To appear in ICCV'09, Kyoto, Japan, Sept. 2009

Shift-Map Image Editing

Yael Pritch Eitam Kav-Venaki Shmuel Peleg
School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Abstract

Geometric rearrangement of images includes operations such as image retargeting, inpainting, or object rearrangement. Each such operation can be characterized by a shift-map*: the relative shift of every pixel in the output image from its source in an input image.*

We describe a new representation of these operations as an optimal graph labeling, where the shift-map represents the selected label for each output pixel. Two terms are used in computing the optimal shift-map: (i) A data term which indicates constraints such as the change in image size, object rearrangement, a possible saliency map, etc. (ii) A smoothness term, minimizing the new discontinuities in the output image caused by discontinuities in the shift-map.

This graph labeling problem can be solved using graph cuts. Since the optimization is global and discrete, it outperforms state of the art methods in most cases. Efficient hierarchical solutions for graph-cuts are presented, and operations on 1M images can take only a few seconds.

1. Introduction

Geometric image rearrangement is becoming more popular as it is being enabled by recent computer vision technologies. While early manipulations included mostly crop and scale, modern tools enable smart photomontage [1], image resizing (a.k.a. "retargeting") [2, 13, 19, 14, 16], object rearrangement and removal [5, 14, 6]. Recent retargeting methods propose effective resizing by examining image content and removing "less important" regions. Fig. 1 and Fig. 2 show comparisons of a few retargeting methods.

Seam carving [2, 13] performs retargeting by iterative removal of narrow curves from the image. As an iterative greedy algorithm no global optimization can be made, and something as simple as removing one of several similar objects is impossible. Since seam carving removes regions having low gradients, significant distortions occur when most image regions have many gradients.

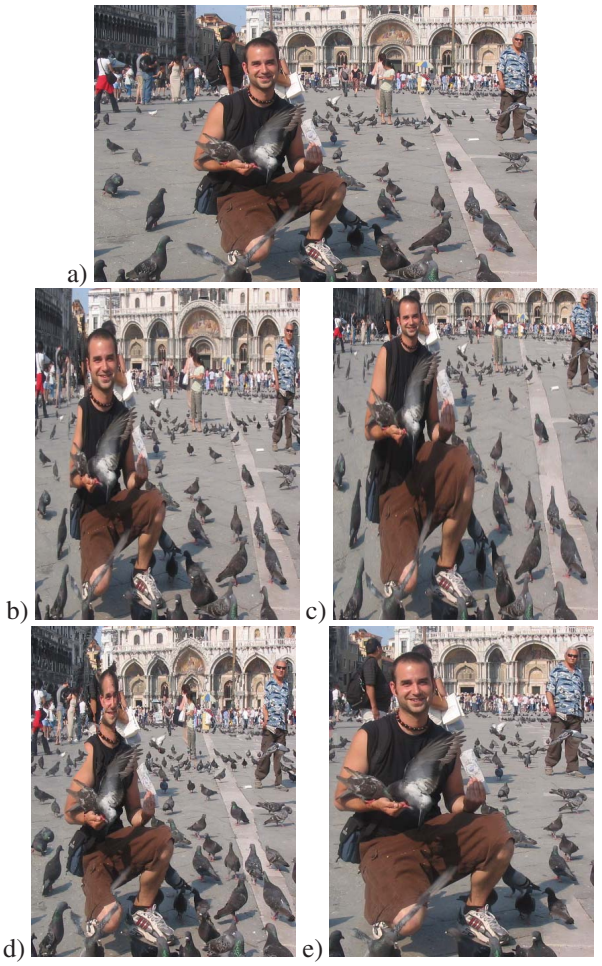


Figure 1. Comparison of a few retargeting methods, reducing width by half. (a) Original image. (b) Video-retargeting [19]; (c) Optimized scale-and-stretch [16]; (d) Improved Seam Carving [13]; (e) **Our shift-map editing**.

The use of a continuous image warping for retargeting was proposed in [19, 16]. While providing global considerations, continuous warping can introduce significant distortions, and good object removal is almost impossible. Both methods use saliency maps (e.g. face detection), and

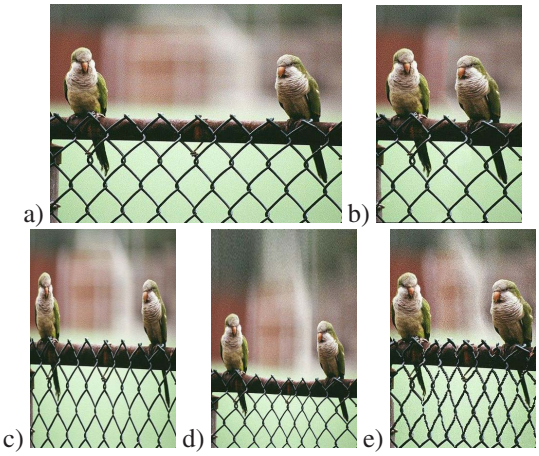


Figure 2. Comparison of a few retargeting methods, reducing width by half. (a) Original image. (b) **Our shift-map editing**. (c) Video-retargeting [19]; (d) Optimized scale-and-stretch [16]; (e) Improved Seam Carving [13];

saliency mistakes cause distorted results. One of the major problems in these methods is the application of different scaling to different objects. This causes most of the distortions visible in Fig. 1.(b-c).

An approach based on bidirectional similarity is presented in [14], which also names retargeting as "summarization". Every feature in the input should appear in the output, and every feature in the output should appear in the input. This method can also be used for image rearrangement.

The method most related to our work is the patch transform [5], which segments the image into patches which are than rearranged using global optimization. The need for prior determination of the patch size is a major drawback of this method. Also, the patches reduce significantly the flexibility for rearrangement and composition. The inherent problems of using patches are also affecting the object removal in [9]. We found that our results, moving individual pixels, significantly improve the results in [5].

In shift-map editing a global optimization for a discrete labeling is performed over individual pixels, overcoming most of the difficulties of previous methods. This is demonstrated in the simple retargeting example in Fig. 2, where best retargeting is very easy: a simple removal of a segment in the net that leaves its structure intact. This is easily possible with shift-map. In general, shift-map avoids scaling and mostly remove or shift image regions. Multi resolution optimization makes the shift-map computation very efficient, and most of the examples in this paper were prepared in less than 30 seconds.

2. Image Editing as Graph Labeling

The relationship between an input image $I(x, y)$ and an output image $R(u, v)$ in image rearrangement and retargeting is defined by a shift-map $M(u, v) = (t_x, t_y)$. The output pixel $R(u, v)$ will be derived from the input pixel $I(u + t_x, v + t_y)$.

The optimal shift-map is defined as a graph labeling, where the nodes are the pixels of the output image, and each output pixel can be labeled by a shift (t_x, t_y). The optimal shift-map M minimizes the following cost function:

$$E(M) = \alpha \sum_{p \in R} E_d(M(p)) + \sum_{(p,q) \in N} E_s(M(p), M(q)), \quad (1)$$

where E_d is a data term providing external requirements, and E_s is a smoothness term defined over neighboring pixels N. α is a user defined weight balancing the two terms, and in all our examples we used $\alpha = 1$. Each term will now be defined in detail. Once the graph is given, the shift-map labeling is computed using multi-label graph cuts [8, 4, 3].

2.1. Single Pixel Data Term

The data term E_d is used to enter external constraints. We will describe the cases of pixel rearrangement, pixel removal, and pixel saliency.

2.1.1 Pixel Rearrangement

When an output pixel (u, v) should originate from location (x, y) in the input image, the appropriate shift gets zero energy while all other shifts get a very high energy. This is expressed in the following equation:

$$E_d(M(u, v)) = \begin{cases} (u + t_x = x) \wedge (v + t_y = y) & 0 \\ otherwise & \infty \end{cases} \quad (2)$$

For example, in changing the width of the image, this constraint is used to determine that both the leftmost and rightmost columns of the output image will come from the leftmost and rightmost columns of the input image.

2.1.2 Pixel Saliency and Removal

Specific pixels in the input image can be forced to appear or to disappear in the output image. A saliency map $S(x, y)$ will be very high for pixels to be removed, and very low for salient pixels that should not be removed. The data term E_d for an output pixel (u, v) with a shift-map (t_x, t_y) will be

$$E_d(M(u, v)) = S(u + t_x, v + t_y) \quad (3)$$

It is also possible to use automatic saliency map computed from the image such as the ones proposed in [19, 16].

2.2. Smoothness Term for Pixel Pairs

The smoothness term $E_s(M(p), M(q))$ represents discontinuities added to the output image by discontinuities in the shift-map. A shift-map discontinuity exists between two neighboring locations (u_1, v_1) and (u_2, v_2) in the output image R if their shift-maps are different: $M(u_1, v_1) \neq M(u_2, v_2)$. The smoothness term $E_s(M)$ takes into account both color differences and gradient differences between corresponding spatial neighbors in the output image and in the input image to create good stitching. This treatment is similar to [1].

$$E_s(M) = \tag{4}$$

$$\sum_{(u,v)\in R}\sum_{i}(R((u,v)+e_i) - I((u,v)+M(u,v)+e_i))^2 +$$

$$\beta\sum_{(u,v)\in R}\sum_{i}(\bigtriangledown R((u,v)+e_i) - \bigtriangledown I((u,v)+M(u,v)+e_i))^2,$$

where e_i are the four unit vectors representing the four spatial neighbors of a pixel, the color differences are Euclidean distances in RGB, $\bigtriangledown I$ and $\bigtriangledown R$ are the magnitude of the image gradients at these locations, and β is a weight to combine these two terms. In most of our experiments we used $\beta = 2$. As both color differences and gradient differences are used for smoothness, structure is better preserved.

As we use non metric distances, many of the theoretical guarantees of the alpha expansion algorithm are lost. However, in practice we have found that good results are obtained. We further found that squaring the differences gave better results than using the absolute value, preferring many small stitches over one large jump. Deviation from a metric distance was also made in [10, 1].

3. Hierarchical Solution for Graph Labeling

Finding the optimal graph labeling as described in the previous section can be computationally infeasible, due to the very large number of nodes and of labels. In some cases a pixel in the output image could originate from any pixel in the input image and the number of possible labels is the number of pixels in the input image.

A heuristic hierarchical approach for finding the optimal graph labeling can substantially reduces the memory and computational requirements of the graph-cut algorithm. It provides good results for most of the shift map editing applications, even though optimality cannot be guaranteed. The speedup obtained by this approach is of several orders of magnitude, turning an intractable problem into a problem that can be solved in a few seconds.

A full shift map is first solved in a coarse resolution, in which both the number of nodes (image pixels) and the number of labels (possible shifts) are reduced. For example, at the $4th$ pyramid level the number of nodes and the number of labels are reduced by a factor of 64 (in the case of both horizontal and vertical shifts). Once a coarse shift-map is found, it is interpolated to an initial guess for a higher resolution using a nearest neighbor interpolation, and the shift-map values are doubled to match the higher image resolution.

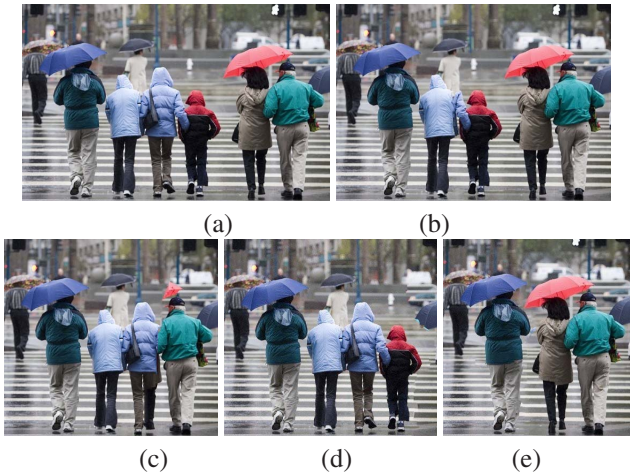


Figure 3. Shift-map retargeting for different output widths. In each case different objects are removed. (a) Original image taken from [13]; (b-c-e) Different output widths using no saliency. (d) Same width as (c), but the child was marked salient.

In the higher resolution levels only small shifts relative to the initial guess are examined. In our implementation, we used three relative shifts, (-1, 0, +1), in each coordinate, giving a total of nine labels for both directions. It is important to note that the data and smoothness terms are always computed with respect to the actual shifts, and not to the labels. We used three to five pyramid levels, such that the coarsest level contains up to 100×100 pixels. The shift map computation took between 0.5 to 30 seconds for most of the examples in this paper. To increase accuracy in label discontinuities, more than three refinement labels can be used. For example, the possible labels of a pixel can represent not only shifts from its own parent, but also shifts from the neighbors of its parent. This will improve accuracy at shift discontinuities, but at higher computational and memory costs.

While the hierarchical approach is not guaranteed to give the global optimum, the results are very good as can be seen in the examples. It is likely that this success, as the success of most pyramid approaches in computer vision, can be attributed to the observation that a natural image includes information in all frequencies. A multi-resolution method for graph-cuts with two labels (min-cut) was used in [11, 13], and was shown to provide good results. The contribution in this paper is to extend this case to multiple labels depicting image displacement.

4. Shift-Map Applications

Shift-map is computed by an optimal graph labeling, where a node in the graph corresponds to a pixel in the output image. In the section we describe how to build the graph and use shift-maps for several image editing applications.

4.1. Image Retargeting

Image retargeting is the change of image size, which is typically done in only a single direction in order to change the image aspect ratio. We will assume that the change is in image width, but we could also address changing both image dimensions.

4.1.1 Label Order Constraint

In image resizing it is reasonable to assume that the shift-map will retain the spatial order of objects, and the left-right relationship will not be inverted. This implies a monotonic shift-map. In the case of reducing image width, if $M(u,v) = (t_x, t_y)$ and $M(u+1,v) = (t'_x, t'_y)$, than $t'_x \geq t_x$. This restriction limits the number of possible labels to be the number of removed pixels: when reducing or increasing the width of the input image by 100 pixels, the label of each pixel can only be one of 101 labels. In addition, the smoothness term will give an infinite cost to cases when $t'_x < t_x$. Note that for reducing image width the values of t_x are non negative, and for increasing image width the values of t_x are non positive and $t'_x \leq t_x$. The order constraint is important also in case a saliency map is used, as it helps to avoid duplication of salient pixels.

Theoretically, only horizontal shifts need to be considered for horizontal resizing. However, this makes it impossible to respect geometrical image properties such as straight diagonal lines. When small vertical shifts are allowed, in addition to the horizontal shifts, the smoothness term will better preserve image structure.

4.1.2 Controlling Object Removal

It is possible to control the size and number of removed objects by performing several steps of resizing, since the smoothness constraints will penalize the removal of an object larger than the step size. In Fig. 4 it is demonstrated that when more steps are performed fewer objects are removed from the image. It is interesting to note that if the number of steps becomes the number of removed columns, each iteration changes the image width by only one pixel, and shift-map retargeting becomes practically equivalent to the seam carving algorithm [13]. Shift-map can therefore be considered as a generalization of seam carving, adding the flexibility to remove larger strips in a single step, possibly removing entire objects. It is also possible to control object removal by marking objects as salient as in Fig. 3.d.

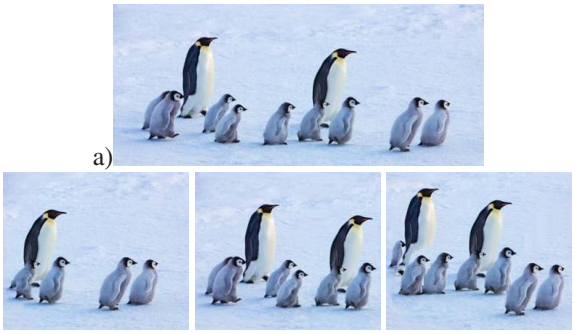


Figure 4. Controlling object removal by changing the number of steps. (a) Original image. (b) Resizing in a single step may cut out some of the objects. (c) Six smaller resizing steps remove fewer objects. (d) Ten even smaller steps remove even fewer objects. Note that in order to fit more objects in a smaller image, the result in (d) has vertical shifts introduced automatically by the optimization process.

4.1.3 comparison to other algorithms

Comparison of shift map with several retargeting algorithms is provided in Fig. 13 at the end of the paper. All our results were done with a single step of the algorithm and the same set of parameters as described before. The other algorithms were run by the original authors, whose cooperation is appreciated. Note some geometric distortions and photometric artifacts in the different images, like the bending of straight lines that do not appear in our results.

4.2. Image Rearrangement

Image rearrangement consists of moving an object to a new image location, or deleting part of the image, while keeping some of the content of the image unchanged. The user selects a region to move, and specifies the location at which the selected region will be placed. A new image is generated satisfying this constraint. This application was demonstrated in [5, 14] and gave impressive results in many cases. A failure example of [5] is shown in Fig. 5, together with a successful result given by shift-map.

Object rearrangement is specified in two parts using the data term. One part forces pixels to appear in a new location using Eq. 2. The second part marks these pixels for removal from their original location using Eq. 3 to avoid their duplication in the output. More rearrangement examples are shown in Fig. 6 and Fig. 7.

In image rearrangement pixels can be relocated by a large displacement, creating a possible computational complexity. The need to allow many possible shifts as labels may cause an exponential explosion. In order to reduce this complexity, the set of allowed shifts for each pixel will include local shifts around its original location, plus local shifts around the displaced location.

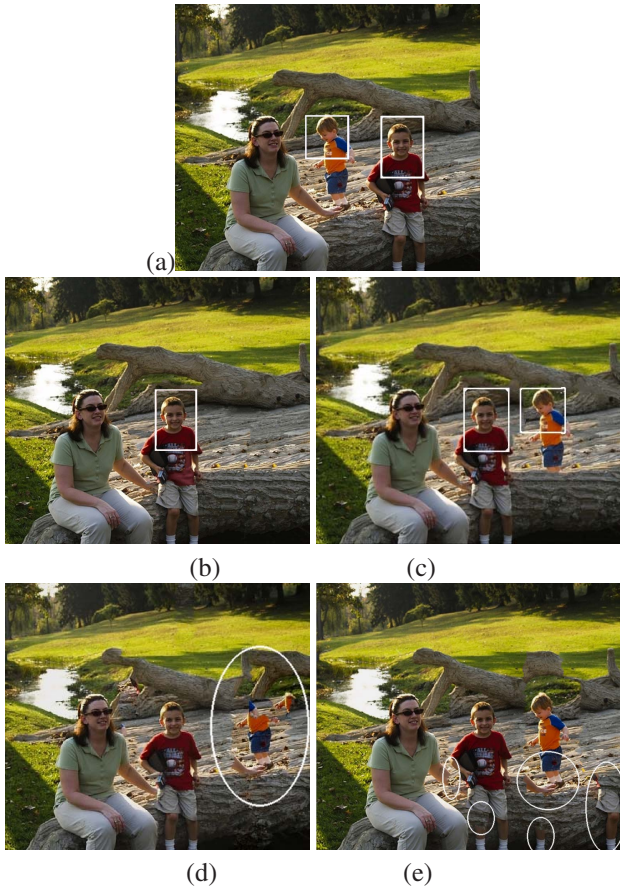
(a) (b) (c)

Figure 5. Image Rearrangement: Comparison of shift-map and patch transform, on a failure case of the patch transform [5]. (a) The original image. (b) The user constraints marked by squares on top of the result given by patch transform: "move the person and a part of the temple to the right, and keep the tourists at their original location in the left bottom corner". (c) **Shift-map result on the same input**.

As the number of labels is growing significantly when there are multiple user constraints, A smart ordering is used for the alpha expansion algorithm of the graph cut [4] to enable fast convergence. The main idea of the alpha-expansion algorithm is to split the graph labels to α and non-α, and perform a min cut between those labels allowing non-α labels to change to α. The algorithm will iterate through each possible label for α until convergence. In the alpha expansion the labels that represent user constraints are considered first, improving the speed and image quality. Since in many cases the user is marking only a small part of the object, first expansion steps on user constraints are getting the rest of the object to its desired location. Alpha expansion on the remaining labels generates the final composition.

4.3. Inpainting

Shift-map can be used for inpainting image regions, a topic extensively studied in computer vision [7, 17, 6]. After interactive marking of unwanted pixels, an automated process completes the missing area from other image regions or from other images. Using shift-maps, the unwanted pixels are given an infinitely high data term as described in Eq. 3. The shift-map maps pixels inside the hole to other locations in the input image. Once the mapping is completed by performing graph cut optimization, the missing pixels are copied from their source location. Most of the existing inpainting algorithm such as [6] are iteratively reducing the size of the hole, and therefore in each step can only make local considerations. The shift-map approach is treating inpainting as a global optimization and therefore the entire filled content is considered at once.



(a) (b) (c) (d) (e)

Figure 6. Image Rearrangement: (a) Original image. Small boy to be removed from the image, big boy to be repositioned to left. (b) **Shift-map results** with the marked user constraints ("move the big boy to the left") on top of the result. (c) Additional rearrangement on the same image: The small boy is re-positioned to the right. (d)-(e) Patch transform results corresponding to (b)-(c). Undesired effects are marked by ellipses.

Examples demonstrating inpainting with shift-maps are shown in Figures 8-9-10-11. Fig. 11 uses a sample image from [18], which suggested that successful removal can be done only with interactive user guidance. Shift-map approach makes a good completion with no user intervention. Inpainting with no user interaction is also done in Fig. 10, an example taken from [15], which also claimed that user interaction is needed to propagate the structure.

In addition of simple inpainting, shift map can also be used for generalized inpainting, where the labels of **all** pixels may be computed, and not only of the pixels in the neighborhood of the hole. This gives increased flexibility to reconstruct visually pleasing images when it is easier to synthesize other areas of the image. However, this approach may change the overall structure of the image as objects and areas have flexibility to move. Fig. 11.(d) demonstrates the

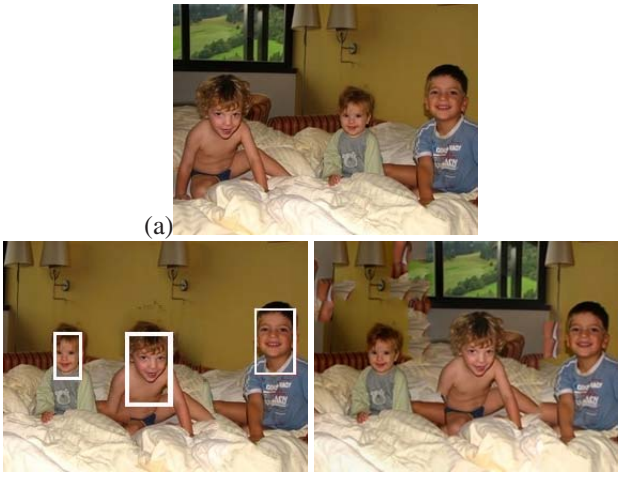


Figure 7. Image Rearrangement: (a) Original image. Kid on the left should move to the center, baby should move to the left, kid on the right should remain in place. (b) **Shift-map results** with user constraints marked on top. (c) Patch transform results on the same input.

generalized inpainting approach, where all image pixels can be shifted. In this example the region of the woman was deleted, and a new region has been synthesized on the right. When some areas should not move and other areas should be removed, user constraints can be added, and inpainting becomes an image rearrangement problem.

4.4. Image Composition

In the shift-map framework the input can consist of either a single image, or of a set of images. If there are multiple input images the shift-map $M(u, v) = (t_x, t_y, t_{ind})$, where t_{ind} is the index of the input image used for each pixel. A very similar labeling for the purpose of creating a collage is described in [12]. It is possible to produce an image rearrangement involving multiple images (selective composition) as was done in "Interactive Digital Photomontage" [1]. In [1] labels were specified only for the source image of each output pixel in the composite image, and therefore the input images had to be perfectly aligned. The shift-map approach is more general, as the label of each pixel consists of both shifting and source image selection. Shift-map can therefore tolerate misalignments between the input images. The resulting composite image can be a sophisticated combination of the input images, as various areas can move differently with respect to their location in the input. An example for image composition is shown in Fig. 14.

5. Concluding Remarks

Shift-maps are proposed as a new framework to describe various geometric rearrangement problems that can be computed as a global optimization.

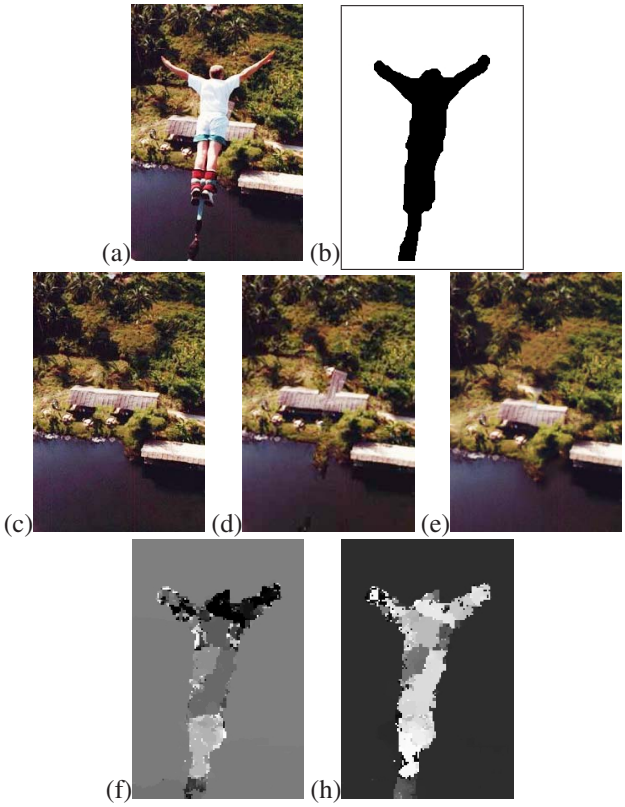


Figure 8. Object Removal: (a) The original image "bungee jumper", taken from [6]. (b) Mask image: black area need to be removed. (c) **Shift-map inpainting**. (d) Comparison to the result of [6] on the same image. (e) Comparison to the result of [14] on the same image. (f-h) Final x and y components of shift-map. Values are scaled for display.

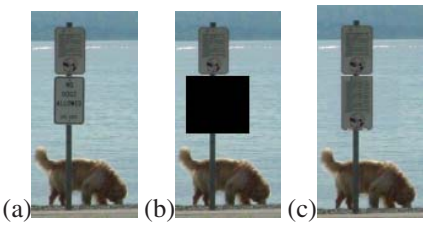


Figure 9. Inpainting example taken from [6]. (a) Original image. (b) The black mask indicates region to be removed. (c) Inpainting removed area by shift-map. The geometric constraints are automatically preserved.

Images generated by the shift map are natural looking, as the method combines several desired properties:

- Minimal and intuitive user interaction, with no need for accurate object selection.
- Distortions that may be introduced by stitching are minimized due to the global smoothness term.
- The geometric structure of the image is preserved, as clearly demonstrated in Fig. 9 and Fig. 10.

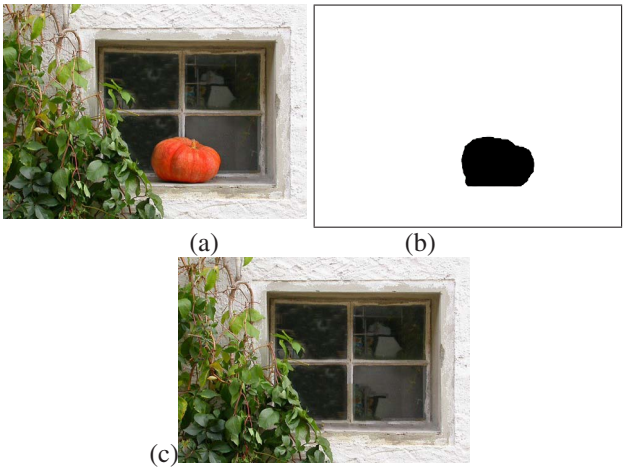


Figure 10. Inpainting example taken from [15], where it was claimed that user interaction is needed to propagate the structure. Shift-map needs no user interaction. (a) Original image. (b) The black mask indicates region to be removed. (c) Completion of removed area by shift-map. The geometric constraints are automatically preserved.

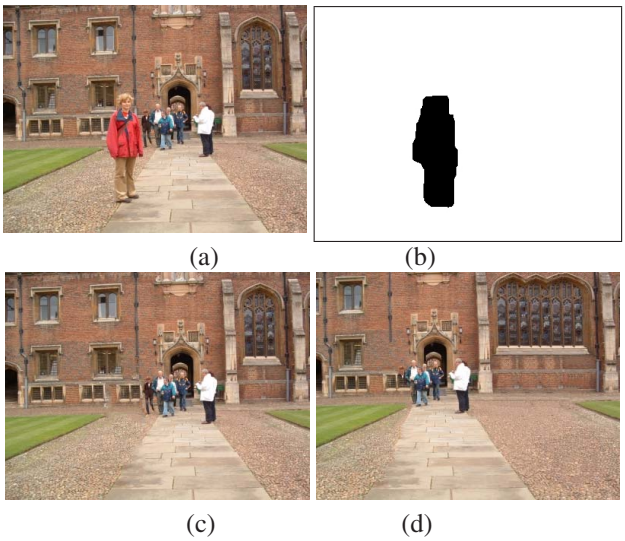


Figure 11. Inpainting using shift-map. (a) Original image from [18]. (b) Black pixels need to be removed. (c) Simple inpainting. (d) Generalized inpainting, where other image pixels are allowed to move. A new region was synthesized on the right.

- Large regions can be synthesized. This appears in all examples, and an isolated demonstration appears in Fig. 12.

Hierarchical optimization resulted in a very fast computation, especially in comparison to related editing approaches. The applicability of shift map to retargeting, inpainting, and image rearrangement was demonstrated and compared to state of the art algorithms.

Although shift-map editing performs well on a large variety of input, it may miss user's intensions. Effects can be controlled by using saliency maps, or by performing the algorithm in several steps.

Extending shift-map to use multiple source images, as described in shift map composition, can also be used for inpainting. Input images can include transformations of the original input image like rotation, scaling etc.

Figure 12. Image expansion using shift-map as texture synthesis. Input images included several rotations of original image. Left: Original; Right: Synthesized.

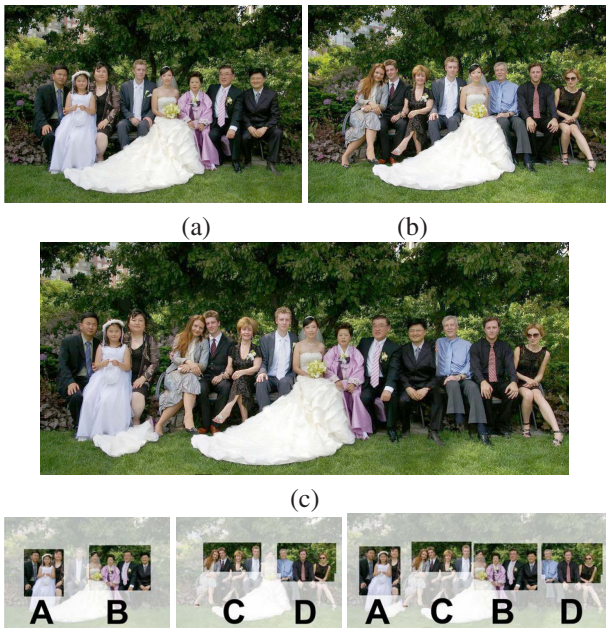


Figure 14. Image Composition. User constrains are given by specifying output locations of selected regions, and other output regions are generated automatically. (a-b) Original images. (c) An image composed from both (a) and (b). (d-e-f) The regions used as user constraints for creating (c) from (a) and (b).

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. In *SIGGRAPH*, pages 294–302, 2004.

[2] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3):10, 2007.

(a) (b) (c) (d) (e)

Figure 13. Comparison to other methods: reducing width by 50%. Soft copy can be magnified for better viewing. (a) Original image. (b) Improved Seam Carving [13]. (c) Video-retargeting [19]. (d) Optimized scale-and-stretch [16]. (e) **Shift-map**.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEET-PAMI*, 26(9):1124–1137, Sept 2004.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEET-PAMI*, 23(11):1222—1239, 2001.

[5] T. Cho, M. Butman, S. Avidan, and W. Freeman. The patch transform and its applications to image editing. In *CVPR'08*, 2008.

[6] A. Criminisi, P. Pérez, and K. Toyama. Object removal by exemplar-based inpainting. In *CVPR'03*, volume 2, pages 721–728, 2003.

[7] J. Hays and A. Efros. Scene completion using millions of photographs. *CACM*, 51(10):87–94, 2008.

[8] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *ECCV'02*, pages 65–81, 2002.

[9] N. Komodakis. Image completion using global optimization. In *CVPR'06*, pages 442–452, 2006.

[10] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *SIGGRAPH'03*, pages 277–286, 2003.

[11] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *ICCV'05*, volume 1, pages 259–265, 2005.

[12] C. Rother, L. Bordeaux, Y. Hamadi, and A. Blake. Autocollage. In *SIGGRAPH'06*, pages 847–852, 2006.

[13] M. Rubinstein, A. Shamir, and S. Avidan. Improved seam carving for video retargeting. In *SIGGRAPH'08*, pages 1–9, 2008.

[14] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR'08*, 2008.

[15] J. Sun, L. Yuan, J. Jia, and H. Shum. Image completion with structure propagation. In *SIGGRAPH'05*, pages 861–868, 2005.

[16] Y. Wang, C. Tai, O. Sorkine, and T. Lee. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph.*, 27(5):1–8, 2008.

[17] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. *CVPR'04*, 1:120–127, 2004.

[18] M. Wilczkowiak, G. Brostow, B. Tordoff, and R. Cipolla. Hole filling through photomontage. In *BMVC*, pages 492–501, 2005.

[19] L. Wolf, M. Guttmann, and D. Cohen-Or. Non-homogeneous content-driven video-retargeting. In *ICCV'07*, 2007.