Rectangling the Panorama

– Re-implementation Report

Adheesh Gokhale
109293558
adheesh.gokhale@stonybrook.edu

Abstract: This report details the challenges faced and solutions used while implementing the paper Rectangling the Panorama[1]. Sometimes, simpler methods were used, than the ones used in original paper. Results obtained have been compared.

1 INTRODUCTION

This paper [1] deals with the problem of changing boundary of stitched panorama to a rectangular boundary. Following image describes the problem:

The solution provided consists of three steps: Mesh-free Local warping, placing a rectangular mesh on the locally-warped image and mesh-based global warping. Mesh-free Local warping uses the seam carving technique to expand the image to the rectangular boundary. Then a rectangular mesh is placed and the image is rolled-back to its initial state. This process also rolls-back the mesh. Then the quad-based mesh expansion is used to globally warp the image to a rectangular boundary. This is done by considering the cumulative quad-energies as well as the energy changes involved when orientation of lines/edges changes from input to output. Finally rescaling is applied to improve the aspect ratio of the rectangle.

2. PRE-PROCESSING

The image is downsampled. In the paper, image is downsampled to the size of 1MB. In this implementation, image is scaled down by factor of 4 in each dimension. Thus the total area reduces by 16 times. This is done to reduce time taken for the remaining process. For experimental basis, downscaling by a factor of 2 (total size reduction of 4) was tried. But that increased the time taken by nearly 6 times.

This is followed by removing the unwanted rectangular white sections from the image (as shown below).

3. SEAM CARVING

Purpose of seam carving is to get a rectangular image on which a mesh can be placed. The result seam carving cannot be used directly, because, it does not expand all the regions equally. As can be seen with the image below, the regions through which orange seam passes, have expanded, but others have not.

Seam carving works like this: First, the longest boundary segment(defined as "as a connected sequence of missing pixels on one of the four sides (top/bottom/left/right) of the target rectangular boundary" [1]) is found.

The corresponding sub-image is then chosen.

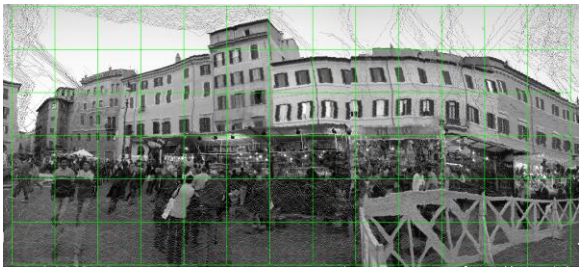
A horizontal (if the boundary segment is on top or bottom) or a vertical seam (if the boundary segment is on left or right edge) is then carved. Energies of missing white pixels are marked at infinite to ensure that seam does not pass through the missing pixels. This seam is added to the sub-image. "In the viewpoint of filling the unknown regions, inserting a seam would reduce the number of missing pixels in the image (this number equals to the number of pixels on the seam)." [1] While inserting the seam, the displacement history is stored for future use. Following is the result of adding five seams. Red circle shows the reduction of missing area. The seams can be found near the bottom.

The process of inserting seams is repeated till entire rectangle is covered and there are no missing pixels. Below are some stages of the process.

A discussion is needed here, of the method employed in the paper, and in this implementation.
A seam cannot be inserted directly into an image. Doing so causes the same seam to be repeatedly added. The right method (when k seams need to be added), is to find k candidates for seam removal and then add them. Now, since seam removal adds energy to the resulting image, a new method for seam carving was implemented in 2008[4]. The authors of [1], use this improved method for finding the k candidates for seam removal and then add them.
While re-implementing, a simpler method was employed. When a new seam is inserted, every second pixel in that seam is assigned a random value. This increases the energy around that pixel, and thus, the reducing chances of same seam being repeatedly carved.

4. MESH PLACEMENT AND ROLLBACK

A rectangular mesh is placed on the image. Authors use a mesh consisting of nearly 400 points. ~100 points have been used in re-implementation. This is done to save on time. Large mesh points mean large processing time.



This is followed by rollback, to get a distorted seam placed on the original image.

The process used for rollback uses the displacement history stored in previous step. Using this history, displacement values are found only for the mesh-points and grid-lines.

5. GLOBAL WARPING

The mesh based global warping tries to find a new configuration of the mesh that minimizes the following function[1]:

$$E(\mathbf{V}, \{\theta_m\}) = E_S(\mathbf{V}) + \lambda_L E_L(\mathbf{V}, \{\theta_m\}) + \lambda_B E_B(\mathbf{V})$$

The last term signifies the boundary conditions. This ensures that final image conforms the rectangular boundary.

$$E_B(\mathbf{V}) = \sum_{\mathbf{v}_i \in \mathrm{L}} x_i^2 + \sum_{\mathbf{v}_i \in \mathrm{R}} (x_i - w)^2 + \sum_{\mathbf{v}_i \in \mathrm{T}} y_i^2 + \sum_{\mathbf{v}_i \in \mathrm{B}} (y_i - h)^2.$$

The middle term ensures that line preservation [8] is enforced. Enforcing this preserves the orientation, parallelism and collinearity of the lines in the image. This energy has not been implemented in re-implementation, due to the complexity of the task involved.

$$E_L(\mathbf{V}, \{\theta_m\}) = \frac{1}{N_L} \sum_{j} \|C_j(\theta_{m(j)})\mathbf{e}_{q(j)}\|^2,$$

The first term preserves shape. This will be discussed in detail in the next section.

The first and the last terms of the main equation have been implemented.

6. SHAPE PRESERVATION

This energy function encourages the mesh points to undergo a transformation, such that following function is minimized.

$$E_S(\mathbf{V}) = \frac{1}{N} \sum_{q} \|(A_q(A_q^T A_q)^{-1} A_q^T - I)\mathbf{V}_q\|^2.$$

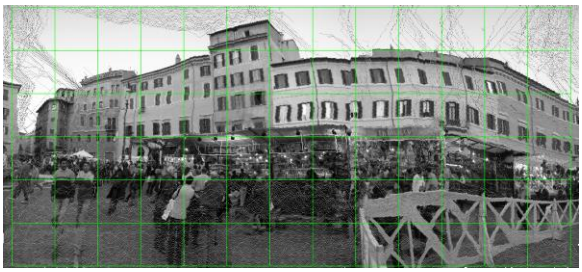
Here,

$$A_q = \begin{bmatrix} \hat{x}_0 & -\hat{y}_0 & 1 & 0 \\ \hat{y}_0 & \hat{x}_0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \hat{x}_3 & -\hat{y}_3 & 1 & 0 \\ \hat{y}_3 & \hat{x}_3 & 0 & 1 \end{bmatrix}, \quad \mathbf{V}_q = \begin{bmatrix} x_0 \\ y_0 \\ \vdots \\ x_3 \\ y_3 \end{bmatrix}.$$

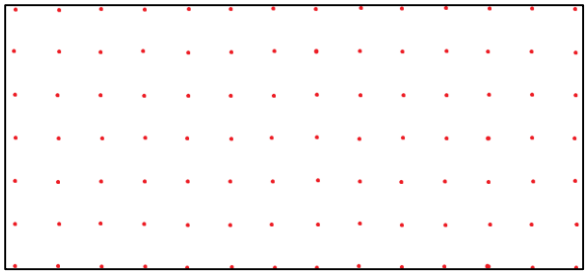
Aq is formed from the 4 quad vertexes of input image and Vq from the vertexes of output image. Following image shows a quad:

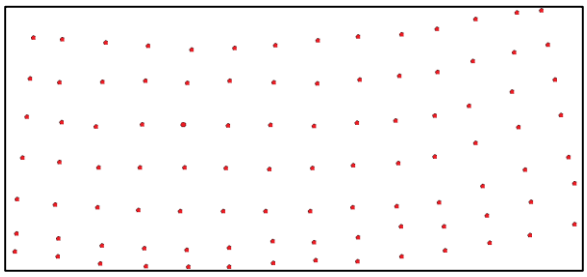
The way this equation is solved during the re-implementation is as follows.

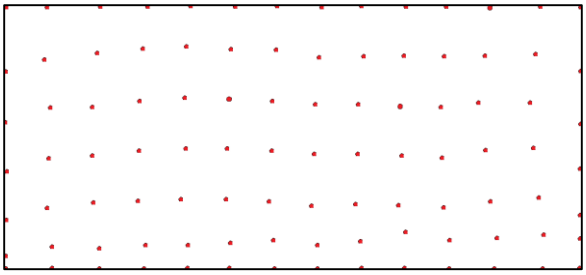
The initial mesh (shown below) is taken as the starting value.



Then all the grid points on the left, right top and bottom boundary of the mesh are pushed to the rectangular boundary. This means, for example, that the grid points on top boundary are moved to the top boundary of rectangle, while preserving their horizontal location.
Now, the following process is applied to each interior quad pixel.
The shape preservation energy is calculated for 5 possible options: the quad pixel at current location, at one pixel location above, at one pixel location below, at one pixel location right and at one pixel location left. Of these the minimum energy value decides the new location for the quad pixel.
Once the above process is applied to each interior pixel, it is also applied to the pixels on the boundary. The only difference is that boundary pixels are allowed to move sideways (for top and bottom boundaries) or up/down (for left and right boundaries). This process defines one iteration. Fifty such iterations are performed. Following image shows the initial mesh, input mesh(from rollback) and result:



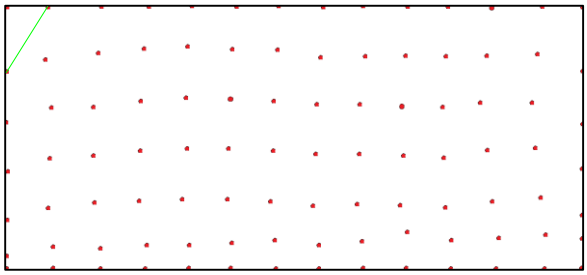




While implementing this, an issue was faced. Since 2D iteration typically runs i->0 to n ; j->0 to m, the grid seemed to be pushed away from the top corner. To resolve this, following approach was selected, in a given row, the pixels would be processed in following order i -> 0,n,1,n-1,2,n-2…… Similarly, the rows were processed in following order, j->0,m,1,m-1,2,m-2….

Once the target grid is achieved, the remaining task is to copy quads. This is a complex task than it apparently seems, mainly because the input and output grids not geometrically similar. The change is in terms of translation, scaling and rotation.

The solution is to use Affine transforms. Implementing Affine transform is easier when mesh is triangular, but the mesh available is a quad mesh. To solve this following approach was considered: find the intersection of diagonals in a quad. This will be give four triangles to operate on. But considering the added complexity, a simpler approach was used. By taking only one diagonal, Affine transform is implemented.



The final result looks like this:

The image is upscaled using linear interpolation to get the original size, before saving to disk. The result of original paper looks like this:

7. OTHER RESULTS

Original

Paper

Re-implementation

Original

Paper

Re-implementation

8. COMPARISON

In terms of efficiency, authors have implemented very fast solutions. While it takes them 1 second to process an image, the re-implementation needs average of about 40 seconds. The high efficiency could be achieved with concurrent programming. Apart from this there are three issues discussed ahead:

9. SCOPE FOR IMPROVEMENT: BETTER SAMPLING AND INTERPOLATION METHODS

One of the test cases showed very choppy result. This is the effect of the simplistic sampling and interpolation methods. This was verified by using a factor of 2 instead of 4 for down and upscaling.

Original

Paper

Re-implementation – Factor of 4, choppy

Re-implementation – Factor of 2, better

10. SCOPE FOR IMPROVEMENT: LINE PRESERVING

There are some images where line preserving becomes very important. Below is the example

Original

Paper

Re-implementation

12. SCOPE FOR IMPROVEMENT:SMART SCALING

Sometimes, the target rectangle has an unpleasant aspect ratio. To get a pleasing image, authors scale the image to a new target rectangle with a "good" aspect ratio (like 4:3, 16:9 etc.). This has not been implemented in Re-implementation.

Original

Paper –good aspect ratio

Re-implementation

13. CONCLUSION AND PLANS FOR FUTURE

The re-implementation spans C code of 4200>lines (~3500 excluding the comments and dev/test purpose code). Of this a large

component was written by me. The only code referred from internet was for tasks like reading/saving .bmp file, matrix inversion, matrix multiplication. This effectively means that not one but three papers ([1],[7],[3]) have been implemented. Developing this over a period of three months was challenging, but at the end rewarding too. Although this was done as a course work, I intend to finish this in future (ie. Remove the above mentioned short comings and improve efficiency). The final purpose is to make an application available for photographers to use(probably for free, and with permission from authors).

14. REFERENCES

[1] K He, H Chang, J Sun (2013) Rectangling
panoramic images via warping. SIGGRAPH 2013 Conference

[2] SETLUR, V., TAKAGI, S., RASKAR, R.,
GLEICHER, M., AND GOOCH, B. 2005. Automatic Image Retargeting. In In the Mobile
and Ubiquitous Multimedia (MUM), ACM Press.

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

[4] M. Rubinstein, A. Shamir, and S. Avidan.
Improved seam carving for video retargeting.
ACM Trans. Graph., 27(3):16 16, 2008.

[5] W Dong, N Zhou, JC Paul, X Zhang 2009.
Optimized image resizing using seam carving
and scaling. Proceedings of ACM SIGGRAPH
Asia 2009.

[6] Y.-S. Wang, C.-L. Tai, O. Sorkine, and T.-Y.
Lee. Optimized scale-and-stretch for image resizing. ACM Trans. Graph., 27(5, article 118),
2008.

[7] ZHANG, G., CHENG, M., HU, S., AND MARTIN, R. 2009. A shape-preserving approach to image resizing. In Computer Graphics Forum, Wiley Online Library, 18971906.

[8] CHANG, C.-H., AND CHUANG, Y.-Y. 2012.
A line-structure preserving approach to image
resizing. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1075
1082.