Computer Vision: Problem set 1 (Part 2)

Donghyuk Shin

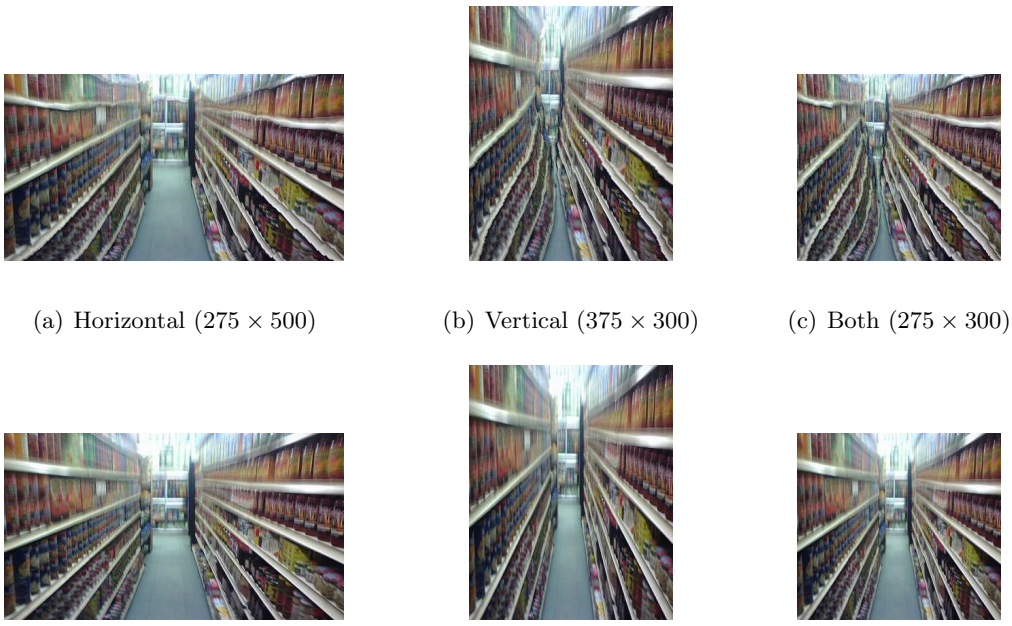
September 22, 2009

2 Programming problem: content-aware image resizing

5. Content-aware image resizing results on a number of images are given in Figures 2 through 8. We first examine how vertical and horizontal seam carving alters an image. We reduce the width, height and both using seam carving and compare it to simple resizing. The original image we use in this example is shown in Figure 1.

Figure 1: `groceries.jpg` original image (375×500).

Figure 2 displays each case of horizontal (Figures 2(a), 2(d)) and vertical (Figures 2(b), 2(e)) seam removal. In horizontal removal we observe that pixels from the white shelves have low energies. This will create a minimal seam path that passes through the shelves and seam carving will start to remove these pixels. Thus, the straight lines of the shelves shown in Figure 2(d) are turned into wobbly lines as in Figure 2(a). In the vertical case, since the middle aisle in the image has low energies, most of it has been taken out. As a result, the two shelves on each side are brought closer together, creating a distorted image. When horizontal and vertical seam removal are applied together (Figures 2(b), 2(e)), seam carving produced an image with both effects. Here, we alternated between horizontal and vertical seam removal.



(a) Horizontal (275×500) (b) Vertical (375×300) (c) Both (275×300)

(d) Simple resizing (275×500) (e) Simple resizing (375×300) (f) Simple resizing (275×300)

Figure 2: `groceries.jpg` with content-aware resizing.

We now examine cases where seam carving produces perceptually pleasing results. Figures 3 and 4 are such examples. Figure 3 shows that we can preserve meaningful objects in its original scale while redundant regions are removed. Seam carving first removes the backgroud regions, the sky, the sea, and the sand between seals. As a result, the group of seals at the bottom of the image are placed closely to each other. However, the algorithm is forced to remove high energy pixels when the image size is reduced too much as in Figure 3(d).

(a) Original (375×500)

(b) Content-aware resizing (175×300)

(c) Simple resizing (175×300)

(d) Content-aware resizing (100×225)

(e) Simple resizing (100×225)

Figure 3: `seals.jpg`

In Figure 4, we can see that the main focus of the image (the stage) is well preserved. Removed pixels are mainly unnoticeable pixels that blend with their surroundings. Note that even when changing an image to a completely different ratio as in this case, we get a more perceptually pleasing result than simple resizing.

(a) Original (375×500)

(b) Content-aware resizing (275×200)

(c) Simple resizing (275×200)

Figure 4: `concert.jpg`

From Figure 5, we can anticipate that the connected shaded and shadow areas of the trees will have low energy pixels by observing the original image (Figure 5(a)) . These are the regions that seam carving removed as in Figure 5(b), making the shadows and trees thinner. Branches are placed more closely together and some unnatural branches appear as a result.

(a) Original (375×500)

(b) Content-aware resizing (275×300)

(c) Simple resizing (275×300)

Figure 5: `trees.jpg`

To this point, it is obvious that content-aware image resizing with seam carving works well when there are many regions of unnoticeable pixels that blend with their surroundings, such as backgrounds. Moreover, important objects should have large differences in intesity values in order to survive seam carving. However, in other cases seam carving produces bad results

compared to simple resizing. Such cases are when the backgrounds contain some pattern of textures or important objects have similar intesity values within.

Figure 6 is an example when seam carving fails. The three main objects in the image, the house and the two vehicles, are severely distorted compared to their original shape. Since the whole object contains similar intesities within its body, the object itself gets carved out, while the surroundings such as the trees are preserved.

(a) Original (375×500)

(b) Content-aware resizing (275×300)

(c) Simple resizing (275×300)

Figure 6: `house.jpg`

Figure 7 is another example where seam carving does not work well. Notice the upper background of the image that has some high energy texture. This makes it hard for the algorithm to remove that region. While the content of the sign is unchanged, the person standing next to the sign is clearly disformed. This is due to the shrit that is very similar to the dirt road, guiding the minimal seam to go pass the person. Furthermore, the right leg of the sign which blends with the dark surroundings of the image is carved out.

(a) Original (375×500)

(b) Content-aware resizing (275×400)

(c) Simple resizing (275×400)

Figure 7: `sign.jpg`

Here, we show an example of different seams-order. Figure 8(b) is the result of alternating between horizontal and vertical seam removal, 8(c) is removing vertical seams first, and 8(d) is removing horizontal seams first. One can see that there is no major difference between these orders.

(a) Original (375×500)

(b) Alternate (225×350)

(c) Vertical seams first (225×350)

(d) Horizontal seams first (225×350)

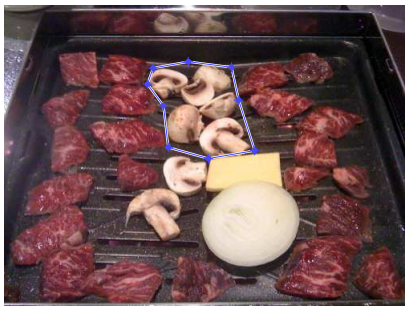
Figure 8: `shark.jpg` (provided by www.dailymail.co.uk)

3 Extra credit

1. We allow a user to mark an object to be removed. Then apply seam carving to remove the desired object in a content-aware manner. In order to remove a desired object with seam carving, we first attempted to remove seams until all marked pixels are gone. However, this resulted in unrealistic images. Normally, objects that are of interest have large energy values. Thus, seams that include marked pixels are likely to be removed after unnoticeable low energy seams and include high energy pixels from other parts of the image. This produces undesired results of content-aware object removal. Even if we leave these low energy seams and only remove seams that contain marked pixels, the resulting images were greatly distorted.

   Therefore, we use a different approach where we force the minimal energy seam to include marked pixels. This can be done by subtracting a large number from the energy values of marked pixels. Thus, the algorithm will start to remove seams that are within the target region. We calculate the smaller of the vertical or horizontal diameter in pixels of the selected object and perform vertical or horizontal removals accordingly.

   The Matlab function `removeObject.m` executes this procedure. The only argument to this method is the input image represented by a $m \times n \times 3$ RGB color matrix. Right after executing the function a pop-up of the original image will be displayed. The user can draw a desired area containing the object to be removed on this image. Once the drawn area is closed the function will execute rest of the procedure. We note that the `removeObject.m` function was executed in Matlab v7.9.0 (R2009a) and does not run under Matlab v7.5.0 (2007b). Figure 9 shows some results of this content-aware object removal.



(a) Selected an area.

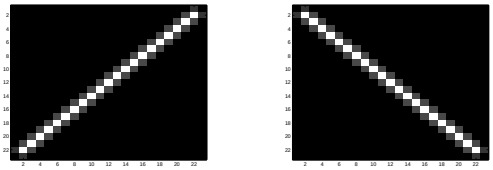
(b) Object is removed.

(c) Selected an area.

(d) Object is removed.

Figure 9: Content-aware object removal. (Bottom image provided by Steven Miller)

2. We apply a different energy function by using the 'motion' filter. This filter approximates the linear motion of a camera by a fixed length of pixels with a given angle. The length and angle are the parameters of this filter and it becomes a vector for horizontal and vertical motions. We denote the two motion filters that were used as m_1 and m_2, respectively, each shown in Figure 10. Here, we use a length 3 motion filter with 45 (m_1) and -45 (m_2) degree angles.



(a) Motion filter m_1. (b) Motion filter m_2.

Figure 10: Motion filters.

The new energy function e_{mot} can be written as:

$$e_{mot} = m_1(I) + m_2(I).$$

This should emphasize regions that have blurry effects representing some kind of motion involved in the image. Thus, images that will benefit from focusing on these types of regions will have better results than with the gradient magnitude e_1.

Figure 11 of the `groceries.jpg` image displays this advantage where the width is reduced by 325 pixels. Clearly, Figure 11(b) better preserves the main aisle that is the focus of the image than Figure 11(a). However, there are some artifacts at the bottom where the blurriness of the image is weak.

(a) Using e_1 (375×175)

(b) Using e_{mot} (375×175)

Figure 11: e_{mot} and e_1 applied to the `groceries.jpg` image (375×500).

Another example that e_{mot} works well is given in Figure 12 using `racing.jpg` (provided by www.travelks.com). The size of this image is 364×243 and is reduced by 100×100 pixels. In Figure 12(b) using e_1, we can see that the crowd of people on top of the image is preserved, since that part of the image has high values of gradient magnitude. Contrary, Figure 12(c) shows that the racing cars and the grass area are preserved in the reduced image producing a more desirable resized image.

(a) Original image (364×243)

(b) Using e_1 (264×143)

(c) Using e_{mot} (264×143)

Figure 12: e_{mot} and e_1 applied to the `racing.jpg` image.

3. We use the HSV channel to avoid warping regions containing people's face. Specifically, the hue (H) channel of the HSV color space is used to perform this task. A simple filter is created using a threshold value on the hue channel combined with the gradient magnitude. By examining the facial region of the hue channel, pixels with a hue value that fall under some threshold value are marked. Then some large number is added to the energy values of these pixels. Thus, we are assigning high energy values to face regions, which in turn will avoid seam carving of people's face. This works in the opposite direction of object removal.

   The threshold value is determined by the user. An image map displaying the positions of a given image that fall under this threshold is provided to the user to see how well the selected threshold distinguishes the face region from other parts of the image. An example of this map is given in Figure 13. If the region marked is not identical to the facial region of the original image, the user should select a different threshold value. The whole procedure is implemented in the Matlab function `faceImage.m` which only requires the input RGB image matrix as an argument.

(a) Original image (500×368)

(b) Positions marked using threshold= 0.07

Figure 13: White pixels on the right image map are points with hue values under the threshold.

Result of an example image is given in Figure 14. It is easy to see that using hue information with seam carving produces images that better preserves the face of a person. One drawback is that other parts of the image that might have a hue value lower than the threshold will also be untouched. Seam carving with the gradient magnitude filter alone treats the face region

same as other regions that are reletively less important. Thus, pixels within a face can get low energy values and get included in the minimal seam. In this example, pixels around the chin are removed as in Figure 14(b).

(a) HSV seam carving (300×268)

(b) Gradient seam carving (300×268)

(c) Simple resizing (300×268)

Figure 14: `obama.jpg`