Andrew Mauboussin, Aran Khanna, Jackson Okuhn, Serguei Balanovich

May 5, 2013

Seam Carving - CS51 Final Project

**Final Report**

**The Plan**

The original idea that we submitted as our project proposal was entirely different from the project that we ultimately created. Initially, we had planned to create a system which would incorporate patterns of high-pitched sounds in order to encode and transmit information from a computer to a mobile device. However, after consulting with our Teaching Fellow, we realized that a project involving image manipulation would be far more algorithmically complex and would better satisfy the scope of the project. Thus, the draft specification that we submitted in the first week [attached] had an entirely different set of goals, as outlined below.

Our planning via the draft spec was very detailed and helped us lay the groundwork for the ensuing weeks during which we developed our code. By outlining precisely what we were hoping to accomplish and laying down a timeline which we adhered to very well, we ensured that our project was never a last-minute scramble. By establishing a clear vision and aiming for the highest goals we ensured that our project was completed in a timely and successful manner

The process was not entirely smooth, however. Though we completed all of the requirements for each milestone, we did miss a few of the midnight deadlines by several hours in order to ensure that all functionality was up to the expectation of each checkpoint. After the technical specification, we knew precisely what we were going to do, and had modularized the work so that on the final night before the first checkpoint deadline we could meet and put all of our work together. All of this worked very well for us, except for the delays caused by the bugs inherent in all of our code merges. Often the seam derivation code ran into unexpected errors because the energy calculation outputs were not quite as expected. At other times, our shrinking and enlargement algorithms expected seam derivation to output a forward ordered list, when really a reverse ordered list was given. Though these were errors we ran into at each

milestone in the development of our project, they ultimately made our project stronger and more coherent. More details about these and other challenges will be outlined below.

Overall, however, the development process for our code went very smoothly despite some group members' lack of experience with the Python programming language. By the time the project was over, however, all of us had full command of it and made changes and updates to our functions with ease. We succeeded greatly in planning out our tests and effectively using Git for branching and working on tiny parts of the code so as not to disrupt the mainframe. More details on our effective learning and use of Git will be described below in the "Modularization" section. Though our group initially seemed somewhat flawed, as we could rarely find time to meet and work together, we quickly figured out an efficient system of interfacing and testing our code separately and convening on small occasions when we had to conduct thorough system-wide tests.

Thus, as a result of our foresight and diligent planning at the outset of the project, all initial problems with organization were resolved in a timely fashion that both ensured that our project was a success and that we all learned valuable lessons about remotely working with a team of collaborators. In addition, our testing platform was extensive and easy to use by all members. A few well-documented changes to energy-test.py could print out numerous different energy maps which we could holistically compare to determine the most useful and powerful ones. Altering lines in main.py allowed us to easily compare all sorts of images - shrunken, enlarged, or object-removed against their originals. This allowed us to compare a variety of combinations of energy plots and seam derivation algorithms in order to evaluate the best results. Finally, if we were ever unsure of our code's functionality, we could run a seam revealing test to highlight those seams that our algorithms derived. Though it was impossible to perform any rigorous tests for our code, the flexibility of the testing module we developed from the very outset of the project proved very effective for us in determining the success or failure of specific implementations throughout the entire development process. Ultimately, this combination of planning, organization, and extensive testing ensured our project's success.

**The Challenges**

The most pleasant of our surprise consistently came in how easily our problem could be broken up and divided amongst us, and how concisely and it really led to a lot of good modularization so that we could effectively work in separate files within a common framework with relatively little major friction with regards to pulling, pushing and merging files. The common framework we created at the beginning of the project had to be tweaked very little as we filled in the parts we had laid out in the initial skeleton with code. The pathfinding, energy finding, file i/o, and integration of components in a general framework were all developed with a large degree of autonomy from each other once the framework was in place, which is something that none of us have experienced to this extent before when working on larger projects like this.

Unfortunately, though the bulk of the project was modularized fairly well, and allowed us to abstract away from the work of our teammates, we had a few unpleasant surprises related to the relative complexity of integrating the individual units that comprised the entire end-to-end process of finding and manipulating seams in an image. The unpleasant problems came with integration tests, which had the entire team hunting through code to debug methods that were not meshing well and producing poor images. This seems like an unavoidable problem though, as we had modularized as best as we could and upon reflection it was truly the edge cases, like inserting large negative energies for object removal, that caused these problems to arise.

While our wrestling with integration with respect to edge cases was annoying, one of the most time consuming surprises was the sudden difficulty we found in implementing the entropy energy algorithm. From the papers we read it seemed complex but doable, but as we got into it, we found many edge cases and intricacies not addressed by the paper, especially when it came to mirroring with 5x5 squares to calculate the energy. This led to a very long process of planning and tweaking to restructure and re-implement this algorithm in a different manner from how the paper initially presented it. While we finally got it, it took up much more time than expected which was quite unpleasant as it led to some late nights.

Another surprise we had was with respect to the efficiency of Dijkstra's algorithm vs the dynamic programming counterpart. We were aware that dynamic programming approach ran in O(V) while Dijkstra ran in O(E*insert + V*deletemin) where V is the number of pixels and E it 3*V and where (with the heap) insert and deletemin are O(log E). The difference is speeds of the two though were massive, non-reflecting a simple logarithmic factor, and after implementing Dijkstra's we were certain that larger images would be impossible to process, until we realized how quickly the new dynamic approach ran in comparison.

We were also surprised by how concisely python allowed us to write the code, and how many support libraries there were to perform functions that allowed us to unit test our code effectively even when code that it was reliant upon was unfinished, a great example of this is when Aran was developing Dijkstra's algorithm and the heap wasn't fully functional, he was able to test his code by making use of python’s heap library. Furthermore the simplicity of python code, and rapid speed of development that it allowed, while allowing modularization complexity with a fully flushed out object system was critical to getting us working together efficiently and putting us on an even playing field language wise. Furthermore the interpreted nature of the project allowed very efficient testing and debugging for our groups unit and integration tests.

The downside to working with Python is the relatively slow nature of the language. The entire process of converting an image is quite arduous as even the most efficient algorithms run much more slowly than their compiled counterparts would. This led to a lot of downtime when it came to waiting for tests of large image sets to complete, but this really was a minor annoyance, and overall our choice of python proved very useful in the end.

Another major choice we made was to use python imaging library. This library added a lot of good frameworks and abstractions about the image to work with. Unfortunately this added a lot of unnecessary junk during the conversion process, which added to the interpretation time. Because we were focused on the relative speed of implemented algorithms as opposed to the overall speed of the program we believed this to be a good trade off in terms of saving time and adding a useful abstraction, as it added a fixed time cost to the program but nothing more.

Finally the choice of implementing only two core pathfinding algorithms as opposed to adding more randomized, non-deterministic ones such as the genetic algorithm, was key as we wanted our algorithm to be as correct as possible on the data sets we gave it. Even though complete correctness did give us a slight performance drop asymptotically, we came to realize that because the fixed cost operations were really dominating the time spent processing a picture it made sense to simply run the deterministic algorithms as for any reasonable sized input they would be in the ballpark of the time the probabilistic algorithms would take.

**Modularization and Teamwork**

Each member of the group contributed much to the final product both by completing individual goals and by integrating well with the entire team to put it all together. At first we had trouble coordinating and spent quite a bit of time interfacing the separate components of the project, before modifying our skeleton to its final version. This, however, proved inefficient, and with urging from our Teaching Fellow we designed a plan of modularization that provided the precise balance of individual and group work that was necessary for a detailed and cohesive project. Though we all got together on the last day and reviewed the code for all bugs and errors, each of us had specific, central contributions outlined below.

***Serguei Balanovich*** was responsible for the energy calculation infrastructure of the project. The majority of the work done on locating pixels, finding their neighbors, and performing the necessary calculations was coordinated and finished by him. He did a large portion of the research and coding involved with the energy.py file and he coordinated the integration of these methods into the rest of the code base. He wrote the methods for both the Sobel and Entropy filters and was later involved with the abstraction of the former into a multitude of filters for tests of effectiveness. He was also in charge of displaying all energy maps through the to_energy_pic method in the image.py file, using these displays for evaluation of the effectiveness of these algorithms.

***Aran Khanna*** was in charge of seam derivation infrastructure. He led the effort in calculating the lowest energy seams using both Dijkstra's Algorithm and the system of Dynamic

Programming. He also worked on transposing each image so that seams could be derived both vertically and horizontally in order to provide the project with maximum versatility. He ensured that his algorithms provided output in such a way as to ensure the functionality of the other code and integrated his functions with the main Image class and the energy modules. The majority of the research and evaluation of algorithm effectiveness was likewise delegated to him. This meant that he was primarily involved with ensuring that the output of the energy calculation was in accordance with the input values of his algorithms, and he worked with Serguei in order to ensure cohesion between these two separate aspects of the project.

***Andrew Mauboussin*** integrated the modules created by the rest of the team to actually manipulate images. He designed the sc_Image and Pixel classes which comprised our representation of images and wrote the functions that allow an image file to be converted to a sc_Image and a sc_Image to be written to a JPG using the Python Imaging Library (PIL). He also created the method that shrink or enlarge images using the energy module developed by Serguei and the seam derivation module developed by Aran. He later improved the efficiency of the shrinking and enlarging by removing redundant energy calculations. During the past week, he implemented object removal,  Finally, he wrote functions to display seams in a clear, red color, allowing us to evaluate the correctness of these seams.

***Jackson Okuhn*** developed functions that improved the project's functionality in many respects. He wrote the priority queue heap function that was used in developing the Dijkstra's implementation of seam derivation. He abstracted all the energy filter methods and added a number of different filters, including the Kroon and Scharr operators on 3x3 and 5x5 squares of pixels. He implemented the mirroring algorithms necessary to ensure that pixels on the edges of the image always had neighboring pixels with which to work and, as a corollary, rewrote the code that managed how energy functions were called in the image class. With Andrew, he also altered the support code to allow for enlargement and other object manipulation.

All group members worked on their individual portions together in separate Git branches, and met as frequently as possible in order to collectively debug and test the overall functionality of the code. Though we tried to work in separate files as much as possible, we were often forced to work in the common image class so that we could add whatever

functionality we have implemented to the object via a method. The vast majority of merge conflict and integration test failures we had along the way did not arise as a result of working in the same file though, but rather from different parts not working well with inputs and expected outputs during execution. Because of this we were really able to split this project up into the equivalent of independent problem sets for each of the group members.

**Lessons Learned**

From this project, everyone on the team learned much about images and the power of image manipulation. Though we had thought about the concepts of image energy and seam carving in general, it was not until we began putting our ideas into code and seeing the actual results that we began to appreciate the implications of working extensively with images in Python. As a foundation, we used the Python PIL Library in order to learn the precise ways in which previous developers had worked with images in this language. From this understanding, we conceptualized the changes we would need to make to the standard library in order to most effectively use it for our own benefit. We then built upon this understanding of image representation through code and performed extensive research on energy calculation. Finding the energy maps for the images was crucial to the success of our project, and we learned much about the different filters, operators, and forms of image entropy that we had never heard of before. Putting these into code and then evaluating the differences between the filters really gave us a visual representation of the theories and solidified our comprehension of how each filter worked outside of the mathematical equations.

However, despite the fact that we learned much about Python, image manipulation, and powerful algorithms such as Dijkstra's and Dynamic Programming throughout the duration of this final project, our biggest growth collectively was in our abilities to coordinate, allocate, and organize time and resources most effectively. Through the mistakes and successes we experienced in the development of our code we learned much about the effective communication and use of Git in order to seamlessly collaborate on the same code. From our combined contributions and relative lack of conflicts, we ensured that each aspect of the final

product was complete and accounted for, with minimal damage to the fundamental framework developed at the very outset.

If we were to begin the project anew, there are a few small changes we would make to our development of the project. The biggest of these would be our method of interfacing. We ran into several troubles when we tried to combine our code and especially in the beginning stages, we would often meet to merge code and would spend far more time than anticipated trying to debug the ensuing errors. Had better planning been conducted, we would have developed our code in adherence to an overarching schematic, so that the bugs would come from errors in code rather than errors in miscommunication. Other than that, our project went quite smoothly and apart from more efficient coding practices and more effective communication, there is not much that we would change if given the chance to do this project again.

Given more time, there are several more things we would complete prior to final submission. For instance, the current procedure for enlarging an image in two dimensions requires first the desired increase in width and then in height to be applied, which may not be the best solution. A more effective but far more complex strategy would have been to enlarge the image seam by seam, first horizontally, then vertically, and with this alternation, achieve beautiful results. In addition to this, we would probably spend slightly more time on image accentuation, since that code was not developed to its utmost extent. We did accomplish our fundamental goals, implemented many of our desired side functions and experimented with far more energy maps than we had anticipated. Thus, although we did not have enough time to fully accomplish all of our goals, we went beyond our fundamental plans and are quite satisfied with the level of complexity and functionality of our final submission.

**[For Running Instructions, see README file in project]**