Improved Seam Carving for Video Retargeting

By Erik Jorgensen,
Margaret Murphy,
and Aziza Saulebay

CS 534 Fall 2015
Professor Dyer
December 21, 2015

Table of Contents

*Our project website is located at http://msmurphy2.wix.com/cs534seamcarving

1. Abstract

Seam carving is used as a content aware option for resizing two-dimensional images by removing one-dimensional seams of pixels. To convert this approach so that it can be applied to video, we will implement content aware resizing of video by resizing consecutive two-dimensional image frames that are temporally related. Our dynamic programming approach to improved seam carving for video sequences focuses on improving visual appearance of resized videos as well as reducing overall computation time. We discuss multiple implementations of video resizing used methods such as random reference frames, varied temporal bound widths, and coarse scale computation acceleration.

2. Introduction

The idea behind the basic seam carving algorithm involves locating, followed by inserting or removing, the “optimal seams” of a given image. A seam is a horizontal or vertical path of connected (either adjacent or diagonal) pixels. The image’s optimal seam is found by using an energy function, which “measures how perceptually noticeable each pixel is”, with the intent of removing or replicating the least noticeable pixels. Thus, it is desirable to update rows or columns in the part of the image from which the lowest amount of “energy” appears. The ideal output image will leave little trace of seam removal or addition, as well as minimize artifacts that would occur when cropping or scaling the original image.

3. Basic Seam Carving Algorithm

Seam carving applied to two dimensional images, either removing or inserting seams, typically results in a smaller or larger output image while retaining photorealism. The seam removal algorithm is a five step process that utilizes dynamic programming. First, calculate the energy for all color channels by adding together the squared gradient values in both the x and y directions; combine the summation of all these energies into one image, resulting in an energy map depicting a two-dimensional gradient. The second and third steps are to find and remove the minimum seam from top to bottom. The removal causes a column shift for all the pixels to the right of this seam as well as a reduction of width by exactly one pixel. The fourth step is to repeat the first through third steps until the desired number of minimum seams are removed. Finally, rotate the image 90 degrees then repeat steps one through four in order to remove the minimum horizontal seams.

In order to find the minimum seam from top to bottom, it is necessary to create an accumulated cost matrix. Iterating through the rows of the energy map, copy the top row from the energy map into the top row of the accumulated cost matrix. Every pixel’s value after the top row is equal to the sum of its corresponding pixel value in the energy

map and the minimum of its top neighbors in the accumulated cost matrix. If a pixel is not on the left or right edge, its top neighbors are the pixels located in the top-left, top-center, and top-right positions. However, if it is on an edge, either its top-left or top-right pixel will be unavailable and should be ignored while finding the minimum value.

If using the Forward Energy method, it is vital to take into consideration that each top neighbor's accumulated cost value will be increased by the amount of forward energy from the corresponding new neighbors after the seam is removed. This new cost value will increase the accuracy of the pixel's top neighbors' minimum energy value. Working from the bottom up to the top row, each pixel of the minimum seam is equal to the minimum value of its corresponding row in the accumulated cost matrix. Record the coordinates of these minimum seams.

In order to insert seams, calculate the same number of minimum energy seams you want to insert first, without affecting the original image. The coordinates recorded during this seam removal determine which seams to insert into the original image (in the same order that seam removal was performed). These inserted seams receive pixel values that are averages of its nearest (either top and bottom or left and right) neighbors. Although forward energy is beneficial for seam removal, it should not be used when inserting a seam; the forward energy from the new neighbors will be inapplicable to the output image.

Objects can be removed from images by creating a binary mask. Start by deciding weather it is more efficient to remove either the vertical or the horizontal seams by measuring the masked region. Highly negative weights given to the areas under the masked region (during energy map calculation) guarantee that the minimum seam will contain this region. Minimum seam removal continues as usual, except for when removing the minimum seam, which also must be removed from the mask. After the masked region has been removed, use seam insertion so that the new image still has the same dimensions as the image before the object was removed.

Object Removal

In the top right picture from the group at left, the center green shoe was removed.

In the bottom right picture of the group, the yellow shoe was removed.

4. Motivation

After having had experience working with content aware image resizing using seam carving during homework three, experimenting with removing the number of columns and rows in an image, we were given the opportunity to implement an expand function in order to expand the number of columns and rows, as well as implementing and comparing additional energy functions. The associated article for this assignment “Seam Carving for Content-Aware Image Resizing”, by Avidan and Shamir, suggested resizing video. Avidan and Shamir wrote another article titled “Improved Seam Carving for Video Retargeting”. This inspired us to pursue our own rendition of their experiment.

5. Problem Statement

The basic seam carving algorithm is normally used on two dimensional images. The goal of this project is to implement this algorithm in a way so that it can be applied to video while incorporating an improved technique. In order to extend the dynamic programming technique for seam carving through time, we will need to treat the video as a sequence of two dimensional frames that relate temporally. The improved technique uses the forward energy method through a graph data structure to obtain improved visual quality.

6. Related Work

There have been numerous projects that involve image resizing and a few for video retargeting. Wang et al. worked on detecting Regions of Interest to adjust video size so that it can be viewed on a smaller screen. Fan et al. created a zoom in effect by cropping out unimportant details and focusing on the important ones at a larger scale. Liu and Gleicher used cropping, scaling, and virtual camera motion, minimizing information loss through saliency and object detection. Setlur and Tao both led projects dealing with segmenting images into foreground and background layers, scaling independently, and then recombining these into the retargeted image.

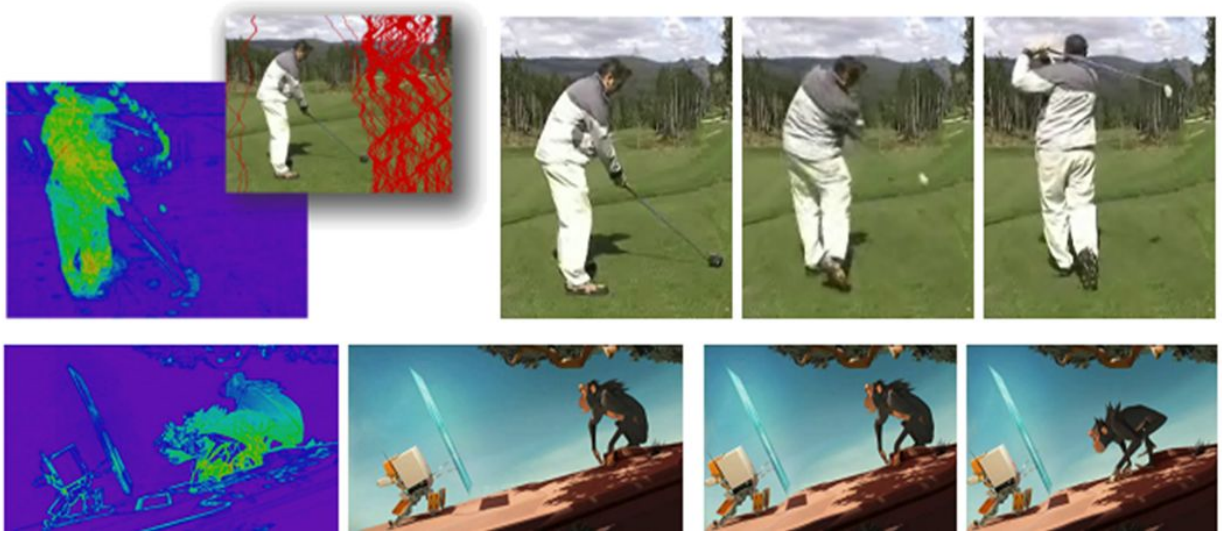
Pritch et al. segmented their input video into objects and activities, then produced a shorter output video based on user query. Wolf et al. retargeted video using non-uniform global warping. Avidan and Shamir retargeted images by finding the optimal seam using dynamic programming. The “Evolving Time Fronts” approach to video retargeting involves mapping 2D surfaces into a new video sequence, rather than removing them from the video, which allows for user manipulation.

Graph based techniques have also been quite popular in areas including image restoration and segmentation as well as object recognition. An image is represented by a graph which is separated by similarity of pixels or voxels in the image, where similarity is measured by change in intensity or gradient values. Kwatra et al. minimized energy in 3D space time video by using graph cuts to join two textures without producing a visible

seam. The challenge is producing a graph that only contains monotonic and connected cuts.

7. Theory

7.1 Applying Algorithm to Video



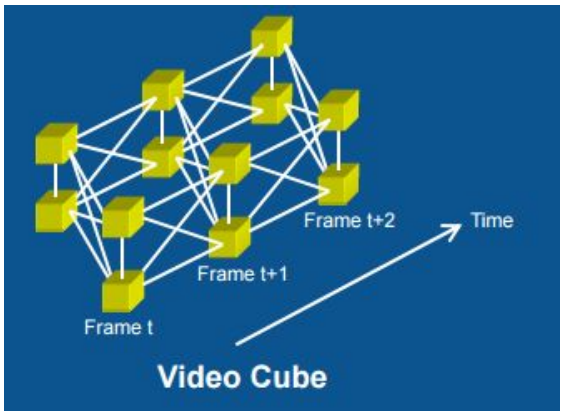
If we apply the basic seam carving algorithm to video as is, to each individual frame of the video separately, it will produce artifacts. Ideal, frame-independent seams may be on opposite sides of the image from one frame to the next. Taking multiple seams out with no frame-to-frame dependence could result in a jittery output video because a big portion of the seams could be taken out of the left in one frame, and the right in the next. The result would show instantaneous shifts of objects in the middle of the video. Another option is to remove areas with the least overall importance across all frames. For each individual image, compute the energy function, then take the maximum energy value at each pixel location. These seams are called “static seams” since they stay the same in all frames. This static method is beneficial since it is simple, fast, and gives good results when a stationary camera was used to produce the video and the foreground and background are separated.

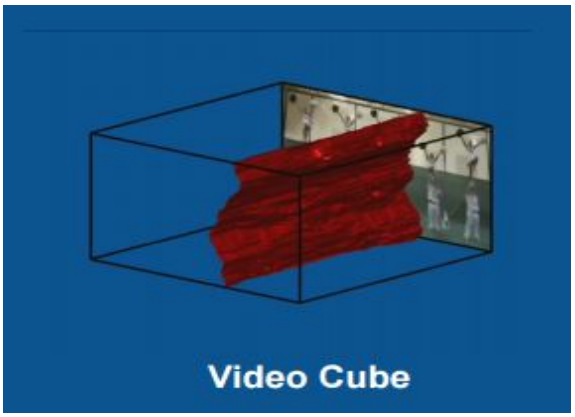
In order to allow for seams to change over time, such as when the camera moves or multiple actions are taking place, remove two dimensional seams in space-time from the three dimensional video. This will remove one seam from each frame. This method allows for adaptive change over time while preserving temporal coherence since the surface is flexible and connected.

7.2 Graph Cuts

Avidan and Shamir used graph cuts to apply the algorithm to video. The main idea behind graph cuts is removing two dimensional seams from three dimensional

video, with time as the third dimension. It involves creating a graph where each node represents a pixel. Connect the pixels that are in the farthest left and farthest right columns to either the virtually created source (beginning node) or sink (end node). Both the source and sink have infinite weights. A single cut in this graph will divide these pixels into two distinct groups.





A cut's cost is defined as the sum of weights for arcs running from top to bottom. The cost of this cut is directed which means that it only sums up the cost of boundary arcs that run in the same direction. Choose pixels to the left of the cut arcs in order to define seams from this cut. The optimal seam is the cut with the lowest cost chosen from all the valid cuts. There are two important things to keep in mind when considering cuts that will create valid seams: monotonicity and connectivity. Monotonicity is the idea that a seam includes only one pixel for every row or column. Connectivity ensures that all pixels of these seams are connected.

7.3 Possible Solutions and Extensions

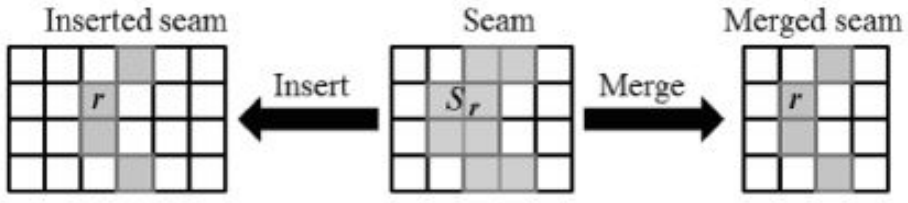
Since a graph implementation is above the scope of our current MATLAB expertise, we instead chose to extend the dynamic programming technique for seam carving through time. We are able to remove a temporally continuous array of seams from each consecutive frame in the input video. Although doing this works, it still produces artifacts when removing vertical seams with objects that move horizontally in the video. When executing our code on an example video of two planes flying horizontally, one of these planes became very distorted when it crossed the area containing most of the removed seams.

We brainstormed and came up with several possible solutions to this problem. Our first suggestion was to allow seams to vary by more than one pixel horizontally from frame to frame, which we tried but it did not make that much of a difference in the video. Our second idea was to calculate the optimal seam through all frames on each of N slices of the video. For example, where N = 4, we would split up the video into four slices:far left, middle left, middle right and far right. We would calculate the seam for

each one of these slices and then compare these seams to find the lowest overall array of seams of these four slices. This implementation takes N times the computation time, (when N = 4, four times the computation time), but could give better overall seams. N could also be equal to three, five, six, etc, depending on which one gives the best results. We chose to avoid this implementation because it would require too much computation time and wouldn't produce ideal results at borders between slices, but it still could make for a good future implementation.

Another possible extension would be to change the reference frame randomly in time throughout the video. Currently we calculate the optimal seam in the first frame and then bound the rest of the frames so that they are continuous in time. We saw in the video that this is a good choice for the first frame but less than optimal for the rest of the frames. Using N random different starting frames, we would calculate the ideal seam at a random reference frame. Then we could calculate the seam both forward and backward in time from that reference, and finally compare the total energy of all the seam arrays resulting from each starting frame and use the optimal, minimum energy one. This extension also costs N times the computation time, but also could give better overall seams because it wouldn't run into problems at the borders of every slice. We chose to implement this extension since it will be a beneficial tradeoff between time and improved results.

Perhaps the most beneficial extension involves calculating an optical flow field between each consecutive frame to add to the energy term. "Optical flow" takes into consideration the velocity of moving objects in the video. This extra field between each frame places higher energy at pixels that have motion. Adding optical flow to the gradient energy already used in our computation, fast-moving edges would result in the highest energy. Ideally, this results in seams that flow around the moving objects. Incorporating this extension will avoid the distortion caused by moving objects. However in order to implement this, we need to use the Computer Science lab computers since they are the only computers that have the computer vision toolbox for MATLAB. With our limited time, and already extensive computation times, we chose not to incorporate this extension.



We also wanted to look at and think about implementing seam merging. Seam merging helps preserve structure in order to avoid artifacts in the retargeted image. Instead of removing seams, merge a two pixel width seam into a one pixel width seam.

It is also possible to enlarge the image using this technique by inserting a new pixel in between the pixels of a seam. Seam merging uses forward energy to choose the seams. This technique also incorporates two other energy functions which help minimize distortion by avoiding excessive removing of pixels in important areas of the image. We did not incorporate this extension since it uses forward energy which requires a graph implementation.

The improved technique used in the “Improved Seam Carving for Video Retargeting” article uses forward energy enhancement. Forward energy, as previously mentioned, is the concept that a new cost value is calculated after removing seams which increases the accuracy of the pixel’s top neighbors’ minimum energy value. Conceptually, it produces beneficial and improved results. However, we chose not to use this extension since it requires graph implementation.

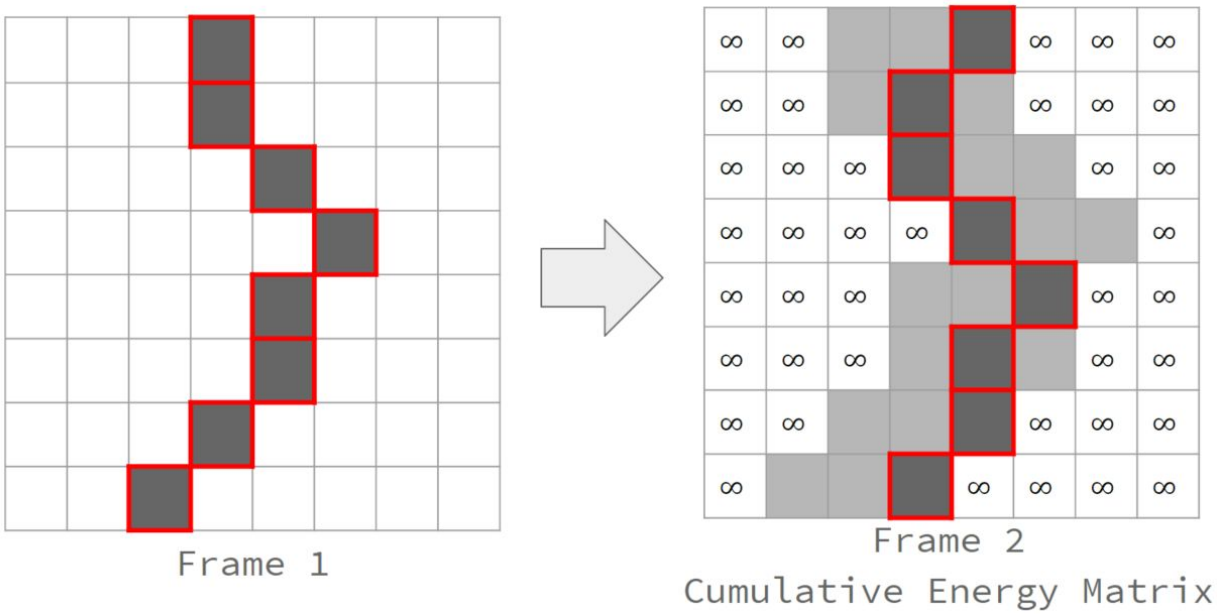
8. Method

We extended the dynamic programming technique for seam carving through time and incorporated ideas of increasing bound width, random reference frames, and coarse scale computation acceleration.

8.1 Implement Seam Carving

In order to apply the seam carving algorithm to three dimensional video, it is necessary to implement seam carving continuously in time. Using the video with individual image frames as input, convert them all into gray-scale frames. After these gray frames have been calculated, compute the energy frames by calculating the gradient using the sobel filter in both the horizontal and vertical directions. For each energy frame, sum the values along minimum energy paths of the energy frames to obtain the minimum cumulative energy matrix. Next, calculate the minimum energy seam of the first frame. For each consecutive frame, calculate its seam using the current frame’s cumulative energy and the previous frame’s seam. Each consecutive frame’s seam must be continuous in time meaning that it cannot vary horizontally by more than one pixel from each of the last seam’s pixels. Lastly, remove the optimal seam from the first frame and the bounded optimal seams from the rest of the frames. Repeat these steps until the desired number of pixels have been removed.

In the depiction below, the first frame's minimum energy seam (dark gray pixels with red border) is shown. The next frame's cumulative energy matrix is then bounded to within one pixel on either side of the prior frame's seam (light gray pixels). All energy values outside those bounds are set to infinity. Finally, the minimum energy seam for the second frame is then calculated from the bounded cumulative energy matrix and a possible resulting seam (dark gray pixels with red border) is shown.



8.2 Increased Bound Width

To help account for objects that move quickly between frames, we allowed for the increase of the bound width of consecutive frames. An object that moves horizontally at a rate of three pixels per frame will be sliced by a temporally continuous seam that only displaces by at most one pixel horizontally per frame, resulting in distorted and sometimes disappearing objects in the output. To account for this possibility, we modified our algorithm to allow the seams to be bounded by a larger pixel window defined by the user. For the example above, an object moving at three pixels per frame might require an 11-pixel-wide bound window to ensure that seams are able to displace faster in either direction than the object in the video.

8.3 Random Reference Frames

An idea to provide better results when removing a temporally continuous seam is to test random frames of reference. Our initial implementation calculated the ideal seam on the first frame, then calculated subsequent seams by bounding the seam of the next frame to be continuous with the frame before it. An ideal seam in the first frame may not be very representative of an ideal seam throughout the whole video, so testing multiple

reference frames would be beneficial. The seams are then calculated backward and forward in time starting from the ideal seam at multiple random reference frames. Then the seam with the least overall energy of the set tested is removed. This will allow the algorithm to remove lower energy seams, at the expense of needing to calculate multiple more seams before removing only one.

8.4 Coarse Scale Computation Acceleration

To enable faster computation while achieving almost identical results, we incorporated coarse scale computation acceleration. This is implemented by scaling the image down to calculate the seam. Then the miniature seam must be interpolated back to the size of the original video before being removed. In our implementation, we tested two types of interpolation for our final results - nearest neighbor pixel, and bilinear interpolation. For example, reducing the dimensions by a factor of 2, the image is half as wide and half as tall, meaning it now has a quarter of the total pixels. Since many of the energy and seam computations are directly dependent on the total number of pixels, this could reduce the computation time by a factor of 4.

9. Experimental Results

9.1 Temporally Continuous Seams

When compared to ideal, frame-independent seam removal, removing temporally continuous seams entirely corrected the jittery back and forth shifting of objects in the middle of the video. Objects now retain their original movement characteristics, but at the expense of some other artifacts. Continuous seams are more strict in that they must be bounded to a certain set of pixels, so they can't necessarily avoid large changes in the energy of consecutive frames. When removing vertical seams from a video with horizontally moving objects, continuous seams tend to perform poorly. As can be seen in the video results of the two planes, objects that quickly pass across seam boundaries tend to show a lot of distortion or even disappear momentarily when removing enough seams. This problem was addressed by varying the bound width of continuous seams.

9.2 Increased Bound Width

Increasing the bound width between subsequent seams would ideally allow seams to 'avoid' fast-moving objects by allowing them to displace faster than the object moves while still maintaining the non-jittery characteristic of temporally continuous seams. In practice, this did not show much better results. The increased bounded width initially allowed the seam to avoid passing through moving objects. In the case of the video with two planes, increasing the bound width didn't prove very effective. The planes move at different rates both throughout time, and relative to each other. To effectively make the seams avoid both moving planes, the bound width had to be

increased by up to 15 pixels on either side. With such a high seam window, consecutive seams tended to lose their continuous characteristics and resulted in a shaky characteristic.

9.3 Random Reference Frames

The implementation of random reference frames made the most dramatic difference in the visual appearance of the output. Computing five seams starting from different reference frames in the video of two planes resulting in generally improved output. Rather than having the predictable distortion of the planes as they crossed through seams that were removed from the middle of the frames, many seams were removed from the far right edge of the frames. We saw that the planes experienced much less distortion in the middle of the video, but the plane on the right showed significant distortion when approaching the right edge of the frames. While these results were distinctively improved, computing five times as many seams meant that the total computation time was increased by nearly five times as well. In hopes of reducing computation time, a coarse scale implementation was tested next.

9.4 Coarse Scale Computation Acceleration

The goal of this final optimization was to reduce computation time significantly to offset the cost of calculating multiple seams from different reference frames, while still retaining similar results compared to the original-size video computations. As can be seen in the result videos on our website, the output differs imperceptibly when scaling down by a factor of three. In practice, scaling any smaller led to significant visual differences, while scaling any less didn't make enough of a computation timing difference to be very useful. As a specific example, removing 100 vertical seams from the wakeboard video took about two hours to process with our initial continuous seam implementation. Converting to compute results from three different random reference frames increased that computation to about take about six hours. Using the coarse scale acceleration by reducing the video size to one ninth of the total pixels, the video was processed nearly nine times faster (in about 40 minutes), with no major visual differences.

The difference between nearest pixel neighbor and bilinear interpolation of the miniature seam turned out to be significant as well. Interpolating the miniature seam back to original size with nearest neighbor interpolation resulted in blocky-looking seams. This could be explained by the fact that every miniature seam pixel was replicated two times, so the seams could only displace once horizontally every three pixels vertically. This was the fastest interpolation computation, but it didn't yield very impressive results.

By simply changing an argument in an interpolation function, we also tested bilinear interpolation of the seams. This allowed a more continuous flow of seams from top to bottom. Visually this eliminated the blocky texture of seams from the nearest neighbor interpolation. However, this implementation did add about 5% to the computation time, since significant calculations were made rather than just duplicating values to interpolate. Given the extra 5% computation time, but significantly improved visual results, we concluded that this was the preferred type of interpolation.

Results can be seen on our website at http://msmurphy2.wix.com/cs534seamcarving Under the Project tab, there are two different results sections. One section contains the Planes videos and the other contains the Wakeboard videos.

10. Concluding Remarks

In conclusion, our project goal was to apply the seam carving algorithm to video. Since graph cuts were too complicated for our level of MATLAB skill, we instead extended the dynamic programming technique for seam carving through time. During the project, we unexpectedly ran into the issue of time constraints of processing the videos so we shifted our focus to accelerating computation time rather than outputting visually perfect videos. It took a long time to process some of these videos: the Wakeboard video took up to six hours on some trials. The bigger and longer the video, the more unrealistic and worthless it would be to actually apply to the video. This is why we looked at smaller videos clips. This was our reasoning for implementing the two main extensions- random reference frames and coarse scale computation acceleration.

As a remark, we wrote and tested code using MATLAB 2015a. MATLAB 2014a doesn't have the same functions which allowed us to load *.mov* video files. A relatively simple modification of the code could be made for older versions to load videos of different formats.

11. References

Adami, Nicola, Sergio Benini, and Masahiro Okuda. “Improved Seam Merging for Spatial and Temporal Video Retargeting.” Universita Degli Studi Di Brescia, 2013/2014. http://projects.i-ctm.eu/sites/default/files/PDF/708_Paolo%20Guglielmini/relation.pdf

Avidan, Shai and Ariel Shamir. “Seam Carving for Content-Aware Image Resizing”. SIGGRAPH. 2007. http://graphics.cs.cmu.edu/courses/15-463/2007_fall/hw/proj2/imret.pdf

Rubinstein, Michael, Ariel Shamir, and Shai Avidan. "Improved Seam Carving for Video Retargeting." Web. http://www.faculty.idc.ac.il/Arik/SCWeb/vidret/index.html

Tao, Xiaofeng. "Seam Carving." Web. http://cs.brown.edu/courses/cs129/results/proj3/taox/

12. Appendix

12.1 Description of Code

All code used in this project was written in MATLAB by our group. A few modified functions from HW 3 seam carving were used, but all other functions were written for the sole purpose of this project. Our project includes about 300 lines of logical code and about another 300 lines of comments and custom display scripts. The majority of the functions were made to convert the video between different formats. Many times, a function was written to, for example, convert a grayscale image to a gradient energy image. A corresponding function was then written to convert a grayscale video to a gradient energy video, using the image function on every frame. Three separate test scripts were used to compare the results obtained from the different implementations; basic temporally continuous seams, temporally continuous seams with random reference frames, and temporally continuous seams with random reference frames and coarse scale computation acceleration.

12.2 Member Roles

Erik Jorgensen

Erik spent the majority of his efforts brainstorming code implementations that could extend the ideas formulated by the group. He also converted the seam carving algorithms to work with video in MATLAB. Additionally, he helped write sections of the paper and provided valuable input for building a website.

Margaret Murphy

Margaret got a head start writing a majority of the paper as well as formed useful input for the brainstorming of the code and evaluating the video results. She also spent considerable time putting together the infrastructure for the website.

Aziza Saulebay

Aziza worked to organize the group's efforts from the beginning and helped set up team meetings for our consistent progress. She also worked to design the presentation slides so that the group could have a more coherent presentation. Aziza also helped to test the code and contributed to writing the paper as well as designing the webpage layouts.