Computer Vision Problem set 1
Eunho Yang (cs id : eunho)

II. Programming

5.

I have tried several combinations of vertical reductions and horizontal reductions in this problem and in the problem of III.4. However, I found out that we couldn't expect entirely different results according to the selecting order. Even the results from the optimal sequences (extra credit problem in III.4) also make almost similar results with those from the hard-coding sequences.

(a) Original input

(b) Content-aware resizing

(c) Image from 'imresize'

(d) Input and output image dimensions
Dimensions of the Original image are 375x500. And dimensions of reduced images are 175x300.
(e) Remove 200 vertical seams first and then select 200 horizontal seams
(f) The region with lower changes such as sand and sky are removed in (b). Instead, (b) contains the similar size of objects(men, sea lions, rocks) with the original image.

(a) Original input

(b) Content-aware resizing

(c) Image from ‘imresize’

(d) Input and output image dimensions
Dimensions of the Original image are 375x500. And dimensions of reduced images are 225x500.
(e) Remove 150 horizontal seams.
(f) Since there is almost no background part in the ‘tree.jpg’, the content-aware resizing looks bad. Tree in the center changed into three different trees. When applying the vertical carving (reduceWidth 100 pixels), the tree at the right is removed first shown as following figure

(a) Original input

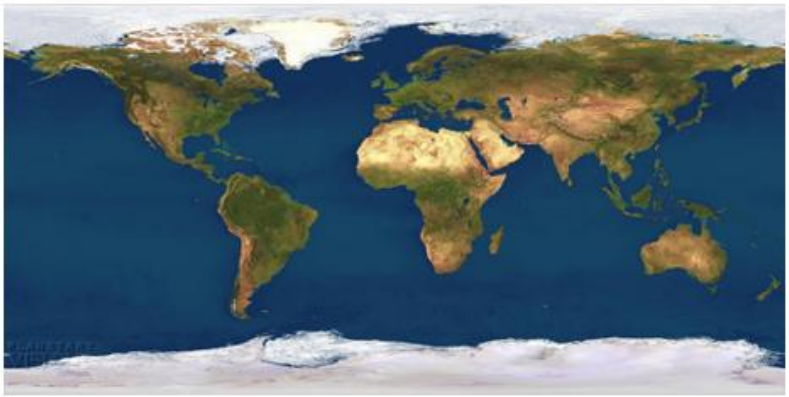
(b) Content-aware resizing

(c) Image from ‘imresize’

(d) Input and output image dimensions
Dimensions of the Original image are 375x500. And dimensions of reduced images are 275x350.
(e) Remove 150 vertical seams first and then select 100 horizontal seams

(f) In this picture, content-aware resizing doesn't work well, either. It looks like that there are a lot of objects. Moreover, it does not consider the distances and directions of the objects. As a result, it transformed the line into the curve and make distorted output.

(a) Original input

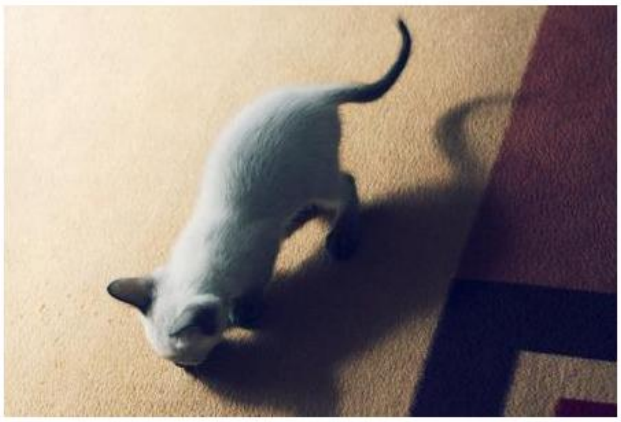


(b) Content-aware resizing

(c) Image from 'imresize'

(d) Input and output image dimensions
Dimensions of the Original image are 231x461. And dimensions of reduced images are 181x211.
(e) Remove 250 vertical seams first and then select 50 horizontal seams
(f) Input image is the earth image. There is no variation in the sea. As a result, the sea becomes the background and lower energy. Content-aware resizing removes the sea and put all continents together like 'Pangaea'.

(a) Original input



(b) Content-aware resizing

(c) Image from ‘imresize’

(d) Input and output image dimensions
Dimensions of the Original image are 300x448. And dimensions of reduced images are 300x298.
(e) Remove 150 vertical seams.
(f) In this image, there are only few objects (a cat and shadow). However, vertical seams pass through the cat and remove some part of the cat. This is because white region of the cat has no variation and is similar with the background. In this case, the image from the ‘imresize’ looks better.

(a) Original input

(b) Content-aware resizing

(c) Image from ‘imresize’

(d) Input and output image dimensions
Dimensions of the Original image are 375x500. And dimensions of reduced images are 285x300.
(e) Remove 200 vertical seams first and then select 90 horizontal seams
(f) In this example, content-aware doesn’t work well since the background has a lot of variations. On the other hand, the bridge is monotonous black. As a result, content-aware resizing has a effect like removing the object.

III. Extra credit

1.

I implemented in two ways to remove selected object. The first way (removeObject.m) is that the seams are selected (but not removed if it doesn’t contain selected pixel) without the modification of original ‘Energy’ value. The second way (removeObject2.m) is that new energy value for the selected pixels is assigned. To be sure that the selected pixel should be selected as a seam if exists, I choose the new assigned value as -255*(# of column).
I used the Matlab version of **‘R2009a’** (with ‘createMask’ function).

```
>> im = imread('surf.jpg');
>> remove1 = removeObject(im);
>> remove2 = removeObject2(im);
```

Figure A represents the selected area. Figure B and C shows the results from the method 1 and 2, respectively

Figure A

Figure B

Figure C

The method 1 tends to remove other objects near (similar column or row) the selected object.

2.

I modify the energy function to consider the location information. Of course, it is not important in all cases. However, in the case of the pictures where the main objects are located in the center of the pictures, the location information (how far from the center) is also good estimator to distinguish the main objects from the background.

I designed very simple function based on location from the center: Weighted distance from the center w.r.t the x direction + weighted distance from the center w.r.t the y direction. The center in x-dimension has the maximum value and the boundaries have 0. The maximum value is the parameter with which we can decide how important the location information is. This new energy function may be useful in the case where the object in the center has lower variations.

in reduceWidth function at 31

```
%For III.2
maxLocation = max(max(Energy))/4;
    for l=1:r
        for m=1:c
            Energy(l,m) = Energy(l,m)+round(maxLocation-2*maxLocation/r*abs(l-r/2)+maxLocation-
2*maxLocation/c*abs(m-c/2));
        end
    end
```

```
>> im = imread('cat.jpg');
% remove the % characters for above code
>> location = reduceWidth(im, 100);
```

Figure A shows the input image. Figure B is the image from ‘imresize’ function. Figure C and D show the result of original and modified schemes, respectively. The maximum value for the weighted distance is maxDerivative/4 where maxDerivative is the maximum value of the derivatives in the original scheme.

Figure A

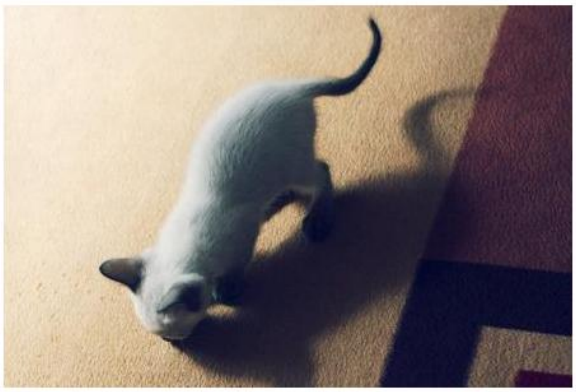


Figure B

Figure C

Figure D

3.
I modified the part of calculating energy function.

'reduceWidth' function at 39

```
%For III.3. Face detection
        energyFace = faceDetection(im);
        Energy = Energy + 255*energyFace;
```

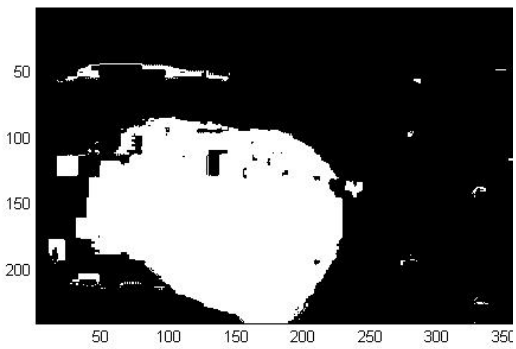
and defined 'faceDection' function to determine the regions of the faces.
In this function, I used the following rule to distinguish the face.
H < 1, S <= 0.8 and V > -0.603 * S -.0.039.
If the pixel is determined as the face, we increase +255 of the energy value of that pixel.
Figure A shows the input image(249x359) and Figure B is the result from the original content-aware resizing (with reduceWidth(100)).
Figure C and Figure D shows the binary image (1:face, 0:not face) and the resizing with energy increase at face, respectively.

Figure A

Figure B

Figure C

Figure D



4.

Using loop statement, fill the table of T(i,j) as shown in the paper. To calculate each element of T matrix, we maintain the r(# of row)*c(# of column) intermediate images. To reduce the overhead we maintain, ‘optimalReduce’ function maintains c (# of column) intermediate images for each column and fill the table T row-by-row. Follows images show the result. (input image: 256x171. Horizontal seams:100, Vertical seams:50). It is really difficult to distinguish, but Figure D and B are slightly different in the top area.

Figure A: original input

Figure B: Horizontal first

Figure C: Vertical first

Figure D: Optimal