CSE554
Yachen Qin(459310)
2018-12-08

Project Report

Seam carving algorithm

Introduction

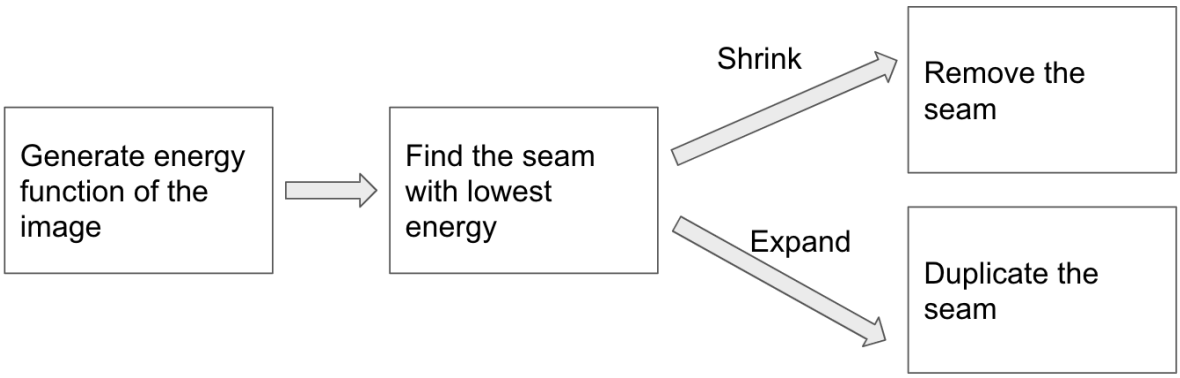
Tool	OpenCV
Coding Language	C++
Final Achievement-1	Could expand/shrink the size of the image via command line and store the output in the laptop.
Final Achievement-2	Could expand/shrink the size of the image in real time by the command from the keyboard and see the operation detail of the image.
Limitation	The expand part works not very well.

Problem Statement

Digital images are often viewed in many different display devices with a variety of resolutions. Variation of resolution makes viewing images difficult because they usually are resized to accommodate limited space. Simple attempts at resizing include scaling and cropping. Scaling reduces perceivable detail and cropping can't be done automatically. Also, cropping alters the image composition and is not always desirable.

This project is an implementation of a different image resizing approach called seam carving. Seam carving allows a change in size of the image by modifying the least noticeable pixels in an image. A typical application for seam carving is to reduce the size of an image along one dimension. This can be done by finding one pixel wide paths from the top to the bottom of the image and removing those paths. If the pixels in those paths are similar to surrounding pixels, then their removal may be unnoticed. Other seam carving applications include increasing the size of an image or changing the size of an image in two dimensions.

Algorithm Introduction



- Generating Energy Map of the Image

There are various methods to extract the unnoticeable pixel from an image.

In this project I choose the method to assign energy to each pixel by using a gradient operator to compute the gradient in both x and Y direction.

The Sobel operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. In this project, I use the function in OpenCV to calculate the gradient for horizon and vertical.

```
59      Sobel(gray, dx, CV_64F, 1, 0);
60      Sobel(gray, dy, CV_64F, 0, 1);
```

The official document of OpenCV said that Scharr filter is used which gives better results than 3x3 Sobel filter. In order to get a better result, I also use this function in OpenCV to calculate the gradient for horizon and vertical.

```
//calculate gradient on direction x
Scharr(gray, dx, CV_64F,  1, 0);
//calculate gradient on direction y
Scharr(gray, dy, CV_64F, 0, 1);
```

Original Image

Image after Sobel operator

Image after Scharr operator

We could see that there is no specific difference between those two images. So we could use either of them in my algorithm. After we know the energy of each pixel in this whole image, what I have to do next is to generate the energy map as I wrote before.

I use a two dimensions array to store the cumulative energy of a path taken from the top (in case of a vertical seam carving) and left(in case of horizontal seam carving) till the pixel(I,j).

I define the cumulative energy function as: M(i,j) = energy(i,j) + min(M(i-1, j-1), M(i-1, j), M(i-1, j+1)).

```
int dp[height][width];
for(int col = 0; col < width; col++){
    dp[0][col] = (int)image.at<uchar>(0,col);
}

for(int row = 1; row < height ; row++){
    for(int col = 0; col < width; col++){
        if (col == 0)
            dp[row][col] = min(dp[row-1][col+1], dp[row-1][col]);
        else if (col == width-1)
            dp[row][col] = min(dp[row-1][col-1], dp[row-1][col]);
        else
            dp[row][col] = min({dp[row-1][col-1], dp[row-1][col], dp[row-1][col+1]});
        dp[row][col] += (int)image.at<uchar>(row,col);
    }
}
```

Find the seam with lowest energy

In order to detect the optimal seam we consider the cumulative energy map such that the Pixel with the lowest value of M(i,j) in the last row is picked and is backtraced to obtain the optimal seam. And store the index of these path for every row in an array.

Image with 25 seams horizontal

Image with 25 seams vertical

Shrink part:

For the shrink part, every time I generate a new output image with height-1 or width-1 follow the command, and then remove that seam from the image.

Expansion part:

For the expand part, every time I generate a new output image with height+1 or width+1 follow the command, then duplicate the seam from this image. The expansion part is different from reduce part because the algorithm always find the smallest seam in my image. So I records the order of the seams and randomly chose in the first k smallest seam to duplicate in order to make the expansion not so weird.

The seam was duplicate after 10 times iteration

The seam was duplicate after 70 times iteration

From the pictures above we could see that if we expand the image much times, some parts in this image will be duplicate continuously, this may make the outcome of the image a little weird. For the expansion of the image, it is actually produce information from nothing, so the output can't be as real as we want.

How to use this tool

For this project, you could expand/shrink the image by giving their new height and width via command line or you could expand/shrink the image to see their change in real time from the keyboard command.

This project has been built as a Unix file to run on the laptop. Then you could just follow the instruction show on the display window of this tool.

You could first provide the path of your image and then choose to do the operation via command line or keyboard in realtime.

Result display

The shrink of image using Sobel energy function

(756x472)

(500x300)

The expand of image using Sobel energy function

(500x406)

(600x500)

The shrink of image using Scharr energy function

(512x384)

(300x200)

The expand of image using Scharr energy function

(500x281)

(600x300)

Limitations and future work

Now from the previous results we could see that there are still some bugs with the expansion part. I think it may because I calculate the energy map every time for remove/ duplicate a seam. So the path of the seam may change. And the part in the image with lower energy will always been chosen and make the output image looks wired of some part.

Another interesting feature that could exploit is object removal. This would require some user input to define the area or object to be removed. Then the algorithm would simply do the seam carving while making sure to select pixels inside the area to be removed.

Appendix

Link to the original code in github:

https://github.com/YachenQin/CSE554

Reference:

[1] S. Avidan and A. Shamir, Seam carving for content-aware image resizing. ACM Trans. Graph., 26(3):10, 2007