Content-Aware Video Frame Decimation

Yili Pan, Shuo Zeng

Dec 14, 2009
Boston University
Department of Electrical and Computer Engineering
Technical Report No. ECE-2009-06

Content-Aware Video Frame Decimation

Yili Pan, Shuo Zeng

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

Dec 14, 2008

.Technical Report No. ECE-2009-06

Summary

Seam carving is an efficient method for content-aware image resizing. When we apply this method to video, temporal constraints should be taken into consideration in order to avoid artifacts. Our goal in this project is to achieve the target moving consistently after reducing the size of the video. In this report, we firstly introduce the concept of seam carving, and the method to find the optimal seams by using dynamic programming. Then, we describe difficulties appearing when two energy functions are used that lack temporal and seam selection constraints. Subsequently, we present three improved algorithms for content-aware video decimation. The results show good performance in terms of target objects maintaining coherent motion throughout the video sequence. At the end, we briefly discuss another optimization algorithm, called graph cuts, that allows a different implementation of video decimation.

Contents

List of Figures

1 Introduction

Seam carving is an efficient method for content-aware image resizing and could be implemented in many devices, such as computers, cell phone, PDA, etc. In the same way, it can be implemented in video. However, if we simply seam carve every frame and combine them together, without temporal constraints artifacts would be created in the resized video. So in this project, we focus on a global approach with both spatial and temporal constraints in cost function in order to keep the video continuous.

2 Literature review

In previous decades, large amount of image resizing research that focused on down-sampling and up-sampling images accumulated. These classical methods, such as scaling and cropping, apply the same local operator everywhere across the image, without considering the image content. They equally propagating the distortion over the entire image and noticeably squeezing prominent objects. To achieve resizing without distortion, many approaches attempt to remove the unimportant information in the image. Recently proposed retargeting methods try to retain prominent objects while reducing or removing other image content.

Image warping [Gal et al.2006;Wolf et al.2007] offers a continuous solution to image resizing. The warping functions are generally obtained by a global optimization that squeezes or stretches homogeneous regions to minimize the resulting distortion.[Gal et al. 2006] warp an image into various shapes, enforcing the user specified features to undergo similarity transformations. [Wolf et al.2007] automatically determine the importance of each pixel and merge the pixels of lesser importance in the reduction direction.

[Zhang et al.2008] employ shrinkability maps and random walk to accelerate the scaling process and decrease the storage requirements. [Wang et al. 2008] present a "scale-and-stretch" warping method. This method iteratively computes optimal local scaling factors for each local region and updates a warped image that matches these scaling factors. The main problem of this technique is because the distortion is distributed in all spatial directions, some objects may be excessively distorted and the globally spatial structure of the original image is damaged.

All these methods depend on saliency maps. The accuracy of the pre-knowledge will directly affect the quality of the final result.

3 Seam Carving for Resizing

3.1 The operator

Seam Carving is a method that reduce the image resize in a certain direction by moving monotonic and connected 1D seams of pixels that run roughly in the orthogonal direction. To reduce the artifacts, this method search for minimal-cost seams that pass through homogeneous regions by computing their forward or backward energy.

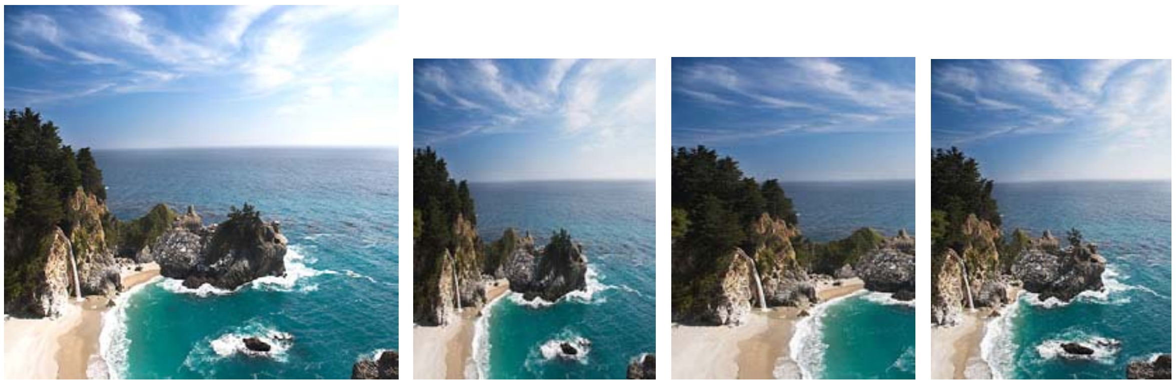
A seam is a connected path of low energy pixels crossing the image from top to bottom, or from left to right. Let I be an $n \times m$ image and its Energy function is $e(i,j) = |\frac{\partial}{\partial x} I(i,j)| + |\frac{\partial}{\partial y} I(i,j)|$. Define a vertical seam to be : $s^x = \{s_i^x\}_{i=1}^n = \{(x(i),i)\}_{i=1}^n$, s.t. $\forall i, |x(i) - x(i-1)| \leq 1$. And a horizontal seam is to be: $s^y = \{s_j^y\}_{j=1}^m = \{(j, y(j)))\}_{j=1}^m$ s.t. $\forall j, |y(j) - y(j-1)| \leq 1$. The cost of a seam is $E(s) = E(I_s) = \sum_{i=1}^n e(I(s_i))$.

We look for the optimal seam S^* that minimizes this seam cost $s^* = \min_s E(s) = \min_s \sum_{i=1}^n e(I(s_i))$. Dynamic programming is an efficient method to search for the minimum cost path. Take vertical seam for example. The first step starts from the second row of the image to the last row, and computes the cumulative minimum energy M for all possible connected seams for each entry (i,j):

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

At the end of this process, the minimum value of the last row in M will indicate the end of the minimal connected vertical seam. In the second step, backtrack from this minimum entry on M to find the path of the optimal seam.

Fig1 comparing the three image resizing methods. From the results, we can see that using seam carving method, the main feature is well kept after resizing. While scaling and cropping cause the image distortion.



(a) original (700*466) (b) scale (173*230) (c) crop (173*230) (d) seam (173*230)

Fig1. Comparing the results of three image resizing methods.

3.2 Forward Energy

Artifacts can be seen from the image removed with several seams. They are created because the original algorithm just focus on removing the seam with lowest gradient value, but ignoring the fact those new edges will be formed in the new resized image. Depending on the direction of the seam, three such cases are possible (Figure 2). In the new energy cost function, we add the energy of the new edges back to the previous energy function, because the optimal seam could choose the path which has the lowest value of both backward and forward energy.

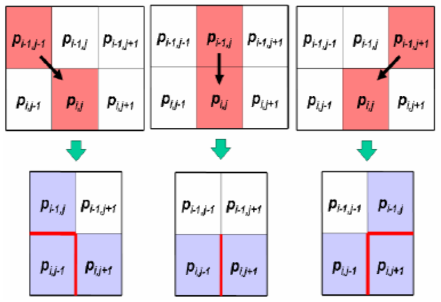


Fig 2. Calculating the three kinds of new edges based on the direction of seam removed

For each of the three possible cases, these are corresponding forward energies:

$$C_L(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j-1)|$$
$$C_U(i,j) = |I(i,j+1) - I(i,j-1)|$$
$$C_R(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)|$$

We use these costs in a new accumulative cost matrix M to calculate the seams using dynamic programming. For vertical seams, each cost is updated using following rule:

$$M(i,j) = P(i,j) + \min \begin{cases} M(i-1, j-1) + C_L(i,j) \\ M(i-1, j) + C_U(i,j) \\ M(i-1, j+1) + C_R(i,j) \end{cases}$$

Where P(i,j) is the gradient energy value in our method

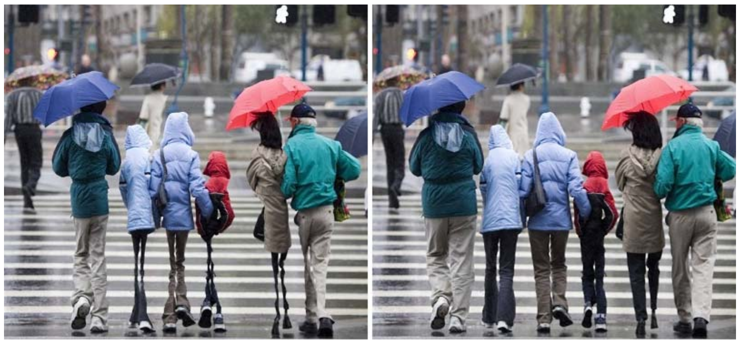
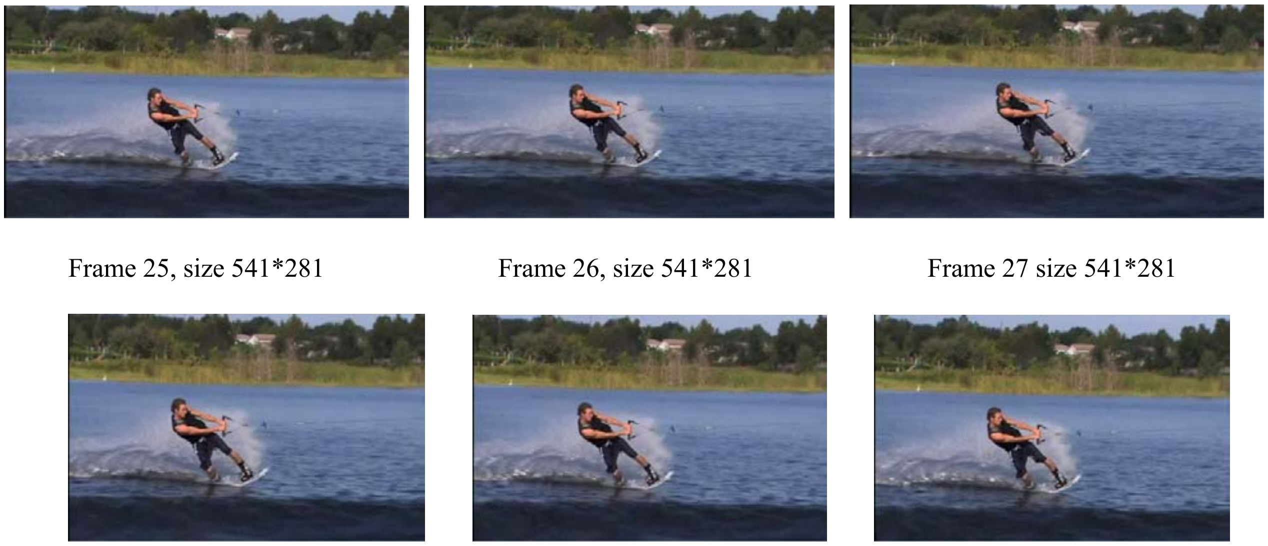


Fig3. comparing backward method (left) and forward method (right)

After adding the forward energy to the cost function, the objects in the image can keep better shapes than the previous method. We can clearly see the improvement in the legs part.

3.3 Seam Carving on Image Sequences

3.3.1 Without Temporal Coherency case



Frame 25, size 541*281 Frame 26, size 541*281 Frame 27 size 541*281

Frame 25, size 400*281 Frame 26, size 400*281 Frame 27, size 400*281

Fig. 4 Results of three frames from waterski.avi video without temporal coherency. The original images are in the first row, and those are in the second row are images after removing 100 vertical seams. Seam carving on each video frame independently creates locally optimal seams that can be totally different over time. This creates a jittery resized video.

For video decimation, if we simply apply the seam carving operator separately to each frame of the video, serious artifacts would be introduced. In Fig2, after removing 100 vertical seams, the waterskiing man is moving forward and backward all the way. This is due to the lack of temporal coherency. So a jittery resized video is created.

3.3.2 Improved Energy Function

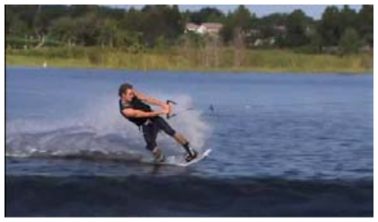
In order to take temporal coherency into consideration, we improve energy function from 2D to 3D. The new energy function is: $e(i,j,t) = |\frac{\partial}{\partial x} I(i,j,t)| + |\frac{\partial}{\partial y} I(i,j,t)| + |\frac{\partial}{\partial t} I(i,j,t)|$

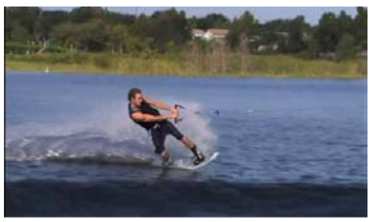
After adding time into the energy function, the seams that are removed are relatively stationary contents in whole video frames, remaining the moving parts. However, the result still has severe artifacts. This is because that there are no constraints on the choice of the seams among each frame. Seam carving on each video frame still independently creates locally optimal seams that can be totally different over time. Fig3 shows the results.

Frame 25, size 541*281

Frame 26, size 541*281

Frame 27 size 541*281


Frame 25,size 486*281


Frame 26, size 486*281

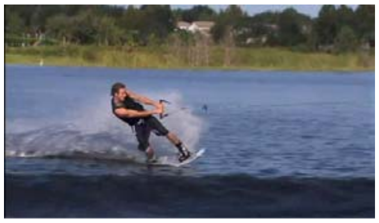

Frame 27 size 486*281

Fig. 5 Results of three frames using temporal constraint. The original images are in the first row, and those are in the second row are images after removing 55 vertical seams. Without constraints on the choices of seams between each frame, the new energy function still creates a jittery resized video.

4 Improved Seam Carving for Video Decimation

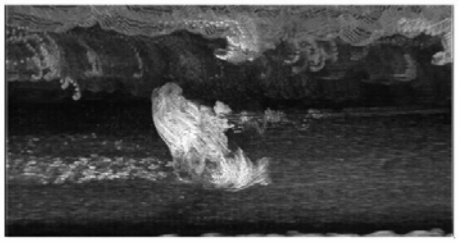
4.1 Static seams

For those videos which have a stationary camera, and the foreground and background are separated, such as Fig2 and Fig3, we could do static seam carving. This method is to search for regions in the image plane that are of low importance in all video frames.

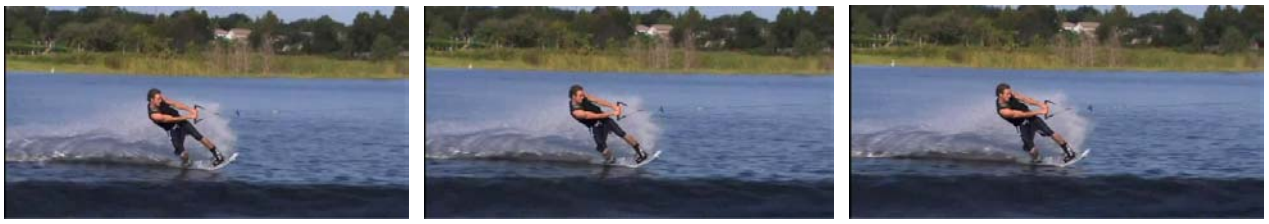
The energy function is as follows:

$$E_{spatial}(i,j) = \max_{t=1}^{N}\{|\frac{\partial}{\partial x} I_t(i,j)| + |\frac{\partial}{\partial y} I_t(i,j)|\}$$
$$E_{temporal}(i,j) = \max_{t=1}^{N}\{|\frac{\partial}{\partial t} I_t(i,j)|\}$$
$$E_{global}(i,j) = \alpha E_{spatial} + (1-\alpha) E_{temporal}$$

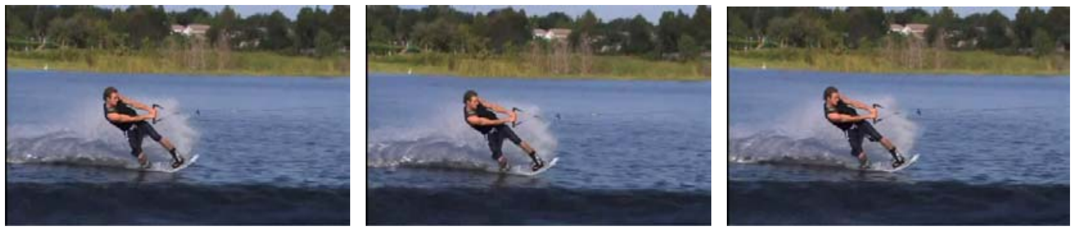
First we compute the energy function on each frame both spatially and temporally, and then taking the maximum energy value at each pixel location. By combining the maximum spatial energy and temporal energy together with different weights, we could obtain a 2D global energy map. We use this global energy to do seam carving on whole video frames. Because the seams do not change along frames, so we call these seams as static seams. Since motion artifacts are more noticeable, we add more weights on temporal energy, taking α=0.3. Fig4 and Fig5 show examples by using static seam carving.



(a) global energy map

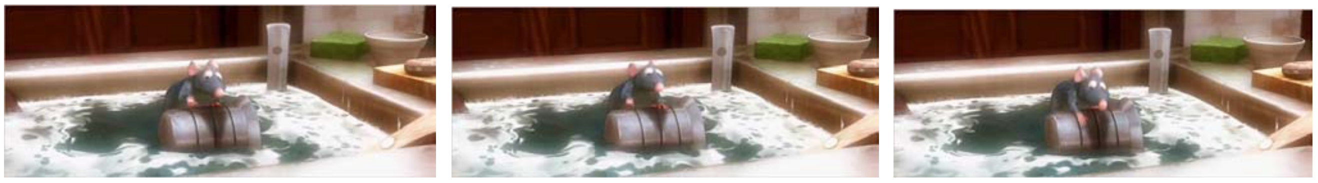


(b) Frame 25, size 541*281 Frame 26, size 541*281 Frame 27 size 541*281

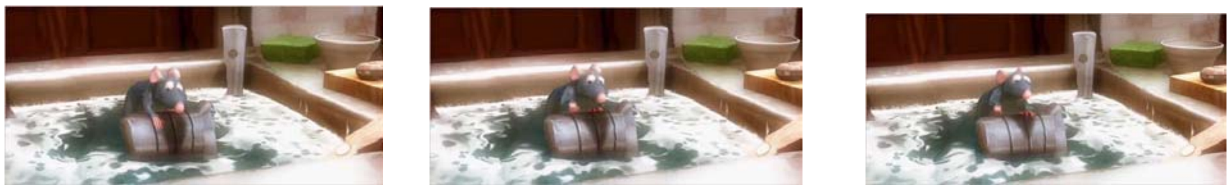


(c) Frame 25, size 400*281 Frame 26, size 400*281 Frame 27, size 400*281

Fig. 6 Results using global energy. (a) is the global energy map, and (b) is the original video, (c) is the decimated video. By using static seam carving, this decimated video shows good results after removing 100 vertical frames. The skiing man is moving continuously along the time, without noticeable artifacts



(a) Frame17, 642*258 Frame18, 642*258 Frame19, 642*258



(b) Frame17, 542*258 Frame18, 542*258 Frame19,542*258

Fig7. Results using global energy from "rat". (a) is the original video, (b) is the decimated video. By using static seam carving, this decimated video shows good results after removing 100 vertical frames. The rat is moving continuously along the time, without noticeable artifacts.

4.2 Cross Correlation Coefficients

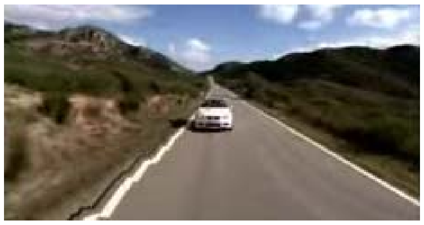
Static seams can improve the video quality by removing global seams with the lowest energy. However, in most videos, both objects and background move along time, which means removing the static seam may distort the video, because the seams do not shift with time. If the shift distance can be calculated, then simply moving the seam with the same distance may be a better way for video decimation.

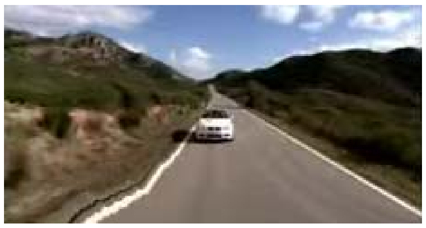
We choose cross correlation coefficients function to calculate the distance:

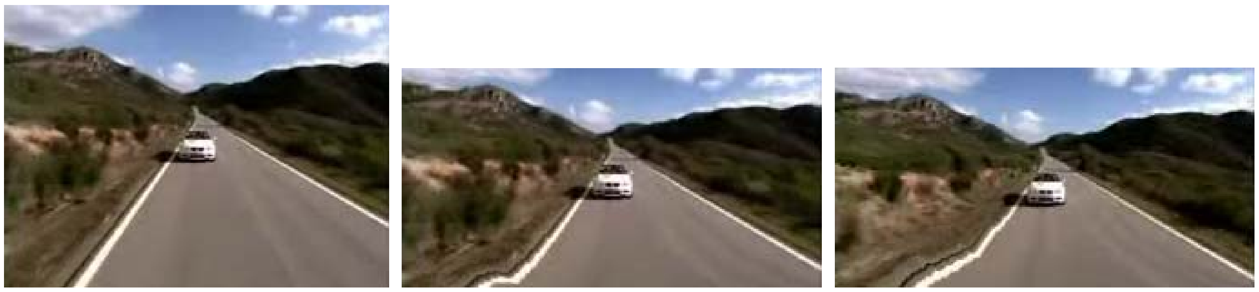
$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

In order to get the distance value, first we shift one of the frame pixel by pixel, then calculate the cross correlation coefficient between the other frame and the shifted one. Then choose the distance with the largest value of r and shift the seam with the same distance. However, the scenery may be completely different after a few frames and the original seam can cause severe distortion. So carving one seam through the whole video has the same problem as what static seam method has. Under the assumption that the objects in video do not change very fast, a new seam can be updated after a certain number of frames, here we choose 15 frames. The distance can be computed for the following 14 frames and seams can be shifted frame by frame.

(a) Frame 20 size 231*170


(b) Frame 20 size 231*120


(c) Frame 20 sizes 231*120



(e) Frame 30 size 231*170 (f) Frame 30 size 231*120 (g) Frame 30 size 231*120

Fig. 8 Comparing cross correlation coefficients with forward energy form “car.avi”. (a)-(c) are original 20th frame and the results from forward energy and cross correlation method respectively. (e)-(g) are the results for 30th frame.

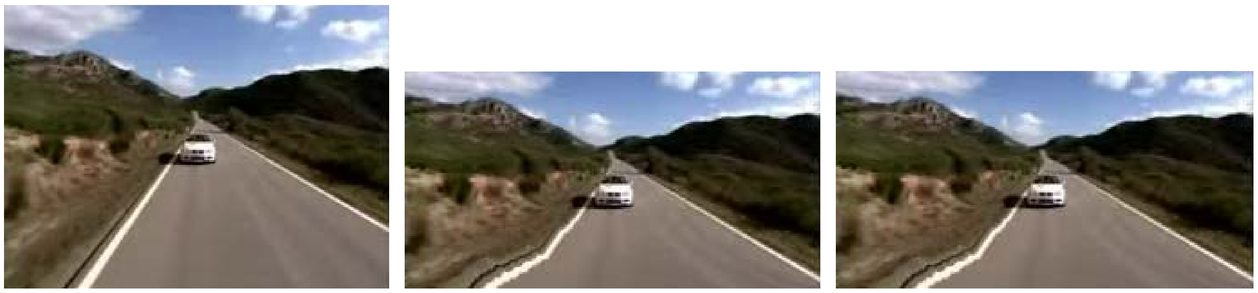
This method does improve the video quality to a certain extent. In the video “car” with pure forward energy seam carving, the car will jump up and down. The cross correlation method shows its advantages compared to the other one: car runs more continuously without obvious artifacts.

4.3 Distance Energy

Because human eyes are not so sensitive to continuous changes, one of the objectives in 3D video resizing is to keep the seam continuous, which means carving the seam close to the seams in both previous and next frame. In order to keep both the energy value and the distance between frames small, a new cost function is needed while searching for the optimal path.

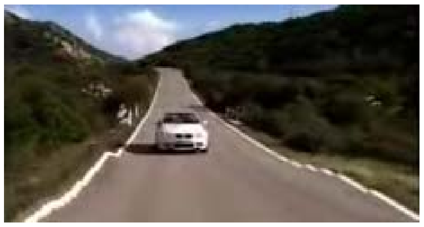
$$E(i,j)=\left|\frac{\partial}{\partial x}I(i,j)\right|+\left|\frac{\partial}{\partial y}I(i,j)\right|+\left|j-j_{seam}\right| \qquad \text{Vertical seam}$$

$$E(i,j)=\left|\frac{\partial}{\partial x}I(i,j)\right|+\left|\frac{\partial}{\partial y}I(i,j)\right|+\left|i-i_{seam}\right| \qquad \text{Horizontal seam}$$



(a) Frame 20 size 231*170 (b) Frame 20 size 231*120 (c) Frame 20 sizes 231* 120

(e) Frame 590 size 231*170


(f) Frame 590 size 231*120

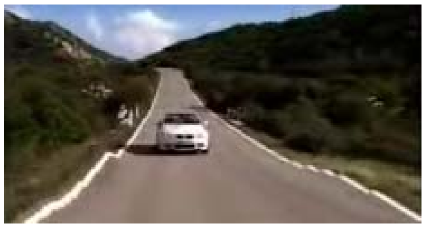

(g) Frame 590 size 231*120

Fig. 9 Comparing results using cross correlation coefficients and distance energy. (a)-(c) are original 20th frame and the results from cross correlation and distance energy method respectively. (e)-(f) are results for the 590th frame.

The distance energy method performs better than the cross correlation method when there are significant changes between frames, because the distance energy method is trying to find the optimal path with lowest value of sum of gradient value and the distance, based on the current frame, so it is still "content -aware".

4.4 Graph cuts

The optimal construction for seams along time is a connected 2D manifold "surface" in space-time that cuts through the video 3D cube. Because the dynamic programming cannot be simply extended to 3D, graph cuts is chosen for finding the optimal surface.

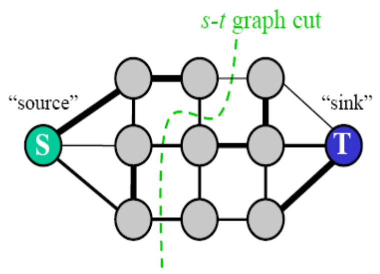


Fig.10 Network flow formulation for graph cuts problem

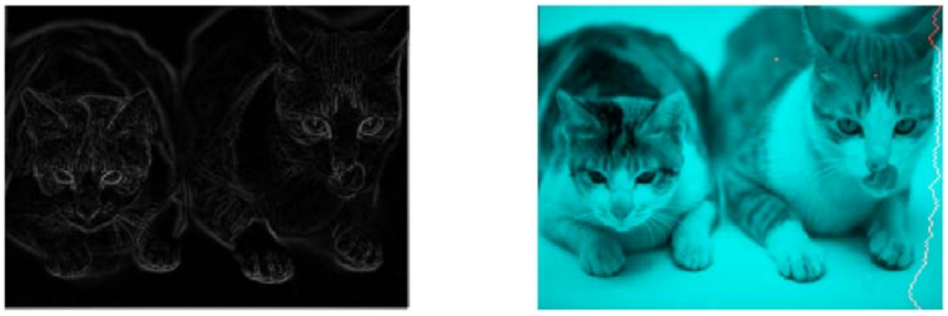
An S/T cut on such a graph is defined as dividing the nodes into two disjoint subsets S and T. The cost of a cut $C = \{S, T\}$ is defined as sum of the cost of the arcs (p, q) where $p \in S$ and $q \in T$. To define a seam from a cut, we consistently choose the pixels to the left of the cut arcs. The optimal seam is defined by the minimum cut which is the cut that has the minimum cost among all valid cuts.

We form this graph cut problem as a linear programming optimization problem and use the linear programming solver in Matlab to get the optimal path. But some difficulties made this problem cannot be simply solved.

(1) Linear Programming Solver in Matlab can only solve small problems. When large image like 300*300 size image was used, Matlab failed to find the optimal solution.

(2) Time consuming. We spent half an hour to find a single path in one frame, while dynamic programming can remove more than one hundred seams with the same size of image.

This means either we have to find a more powerful interface to solve this or program by ourselves to get the results. Without enough time, we can only apply graph cuts in images and need more time to go to the further step.



(a) size 200*150 (b) size 200*150

Fig11. Gradient map (a) and graph cuts result (b)

5. Conclusions

We propose three methods to do video decimation. The static seam carving has good result on the videos whose camera is stationary and foreground and background are separated. If the motion is complex and camera is not stationary with the target, artifacts would occur. Using cross correlation coefficients to get the distance with which the objects are shifted can improve the video quality under a certain circumstances. But when there is a big change between adjacent frames, the resized video will have visible artifacts. Adding distance between seams of adjacent frames to the cost function can solve this problem, the experiment shows that the resized video does not have severe artifacts and objects can move continuously along time.

Graph cuts could achieve impressive result on the videos with complex motions. However, the complexity of this algorithm and the limitation of the linear programming interface make this formulation much more complicated than just using an interface to solve the optimization problem. We will need more time to do our own programming and take a further step in the future.

6. References

[1]AVIDAN,S.,AND SHAMIR,A.2007.Seam carving for content-aware image resizing. ACM Trans.Graph.26,3,10.
[2] RUBINSTEIN,M.,SHAMIR,A.,AND AVIDAN,S. 2008. Improved seam carving for video retargeting. ACM Trans. Graph.27,3.
[3] ZHUANG,,L.,PRAKASH I.,AND JANUSZ k., Video Condensation by Ribbon Carving. IEEE,2009
[4] BOYKOV, Y.,AND KOLMOGOROV, V., 2004. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Computer Vision. In IEEE Transactions on Pattern Analysis and Machine Intelligence, September 2004.

7. Appendix

```
function [imgNew,energyReduction,imgL]=oneSeamForward(img,EnergyMap,flag,imgLabel)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%if flag==0, vertical seam; if flag==1, horizontal seam
%%img can either be color image or grayscale image
%%imgLabel can only be color image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
imgG=zeros(size(img,1),size(img,2));
if(length(size(img))==3)
   imgG=double(rgb2gray(uint8(img)));
else
   imgG=double(img);
end

if(flag==1)
   EnergyMap=EnergyMap';
   imgG=imgG';
end
[h,w]=size(EnergyMap);
infi=Inf('double');
imgG=[ones(size(imgG,1),1)*infi,imgG,ones(size(imgG,1),1)*infi];
%% dynamic programing
mask=EnergyMap*255*6;
q=zeros(h,w+2);
q(:,1)=Inf('double');
q(:,end)=Inf('double');
q(1,2:end-1)=mask(1,:);
p=zeros(size(mask));
for i=2:size(EnergyMap,1)
   for j=2:(size(q,2)-1)

eng=[q(i-1,j-1)+abs(imgG(i-1,j)-imgG(i,j-1))+abs(imgG(i,j+1)-imgG(i,j-1)),q(i-1,j)+
abs(imgG(i,j+1)-imgG(i,j-1)),q(i-1,j+1)+abs(imgG(i-1,j)-imgG(i,j+1))+abs(imgG(i,j+1
)-imgG(i,j-1))];
      q(i,j)=mask(i,j-1)+min(eng);
      p(i-1,j-1)=find(eng==min(eng),1)-2;
   end
end

q1=q(:,2:end-1);
pix=zeros(size(q1,1),1);
lastRow=q1(end,:);
pix(end)=find(lastRow==min(lastRow),1);
for i=1:size(q1,1)-1
   j=size(q1,1)-i;
   pix(j)=pix(j+1)+p(j,pix(j+1));
end

energyReduction=0;
for i=1:size(pix)
   energyReduction=energyReduction+EnergyMap(i,pix(i));
end
```

```
%% cancel the seam
%color image
if (length(size(img))==3)
    if(flag==0)
        imgNew=zeros(size(img,1),size(img,2)-1,size(img,3));
        for i=1:size(img,1)
            imgNew(i,1:(pix(i)-1),:)=img(i,1:(pix(i)-1),:);
            imgNew(i,pix(i):end,:)=img(i,(pix(i)+1):end,:);
        end
    end
    if(flag==1)
        imgNew=zeros(size(img,1)-1,size(img,2),size(img,3));
        for i=1:size(img,2)
            imgNew(1:(pix(i)-1),i,:)=img(1:(pix(i)-1),i,:);
            imgNew(pix(i):end,i,:)=img((pix(i)+1):end,i,:);
        end
    end
end
%gray image
if(length(size(img))==2)
    if(flag==0)
        imgNew=zeros(size(img,1),size(img,2)-1);
        for i=1:size(img,1)
            imgNew(i,1:(pix(i)-1))=img(i,1:(pix(i)-1));
            imgNew(i,pix(i):end)=img(i,(pix(i)+1):end);
        end
    end
    if(flag==1)
        imgNew=zeros(size(img,1)-1,size(img,2));
        for i=1:size(img,2)
            imgNew(1:(pix(i)-1),i)=img(1:(pix(i)-1),i);
            imgNew(pix(i):end,i)=img((pix(i)+1):end,i);
        end
    end
end

%% resize the label map
if(nargin==4)
    if(flag==0)
        imgL=zeros(size(imgLabel,1),size(imgLabel,2)-1,size(imgLabel,3));
        for i=1:size(imgLabel,1)
            imgL(i,1:(pix(i)-1),:)=imgLabel(i,1:(pix(i)-1),:);
            imgL(i,pix(i):end,:)=imgLabel(i,(pix(i)+1):end,:);
        end
    end
    if(flag==1)
        imgL=zeros(size(imgLabel,1)-1,size(imgLabel,2),size(imgLabel,3));
        for i=1:size(imgLabel,2)
            imgL(1:(pix(i)-1),i,:)=imgLabel(1:(pix(i)-1),i,:);
            imgL(pix(i):end,i,:)=imgLabel((pix(i)+1):end,i,:);
        end
    end
end
```

```
function EngMask=EngGlobal(imgCube)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
static seam carving energy function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[frNum,height,width]=size(imgCube);
h=size(imgCube,2);
w=size(imgCube,3);
EngMask=zeros(h,w);



%%  intensity difference
imgNew=zeros(frNum+2,height+2,width+2);
imgNew(2:(end-1),2:(end-1),2:(end-1))=imgCube;
imgNew(1,:,:)=imgNew(3,:,:);imgNew(end,:,:)=imgNew((end-2),:,:);
imgNew(:,1,:)=imgNew(:,3,:);imgNew(:,end,:)=imgNew(:,(end-2),:);
imgNew(:,:,1)=imgNew(:,:,3);imgNew(:,:,end)=imgNew(:,:,(end-2));

imgw=abs(imgNew(2:(end-1),2:(end-1),1:(end-2))-imgCube)+abs(imgNew(2:(end-1),2:(end
-1),3:end)-imgCube);
imgh=abs(imgNew(2:(end-1),1:(end-2),2:(end-1))-imgCube)+abs(imgNew(2:(end-1),3:end,
2:(end-1))-imgCube);
imgt=abs(imgNew(1:(end-2),2:(end-1),2:(end-1))-imgCube)+abs(imgNew(3:end,2:(end-1),
2:(end-1))-imgCube);

Espatial=imgw+imgh;
Etemporal=imgt;

 alpha=0.3;

for i=1:h;
    for j=1:w;
       max_spatial=max(Espatial(:,i,j));
       max_temporal=max(Etemporal(:,i,j));
       EngMask(i,j)=alpha*max_spatial+(1-alpha)*max_temporal;
    end
end
imshow(uint8(EngMask));




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
graph cuts, 2D one seam
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
name='C:\Users\Shuo\Desktop\seamcarving images\waterfall.png';
img=double(imresize(rgb2gray(imread(name)),[100,100]));
sidelink=10000;
infi=10000;

[m,n]=size(img);
A=sparse(m*n+2,m*n+2);
Ix=abs(filter2([1 -1],img,'same'));
Iy=abs(filter2([1 -1]',img,'same'));
I=Ix+Iy+rand(size(Ix))+0.5;
```

```
for i=1:m
   for j=1:n
      node=zeros(m,n);
      st=[0;0];
%% Backforward method

      if(j~=1)
         node(i,j-1)=infi;
      else
         st=[0;0];
      end
      if(j~=n)
         node(i,j+1)=I(i,j);
      else
         st=[0;sidelink];
      end
      if(i~=1 && j~=1)
         node(i-1,j-1)=infi;
      end

      if(i~=m && j~=1)
         node(i+1,j-1)=infi;
      end

      node=node';
      node=[node(:);st];
      A(((i-1)*n+j),:)=node';
   end

   i
end
%% Add node s and t

nodes=zeros(m,n);
nodet=zeros(m,n);
nodes(1:m,1)=sidelink;
nodet(1:m,n)=0;
nodes=nodes';
nodet=nodet';
nodes=[nodes(:);0;0];
nodet=[nodet(:);0;0];
A(end-1,:)=nodes;
A(end,:)=nodet;

%% construct W maxtrix

W=sparse([]);
u=[];
arc=[];
for i=1:m*n+2
   node=A(i,:);
   index=find(node~=0);
   for j=1:length(index)
      W=[W,zeros(m*n+2,1)];
      W(i,end)=1;
      W(index(j),end)=-1;
      u=[u,node(index(j))];
      arc(i,j)=index(j);
   end
end

W1=W(1:m*n,:);
Ws=W(m*n+1,:);
Wt=W(end,:);

arc1=arc>0;
n1=size(W1,2);
%% optimize
cvx_begin
  variables f(n1) bs bt
  dual variables y z p
  maximize(bs)
  subject to
    y : W1 * f == 0;
        Ws * f - bs == 0;% -bs
        Wt * f - bt == 0; % -bt
        bs+bt ==0;
    z : f >= 0;
    p : f<=u';
cvx_end

%% find the seam
% one point for each row

pCopy=p;
afterGC=img;
afterGC=afterGC(:);
for i=1:m
   arc3=arc(((i-1)*n+1):(i*n),:)';
   index3=find(arc3>0);
   pTemp=pCopy(1:length(index3));
   [pMax,ind]=max(pTemp);
   nodeNum1=ceil(index3(ind)/size(arc,2))+(i-1)*n;
   sub=ind2sub(size(arc3),index3(ind));
   nodeNum2=arc3(sub);
   nodeNum=[nodeNum1,nodeNum2];
   afterGC(nodeNum(find(nodeNum==min(nodeNum))))=-1000;
   pCopy(1:length(index3))=[];
end

afterGC=reshape(afterGC,n,m);
afterGC=afterGC'<=0;
img3(:,:,1)=255*afterGC;
img3(:,:,2)=img;
img3(:,:,3)=img;

figure;
subplot(1,2,1);imshow((Ix+Iy),[]);
subplot(1,2,2);imshow(uint8(img3),[]);
```