Liquid rescale of images

Team 17

TANISHQ JAIN

1 INTRODUCTION

Liquid re-scaling (Seam carving) is an algorithm for content-aware image resizing. The variety of display devices today imposes new demands on digital media. E.g. In a website, we need to design multiple layouts to support dynamic changes as per the device. Similarly, we need an algorithm to resize images without distorting their content. There are limitations to simple operations. We can't modify the aspect ratio effectively with standard scaling. Cropping can only remove pixels from the boundaries. On the contrary, Liquid re-scaling uses an energy function that defines the importance of pixels. A vertical (horizontal) seam is a connected path of low-energy pixels crossing the image from top to bottom (left to right). Hence, the algorithm optimally removes (inserts) seams successively to reduce (extend) the size of the image. This enables considering the image content along with the geometric requirements while resizing. Apart from the primary focus of image re-targeting, the algorithm can also be extended for content amplification, object removal, and multi-size images.

2 LITERATURE REVIEW

In their research paper, Avidan and Shamir [1] introduce the Seam carving algorithm and explain the details of its implementation. They define an energy function, $e_1(I) = |\frac{\partial I}{\partial x}| + |\frac{\partial I}{\partial y}|$ to measure the importance of a pixel. It is the L1-norm of the intensity gradient of a pixel. Removing pixels with the lowest energy (overall) destroys the rectangular shape of the image. Hence, we need to remove an equal number of low-energy pixels from each row (column). Furthermore, successive pixels need to be close to prevent the zig-zag effect. Formally, if I is an nXm image, we define a vertical seam as:

$$S^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \ s.t. \ \forall i, |x(i) - x(i-1)| \leq 1$$

where x(i) maps i^{th} row to column of the pixel in seam. Hence, it is an 8-connected path of pixels from top to bottom, containing one pixel in each row. Similarly, we can define a horizontal seam as:

$$S^y = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m, \ s.t. \ \forall j, |y(j) - y(j-1)| \leq 1$$

Pixels of a vertical seam s can be denoted as $I_s = \{I(x(i), i)\}_{i=1}^n$. Given energy function e, we can define the cost of a vertical seam s as $E(I_s) = \Sigma_{i=1}^n e(I(x(i), i))$. We can find the optimal vertical seam s^* with minimum cost using dynamic programming. Suppose that M(i, j) stores the value of the minimum cumulative energy of all seams arriving at pixel (i, j) from the top. Hence we have the recurrence relation:

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

The base case is $M(i, j) = e(i, j) \forall$ pixels in the top row. Finally, the minimum value of the last row indicates the end of the minimal vertical seam. This seam can be reconstructed by backtracking to the top row. Thus, we can successively carve out minimal vertical (horizontal) seams to reduce the width (height).

To enlarge an image, we need to insert new seams. We can approximate inverting the seam removal process. Hence, we can compute an optimal seam s in I and average it with neighboring pixels. Then, insert the computed seam next to s. However, this can create an uneven stretching by choosing the same seam repeatedly.

To achieve effective enlarging, we need to balance the artificially inserted parts with the original content. Hence, to enlarge an image by k seams, we consider first k seams for removal and duplicate them. This can be visualized

Author's address: Tanishq Jain, tanishq21294@iiitd.ac.in.

as traversing back in time and recovering those pixels removed from a larger image to get the current image. But, we need to perform excessive enlarging in multiple steps. Each step enlarges image size up to a limit (say 25%), guarding the important content from being distorted.

Image re-targeting involves resizing image I of size nXm to size n'Xm'. This leads to the question of finding the optimal order of horizontal/vertical seam operations. We define the problem as minimization (s^x, s^y, α) of the objective function:

$$\Sigma_{i=1}^{k} E(\alpha_i s_i^x + (1 - \alpha_i) s_i^y)$$

where r = |m - m'|, c = |n - n'|, k = r + c and α_i determines whether we work with a horizontal/vertical seam at step i. $\alpha_i \in \{0, 1\}$, $\Sigma_{i=1}^{k} \alpha_i = r$ and $\Sigma_{i=1}^{k} (1 - \alpha_i) = c$. The implementation involves dynamic programming. Let T is a transport map that stores the cost of the optimal sequence of seam operations for each desired size n'Xm'. T(r, c) = minimal cost to obtain the image of size n-r X m-c. Base case is T(0, 0) = 0 and recurrence relation is:

$$T(r, c) = min(T(r - 1, c) + E(s^x(I_{n-r-1Xm-c})), T(r, c - 1) + E(s^y(I_{n-rXm-c-1})))$$

where I_{aXb} denotes image of size aXb, $E(s^x(I))$ and $E(s^y(I))$ are cost of horizontal, vertical seam operations respectively.

Content amplification can also be achieved by combining Seam carving and standard scaling. First, use standard scaling to enlarge the image and then seam carving to revert to its original size. This results in an amplification of content-rich zones in the image without altering the size.

Multi-size images encode the order of Seam operations for Image re-targeting to a range of sizes. This information has a low memory requirement and vastly improves the computation speed. It is great for images embedded in a web page, allowing them to resize in real-time. It is implemented using an index map V of size nXm which stores for each pixel, the index of the seam that removed it. This can support image reduction as well as enlarging. However, we can't support optimal resizing in both directions using H and V, as removing a seam in one direction may destroy the index map of the other.

In object removal, the user marks the target object to be removed from an image using an interface. Then seams are removed using the smaller out of vertical, and horizontal diameters until all marked pixels have been removed. Finally, seam insertion is used to regain the original size of the image.

3 MILESTONES

S. No.	Milestone	Member
	Mid evaluation	
1	Image setup and Intensity evaluation	Tanishq
2	Energy function evaluation	Tanishq
3	Computing optimal seams (vertical, horizontal)	Tanishq
4	Reduction using seam removal	Tanishq
	Final evaluation	
5	Enlarging using seam insertion	Tanishq
6	Computing optimal order for image re-targeting	Tanishq
7	Implement content amplification	Tanishq
8	Implement multi-size images	Tanishq
9	Implement object removal	Tanishq

4 APPROACH

On running the application, the user is prompted for the name of the input image and the desired dimensions of the output image. The image is then set up using imread() method of the OpenCV library. An intensity matrix is set up which stores the intensity value and original coordinates for each pixel. Given the RGB values for a pixel, intensity is calculated as $Intensity, I = 0.3 * R + 0.59 * G + 0.11 * B$ (Refer: Luminosity method, https://www.baeldung.com/cs/convert-rgb-to-grayscale).

Now, an energy matrix is set up which stores the energy (denoting importance) for each pixel. Energy is computed as the 2-norm of intensity gradient for the pixel. $Energy, E = || \ [I_{right} - I_{left}, \ \ I_{down} - I_{up}] \ ||_2$. Then, a cost matrix is set up using dynamic programming. Suppose we have to find the optimal vertical seam. Hence, cost matrix stores the value of minimum cumulative energy out of all seams arriving at a pixel from the top. We have the recurrence relation as $Cost, C[i][j] = E[i][j] + min(C[i-1][j-1], C[i-1][j], C[i-1][j+1])$. Base case is $C[i][j] = E[i][j] \ \forall$ pixels in the top row (i.e. i = 0).

Now, the optimal seam can be constructed by backtracking. We start with the pixel having minimum cost in the last row, say (x_1, y_1). Next, select a pixel (x_2, y_2) with minimum cost from the row above, such that $|x_2 - x_1| \leq 1$. Repeating the process till the first row yields the optimal vertical seam. To find the optimal horizontal seam, apply the entire procedure on the image from left to right, instead of top to bottom. Once the optimal seam is obtained, image reduction is simple. For all pixels in the seam, set the intensity value to -1. Then, apply a swapping procedure which shifts all removed pixels to form a single column (row) at the right (bottom) end of the image, after a vertical (horizontal) seam removal.

After applying seam removals to achieve the desired dimensions, a final intensity matrix is obtained. To construct the output image, iterate over the final intensity matrix and map the pixel colors using the original image. This can be done since the intensity matrix also contains information on the original coordinates for each pixel. The output image is then written using imwrite() method of the OpenCV library.

To achieve effective seam insertion, we insert k seams at a time. This is done by inverting the seam removal process. We create a copy of the intensity matrix and simulate the removal of k seams on it. The pixels removed in these operations are tracked using a boolean matrix. These pixels are finally duplicated in the original matrix. We also define a recursive function that can correctly return the color for inserted pixels.

Optimal order image re-targeting can be used when both width and height need to be reduced. It involves setting up a 2D DP and a bitmap. dp[r][c] stores the optimal cost of reducing height, and width by r, c respectively. It also stores the intensity matrix after the optimal operations. At the same time, we maintain a bitmap to track the next choice at each state. This bitmap can be used to recover the optimal order of seam operations.

For content amplification, we input a scaling factor. The input image is scaled by this factor, keeping the aspect ratio constant. After that, we use seam removal in both directions to regain the original size of the image. This results in the amplification of important content in the original image.

For multi-size image, we first input the maximum supported width and height. Using simple retargeting, we obtain the largest supported image from the input image. Then, we need to remove seams one by one and track the index of the seam removing any given pixel. Finally, we write these indices in a text file to support multi-size image functionality.

In object removal, we work on a marked image, i.e. the user marks regions to be removed using a highly specific color. We chose a red color with RGB (255, 0, 0) for the purpose. We achieve the goal using seam removals. The important step is that while setting the energy of pixels, we set a large -ve value for marked pixels to incentivize their removal. Finally, seam insertion is employed to regain the original image dimensions.

5 RESULTS

The image reduction algorithm was tested on multiple input images. It was tested for both width and height reduction. The results are demonstrated below. Images on the left and right are the input and output images respectively. Dimensions are represented as Width x Height.

Fig. 1. Width reduction (source: https://en.wikipedia.org/)

Width reduction using vertical seam removal - The input image has dimensions 274 x 186 pixels and the output image has dimensions 200 x 186 pixels (fig. 1). Notice that lower energy pixels corresponding to empty grasslands and sky have been removed while preserving the castle and the person looking at it.

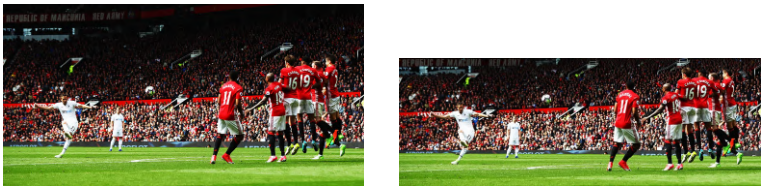

Fig. 2. Height reduction (source: https://premierskillsenglish.britishcouncil.org/)

Height reduction using horizontal seam removal - The input image has dimensions 1140 x 570 pixels and the output image has dimensions 1140 x 380 pixels (fig. 2). Notice that lower energy pixels corresponding to the audience have been removed while preserving the players and the club flag.

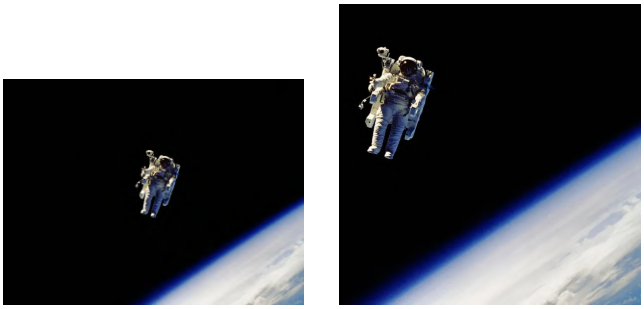

Fig. 3. Width and height reduction (source: https://www.smithsonianmag.com/)

Width and height reduction using vertical, and horizontal seam removal - The input image has dimensions 1000 x 750 pixels and the output image has dimensions 600 x 600 pixels. Notice that lower energy pixels corresponding to empty space have been removed while preserving the astronaut and the Earth.

The results of image enlarging by seam insertion are demonstrated below.

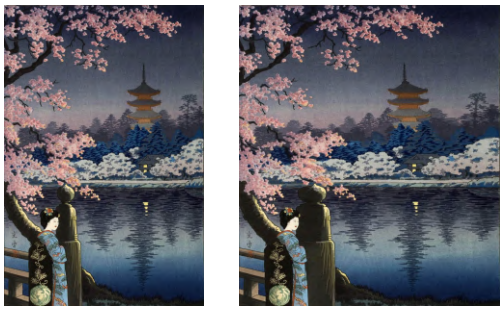

Fig. 4. Width enlarging (source: https://www.artelino.com/)

Width enlarging using vertical seam insertion - The input image has dimensions 1000 x 1520 pixels and the output image has dimensions 1300 x 1520 pixels. Notice that lower energy pixels corresponding to the tree and lake have been duplicated while preserving the lady and the tower.

Fig. 5. Height enlarging (source: https://www.discoverwildlife.com/)

Height enlarging using horizontal seam insertion - The input image has dimensions 1030 x 685 pixels and the output image has dimensions 1030 x 820 pixels. Notice that lower energy pixels corresponding to the sky and lake have been duplicated while preserving the elephants.

Fig. 6. Width and height enlarging (source: https://www.easyplanettravel.com/)

Width and height enlarging using vertical, horizontal seam insertion - The input image has dimensions 890 x 560 pixels and the output image has dimensions 1200 x 900 pixels. Notice that lower energy pixels corresponding to sky and grasslands have been duplicated while preserving the castle and reflection, giving a zoom-out effect.

Optimal order image re-targeting can be performed when both height and width need to be reduced. It involves more computation (solving a dynamic programming problem) and hence takes significantly more time.

Fig. 7. Optimal order image re-targeting (source: https://www.livemint.com/)

The leftmost image is the input image with dimensions 690 x 390 pixels. The middle image is output from simple re-targeting and the right image is output from optimal order re-targeting. Both have dimensions of 450 x 390 pixels. In practice, it was observed that outputs were largely similar for most inputs.

The results for content amplification are demonstrated next. The input image has dimensions of 2050 x 1360 pixels and 1.2 is the scaling factor. Output has the same dimensions but the content-rich zones are amplified.

Fig. 8. Content amplification (source: https://dailystoic.com/)

Finally, we demonstrate the results of object removal.

Fig. 9. Object removal (source: https://www.thepopverse.com/)

The first image is the original image. The middle image is the marked version, which is the input. In this image, we mark the second sun to be removed. The last image is the output.

Fig. 10. Object removal

In the above 2 examples, the algorithm produces desirable output and magically removes the marked objects. However, a limitation was observed. The algorithm may not perform well in crowded images, or when we try to remove a major portion of the image. This is demonstrated in the following example.

Fig. 11. Object removal (source: https://www.esquire.com/)

6 CONCLUSION

All the milestones proposed initially have been achieved. They cover all the major ideas presented by authors Avidan and Shamir [1] in their research paper. The strengths and shortcomings of the algorithms have been demonstrated in the results.

REFERENCES

[1] Shai Avidan and Ariel Shamir. 2007. Seam carving for content-aware image resizing. *ACM Transactions on Graphics* 26, July (2007).