To Appear in IJCV

Minimal Aspect Distortion (MAD) Mosaicing of Long Scenes *

Alex Rav-Acha Giora Engel Shmuel Peleg

School of Computer Science and Engineering

The Hebrew University of Jerusalem

91904 Jerusalem, ISRAEL

Abstract

Long scenes can be imaged by mosaicing multiple images from cameras scanning the scene. We address the case of a video camera scanning a scene while moving in a long path, e.g. scanning a city street from a driving car, or scanning a terrain from a low flying aircraft.

A robust approach to this task is presented, which is applied successfully to sequences having thousands of frames even when using a hand-held camera. Examples are given on a few challenging sequences. The proposed system consists of two components: (i) Motion and depth computation. (ii) Mosaic rendering.

In the first part a ”direct” method is presented for computing motion and dense depth. Robustness of motion computation has been increased by limiting the motion model for the scanning camera. An iterative graph-cuts approach, with planar labels and a flexible similarity measure, allows the computation of a dense depth for the entire sequence.

In the second part a new minimal aspect distortion (MAD) mosaicing uses depth to minimize the geometrical distortions of long panoramic images. In addition to MAD mosaicing, interactive visualization using X-Slits is also demonstrated.

*This research was supported by the Israel Science Foundation. Video examples and high resolution images can be viewed in *http://www.vision.huji.ac.il/mad/*

1 Introduction

Many mosaicing applications involve long image sequences taken by translating cameras scanning a long scene. This includes a video camera mounted on a vehicle scanning city streets [25, 1, 30, 35, 32], or a video camera mounted on a low altitude aircraft scanning a terrain [36]. We present a direct method to compute camera motion and dense depth that are needed for this mosaicing. The computed motion and depth information are used for two visualization approaches. (i) A new Minimal Aspect Distortion (MAD) mosaicing of the scene. (ii) An immersive 3D visualization using X-Slits [37].

MAD mosaicing is using the motion and depth information to create mosaic images where distortion of objects is minimal. Mosaicing from translating cameras normally gives pushbroom distortions, where far away objects become wider and closer objects become narrower. In MAD mosaicing the appearance of objects in the mosaic is closer to their appearance in the original images, avoiding as much as possible the pushbroom distortion. While MAD mosaicing reduces local distortions, it may not be convenient for the generation of new views. This can be done by X-Slits mosaicing, allowing interactive changes of viewpoint and viewing direction.

The effectiveness of the proposed approach is demonstrated by processing several video sequences taken from moving cars and from helicopters. Long MAD mosaics, and a few fly through videos using X-Slits, are presented online in *http://www.vision.huji.ac.il/mad*.

While the proposed computation of camera motion is accurate, depth may not be computed accurately for many scene points. Occlusions, low contrast, varying illumination, and reflections will always leave many scene points with no depth values, or even with a wrong depth. Traditional model based rendering approaches may fail when depth is not accurate everywhere. But since image based rendering such as MAD mosaicing and X-Slits projection use only statistics of depth, such problems can be avoided.

The proposed approach uses only image data, and does not require external motion information. If motion and/or depth information is available, e.g. when using a GPS or laser scanners, [19, 25], it could replace or enhance the motion computation part.

1.1 Overview of Proposed Approach

Panoramic images of long scenes, generated from images taken by a translating camera, are normally distorted compared to perspective images. When large image segments are used for stitching a panoramic image, each segment is perspective but the seams between images are apparent due to depth parallax. When narrow strips are used the panoramic image is seamless, but its projection is normally pushbroom, having aspect distortions. The distortions become very significant when the variations in scene depth are large compared to the distance from the camera.

In Section 5 we present ”MAD Mosaicing”, which is a long mosaic having minimum aspect distortions. It is based on the observation that only the perspective projection is undistorted in a scene with large depth variations, while for a scene at a constant depth almost any projection can give an undistorted mosaic. MAD mosaicing changes the panoramic projection depending on scene structure at each location, minimizing two costs: A distortion cost and a stitching cost. Minimization is done using dynamic programming.

An alternative visualization method is the X-Slits projection, described in Section 6. Even though X-Slits images are more distorted, they enable to control the viewpoint and create fly through sequences. In this case the depth information is mostly needed when handling large displacements, when the stitching process requires better interpolation.

Knowledge of camera motion and dense depth is needed for mosaicing, and this information can be provided by any source. A robust approach is proposed, alternating between direct ego-motion computation (Section 3) and depth computation (Section 2). This results in a robust method to compute both motion and depth, which can overcome large disparities, moving objects, etc. A general description of the alternation between motion and depth computation is described in Section 4.

By limiting our analysis to the most common case of a camera moving sideways, motion and depth can be computed robustly for otherwise ambiguous sequences. We can robustly handle cameras mounted on moving cars and scanning city streets, down-looking cameras scanning the ground from a low altitude aircraft, etc. The camera is allowed, of course, to rotate, as rotations are common in such cases due to the vibrations of the vehicle.

The computation of camera motion uses a simple variation of the Lucas-Kanade method [5] that takes into account the estimated scene depth. Given the estimated motion, depth is computed using a graph cuts approach to detect planar surfaces in the scene. In long image sequences, planar surfaces that were computed for previous frames are used as priors for the new frames, increasing robustness. Additional robustness is achieved by incorporating a flexible pixel dissimilarity measure for the graph cuts method. The variations of motion and depth computations that allowed handling inaccurate inputs are described in Sections 3 and 2.

A note about notation: We use the terms "depth" and "inverse depth" when we actually refer to "normalized disparity": The horizontal disparity due to translation, divided by the horizontal camera translation. This normalized disparity is proportional to the inverse of the depth. The exact meaning will be clear from the context.

1.2 Related Work

One of the declared benefits of the X-Slits mosaicing is its reduced distortion compared to pushbroom [10]. But for very long mosaics, the X-Slits images become very close to pushbroom with its significant distortions. Attempts to reduce the distortion of the long mosaic were presented in [30, 31], using different X-Slits projections for different scene segments. This work inspired MAD mosaicing, which extends the piecewise constant approach to a continuous variation of X-Slits. Another source of inspiration is [33], where a mosaic image is generated by minimizing a stitching cost using dynamic programming. Other papers on mosaicing of long scenes include [32, 35], where long mosaics are generated from a narrow slit scanning a scene. In these papers the camera is assumed to move slowly in a roughly constant velocity, and the scene depth can be estimated from stationary blur. In [2] a long panorama is stitched from a sparse set of still images, mainly addressing stitching errors.

The motion and depth computation in this paper is closely related to many intensity based ("direct") ego motion computations such as [16, 13]. While these methods recover unrestricted camera motions, they are relevant only for short sequences since they require an overlapping region to be visible in all frames. For mosaicing long sequences, a robust motion computation method is required, which can also handle degenerate cases where general motion can not be recovered.

In particular, the algorithm should give realistic mosaic images even when no 3D information is available in large portions of the scene. Earlier versions of our work on ego-motion computation for sideways moving cameras were proposed in [29, 27]. They had initialization and robustness problems that are addressed in this paper. In addition, they did not address the computation of dense depth maps and the creation of undistorted mosaics.

In [1, 30] methods are described for creating a multi-perspective panorama. These methods recover camera motion using structure-from-motion [14], matching features [23] between pairs of input images. Matched points are used to recover the camera parameters as well as a sparse cloud of 3D scene points, recovery that is much easier when fisheye lens are used as in [1]. This is an opportunity to discuss the differences between our approach and earlier work. The first difference is the use of a "direct" method rather than using feature point matches. Direct methods can be preferred when feature points may be rare, ambiguous, or noisy. Feature points will be preferred in clean, high contrast, and unambiguous imagery.

All mosaicing work can be regarded as a special case of creating a full model of the observed scene [26]. Having multiple images of the scene theoretically enables both the computation of camera parameters and the geometric and photometric structure of the scene. As the mosaicing process is much simpler than the creation of a scene model, it is likely to work in more cases. Mosaicing works especially well when long scenes are involved, having motion only in one direction. Even when a scene model has successfully been constructed, the generation of a very long panoramic image of the entire scene, having minimum distortion, is a challenging problem.

For depth computation we use a variation of the iterative graph cuts approach as described in [20], which is based on [21, 8]. Instead of extracting constant disparities, we segment the image into planar surfaces. Combining the graph cuts approach with planar surfaces was described in [7, 15]. The main differences between [7] and our method is in the initialization of the planar surfaces and in the extension of the two-frames algorithm to long un-stabilized sequences. A more detailed discussion can be found in Section 8.

2 Graph Cuts for Depth Computation (Assuming Known Motion)

Given a stereo image pair, depth can be computed by finding for each pixel in the left image its corresponding pixel in the right image. Many methods improve stereo computation by incorporating the depth consistency between neighboring points. A method that gives excellent results uses a graph cuts approach to compute depth from stereo [8]. We start by briefly describing the basic formulation of the graph cuts approach. In Sections 2.1 and 2.2 we describe our variants on this formulation: defining a flexible data penalty, and using planar surfaces instead of constant disparities.

Let L be the set of pixels in the left image. In the common graph cuts approach to stereo each pixel p in the left image is labeled with its disparity f_p. A desirable labeling f usually minimizes the Potts energy [8]:

$$E(f) = \sum_{p \in L} C_p(f_p) + \sum_{p,q \in N} V_{p,q} \delta(f_p \neq f_q), \tag{1}$$

where $C_p(f_p)$ is a cost for the pixel p to have the disparity f_p based on image pair similarity, N denotes the set of pixel pairs in the left image which are in a neighborhood and $\delta(\cdot)$ is 1 if its argument is true and 0 otherwise. Each $V_{p,q}$ represents a penalty for assigning different disparities to neighboring pixels p and q. The value of the penalty $V_{p,q}$ is smaller for pairs $\{p, q\}$ with larger intensity differences $|I_p - I_q|$.

In stereo computations using the graph cuts approach, it is assumed that disparities can only have a finite set of values $[0, \ldots, d_{max}]$. Minimizing the energy in Eq. (1) is still NP-hard, but in [8] it was shown that using a set of "α-expansion" moves, each finding a minimal cut in a binary graph, can give good results that are very close to the global optimum.

Following [8] improvements to graph-cuts stereo were introduced in [20, 9]. These include better handling of occlusions, and symmetrical formulation of stereo. While we used the basic formulation of [8], the newer approaches can be incorporated into the proposed method if needed.

It should be noted that since we pass the depth values from frame to frame, we need a consistent description that is independent of the relative motion between frames. Therefore, after computing the disparities, we normalize them by the camera translation T_x between the two frames (In this section, we assume that the camera motion is given).

2.1 Flexible Pixel Dissimilarity Measure for Graph Cuts

Depth computation using iterative the graph cuts approach, as most other stereo computation methods, assumes one dimensional displacements between input images. This is a valid assumption when accurate image rectification is possible. Rectification needs accurate camera calibration, where both internal and external camera parameters are known (or can be accurately computed). However, when the input frames are part of an uncalibrated video sequence, the computed motions usually accumulate small errors in a few frames. In addition, the internal parameters of the camera are not always known. As a result, methods that assume accurate calibration and rectification fail for such video sequences. Moreover, the presence of small sub-pixel miss-registrations between frames not only reduces the accuracy of the computed depth, but usually results in a totally erroneous depth computation.

A possible way to overcome this problem is by computing a two dimensional optical flow rather than a one-dimensional optical flow. This approach increases the size of the graph and the computational complexity. We therefore keep the original structure of the graph using only horizontal displacements, but change the dissimilarity measure to be more tolerant for small vertical displacements.

2.1.1 Allowing 2D Sub-Pixel Displacements

The first step towards a flexible graph cuts approach is to allow horizontal and vertical sub-pixel displacements. To do so we extend an idea suggested in [6], where a pixel dissimilarity takes into account image sampling. Let (x, y) be a coordinates of a pixel in the image I_1 and let f_p be some candidate disparity. Instead of using the pixel dissimilarity $C_p(f_p) = |I_1(x, y) - I_2(x + f_p, y)|$, they suggested the following pixel dissimilarity:

$$C_p(f_p) = \min_{-\frac{1}{2} \leq s \leq \frac{1}{2}} |I_1(x, y) - I_2(x + f_p + s, y)|. \tag{2}$$

Eq. (2) is more accurate due to the fact that only a discrete set of disparities is possible. When the sampling of the image values at sub-pixel locations are computed using a linear interpolation, the above dissimilarity measure can be computed efficiently by:

$$C_p(f_p) = \max\{0, I_1(x, y) - v_{max}, v_{min} - I_1(x, y)\}, \tag{3}$$

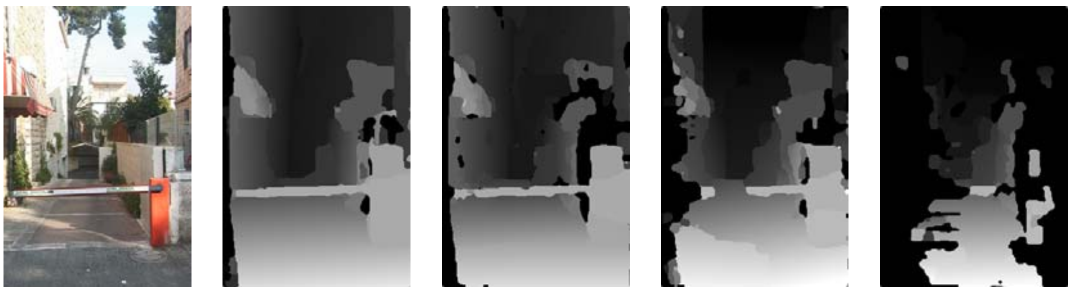

(a) (b) (c) (d) (e)

Figure 1: The advantage of using flexible graph cuts. One of two inputs frames is shown in (a). The disparities computed by the flexible and the regular graph cuts approaches on the original frames are shown in (b) and (c) respectively. The disparities computed by the flexible and the regular graph cuts approaches after rotating the right frame by one degree are shown in (d) and (e) respectively (unknown depths are marked in black).
It can be seen that the flexible and regular graph cuts approaches have similar results when using two calibrated frames, but when one of the images is slightly rotated, the flexible graph cuts approach successfully recovers the disparities for most of the pixels, while regular graph cuts approach fails.

where v_{max} and v_{min} are respectively the maximum and minimum of the two pixel values $\{I_2(x + f_p \pm \frac{1}{2}, y)\}$.

To allow sub-pixel vertical displacements, we will further change the range of the target pixel $(x + f_p, y)$ from a 1D interval to a 2D region:

$$C_p(f_p) = \min_{-\frac{1}{2} \le s,r \le \frac{1}{2}} |I_1(x,y) - I_2(x + f_p + s, y + r)|, \tag{4}$$

which can also be efficiently computed, in a similar way to the one dimensional case (Eq. (3)).

2.1.2 Handling Larger Vertical Displacements

The next step is to handle larger vertical displacements. This is most common when having small rotation about the optical axis, in which case the pixels at the left and right boundaries have large vertical displacements. Allowing large pixel displacements without any penalty will reduce the accuracy of the 1D search that overcomes the aperture problem. Therefore, unlike the sub-pixel displacements, larger displacements have some penalty. Considering all possible vertical displacements of up to a single pixel, gives our final pixel dissimilarity measure:

$$C_p(f_p, l) = \min_{-\frac{1}{2} \le s, r \le \frac{1}{2}} |I_1(x, y) - I_2(x + f_p + s, y + r + l)| + |l| \cdot K, \tag{5}$$
$$C_p(f_p) = \min_{l \in \{-1,0,1\}} C_p(f_p, l),$$

where l is a vertical displacement and K is a penalty factor (we used $K = 5$). Note that for a very large K, Eq. (5) reduces to the sub-pixel case in Eq. (4).

The advantage of using the flexible graph cuts approach is demonstrated in Fig.1.

2.2 A Planar Representation of the Scene using Graph Cuts

In the proposed framework depth is computed by the graph cuts approach only for a partial set of frames, and is propagated to the rest of the frames by depth warping. In order to propagate depth, it should be accurate and piecewise continuous. The widely used graph cuts methods give piecewise constant depth values. As a result, they tend to over-segment the image and do not obtain sub-pixel accuracy.

Instead, we compute a piecewise planar structure, as also suggested by [15, 9, 34, 7]. The main differences between [7] and our method is in the initialization of the planes and in the extension of the two-frames algorithm to long un-stabilized sequences. A detailed discussion of important differences can be found in Section 8.

There are several advantages in using a planar representation of the depth rather than discrete disparity values: (1) The piecewise planar model gives a better representation of the scene especially in urban areas. (2) The planar disparity surfaces can be estimated with sub-pixel accuracy, and therefore can be used to predict the depth even at far away frames without losing its accuracy. (3) Description of the depth map with planar surfaces requires a smaller number of segments compared to constant depths. Having a smaller number of more accurate segments significantly reduces the number of pixels marked as occlusions due to quantization errors.

The depth of a planar scene surface can be denoted as $a'X + b'Y + c'Z + d' = 0$ in the coordinate system of frame I_1. Assuming a perspective projection ($x = fX/Z$ and $y = fY/Z$, where f is the focal length) and multiplying the surface equation by $\frac{f}{d'Z}$ yields:

$$\frac{a'}{d'}x_1 + \frac{b'}{d'}y_1 + \frac{fc'}{d'} + \frac{f}{Z} = 0. \tag{6}$$

Dividing by d' is valid as $d' = 0$ only for planes that pass through the focal point, and these are planes that the camera does not see. Assuming a horizontal camera translation T_x between frames I_1 and I_2, the disparity between the corresponding pixels x_1 and x_2 is $x_1 - x_2 = f\frac{T_x}{Z}$ so the normalized disparity $\frac{x_1-x_2}{T_x}$ equals the inverse depth $\frac{f}{Z}$. From Eq. 6 it can be seen that the normalized disparity (or inverse depth) of a planar surface in the scene can be expressed as an affine function in the image coordinates:

$$\frac{x_1 - x_2}{T_x} = \frac{f}{Z} = -\frac{a'}{d'}x_1 - \frac{b'}{d'}y_1 - \frac{fc'}{d'}. \tag{7}$$

This formulation suggests that planar surfaces in the world induce affine disparities between the images (and vice versa). We will refer to planes in the world and "planar" disparities in the image in the same manner.

The process of computing planar disparities using graph-cuts can be described schematically as follows:

1. Run regular graph-cuts with constant disparities in the range of $[0, \ldots, d_{max}]$.
2. Find the parameters of a new plane and add them to the planes list.
3. Run the graph cuts method with planar labels from the list (described in Section 2.2.2).
4. Optionally, remove unused planes and return to Step 2.

This general scheme is described in a more detail in the following paragraphs.

2.2.1 Finding Candidate Planes (Steps 1-2)

Our purpose is to determine planes that will be used as labels in the planes-based graph-cuts (Step 3 and Section 2.2.2). A naive way to do so would be to select representatives of all planes, as is done in the case of constant disparities. However, this is not realistic as the space of all planar surfaces is too big, and sub-pixel accuracy is required. Therefore, the list of planes should be determined in a more efficient way.

Many times, an initial list of planes is already available. This is the case, for example, when the list of planes can be transferred from another image in the sequence where such a list has already been computed (See Section 2.3). In other cases, where no initial list of planes is available, we apply the following scheme:

Step 1: Run regular graph-cuts with constant disparities. These disparities can be viewed as disparities of planar surfaces:

$$m_i = 0 \cdot x + 0 \cdot y + i,$$

$$0 \leq i \leq d_{max}$$

Computing stereo with constant disparities can be done with the regular graph-cuts [8]. Note, however, that we always use a flexible pixel dissimilarity measure as described in Section2.1.

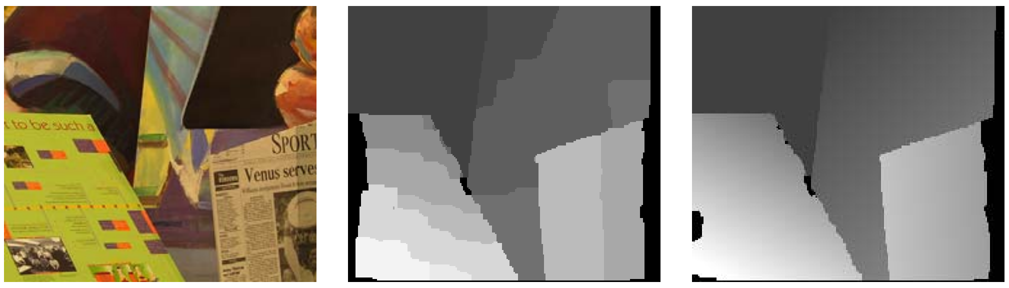
Step 2: The output of the graphcuts process is used to segment the image into connected components of equal disparity. Our assumption is that each planar surface is represented by multiple segments with constant disparities as demonstrated in Fig.2. Based on this assumption, we calculate the affine motion parameters for each large enough segment. Let S be such a segment, then the computed affine motion parameters $m_s = (a, b, c)$ are those that minimize the error:

$$E(a,b,c) = \sum_{(x,y) \in S} \left[I_1(x,y) - I_2(x + ax + by + c, y) \right]^2. \tag{8}$$

These motion parameters are computed directly from the image intensities using a gradient descent algorithm in a multiresolution framework as suggested by [5]. With the method of [5], the affine parameters can be computed in a sub-pixel accuracy. If a plane consists of several disparity segments, it is sufficient that only one of the corresponding parametric motion computations will converge, while the computations that do not converge are ignored. Having multiple descriptions for the same plane is allowed. The task of segmenting the image into planar disparities according to the planes list is left to the plane based graph-cuts described next.

2.2.2 Graph Cuts with Planar Labels (Step 3)

In this section we assume that a list of candidate planes is given, and we would like to represent the disparity map between the two images with these candidate planes. An iterative graph cuts approach is performed, where each pixel $p = (x, y)$ can be assigned with a single corresponding



(a) Input image (b) Constant Disparities (c) Planar Disparities

Figure 2: A regular graph cuts approach with constant disparities is used to obtain an initial over-segmentation of the image disparities. It is assumed that each planar surface is represented by multiple segments with constant disparities. This is demonstrated in (b). For each segment in (b) an parametric motion model is computed directly from the image intensities (using the Lucas-Kanade method) and all planes where this computation converge are used as planar labels in the planes-based graph-cuts. The result of the planes-based graph-cuts is shown in (c). This result is distinguishable from the ground truth in only a few pixels.

plane denoted as $m_p = (a, b, c)$. Similar to the classical graph cuts approach we use the Potts model as in Eq. (1),

$$E(f) = \sum_{p \in L} C_p(f_p(m_p)) + \sum_{p,q \in N} V_{p,q}\delta(m_p \neq m_q) \tag{9}$$

where $f_p(m_p)$ is the disparity of the pixel p according to the planar disparity $f_p(m_p) = ax + by + c$. As a data penalty function C_p we use the flexible pixel dissimilarity measure introduced in Section 2.1 (Eq. (5)). Note that the smoothness term penalizes transitions between different planes, and not transitions between different disparities. This implies that a single planar label will be the preferred representation of a single planar surface, as using multiple planes having similar parameters will be penalized. As a result, the planar representation of the scene will tend to be sparse even if the list of planes is redundant.

In addition to the labels representing planar surfaces, a special label, denoted as 'unknown' is used to represent pixels with unknown disparities. This label is assigned with a constant penalty. Although stronger formulations exist for the specific case of occlusions (such as [20]), we use this general purpose label to handle both occlusions, moving objects and deviations from the motion model.

The energy function defined above is very similar to the constant-disparities case, and can be minimized in the same manner. The result of the energy minimization will be an assignment of a planar surface to each pixel in the image. As noted before, the process of finding new candidate planes (while removing unused planes) can be repeated, this time with the better segmentation obtained by the plane-based graphcuts.

2.3 Forward Warping of Planes

A basic building block in the proposed scheme is the mapping of the inverse depths to the next frames. Let the inverse depth map for frame I_1 be described by planar surfaces, and let each pixel in I_1 be labeled as belonging to one of these planes.

The inverse depth of I_2 can be estimated from that of I_1 and from the horizontal translation T_x between the two frames in the following way:

1. The parameters of each planar surface are translated from the coordinate system of the source frame I_1 to the coordinate system of the target frame I_2. This is done as follows: Let

$$D_1(x,y) = \frac{f}{Z} = ax + by + c$$

describe a planar (normalized) disparity in I_1. Using Eq. 7, one can go back to the coefficients of the plane in the 3D space (up to a scale factor) giving: $aX + bY + c\frac{Z}{f} - 1 = 0$. Applying a horizontal translation T_x to get the representation of the plane in the coordinate system of I_2 yields: $a(X - T_x) + bY + c\frac{Z}{f} - 1 = 0$, or

$$aX + bY + c\frac{Z}{f} - (aT_x + 1) = 0.$$

Using Eq. 7 gives the normalized disparity in frame I_2:

$$D_2(x,y) = \frac{a}{a \cdot T_x + 1}x + \frac{b}{a \cdot T_x + 1}y + \frac{c}{a \cdot T_x + 1}. \quad (10)$$

The parameters of a planar disparity in I_2 can therefore be computed simply from the corresponding plane parameters in I_1 and the relative horizontal translation T_x.

2. The pixel labeling of I_2 can be computed by warping forward the pixel labeling of I_1. When two labels are mapped to the same pixel in I_2, the label corresponding to a smaller depth is

used to account for occlusion. Pixels in I_2 that were not assigned with a label by the forward warping are marked as "unknowns", and are not used in further computations.

The forward warping of inverse depth may leave some pixels in I_2 with no assigned label. This is not an immediate problem to motion computation, as the depth of all pixels is not required for motion analysis. At a later stage, the labels are completed from neighboring frames or interpolated from other pixels.

2.4 Implementation and Efficiency

In general, stereo computation of each frame includes: (a) A single call to graph-cuts, using planes warped from the previous computation plus constant disparities. (b) A single call to graph-cuts after removing unused planes and adding new planes using the parametric affine motion computation. Each call to graph-cuts involves two cycles of graph-cuts, each cycle consists of a single expansion move for each label. We found that similar results were produced with only a single cycle in the first call and two cycles in the second call. The second cycle is important mainly to obtain a clean segmentation, which is not crucial in the first call.

For scenes having high depth variability, e.g. Street Sequence, fifty constant disparities were used in the initialization step. For scenes with lower depth variability, e.g. Boat Sequence, twenty constant disparities were enough.

Stereo computations using graph-cuts require most of the computational costs in the proposed framework, and take $\sim$20 seconds per frame of size 360 by 240. In our experiments we used the graph-cuts implementation of Kolmogorov (http://www.cs.cornell.edu/$\sim$rdz/graphcuts.html), and added to it the flexible dissimilarity measure and the planar labels. These additions did not have a significant effect on the performance, since the number of candidate planes was practically smaller than the number of constant disparity labels.

The stereo computations may be accelerated significantly using a multi-resolution framework. In addition, our formulation of planar and flexible graph-cuts can be incorporated with other methods for solving Markov Random Fields. For example, in [11], a fast multi-resolution implementation of Belief-Propagation, which is an alternative approach for solving MRFs, was shown to produce good results much more efficiently. [4] even introduced a real-time implementation of stereo matching using graphics hardware.

3 Computing Ego Motion (Assuming Known Depth)

Assume that the image I_{k-1} has already been aligned and de-rotated according to the motion parameters that were computed in previous steps. Let I_k be the new frame to be aligned. We are also given the inverse depth map D_{k-1} corresponding to (the de-rotated) I_{k-1}. Our motion model includes a horizontal camera translation T_x and camera rotations R about the x and z axis:

$$P + \vec{T_x} = R^{-1}P', \tag{11}$$

where P and P' are corresponding 3D points in the coordinate systems of I_{k-1} and I_k respectively, and $\vec{T_x} = [T_x, 0, 0]^t$ denotes the horizontal translation. Note that the rotation matrix R is applied only on frame I_k, as we assume that I_{k-1} has already been aligned and de-rotated. On the other hand, the translation is applied on I_{k-1}, since the depths are known only for frame I_{k-1}. A schematic diagram of the ego-motion computation is shown in Fig. 3(a).

Assuming small rotations, the image displacements can be modeled as:

$$\begin{aligned} x' &= x + T_x \cdot D_{k-1}(x,y) + \cos(\alpha)x' - \sin(\alpha)y' \\ y' &= y + b + \sin(\alpha)x' + \cos(\alpha)y' \end{aligned} \tag{12}$$

The camera rotation about the z axis is denoted by α, and the tilt is denoted by a uniform vertical translation b. in cases of a larger tilt, or when the focal length is small, the fully accurate rectification can be used.

To extract the motion parameters, we use a slight modification of the Lucas-Kanade direct 2D alignment [5], iteratively finding motion parameters which minimize the sum of square differences using a first order Taylor approximation. The approximations $cos(\alpha) \approx 1$ and $sin(\alpha) \approx \alpha$ are also used, giving the following error function to be minimized:

$$E(T_x, b, \alpha) = \sum_{x,y} \{I_{k-1}(x - T_x \cdot D_{k-1}(x,y), y) - I_k(x' - \alpha y', y' + b + \alpha x')\}^2. \tag{13}$$

We use the first order Taylor expansion around $I_{k-1}(x,y)$ and around $I_k(x',y')$ to approximate:

$$\begin{aligned} I_{k-1}(x - T_x \cdot D_{k-1}(x,y), y) &\approx I_{k-1}(x,y) - \frac{\partial I_{k-1}}{\partial x} T_x D_{k-1}(x,y) \\ I_k(x' - \alpha y', y' + b + \alpha x') &\approx I_k(x',y') + \frac{\partial I_k}{\partial x'}(-\alpha y') + \frac{\partial I_k}{\partial y'}(b + \alpha x'), \end{aligned} \tag{14}$$

which results in the following minimization:

$$E(T_x, b, \alpha) = \sum_{x,y} \{I_{k-1}(x, y) - I_k(x', y') - \frac{\partial I_{k-1}}{\partial x} T_x D_{k-1}(x, y) - \frac{\partial I_k}{\partial x'}(-\alpha y') - \frac{\partial I_k}{\partial y'}(b + \alpha x')\}^2 \quad (15)$$

The minimization can be solved efficiently by taking the derivatives of the error function E with respect to each of the three motion parameters and setting them to zero, giving the following linear set of equations with only three unknowns:

$$A^T A \begin{bmatrix} T_x \\ \alpha \\ b \end{bmatrix} = cA^T, \quad (16)$$

where

$$A = [\sum_{x,y} \frac{\partial I_{k-1}}{\partial x} D_{k-1}(x, y), \sum_{x,y} (\frac{\partial I_k}{\partial y'} x' - \frac{\partial I_k}{\partial x'} y'), \sum_{x,y} \frac{\partial I_k}{\partial y'}],$$

and

$$c = \sum_{x,y} (I_{k-1}(x, y) - I_k(x', y')).$$

Similar to [5], we handle large motions by using an iterative process and a multi-resolution framework. In our case, however, we simultaneously warp both images, one towards the other: we warp I_{k-1} towards I_k according to the computed camera translation T_x (and the given inverse depth), and we warp I_k towards I_{k-1} according to the computed rotation α and the uniform vertical shift b.

For additional robustness, we added outlier removal and temporal integration:

- Pixels having a large intensity difference are marked as outliers and are omitted from the motion computation. Specifically, we omit pixels with

$$\frac{|\sum_W (I_{k-1}(x, y) - I_k(x', y')) \cdot \frac{\partial I_{k-1}}{\partial x}|}{\sum_W \frac{\partial I_{k-1}}{\partial x}^2} > s, \quad (17)$$

  where W is a 5×5 neighborhood, and s is a threshold (we used $s = 1$). Other schemes such as re-weighted least squares can also be used [24]. Obviously, pixels that were marked by the depth computation as having an unknown disparity are also not used.

- Frames that were already aligned are averaged with earlier frames, also known as "Temporal Integration". Instead of computing motion using a single reference frame, we use the temporal integration that was shown in [18] to add stability and robustness to outliers in traditional 2D alignment methods.

Since the computed depth is quantized, a consistent depth pattern (e.g. the common case when near objects are on the bottom of the image and far objects are on the top of the image) can cause a small rotational bias, which accumulates in long sequences. A small modification of the motion computation method described earlier can overcome this problem: Since the depth parallax is horizontal, only vertical displacements are used to compute image rotation. To do so, we change the error function in Eq. (13) to:

$$E(T_x, b, \alpha) = \sum_{x,y} \{I_{k-1}(x - T_x \cdot D_{k-1}(x, y), y) - I_k(x', y' + b + \alpha x')\}^2. \tag{18}$$

As in the original method, the iterative image warping is done using the accurate rotation matrix. It should be noted that more general motion models can be computed, however our experience showed that adding motion parameters that are not independent (e.g. pan with horizontal translation) may reduce the robustness of the motion computation for scenes with small depth variations. Our current approach can even handle scenes that are entirely flat.

4 Interleaving Computation of Depth and Motion

The recovery of depth and motion is initialized by computing the inverse depth using the first two frames. It is continued by interleaving stereo and motion computations, until the motion of the camera and the corresponding inverse depths are computed for the entire sequence. The initialization is described in Section 4.1 (and Fig. 5), and the interleaving process is described in Section 4.2. A schematic work-flow of the interleaving process is shown in Fig.6.

As the process seems to depend on the selection of the initial frames, we tested this dependency by randomly selecting 30 different starting points in a sequence of two thousand frames. In this experiment all different starting points converged to very similar mosaic results.

4.1 Initialization: First Two Frames

A 2D image translation (u, v) is initially computed using the traditional Lucas-Kanade method [5], between I_1 (the reference frame) and I_k, for $k = 2, 3, \ldots$. This is performed until a frame I_k having sufficient horizontal displacement from I_1 is reached. Given (u, v), I_k is warped vertically towards I_1 according to v, and it is assumed that $T_x = u$. The vertical motion v is estimated accurately since the parallax is mostly horizontal.

The graph cuts algorithm is applied on I_1 and the warped I_k to estimate the depth map of I_1 as described in Section 2. Despite the rotational component which has not yet been compensated, a correct depth map for most of the pixels can be estimated by using the "Flexible Dissimilarity Measure" (Section 2.1). The "unknown" label is automatically assigned to pixels with large vertical displacements induced by the rotation. The pixels that get valid depth values can now be used to obtain a better estimate for the relative camera motion between I_1 and I_k (using the ego motion computation described in Section 3).

After warping I_k towards I_1 according to the estimated rotation matrix, the depth map of I_1 is re-computed. This iterative process is continued until convergence.

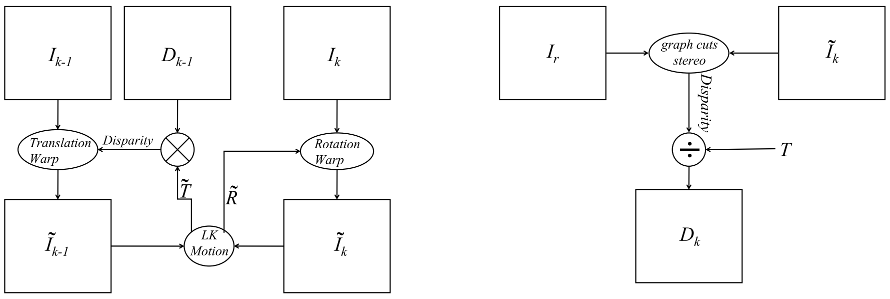
An example for the process is shown in Fig.4. In this example, two frames from a video sequence were selected, and one of the frames was manually rotated by 2°. The intermediate depth maps that were obtained during the iterative process are shown, demonstrating the effective convergence of the proposed method.

4.2 Interleaving Computation for the Entire Sequence

During the initialization process the inverse depth map D_1 was computed, corresponding to frame I_1. The reference frame I_r is set to be I_1, and its inverse depth map D_r is set to be D_1.

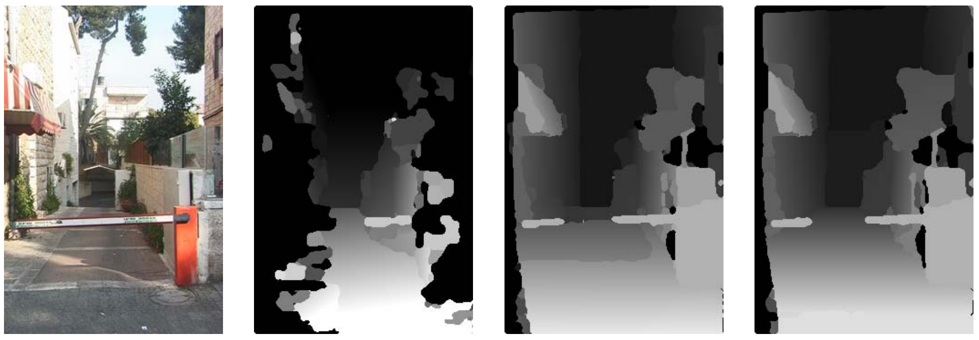
With the inverse depth D_r the relative camera motion between the reference frame I_r and its neighboring frame I_k can be computed as described in Section 3. Let (T_k, R_k) be the computed camera pose for frame I_k compared to the reference frame I_r. Given (T_k, R_k) and the inverse depth map D_r for image I_r, the inverse depth values of D_r can be mapped to the coordinate system of I_k as described in Section 2.3, giving D_k. A schematic diagram of the work-flow is shown in Fig.3.

Camera motion is computed as described above between I_r and its neighboring frames I_{r+1}, I_{r+2},



(a) Motion Computation (b) Depth Computation

Figure 3: Schematic diagrams of the motion and depth computations:
(a) Motion computation between frame I_{k-1} with computed inverse depth D_{k-1}, and a new image I_k. (i) Initial translation $\widetilde{T}$ and $\widetilde{R}$ are estimated (e.g. same as last frame, or zero). (ii) I_{k-1} is warped with the estimated disparity, equivalent to $\widetilde{T}$ multiplied by D_{k-1}, to give the warped $\widetilde{I_{k-1}}$. I_k is rotated by $\widetilde{R}^{-1}$ to give $\widetilde{I}_k$. (iii) New estimations for rotation $\widetilde{R}$ and translation $\widetilde{T}$ are computed between the warped images $\widetilde{I_{k-1}}$ and $\widetilde{I}_k$. (iv) The process is repeated from step (ii) until convergence.
(b) Computation of inverse depth for a new reference frame I_k is performed between the previous reference frame I_r, and $\widetilde{I}_k$. $\widetilde{I}_k$ is the new frame I_k after it has been rotated to the coordinate of I_r by the rotation estimated in part (a). The inverse depth D_k is the disparity computed between I_r and $\widetilde{I}_k$ divided by the translation T previously computed between these two frames.



original frame 1^{st} iteration 2^{nd} iteration 3^{rd} iteration

Figure 4: Intermediate depth maps computed during the iterative depth and motion computations process (unknowns are marked in black). To make this example more interesting, we rotated the right frame by two degrees before applying the algorithm. It can be seen that the columns at the left and right margins are marked as unknowns in the first iteration due to their large vertical displacements. The support of the center pixels was sufficient to correctly extract the relative motion between the two frames.

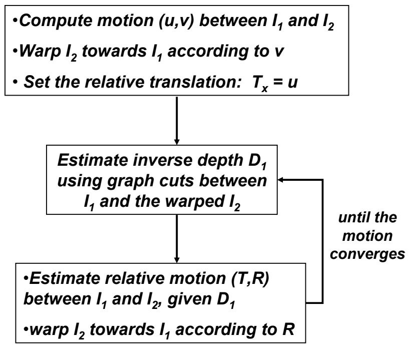

•Compute motion (u,v) between I_1 and I_2
•Warp I_2 towards I_1 according to v
• Set the relative translation: $T_x = u$
Estimate inverse depth D_1 using graph cuts between I_1 and the warped I_2
until the motion converges
•Estimate relative motion (T,R) between I_1 and I_2, given D_1
•warp I_2 towards I_1 according to R

Figure 5: A schematic work-flow of the initialization stage.

etc., until the maximal disparity between the reference frame I_r and the last frame being processed, I_k, reaches a certain threshold. At this point the inverse depth map D_k has been computed by forward warping of the inverse depth map D_r. D_k is updated using the iterative graph cuts approach between I_k and I_r to get better accuracy. To encourage consistency between frames, small penalties are added to all pixels that are not assigned with labels of their predicted planes. This last step can be done by slightly changing the data term in Eq. (1).

Recall that for keeping the consistency between the different frames, the disparities computed by the graph cuts method should be normalized by the horizontal translation computed between the frames, giving absolute depth values (up to a global scale factor).

After updating D_k, the new reference frame I_r is set to be I_k, and its inverse depth map D_r is set to be D_k. The relative pose and the inverse depth of the frames following I_r are computed in the same manner, replacing the reference frame whenever the maximal disparity exceeds a given threshold.

The process continues until the last frame of the sequence is reached. In a similar manner, the initial frame can be set to be one of the middle frames, in which the interleaving process continues in the both positive and negative time directions.

This scheme requires a computation of the inverse depths using the graph cuts method only for a subset of the frames in the sequence. Besides the benefit of reducing the processing time, disparities are more accurate between frames having larger separation. While depth is computed only on

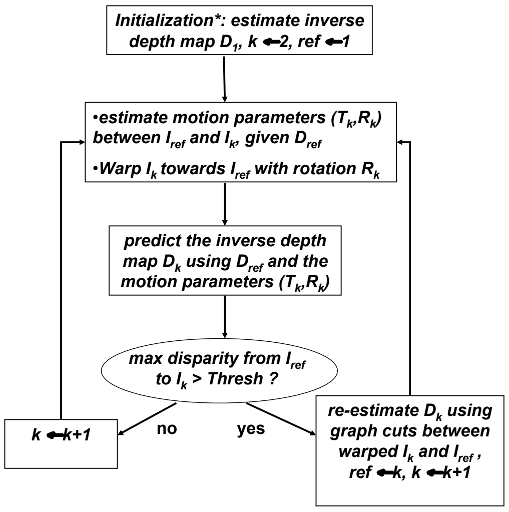

Initialization*: estimate inverse depth map D_1, $k \leftarrow 2$, ref $\leftarrow 1$
•estimate motion parameters (T_k,R_k) between I_{ref} and I_k, given D_{ref}
•Warp I_k towards I_{ref} with rotation R_k
predict the inverse depth map D_k using D_{ref} and the motion parameters (T_k,R_k)
max disparity from I_{ref} to I_k > Thresh ?
no
yes
$k \leftarrow k+1$
re-estimate D_k using graph cuts between warped I_k and I_{ref}, ref $\leftarrow k$, $k \leftarrow k+1$

Figure 6: A schematic work-flow of the interleaving process.

a subset of frames, all the original frames are used for stitching seamless panoramic images.

4.3 Panoramic Rectification

In real scenarios, the motion of the camera may not perfectly satisfy our assumptions. Some small calibration problems, such as lens distortion, can be treated as small deviations from the motion model and can be overcome using the robust tools (such as the "unknown" label or the "flexible" graph cuts approach presented in Section 2.1).

However, a bigger challenge is introduced by the camera's initial orientation: if the camera is not horizontally leveled in the first frame, the motion computations may consist of a false global camera rotation. The deviation of the computed motion from the actual motion may hardly be noticed for a small number of frames (making traditional calibration much harder). But since the effect of a global camera rotation is consistent for the entire sequence, the error accumulates, causing visual artifacts in the resulting panoramas.

This problem can be avoided using a better setup or by pre-calibrating the camera. But very accurate calibration is not simple, and in any case our work uses videos taken by uncalibrated cameras, as shown in all the examples.

A possible rotation of the first image can be addressed based on the analysis of the accumulated motion. A small initial rotation is equivalent to a small vertical translation component. The effect for a long sequence will be a large vertical displacement. The rectification of the images will be based on this effect: After computing image translations (u_i, v_i) between all the consecutive frames in the sequence, the camera rotation α of the first frame can be estimated as $\alpha = \arctan(\sum_i v_i / \sum_i u_i)$. A median can be used instead of summation in the computation of α if a better robustness is needed. All the frames are de-rotated around the z axis according to α.

5 Minimal Aspect Distortion (MAD) Panorama

The visualization process can start once the camera ego-motion and the dense depth of the scene have been computed. Motion and depth computation can be done as proposed in the previous sections, or can be given by other processes. We propose two approaches for visualizing long scenes. In Section 6 the existing X-Slits approach is used for rendering selected viewpoints. In this section a new method is presented for generation of a long minimal aspect distortion (MAD) panorama of the scene. This panorama should satisfy the following properties: (i) The aspect distortions should be minimal. (ii) The resulting mosaic should be seamless.

Since the visualization stage comes after motion computation, the images in this section are assumed to be de-rotated and vertically aligned. The residual displacements between the images are therefore purely horizontal. It is also assumed that the depth maps are dense. The depth values for pixels that were labeled as unknowns are interpolated from other frames (see Section 2.3). When pixels are occluded in all neighboring frames, they are interpolated from neighboring pixels according to the planar surface models.

5.1 Panorama as a Cut in the Space-Time Volume

A panoramic image is determined by a cut $C(t)$ through the space-time volume, as seen in Fig. 7. For each image t, $C(t)$ determines the left column of a strip S_t in image t to be stitched into the panorama.

To obtain a seamless stitching, the right border of the strip S_t is a curved line, corresponding to the left side of the next strip, i.e. $C(t+1)$ (as shown in Fig. 8). This curved line is computed using the camera motion and the known inverse depth. The image strip S_t is warped to a rectangular strip S'_t before being pasted into the panorama. The warping is done by scaling each row independently,

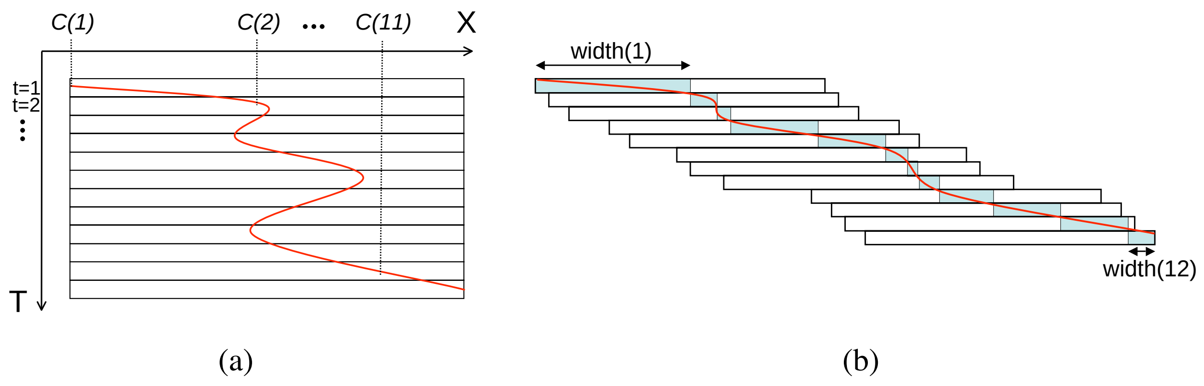


Figure 7: (a) A general cut $C(t)$ through the space-time volume. (b) The same cut in the spatially aligned space-time volume. $C(t)$ designates the leftmost column of the strip S_t taken from frame t.

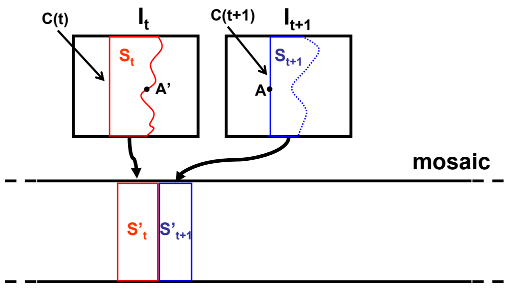


Figure 8: The panorama is stitched from a collection of rectangular strips S'_t warped from the input frames S_t. The stitching is seamless because each point A' on the right border of the image strip S_t corresponds to the point A on the left border S_{t+1} according to its the computed disparity. A and A' are the same point in the mosaic's coordinates.

from its width in S_t to the width of S'_t given by:

$$width(S'_t) = C(t+1) - C(t) + alignment_{t+1}(C(t+1)), \tag{19}$$

where $alignment_{t+1}(x)$ is set to be the average disparity of the pixels in column x of image $t+1$ relative to image t. This average disparity equals to the average inverse depth at this column multiplied by the relative horizontal translation T_x. In the warping from S_t to S'_t far away objects are widened and closer objects are becoming narrower.

In pushbroom and X-Slits projections (See Fig. 13), $C(t)$ is a linear function of the camera translation. The local derivative of a general cut $C(t)$ can represent a local X-Slits slice having a slope of:

$$slope(t) = dC(t)/dt, \tag{20}$$

as demonstrated in Fig. 9. The local X-Slits slope can change spatially throughout the panorama. Special slopes are the pushbroom projection $(slope(t) = 0)$ and the perspective projection $(slope(t) \rightarrow \infty)$.

In [31] a minimum distortion mosaic is created such that the $slope$ function is piecewise constant, and a non-linear optimization is used to minimize this distortion. In MAD mosaicing the cut $C(t)$ is allowed to be a general function, and simple dynamic programming is used to find the global optimum.

5.2 Defining the Cost of a Cut

The optimal cut through the space-time volume that creates the MAD mosaic is the cut that minimizes a combination of both a distortion cost and a stitching cost. The cost of a cut $C(t)$ is defined as follows:

$$cost(C) = \sum_t distortion_t(C_t, C_{t+1}) + \alpha \sum_t stitching_t(C_t, C_{t+1}), \tag{21}$$

where t is a frame number and α is a weight. The distortion term estimates the aspect distortion in each strip, and the stitching term measures the stitching quality at the boundaries. Both are described next.

Distortion cost: As described before, a cut C determines a set of strips $\{S_t\}$. We define the distortion cost $distortion_t(C(t), C(t+1))$ to be the variance of disparities of the pixels in strip S_t. This is a good measurement for distortion, as strips with high variance of disparities have many dominant depths. Objects in different depths have different motion parallax, resulting in a distorted mosaic. In that case, we prefer such strips to be wider, giving a projection that is closer to perspective, as shown in Fig. 10. A single wide strip in a high variance region will be given a lower cost than multiple narrow strips.

We have also experimented with a few different distortion functions: spatial deviation of pixels relative to the original perspective; aspect-ratio distortions of objects in the strip ([31]), etc. While the results in all cases were very similar, depth variance in a strip (as used in our examples) was preferred as it consistently gave the best results, and due to its simplicity.

Stitching cost: The $stitching$ cost measures the smoothness of the transition between consecutive strips in the panoramic image, and encourages seamless mosaics. We selected a widely used stitching cost (similar to the one used in [3]), but unlike the common case, we also take the scene depth into consideration when computing the stitching cost. This $stitching$ cost is defined as the sum of square differences between the $C(t+1)$ column of image $t+1$ and the corresponding column predicted from image t. To compute this predicted column an extra column is added to S'_t, by generating a new strip of width $width(S'_t)+1$ using the method above. The right column of the new strip will be the predicted column.

In addition to the distortion and stitching costs, strips having regions that go spatially backwards are prohibited. This is done by assigning an infinite cost to strips for which $C(t+1) < C(t) + D_{min}(C(t+1))$, where $D_{min}(C(t+1))$ is the minimal disparity in the column $C(t+1)$ of image $t+1$. For efficiency reasons, we also limit $C(t+1) - C(t)$ to be smaller than $1/5$ of the image's width.

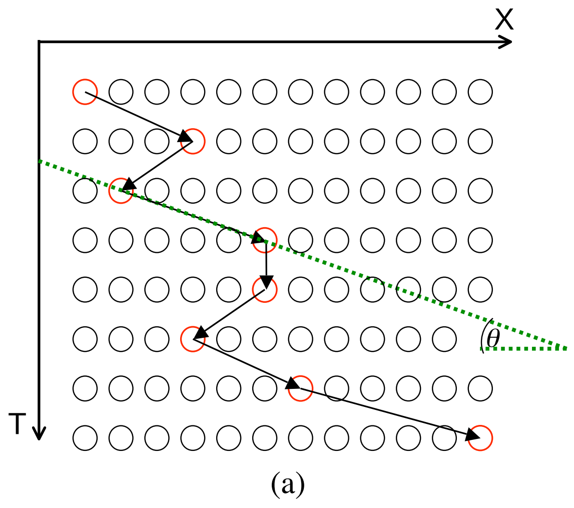
The distortion cost and the minimal and average disparities are computed only from image regions having large gradients. Image regions with small gradients (e.g. a blue sky) should not influence this cost as distortions are imperceptible at that regions, and their depth values are not reliable.

5.2.1 Graph Construction and Minimal Cut

The graph used for computing the optimal cut is constructed from nodes representing image columns. We set a directed edge from each column x_1 in frame t to each column x_2 in frame $t+1$ having the weight

$$V_t(x_1, x_2) = distortion_t(x_1, x_2) + \alpha \cdot stitching_t(x_1, x_2).$$

Each cut C corresponds to a path in the graph, passing through the column $C(t)$ at frame t. The sum of weights along this path is given by $\sum_t V_t(C(t), C(t+1))$, and is equal to the cost defined in Eq. 21.



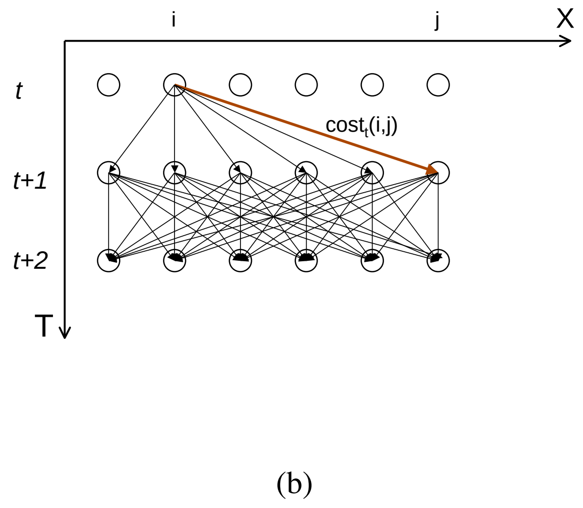


Figure 9: (a) A path C in the graph. $C(t)$ indicates the node selected in time t. Locally $C(t)$ is a X-Slits projection as shown by the dotted line, with $slope = cot(\theta)$. (b) Graph Construction: the nodes are all the columns in the $X - T$ space. There are edges from each column of image t to all the columns of image $t + 1$.

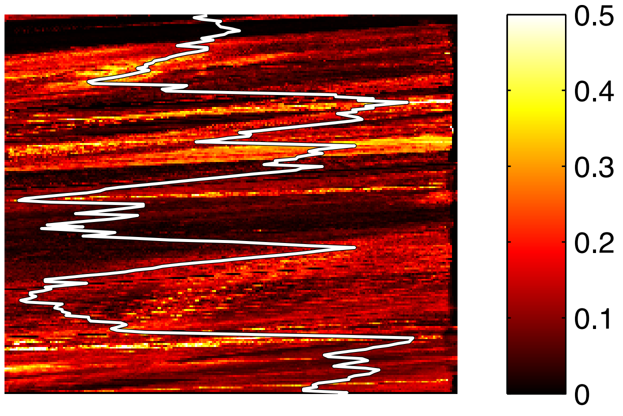


Figure 10: Disparity variance at columns in the non-aligned $x - t$ volume corresponding to Fig. 12. Each row represents an input image, and the value at location (x, t) represents the variance of normalized disparity at column x of image t. The $distortion$ cost is the variance of each strip, and here we show the variance of each column. It can be observed that the selected cut has a large slope when it passes through problematic areas that have high variance. A large slop is equivalent to wide strips, giving a result that is close to the original perspective.

Finding the optimal cut that minimizes the cost in Eq. 21 is therefore equivalent to finding the shortest-path from the first frame to the last frame in the constructed graph. Any shortest-path algorithm can be used for that purpose. We implemented the simple Bellman-Ford dynamic programming algorithm with online graph construction.

Fig. 11 compares MAD mosaics with pushbroom mosaics on two regions taken from a long panorama. The differences between the two mosaics are large because the camera is very close to the scene, and depth differences inside the scene are very large compared to the distance between the camera and the scene.

5.2.2 Implementation and Efficiency

The time complexity of the stitching process is dominated by the computation of the $stitching$ cost. By definition the $stitching$ cost has a complexity of $w^2 \cdot h$ operations per frame. The computation of the $distortion$ cost is relatively negligible as it has a complexity of w^2 operations per frame. We do not have to actually calculate the variance for each strip, because we can pre-calculate the first two moments of each column, and use them to estimate the variance of every strip. The computation time for the dynamic programming is negligible as well.

Online graph construction enables us to run on any number of frames with a low memory constraint: the frames are loaded and processed sequentially and the graph-solving advances one frame at a time. The memory used in our implementation is of size $w \cdot N$, where w is the image width, h is the image height, and N is the number of frames.

Our non-optimized Matlab implementation has a processing time of 2.4 seconds per frame with the stitching cost (not including loading the images). The processing time without the stitching cost is 0.2 seconds per frame. The processing was performed on 360×240 images with a 3Ghz CPU.

6 Dynamic Image Based Rendering with X-Slits

MAD mosaicing generates a single panoramic image with minimal distortions. A MAD panorama has multiple viewpoints, and it can not be used to create a set of views having a 3D effect. Such 3D effects can be generated by X-Slits mosaicing.

New perspective views can be rendered using image based rendering (IBR) methods when the

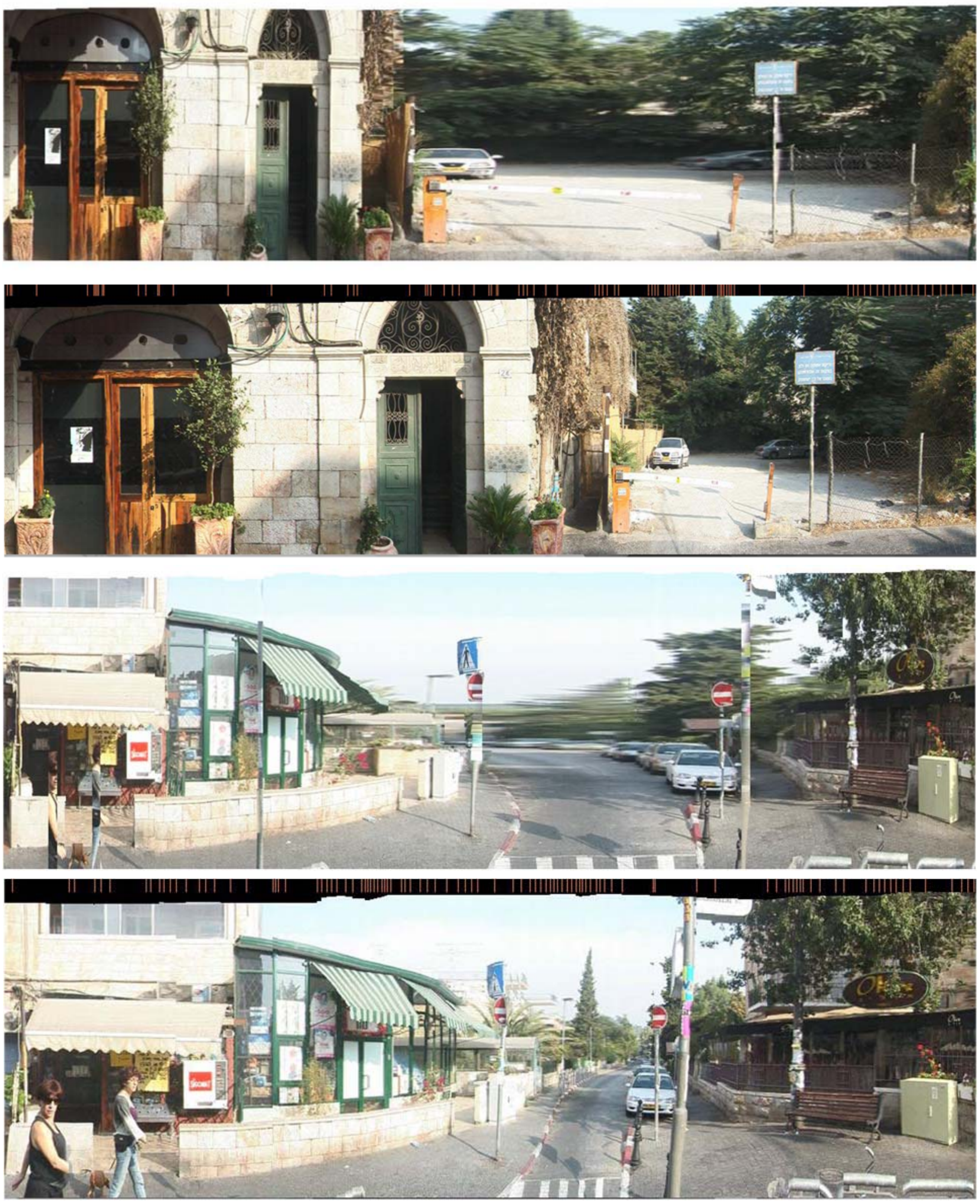
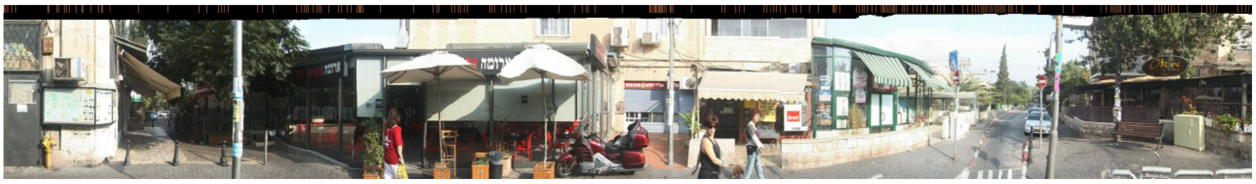
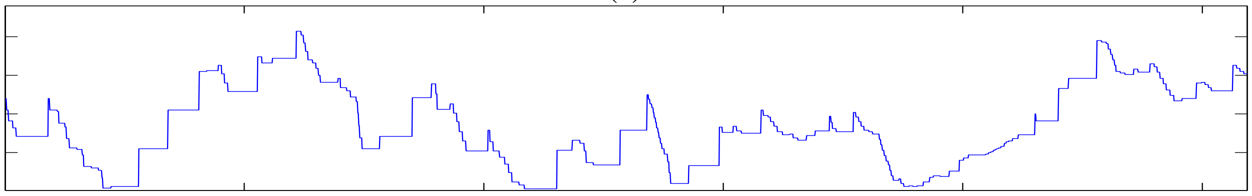

(a)
(b)
(c)
(d)

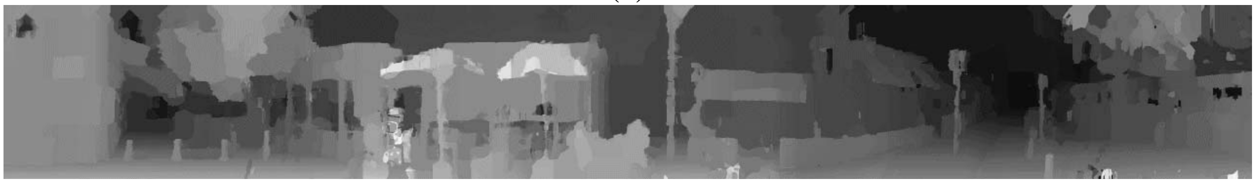
Figure 11: Pictures (a, c) are segments from a long pushbroom mosaic, while (b, d) show the same area in a long MAD mosaic. Substantial widening of far away objects and narrowing of close objects in evident in pushbroom. While width of strips is uniform in pushbroom, strip boundaries are marked on top of the MAD Mosaic image. Note also that the MAD mosaic avoids cutting inside moving objects.



(a)



(b)



(c)

Figure 12: (a) A MAD mosaic of a street in Jerusalem. The small marks on the top are the strip boundaries. It can be noticed that large vertical objects that cannot be sampled correctly (e.g. moving people or objects that are too close and have bad depth values) are preserved in a single wide strip due to the stitching cost. A top view of the space-time volume of this sequence is shown in Fig. 10. (b) A graph showing $C(t)$, the left strip boundary for each frame. The width of the graph is scaled to the mosaic's coordinates, and an identical value is shown for all columns corresponding to the same strip. Each location in the graph therefore corresponds to the strip in (a) above it. Lower values of $C(t)$ represent a strip from the left side of the input image (and a view to the left). Higher values of $C(t)$ represent a strip from the right side, and a view to the right. (c) The constructed depth map of the mosaic.

multiple input images are located densely on a plane [22, 12]. In our case the camera motion is only 1D and there is only horizontal parallax, so perspective images can not be reconstructed by IBR. An alternative representation that can simulate new 3D views in the case of a 1D camera translation is the X-Slits representation [37]. With this representation, the slicing function $C(t)$ is a linear functions of the horizontal translation (See Fig 13).

Constant slicing functions ($C(t) = const$), as in Fig. 13.b, correspond to pushbroom views from infinity. More general linear functions,

$$C(t) = a \cdot U_x + b$$

where U_x is the accumulated horizontal camera translation, correspond to finite viewing positions. The geometrical interpretation of these slices appears in [37]. It was shown in [10] that synthesized views obtained using the X-Slits projection are the closest to a perspective projection under the constraint of using linear slices.

As the linear slicing functions of the X-Slits are more constrained than those used in MAD mosaicing, they can not avoid distortions as in MAD mosaicing. On the other hand, X-Slits projections are more powerful to create a desired viewpoint. For example, in MAD-mosaicing distortions are reduced by scaling each strip according to the average disparity, while in the X-slits representation image strips are not scaled according to disparity. This is critical for preserving the geometrical interpretation of X-Slits, and for the consistency between different views, but it comes at the expense of increasing the aspect distortions.

6.1 Seamless Stitching for X-Slits

Since a video sequence is not dense in time, interpolation should be used to obtain continuous mosaic images. When the displacement of the camera between adjacent frames is small, reasonable results can be obtained using some blurring of the space-time volume [22, 12]. For larger displacements between frames, a depth-dependent interpolation must be used. The effects of using depth information for stitching is shown in Fig. 14. Fig. 14.a shows results without using the dense depth. While the restaurant (whose depth is close to the dominant depth in the scene) is stitched well, closer objects (rails) are truncated and faraway objects are duplicated. In contrast, stitching

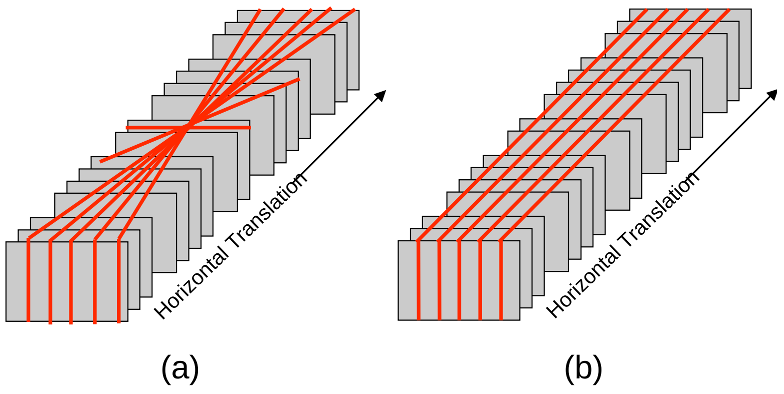


Figure 13: New views are generated as linear slices in the horizontal translation $C(t) = a \cdot U_x + b$, where U_x is the accumulated horizontal camera translation. (a) Changing the slope of the slice simulates a forward-backward camera motion, while shifting the slice simulates a sideways camera motion. (b) The special case of parallel slices in the space-time volume ($C(t) = const$) results in different viewing directions of oblique pushbroom images.

according to the dense depth map (Fig. 14.b) gives a far better stitching result. Note that very narrow objects are still a challenge to our method and may not be stitched well (Such as the narrow pole on the right side of the restaurant). This problem is usually avoided in the MAD mosaicing, which tends to keep such narrow objects in a single strip.

We used two different rendering approaches of X-Slit images. An accurate stitching using the dense depth maps, and a faster implementation suitable for interactive viewing. The accurate stitching is similar to the one described in section 5.1, with the only difference that the image strips are not scaled according to the average disparity. To get real-time performance suitable for interactive viewing each row cannot be scaled independently. Instead, the following steps can be used in real-time X-Slits rendering: (i) Use a pre-process to create a denser sequence by interpolating new frames between the original frames of the input sequence. This can be done given the camera motion between the frames, and their corresponding inverse depth maps. (ii) From the denser sequence, continuous views can be obtained by scaling uniformly each vertical strip pasted into the synthesized view, without scaling each row separately. The uniform scaling of each strip is inversely proportional to the average disparity in the strip.

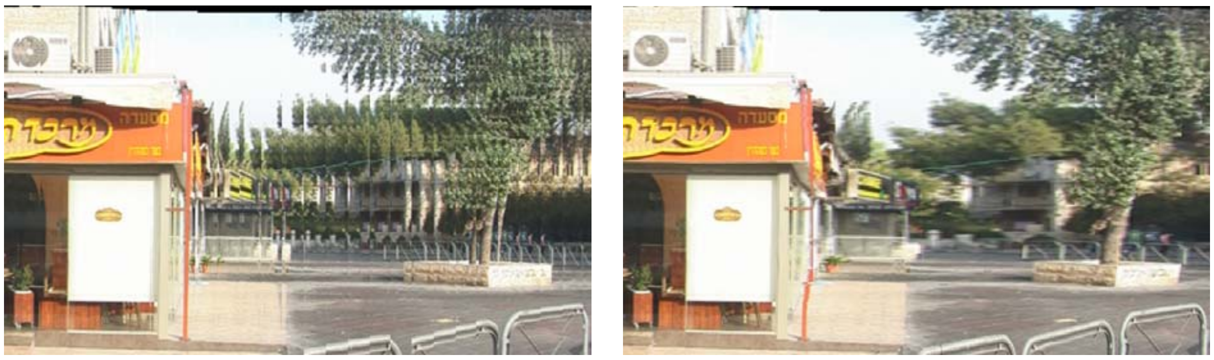

(a) (b)

Figure 14: The benefit of stitching using dense depth. (a) A street view (X-slits) obtained by stitching without depth scaling. The restaurant (whose depth is close to the dominant depth in the scene) is stitched well, but closer and farther objects are truncated or duplicated. (b) Depth-based stitching eliminates duplications and truncations. Note that very narrow objects may still pose a problem for our method (Such as the narrow pole on the right side of the restaurant).

7 Experimental Results

All experiments discussed in this paper were performed on videos without camera calibration other than the removal of lens distortion. The only manual involvement was the setting of the maximal allowed disparity, which was done only to speed up performance.

The data and the full panoramas are available online at *http://www.vision.huji.ac.il/mad*. The examples demonstrate the applicability of our method to a variety of scenarios, including a sequence captured from a river boat (Fig. 15-16), a helicopter (Fig. 17) and a driving car in a long street (Figs. 11,12,18). The long MAD panorama of the entire street can be seen online. We constructed panoramic images using X-Slits (Fig. 15,17) as well as sample images from a virtual walk-through (Fig. 16 and 18). In Figs. 11-12 MAD mosaicing was used to reduce distortions.

To successfully process all these different scenarios, the method had to handle different types of camera motions (e.g. highly unstable camera in the helicopter sequence), and different kinds of scenes (such as the street sequence, where the depths varies drastically between the front of the buildings and the gaps between them).

MAD mosaicing was proven to work particularly well on long and sparse sequences, as shown in Figs. 11-12. Some objects have such a large disparity that the stereo computation could not register them well, e.g. the close traffic signs that have a disparity greater than 50 pixels. In such cases the stitching cost is responsible for rendering these objects using a wide strip from a single frame. As shown in Fig. 12, moving people and some of the very close traffic signs were rendered correctly in most cases. The long MAD panorama of the entire street is available online.

The total computational time ranges from 10-30 seconds for each frame of size 360x240, depending on the motion between frames and the depth variation. The stereo computations, which are the most time consuming part, take approximately 20 seconds per frame. In a dense video sequence, stereo computations run only once in a few frames. The ego-motion computations and the stitching of MAD mosaics take approximately 5 and 2.5 seconds per frame respectively in an unoptimized Matlab implementation.

More details about each sequence are described next:

Boat: The input sequence used to produce the panoramic boat image in Fig. 15 consists of 450 frames. The camera is relatively far from the scene, and therefore the variance of the depth was small relative to other sequences (such as the street sequence). This allowed us to limit the maximal disparity to 20 pixels, and reduce the run-time which is approximately linear in the number of labels. In addition, the iterative graph-cuts was applied on this sequence only once in 5 frames on average (The graph cuts method is performed only when the maximal disparity reaches a certain threshold, as described in Section 4).

Street: The length of the street as shown online and in Figs. 11-12 is about 0.5Km, and the sequence consists of 1800 frames. This sequence is very challenging, as it is sparsely sampled: The average disparity between pairs of frames is about 20 pixels, and the depth variations are also very large. Most stitching problems in this sequence are due to very close objects having a disparity larger than our pre-determined limit of 50 pixels. In this sequence, the iterative graph-cuts was called almost for each frame.

Shinkansen: The input sequence used to produce the panoramic image in Fig. 17 consists of 333 frames. The derailed Shinkansen train is not moving. Some intermediate depth maps and the motion parameters are also shown. A panoramic image of the same scene consisting of 830 frames appears in the homepage.

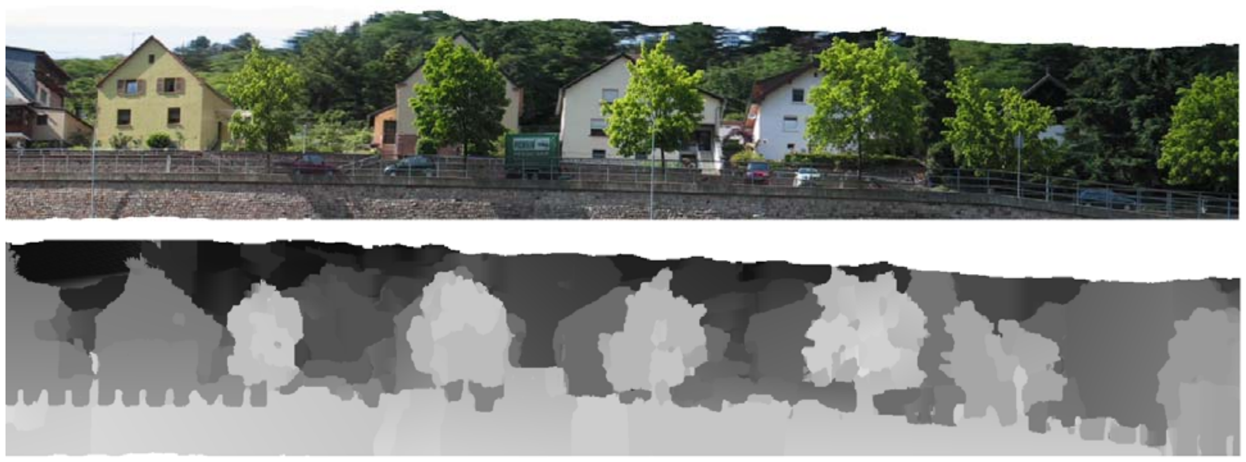


Figure 15: The panoramic image was constructed by X-Slits mosaicing from a sequence captured from a river boat. The corresponding panoramic inverse depth map is also shown.

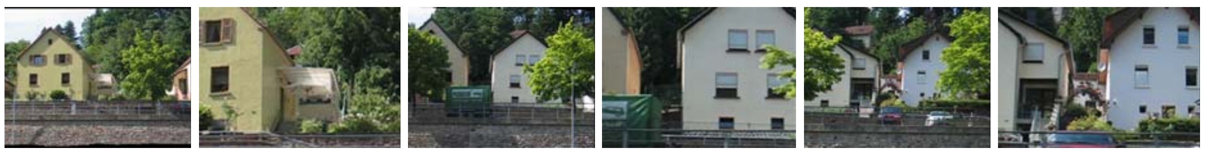


Figure 16: New synthetic views can be generated from the original boat sequence using X-Slits, resulting in a new sequence where the camera travels inside or outside the street, and it can even see behind the trees. Few frames from such a sequence are shown, and the full video is available at http://www.vision.huji.ac.il/mad.

8 Discussion: What Makes This Work

Our goal was to build a system that will enable the robust creation of urban and terrain models from cameras mounted on vehicles or aircraft scanning an urban scene or a terrain. The system should work for a reasonably captured video, even in cases that other methods fail. We believe that this goal has been achieved, and in this section we discuss the elements that contributed to that.

The most important decision was to restrict the motion model. A one dimensional translation (say horizontal), and rotations about the x and z axis, is the most appropriate motion model for a camera scanning a scene from a moving vehicle. While this restriction keeps most classes of videos applicable, it adds significantly to the robustness of the system. All three components of the motion are orthogonal, so no component can compensate for the other. Motion analysis works

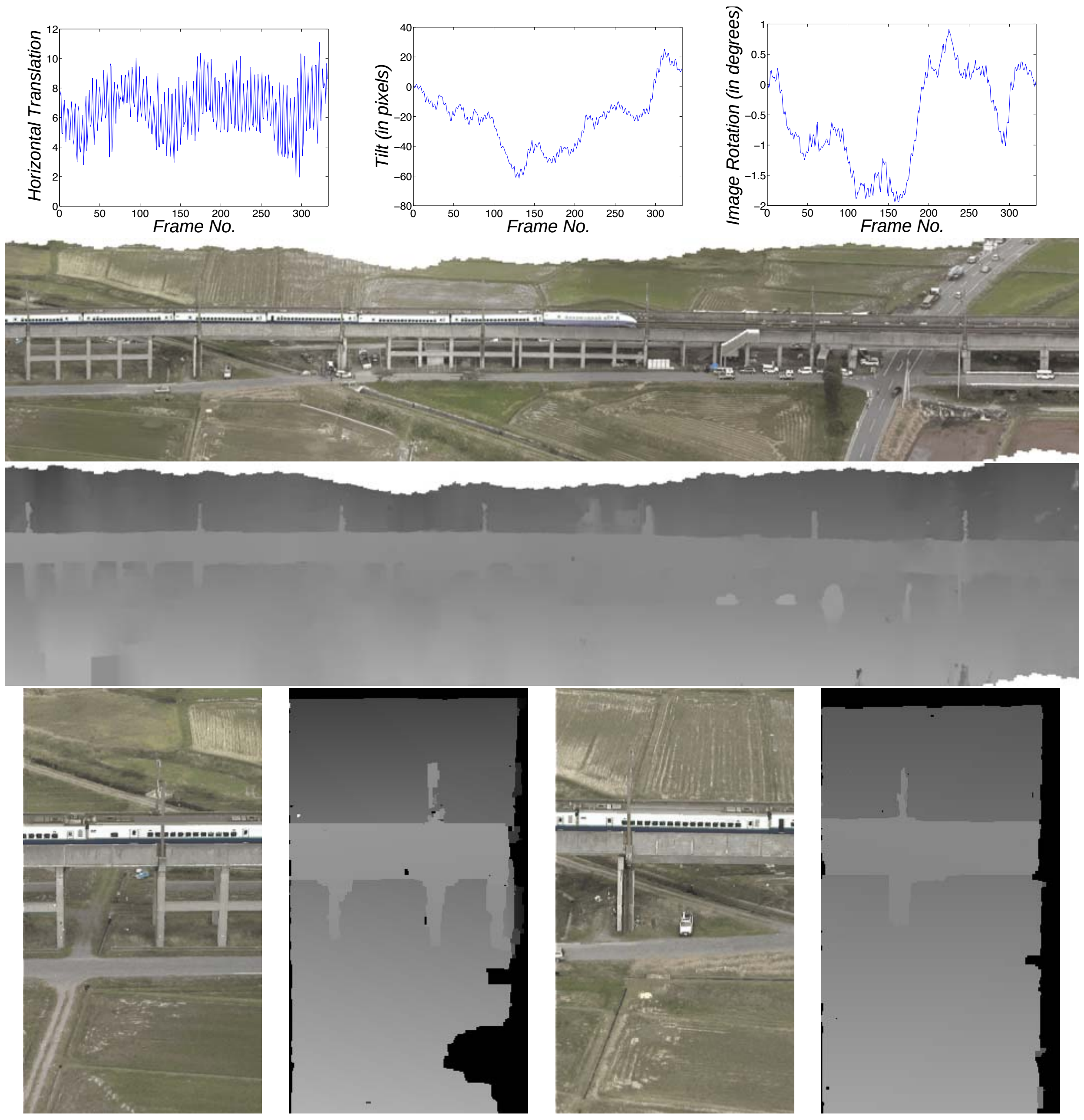
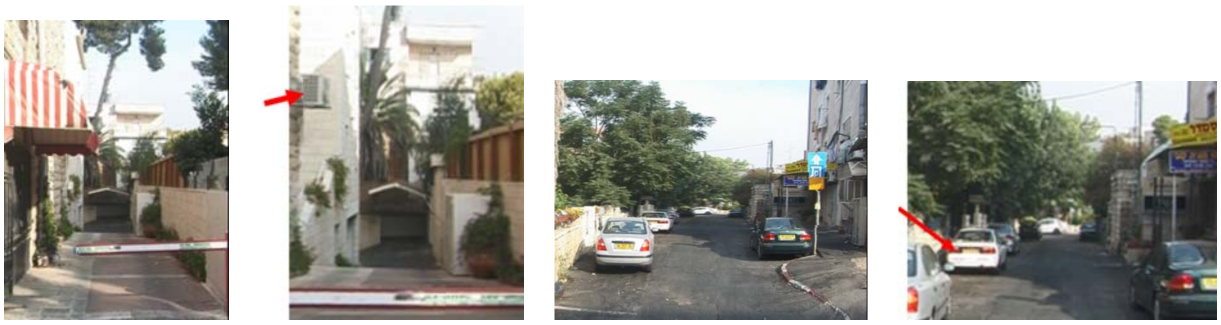


Figure 17: A panoramic view of a derailed Shinkansen train, with its corresponding inverse depth and motion parameters (T_x and accumulated tilt and image rotations). Inverse depth maps are also shown for a few frames (unknowns are marked in black).



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

Figure 18: A simulation of a forward motion from the original sequence moving sideways. (a) and (c) are synthesized to appear from a distance similar to the camera location, while (b) and (d) are synthesized from closer into the scene. Synthesis is done using X-Slits (slicing the space time volume). Objects marked by arrows were occluded in (a) and (c), and the changes in occlusions give a strong feeling of a walk-through.

perfectly well even in cases considered degenerate for more general motion models (e.g. a flat scene having a uniform depth).

The direct Lucas-Kanade motion computation using image intensities, avoiding feature detection, succeeds even when features are ambiguous or hard to detect. The sideways motion may also pose a problem for some feature based methods that need to track all features in all frames. In sideways motion feature points exist in a very small number of frames relative to other types of motion.

The depth computation of our system uses a graph cuts method to find planar surfaces in the scene. While we believe that other methods can replace the graph cuts method, the detection of scene surfaces rather than computing a single depth for each pixel enables better prediction of the depth of new frames from the depths computed in earlier frames. Another important element in the depth computation is the "flexible" modeling of correspondences between images. Rather than assuming purely horizontal displacements, we allow slight vertical displacements as well. This allows convergence even from a wrong initial motion model, and even under some camera distortions.

Temporal integration, that helped robustness in traditional 2D Lucas-Kanade [17, 28], was used as well. The computation of motion between the new frame and a temporal average of some recent frames is equivalent to simultaneous alignment of the current frame and all previous frames, which can overcome noise and corrupted frames.

While all the above elements contributed substantially to the robustness and success of the computation of camera motion, depth computation can never be perfect for all points at all locations. Ambiguity and occlusions always occur, in which case depth is impossible to compute. If no user interaction is used for correcting the scene depth, the available computed depth would not be sufficient for model-based rendering. Here comes the decision to use image-based rendering methods for creating panoramic images and for a virtual walkthrough.

The natural human motion is on the ground plane: left, right, forward, and backwards. All these motions are on a plane that includes the camera path, and they all can be simulated using X-Slits with no need for an accurate scene depth. We found that when rendering a narrow field of view, or when the scene is relatively flat, the distortion of the X-Slits projection compared to perspective projection is barely noticeable. These distortions become more visible when the depth in the scene is large compared to the distance from the camera, and when the field of view becomes wider.

When only a single panoramic image of the long scene is needed, without creating multiple views, MAD mosaicing creates an undistorted panoramic image. A pleasant side effect of MAD mosaicing is avoiding the truncation of objects where the depth computations fails (such as walking people that are always present when imaging city streets).

References

[1] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. *ACM Trans. Graph.*, 25(3):853–861, 2006.

[2] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. In *SIGGRAPH'06*, pages 853–861, July 2006.

[3] 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.

[4] Q. Yang amd L. Wang and R. Yang. Real-time global stereo matching using hierarchical

belief propagation. In *BMVC*, pages 989–998, Edinburgh, September 2006.

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

[6] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *PAMI*, 20(4):401–406, 1998.

[7] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *ICCV*, volume 1, pages 489–495, 1999.

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

[9] Y. Deng, Q. Yang, X. Lin, and X. Tang. A symmetric patch-based correspondence model for occlusion handling. In *ICCV*, pages 1316–1322, Washington, DC, USA, 2005.

[10] D. Feldman and A. Zomet. Generating mosaics with minimum distortions. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, volume 11, pages 163–170, Washington, DC, USA, 2004.

[11] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[12] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. *SIGGRAPH*, 30:43–54, 1996.

[13] K. Hanna. Direct multi-resolution estimation of ego-motion and structure from motion. In *MOTION91*, pages 156–162, 1991.

[14] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, second edition, 2004.

[15] L. Hong and G. Chen. Segment-based stereo matching using graph cuts. In *CVPR*, volume 1, pages 74–81, Los Alamitos, CA, USA, 2004.

[16] M. Irani, P. Anandan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. *PAMI*, 24(11):1528–1534, November 2002.

[17] M. Irani and S. Peleg. Motion analysis for image enhancement: Resolution, occlusion, and transparency. *Journal of Visual Communication and Image Representation*, 4:324–335, 1993.

[18] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *ECCV'92*, pages 282–287, 1992.

[19] H. Kawasaki, M. Murao, K. Ikeuchi, and M. Sakauchi. Enhanced navigation system with real images and real-time information. In *ITSWC2001*, October 2001.

[20] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *ICCV*, volume 2, pages 508–515, July 2001.

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

[22] M. Levoy and P. Hanrahan. Light field rendering. *SIGGRAPH*, 30:31–42, 1996.

[23] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[24] R. Montoliu and F. Pla. Robust techniques in least squares-based motion estimation problems. In *Progress in Pattern Recognition, Speech and Image Analysis*, volume 2905 of *Lecture Notes in Computer Science*, pages 62–70. Springer-Verlag, 2003.

[25] S. Ono, H. Kawasaki, K. Hirahara, M. Kagesawa, and K. Ikeuchi. Ego-motion estimation for efficient city modeling by using epipolar plane range image. In *ITSWC2003*, November 2003.

[26] M. Pollefeys, L. VanGool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *IJCV*, 59(3):207–232, 2004.

[27] A. Rav-Acha and S. Peleg. A unified approach for motion analysis and view synthesis. In *Second IEEE International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, Thessaloniki, Greece, September 2004.

[28] A. Rav-Acha and S. Peleg. Lucas-kanade without iterative warping. In *ICIP'06*, pages 1097–1100, 2006.

[29] A. Rav-Acha, Y. Shor, and S. Peleg. Mosaicing with parallax using time warping. In *Second IEEE Workshop on Image and Video Registration*, Washington, DC, July 2004.

[30] A. Román, G. Garg, and M. Levoy. Interactive design of multi-perspective images for visualizing urban landscapes. In *IEEE Visualization 2004*, pages 537–544, October 2004.

[31] A. Román and H. P. A. Lensch. Automatic multiperspective images. In *Proceedings of Eurographics Symposium on Rendering*, pages 161–171, 2006.

[32] M. Shi and J.Y. Zheng. A slit scanning depth of route panorama from stationary blur. In *CVPR'05*, volume 1, pages 1047–1054, 2005.

[33] Y. Wexler and D. Simakov. Space-time scene manifolds. In *ICCV'05*, volume 1, pages 858–863, 2005.

[34] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *PAMI*, 27(10):1644–1659, 2005.

[35] J.Y. Zheng. Digital route panorama. *IEEE Multimedia*, 7(2):7–10, April-June 2000.

[36] Z. Zhu, E. Riseman, and A. Hanson. Generalized parallel-perspective stereo mosaics from airborne videos. *PAMI*, 26(2):226–237, Feb 2004.

[37] A. Zomet, D. Feldman, S. Peleg, and D. Weinshall. Mosaicing new views: The crossed-slits projection. *PAMI*, pages 741–754, June 2003.