Ryan Ebanks
CS 376
Problem Set #1

I.) Short Answer Problems

1.) The Associative property of convolusion states (f*g)*h = f*(g*h). This can be used to more efficiently filter an image by applying convolusion to the two smaller matrices first, and the using that result to apply to the larger matrix. This will reduce the number of computations necessary.

2.) Result of dilation is [0 1 2 2 1 1 2 2]

3.)

4.) One could reduce the amount of fine edges detected by the setting high thresholds and also one could use a large area filter to smooth the picture.

5.)

6.) For these methods to work I am assuming that the camera remains at a stationary above the conveyor belt so it will always have the same scale. Also the objects on the conveyor belt are of same type of object, will always be oriented the same way on the conveyor belt, will always be at the same distance away from the edge of the conveyor belt, and will be equally spaced on the belt. The conveyor belt itself will be a solid color that greatly contrasts the color of the part. The first step would be to get a still image (a single frame) of the video when the part is passing directly under the camera. With this image I would apply a Guasian filter to smooth the edges. The we could run a canny edge detector and see if matched the outline that we were expecting. If it did not we could alert a quality control person to remove the part from the line.

II.) MATLAB CODE

```
function xgrad = xgradient(img)

img = im2double(img);
img_gray = rgb2gray(img);
h1 = fspecial('gaussian',5,5);
smooth = imfilter(img_gray, h1);
h2 = [-1,1];
xgrad = imfilter(smooth, h2);


function ygrad = ygradient(img)

img = im2double(img);
img_gray = rgb2gray(img);
h1 = fspecial('gaussian',5,5);
smooth = imfilter(img_gray, h1);
h2 = [-1;1];
ygrad = imfilter(smooth, h2);


function energy = calcEnergy(img)

xgrad = xgradient(img);
ygrad = ygradient(img);
energy = sqrt(xgrad.^2 + ygrad.^2);
```

```
function seams = vertEnergyMap(img)

en_mat = calcEnergy(img);
dim = size(en_mat);
for i=1:1:dim(1)
            for j=1:1:dim(2)
                        if i == 1
                          seams(i,j) = en_mat(i,j);
                elseif j - 1 < 1
                          seams(i,j) = en_mat(i,j) + min(seams(i-1,j), seams(i-1, j+1));
                elseif j+1 > dim(2)
                          seams(i,j) = en_mat(i,j) + min(seams(i-1,j), seams(i-1, j-1));
                else
                          seams(i,j) = en_mat(i,j) + min(seams(i-1,j+1), min(seams(i-1,j), seams(i-1, j-1)));
                end
       end
end


function seams = horzEnergyMap(img)

en_mat = calcEnergy(img);
dim = size(en_mat);
for j=1:1:dim(2)
            for i=1:1:dim(1)
                        if j == 1
                          seams(i,j) = en_mat(i,j);
                elseif i - 1 < 1
                          seams(i,j) = en_mat(i,j) + min(seams(i,j-1), seams(i+1, j-1));
                elseif i+1 > dim(1)
                          seams(i,j) = en_mat(i,j) + min(seams(i,j-1), seams(i-1, j-1));
                else
                          seams(i,j) = en_mat(i,j) + min(seams(i+1,j-1), min(seams(i,j-1), seams(i-1, j-1)));
                end
       end
end


function optimal = optVertSeam(img)

seams = vertEnergyMap(img);
dim = size(seams);
mina = 99999;
for c=1:1:dim(2)
            if seams(dim(1),c) < mina
              col = c;
          mina = seams(dim(1), c);
        end
end
optimal = zeros(dim(1), 2);
optimal(dim(1), 2) = col;
optimal(dim(1), 1) = dim(1);
```

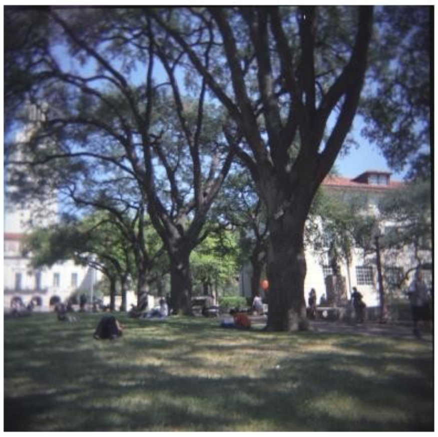
```
            for r=dim(1)-1:-1:1
                        optimal(r,1) = r;
                        if col - 1 < 1
                            if seams(r, col) < seams(r, col+1)
                      optimal(r, 2) = col;
                            else
                              optimal(r, 2) = col+1;
                      col = col+1;
                            end
                   elseif col + 1 > dim(2)
                            if seams(r, col) < seams(r, col -1)
                              optimal(r,2) = col;
                    else
                              optimal(r,2) = col -1;
                      col = col - 1;
                    end
                   else
                         if seams(r, col) < seams(r, col -1) && seams(r, col) < seams(r, col+1)
                           optimal(r, 2) = col;
                 elseif seams(r, col-1) < seams(r, col) && seams(r, col-1) < seams(r, col+1)
                           optimal(r, 2) = col - 1;
                   col = col - 1;
                         else
                           optimal(r, 2) = col+1;
                   col = col +1;
                  end
               end
           end


function optimal = optHorzSeam(img)

seams = horzEnergyMap(img);
dim = size(seams);
mina = 99999;
for r=1:1:dim(1)
            if seams(r,dim(2)) < mina
              row = r;
          mina = seams(r, dim(2));
        end
end
            optimal = zeros(2, dim(2));
optimal(1, dim(2)) = row;
optimal(2, dim(2)) = dim(2);
            for c=dim(2)-1:-1:1
                        optimal(2,c) = c;
                        if row - 1 < 1
                            if seams(row, c) < seams(row+1, c)
                      optimal(1, c) = row;
                            else
                              optimal(1, c) = row+1;
                      row = row+1;
```

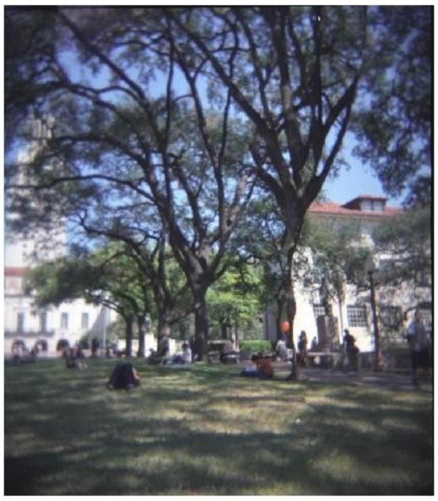
```
                end
            elseif row + 1 > dim(1)
                if seams(row, c) < seams(row-1, c)
                    optimal(1,c) = row;
             else
                    optimal(1,c) = row -1;
              row = row - 1;
             end
            else
                if seams(row, c) < seams(row-1, c) && seams(row, c) < seams(row+1, c)
                    optimal(1, c) = row;
           elseif seams(row-1, c) < seams(row, c) && seams(row-1, c) < seams(row+1, c)
                    optimal(1, c) = row - 1;
            row = row - 1;
                else
                    optimal(1, c) = row+1;
            row = row +1;
           end
         end
      end


function imgwseam = displaySeam(img, HorV)

if HorV == 'v'
  vseam = optVertSeam(img);
  imshow(img);
  hold on;
  plot(vseam(:,2),vseam(:,1))
elseif HorV == 'h'
  hseam = optHorzSeam(img);
  imshow(img);
  hold on;
  plot(hseam(2,:)' , hseam(1,:)')
else
  vseam = optVertSeam(img);
  hseam = optHorzSeam(img);
  imshow(img);
  hold on;
  plot(vseam(:,2),vseam(:,1));
  hold on;
  plot(hseam(2,:)' , hseam(1,:)')
end


function redu = reduceWidth(img, amount)
temp = img;
for seams_rm=1:1:amount
            rm = optVertSeam(temp);
        rm = rm(:,2);
        dim = size(temp);
        reduced = zeros(dim(1), (dim(2)-1), dim(3));
```

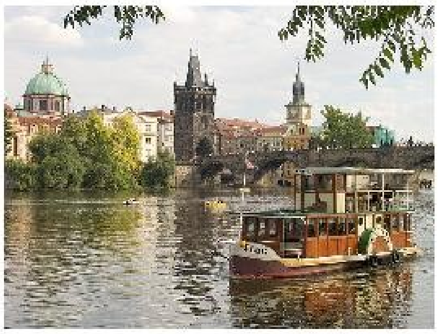
```
               ed = dim(2);
               for r=1:1:dim(1)
                               if rm(r) < dim(2) && rm(r) > 1
                                 reduced(r,:,1) = [temp(r,1:(rm(r)-1),1) temp(r,(rm(r)+1):ed,1)];
                         reduced(r,:,2) = [temp(r,1:(rm(r)-1),2) temp(r,(rm(r)+1):ed,2)];
                         reduced(r,:,3) = [temp(r,1:(rm(r)-1),3) temp(r,(rm(r)+1):ed,3)];
                       elseif rm(r) == 1
                         reduced(r,:,1) = temp(r,2:ed,1);
                         reduced(r,:,2) = temp(r,2:ed,2);
                         reduced(r,:,3) = temp(r,2:ed,3);
                               else
                                     reduced(r,:,1) = temp(r,1:(ed-1),1);
                         reduced(r,:,2) = temp(r,1:(ed-1),2);
                         reduced(r,:,3) = temp(r,1:(ed-1),3);
                       end
                   end
                   temp = uint8(reduced);
end
redu = temp;


function redu = reduceWidth(img, amount)
temp = img;
for seams_rm=1:1:amount
                   rm = optVertSeam(temp);
               rm = rm(:,2);
               dim = size(temp);
               reduced = zeros(dim(1), (dim(2)-1), dim(3));
               ed = dim(2);
               for r=1:1:dim(1)
                               if rm(r) < dim(2) && rm(r) > 1
                                 reduced(r,:,1) = [temp(r,1:(rm(r)-1),1) temp(r,(rm(r)+1):ed,1)];
                         reduced(r,:,2) = [temp(r,1:(rm(r)-1),2) temp(r,(rm(r)+1):ed,2)];
                         reduced(r,:,3) = [temp(r,1:(rm(r)-1),3) temp(r,(rm(r)+1):ed,3)];
                       elseif rm(r) == 1
                         reduced(r,:,1) = temp(r,2:ed,1);
                         reduced(r,:,2) = temp(r,2:ed,2);
                         reduced(r,:,3) = temp(r,2:ed,3);
                               else
                                     reduced(r,:,1) = temp(r,1:(ed-1),1);
                         reduced(r,:,2) = temp(r,1:(ed-1),2);
                         reduced(r,:,3) = temp(r,1:(ed-1),3);
                       end
                   end
                   temp = uint8(reduced);
end
redu = temp;
```

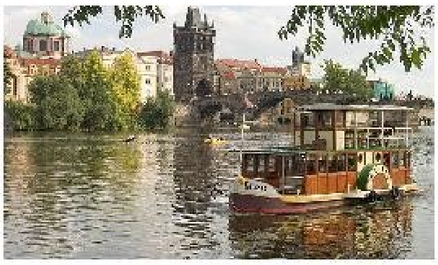
RESULTS****************************

1.)

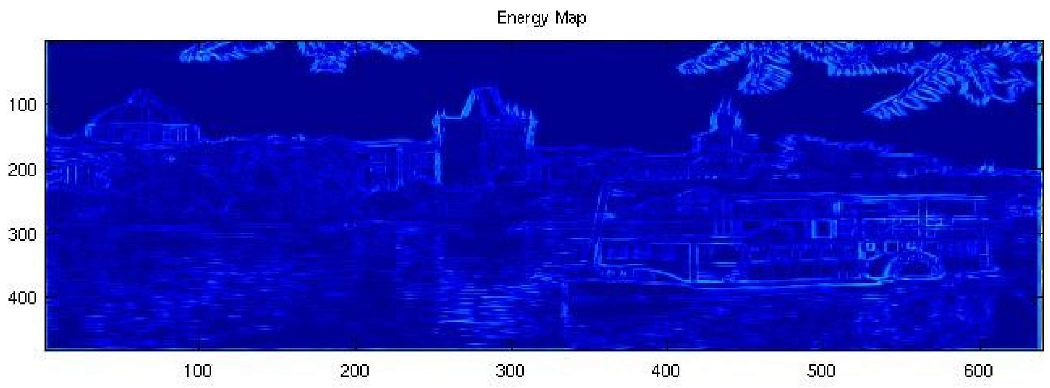

Orginal

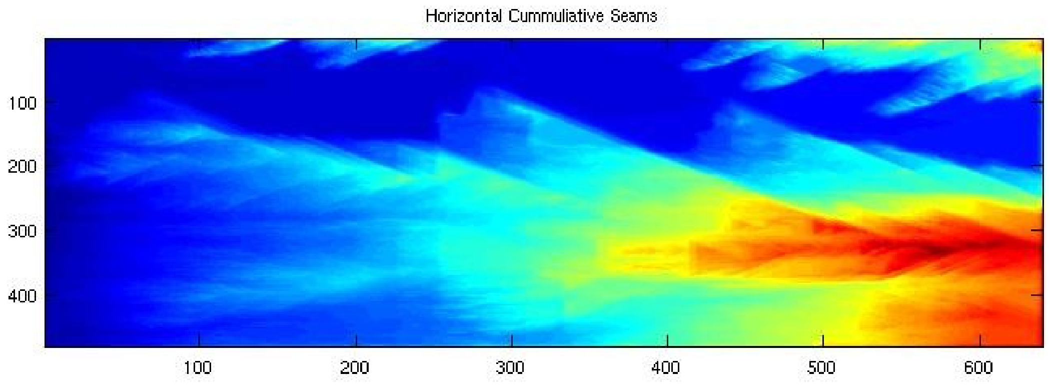

reduceWidth() by 100

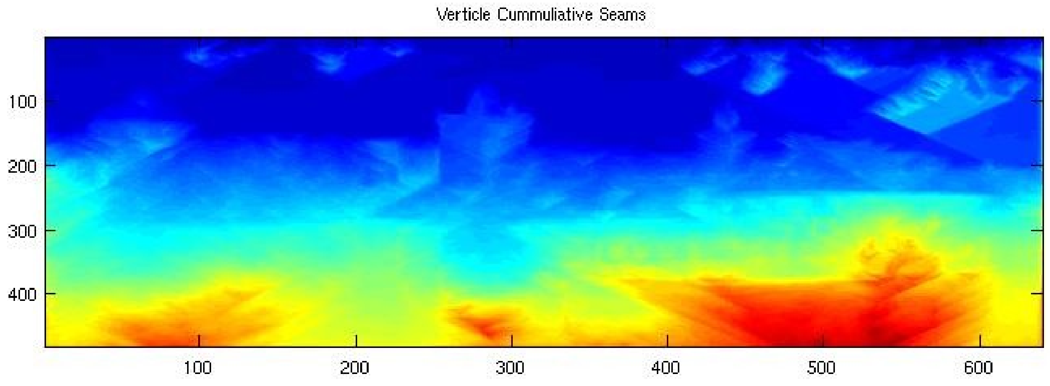

Original


reduceHeight() by 100

2.)



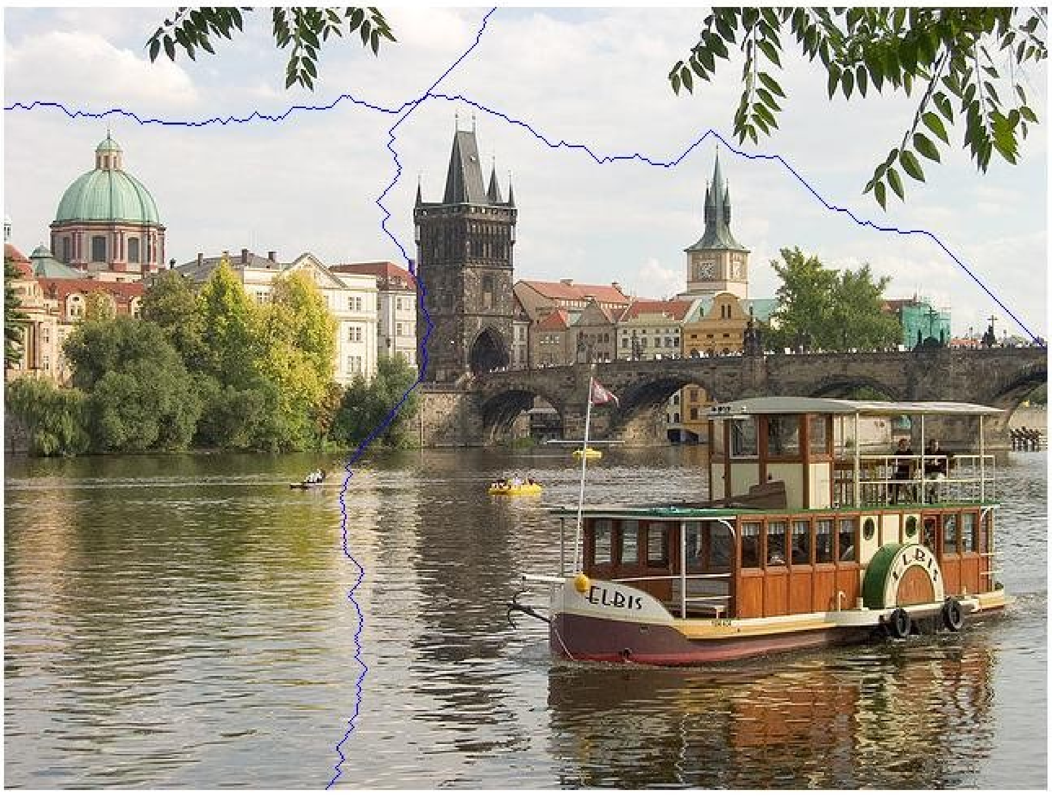




The energy map is going to high where ever there is a edge. When you look at the energy map you can basically see the outline of the prague.jpg which is what we would expect. The Cumulative Seams graph also shows what we expect. The values on the right side of the image are going to be hight as the function sums to the right. Also there are low values in the area where the sky is located in the photo because there are a small amount of edges there. You can also notice the values increase around where the seams have to pass over the buildings and are extremely high where the seams have to pass over the boat.
For the Vertical Cumulative seams, it is also as we expect. The values continue rise as you go farther down the graph. The highest range is at the bottom right where the seams have to pass over

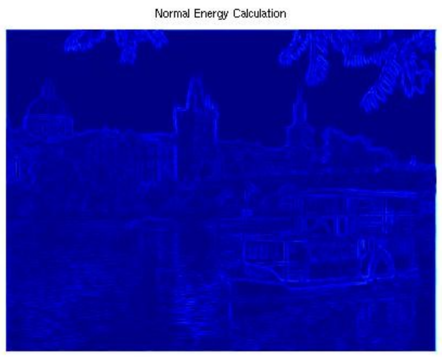
the boat. The rest of the graph has a pretty even vertical distribution of intensity as all vertical seams pass throught the same three layers of sky, buildings, and water.

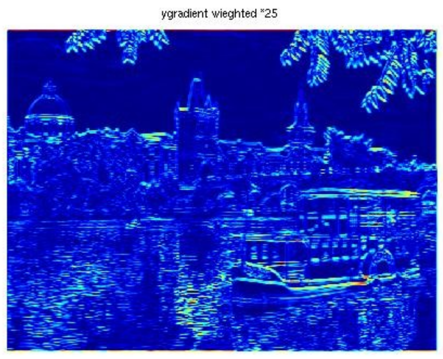
3.)



This picture displays the optimal vertical and horizontal seam for prague.jpg. If you follow the vertical seam, from the top to the bottom, you will notice it tries to stay in blue area of the sky where there are very little edges to cross. As it enters the building level of the photo, it follows along the dark building that has consistent face with no window. And finally, as turns to be able to travel through the water at place where there are not many ripples, again to avoid ass many edges as possible. The Horizontal seam follows the path across the sky where it does not have to cross edges which is the exact behavior we would expect.

4.)





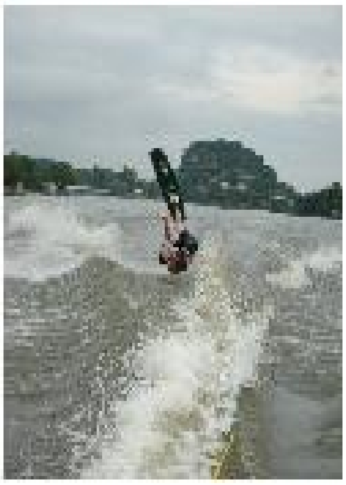
```
function energy = altCalcEnergy(img)

xgrad = xgradient(img);
ygrad = ygradient(img);
energy = sqrt(xgrad.^2 + 25*ygrad.^2);
```

So here, I weight the y-gradient 25 times more than the x-gradient. As you would expect it returns overall high energy per pixel, and you will notice that the mostly horizontal seams have a lot hight energy than the mostly vertical seams. This would cause the the the seam select function to prefer to cross the mostly vertical seams than the mostly horizontal.

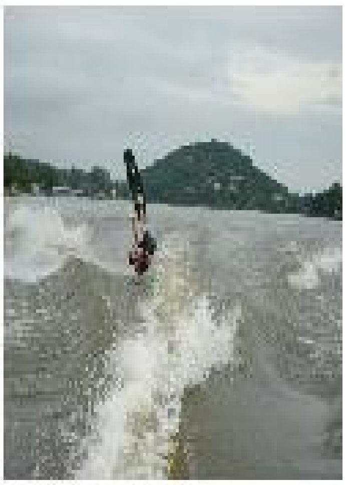
5.)

Original

redueWidth() by 200

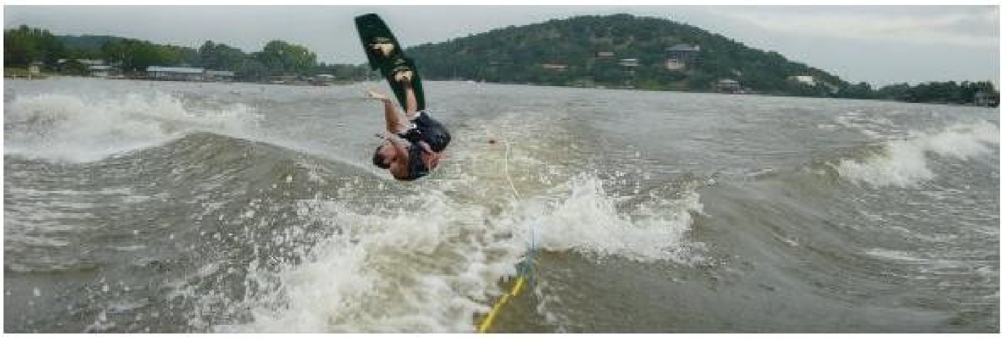


Resized to 453 x 404

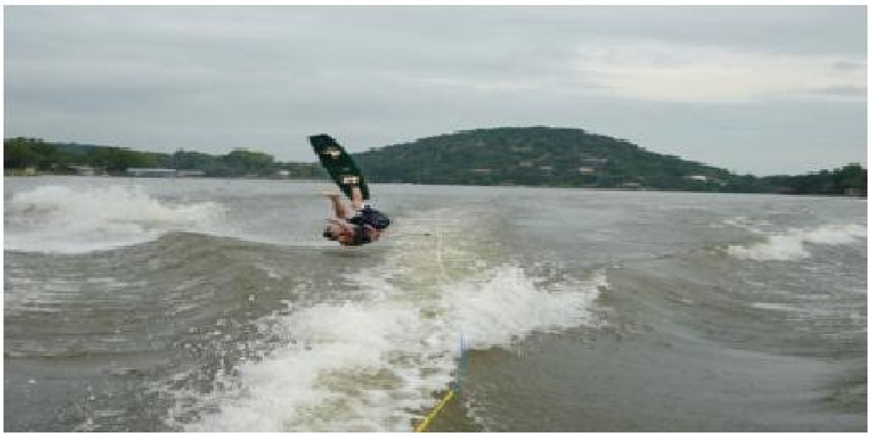


The original photo is 453x604, and is a picture of me falling while wakeboarding. The pictures are slightly off due to importing them from matlab. In the picture that as produced from seam carving you can see that seams cut through the water where it was splashing up and creating 'white water'.

reduceHeight() by 253

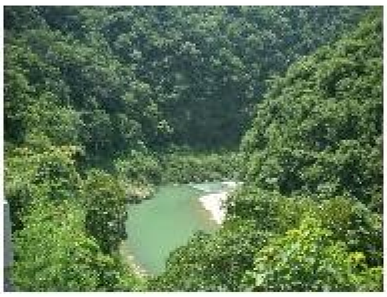


matlab resize 200x404

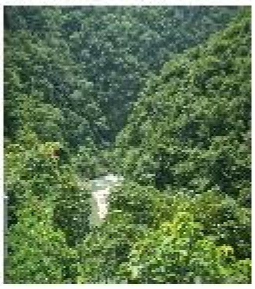


This is the same original as the above alterations but this time we reduced the height. As you can see it removed the sky because the lack of distinct edges up there, but then also removed a portion of the wake board since it is the same color as the hills. And again some of the non-'white water' was removed.

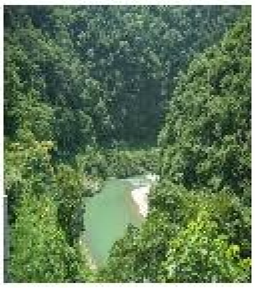
Orignial



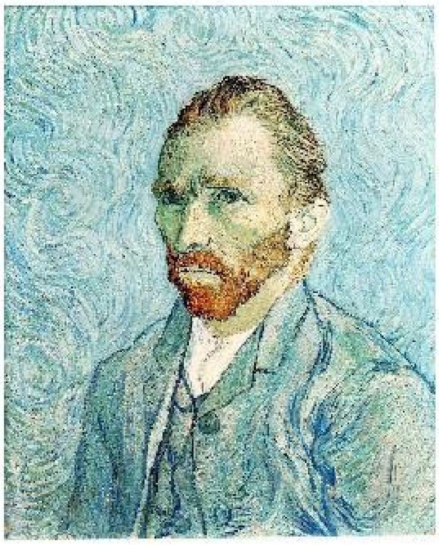
reduceWidth() by 200



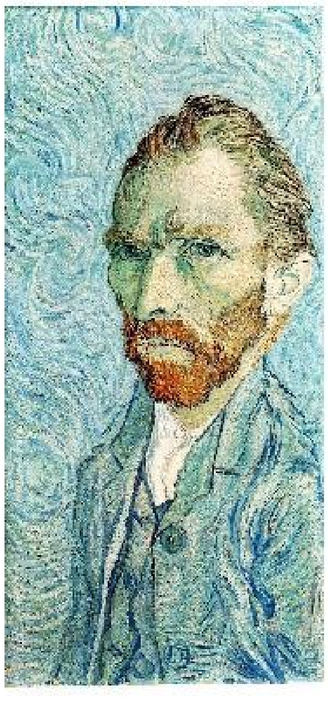
Matlab resize to 454x404



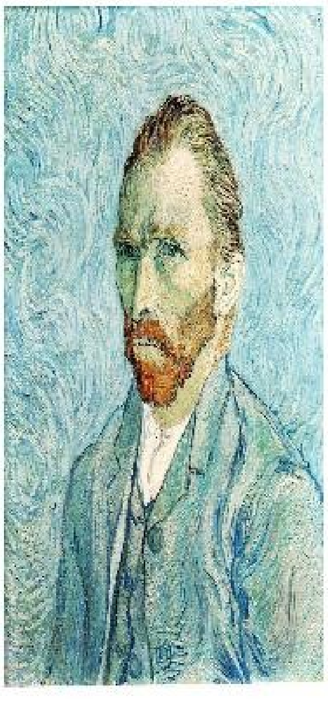
This a picture from a vacation I took from when I went to China, size 454x604. When seam carved the river in the middle almost gets completely cut out. The river contains almost no edges, so it was the first thing to get removed.



reduceWidth() by 350



Matlab resize 1026x477



This is an image of a self portrait by Van Goh, orginal size 1026x1127. I picked this picuter because it basically has edges every where and wanted to see what the result would be. You can see it avoid carving most of the face. This is probably because the most pronounced edges are on the face to due to the red hair color and beard. The rest of the photo is a varying shade of blue os the edges would be less pronounced.