Patch-Based Synthesis for Single-Image Hole-Filling and Image Retargeting

SHAUN CHEN, University of California - Santa Barbara

When taking photographs in everyday life, people often capture undesirable traits in their image. Whether it's ugly unwanted obstruction in the background such as an ugly structure or large areas of blank space, these uncontrollable factors are typically not desired in an image. Computational photography has drastically improved over the years to address these issues. Patch-based synthesis are among the many methods to improve an image in post-production.

General Terms: photography, patch-based synthesis, hole-filling, image retargeting

1 INTRODUCTION

Single-image hole-filling and image retargeting are two methods of patch-based synthesis. However, on its own, these two methods are not sufficient for creating good results. Particularly, being content-aware using bidirectional similarity [1] measures is extremely important to preserve key features while eliminating excess pixels.

Single-image hole-filling is the process of filling a patch of empty pixels using the rest of the same image. Historically, this was a very time-consuming process, but with the use of a PatchMatch algorithm [2,3] that incorporates random search, the result was a drastically quicker runtime.

Image retargeting is makes sure undesirable "boring" pixels are eliminated and key features are preserved in a smaller frame. Again, content-awareness is important so that key features are kept when downsizing an image.

In this report, I go through my workflow and discuss my results and findings for the respective patch-based synthesis methods.

2 SINGLE-IMAGE HOLE-FILLING

The motivation for single-image hole-filling comes from images with distractions that affect the overall appearance.

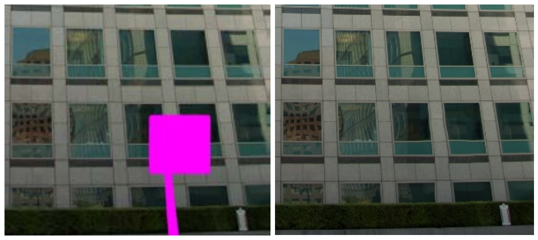
Figure 1: A street sign was hole-filled by using its surroundings to patch match

2.1 Computational Overview

This method spans three main steps: discovering the hole, downsampling the image, and then recovering it recursively while performing patch search-and-vote.

Hole Discovery. Generally, inputted image has 3 channels, RGB, with the exception of PNGs that offer 4 channels, RGBA. For the purpose of hole discovery, the alpha channel is important as it lets the PatchMatch algorithm [2,3] know which respective areas of the image are the target and the remaining areas are the source.

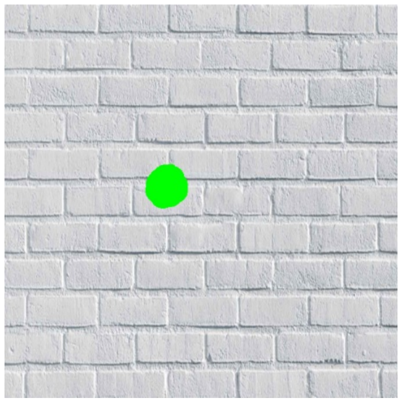
Figure 2: Hole is discovered as the green patch

Image Downsampling. The original image with the hole is then downsampled until the target patch matches the desired size. The desired size can vary between the factors of accuracy and speed. In this report, a patch-size of 11x11 pixels was used for a fair medium.

Patch Search-and-Vote and Recursive Recovery. Inputting the target patch, it is compared to the rest of the image, now set as the source patch. In this step, the PatchMatch algorithm [2,3] does a random search on the source image and computes the relative coherence. The lowest value becomes and replaces the target patch. While upsampling and interpolating the target patch for comparison to the next upsampled source image, the patch search-and-vote recursively continues until the original dimensions are recovered.

2.2 Performance

While not limited, this algorithm performs extremely well for images with target patches that are pattern-based due to the algorithm's strength for finding a matching patch. As shown by Figure 1, the PatchMatch algorithm [2,3] can even account for rotation and scale for the best fit.

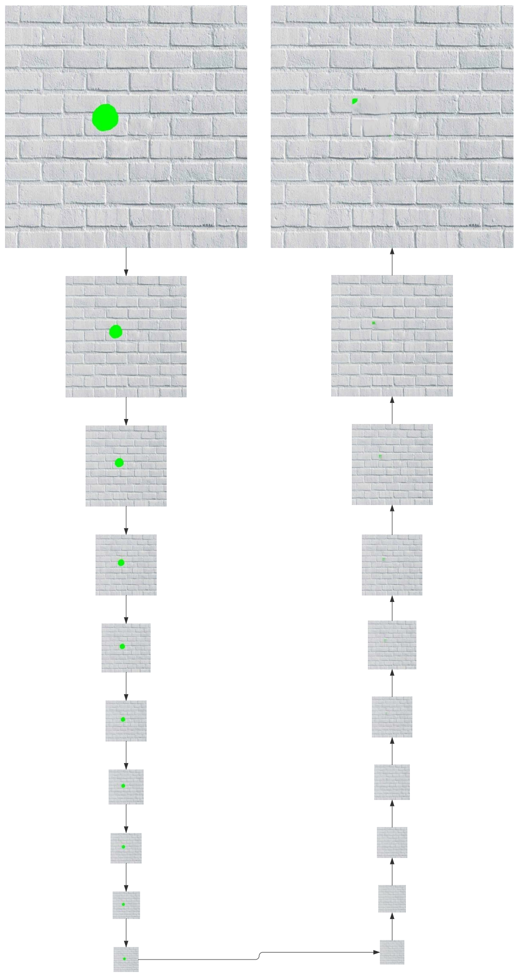
Figure 3: Step-by-step demonstration of Single-Image Hole-Filling Method

2.2 Observations

As you can from Figure 2, my hole-filling was not perfect due to my lack of an alpha value check in my code implementation, but besides that, the results were still significant.

3 IMAGE RETARGETING

While cropping and scaling are quick and simple, it fails the maintain the integrity of all of the key features in an image. Thus, image retargeting has many purposes, including image summarization, rescaling for different screen sizes, and thumbnail formation.

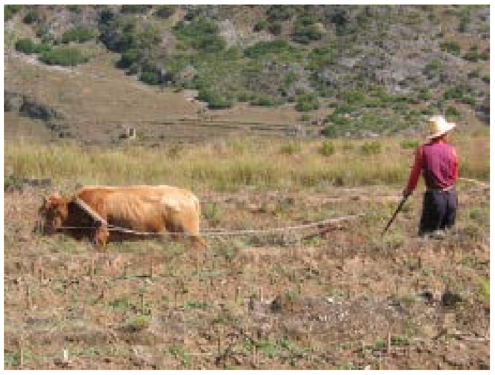
Figure 4: Original Source Image

3.1 Computational Overview

This method contains two main steps: downscaling and retargeting, and gradual resolution refinement.

Downscaling and Retargeting. In order to optimize computational performance, the image must be downscaled first. At the lowest resolution, the desired image proportions are set before performing patch search-and-vote to resize and thus retarget the image.
Gradual Resolution Refinement. Taking the coarsest image, the resolution is gradually recovered with the previous target patch and repeated across multiple iterations before the final resolution is recovered.

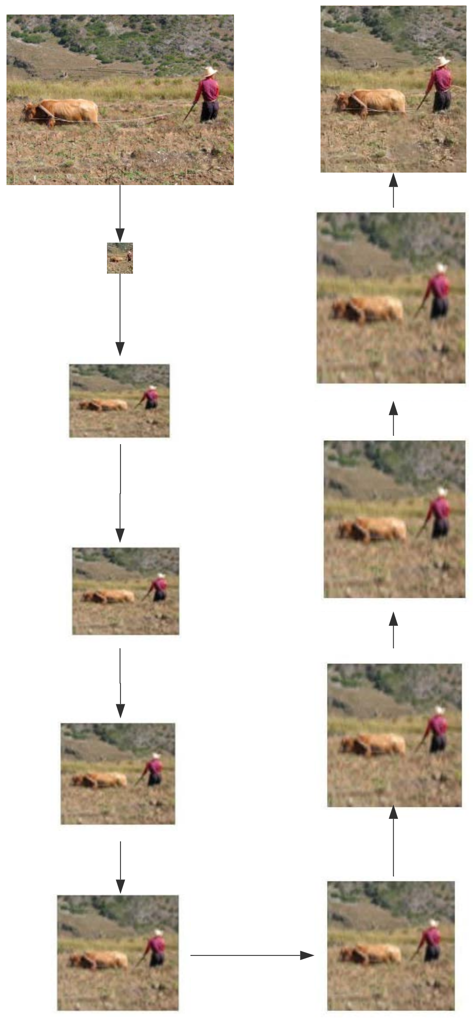
Figure 5: Step-by-step demonstration of Image Retargeting Method

3.2 Algorithm Performance

Seam-Carving vs. Image Retargeting

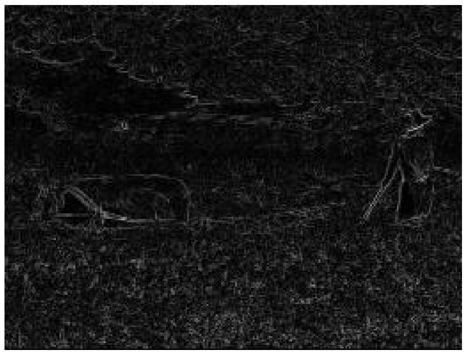
Figure 6: Energy map used for seam-carving

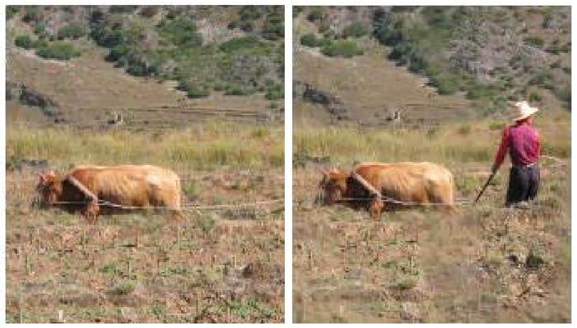
Figure 7: Comparison between the results of seam-carving (left) and image retargeting (right)

In the image comparison above, image retargeting preserved while seam-carving completely eliminated the farmer, a key feature of the overall image. Both images have the same resulting dimensions.

3.3 Observations

The image targeting method ended up working extremely well, with zero noticeable discontinuities in the image.

4 CONCLUSIONS

Both single-image hole-filling and image retargeting proved to be extremely powerful forms of patch-based synthesis. They each handle a different aspect of computational photography; one fills in undesirable obstructions while the other maintains the integrity in the case of resizing.

Although I wasn't able to fully implement hole-filling due to time constraints, the overall algorithm still worked really well.

The trickiest part of this assignment was the various ways each variable was stored as. For example, the NNF had x and y variables stored in separate channels, but that was confusing relative the RGB pixel values stored in separate channels.

Unfortunately, I didn't have time to modify the weights during the voting process, although that would have definitely helped improve the coherence of the reconstructed result.

One significant tweak I made was to store the alpha values in hole-filling as the fourth channel of the image. Then, I was able to use the "imresize()" function on all four channels and keep them consistent. It allowed randoms search to be more efficient by eliminating the target patch in the source image from being searched.

ACKNOWLEDGEMENTS

This work was supported by Professor Pradeep Sen and the TAs of the ECE178 course at University of California - Santa Barbara during the Fall Quarter of 2021.

REFERENCES

[1] Simakov et al. Summarizing Visual Data Using Bidirectional Similarity. 2008.

[2] Barnes et al. PatchMatch. 2009.

[3] Barnes et al. Generalized PatchMatch. 2010.