Appeared in ICCP 2012 (IEEE Int. Conf. on Computational Photography)

Alignment and Mosaicing of Non-Overlapping Images

Yair Poleg Shmuel Peleg
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel

Abstract

Image alignment and mosaicing are usually performed on a set of overlapping images, using features in the area of overlap for alignment and for seamless stitching. Without image overlap current methods are helpless, and this is the case we address in this paper. So if a traveler wants to create a panoramic mosaic of a scene from pictures he has taken, but realizes back home that his pictures do not overlap, there is still hope.

The proposed process has three stages: (i) Images are extrapolated beyond their original boundaries, hoping that the extrapolated areas will cover the gaps between them. This extrapolation becomes more blurred as we move away from the original image. (ii) The extrapolated images are aligned and their relative positions recovered. (iii) The gaps between the images are inpainted to create a seamless mosaic image.

1. Introduction

Mosaicing several images to give a panoramic view has been extensively studied, and the technology has matured enough to be performed in real time inside many cameras. The area of overlap between the pictures is critical for these methods. Features in the overlap area are used for image alignment, and once aligned the overlap area enables seamless stitching. We will skip a survey of mosaicing since it is very common, and interested readers can look at [15].

But what happens when no overlapping areas exist, as illustrated in Fig. 1? A simple case of non-overlap is addressed by square jigsaw puzzle solvers, such as [12, 8], where an image is to be reconstructed from non-overlapping square patches. In [12] a single pixel extrapolation is used to determine the positions of the patches. Placement and stitching are simple since the finite canvas size is known in advance, and the patches are contiguous and have a relatively small number of possible positions. In contrast to [12], in our case the relative placements, the gap widths, and the canvas size are all unknown.

People are very good at alignment of non-overlapping images. Linear features such as skyline and roads provide important clues. Prior knowledge about the world, such as the knowledge that sky and clouds are normally above the ground, is also useful. It is conceivable that detecting linear features and aligning images such that these are continued smoothly could align successfully non-overlapping images. Instead of taking this non-trivial task we want to examine the power of much simpler, intensity based methods, involving no feature detection or scene understanding.

Our simple attempt includes three stages. The first stage is image extrapolation, hoping that such extrapolation will cover the gaps between the images. Image extrapolation is covered in Sec.2. The second stage is the alignment of the extrapolated images, presented in Sec. 3. The third stage described in Sec. 4 is the filling of the gaps between the images. Examples are shown in Sec. 5 and final remarks are made in Sec. 6.

The main contribution of this paper is presenting the problem of aligning and mosaicing non-overlapping images, and showing that in some cases it can have a simple solution. In addition, a new approach to alignment of multiple images is presented.

2. Image Extrapolation

While image inpainting is very common [9, 13, 1], image extrapolation is not as common. Early usage was to allow image filtering without reducing the image size [16]. In [11] video extrapolation was used to avoid cropping when performing video stabilization. In [5, 3] video extrapolation is performed to improve viewing experience of video. Our approach to image extrapolation is similar to [3] applied to single images rather than to video. Note that traditional image inpainting is done to please the human viewer and inpainted regions should look indistinguishable from the original image. Image extrapolation is used in our case to enable alignment, and "hallucinated" high-frequency information such as produced by [10], should be avoided. Since the Fourier transform of the autocorrelation function is the power spectrum, it is clear that the autocorrelation of high

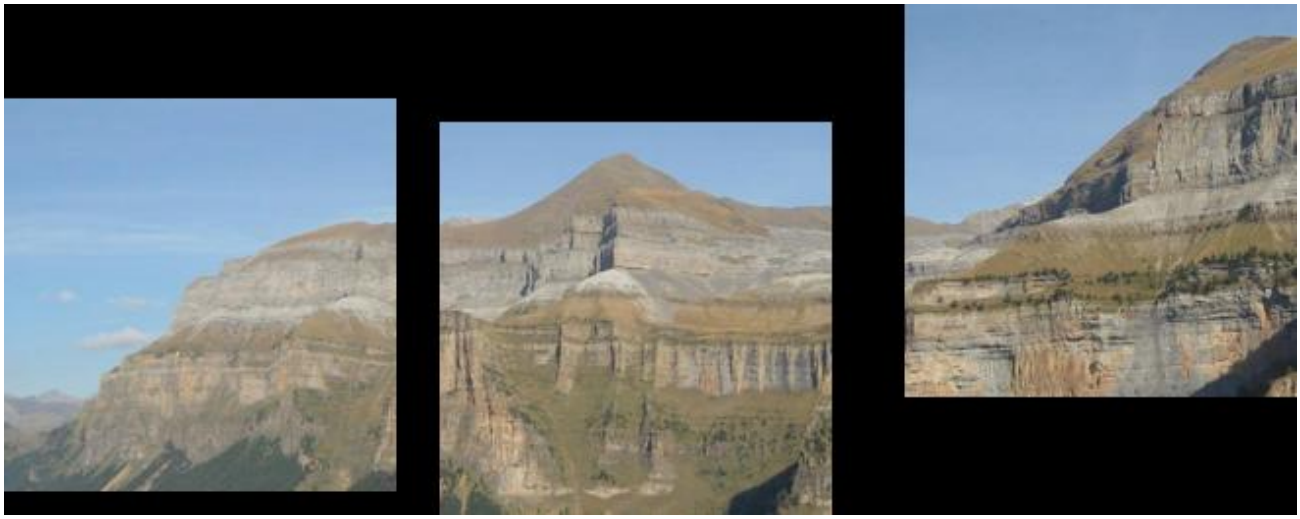


Figure 1. A set of non-overlapping images which we would like to align and to mosaic. While image locations have been computed using our proposed alignment method, input images are given unordered and without any position information.

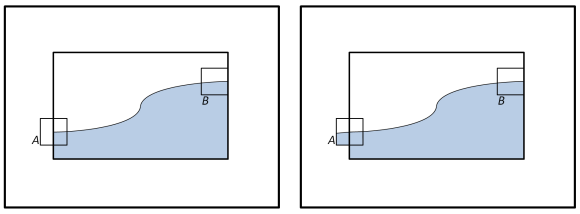
frequencies decays much faster than the autocorrelation of lower frequencies. Therefore, high frequencies should be reduced in extrapolated regions as we get further away from the original image.

Extrapolation using a multi-resolution Gaussian Pyramid [2] gives the appropriate reduction of high frequencies. Extrapolation starts at the coarsest level (smallest pyramid image), and continues to higher resolution levels until the full image resolution is reached (See Fig. 3). As desired, the obtained extrapolation has higher frequencies closer to the given image, and reduced details further away from the image.

2.1. Single Scale Extrapolation

In this section we describe image extrapolation as performed in a single level of the Gaussian pyramid. This is a patch based method, and is similar to [9, 3], whose basic step is shown in Fig. 2. Let A be a rectangular region whose right half is inside the image and left half is outside the image. To extrapolate the pixels in the left half of A that are outside the image we search for a patch B in the image whose right part is most similar to the right part of A. Once such a patch is found, we copy to the pixels in the left part of A the colors of the corresponding pixels in the left part of B.

When we create a mosaic from a set of images we use the knowledge that all images are from the same scene, and search for the closest patch in all images, as well as in all levels in their Gaussian pyramids. Copying from one scale into another scale represents the assumption of a fractal world, and enables to use similar areas that are at different distances from the camera.



(a) (b)

Figure 2. (a) The right side of the patch A is inside the original image and its left side is outside the image. To extrapolate the left part of A we search in the image for a patch B whose right side is most similar to the right side of A. (b) The left side of patch B is used to paint the left side of patch A.

To extrapolate an image by k pixels in each direction we start by centering a patch of size $2k \times 2k$ around every boundary pixel (we used $k = 5$). Half of such a patch is inside the image and half is outside the image and needs extrapolation. For each such patch on the boundary a most similar patch is sought from all scales of all images as in Fig. 2. Patch similarity is inverse to the sum of distances of all corresponding pixels. The distance between two pixels p and q is is computed in the LAB color space, and is based on the difference in each of the L, A, and B color components:

$$D(p,q) = \sqrt{(p_L - q_L)^2 + (p_A - q_A)^2 + (p_B - q_B)^2} \quad (1)$$

Each pixel in the extrapolation area is covered by $2k$ patches. In our case, where $k = 5$, each pixel is covered by 10 extrapolations, each centered around a different bound-

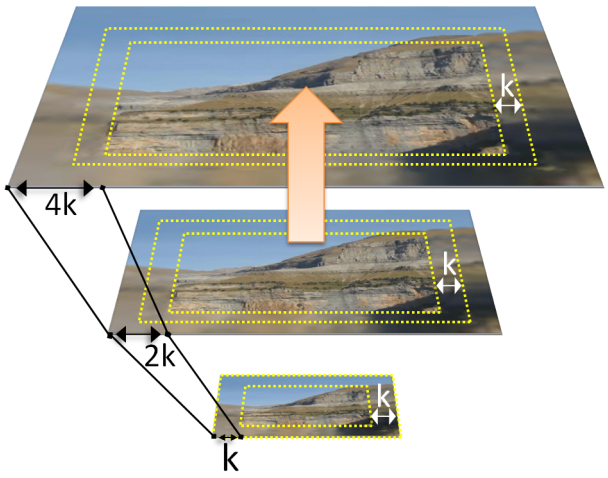


Figure 3. Multi scale extrapolation, using a multi-scale pyramid built for an image. The coarsest scale (smallest pyramid level) is extrapolated by k pixels in each direction using the closest patches approach. The extrapolated region is magnified by 2, and is attached to the higher resolution level. At that higher level we extrapolate again the k closest pixels to the image, updating the extrapolated region coming from the lower resolution. The other pixels that were magnified from lower resolution levels are not updated. This process continues to higher resolutions until the full image resolution is reached.

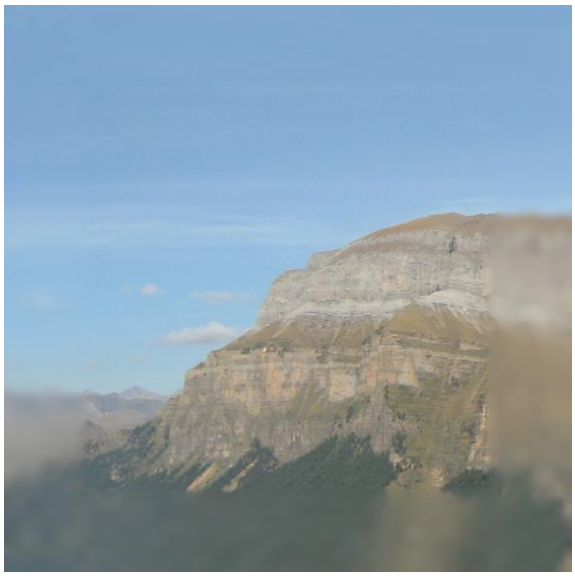


Figure 4. An example of an extrapolated image generated by three steps of iterative multi-scale extrapolation. The extrapolated region is blurred. We did not draw a line around the the original image as it conceals the transition from real to extrapolated parts.

ary pixel. The extrapolated value of a pixel is computed by averaging all overlapping patches covering it, and this average is done in a feathered way, giving higher weight to points closer to centers of patches. This patch averaging is done to avoid sharp seams, and is different from [3], where non-overlapping patches were used.

2.2. Multi-Scale Extrapolation

We perform the image extrapolation in a multi-scale approach as shown in Fig. 3. We first build a multi-scale Gaussian pyramid for each image. We start the extrapolation from the smallest pyramid level (coarsest scale). At this level we extrapolate a band of k pixels around the image using the approach described in Sec. 2.1. Once the extrapolation of the smallest level is complete, we continue to the higher resolution level.

The extrapolated strip is magnified by 2, and is placed around the image at the higher resolution level. The magnification blurs the extrapolated strip. The k pixels closest to the current pyramid level are extrapolated again, this time from the image at the current pyramid level. To preserve continuity and to avoid outliers we use also the extrapolated values of the target patch, as computed from the coarser scale, when searching for the closest patch. The source patches, though, are only patches from given images and their pyramids, and do not include areas generated by extrapolation in previous steps.

We continue the extrapolation from each level to the higher resolution level until the original image resolution is reached. If the pyramid had l levels above the original image, the extrapolated area is of width $k * 2^l$ in each direction. The k pixels closest to the source image have the highest resolution, and as we move away from the image the extrapolated areas are more blurred due to multiple magnifications.

2.3. Iterated Multi-Scale Extrapolation

The width that a multi-scale extrapolation can add to an image is limited to $k * 2^l$ in each direction, where k is the single-level extrapolation width and l is the number of pyramid levels. Larger extrapolation can be done in several ways. One option is to increase k (we currently use $k = 5$). We found that using a large k brings undesirable artifacts as shown in Fig. 5. Another approach to increase the extrapolated area is to repeat the entire extrapolation step again, and extrapolate the already extrapolated images. Since the already extrapolated areas are very smooth, we do not want to include high frequency details in the extrapolated areas. We therefore use blurred versions of the images for creating the additional extrapolated areas. An example of an extrapolated images after three iterations is shown in Fig. 4.

(a) (b) (c)

Figure 5. An example of an extrapolated image generated with different values of k. (a) Input image. (b) Extrapolation with $k = 5$. The horizon is extrapolated as expected. (c) Extrapolation with $k = 15$. The extrapolation duplicates the peak of the mountain.

3. Alignment of Extrapolated Images

Given two extrapolated images we align them in a similar way to the alignment of regular images, but we need to take into account that the areas of overlap will now be extrapolated areas rather than real image areas, and that the overlap area will be relatively small. The cost function used for the alignment will take this into account. The methodology for multiple image alignment initially places all images at the same location on top of each other, and images are iteratively shifted to reduce the global cost function.

The proposed simultaneous alignment of multiple images assumes that most of the time we have overlap areas only between two images. This justifies the use of a simple cost function between two images as done in this paper. Methods exist for alignment of multiple overlapping images [14], but they are not applicable in our case.

3.1. Cost Function: Alignment of Two Images

Like the cost function between patches (Eq. 1), the cost function comparing two extrapolated images is also based on a weighted sum of pixel distances for each of their LAB color components. The distance between two pixels p and q is now

$$D_\alpha(p,q) = \sqrt{\alpha(p_L - q_L)^2 + (p_A - q_A)^2 + (p_B - q_B)^2}, \tag{2}$$

where in our experiments $\alpha = 0.5$. The distance between two images P and Q, after the suggested shift has been applied to them, is the weighted sum over their overlapping area divided by the area of overlap $\Omega = P \cap Q$:

$$Cost(P,Q) = \frac{\sum_{(x,y)\in\Omega} D_\alpha(P(x,y), Q(x,y))W(x,y)}{|\Omega|}, \tag{3}$$

where (i) The sum over $(x,y) \in \Omega$ uses the global coordinate system after alignment, and (ii) $W(x,y)$ is a weight function that increases the cost between pixels with no (or low) gradients in RGB. $W(x,y)$ is computed as follows:

$$W(x,y) = 1 - \beta\|\nabla_p(x,y)\|\|\nabla_q(x,y)\|, \tag{4}$$

Where β was found to be best for values in the range $[0.1, 0.3]$.

3.2. Batch Alignment of All Images

In batch alignment we try to find an arrangement of all input images that will minimize the sum of all pairwise image costs. This is performed in an iterative process: In the initial arrangement all images are placed on top of each other, i.e. all input images initially have the same location. In each iteration step we examine all possible translations for each image (within a limited range, say 20 pixels), and select the image and its translation giving the largest cost reduction. Iterations repeat until the cost can not be reduced any further.

Similar methods of multi resolution search of multiple possible translations were traditionaly used for alignment of two images. Why will it work for multiple image alignment? The insight comes from [7], addressing image alignment when each image has two different motions (e.g. transparency). It was shown in that paper that accurate motion can be computed using an ordinary two-frame method, even when multiple motions are present, since one motion always becomes dominant, and the other motions do not spoil the solution. In our case it means that if we compute the error at several shifts of two unrelated images, the error will not change much. But when the two images are related, a shift towards the correct motion will reduce the error substantially. When adding all errors from all images, only the related images will affect the location of the minimum.

A visualization of the convergence of batch alignment is shown in Fig. 6. It should be mentioned that the greedy minimization described above can converge to a local minimum, and no movement of a single image could reduce the energy. This is the case where methods like simulated annealing may overcome such local minima. All our experiments converged successfully without simulated annealing.

3.3. Handling Rotations

The method described so far assumed pure translations between images. In reality, most images taken with handheld cameras have some rotations. We have found that such small, unintentional rotations (around 2°) do not hurt the alignment and the quality of the final results. Unlike ordinary mosaicing where rotation errors are clearly visible, in our case the inpainting stage (describe in Sec. 4) serves to conceal discrepancies.

In order to cope with larger rotations, such as in Fig. 7, adjustments need to be made to the batch alignment process

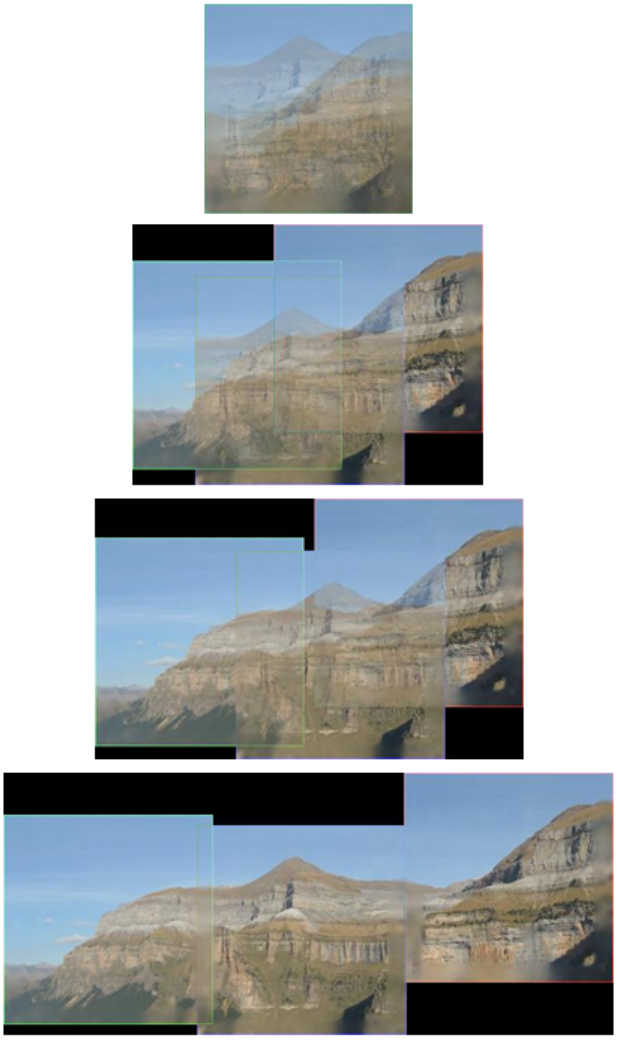


Figure 6. Visualization of the convergence of batch alignment. The top image shows the initial step where all three extrapolated images are superimposed at the same location. With every iteration the extrapolated images slide to their correct relative locations, and the bottom image shows the image locations after convergence. The gaps are not visible since the images are extrapolated. Once the extrapolated regions are removed the gaps are revealed as in Fig. 1.

described in Sec. 3.2. A simple solution to this problem might be to add a few rotations to the search for optimal alignment. While this solution might work well, it comes with a substantial increase in the complexity of the search. Instead, we propose a method for aligning multiple images with rotations that takes the observations made in Sec. 3.2 one step further.

The proposed alignment approach has three steps. We begin by integrating each input image with rotated copies

Figure 7. Extrapolated input images with 15° rotations. It should be noted that the original images have a small area of overlap that does not interfere with the alignment process.

(a) (b) (c)

Figure 8. Creating integrated images. Each row shows the process for a different input image. (a) Input image. (b) one of 21 rotated copies of the input image. (c) Integrated image created from 21 rotated copies of the input image.

of itself, as shown in Fig. 8. We proceed to batch alignment of the integrated images, as described in Sec. 3.2. Once we found the approximate alignment, we perform a local search for the best rotation for each pair of images. We will now describe each step in detail.

The first step is building an integrated image $\hat{I}_i$ for each input image P_i of the N input images. Each such image is an integration of an input image and rotated copies of the input image. The angles of the rotated copies we integrate depend on the maximal rotation angle θ_{max} we expect in the input. The rotated copies we create cover the range $[-\theta_{max}, \theta_{max}]$ in steps of ω degrees. Formally, let us denote the set of rotation angles by S:

$$S = \bigcup_{j=0}^{\lceil \frac{2\theta_{max}}{\omega} \rceil} \{-\theta_{max} + j\omega\}, \tag{5}$$

and $S_0 = S \cup \{0\}$. Then the integrated image $\hat{I}_i$ is defined by the average of all rotated images:

$$\hat{I}_i(x, y) = \frac{1}{|S_0|} \sum_{\alpha \in S_0} P_{i,\alpha}(x, y), \tag{6}$$

where $P_{i,\alpha}$ is the input image P_i rotated by α degrees. The sum is done for each RGB channel separately. The values

of θ_{max} and ω define the amount of rotated images we integrate, but they also affect the next steps, as we describe next.

The second step is batch alignment of the integrated images as described in Sec. 3.2. We have found that the alignment process fails for high values θ_{max} or low values of ω. This is because high values of θ_{max} or low values of ω produce integrated images that do not contain any useful data. The resulting integrated images are too blurred and smeared to tell one from another. We have successfully experimented with $\theta_{max} \leq 21^\circ$ and $\omega \geq 3^\circ$. Fig. 9 shows an alignment result of integrated images.

The third step is selecting a rotation angle $\alpha \in S_0$ for each input image. We scan the alignment result for pairs of overlapping integrated images. For each overlapping pair $\hat{I}_i$ and $\hat{I}_j$, we search for $\hat{\alpha_i}$ and $\hat{\alpha_j}$ such that

$$(\hat{\alpha_i}, \hat{\alpha_j}) \in \underset{(\alpha_i,\alpha_j)\in S_0 \times S_0}{\arg\min} Cost(P_{i,\alpha_i}, P_{j,\alpha_j}), \qquad (7)$$

where the cost function is as defined in Eq. (3). We add $\hat{\alpha_i}$ as candidate rotation angle for P_i, and $\hat{\alpha_j}$ as a candidate rotation angle for P_j. When we are done, we have a list of candidate rotation angles for each input image. In many cases this list contains one candidate angle per input image. This is because both the alignment process and the rotation selection process are based on the same pairwise cost. In these cases, we simply replace each integrated image $\hat{I}_i$ by a rotated copy of the input $P_{i,\alpha}$. Fig. 10 shows the final result of the alignment process, after replacing the integrated images with rotated versions of the input images. Fig. 11 shows a complete example over three images.

If there is more than one rotation angle candidate for some input image, we can do one of the following: (i) Choose the most popular candidate. (ii) Calculate a weighted average of all the candidate angles and use the result. (iii) Perform a global search for the best angle for each image. A brute force approach would require evaluation of all $|S_0|^N$ permutations in the worst case. We can, however, narrow down the search space by searching only around candidate rotation angles found earlier.

3.4. Multi Resolution Alignment

As usual in direct image alignment methods [4, 6], alignment is performed in a multi-resolution approach. Gaussian pyramids are built for the extrapolated images, and the alignment starts at the lowest resolution. At that level all possible translations and rotations are examined. Once an optimal alignment is found at a low resolution level, a higher resolution level is examined. When moving from a lower resolution to a higher resolution, all translations found so far are scaled to the new level. In our case they are multiplied by two. The search for the minimum is now performed only around the previously found translations and rotations.

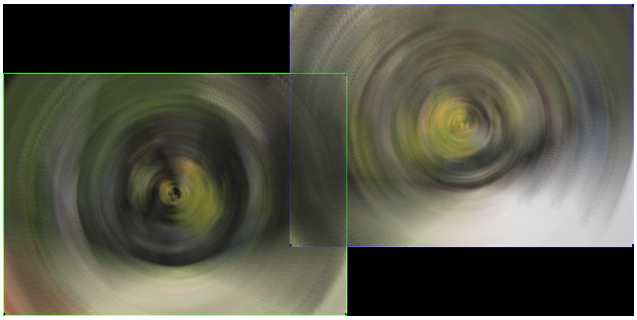


Figure 9. The alignment recovered for the integrated images in Fig. 8.c.

Figure 10. Final result, after alignment using integrated images and inpainting. The integrated images showed in Fig. 9 replaced with rotated copies of the input images and the gaps are inpainted.

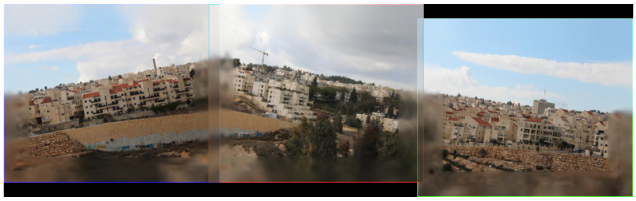
4. Inpainting the Gaps

Once the images are arranged properly, the gap between them needs to be inpainted in order to create a seamless panoramic image. Any of the many existing inpainting tools will work. We found the inpainting software in [13, 1] very easy to use.

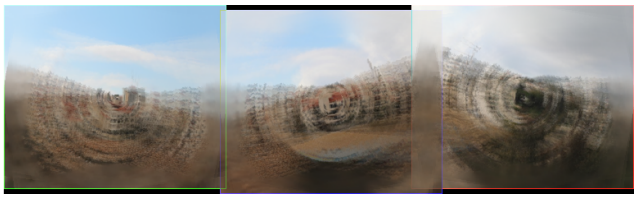
5. Experimental Results

The input images for the mountains scene in Fig.1 were cut from a high resolution image. Fig. 6 visualizes the convergence of the alignment process and Fig. 12 shows the seamless inpainted panorama alongside the ground truth image. The normalized RMS error of the image locations from ground truth is 0.041 in the X-axis and 0.007 in the Y-axis. The normalization is with respect to the geometric mean of the ground truth image dimensions. It is reasonable that the Y alignment is more accurate than the X alignment in this panoramic case, and the main achievement in the X alignment is the correct ordering of the images.

(a)



(b)



(c)

(d)

Figure 11. (a) Input images. (b) Alignment result without the use of integrated images. (c) Alignment result using integrated images. (d) Integrated images replaced by the best rotation angle image. While there are some errors in the horizontal alignment, the vertical alignment and the rotation angles are approximately correct.

The experiment in Fig. 14 shows mosaicing in a 2×2 grid structure of the input images. As in the previous experiment, the input images here were cut out from a high resolution image as well. The normalized RMS error of the image locations from ground truth is 0.056 in the X-axis and 0.074 in the Y-axis.

Our last example is presented in Fig. 13. The input images were shot with a hand-held camera. The ground truth data for this experiment comes from a normal panoramic mosaic image (with overlap...). The normalized RMS error of the image location from ground truth is 0.053 in the X-axis and 0.016 in the Y-axis. In this panoramic case it is also reasonable that the Y alignment is more accurate than the X alignment, and the main achievement in the X alignment is the correct ordering of the images.

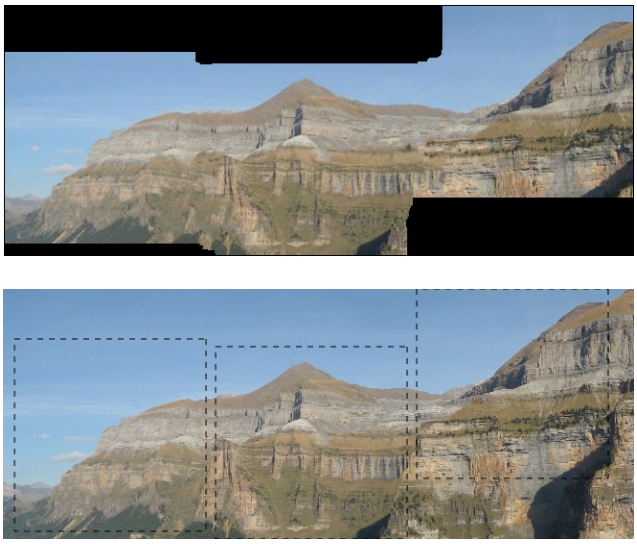


Figure 12. Top: A panoramic image created after inpainting the gaps as shown in Fig. 1. Bottom: Ground truth image from which the input images were taken.

(a) (b)

(c) (d)

Figure 14. A 2×2 example: (a) Input images. (b) Input images after alignment. (c) Final mosaic after inpainting. (d) Ground truth image from which the input images were taken.

Figure 13. Panoramic input images shown after alignment, and the generated panoramic image created from them by inpainting.

6. Concluding Remarks

Automatic alignment and mosaicing of non-overlapping images has been introduced. The general scheme includes steps of image extrapolation, alignment of extrapolated images, and inpainting the gaps. We believe that the described approach will work well even if one or more of the three steps will be implemented in methods that are different from those described in this paper. Some of our experiments were performed assuming a pure translation between images, and the inpainting process conceals small rotations. We have also introduced an approach for handling relative rotations between images.

We must admit that we were surprised by the success of the simple batch alignment of multiple images, and from the success of integrated images in aligning rotated images. We hope that this simple approach can be used for different tasks where alignment of feature points may not be possible.

7. Acknowledgement

This work was supported by the Israel Science Foundation and the Ministry of Science and Technology.

References

[1] Content Aware Fill, Adobe Photoshop CS5.

[2] E. Adelson, C. Anderson, J. Bergen, P. Burt, and J. Ogden. Pyramid methods in image processing. *RCA Engineer*, 29:33–41, 1984.

[3] A. Aides, T. Avraham, and Y. Schechner. Multi-scale ultrawide foveated video extrapolation. In *ICCP*, 2011.

[4] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

[5] T. Avraham and Y. Schechner. Ultrawide foveated video extrapolation. *IEEE Journal of Selected Topics in Signal Processing*, 5:321–334, 2011.

[6] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, pages 237–252, 1992.

[7] J. Bergen, P. Burt, R. Hingorani, and S. Peleg. A three frame algorithm for estimating two-component image motion. *IEEE Trans. PAMI*, 14:886–896, 1992.

[8] T. S. Cho, S. Avidan, and W. T. Freeman. A probabilistic image jigsaw puzzle solver. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[9] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based inpainting. *IEEE Trans. Image Processing*, pages 1200–1212, 2004.

[10] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *In International Conference on Computer Vision*, pages 1033–1038, 1999.

[11] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H. Shum. Fullframe video stabilization with motion inpainting. *IEEE Trans. PAMI*, pages 1150–1163, 2006.

[12] D. Pomeranz, M. Shemesh, and O. Ben-Shahar. A fully automated greedy square jigsaw puzzle solver. In *CVPR*, 2011.

[13] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *ICCV'09*, pages 151–158, Kyoto, Sept 2009.

[14] H. Sawhney and R. Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. *IEEE Trans. PAMI*, pages 235–243, 1999.

[15] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2:1–104, Dec 2006.

[16] B. White and D. Brzakovic. Two methods of image extension. *Computer Vision, Graphics, and Image Processing*, 50:342–352, 1990.