Image Resizing by Seam Carving in Python and Matched Masks

Alexander Converse

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: alexander.converse@case.edu

ABSTRACT

This paper explores a recently developed technique called "seam carving" [1] to remove low energy seams from the image to create a crop that preserves more information. Rather than interpolating new pixels or removing a rigid column of pixels, seam carving removes fluid seams of 8-connected pixels. It further explores the concept of a matched mask to prevent distortion.

KEYWORDS

Image, Resize, Crop, Retarget, Seam Carving, Retargeting, Matched Mask, Python

INTRODUCTION

Shrinking images to fit in a smaller space than the original image traditionally has employed scaling (e.g. bicubic resizing) or cropping rows and columns edge pixels. Researchers at the Mitsubishi Electric Research Lab have developed a new technique called seam carving to remove seams of 8-connected pixels constrained in such a fashion that there is one pixel per row for vertical seams or per column for horizontal seams [1].

Seam carving has been grossly popular since its introduction spawning many implementations [2], [3], [4], [5]. In photographs of people seams may often travel through faces causing a disproportion perceptional deforming compared to energy removed. To combat this seam carving can be combined with automatic face detection [6] and a weighting mask causing marked areas to repel seams.

Another use of this algorithm is to remove unwanted objects from an image. This can also be achieved with a mask this time to attract seams to certain areas. The process can also be used to make images larger by adding seams [1].

The algorithm does cause heavy distortion on some images. Sometimes it can be combated by a simple mask applied to the images energy function but sometimes that doesn't help. A matched mask can be created to prevent distortion of a region of interest. This approach does not seem to be covered in the original literature [1].

ALGORITHM

An energy/complexity function is applied to the image. If present the mask is applied to the energy function at this time. A minimal energy seam through the energy image is calculated at this point. The seam is then removed from the image shifting pixels left for a vertical seam or up for a horizontal seem. This process is repeated for each seam to be removed.

The complexity function chosen was the absolute sum of gradients:

$$\begin{aligned} e_1[x,y] = &\left|im[x-1,y]-im[x,y]\right| + \\ &\left|im[x+1,y]-im[x,y]\right| + \\ &\left|im[x,y-1]-im[x,y]\right| + \\ &\left|im[x,y+1]-im[x,y]\right| \end{aligned} \tag{1}$$

This energy function was one of the two that the original creators of the technique found most successful [1]. If the image is color it is converted to grayscale for this step for computational simplicity, though there is no other reason why the gradient couldn't be applied to each channel. Half point symmetric padding is used at image edges. An example of an image and its energy image can be seen in Figures 1 and 2.

If present the area attractance/avoidance matrix is applied to the energy function by adding four times the green channel of the mask to the energy image for areas to preserve and subtracting four times the red channel of the mask to the matrix for areas to remove. The scaling factor of four is present in both cases because four absolute gradients are added.

To find a minimal energy seam, first the energy image is converted into a cumulative energy image. This is done by starting one row below the bottom and adding the minimum of the 3 8-conencted pixels from the row below:

$$\begin{aligned} e_{cum}\left[x,y\right] = &\, e_1\left[x,y\right] + \\ &\min\begin{pmatrix} e_{cum}\left[x-1,y+1\right], \\ e_{cum}\left[x,y+1\right], \\ e_{cum}\left[x+1,y+1\right] \end{pmatrix} \end{aligned} \tag{2}$$

The direction of the movement is saved in a matching paths image:

$$path\left[x,y\right] = \operatorname{argmin}\begin{pmatrix} e_{cum}\left[x-1,y+1\right], \\ e_{cum}\left[x,y+1\right], \\ e_{cum}\left[x+1,y+1\right] \end{pmatrix} \tag{3}$$

This process is repeated for each row working toward the top of the image. The seam chosen is then the lowest vale from the top row of the cumulative energy image and the

appropriate movements from the paths matrix. Because paths converge rapidly the energy and path information must be computed from scratch for every seam removed.

Removing the best of a set of random seams was suggested by one implementer in his work [2] and in his response to others [3]. However, this caused significant performance degradation in my implementation.

The seam is removes by shifting pixels left or up (for vertical or horizontal seems respectively) in place of the seam.

When both horizontal and vertical seams need to be removed, the seams are removed in alternating order. This is not ideal removal order but it is close and saves a larger computation step [1].

There is also a seam visualizer that colors in the seams rather than removing them. It works by maintaining a mapping of pixel positions in the resized image to their original positions in the original sized image. The mapping is used to translate the seam to be removed to its original coordinates and color it in on a copy of the original image. The map is update by removing the seam from the map after coloring using the exact same method as removing the seam from the image.

This method requires a storage size of twice the image size but seems to be the only sane way to deal with crossing the same seams multiple times and differentiating seam direction when compensating for removed seams.

MATCHED MASKS

Sometimes due to the nature of the image the seam carving algorithm causes severe distortion. Additive masks can be used to adjust this. Additive masks were discussed earlier in the algorithm section. The problem is that in some areas that are of equal visual importance energy varies in bands having some bands of high energy and some of low energy, this causes seams to condense in the low energy areas. If the area is irregularly shaped this often causes huge distortion (figures 9-10). The easy solution to this is to equalize the energy over the area. This is easily accomplished by taking the energy of the region of interest and subtracting it from the regions maximum value to create an additive mask. This however usually causes all seams to avoid the area causing the area to undesirably dominate the image. This can be combated by creating a second mask, this time subtractive, in the same shape but of constant intensity. The constant intensity should be around or a little below the average of the additive mask. The mask should be normalized by dividing by the scaling constant used when applying the mask (in this case 4). An example of such a mask can be seen in figure 15.

IMPLEMENTATION DETAILS

The algorithm is implemented in python using the Numpy [7] library for numerical computation and the Python Imaging Library [8] for file input/output. Matrix and vector computations are used so that the heavy lifting is done in Numpy's compiled and vectorized C and FORTRAN instead of element-by-element in Python. The usage of Numpy disallows the use of more modern, faster, more experimental python interpreters like IronPython, Jython, or PyPy. However, the psyco JIT for python can be used to optimize code on platforms where it is present and supported. Scipy's weave module can be used to further optimize the code [9].

All algorithms are implemented in the vertical direction only. Horizontal seam removal is done by the vertical methods after transposition.

The overall performance is a little under 1 second per seam removed including marking the seams which is unnecessary in most cases where analysis afterwards is not requires.

DISCUSSION OF RESULTS

The Lena test image is shown resized from 512x512 to 480x480 in figures 1-4. The seams do a pretty good job of avoiding the important areas of the image however several lines go straight through her face causing an odd distortion. This can be combated by using a weighting mask as shown in figures 5-7.

Figure 1: Lena Image

Figure 2: Lena's energy map

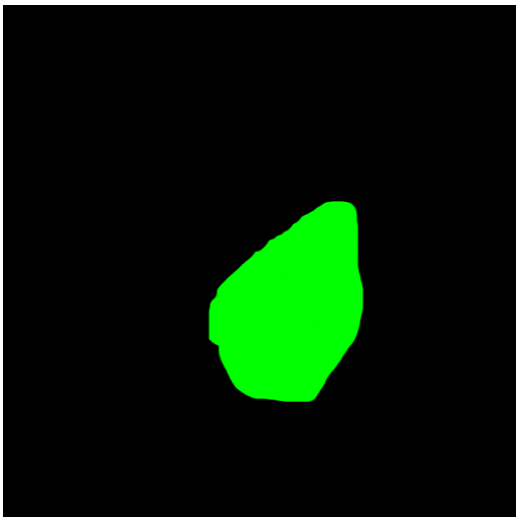

Figure 5: Additive mask applied to Lena image

Figure 3: Lena's seams marked for removal

Figure 6: Seams to be removed from masked Lena

Figure 4: Lena resized

Figure 7: Lena resized with mask

Large resizes can cause significant distortion in the image. In figures 8-10, a picture was resized from 768x1024 to 480x640. The trees distorted to the level of a Dr. Seuss

illustration, and where the waves start breaking the trunks get pinched out. The pinching out of the trunks can be removed with a simple mask (figures 11-12). The distortion of the trunks is a little more complicated. The trunks have a banded texture that seems to concentrate the seams in bundles on the bands. Making the mask bigger to include the whole tree trunks causes the trunks to dominate the image and strange diagonal sheering is visible on the trunks (figures 13-14).

This problem can be solvable by a variable intensity mask that evens energy on the trunk (figures 15-16). The matched mask was generated manually in an image manipulation program but the process could be automated only requiring manual specification of a region of interest. The process is described on in the section of this paper titled Matched Masks. The sky still looks a little damaged but overall it looks considerable better than the first attempt at resizing. It is important to remember that in this case over half of the pixels in the image were removed.

Figure 8: Venice Beach Image

Figure 9: Seams to be removed from Venice Beach

Figure 10: Venice Beach resized

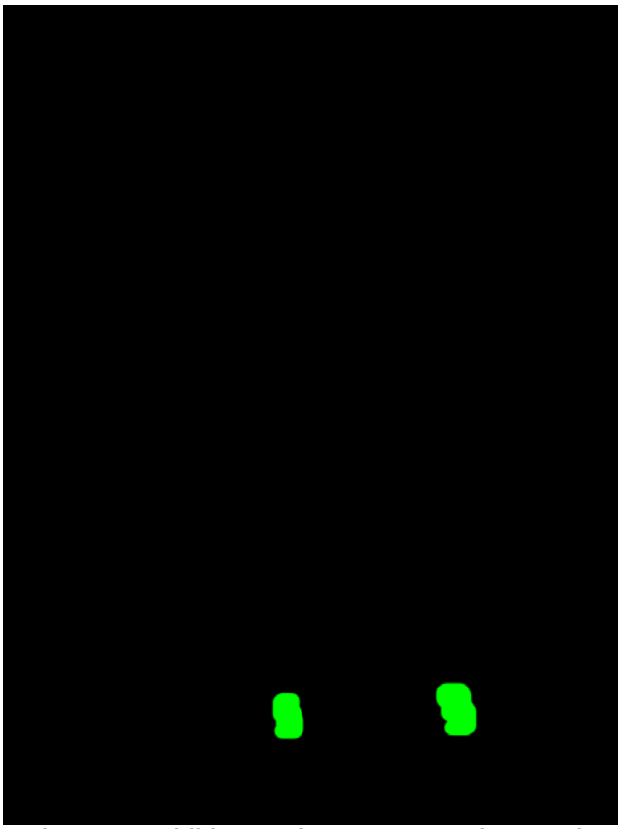

Figure 11: Additive mask to protect Venice Beach stumps

Figure 12: Venice Beach resized with mask protecting stumps

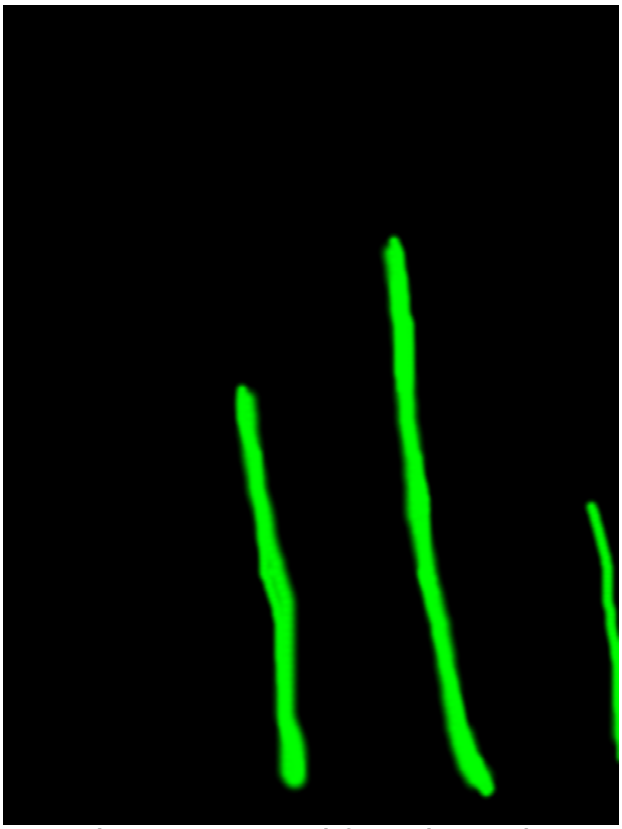

Figure 13: Large mask for Venice Beach

Figure 14: Venice Beach resized with large mask

Figure 15: Matched mask for Venice Beach (green is additive, red is subtractive)

Figure 16: Venice Beach resized with matched mask

Object removal is demonstrated in figures 17-21. A simple subtractive mask is painted over the surfer to be removed and the image is resized in one dimension only. In this case resizing in two dimensions causes nasty warping in place of the object (figure 19). If resizing in a second dimension is required it can be done is a second independent pass of the program. The distortion is not universal however it is based significantly on the seams selected in this image (figure 20), but in most cases one dimension will be mostly undistorted. The algorithm does a wonderful job of removing an undesirable object dead center from the image.

Figure 17: Image of Morro Rock at Big Sur

Figure 18: Big Sur reduced by 50 pixels on each axis

Figure 19: Mask for removing a surfer

Figure 19: Big Sur with surfer removed (1-axis)

Figure 20: Big Sur with surfer removed (2-axes)

Figure 21: Selection of seams that causes distortion

FUTURE DEVELOPMENT

There are several changes to this program that can be made to improve upon it. Image upsizing can be implemented as seen in the original paper [1]. The code can be optimized by rewriting functions dedicated to horizontal seams, using scipy.weave [9], and having an option to turn off drawing seams. A front end to call the program from The GIMP, a popular free image editor [10], can be added. Most importantly, matched mask generation can be automated.

SUMMARY

The techniques developed at M.E.R.L. [1] for image retargeting seam to work quite well for simple resizing and object removal. Overall the technique is quite sound. The addition of matched masks seems to help out considerably in tricky cases. There are many other things that can be explored based on this including video resizers (as proposed by the original authors [1]) and new energy functions.

ACKNOWLEDGMENTS

The photographs of the California coast included are public domain from pdphoto.org. Specifically:

- http://www.pdphoto.org/PictureDetail.php?mat=pdef&pg=5101
- http://www.pdphoto.org/PictureDetail.php?mat=pdef&pg=8165

APPENDIX – CODE LISTING

Seamcarve.py: The image resizer written in python. Requires Numpy [7] and PIL [8].

REFERENCES

[1] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing." *ACM SIGGRAPH 2007 Papers* (San Diego, California, August 05 - 09, 2007). SIGGRAPH '07. ACM, New York, NY, 10. URL: http://doi.acm.org/10.1145/1275808.1276390

[2] H. Yee, "Seam Carving for Image Resizing - My Quick and Dirty Implementation," *Hectorgon - Graphics, Books and Technology.* URL : http://hectorgon.blogspot.com/2007/08/seam-carving-my-quick-and-dirty.html

[3] M. Klingemann, "Optimizing Seam Carving," *Quasimondo - Mario Klingemann's Flash Blog*. URL: http://www.quasimondo.com/archives/000652.php

[4] J. Ebert, "Content-aware image resizing," *blog.je2050.de - blog and database of joa ebert.* URL: http://blog.je2050.de/2007/09/02/content-aware-image-resizing/

[5] S. Ramin, "Liquid Resize." URL: http://www.thegedanken.com/retarget/

[6] Rein-Lien Hsu; M. Abdel-Mottaleb; and A.K. Jain, "Face detection in color images," *Transactions on Pattern Analysis and Machine Intelligence*, vol.24, no.5, pp.696-706, May 2002. URL: http://ieeexplore.ieee.org/iel5/34/21601/01000242.pdf?isnumber=21601&prod=STD&arnumber=1000242&arnumber=1000242&arSt=696&ared=706&arAuthor=Rein-Lien+Hsu%3B+Abdel-Mottaleb%2C+M.%3B+Jain%2C+A.K.

[7] "Numpy Home Page."
URL: http://numpy.scipy.org
[8] "Python Imaging Library (PIL)."
URL: http://www.pythonware.com/products/pil/
[9] "PerformancePython."
URL: http://www.scipy.org/PerformancePython
[10] "The Gimp."
URL: GIMP - The GNU Image Manipulation Program