ch

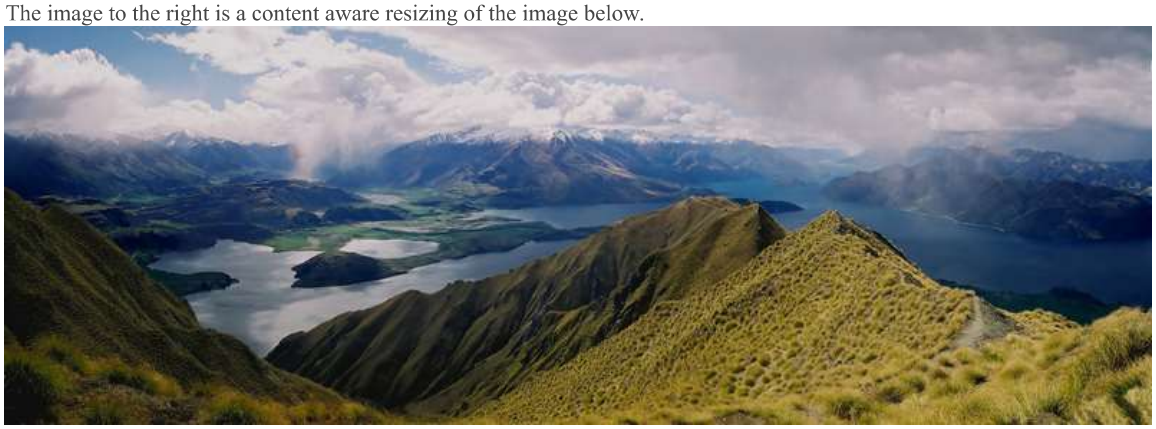
Seam Carving

by Xiaofeng Tao (taox)

Introduction

The objective of this seam carving algorithm is to perform content aware resizing of images. This allows image to be resized without losing meaningful content from cropping or scaling. The idea is to locate the image's optimal seams, connected pixel paths going from top to bottom or left to right, to remove or insert while preserving the photorealism of the image. Furthermore, manipulating the gradient energy map that describes how optimal a seam is allows for functionality such as object removal. The technique described below is an implementation of the algorithm presented in (Avidan and Shamir). In addition, an improvement to seam removal is implemented using forward energy and the algorithm is extended to run on video based on (Avidan, Rubinstein, and Shamir).

The image to the right is a content aware resizing of the image below.



Seam Removal Algorithm Overview

The following steps describe seam removal, which is the basis for all the different functionality described later.

1. Calculate energy map

For each color channel, the energy is calculated by adding the absolute value of the gradient in the x direction to the absolute value of the gradient in the y direction. The energy for all color channels is summed into one 2D image to create the energy map.

2. Find minimum seam from top to bottom edge

First, an accumulated cost matrix must be constructed by starting at the top edge and iterating through the rows of the energy map. The value of a pixel in the accumuluated cost matrix is equal to its corresponding pixel value in the energy map added to the minimum of its three top neighbors (top-left, top-center, and top-right) from the accumulated cost matrix. The value of the very top row (which obviously has no rows above it) of the accumulated cost matrix is equal to the energy map. Boundary conditions in this step are also taken into consideration. If a neighboring pixel is not available due to the left or right edge, it is simply not used in the minimum of top neighbors calculation.

The minimum seam is then calculated by backtracing from the bottom to the top edge. First, the minimum value pixel in the bottom row of the accumulated cost matrix is located. This is the bottom pixel of the minimum seam. The seam is then traced back up to the top row of the accumulated cost matrix by following. The minimum seam coordinates are recorded.

This step is implemented with dynamic programming. An additional enhancement to this step that gives a more accurate energy value is described in the forward energy section.

3. Remove minimum seam from top to bottom edge

The minimum seam coordinates from Step 2 are then used to remove the minimum seam. All the pixels in each row after the pixel to be removed are shifted over one column. Finally, the width of the image has been reduced by exactly one pixel.

4. Repeat Steps 1 - 3 until desired number of seams are removed

This process is repeated until the desired number of minimum seams are removed. The energy map needs to be recomputed everytime a seam is removed.

5. Repeat Steps 1 - 4 for left to right edge

Steps 1-4 are originally describing reducing the image's width. The same code can easily be used for reducing the image's height by simply taking the transpose of the input image.

Forward Energy

The forward energy enhancement is implemented by taking into account the energy created from the new neighbors after the seam is removed. The original algorithm is modified by adding this extra forward energy during the creation of the accumulated cost matrix. When deciding which top neighbor (top-left, top-center, top-right) in the accumulated cost matrix has the minimum energy value, the forward energy from the corresponding new neighbors is added to each top neighbor's accumulated cost value.

The benefits of forward energy usage can be seen below in how the angular shape of the mountains is preserved.

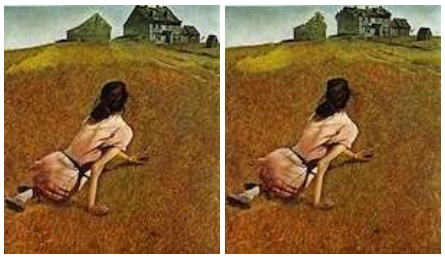
Input (top)

Forward Energy (left) vs No Forward Energy (right)

Here, the image with forward energy has a much better looking house.

Input (top)

Forward Energy (left) vs No Forward Energy (right)



Again, the angular shape of the mountains is preserved.

Input (top)

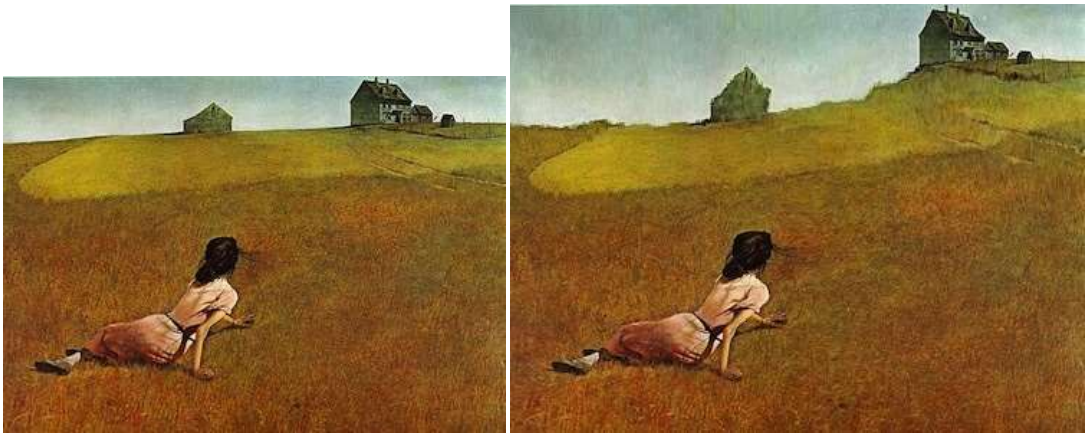
Forward Energy (left) vs No Forward Energy (right)

Seam Insertion

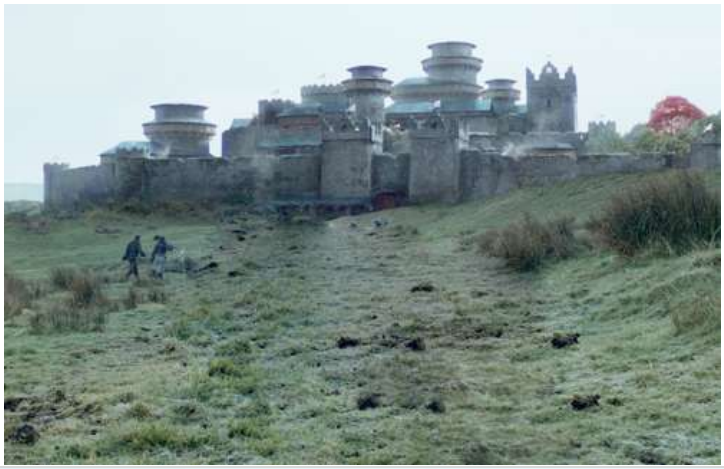
Seam insertion can be thought of as performing seam removal in reverse. If n seams are to be inserted, then seam removal must first be performed for n seams. The coordinates and order of these minimum seams is recorded during removal. The original image is not actually modified during this process, because the purpose is only to find the coordinates and order of the minimum seams. Finally, the coordinates are used to insert new seams to the original image in the same order that they were removed. The pixel values of the new seams are derived from an average of the top/bottom or left/right neighbors.

Some examples of seam insertion are shown below.

Input (left) vs Resized (right)



Input (top) vs Resized (bottom)



It is important to note that forward energy should **not** be used for seam insertion. This is because the forward energy from the new neighbors has no relevance to the expanded image. Actually, the original algorithm is sufficient for estimating the energy from the new neighbors in seam insertion, since the new seam is just an average. A comparison of the benefits of forward energy vs no forward energy in seam insertion is shown below. 50 pixels have been inserted to each dimension.

Forward Energy (left) vs No Forward Energy (right)

Object Removal

Object removal is implemented by first creating a binary mask during preprocessing of the object to be removed. First, the masked region's area is measured so it can decided whether it is more efficient to remove the object by removing top to bottom seams or left to right seams. Then, when generating the image's energy map, areas under the masked region are weighted with a very high negative value. This guarantees that the minimum seam will be routed through the masked region. Minimum seam removal is then performed as usual. The only difference is that when removing the minimum seam from an image, the same minimum seam must also be removed from the mask so that the next step's energy calculation is accurate. After the masked region has been completely removed, seam insertion is used to return the image back to it's original dimensions. Forward energy is used during the seam removal but not during the seam insertion.

An example of object removal is shown below.

Input/Mask (Top) vs Object Removed (bottom)

Video

Seam removal is also implemented in video by taking into consideration the dimension of time. The algorithm used is similar to that used for a 2D image, except it is being performed on a 3D block of video frames, with time as the z axis. The energy map used is actually a combination of the spatial and temporal energies. The spatial energy map is calculated by doing the energy map calculation described in the algorithm overview on all video frames and then finding the max at each pixel coordinate. The temporal energy map is calculated by taking the gradient between the frames (along the z axis) for each pixel. The spatial and temporal energies are then merged with a linear combination:

*energy_map = (alpha)*spatial_energy + (1 - alpha)*temporal_energy*

Therefore, an *alpha* value of 0.5 would be the same as averaging the two energies together. An *alpha* value of 0.3 is used as suggested in (Avidan, Rubinstein, and Shamir). The reasoning behind weighting the temporal energy higher is that the human eye is more sensitive to motion artifacts. After the energy map is found, the minimum seam is calculated. Finally, the same minimum seam from all frames in the video are removed.

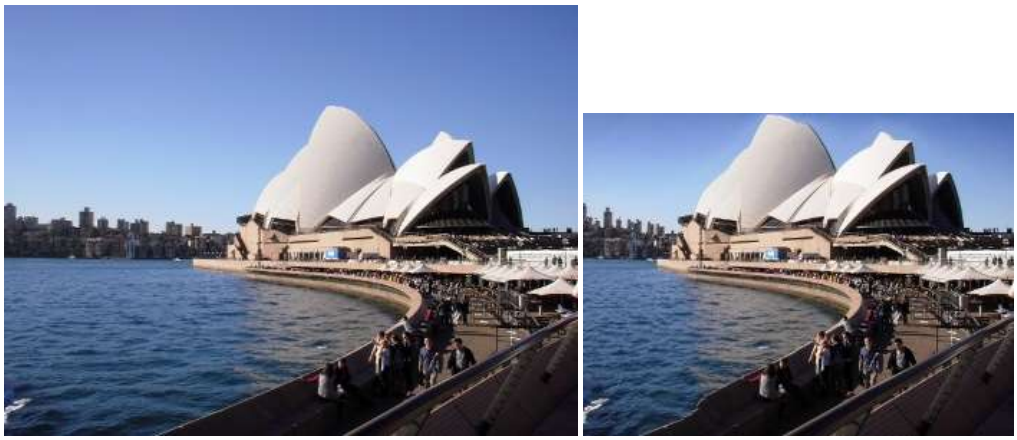
An example is shown below of the results of this algorithm on video. The resulting video has 100 pixels removed from the width of the original video.

Bear Video

Bear Video With Reduced Width

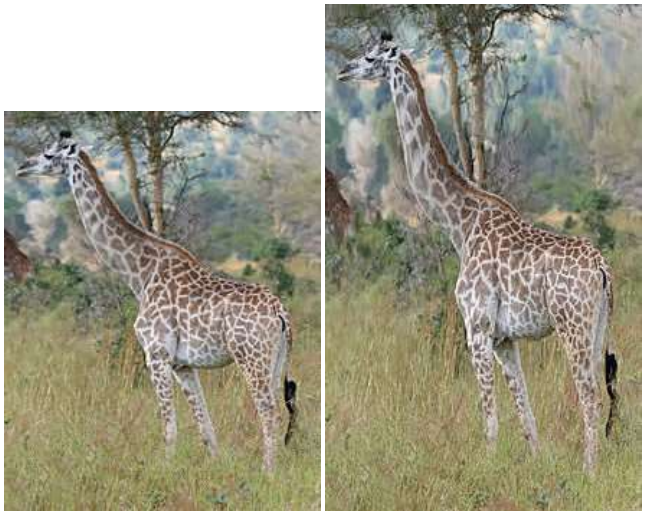
Results from Original Images

Input (left) vs Resized (right)



Input (top) vs Resized (bottom)

Input (left) vs Resized (right)



Failure Case

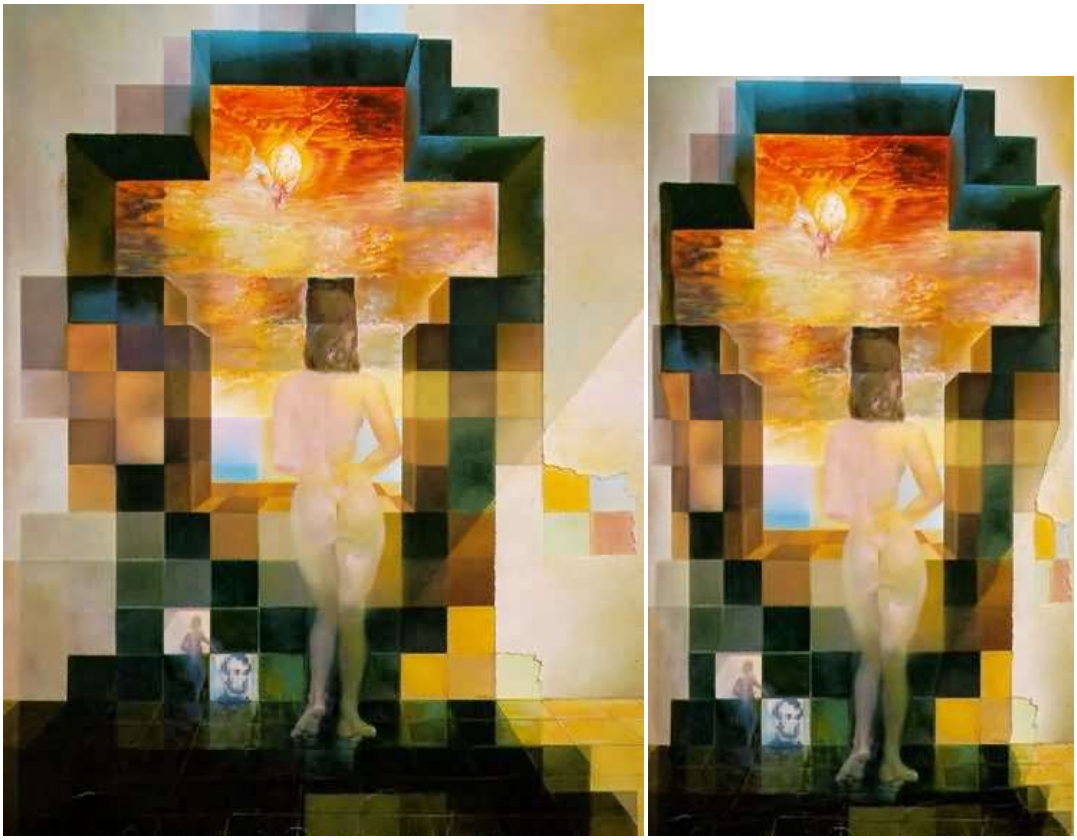
This is clearly a failure case, because Larry Bird and Magic Johnson have become distorted instead of the background. This is probably because some areas of the players' skin and uniforms have lower energy than the crowd areas in the background. The gradient in these areas is relatively smoother than in the crowd. Also, the algorithm does a poor job of preserving the shape of their legs, especially where the legs are overlapping. A possible algorithm enhancement would be some way of telling that algorithm that these areas are important so that the crowd is removed instead.

Input (left) vs Resized (right)

Results from Default Images

Input (left) vs Resized (right)

Input (left) vs Resized (right)

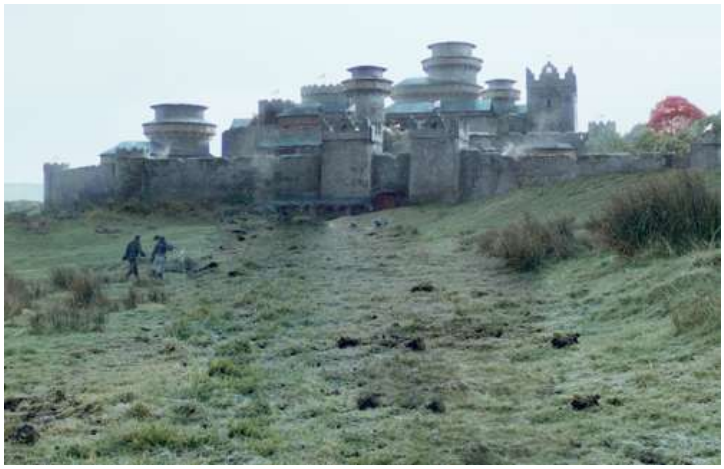


Input (left) vs Resized (right)

Input (left) vs Resized (right)

Input (left) vs Resized (right)

Input (left) vs Resized (right)



Input (left) vs Resized (right)