate, and combine them into a cohesive whole, which gives the appearance that all elements were shot together in one location. High-quality algorithms for isolating individual elements in one image, and combining the pieces seamlessly, are required to create convincing composites. Further, algorithms alone are of little use to the compositing artists responsible for constructing these images. They require practical tools that fit into an existing workflow, which implement the procedures in a usable manner.

**Existing Approaches:**

The current state-of the-art for interactive matting is an algorithm called Soft Scissors3 which unlike Intelligent Scissors4 and GrabCut2, keeps many details of the edge intact. More recently Deep Learning has been applied to Image Matting5 with hopes to refine the edge of an image even further. To combine Deep Learning and Image Matting however, this approach removes the interactive aspect of the design, and does not allow a user to define a specific area of interest in an image. Another recent paper used some of the ideas of image matting but rather than processing a single image, a sequence of images was used with the goal to extract context regarding the background8. Unfortunately, many of these more recent topics have strayed away from the ability of nonexperts to use them.

**Proposed Approach:**

We will implement a pair of existing techniques for interactively selecting elements (called foreground elements) from an image, specifically the GrabCut2 technique from Rother et al., and Lazy Snapping1 as described by Li et al. These were suggested in the 5th project on the course ideas page. Our program will then place the foreground elements onto another image (called the background element), and automatically adjust the pixel data of foreground elements to match the color characteristics of the background. The specific method for adjustment will depend upon the development time available after implementing the selection algorithms. We may attempt to use an existing approach (several existing approaches are mentioned in section 9.3.4 of the course textbook6), or attempt to develop our own simple heuristic and adjustment procedure. In the future, it may be possible to extend the implementation to handle image sequences, in the form of video files.

It does not appear that much work has been done to integrate interactive image editing directly with automatic color matching of composited elements. Kwatra et al.7 presented some results showing automated matching of elements from separate images, though their work typically blended broad regions of each image, rather than tightly-segmented objects. Available end-user tools for image compositing typically leave color matching to the artist, rather than performing automated matching. Thus, this project could fill a role in image compositing that has not been adequately addressed in any direct way by previous works. Integrating these two parts of an image-editing workflow tightly into a single tool may allow for some additional improvements that would be non-obvious when considering the parts individually. For example, when separating foreground elements from one plate, the unused portions of the plate would typically be discarded. However, when we anticipate color matching as a next step in the compositing process, we note that the discarded image data may contain valuable clues about the lighting and color characteristics of the foreground image. Thus, our color matching implementation may refer to the entire source image of a foreground element when determining the best color adjustment to make. On the other hand, when considering color adjustments alone, we would typically expect to match foreground elements as well as possible to an entire background image. This could lead to complications when trying to automatically match a red object to a predominantly green background, for example. For a tool that already implements interactive segmentation, however, it may be possible to allow users to select objects within the background image as targets for color matching. This could potentially improve the resulting color match.

**Evaluation Criteria:**

Performance evaluation of our solution will be based on the Ease of Use of the solution, the amount of interaction that is required from the user, image quality, and how much time is required for task completion. The implementation must be designed to allow for nonexpert users to be able to perform the task with minimal instruction. In order to meet this design constraint, many of the performance metrics revolve around user experience when interacting with the tool. Originally these metrics were evaluated through a mixture of novice user feedback, and by evaluating how long it took novice users to produce acceptable results. In addition to user feedback, the resulting images will be compared to images produced by existing Image Matting techniques to evaluate the results of our solution.

**Timeline:**

|  |  |
| --- | --- |
| Complete By: | Task: |
| 3/7 | Implement GrabCut segmentation |
| 3/14 | Create GUI to allow interactive GrabCut |
| 3/21 | Implement drawing of free-hand lines on GUI (preparation for Lazy Snapping) |
| 4/11 | Implement Lazy Snapping |
| 4/13 | Finalize choice of color matching/gradient domain blending algorithm |
| 4/23 | Construct project website; Prepare presentation |
| 4/30 | Implement automated color matching |
| 5/7 | Debug, clean, and stabilize code; Finalize project website |

**References:**

1https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/lazysnapping\_siggraph04.pdf

2https://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

3http://vis.berkeley.edu/papers/softscissors/softscissors-SIG07.pdf

4https://courses.cs.washington.edu/courses/csep576/15sp/readings/mort-sigg95.pdf

5https://arxiv.org/pdf/1703.03872.pdf

6http://szeliski.org/Book/drafts/SzeliskiBook\_20100903\_draft.pdf

7https://dl.acm.org/citation.cfm?id=882264

8http://grail.cs.washington.edu/wp-content/uploads/2015/08/Chuang-2002-VMC.pdf