Hole Filling Through Photomontage

Marta Wilczkowiak,* Gabriel J. Brostow, Ben Tordoff,* and Roberto Cipolla
Department of Engineering, University of Cambridge, CB2 1PZ, U.K.
http://mi.eng.cam.ac.uk/research/Projects/HoleFilling/

Abstract

To fill holes in photographs of structured, man made environments, we propose a technique which automatically adjusts and clones large image patches that have similar structure. These source patches can come from elsewhere in the same image, or from other images shot from different perspectives. Two significant developments of this work are the ability to automatically detect and adjust source patches whose *macrostructure* is compatible with the hole region, and alternately, to interactively specify a user's desired search regions. In contrast to existing photomontage algorithms which either synthesize microstructure or require careful user interaction to fill holes, our approach handles macrostructure with an adjustable degree of automation.

1 Introduction

The clonebrush is a favorite tool in digital image manipulation because it allows interactive painting using pixels from the original image. While pixels from the same image are more likely to be self-similar, a user wanting to paint over a hole still faces three challenges: (1) choosing the right constant offset to the source pixels, (2) choosing an appropriate brush size, and (3) finding source pixels featuring structure that is compatible with the destination. We propose that these tasks can be assisted with automation.

Like recent works on texture synthesis [9] and photomontage [1], we also employ the graphcuts algorithm [3] to find the *shape* of a source patch which best fits a target location. For the related but specific task of filling image regions marked as holes, we introduce a form of automated clonebrush which finds the best source pixel *location* to match the large scale structure around the hole. By targeting macrostructure, the system can rectify and register regions from one or more non-aligned photos that were otherwise incompatible. The algorithm runs iteratively to produce the best output image which respects all the hard and soft constraints specified by the user.

2 Related work

The two main approaches to restoring missing image parts are inpainting and texture synthesis, where texture synthesis can be further divided into parametric and exemplar based techniques. A thorough review of the literature on these methods suggests overall agreement about their capabilities. In general, image inpainting propagates color information inward with variational approaches to fill a hole, and is best suited for small, smooth, low-textured regions. Parametric texture synthesis deals with determining a minimal set

*Currently employed at Mathworks Ltd.

BMVC 2005 doi:10.5244/C.19.52

of statistical measurements for characterization of texture. Naturally, they have impressive results for highly stochastic textures, but are not well suited to filling holes in highly structured elements such as building façades and other man made environments.

The exemplar-based sampling approaches have had the greatest success. We have endeavored to combine the progress others have made in automating the respective subproblems of source patch selection and placement with the user interaction that is necessary to replicate large scale structure. Drori et al. [5] is a recent example in which candidate source patches are automatically flipped and rotated to find combinations that will better fill the hole region. Understandably, it has problems with macrostructure and produces blurry output when even the best source samples are a poor match for the target zone. Criminisi et al. [4] showed how continuations of strong edges could be propagated inward, allowing for high-quality regeneration of relatively simple structure.

Higher level coherency is achieved when drawing source patches from multiple images which were either shot using a tripod [1] or whose contents were aligned implicitly based on motion [15]. While Agarwala et al. [1] was not intended as a region-filling tool, its elegant user interface makes it easy to replace one version of a pixel patch with an approximately co-located patch in another image, containing preferred lighting, facial expressions, etc. We extend that user interaction to draw source patches from elsewhere entirely. The work of Wexler et al. [15] samples spatio-temporal patches from video to complete both static scene parts and dynamic behaviors. While their full automation is generally desirable, we will show that large structured environments can be region filled by adding just a modicum of user interaction.

Finally, the work of Kwatra et al. [9] is most relevant to our approach. They too had an interactive mode for cloning of both small and very large regions, but they drew from user specified locations. Although we do not implement their iterative texture synthesis mode, we extend their interactive work with controllable automation of source selection, alteration, and placement.

3 Patch fitting

In the input image I, we refer to the area of the target region that needs to be filled as R. The texture is taken from the same or another image from a neighborhood S, which in the trivial case is the remainder of I: $\mathsf{I} \setminus \mathsf{R}$. All exemplar-based methods rely on an efficient mechanism for finding patches in S which fit neighborhood R. In the simple case of rectangular patches, preprocessing the image data greatly accelerates the search. Indeed, all the fixed-shape image regions can be clustered using approximate nearest-neighbor search mechanisms, such as tree-structured vector quantization [14, 7] or kd-trees [10, 6, 12]. However, in the hole-filling problem, the shape of the neighborhood varies with the shape of the target region, preventing such preprocessing.

To find the mask J associated with the target region R, we first expand the boundary of R by h pixels. The mask J is painted as a thin band surrounding the target region by using a distance image I_d, where pixels contain the distance to the nearest hole pixel. Values are then assigned to J as follows:

$$\mathsf{J}_{i,j} = \begin{cases} 0 & \text{if } \mathsf{I}_d(i,j) > h \vee (i,j) \in \mathsf{R} \\ 1 & \text{if } \mathsf{I}_d(i,j) < h \end{cases} \tag{1}$$

Although it is possible to weight the importance of different pixels in the matching procedure using e.g. a gaussian centered on the hole's outline, this does not produce observably better matches.

3.1 Finding the best matches

Let us now consider finding a patch m in the source region S which fits the patch p from the target region R. The mask J (as above) defines pixels which should be taken into account while matching. A search is performed for the translation $\mathbf{x}_0$ of the source region S which minimizes the (weighted) sum-of-squared-differences (SSD) between source and target:

$$SSD(\mathbf{x}_0) = \sum_{\mathbf{x}\in\mathsf{p}} \mathsf{J}(x)\left(\mathsf{I}(x) - \mathsf{S}(\mathbf{x}_0 + x)\right)^2. \tag{2}$$

Assuming that all the pixel values in the target region are zeroed, following [13] the above expression can be rewritten as:

$$SSD(\mathbf{x}_0) = \sum_{\mathbf{x}\in\mathsf{p}} \mathsf{I}(x)^2 - 2\mathsf{J}(x)\mathsf{S}(\mathbf{x}_0 + x) + \mathsf{J}(x)\mathsf{S}(\mathbf{x}_0 + x)^2. \tag{3}$$

The second and third terms of the above expression are correlations of the patch p with the source image and its supporting mask J with the squared source image. These correlations can be computed in $O(n\log(n))$ time using the Fast Fourier Transform (FFT).

Representing pixel values in CIE-Lab color space makes the Euclidean distance between two color vectors more closely reflect their perceptual difference. The SSD can therefore be summed across all three color channels to give a full color difference measure.

For highly structured images, it is very important that the two patches p and m fit well in areas corresponding to large gradients in the image. A simple way of taking this into account is to compute the gradient magnitude image I_G and use it as a fourth color channel when matching the patches.

4 Using graph cuts for region filling

We base our algorithm on the assumption that visual coherence is more important than keeping the fidelity of the original image around the designated fill region. The algorithm is therefore permitted to extend the hole to allow a visually coherent replacement. This extends the texture synthesis algorithm of [9] where texture is synthesized patch by patch, each new patch being chosen from the source region based on similarity to the target neighborhood. Consider placing a new patch m, slightly overlapping a previously pasted patch p. All the pixels in the overlapping region are associated with nodes in the graph. All the neighboring pixels are connected by edges with weights corresponding to the color difference between them. Specifically, let s and t be two adjacent pixels in the overlap region, while $\mathsf{p}(s)$ and $\mathsf{m}(s)$ are color values at the position s in old and new patches, respectively. The similarity factor between nodes representing pixels s and t may be defined as follows [9]:

$$M(s,t,\mathsf{p},\mathsf{m}) = \|\mathsf{p}(s) - \mathsf{m}(s)\| + \|\mathsf{p}(t) - \mathsf{m}(t)\|. \tag{4}$$

Finally, two terminal nodes, representing patches p and m, are added to the graph. Edges with infinite weight are added between p, m and those pixels which *have to* come from

them (*e.g.,* pixels in the hole *must* come from the replacement patch). Having defined such a graph, finding a cut minimizing errors between adjacent pixels from different patches is equivalent to finding a labeling f which assigns each pixel in the overlapping region to either patch p or m. f is computed to minimize the energy

$$E_{smooth}(f) = \sum_{s,t \in \mathscr{N}} V(f_s, f_t), \tag{5}$$

where $\mathscr{N}$ is a set of neighboring pixels and $V(f_s, f_t)$ is a penalty for pixels s, t being assigned to different patches. In our case we have

$$V(f_s, f_t) = \mathsf{M}(\mathsf{s}, \mathsf{t}, \mathsf{p}, \mathsf{m}). \tag{6}$$

This minimization problem can be efficiently solved using graph cuts [3].

4.1 Computing the extended target region

To find a better minimum, the system can expand the fill region. Let us suppose that the patch m matching the patch p covering the target region has been computed as explained in Section 3. Patches m' and p' are created by expanding patches m and p by k pixels.. This number specifies the width of a band around the original patches in which the expansion of the original patch will be allowed. Now the problem of finding an optimal replacement of pixels in patch p' from those in patch m' is equivalent to finding a path in the graph such that the error between the pixels coming from different patches is minimized. However, a lower energy path in an unconstrained region R could lead to R expanding unnecessarily. It is thus necessary to introduce a factor attracting the cutting path to the border of the original, user-defined region. This factor can be naturally incorporated in the graph-cut framework. Instead of minimizing (5), we compute a labeling f, minimizing the function

$$E(f) = \alpha E_{data}(f) + E_{smooth}(f), \tag{7}$$

where α corresponds to the relative importance of both factors, $E_{smooth}(f)$ is defined as in (5) and constrains the labeling to be spatially coherent in the image space, and $E_{data}(f)$ corresponds to the quality of the partitions in the feature space.

$$E_{data}(f) = \sum_{s \in \mathscr{P}} D_s(f(s)), \tag{8}$$

where $\mathscr{P}$ is the set of all pixels in region p' (or matching m'). $D_s(f(s))$ measures how appropriate a label $f(s)$ is for the given pixel s based on its intensity. In this application, for a given pixel s we model its preference of belonging to the patch $f(s)$ as:

$$D_s(f(s)) = \exp \frac{d_f(s)^2}{2\sigma^2}, \tag{9}$$

where $d_f(s)$ is the distance of pixel s to the closest pixel constrained to belong to patch $f(s)$, and σ controls the width of the region where pixels are constrained to be attracted to patch p or m.

(7) is a full version of the energy function which can be directly minimized using graph cuts [3], given a graph which reflects all of the constraints. The nodes of the graph

correspond to the pixels in the overlapping patches p' and m'. There are two kinds of edges, referred to as N-links and T-links. N-links are defined between neighboring pixels and their weights correspond to the similarity criterion defined in (5). T-links connect pixels to terminals (patches p', m'). To ensure that pixels in region R belong to the new patch m', the T-links between them and the node representing the patch m' are assigned infinite weight. Similarly, the T-links between patch p' and pixels on the maximum border of the replacement area are also assigned an infinite weight. Finally, the T-links between all the remaining pixels and nodes representing patches p' and m' are assigned weights $\alpha D_s(\mathsf{p}'(s))$ and $\alpha D_s(\mathsf{m}'(s))$, respectively. An illustration of such a graph is shown in Figure 1. Note that the smoothness factor $V(f_s, f_t)$ is metric, so following [3], the graph-cut optimization still guarantees a global minimum.

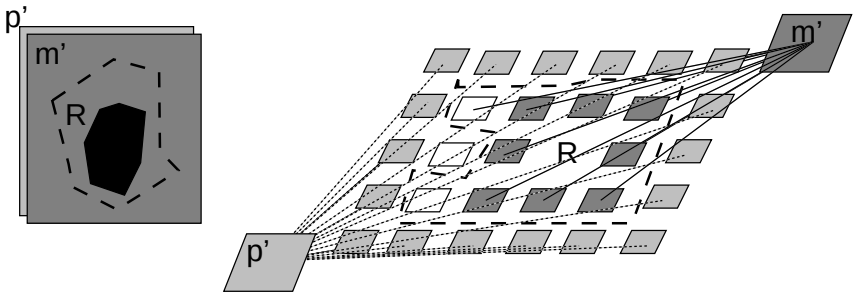
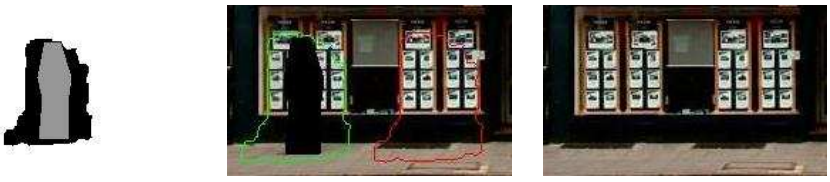


Figure 1: Finding an optimal region for the hole filling. For clarity, the edges between neighboring pixels are not marked.

4.2 User Constraints

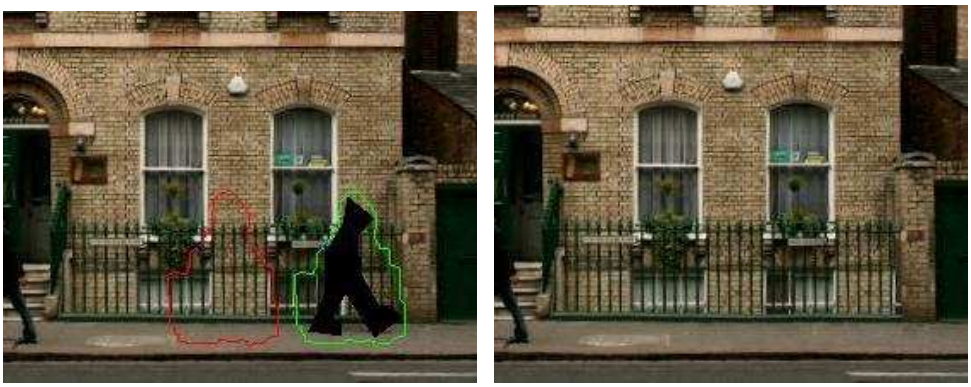
While the system's default behavior is to seek out S within the input image which best fills the hole, a user can limit the search area however they want, specifying a bounding box in other images, or even paint-selecting specific pixels. This is especially useful when the best-available replacement patch in an image is incompatible with artistic decisions about the content of the result. When one such constraint is specified, the algorithm runs normally but ignores candidate source patches outside the designated search area.

In a great number of images, a region being filled covers not one but *multiple* types



(a) (b) (c)

Figure 2: Region filling process: (a) Target region extended using graph cuts. (b) The extended target region and its matching part in the image. (c) The final result.



(a) (b)

Figure 3: (a) The extended target region and its matching part in the image. (b) The generated result.

of structured texture. Although our approach is intended for tools where a user refines an image, progressively removing different pieces, it is faster for the user to specify multiple constraints each time. This means the optimization must handle multiple search areas to fill adjoining destination regions. Still employing graph cuts, which can draw a seam between only two regions, the algorithm iterates, performing α-expansion steps [3] each time to propagate constraints one after another. Note that when given hard constraints that force certain pixels to be copied into specific places in the target region, the data factor $D_s(f(s))$ could be used to individually control the attraction of neighboring pixels to the user-specified region, but we used it only with fixed parameters. This mechanism would correspond to explicitly changing the size of a brush in the clonebrush tool. Since the factor $V(f_s, f_t)$ is a metric, α-expansion steps produce a solution within a known factor of the global minimum of $E(f)$, corresponding to a perfect cut between all the user's constraints. A proof of this can be found in [3].

Single image rectification In using our system, we found a great many scenes where the available source texture is either too big or too small to be applied directly, because of perspective effects. Particularly in images involving buildings, a single planar façade dominates the scene. This allows perspective effects to be removed for all parts of the façade by the process of *metric rectification* [11]. Line segments are grouped to determine the vanishing directions [2] and the image warp H_w found that moves these vanishing points to infinity (*i.e.,* makes horizontal and vertical lines on the façade parallel to the image x- and y-axis respectively). Once rectified, the quality of hole-filling of regions on the dominant plane is greatly improved. Note that the warp is only needed to decide how to fill the hole, and the actual replacement can be done on the original image.

Multiple image registration Where the region to be filled appears in two images, it is desirable to look for sources in both images. For this to work, the second image must be *registered* to the first in order to maximise the similarity. For images taken as part of a panoramic mosaic, this registration is exact and involves warping the second image using

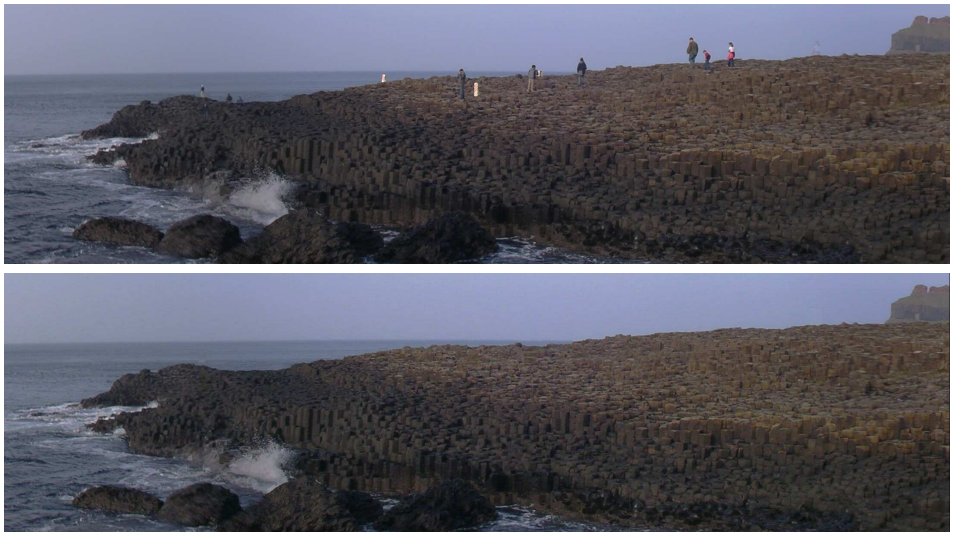


Figure 4: Giants Causeway Panorama: (Top) Original image. (Bottom) Result after filling the user defined regions. The abundance of example structure in the scene makes this a standard and easy hole-filling example.

an infinity homography. When the two views are taken from different viewpoints, the best that can be achieved is to remove rotational and scale differences by warping the second image to give the best alignment of the epipoles [8]. The warp itself is a homography and is only used to determine the best replacement — again, the actual hole-filling occurs on the original image.

5 Results Image Gallery

Figures 4, 5, 6, 7, and 8 illustrate results of our algorithm and describe the different modes of interaction that are possible. Except for the Giants Causeway example, these pose a significant challenge to existing exemplar based techniques because they either require that multiple user constraints be satisfied simultaneously, or because the original images contain no suitable candidate patches that would line up with the macrostructure around the hole.

6 Conclusion

We have demonstrated a hole filling algorithm which can automatically locate candidate source patches and can adhere to user specified constraints. Our approach gives satisfactory results in images containing significant structure. The automatic extension of the user-defined target region allows for fast and visually coherent merging of image patches. Our iterative approach allows the system to optimize for multiple constraints at a time. Finally, the automatic rectification and registration of multiple images allows relevant source pixels to be copied from regions that would otherwise be incompatible with the target region's neighborhood.

To address the limitations of the system, future work will incorporate more advanced vanishing point detection, color histogram correction, and selective multi-frequency im-

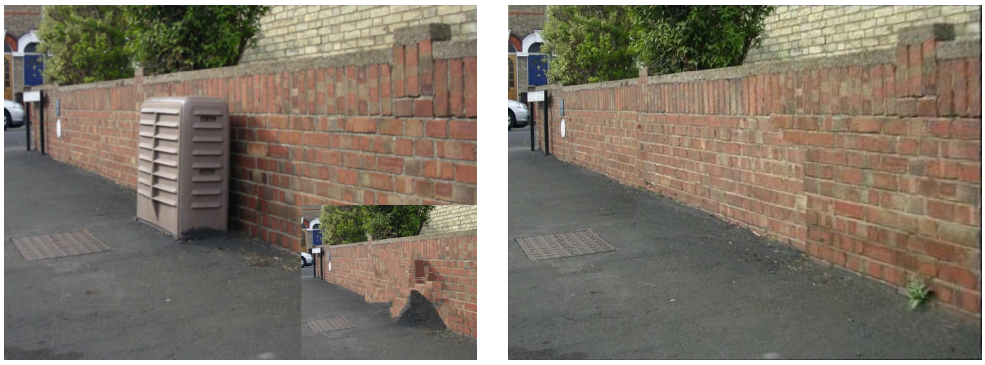
(a) (b) (c)

(d) (e)

Figure 5: N. Irish Parliament: (a),(b) Original images. (c) Our result after filling people pixels. (d),(e) Zoomed results of using M.S. Digital Image Pro v.10® vs. our approach. Without registering the two inputs, our replacement patches of stair texture would also appear with visible artifacts.

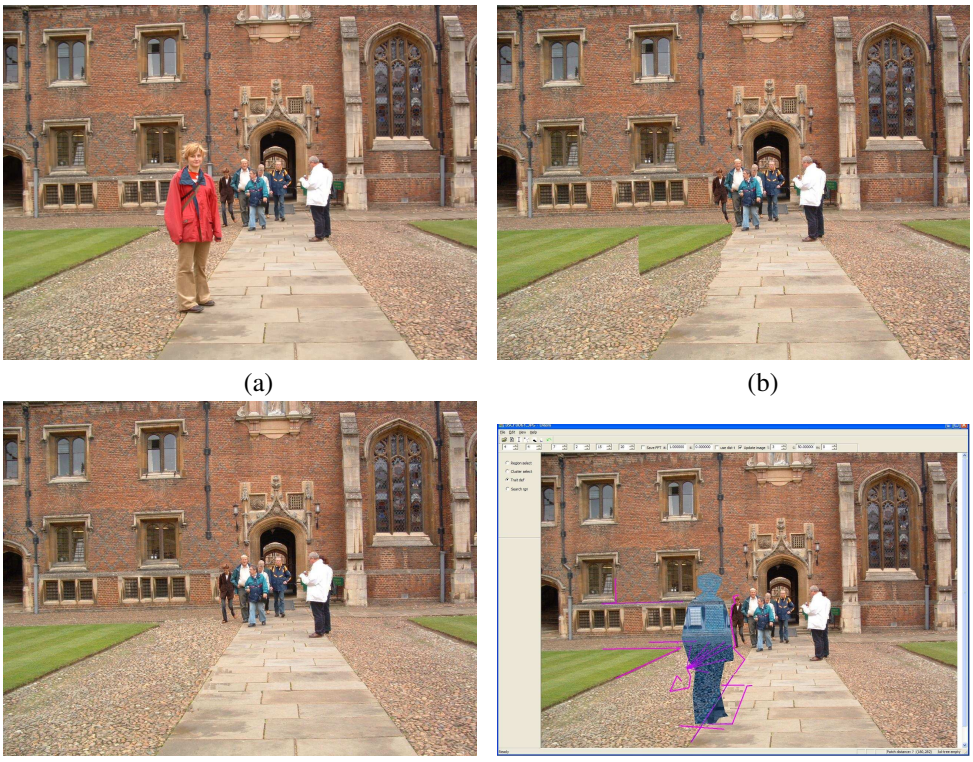
(a) (b)

Figure 6: Romsey traffic light: (a) Original image. (b) Result after filling the user defined regions. The cloned window and brick texture was compatible due to rectification. The user specified the right-of-doorway pillar as a hard constraint.



(a) (b)

Figure 7: Transformer: (a) Original. Inset is result of Smart Erase tool in M.S. Digital Image Pro v.10®. (b) Our result after filling the user defined region. Bricks replacing the transformer are brighter than appropriate because no color correction is activated.



(a) (b)

(c) (d)

Figure 8: St. John's College: (a) Original image. (b) Result without user interaction where system replaced the girl using one large compatible patch. (c) Result with user interaction. (d) User interface, showing that the user specified the girl as the fill-region. Also, the woman wearing brown and the edges of the sidewalk, window, and lawn are tagged as hard constraints, as are the cobblestones.

age splicing to reduce artifacts. Also, with the ability to adapt texture from other images, we intend to explore opportunities for efficiently filling holes using image libraries.

7 Acknowledgements

We are grateful to Vivek Kwatra for helpful discussions, and to Vladimir Kolmogorov for the graph-cut optimization libraries. The authors were supported in part by Cambridge-MIT Institute funding and a Marshall Sherfield Postdoctoral Fellowship.

References

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Trans. Graph.*, 23(3):294–302, 2004.

[2] M.E. Antone and S.J. Teller. Automatic recovery of relative camera rotations for urban scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2282–2289, 2000.

[3] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[4] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 721–728, June 2003.

[5] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun. Fragment-based image completion. *ACM Trans. Graph.*, 22(3):303–312, 2003.

[6] J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw*, 3:209–226, 1977.

[7] A. Gersho and R. M. Gray. *Pyramid-based texture analysis/synthesis*. Kluwer, 1992.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry, 1st edition*. Cambridge University Press, 2000.

[9] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003.

[10] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001.

[11] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 482–488, 1998.

[12] D. M. Mount. Ann programming manual. Technical report, Department of Computer Science, University of Maryland, College Park, Maryland., 1988.

[13] Cyril Soler, Marie-Paule Cani, and Alexis Angelidis. Hierarchical pattern mapping. In *Siggraph*, July 2002.

[14] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[15] Yonatan Wexler, E. Shechtman, and Michal Irani. Space-time video completion. In *CVPR (1)*, pages 120–127, 2004.