MARKOV RANDOM FIELD-BASED INPAINTING WITH SUPERPIXEL-DERIVED BOUNDARY CONDITIONS

Stephen Xu, Sina Farsiu, PhD.

Department of Biomedical Engineering, Duke University

ABSTRACT

In this work, we explore the work done by Cheng et al. on image inpainting which uses a global optimization of a Markov Random Field in order to fill in a missing region [1]. Improvements are subsequently made on the algorithm with the inclusion of superpixels, which has the potential to both preserve structure and decrease computational cost. Results show that the proposed algorithm performs better than that of Cheng et al. for images that have regions with repetitive information, but not for regions with unique features.

1. INTRODUCTION

Image inpainting is the process of taking an image, masking it with a region to remove, and then somehow filling it in. It is is widely used in the media industry with respect to editing, especially in well-known applications such as Adobe Photoshop. Much of the research being done revolves around the following two aspects: structure and speed. In this project, a specific implementation of image inpainting by Cheng et al. is explored. Additionally, a new method of inpainting using superpixels is considered in parallel.

1.1. Previous Works

Image painting methods fall into one of four categories: diffusion-based methods, sparse-based methods, exemplar-based methods, and deep-learning based methods. In diffusion-based methods, the region is modeled using partial differential equations to infer the diffusion of pixels. One of the pitfalls of this method is that it does not preserve the structure of the image well. Sparsity-based methods are particularly good for filling in degraded images rather than filling in missing regions by using wavelet transforms or iterative energy constraint optimizations. In exemplar-based methods, a region from the known image is composited into the unknown region [2]. This method can further be split into matching-based algorithms and Markov Random Field (MRF)-based algorithms. There are several pitfalls with the former which include exponential run-time complexity as the size of the missing region increases and additional computations to find where each patch goes. In addition, there is no computational photography topic that is untouched by machine learning (ML) and image inpainting is no exception. ML-based methods generally take the form of Generative Adversarial Networks (GANs) that have a discriminator trained to minimize the ability to detect if the region is below a certain error criteria. The pitfalls of this method are the enormous computational power needed to train the network as well as the need for a large and reliable data set.

2. METHODS

Cheng et al. proposed a MRF-based approach that seeked to ameliorate some of the mentioned pitfalls of other methods. They stated that the structure of an inpainting can be preserved if statistics of patch offsets are used. Afterwards, to speed up the process, they opted to use a global optimization using graph-cuts, both of which are similar to the method proposed by He et al. [3]. Both works used a single image with a missing region denoted Ω with boundary $\partial\Omega$. The statistics from the PatchMatch algorithm is was used to create an initial labeling map [4]. The difference between the two lies, as well as in the proposed method, lies in the pre-processing detailed in the following sections.

2.1. Structural Transforms and Feature Selection

2.1.1. Curvelet Transforms

To preserve structure, Cheng et al. selected components of a Curvelet transform that would provide general structural information. The Curvelet transform is a multi-scale directional transform developed by Ma et al. that is similar, but more descriptive than a traditional Wavelet transform [5]. Where a wavelet transform, such as a Haar transform, takes the gradient in the horizontal, vertical, and diagonal directions at different scales, the Curvelet transform breaks down an image's structure by the curvature of its progressively higher-frequency data. A non-degraded image I is fed into the Curvelet transform and is split into eight regions of coefficients which represent different features of directionality. As shown in Fig. 1, only the coefficients Z from the second to fifth layers were considered, which can be interpreted as only taking the finer, higher-frequency structures and edges of the image, specifically in the $0°$, $45°$, $90°$, and $135°$ di-

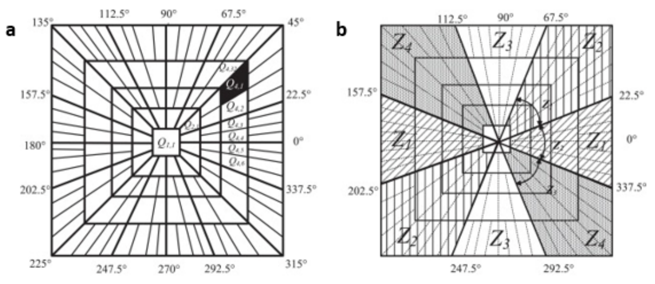

Fig. 1. a) Curvelet transform coefficient structure. b) Components of the Curvelet transform used in inpainting [1].

rections for $n = 1, 2, 3, 4$, respectively. This process results in four separate representations of the same image, denoted H_n where $n \in \{1, 2, 3, 4\}$. The inverse Curvelet transform is taken for each H_n and represented by the term A_n, where $n \in \{1, 2, 3, 4\}$. Beginning from the stock `cameraman` image, the resulting four direction feature images look similar to those in Appendix Fig. 10. Note some interesting characteristics of this transform: the $45°$ and $135°$ representations highlight the legs of the tripod as well as the general shape of the cameraman.

The images in set A_n are then selected based on the variance their direction gradient magnitude, denoted I^n_{gs}. The edge features on boundary $\partial\Omega$, denoted by $I^n_{gs\_d}$ where $n \in \{1, 2, 3, 4\}$, is defined by multiplying a dilated mask boundary with I^n_{gs}. The larger variances of each of the four representations in $I^n_{gs\_d}$ are considered. A greater variance can be interpreted as a greater number of pertinent features. similar to how edges in images have greater variance than a smooth region. This threshold variance was selected manually.

2.1.2. *Superpixels*

Super pixels were examined as an extension to the original work done by Cheng et al. Super pixels are another form of structural information extraction as detailed in the work done by Achanta et al. called Simple Linear Iterative Clustering (SLIC) [6]. It is able to capture an image's redundancies, provide a convenient primitive with which to compute image features, and generally increases the speed of image processing algorithms by reducing pixel number through an adaptation of the K-means clustering algorithm. The algorithm is controlled by a single parameter K which represents the number of roughly equally-sized clusters in I. Formally, determination of superpixels is executed by the optimization of 5-D Euclidian distance of a pixel in the image D in CIELAB colorspace.

$$D = \sqrt{d_c^2 + (\frac{d_s}{S})^2 m^2}, \tag{1}$$

where S is a sampling interval, and d_c and d_s are the normalized cluster distances in a CIELAB-color and spatial sense, respectively defined by,

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}, \tag{2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{3}$$

With this, each pixel from the original image is assigned to the nearest cluster centroid whose region overlaps its location. The result of the algorithm gives a segmentation map, but the structural information can be used for inpainting. The output of `cameraman.tif`, after using the super-pixel algorithm is shown in Fig. 2. Note that for this particular image of size 256-by-256 with a total of 256^2 pixels, the total number of pixels for subsequent computations is reduced to at most 178. Other images exhibit similar significant decreases in pixel density in the new representation. It is possible to take advantage of this representation in several ways. One is that it can be used to give more accurate boundary information for the search region in the PatchMatch algorithm. Alternatively, it can be used as a heuristic for finding other similar regions based solely on the mean value through out the entire image, as opposed to a small bounded region around Ω. In this work, the former is considered.

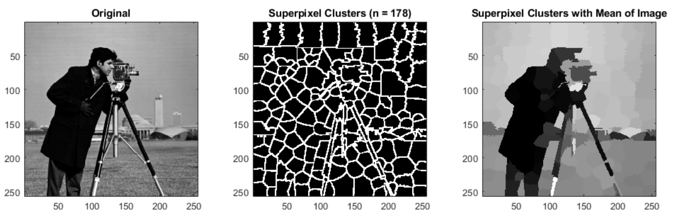
Original
Superpixel Clusters (n = 178)
Superpixel Clusters with Mean of Image

Fig. 2. Results of using superpixels. Note that the original `cameraman.tif` of size 256-by-256, has been reduced to a total of 178 superpixels.

2.2. PatchMatch

Before any optimization is done, the candidate offsets or initial conditions must be selected. This step is done by PatchMatch, an algorithm which finds the nearest pixel offset $\mathbf{x} + \mathbf{s}$ of an N-by-N patch similar to the patch at $\mathbf{x}$; this is referred to as a nearest-neighbor field (NNF). In this implementation $N = 9$.

The original algorithm is split into three distinct processes: initialization, propagation, and searching. Initialization finds the NNF by assigning random values or using a prior which aids in making the optimization convex. Barnes et al. used an iterative approach to find the minimum least-squared error solution between pixels. The second step was propagation which took advantage of a previous pixel's neighbors' offsets when determining an offset for the current pixel $\mathbf{x}$. The final step was a random search which creates a search window of some radius r, attempts to find a pixel that is most similar, decreases the window size, and continues until the window size is smaller than 1 pixel [4].

Much progress has been made on this specific algorithm in terms of speed. He et al. developed a Propagation-Assisted KD-tree method which reduced the optimization to logarithmic time complexity [7]. In the algorithm, they take advantage of the distribution of the patches in target image B and in the source image A. The algorithm is fast because the tree structure only checks a small number of candidates during propagation and has no backtracking step. Formally, the optimization is,

$$S_n(x) = \underset{\mathbf{s}}{argmin}||p(\mathbf{x}+\mathbf{s}) + p(\mathbf{x})||^2, s.t.|\mathbf{x}| > \tau \qquad (4)$$

where for each patch $p(\mathbf{x})$, there exists a $\mathbf{s} = (u, v)$ or offset and a τ which is the threshold for the minimum distance allowed from the current pixel to the respective patch. This specific version of PatchMatch was used.[1]

In Cheng et al., the selected directional feature images I^n_{gs} are binarized into I^n_d and combined with the image, i.e. $I^n_s = I \times I^n_d$. An additional feature image, I_{ns}, is used as a consideration for all features, which is represented by the union of all selected directional features images in I^n_s. The respective selected features I^n_s and I_{ns} are fed through the PatchMatch algorithm to obtain the initial offsets. For the superpixel implementation, the same PatchMatch optimization function was used, but the region was bounded by the edges of the superpixels that overlap with Ω.The intuition behind this step stems from the fact that because super pixel clustering preserve structure, using it as a guideline should increase the accuracy of the initial offsets and thus, improve the resulting inpainting. An example is shown in Fig. 3. The result of this process is S_n which is a set of up to five offset maps.

2.3. Determining Statistics

To finalize the initial labels, Cheng et al. adopts a statistical method of selecting which offsets are most important; this is responsible for decreasing the overall computational cost. Here, a K_1 offsets correspond to the individual variance images I^n_s and K_2 correspond to the structured union image. I_{ns}. After PatchMatching, the offsets S_n are examined in a histogram, such as the one shown in Appendix Fig. 12, and formalized using,

$$h(u, v) = \sum_{\mathbf{x}} \delta(S_n(\mathbf{x} = (u, v))) \qquad (5)$$

The top $K_1 + K_2$ are considered the initial labelings $L(\mathbf{x})$ for Ω.

[1]Details are in Appendix 6.1.1.

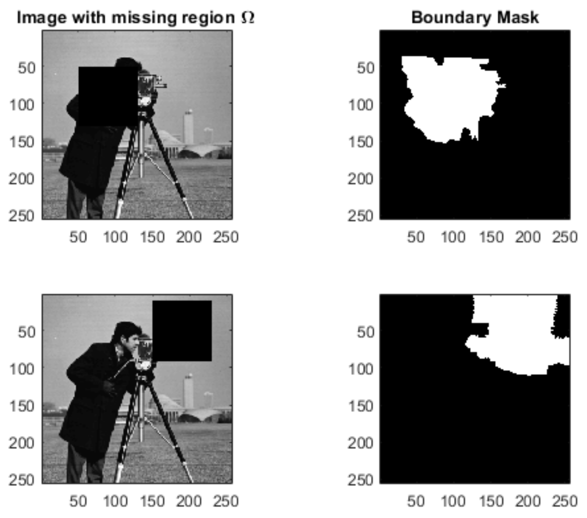

Fig. 3. Boundary of consideration for the PatchMatch algorithm (left) with superpixel boundary masks (right).

2.4. Markov Random Fields

Markov Random Fields (MRFs) represent images with a graph. Typically, an image is thought of as one or many matrices of discrete values, depending on the number of channels. However, in the case of MRFs, each pixel is represented by two nodes: a site and a label. The site node can be interpreted as the location of a certain pixel, 4-connected with its north, south, east, and west neighbors. The label node can be interpreted as the value that the site holds; this value can range from intensities, colors, or in this case, an associated offset. The most important aspect of this is the ability to stoichastically represent the relationship between the sites with other sites as well as the sites with its respective label nodes. A visualization of a 3-by-3 image as an MRF is shown in Fig. 4.

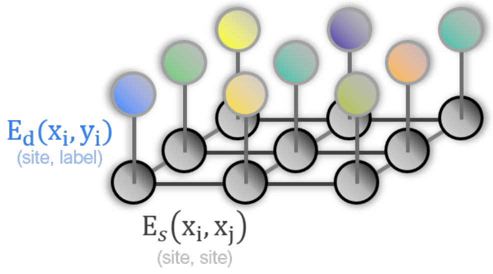

Fig. 4. Visualization of a Markov Random Field.

MRFs provide a descriptive framework for inpainting optimization problems because the known patch and the missing patch Ω are respectively regarded as a label and a node. Overall, for every pair of nodes, the stochastic association contributes to an overall energy function. In image inpainting, the energy function is denoted as follows,

$$E(L) = \sum_{\mathbf{x}\in\Omega} E_d(L(\mathbf{x})) + \sum_{\mathbf{x},\mathbf{x}'\in\Omega} E_s(L(\mathbf{x}), L(\mathbf{x}')), \quad (6)$$

where E_d can be interpreted as a validity term and E_s can be interpreted as a smoothness penalization [1, 3, 8]. More formally, the validity energy associated with the label at pixel $\mathbf{x}$, $L(\mathbf{x})$, is $+\infty$ if the shifted pixel is in Ω or outside of image I, and 0 if the shifted pixel is in a known region. The smoothness energy, E_s, is formally written as,

$$\begin{aligned} E_s(a, b) = ||I(\mathbf{x} + \mathbf{s}_a) - I(\mathbf{x} + \mathbf{s}_b)||^2 + \\ ||I(\mathbf{x}' + \mathbf{s}_a) - I(\mathbf{x}' + \mathbf{s}_b)||^2, \end{aligned} \quad (7)$$

where $L(\mathbf{x})$ and $L(\mathbf{x}')$ are the labelings for the current and neighboring pixel. and a and b are the offsets $\mathbf{s}$ of the current and neighboring pixels, respectively. Note that when the value of the shift of the current pixel differs greatly from that of its neighbor, the energy or penalization is relatively large. Conversely, the relationship is conducive to smoothness since similar shifted areas have lower energies. Thus, the optimization becomes a problem of assigning a suitable label to a node or minimizing the energy associated with the region Ω.

2.4.1. *Graph Cut Optimization*

Most MRF minimization problems can be simplified using a representation of two labels nodes: a source (s) and a sink (t), as shown in Appendix Fig. 7. To reiterate, the site nodes are connected to other site nodes via Ψ links (whose energy is defined by E_d) and to label nodes (s and t) via Φ links (whose energy is defined by E_s).

The lowest energy is defined as the minimum cut through the graph's edges that amasses the least cost, where cost is defined as the accumulation of the energies that were set to zero. In order to find this, the maximum flow is found from the source s to the sink t. This operation separates the nodes into two disjointed sets $\{S, T\}$ which correspond to a minimum cut, making min-cut and max-flow exactly the same [9].

As an overview, the algorithm proceeds in the following steps: a "growth" stage where the graph is searched from the s and t label nodes until two active nodes meet resulting in a direct path from s to t, an "augmentation" stage where the search tree is broken into forests, and an "adoption" stage where S and T are restored to be reused. In order to accommodate for multiple labels, $K_1 + K_2$ in this case, it is necessary to perform an operation called α-expansion or β-swap whose implementation can also be found in the original work. Generally, α-expansion is used, but requires the graph to be metric [9].

3. RESULTS AND DISCUSSION

The proposed algorithm outperformed Cheng et al.'s algorithm in several, but not all cases in terms of qualitative inspection, PSNR, and SSIM. Run-time was also examined, however the fluctuations in CPU processing latency made the quantification difficult; qualitatively, there was no significant differences.[2] The proposed algorithm tends to out-perform that of Cheng et al. when the missing region contains repetitive features. Fig. 5 shows different regions of the `cameraman` image removed and inpainted. The algorithm performed fairly well in these cases because the features in the sky were bounded by the superpixel-derived mask. Even when the superpixel mask were not used, the PatchMatch algorithm was able to find similar boundary features to propagate into Ω.

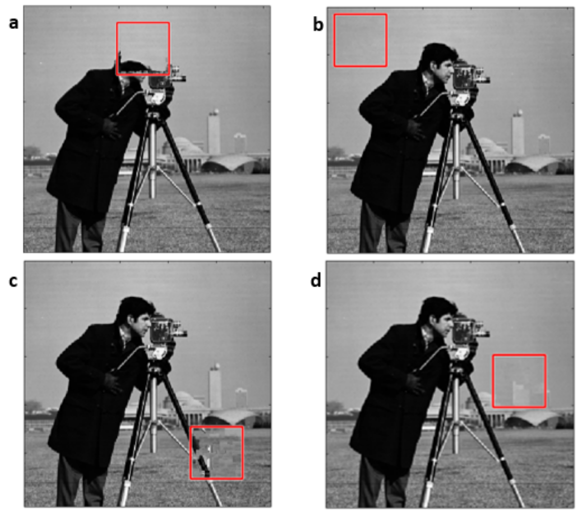

Fig. 5. Stock `cameraman` image with different regions inpainted (outlined in red).

In Fig. 6, multiple images images are shown along with the respective PSNR and SSIM values in Tables 1 and 2. Again, the proposed algorithm using superpixel-derived boundaries seems to out-perform Cheng et al.'s algorithm when there are repetitive features. For Image 1, both algorithms generate an image that is nearly identical to the original; however, there are small seam imperfections in the non-superpixel method. Realistically, images are almost never tessellations, and in the case of Image 2, the borders formed by the colors did not propagate evenly. In the case of Image 5, although edges of the boundary are more visually appealing in the proposed algorithm, both algorithms failed

[2]It should be noted that because the goal is subjectively visual, PSNR is not a very relevant metric; however, it is shown for literature convention. SSIM is a more relevant metric since it quantifies the structural similarity, but the goal of inpainting is to remove what is in the mask.

Table 1. PSNR of inpainted images

	1	2	3	4	5
Cheng's	85.807	**54.633**	65.434	58.395	(64.949)
Proposed	**85.849**	54.502	**68.276**	**60.226**	(69.065)

Table 2. SSIM of inpainted images

	1	2	3	4	5
Cheng's	0.9490	**0.9069**	0.9558	0.8640	(0.9553)
Proposed	**0.9517**	0.9063	**0.9792**	**0.9042**	(0.9924)

dramatically when attempting to inpaint the region containing the woman's face. Image 5c and 5d show the face as a result of not being able to optimize to a region outside of the mask, which is why the SSIM values are relatively large. Note that for all reconstructions and metrics, no Poisson-blending was used, which is a standard in the literature.

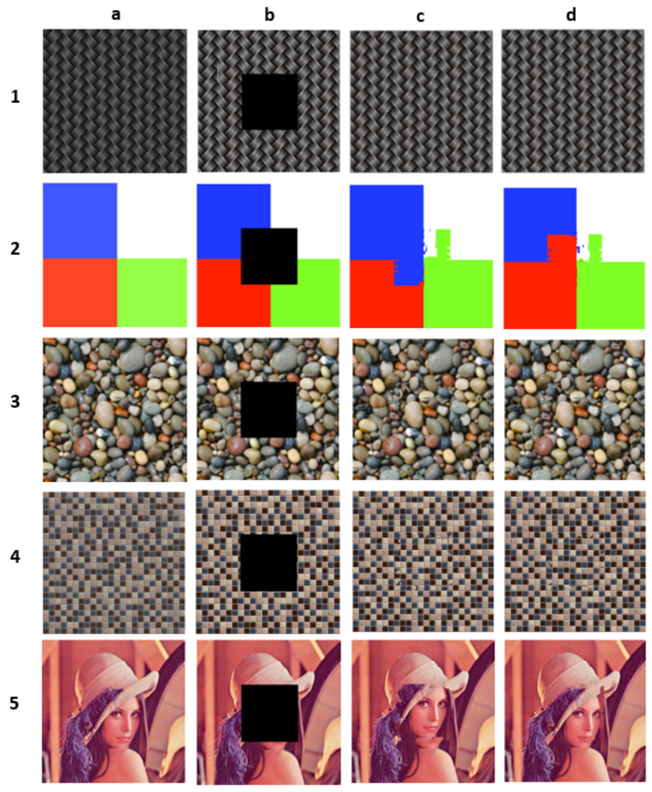

Fig. 6. Results from inpainting. a) Original Image; b) Masked Image c) Cheng et al. d) Proposed.

4. CONCLUSION

Including superpixels as a guidance for structure in MRF-based inpainting shows potential in improving existing techniques. Already, when utilizing its preservations in structure in boundary conditions, it outperforms existing algorithms in both visual and SSIM sense for regions with patterns. The algorithm performs poorly when Ω's edges contain unique information.

In future works, it would be worth-while to investigate the results the proposed method with another graph-based approach which can better take advantage of the properties of superpixels.

5. REFERENCES

[1] J. Cheng and Z. Li, Markov random field-based image inpainting with direction structure distribution analysis for maintaining structure coherence, *Signal Processing*, vol. 154, pp. 182-197, 2019.

[2] A. Criminisi, P. Perez, and K. Toyama, Region Filling and Object Removal by Exemplar-Based Image Inpainting, *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200-1212, 2004.

[3] K. He and J. Sun, Statistics of Patch Offsets for Image Completion, *Computer Vision ECCV 2012 Lecture Notes in Computer Science*, pp. 16-29, 2012.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, PatchMatch, *ACM SIGGRAPH 2009 papers on - SIGGRAPH 09*, 2009.

[5] J. Ma and G. Plonka, The Curvelet Transform, *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 118-133, 2010.

[6] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk, SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274-2282, 2012.

[7] K. He and J. Sun, Computing nearest-neighbor fields via Propagation-Assisted KD-Trees, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[8] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, Interactive digital photomontage, *ACM SIGGRAPH 2004 Papers on - SIGGRAPH* 04, 2004.

[9] Y. Boykov, O. Veksler, and R. Zabih, Fast approximate energy minimization via graph cuts, *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999.

6. APPENDIX

6.1. Details for implementation and running

6.1.1. *Compiling the PatchMatch* `nnmex.m` *file*

This uses the original implementation which requires a 2013 C++ compiler to obtain the `mex` file due to a deprecated `hash_set` library. As a fair warning, the process to obtain this is extremely tedious since the 2017 C++ compiler is the current default at the time of writing this paper. Other versions of the PatchMatch algorithm exist online (on GitHub), including that from user `ikuwow`, however, only the original implementation allows for the crucial τ term in the PatchMatch optimization in the form of a binary mask. Therefore, this must be done to obtain proper results.

1. Download Visual C++ Compiler 2015 or earlier (need to join Visual Studio Dev Essentials)
2. Run command `mex -setup C++` in Matlab
3. Choose the Visual C++ compiler
4. Copy and run the mex commands from `build_windows.bat` in patch match folder
5. This will give errors saying that min and max identifiers are not found. Go to the files listed (`vecnn.h` and `simnn.h`) and add the header `#include "minmax.h"`

6.1.2. *Superpixels algorithm*

This algorithm was provided within the MATLAB Image Toolbox library. To use the command, the toolbox must be installed on the machine. Documentation is provided at `https://www.mathworks.com/help/images/ref/superpixels.html`, and the original paper can be found via [6].

6.1.3. *Graph-cut algorithm*

In order to optimize the MRF, the data cost and smoothness penalty must be defined in the form of an adjacency matrix across the missing region Ω. The prewritten multi-label graph-cut implementation by Olga Veksler and Andrew Delong is available at `https://github.com/nsubtil/gco-v3.0`. The unitary or data matrix was defined such that each of K labels was associated with $M \times N$ sites, where M and N are the height and width, respectively. More information can be found in [9].

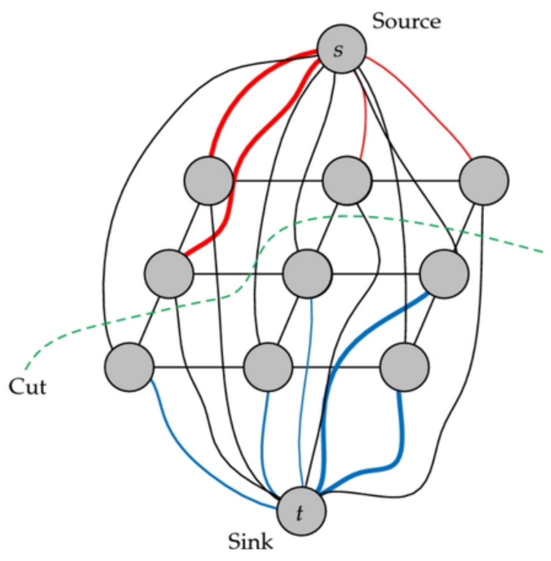

Fig. 7. Representation of a graph cut through a 3-by-3 grid [9].

6.1.4. *Executing the algorithm*

The main algorithm is in a file called `mrf_image_inpainting_w_direction_structure_dist_analysis_color.m`. Once all the files and executables are available, find a 256×256 `png` image and put it in the `images folder`. Then, set the `hole` variable (Ω) according to the starting row number, starting column number, the width, and the height, respectively. The size limitation is due to a segmentation fault in the PatchMatch algorithm. Click the `Run` button and allow for the intermediates (refer to 6.2.1) to pop up. Two output images will appear in a folder called `output`: the masked input and the inpainted image.

6.2. Supplementary Materials

6.2.1. *Intermediate results from Algorithm.*

Cheng et al. made implementation and replication fairly difficult due to the sparsity of information. This is why it is necessary to document every single aspect done in this work.

When the code is run, there are quite a few figures that will pop up. During the very, very painful implementation process, these figures primarily for debugging. If something goes wrong, here is a run-down on what each figure means.

At the point of Fig. 9 in the implementation, the Curvelet transform has been taken and the coefficients have been separated into different components for A_n. There are a total of four representations that will pop up in the same figure, one for each of the transforms.

In order to obtain more information on the edges in which

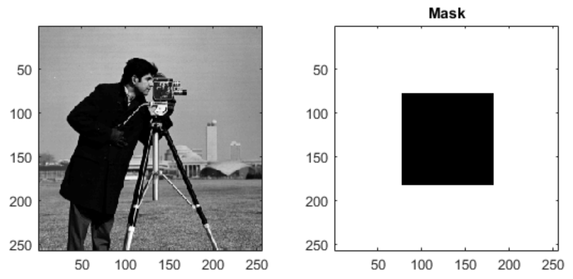

Fig. 8. (left) The input image. (right) The mask with Ω as 0.

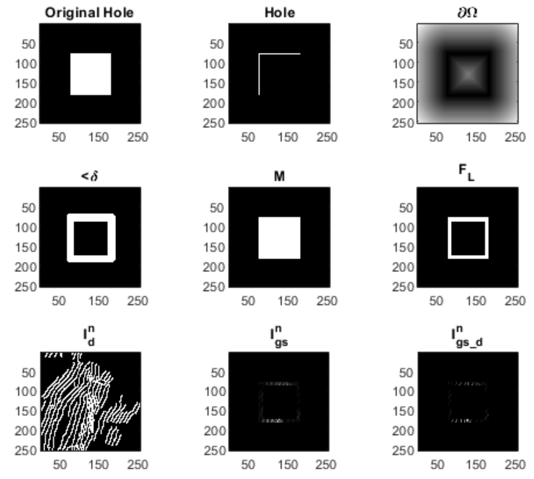

Fig. 9. Components of the edge information that PatchMatch will use, as detailed in [1].

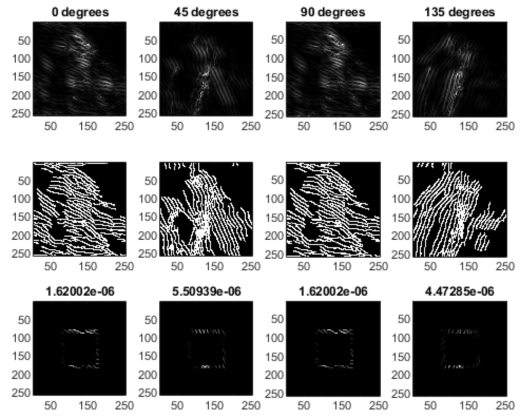

Fig. 10. Individual components for determining variance of Curvelet transform.

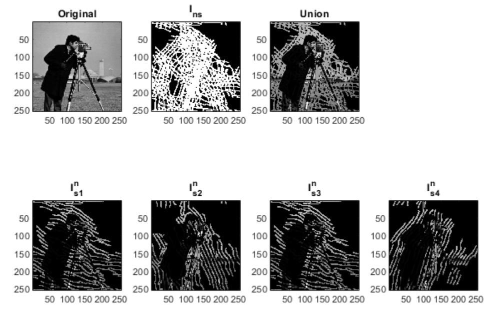

Fig. 11. Transformations into feature images

to inpaint, a term δ is essentially used as a dilation factor for a border mask. After all of the direction values are obtained and masked with the border mask, we arrive at Fig. 10

The PatchMatch algorithm is then performed on each of the selected structural variance components, including I_{ns}. Sometimes, if the variance is not significant enough, the algorithm will resort again to I_{ns}. The statistics for certain offsets as calculated by (5) are represented as a histogram. The axis is 2-times that of the size of the image because an offset can be either direction. Zero offset is represented by the pixel at (M, N) in the plot.

The next figure that appears (Fig. 13) is the initial labeling map for the energy optimization. This is crucial in order to get a meaningful reconstruction. The left figure shows the label index from the $K_1 + K_2$ labels, and the middle and right figures show the associated offsets.

For energy optimization, a visualization of the energy map associated with the Markov Random Field is necessary to avoid mistakes. This is represented in Fig. 14. The left-most figure is a binary representation of the data cost. If the pixel in the region has any label that is valid, that pixel has a cost of 0; conversely, if there is no valid pixel, then the value is $+\infty$ as described in Section 2.2. The middle figure is the data cost separated by label, or in other words, the adjacency matrix; the left and middle figures are simply different representations of the same thing. The right-most figure is a visualization of the pairwise cost E_s adjacency matrix. It is imperative to ensure that it is not all 0.

The resulting labeling map, shown in Fig. 15, after graph cuts is represented here. It is normal to see a non-zero offset patch near the middle since it is the region to inpainting with another part of the image. The final, zoomed-in image with associated PSNR and SSIM values are shown in Fig. 16.

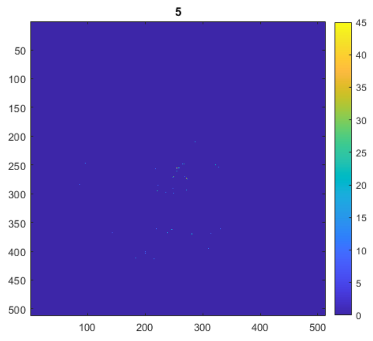

Fig. 12. Top $K1$ and $K2$ offsets as represented by a histogram.

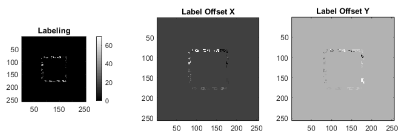

Fig. 13. Initial labeling map after selection from histogram.

6.3. Thank you

If you actually read up to here, thank you for the wonderful semester!

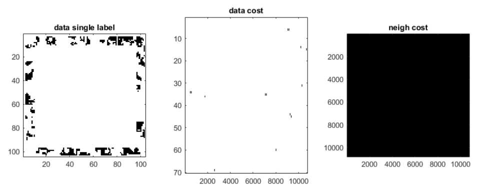

Fig. 14. Energy visualization for E_d (left, middle) and E_s (right)

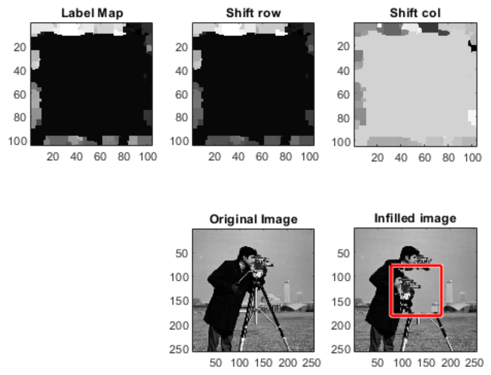

Fig. 15. Resulting labeling map and images after graph cuts.

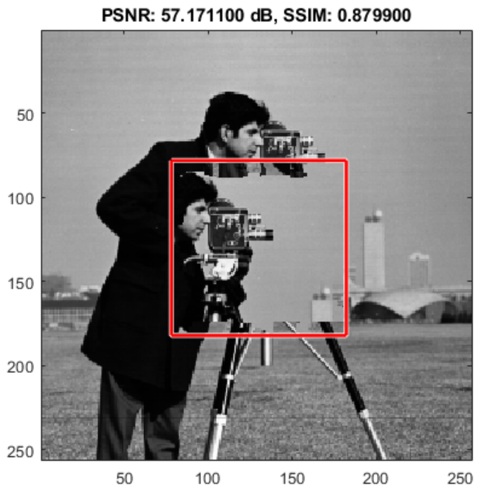

Fig. 16. Final image with associated metrics.