CONTENT-AWARE VIDEO- FRAME RESIZING USING SEAM CARVING

M. Clara De Paolis Kaluza,
Sutawat Poomcharoenwatana

Dec 14, 2009

Boston University

Department of Electrical and Computer Engineering

Technical report No. ECE-2009-07

CONTENT-AWARE VIDEO- FRAME RESIZING USING SEAM CARVING

M. Clara De Paolis Kaluza, Sutawat Poomcharoenwatana

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
`www.bu.edu/ece`

Dec 14, 2009

Technical report No. ECE-2009-07

Summary

Video acquisition can quickly generate large amounts of data that can be stored easily in local memory. However, it is undesirable to transmit large amount of data from the acquisition site to a remote receiver. The amount of data can be reduced either spatially or temporally, or both, thereby reducing transmission costs. While many solutions exist, we take advantage of the fact that video sequences often contain information that is not relevant to the end user. Content-aware retargeting of images and videos allows for the transmitter to only send data that is relevant to the end-user.

In this case, content-aware spatial retargeting of environmental monitoring video data is considered. We achieve spatial retargeting through seam carving of video frames as described by [Avidan, et. al] for single images. Seam carving allows for the removal of the lowest energy connected, monotonic paths in an image. Removing such paths in a still image produces a spatially-reduced version of the image with less relevant (lowest cost) areas (collection of seams) removed. By identifying objects of interest, the cost of areas containing these objects may be modified so that seam carving preserves the objects. A video sequence of seam-carved frames may be used to transmit environmental monitoring video as a reduced rate. Although object proximity data may be lost, temporal data of object interaction or object presence may be observed.

However, in a sequence of frames that make up a video, arbitrary seams cannot be removed without introducing considerable artifacts such as jitter and rapid frame-size change from frame to frame. We define a subsequence as a subset of the total set of frames in the sequence. By summing the costs associated with each pixel across an entire subsequence, a set of static seams can be removed from all frames in that subsequence. This approach restricts the minimum size of frames in a subsequence to the limitations imposed by all the frames. However, this static-seam subsequence approach allows for lower computational complexity and periods of size cohesion between frames. Adjusting the length of the subsequence allows for a trade-off between the lengths of time of size stability in the sequence and the minimum cost across all frames in the subsequence.

Contents

List of Figures

1 Introduction

Current video acquisition techniques allow for quick generation of large data sets that are easily stored in local memory. The transmission of this data from an acquisition site to a remote receiver comes with costs that are directly related to the data size. The amount of data can be reduced either spatially, temporally, or both thereby reducing transmission costs incurred. In this paper, we propose a method which takes advantage of the fact that video often contains information that is irrelevant to the end user.

Content-aware retargeting of images and videos, allows for greater efficiency through the transmission of only relevant data to a receiver. Specifically, content-aware spatial retargeting of data from environmental monitoring data is considered. This is achieved through a technique known as seam carving, which allows for the removal of the lowest energy connected, monotonic paths in an image. The lowest energy areas correspond to the less relevant regions of the video frame. By identifying the objects of interest, the image may be seam-carved to preserve the most relevant objects and reduce the amount of irrelevant data transmitted. A video sequence of seam-carved frames may be used in environmental monitoring. Although object proximity data may be lost, temporal data of object interaction or object presence may be observed.

1.1 Prior Work

Attempts to reduce image and video data transmission have been implemented using many different methods. In [1], seam caving is proposed as a method of content-aware resizing of still images. The authors define seams, on which we base our definition, as connected and monotonic least-energy paths. Several energy functions are explored in this paper, including the gradient function which we use. Among the other energy functions considered, the authors conclude that all behave comparably and no one function performs better for all image types. Seam removal is used to decrease the size of an image, but the authors also expand the least-cost seam selection to inserting of least cost seams in order to expand an image. We do not consider this as we are only concerned in reducing data transmission. Lastly, [1] addresses object removal and object preservation by applying masks that indicate infinite or zero cost regions to influence seam selection.

In [2], the authors propose improved seam carving for video retargeting using forward energy seam carving. Removing a seam from an image introduces new pixel neighbors which may add energy to the image. Forward energy seam carving attempts to minimize this energy by calculating the introduced energy before actually removing a seam. A seam is therefore chosen so as to introduce the least cost after removal. This method improves results overall and thetechnique is extended to video retargeting. [2] also introduces graph cuts in combination with seam carving for video retargeting. A graph cut separates a graph (or an image in this case) into two disjoint subsets. This cut is defined by the optimal seam. In order to implement this for natural videos, the seams must be monotonic and connected over all frames.

Lastly, in [3], seam carving is extended to three dimensions through ribbon carving. Ribbon carving is introduced as a means to reduce video sequences temporally rather than spatially by condensing a video based on motion presented in frames. Rather than removing seams to decrease the size of a video, ribbons are removed to preserve motion throughout the video sequence. However, we only consider special reduction in this paper.

1.2 Assumed Constraints

The concepts discussed may be implemented using RGB components, but in the interest of computation time, here we only consider grayscale images. Also to save on computation time, we calculate only vertical seams but horizontal seams may be used in addiction or instead using the same concepts.

2 Implementation

To implement our content-aware resizing approach, we assign energy costs to each pixel in each image (representing a frame in a video sequence). Once the costs are known at every pixel, a least-cost seam may be identified and removed. The single-image implementation is then extended to video sequences.

2.1 Assigning Cost

The cost is given by the sum of the absolute values of the horizontal and vertical gradients of the intensity of the grayscale image:

$$\text{cost} = \left| \frac{\partial}{\partial x} I \right| + \left| \frac{\partial}{\partial y} I \right| \tag{1}$$

The gradient as described by equation (1) is calculated for every pixel. For each frame, there may be objects which should be preserved after seam carving. These objects detected and identified as relevant in each frame are assumed to be detected by a separate algorithm and pixel locations of these objects are assumed to be known at the outset of our project. That is, an object detection algorithm is used to find the pixel locations that belong to a desired object. These pixel locations are input to our system. Positions at which desired objects exist are assigned higher energy costs by adding a significantly high offset. In our system, typical values for gradients are on the order of a few hundred (minimum zero, maximum 500), therefore, a significantly high offset is on the order of 1000. Offsetting the cost by such a high value ensures that when removing least-cost seams, pixels which belong to a desired object will be avoided. In most natural images, the maximum value will not be common throughout the image because there are many areas with relatively constant intensity, a smaller offset could therefore be used.

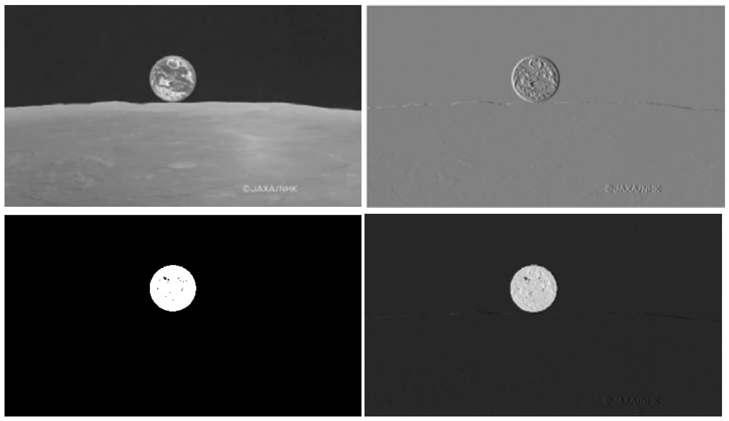


Figure 1: Clockwise from top left: an original grayscale image; a gradient map of the image; an object identified and detected (white pixels indicate pixels belonging to the object); a final cost map of the image which is the original gradient map plus the object cost offset

Source: http://www.space.com/common/media/video/player.php?videoRef=080414-Kaguya&mode=

2.2 Seam Selection

We define seams to be monotonic (exactly one pixel per row, or column) and connected vertical or horizontal path from one side of the frame to the opposite side as described by [Avidan, et. al] and [Rubinstein, et. al]. The seam is grown from the top for vertical seams starting at pixel location (1, j) and the next pixel is chosen between pixels at locations (2, j-1), (2, j), and (2, j+1) such that the sum of the costs of the first pixel and the second is minimized. In general for a vertical seam, for a pixel (i, j), the next pixel is chosen from (i+1, j-1), (i+1, j), and (i+1, j). Horizontal seams are created similarly where pixel (i, j) leads to either (i-1, j+1), (i, j+1), or (i+1, j+1). The cost of a seam is given by the sum of the cost values of each pixel in the seam. The order in which the seams are selected to be removed is such that the lowest cost seam is chosen first, followed by seams of increasing costs. This approach reflects the backward energy selection criteria with which seams are chosen in [Avidan, et. al]. Backward energy refers to the fact that this seam selection method seeks to minimize the removed energy with every seam. The alternative to the backward energy approach is a forward-energy selection criterion as described by [Rubinstein, et. al] which seeks to minimize energy introduced *after* a seam is removed.

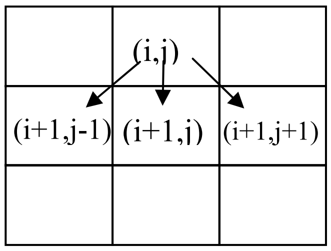


Figure 2a: Next pixel selection: possible next pixels shown at row i+1

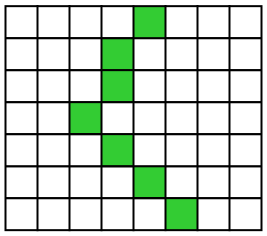


Figure 2b: Example of a valid seam (both monotonic and connected)

While backward seam energy does reasonably well in preserving content, another approach may be considered for improved performance. When a seam is removed from an image, new pixel neighbors are introduces. These new neighbors may add new energy to the cost of the overall image. A way to minimize this energy is by implementing a method called forward energy seam carving, which aims to reduce the *introduced* energy rather than the *removed* energy.

A seam has three possible paths to travel (downward left, downward, and downward right). We use the following equations to calculate cost of the introduced energy for each possible path:

$$\begin{aligned} C_L(i,j) &= | I(i,j+1) - I(i,j-1) | + | I(i-1,j) - I(i,j-1) | \\ C_U(i,j) &= | I(i,j+1) - I(i,j-1) | \\ C_R(i,j) &= | I(i,j+1) - I(i,j-1) | + | I(i-1,j) - I(i,j+1) | \end{aligned}$$

Both the downward left and downward right scenarios ($C_L(i,j)$ and $C_R(i,j)$) will introduce two new energy values while the downward case will only introduce one new energy value. These values are then added to the original cost matrix:

$$\begin{aligned} M(i,j) = C(i,j) + \min(&M(i-1,j-1) + C_L(i,j), M(i-1,j) \\ &+ C_U(i.j), M(i-1,j+1) + C_R(i,j)) \end{aligned}$$

As the equation shows, the new seam will choose the minimum cost path of the old cost matrix plus the new costs from the introduced energy from seam removal. It is possible to add a user specified cost to the matrix at that pixel location ($C(i,j)$). These locations can be determined either by user selection or an object detection algorithm. Costs at these locations can either be increased to preserve content or decreased to remove content.

If implemented using only vertical or only horizontal seam removal, each seam is removed from the original image resulting in an image that is spatially reduced by one total row or column. The cost function for the new image is calculated before identifying the next lowest-cost seam to be removed. When the application requires spatial reduction in both the horizontal and vertical directions, the lowest cost vertical seam is calculated and removed, the gradient is recalculated and a horizontal seam is removed, and the process repeats. We prove functionality for removing seams in both directions, as shown in figure 2b but restrict other tests to only vertical seam removal for calculation speed and simplicity.

2.3 Video Sequences

For single images in our implementation, resizing depends only on the current image. We extend this to an implementation to resizing depending on several frames in a video sequence. We define a subsequence as a set of frames from a given video sequence. A global energy cost function is defined as the sum of all energies in the subsequence. That is, for i frames,:

$$\text{global cost } = \sum_i \left(\left| \frac{\partial}{\partial x_i} I \right| + \left| \frac{\partial}{\partial y_i} I \right| \right) \tag{2}$$

where x_i and y_i are pixel positions for a given frame. Seams are identified as least-cost such that this global cost is minimized. The above cost function assumes that no objects have been detected for preservation. However, the same concept applied to a single image may be extended to a video sequence and a cost offset by some amount so as to preserve a desired object (that might even be moving across frames) may be applied.

The set of seams are static across the subsequence. That is, a static seam is identified for all frames in a subsequence and is lowest cost if the sum (across all frames) of the energy of the pixels is minimized. Removing static seams from a subsequence will ensure consistent consecutive frame sizes across the subsequence. Moreover, removing a static seam will cause a similar location in the scene to be distorted, rather than removing seams from different locations in the scene from frame to frame. The size of frames will be consistent across a subsequence and allowed to change for the next subsequence. The maximum size may be set by the user, or dependent on the area populated by objects of interest. Choosing a longer subsequence results in greater consistency over time (of the area that is distorted in all frames and of the size across the frames). However, if an object that is intended to be preserved moves across a large area of the field of view across frames, fewer seams may be removed while preserving the object. In such a case, a shorter subsequence might be desired.

3 Experimental results

First, we look at seam carving a single image before moving on to video sequences. Figure 2 below shows the least cost seams for the image. It can be seen that the seams avoid the table leg and, for the most part, sharp borders on the arms. This result is a consequence of minimizing the gradient in the horizontal and vertical direction. In our current implementation, we do not consider detected objects, thereby allowing possible distortion of objects of interest, such as the arm in this image. Both images have been seam-carved using the backwards energy approach.

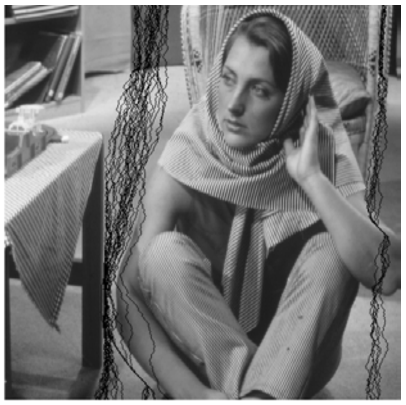


Figure 3a: The above image represents the original image with the first thirty lowest-cost seams identified

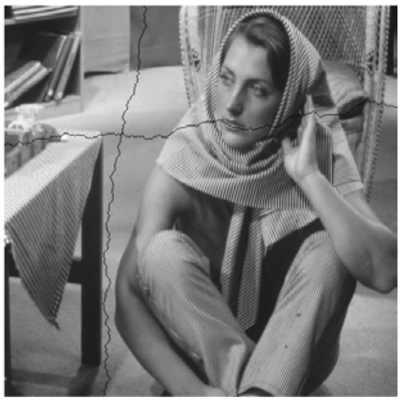
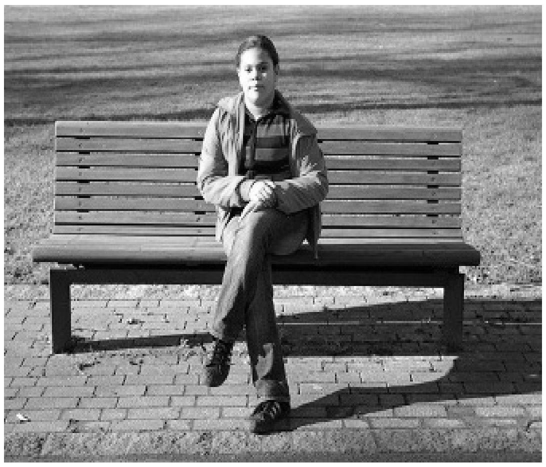


Figure 3b: The above image represents the original image with the first horizontal and first vertical seams identified, before being removed.

Comparing backward energy versus forward energy, we noticed that the backward energy approach introduces many more distortions. These distortions are very noticeable in the Figure 4. The girl's face is highly distorted compared to the result for forward energy seam carving. As stated earlier, this result is due to backward energy not minimizing the introduced energy from removing a seam. However, this minimization is much less important in images with large amounts of low cost areas. This can be observed in Figure 5. This figure shows that both methods produce similar results. While the lamp is somewhat distorted in the backward energy result, both results still maintains most of the image content.



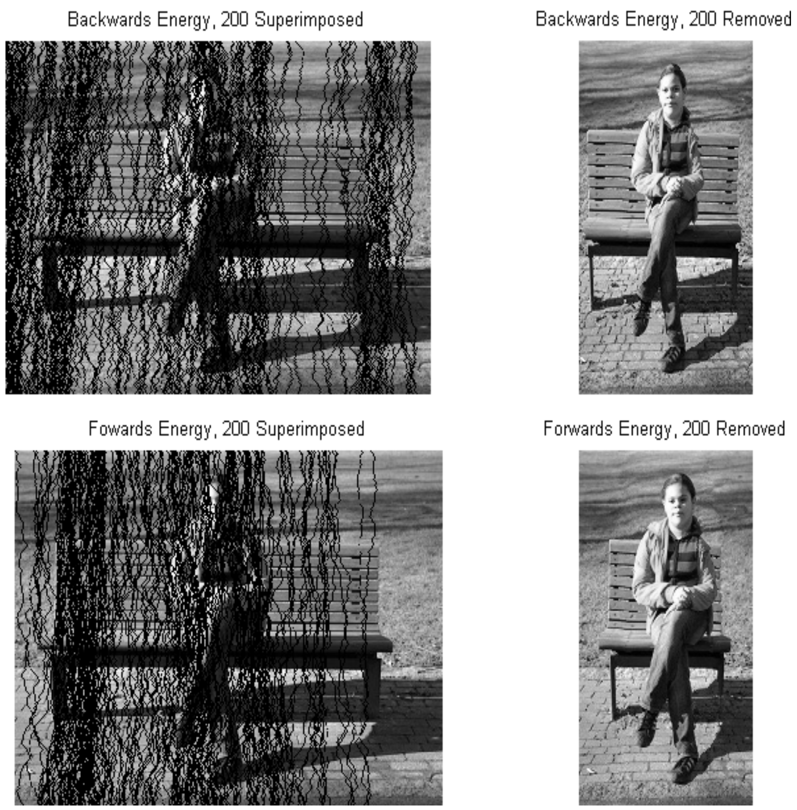
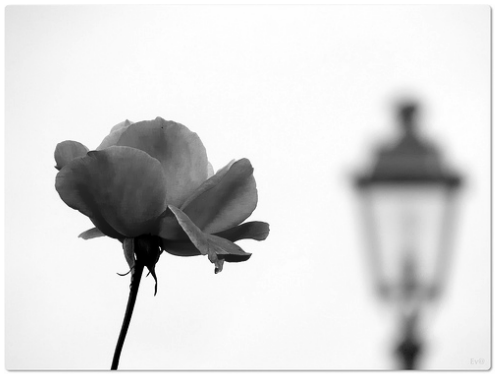

Backwards Energy, 200 Superimposed
Backwards Energy, 200 Removed
Fowards Energy, 200 Superimposed
Forwards Energy, 200 Removed

Figure 4: Shows the difference between forward and backward energy approaches to seam carving the same image, with 200 seams removed. The original image is shown above.



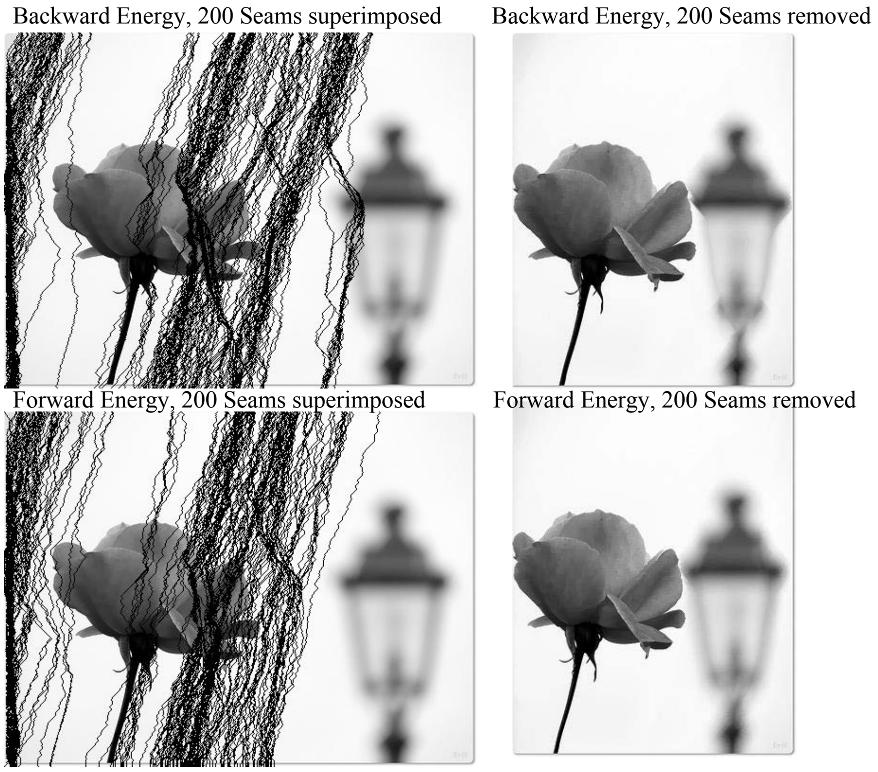

Backward Energy, 200 Seams superimposed
Backward Energy, 200 Seams removed
Forward Energy, 200 Seams superimposed
Forward Energy, 200 Seams removed

Figure 5: This image shows backward and forward energy minimizing approaches. For this image, both methods roduce very similar results. The original image is shown above. The backward energy minimizing method is shown at the top right and the forward energy minimizing approach is shown on the bottom right.

Lastly, we look at results for video sequences. Figure 6 shows frames from a video sequence that has been seam carved using static seams. We specify our subsequence to be a length of ten frames. As shown below, this choice of subsequence length gives unique seams every ten frames. The seams are imposes onto the original video.

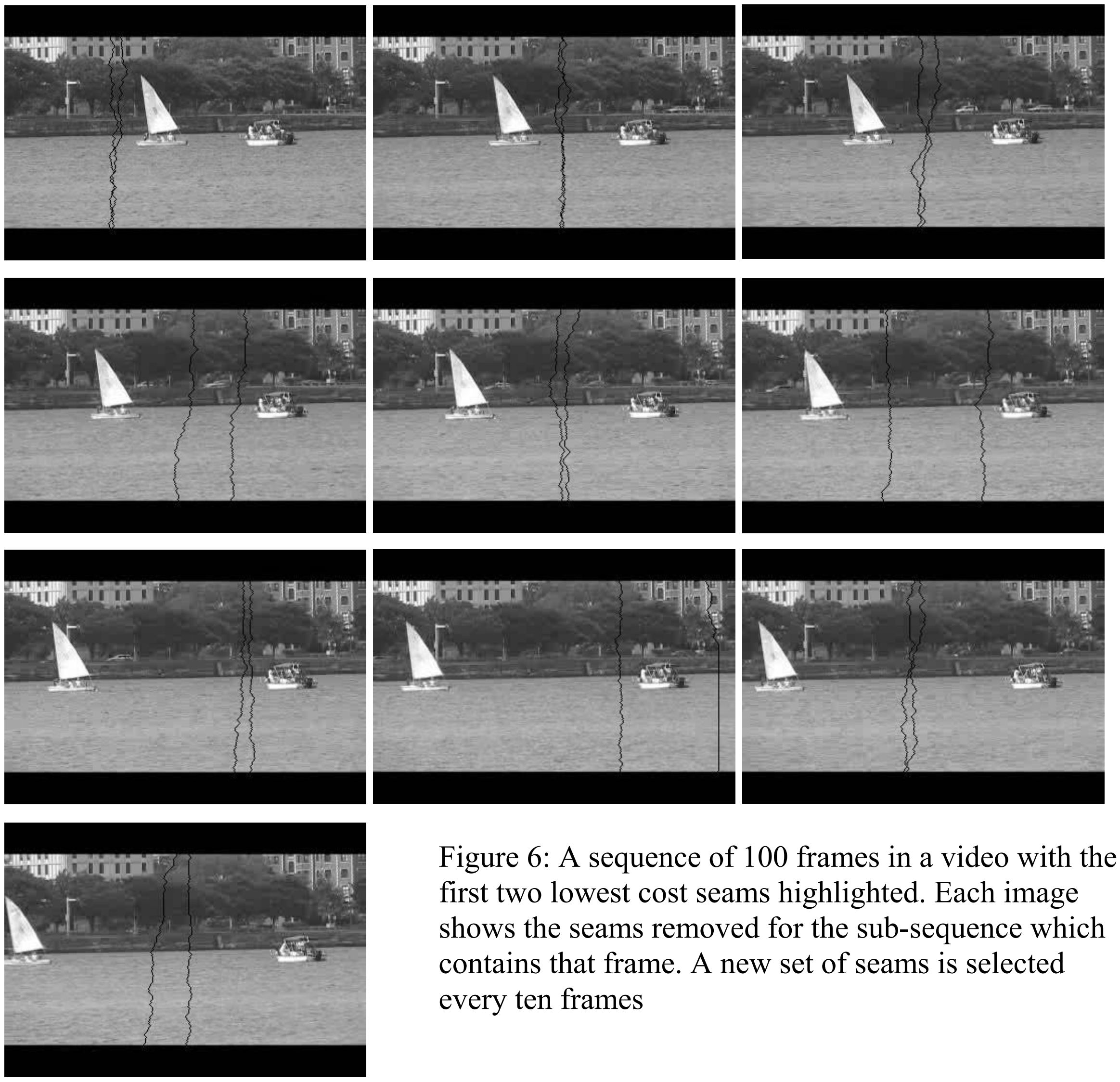


Figure 6: A sequence of 100 frames in a video with the first two lowest cost seams highlighted. Each image shows the seams removed for the sub-sequence which contains that frame. A new set of seams is selected every ten frames

4 Conclusions

Many solutions exist for content-aware special retargeting of images and videos. Here we considered mathematically and computationally simple methods Although more complex cost functions and methods exist, our approach proved robust in preserving content in an image. One Improvement to our method may be made by considering a different cost function that may increase performance of seam carving for some images as explored in other works. Another improvement is in implementing graph cuts. Graph cuts are used in order to preserve monoticity and connectivity of seams across multiple frames in a video. While our methods reduce video sizing by resizing video dimension, Video Condensation by Ribbon Carving [3] is another viable way to reduce transmission size/length. Rather than removing seams to decrease the size of a video, ribbons are removed to preserve motion throughout the video sequence.

Appendix

Below is listed Matlab source code developed for this project

```
%%Main Program
clear all
[image map] = imread('barbara.tif'); %Load the image

seam = 100; %Define the size of the image.

%Calls the ForwSeam function to perform forward energy seam carving. (See
%ForwSeam.m for details.
[fimposed fremoved] = ForwSeam(image,seam);
%Calls the Backseam function to perform backward energy seam carving. (See
%BackSeam.m for details.
[bimposed bremoved] = BackSeam(image,seam);

%Plot the Backward and Forward energy seam carving results in a 2x2 subplot.
figure
subplot(2,2,1)
imshow(bimposed,[])
title(['Backward Energy, ', num2str(seam), ' Superimposed'])

subplot(2,2,2)
imshow(bremoved,[])
title(['Backward Energy, ', num2str(seam), ' Removed'])

subplot(2,2,3)
imshow(fimposed,[])
title(['Foward Energy, ', num2str(seam), ' Superimposed'])

subplot(2,2,4)
imshow(fremoved,[])
title(['Forward Energy, ', num2str(seam), ' Removed'])
```

```
function [imposed removed] = BackSeam(barb,seam)
% Backward Energy Seam Carving
%barb = rgb2gray(barb %Uncomment this if picture has RGB components.

%Define the original image.
barb = double(barb);
barb2 = barb;
[r c] = size(barb);

%Use r or c depending on which seam you are calculating (hor vs. ver)
globalind=zeros(r,2);
for g = 1:seam
    %Define the gradient of the image as the cost matrix of the image.
    [FX,FY] = gradient(barb);
    FX = abs(FX);
    FY = abs(FY);
    F = FX + FY;
%Recalculate the dimensions of barb everytime a seam is removed.
[r c] = size(barb);
globalind_opt = zeros(r,2);

%This returns a vector with the seam index. See vseam.mat for more details.
for col=1:c

    globalind_opt = vseam(col,F,globalind_opt);

end

barb_new = zeros(r,c-1);

%This for loop will remove the seam from an image. It is necessary to convert
%the image into a 1-D vector since Matlab cannot remove from a single
%element from an error without returning an error.
for h = 1:r
    vec = barb(h,:);
    vec(globalind_opt(h,1))= [ ];
    barb_new(h,:) = vec;
end
barb = barb_new;

%This for loop will remove the seam from a image. It is necessary to convert
%the image into a 1-D vector since Matlab attempts to remove an element one
%at a time from the image matrix.
for i = 1:r
    for j = 1:c

barb2(i,globalind_opt(i,1)) = 0;
    end
end
end
```

```
%Returns the orignal image with seams imposed.
imposed = barb2;
%Returns the original image with seams removed.
removed = barb;
function [imposed removed] = ForwSeam(barb,seam)
% Forward Energy Seam Carving
%barb = rgb2gray(barb); %Uncomment if using color image
I = double(barb); %Convert to double
[r c] = size(I); %Size of image
for g=1:seam %Number of seams to be removed

[r c] = size(I);

[FX FY] = gradient(I); %Generate gradient matrix
M = abs(FX) + abs(FY); %Add together to consider both horizontal and vertical
gradients

%Add forward energy cost to the original cost (gradient) matrix.
for j = 1:c
    for i = 1:r

        if i == 1 %If pixel is at the top of the image, do nothing

        elseif j == 1 %If pixel is on left border of the image

        C_u = I(i,j+1);
        C_r = M(i,j+1) + abs(I(i-1,j)-I(i,j+1));
        M(i,j) = min([(M(i-1,j)+C_u) ((M(i-1,j+1))+C_r)]);

        elseif j == c %If pixel is on the right border of the image
        C_l = M(i,j-1) + abs(I(i-1,j)-I(i,j-1));
        C_u = I(i,j-1);
        M(i,j) = min([(M(i-1,j-1)+C_l) (M(i-1,j)+C_u)]);

        else
        C_l = abs(I(i,j+1)-I(i,j-1)) + abs(I(i-1,j)-I(i,j-1));
        C_u = abs(I(i,j+1)-I(i,j-1));
        C_r = abs(I(i,j+1)-I(i,j-1)) + abs(I(i-1,j)-I(i,j+1));
        M(i,j) = min([(M(i-1,j-1)+C_l) (M(i-1,j)+C_u) (M(i-1,j+1)+C_r)]);
         end

    end
end


%Initialize variable
globalind_opt1 = zeros(r,2);


for col = 1:c

    %This function returns the optimal seam. It takes in the cost matrix,
    %the old seam, and which column to perform seam carving on. See
    %forwvseam.m for details
    globalind_opt1 = vseam(col,M,globalind_opt1);

end

%Intialize the variable
I_new = zeros(r,c-1);
%Remove the seam from the image.
for h = 1:r
    vec = I(h,:);
    vec(globalind_opt1(h,1))= [ ];
    I_new(h,:) = vec;
end

I = I_new;

%Highlight where the seam is in the original picture.
for i = 1:r
barb(i,globalind_opt1(i,1)) = 0;
end

end
imposed = barb;
removed = I;
```

```
function globalind_opt = vseam(col,F,globalind_opt)
%This function returns the vertical backward energy seam column index.
%First, it recursively finds one seam based on the starting column. This
%seam is then stored. The next seam at the next column is then calculated
%and compared against the old one. Depending on which has the least total
%minimum cost, the larger one is dumped.

    %Initialize variables
    globalind_old = globalind_opt;
    ind = 1;
    [r c] = size(F);

    %Calculates the seam column index for each row.
    for jt=1:r
        %whatis_col and whatis_vector is used to ensure monoticity and
        %connectivity of the seam. Please see whatis_col.m and
        %whatis_vector.m for further details.
        col = whatis_col(col,ind);
        ivec = whatis_vector(c,jt,F,col);
        [minC, ind] = min(ivec);
        globalind(jt,1) = col;
        globalind(jt,2) = minC;



    end

    %The following code will determine if the current calculated seam has
    %lower cost than the old calculated seam. If so, it will replace it and
    %become the old calculated seam. If not, the old seam will stay
    %unchanged.
    if sum(sum(globalind_old)) ~= 0;

        sumold = sum(globalind_old(:,2));
        sumnew = sum(globalind(:,2));
        [dummy opt] = min([sumold sumnew]);
        if opt == 1;
            globalind_opt = globalind_old;
        else
            globalind_opt = globalind;
        end

    else
        globalind_opt = globalind;
    end
```

```
function ivec = whatis_vector(maxC,jt,FY,col)
%This function takes in the position of the current pixel(jt and col).
%It then returns what possible paths the seam can take. The restrictions
on
%these paths is that the seam has to be both monotonic and connected. This
%program also considers boundary conditons (the edge of the image) and
%returns the appropriate paths.

       if col == 0
       elseif col == 1; %left-hand boundary condition
          b = FY(jt,col);
          c = FY(jt,col+1);
          ivec = [b c];
       elseif col == maxC; %right-hand boundary condition
          a = FY(jt,col-1);
          b = FY(jt,col);
          ivec = [a b];
       elseif col > maxC %right-hand boundary condition
          a = FY(jt,col-2);
          b = FY(jt,col-1);
          ivec = [a b];
       else
          a = FY(jt,col-1);
          b = FY(jt,col);
          c = FY(jt,col+1);
          ivec = [a b c];
       end
```

```
function col = whatis_col(it,ind)
%Returns the column position of the seam depending on the input variables
%ind and it.
       if it == 1;
          col = ind;
       elseif ind == 1;
          col = it-1;
       elseif ind == 2;
          col = it;
       elseif ind == 3;
          col = it+1;
       end
```

```
function globalcostmat = costMatAdder(image)
%Takes in a 3-D Matrix and add the gradient (cost matrix) of each frame and
%adds them together to output the global cost matrix.
[r c t] = size(image);
dummy = zeros(r,c);
for frame = 1:t
    [Fx Fy] = gradient(image(:,:,frame));
    dummy = dummy + abs(Fx) + abs(Fy);
end

globalcostmat = dummy;
```

```
%% BackSeamVid.m
% This program performs static frame video resizing using backward energy
% seam carving. It loads a video from a .mat file (basically a 3-D matrix
% of video frames.


image = frameseq;        %Define the original video.
seamimp = image;         %Defined the original video to have seams
superimposed.
[r c time] = size(image); %Define the original video dimensions.

for g = 1:6 %This for loop determines how many seams to be removed.

image_fin = zeros(r,c-1,time); %Initialize variable
    %The following for loop is the "static frame" part of this program. The
    %index variable will tell the program what set of frames to find the
    %global cost matrix with.
    for index = 1:time/10

        if index == 1

    F = costMatAdder(image(:,:,1:10)); %See costMatAdder for details.

        else

            F = costMatAdder(image(:,:,((index-1)*10):(index*10)));

        end

    for t = 1:time %This for loop determines how many frames will be processed.


[r c dum] = size(image); %Calculate new video size.
globalind_opt = zeros(r,2); %Initalize the variable that stores seams.


%Perform seam carving using our global cost function.
for col=1:c

    globalind_opt = vseam(col,F,globalind_opt); %See vseam.mat for details

end

image_new = zeros(r,c-1); %Intialize variable

%This for loop will remove the seam from a frame. It is necessary to convert
%the image into a 1-D vector since Matlab attempts to remove an element one
%at a time from the image matrix.
```

```
for h = 1:r
    vec = image(h,:,t);
    vec(globalind_opt(h,1))= [ ];
    image_new(h,:) = vec;

end

%Redfine the original image with the seam removed.
image(:,1:c-1,t)=image_new;

if index == 1
    image_fin(:,:,t) = image(:,1:c-1,t);
else
    image_fin(:,:,t) = image(:,1:c,t);
end

% To superimpose the seam on original frame: It uses the seam variable to
% redfine the old frame with high intensities where the seam would be
% removed.
if index == 1

    for i = 1:r
    seamimp(i,globalind_opt(i,1),1:10) = 0;
    end

else

    for i = 1:r
    seamimp(i,globalind_opt(i,1),((index-1)*10):(index*10)) = 0;
    end
end

    end

    image = image_fin;
        [r c dum] = size(image);
    end
end
```

References

[1] S. Avidan, and A. Shamir, “Seam carving for content-aware image resizing,” *ACM Trans. Graph*, vol. 26, no. 3. 2007.

[2] M. Rubinstein, S. Avidan, and A. Shamir, “Improved seam carving for video retargeting,” *ACM Trans. Graph*, vol. 27, no. 3., 2008.

[3] Z. Li, P. Ishwar, and J. Konrad, "Video condensation by ribbon carving," *IEEE Trans. Image Processing*, vol. 18, pp. 2572-2583, Nov. 2009.