Improving Patch-Based Synthesis by Learning Patch Masks

Nima Khademi Kalantari[1] Eli Shechtman[2] Soheil Darabi[2] Dan B Goldman[2] Pradeep Sen[1]
[1]University of California, Santa Barbara
[2]Adobe Research, Seattle, WA

Abstract

Patch-based synthesis is a powerful framework for numerous image and video editing applications such as hole-filling, retargeting, and reshuffling. In all these applications, a patch-based objective function is optimized through a patch search-and-vote process. However, existing techniques typically use fixed-size square patches when comparing the distance between two patches in the search process. This presents a fundamental limitation for these methods, since many patches cover multiple regions that can move, occlude, or otherwise behave independently in source and target images. We address this problem by using masks to down-weight some pixels in the patch-comparison operation. The main challenge is to choose the right mask according to the content during the search-and-vote process. We show how simple user assistance can lead to excellent results in challenging hole-filling examples. In addition, we propose a fully automated solution by learning a model to predict an appropriate mask using a set of features extracted around each patch. The model is trained using a manually annotated dataset, augmented with simulated divergence from ground truth. We demonstrate that our proposed method improves over existing approaches for single- and multi-image hole-filling applications.

1. Introduction

Non-parametric patch-based synthesis methods have shown impressive results for challenging image and video editing problems. These include image-based rendering [7], hole-filling [9, 12, 24], texture synthesis [13], retargeting and reshuffling [4, 18], super-resolution [8], morphing [17], image compositing and interpolation [6], video summarization [3], and HDR reconstruction [11, 16]. The success of this approach has led to implementations in popular commercial tools such as Adobe Photoshop [1].

These methods are based on optimizing a patch-based objective function through a multi-scale patch search-and-vote process. Current patch-based methods use fixed-size square patches, but may produce poor results when a target patch covers multiple regions and the available patch examples containing the same layout of the different regions are scarce or do not exist at all. In these cases, as shown in Fig. 1 for the Single-Image Hole-Filling (SIHF) application, using only a fragment of the patch may lead to finding more relevant examples and thus better synthesis results.

This problem is also common in Multi-Image Hole-Filling (MIHF), using other photos of the same scene captured from different views and/or at different times, as can often be found in personal photo albums or web collections [6, 25]. In MIHF, all the information needed to fill the hole might be present in the example source image, but it may contain parallax between objects at different depth layers or motion between exposures, and the patches covering two objects at different depths may thus be dissimilar between the source and target images.

A similar issue occurs in related problems such as optical flow, stereo, and image matching. In optical flow estimation, this problem is usually handled by optimizing for the flow and depth layer segments simultaneously [19, 20, 21] and requires multiple inputs. Bobick and Intille [5] proposed shifted windows to handle depth discontinuities in stereo matching. Trulls *et al.* [23] take segmentation into account for computing image matches. However, in our application, segmentation cannot be computed a priori in the to be synthesized region (e.g., the "hole" in hole-filling) because its content is unknown. As far as we know, no research to date has tackled this issue for patch-based synthesis or other image editing approaches.

In this paper, we address the problem of fixed-shape patches by using content-adaptive masks. Our method chooses an appropriate mask for a patch during the search-and-vote process, and down-weights the distance function in the fragment of a patch for which the match is not valid. We show how a simple manual annotation of the boundary edges inside the hole can lead to excellent hole-filling results when masks are employed.

Furthermore, we propose a fully automated solution by learning a mask selection model that can predict the best mask for each patch in the to be synthesized region using a set of features extracted in its vicinity. Unlike typical computer vision problems where the training set is fully consistent with testing set, in image synthesis the "testing set" arises from an iterative process which does not contain only "real-world" images or patches. This is because the patch-based search-and-vote process is done repeatedly at different scales and multiple iterations to compute a single image result. Therefore, a small prediction errors in early iterations can affect the types of patches seen in later iterations. Thus, it is a challenge to construct a training set that appropriately samples the space of possible patches that may be seen at testing time in the middle of the iterations. We propose to construct the training set by extracting data from a hole-filling process using ground truth masks, augmented with data from simulations of typical iterative divergence from an optimal hole-filling process. We demonstrate the results of our method on both the SIHF and MIHF tasks. Our results show significant improvement over current patch-based techniques for complex, layered scenes.

2. Problem Explanation

We start by explaining the method for Single-Image Hole-Filling (SIHF). In this application, as shown in Fig. 1, the user defines a hole region in an image which divides it into target T (shown in pink) and source region S (the region outside the hole). The goal is to complete the missing target region using information (patches) from the source region. This is done by minimizing an objective function which has the following patch-based energy term [24],

$$\text{Coherence}(T,\ S) = \frac{1}{|T|}\sum_{Q\in T}\min_{P\in S}\text{D}(Q,\ P), \qquad (1)$$

where P and Q are patches of size $w \times w$ in the source and target images, respectively, and D is defined as the $\mathcal{L}_2$ distance between the two patches and is typically weighted equally for all pixels. This objective function is minimized by performing the "search" and "vote" process iteratively, in a coarse-to-fine fashion. In the search step, for every patch in the target image, the most similar patch in the source image is found. Then in the voting stage, for each pixel in the target, all patches found in the search stage that overlap that pixel are averaged to obtain its color for the next iteration. Starting from the coarsest scale, the search-and-vote process is performed iteratively until convergence and the converged target is upsampled and used at the next finer scale. This process is continued until convergence in the finest scale.

Fig. 1 (top row), illustrates the problem with conventional patch-based techniques. Although all the content in the target area has some plausible match in the source, the orange and red squares in the target region have no single full matching patch in the source region. This problem is more severe at coarser scales where patches are larger relative to the image, and therefore the chance of covering multiple regions is higher. Since the algorithm depends on converging to good local optima at each scale, convergence to a bad solution at coarse scales is generally irreversible at finer scales, thus leading to bad results overall as shown in Fig. 1 (top row). We address this problem by introducing masks for patches, as explained in the next section.

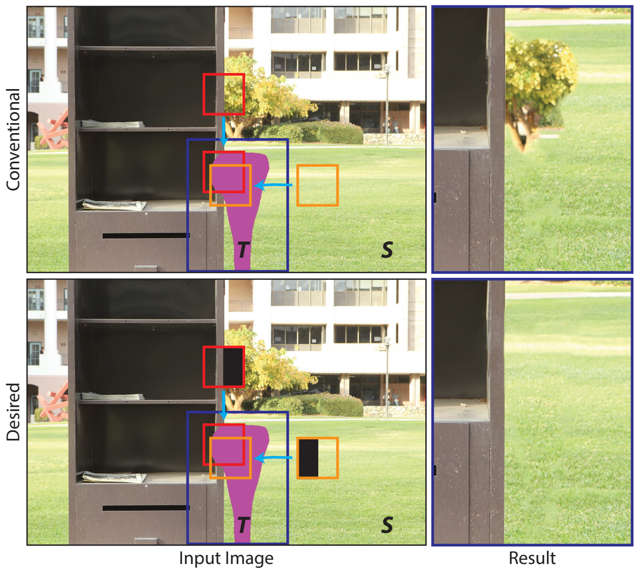

Figure 1. The problem with conventional patch-based methods. **(top)** In the standard approach, for every patch in the hole (shown in pink) a comparable full patch should be found in the source (the region outside the hole). However, some patches, such as those shown here with red and orange squares, do not have a comparable full patch. Therefore, the synthesized result is not plausible. **(bottom)** In our approach, the portion of the patch with irrelevant data is down-weighted with the masks shown to generate a plausible result.

3. Proposed algorithm

One way to address this issue is to down-weight the portion of a patch that has irrelevant content and only take into account the part that is useful for synthesis. To do this, we propose to modify the distance function D as follows:

$$\text{D}(Q,\ P) = \sum_{i=1}^{w\times w} m(i)(Q(i) - P(i))^2, \qquad (2)$$

where w is the patch width, and m is a mask that weights the distance between source and target patches at every pixel and belongs to a large, w^2-dimensional set of masks $\mathcal{M}$. The voting algorithm is modified accordingly: the average of overlapping patches becomes a weighted average using the corresponding mask values as weights.

Theoretically, this simple modification can solve the problem by taking into account only the part of each patch which is relevant to the distance computation for the search and vote processes, as shown in Fig. 1 (bottom row). However, choosing the right mask from a large set $\mathcal{M}$ for every patch is a difficult problem.

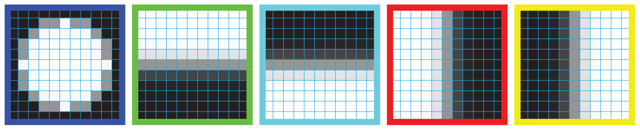

Figure 2. A set of five predefined masks $\mathcal{M} = \{m_1, \cdots, m_5\}$. Each mask is normalized so its pixel weights sum up to one. The leftmost mask m_1 is called the *center mask* and the rest are *non-center masks*. The colored border around each mask will be used to identify which mask is used at every patch in later figures.

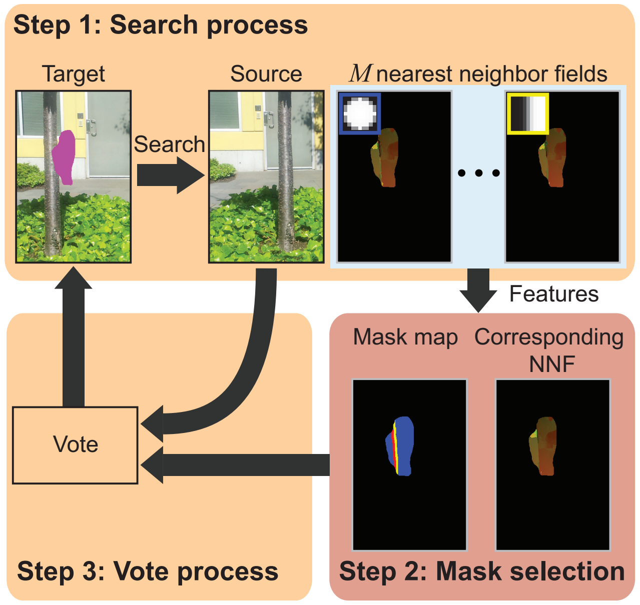

Figure 3. Our framework for the Multi-Image Hole-Filling (MIHF) application. The mask map is color-coded using the colors depicted in Fig. 2.

To make the problem more tractable, we propose to limit $\mathcal{M}$ to a small set of pre-defined masks, $m_1, \cdots, m_M$, an example of which is shown in Fig. 2. Our framework has three main steps: 1) In the search step we perform a separate patch search process for *every* mask in this set, resulting in M Nearest Neighbor Fields (NNFs). 2) Then for every patch in the target image, we select an appropriate mask and its corresponding nearest-neighbor source patch using the model described in Section 3.1. 3) Finally, we use the masks chosen in the previous step in the weighted-average voting process. An overview of our framework is shown in Fig. 3 for the MIHF application.

The main challenge in this process is how to choose an appropriate mask at every pixel (step 2). In the next section we explain our machine learning approach to this problem.

3.1. Learning the masks

Ideally, if we had semantic segmentations of source and target regions, the mask would be defined simply as the

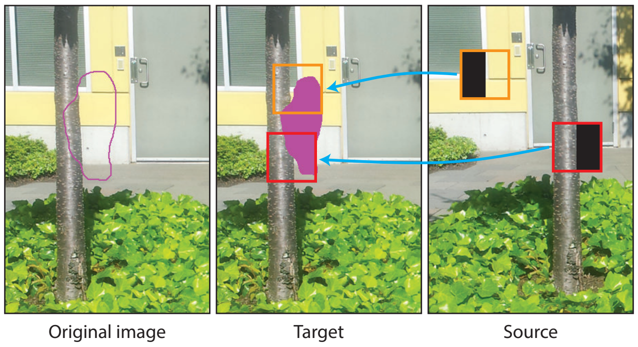

Figure 4. For a set of two images taken from different points of view, we define a hole region in one of the images. The red and orange squares show two patches with their appropriate masks. The depth discontinuity is a good clue for how to select the masks.

fragment of the patch belonging to the same region as the center pixel. For example, in the MIHF case shown in Fig. 4, the tree is moving in front of a complex background due to parallax in the two images. In this case, it would be ideal to label the tree and background as two different segments and define the masks according to that labeling. However, depth alone is not enough: in some cases, patches may have similar appearance to the foreground portion but have different backgrounds, even if the backgrounds come from the same object and have the same depth. Computing such a segmentation automatically is an ill-defined open vision problem we wish to avoid here. Moreover, the target often has pixels with *unknown* color values for which it is not clear how to compute segmentation in advance.

Our approach is to learn a model for choosing the best mask from a set of M possible masks. The prediction is done by using the target patch and the M matching source patches, each found using the corresponding NNF. Such a model could depend on many factors such as the cues for depth discontinuity within the target and source patches, incoherency of the NNF, and the target-to-source patch distances (absolute and relative). The optimal model could be quite complex, so we propose to learn it from data generated by a successful patch-based synthesis process that makes use of masked patches. Our training data should include features extracted from target and matched source patches as well as the corresponding "ground truth" masks that led to a successful synthesis in the end of the process. That means for each set of features, there is a ground truth mask which is one of the M pre-defined masks. This is a multiway classification problem which can be addressed using random forests [14]. Next, we will explain the process of generating the data for our learning system.

Dataset - We simulate such data using the MIHF application, in which we want to fill a hole in a target image that contains multiple depth layers, using another image captured from another view as a source. In this case, we can ensure that the missing data in the target exists in the source either in the form of full or partial patches and it is easy to visually verify the plausibility of the fill.

Ground truth masks - For such a dataset it is also easy to generate corresponding "ground truth" masks. As shown in Fig. 4, we have two images from different view points, and choose a region that covers multiple objects in one of the images as our synthetic "hole". Note that in these training images, we have access to the original image and therefore we know the true content behind the hole. Thus, it is easy to manually define good masks for all patches according to the important boundary edges in the scene. For example, for the red patch, a mask that down-weights the right side of the patch would work best and vice-versa for the orange patch. Based on this observation, we manually draw the important boundary edges on the target image (called an edge image) as shown in Fig. 5. We then convert the boundary to a mask index map using the following patch-based process.

The goal of this process is to convert an edge image (Fig. 5 left) to a mask map that defines which mask should be used at each patch (Fig. 5 right). Intuitively, for patches that do not overlap an edge, a center mask will be a good default choice. For patches overlapping an edge, if the edge falls on the left side of the patch, the best choice would be a mask that down-weights the left side. Likewise, a mask that down-weights the right side would be the best choice for patches with an edge on the right side. To do this, we compute a score for every non-center mask:

$$\text{score}_j = \frac{\sum_{i=1}^{w\times w} m_j(i)E(i)}{\sum_{i=1}^{w\times w} E(i) + \epsilon},\ j \in \{2, \cdots M\}, \qquad (3)$$

where E is a patch on the edge image (Fig. 5 left) and ϵ is a small number (10^{-6}). We then set the score for the center mask equal to 0.3 and assign the mask that have minimum score. This tries to select the mask such that the pixels receiving the larger weights have as few edges as possible.

The process of computing mask map is done over multiple scales and the computed mask maps are then used as the ground truth mask in a hole-filling process. This process provides our *training set*: we extract a set of features (described in Section 3.2) for each patch, at each iteration, and in each scale, and use the ground truth masks as the target labels. We then use random forests [14] to learn a model for mask selection given the features for the target and matching source patches and their corresponding ground truth masks.

This construction of edge masks using manual marking of boundary edges led to excellent hole-filling results in all image pairs in our MIHF learning set as well as many SIHF examples we tried. Thus, one contribution of our work is a simple but effective user-assisted hole-filling method for challenging images with multiple depth layers inside the hole. Note that in our fully automatic system the manual

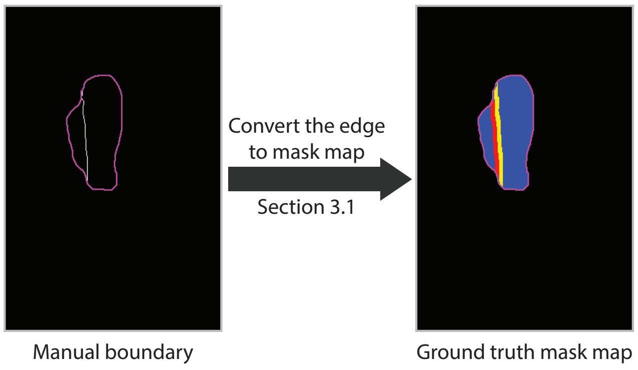

Figure 5. We manually draw the important boundary on the target image. This boundary is then converted to a mask map which we use as ground truth mask map. The manual boundary and mask map are for the dataset in Fig. 4.

boundary edges are only used to create the ground truth masks for the learning stage. Once a model is learned, it can predict the masks from the features (Section 3.2) in a fully automated way.

Simulating divergence - So far, we have explained how to learn a model from features extracted during synthesis by bootstrapping it with a ground truth oracle for mask prediction. Although we can learn a model that has high accuracy in predicting masks, it is not perfect. When using this model in a hole-filling process where the ground truth masks are unknown, it would likely diverge from a good solution due to the iterative nature of the optimization and inaccuracies in the model. However, our model is only trained on the ground truth examples, so its predictions will become poorer after such divergence occurs. Once this happens, it is very hard for the method to recover, because the test set no longer resembles the training set.

Specifically, at every iteration in the hole-filling process, the learned model will have inaccuracies in mask prediction which cause the computed target for the next iteration to diverge from the good solution. Then, in the next iteration, the model has to predict the masks using the features extracted from this new diverged target. If all the features are extracted from good target images (generated using the ground truth masks) in the learning stage, the model cannot accurately predict the masks and, consequently, the quality of the target for the next iteration worsens. Therefore, we want to learn a model that can predict the masks even when the features are extracted from diverged targets (a target image that results from inaccuracies in mask prediction).

In order to achieve this, we enrich the training set by using features from a *diverged* synthesis process. Specifically, we simulate divergence of the fill process and include the features extracted from the diverged targets along with their corresponding ground truth mask in the training set. Learning a model for this enriched training set increases the

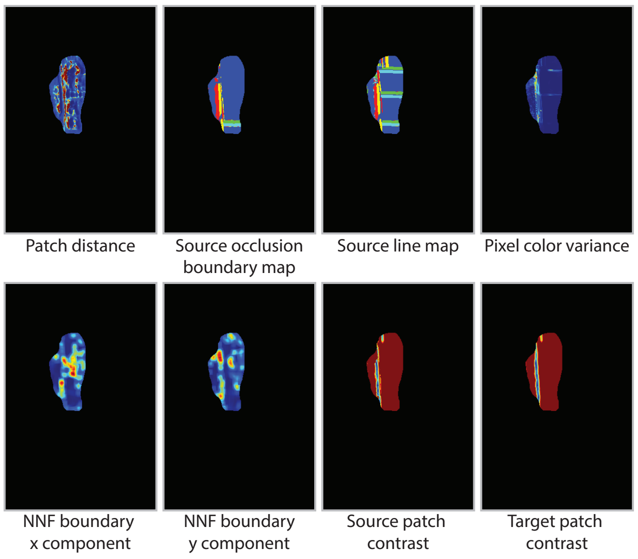

Figure 6. The features used in our learning system: All the features are extracted from the finest scale for the dataset shown in Fig. 4. The source occlusion boundary map and source line map are color coded using the colors defined in Fig. 2. Note some of these features are calculated for all M NNFs, but here we show only one of them.

robustness of the mask prediction to cases where divergence has occurred.

Since the reason for divergence in the actual hole-filling process is the inaccuracies in the model, we found the best way to simulate such divergence is by bootstrapping from the model itself. To do this, we first learn an initial model using the process explained above. Then we repeat the entire process with one key difference: at every iteration, instead of extracting the features from the current result, we perform a few iterations of search-and-vote using the masks predicted by the initial model. We then extract the features from the *diverged* target and append them along with their corresponding ground truth mask to the training set. Basically, this process simulates what happens if we use the initial model in an actual hole-filling process.

3.2. Features

In order to learn a reliable model, we need to extract good features. Below we explain the list of features used in our learning system, a summary of which can be found in Table. 1. Fig. 6 shows an example of each feature which has been extracted from the finest scale for the dataset in Fig. 4. Features that are defined on the source image are mapped to the target image using all or some of the M NNFs that we compute in the search process. Color related features are extracted in the L*a*b* color space.

Patch distance - We compute distance between the target and its matching source patch using Eq. 2 for NNFs 2 to M and normalize them by the distance of the center mask NNF. One might assume that the mask that gives minimum patch distance is always the best, but we observed that this often causes the optimization to converge to poor results.

Feature name	Dimension
Patch distance	$M-1$
Source occlusion boundaries	$M-1$
Source lines	$M-1$
Pixel color variance	1
Nearest neighbor field boundaries	$2M$
Source patch contrast	$M-1$
Target patch contrast	$M-1$
Scale index	1

Table 1. List of features and their dimensions.

Source occlusion boundaries - We use the method of Hoiem *et al*. [10] to extract occlusion boundaries on the source image. The extracted edges are transformed to a more useful feature by converting them to a mask map as explained in Section 3.1. This mask map is defined on the source image and we map it to the target using NNFs 2 through M. This is an important feature since it captures depth discontinuities, where non-center masks are often needed.

Source lines - We extract straight lines in the source image using the method of Akinlar and Topal [2]. These often correspond to edges in man-made objects which tend to correspond to depth or motion discontinuities and can complement the occlusion boundary detector. We convert them to a source mask map and the convert it to a target map as described above.

Pixel color variance - In order to measure the consistency of the overlapping patches at every pixel, we compute the variance of the pixel values from all overlapping source patches of that pixel in the voting. This variance is computed only for the center mask to measure the consistency of patches. Intuitively, this variance should be low whenever the patches are consistent, which we expect to be a good cue to use the center mask. Whenever the variance is large, non-center masks may be a better choice. We compute the variance for each channel separately and find the maximum of the three channels to penalize patches for which one channel is inconsistent.

Nearest neighbor field boundaries - The goal of this feature is to calculate the discontinuities in the nearest neighbor field. We observed that in the regions where the NNF is coherent we often do not need to use non-center masks, and masking is more helpful in the regions near discontinuities in the NNF. To do this, we compute the amplitude of the gradient of the M NNFs for the x and y components. We then set the small gradients to zero and filter the thresholded gradient with a Gaussian filter.

Source patch contrast - We compute the normalized luminance contrast of the source patch as follows:

$$c_j = \frac{\sum_{i=1}^{w\times w} m_j(i)Q(i) - \sum_{i=1}^{w\times w}(1-m_j(i))Q(i)}{\sum_{i=1}^{w\times w} Q(i)}. \quad (4)$$

Note that we compute the contrast only for the luminance channel and for the non-center masks ($j \in \{2, \cdots M\}$).

Target patch contrast - The contrast of the target patch is computed for the luminance channel using Eq. 4 for the masks 2 to M.

Scale index - The index of the current scale in the coarse-to-fine process. Since we learn a single model for all scales, this feature is important, so that the model could adapt the best combination of features and their parameters for each scale.

These features are used during training along with their corresponding ground truth masks to learn a model. At runtime, they are extracted from the input image and fed into the model to predict a reliable mask.

4. Results

We used 10 pairs of images for the training set, some of which are shown in Fig. 7. For each pair, we took two images of a scene from different viewpoints and used one of the images as the source image and the other as a target. We marked a hole in the target image in a region with two or three depth layers so that the model can learn how to handle such cases. Since the images had mostly vertical depth discontinuities, we added vertically-flipped, 90° rotated, and 90° rotated + horizontally-flipped versions of the images to the training set (40 pairs in total) to train the system symmetrically and to learn to account for horizontal depth discontinuities. We used a total number of more than one million patches from all the scales covering all different depth discontinuities and layouts. Moreover, since the ground truth mask maps contain mostly center masks (see blue color in right image of Fig. 5), we chose half of the training data from center masks and the other half from non-center masks to avoid overfitting. We used the PatchMatch method of Barnes *et al.* [4] to accelerate the search process. The patch width w was equal to 11 in all cases, and the sum of weights in all masks was equal to one. Finally, we used an online implementation of random forests[1] with 100 trees and default parameters.

Fig. 8 shows one of the examples from our training set, with our manually annotated boundaries and the "ground truth" result obtained by using masks derived from the annotated boundaries. See more examples and results in supplementary material. Annotating the important depth

[1]`https://code.google.com/p/randomforest-matlab/`

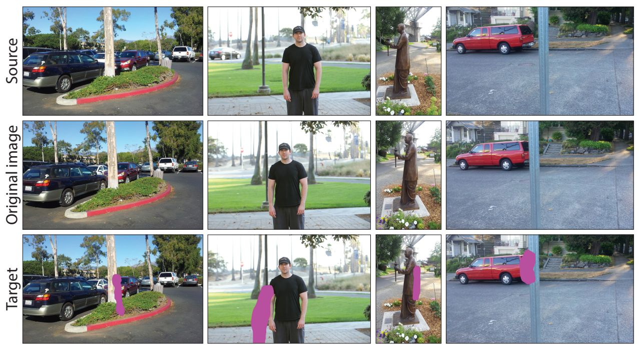

Figure 7. Some of the images used in the training set. We defined the hole (shown in pink) in the original images and used them with the source image to simulate the hole-filling process.

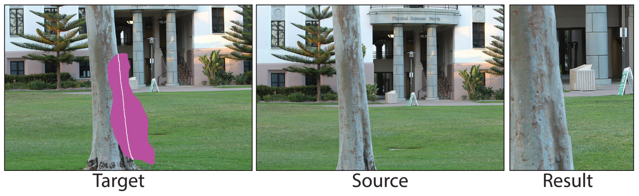

Figure 8. One example from our training set with the manual boundary edge annotated inside the hole and the corresponding result. Such simple user assistance led to excellent results in all our training examples (see supplementary material).

boundaries is an easy task (took less than a minute per image) and led to excellent hole-filling results in all of the image pairs in our MIHF learning set, as well as in many SIHF examples we tried. Note that this type of "guided" hole-filling is different from traditional edge guidance [4, 22] that indicates locally the location of the search region along an edge. These methods do not address the case where content is inconsistent across the edge, and thus are likely to have difficulties when applied to our examples.

In order to test the accuracy of our learning framework, we randomly selected one of the training set images, excluded it from our training set and trained a model using the rest of the datasets. We then performed hole-filling using the computed model on the excluded training image and compared the predicted masks with its ground truth masks. As before, we sampled half of the patches with center and the other half with non-center masks. We repeated this process five times and averaged the correct classification rates. The overall accuracy of our method across all scales was higher than 65%, increasing gradually from coarse scales to the fine ones (73.6% at the finest). This is compared to 20% for a random selection and 50% for always selecting the center mask.

We compare our algorithm against regular patch-based synthesis on the MIHF application in Fig. 9. In all these examples, the fixed-shape patch-based method has difficulties around depth discontinuities, resulting in broken edges and other artifacts. Our method produces plausible results with

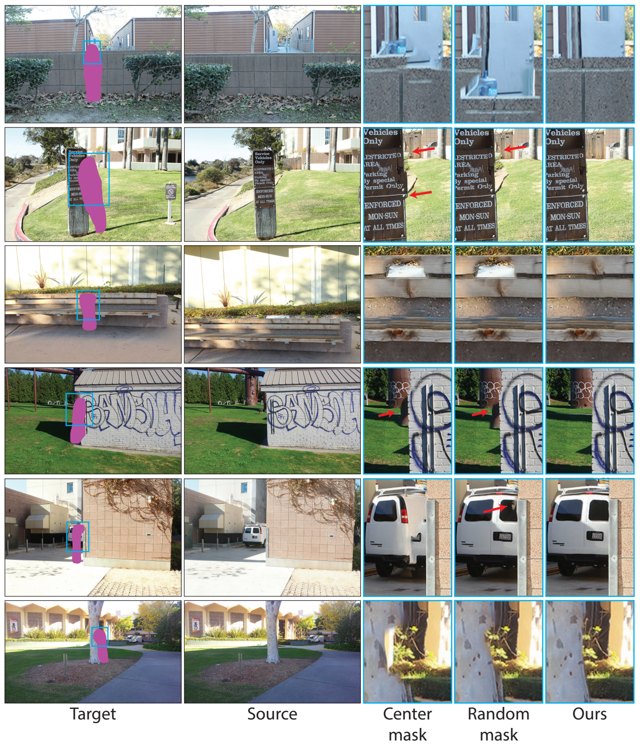

Figure 9. Comparison to regular patch-based method on MIHF.

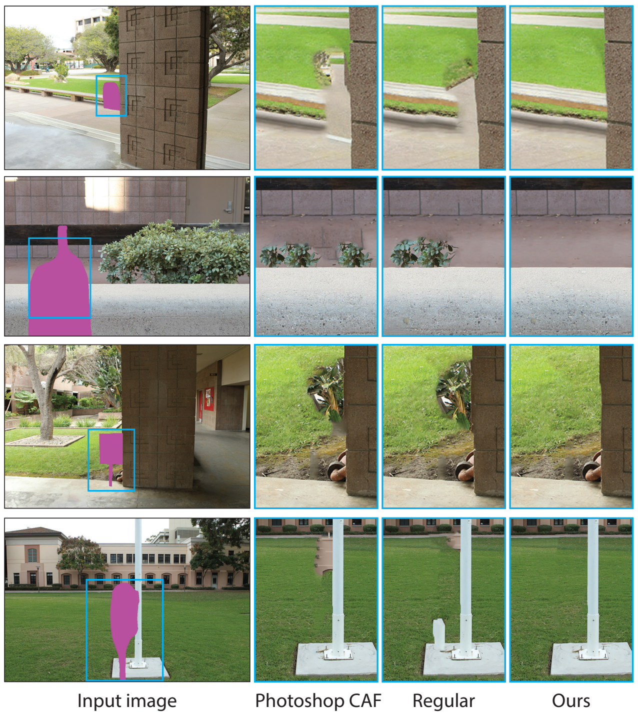

Figure 10. Comparison against Adobe Photoshop content-aware fill (CAF) and our implementation of a regular patch-based method on single image hole-filling.

minimal artifacts.

Next we use the learned model and test it on the SIHF application. In Fig. 10 we compare our results with regular hole-filling (our implementation using only center masks) and Photoshop's Content-Aware Fill [1]. Our method generates plausible results where others contain artifacts.

5. Discussion

The learned model relies on the features to predict a good mask. We observed that the source occlusion boundary is an important feature in our system. In some cases, the method of Hoiem *et al*. [10] fails to detect the occlusion boundaries, so the learned model might have a problem predicting the correct masks. Fig. 11 (left) shows the detected occlusion boundaries for the dataset in the second row of Fig. 10. If we simply remove all the detected edges, our algorithm generates the result on the right due to failure of the model in predicting the right masks.

Our approach has also limitations in case where the scene has many depth layers. One example of such a case is shown in Fig. 12. In this case, as shown by the red arrow, our method cannot fix the small parallax in the background so the back of the building slightly bends, similar to the regular patch-based synthesis result. However, our result looks more plausible overall.

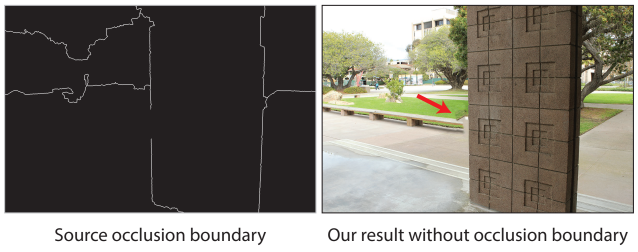
Source occlusion boundary
Our result without occlusion boundary

Figure 11. Source occlusion boundaries are an important feature in our system. The detected occlusion boundaries using the method of Hoiem *et al*. [10] for the dataset in Fig. 9 (first row) are shown. Our result without this feature is shown on the left.

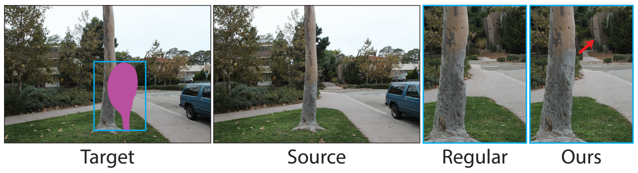

Figure 12. In cases where a scene has a lot of depth layers, our method does not fix small parallax shifts. Nonetheless, the overall result looks more plausible than the regular patch-based result.

Bobick and Intille [5] used nine shifted windows when comparing the patches to handle the depth discontinuities in stereo matching which is very related to our idea of masking the patches. Our mask patches can be seen as a generalization of this idea since the nine shifted windows can be generated by masking the appropriate pixels on a bigger patch. However, their shifted windows cannot produce the smooth masking effect like the circular mask we show in Fig.2.

In this paper we explored the effect of different layouts of image content on image editing. Although the proposed solution was specific to patch-based methods, this problem exists in other image editing methods like Shift-Map [15]. In Shift-Map, the gradient filters at coarse scales might capture a mixture of different depth layers and the simple neighborhood term in its objective function might not be sensitive enough to separate them. We believe a similar learning-based approach could be applied to other image and video editing methods, and leave this for future research. Our mask selection approach could be easily combined with recent extensions of patch-based hole-filling, like adaptive local search [12] and using statistics of patch offsets [9], that demonstrated impressive improvements.

6. Conclusion

We have presented a method that addresses a common problem of existing patch-based methods where patches cover multiple regions. We propose to use masks to down-weight some fragments of each patch, producing a more meaningful search-and-vote process in the inner loop of the synthesis algorithm. Predicting accurate masks is a hard problem that involves many factors, so we chose a learning approach using a dataset of successful synthesis data to predict a reliable mask. We show improvement over traditional patch-based techniques on single- and multi-image hole-filling applications.

7. Acknowledgments

The authors would like to thank the reviewers for their insightful suggestions. This work was partially funded by Adobe, and National Science Foundation grants IIS-1342931 and IIS-1321168.

References

[1] Adobe. Photoshop CC Content-Aware Fill, 2013. http://www.adobe.com/technology/projects/content-aware-fill.html.

[2] C. Akinlar and C. Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633 – 1642, 2011.

[3] C. Barnes, D. B. Goldman, E. Shechtman, and A. Finkelstein. Video tapestries with continuous temporal zoom. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 29(3), Aug. 2010.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24:1–24:11, July 2009.

[5] A. F. Bobick and S. S. Intille. Large occlusion stereo. *Int. J. Comput. Vision*, 33(3):181–200, Sept. 1999.

[6] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.*, 31(4):82:1–82:10, July 2012.

[7] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *ICCV*, page 1176, Washington, DC, USA, 2003. IEEE Computer Society.

[8] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009.

[9] K. He and J. Sun. Statistics of patch offsets for image completion. In *ECCV*, 2012.

[10] D. Hoiem, A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *International Journal of Computer Vision*, 91(3):328–346, 2011.

[11] N. K. Kalantari, E. Shechtman, C. Barnes, S. Darabi, D. B. Goldman, and P. Sen. Patch-based High Dynamic Range video. *ACM Trans. on Graphics (TOG) (Proc. of SIGGRAPH Asia 2013)*, 32(6), 2013.

[12] J. Kopf, W. Kienzle, S. Drucker, and S. B. Kang. Quality prediction for image completion. *ACM Trans. Graph.*, 31(6):131:1–131:8, Nov. 2012.

[13] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, August 2005.

[14] A. Liaw and M. Wiener. Classification and regression by randomForest. *R news*, 2(3):18–22, 2002.

[15] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-Map image editing. In *ICCV*, pages 151–158, Kyoto, Sept 2009.

[16] P. Sen, N. K. Kalantari, M. Yaesoubi, S. Darabi, D. B. Goldman, and E. Shechtman. Robust patch-based HDR reconstruction of dynamic scenes. *ACM Trans. on Graphics (TOG) (Proc. of SIGGRAPH Asia 2012)*, 31(6), 2012.

[17] E. Shechtman, A. Rav-Acha, M. Irani, and S. Seitz. Regenerative morphing. In *CVPR*, San-Francisco, CA, June 2010.

[18] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*, pages 1–8, June 2008.

[19] D. Sun, E. Sudderth, and M. Black. Layered segmentation and optical flow estimation over time. In *CVPR*, pages 1768–1775, 2012.

[20] D. Sun, E. B. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. *Advances in Neural Information Processing Systems*, 23:2226–2234, 2010.

[21] D. Sun, J. Wulff, E. Sudderth, H. Pfister, and M. Black. A fully-connected layered model of foreground and background flow. In *CVPR*, pages 2451–2458, 2013.

[22] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum. Image completion with structure propagation. *ACM Trans. Graph.*, 24(3):861–868, 2005.

[23] E. Trulls, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer. Dense segmentation-aware descriptors. In *CVPR*, pages 2890–2897, 2013.

[24] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *CVPR*, 2004.

[25] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! Internet-based inpainting. In *BMVC*, pages 1–11, 2009.