Dense Structure-from-Motion: An Approach Based on Segment Matching

Fabian Ernst, Piotr Wilinski, and Kees van Overveld

Philips Research, Prof. Holstlaan 4,
NL 5656AA Eindhoven, The Netherlands,
fabian.ernst@philips.com

Abstract. For 3-D video applications, dense depth maps are required. We present a segment-based structure-from-motion technique. After image segmentation, we estimate the motion of each segment. With knowledge of the camera motion, this can be translated into depth. The optimal depth is found by minimizing a suitable error norm, which can handle occlusions as well. This method combines the advantages of motion estimation on the one hand, and structure-from-motion algorithms on the other hand. The resulting depth maps are pixel-accurate due to the segmentation, and have a high accuracy: depth differences corresponding to motion differences of $1/8^{th}$ of a pixel can be recovered.

1 Introduction

Video technology at the turn of the century is facing the dilemma of finding features which increase product competitiveness. The third dimension has the potential of revolutionizing future video and TV products. Recent progress in computer vision brings closer the possibility of extracting depth information from video sequences. For many 3-D video applications, this depth information should be *dense*, i.e., every pixel should have depth information.

Several methods to extract depth out of video exist. Structure-from-motion (SFM) techniques are based on the extraction of structure from apparent motion. Two kinds of motion can be found in video sequences. The first one is due to camera motion: Objects at different depths have different apparent motion. The second one is independent motion of the objects. Handling independent motion of the objects requires handling of static scenes as a necessary first step. Therefore we assume for the time being that our scenes are static. Alternatively, we can extract depth from stereo sequences, allowing us to apply the same algorithm for all kinds of dynamic and static scenes.

SFM techniques have a long history in computer vision, and several overviews of techniques are available (for instance [9]). Feature-based algorithms [3] first extract predefined features, such as corners, and then match these. The difficulty in correspondence estimation lies in the size of the search space, and the presence of 'false correspondences'. Also features may disappear, or change their characteristics. Furthermore, the accuracy of feature estimation in images is often limited. The accuracy of the structure estimation can be improved using multiple frames, for instance through dynamic models [15]. Since the use of

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2351, pp. 217–231, 2002.

feature points is limited to regions in the image where enough information is contained, they are, in general, of limited use for direct dense matching.

A possibility to obtain dense depth maps is to use a feature-based technique as a starting point, and subsequently to generate a dense depth map [11]. Dense stereo matching methods [12] are commonly devised specifically for horizontal motions. Optical flow techniques are useful for obtaining dense depth maps in the presence of small displacements, but require regularization to obtain smooth depth maps [2]. This problem is avoided when we compare small regions of the image with corresponding regions in the following images [1,16]. For each region in the first image, the region in the second image is sought which is most similar to it according to a certain criterium (the *match penalty*). For depth reconstruction, we need additionally an (assumed) camera transformation, that converts a motion vector into a depth value. We assume that this camera transformation is available or can be estimated using calibration algorithms.

Generally, all pixels within a given region are assumed to move uniformly, and hence have the same depth. Square blocks of pixels are most common, and have been used with success in video coding applications [8] for displacement estimation and motion compensation. However, with a fixed given region shape (such as a block), within one region several objects with different depth values could be present, which leads to artifacts. Dividing the blocks into smaller entities can avoid this [13]; however, regions of constant depth in an image are commonly (much) larger than blocks. Hence, we propose to create *segments* corresponding to underlying objects, or more accurate, to segment the image in regions containing only a single depth. This leads to a chicken-and-egg problem (for the segmentation we need the depth; however, for depth estimation we use the segments). For foreground-background segmentation or scenes containing a small number of well-defined objects, one could iteratively solve for scene structure and segmentation [22]. For general video scenes, we have to make a key assumption (which is valid for a large amount of video footage, but of course fails in some situations): *Discontinuities in depth coincide with discontinuities in color.*

In the remainder of this paper we discuss our dense structure-from-motion algorithm (DSFM), which is based on segment matching. In section 2, we discuss the choice of our error norm and how to minimize it in order to find the depth per segment. We also shortly explain the segmentation algorithm. Due to the presence of noise and regions with low texture, the depth map may contain artifacts. We apply a post-processing procedure for smoothing depth artifacts, taking into account depth uncertainty intervals. Results are shown in section 3. To improve the matching, we introduce a way of handling occlusions in section 4.

2 Dense Structure-from-Motion

2.1 General Overview

Our dense SFM algorithm consists of the following key components:
Camera calibration. We have to determine the internal geometric and optical characteristics of the camera (intrinsic parameters) and the 3-D position and

orientation of the camera"s frame relative to a certain world coordinate system (extrinsic parameters). This is required to enable the conversion of an apparent motion to a depth value. For instance, if the camera motion consists of horizontal translation, the parallax is inversely proportional to depth. Several calibration methods have been described in the literature [4,18]. An overview of general algorithms is given in [6]. Camera calibration is not discussed in this paper.
Image segmentation. We assume that depth discontinuities coincide with color discontinuities. Segmentation divides the image into regions of more or less constant color, and hence, depth. The segmentation algorithm is discussed in section 2.4.
Segment matching. This is the key element of this paper and is discussed in sections 2.2 and 2.3. It results in a pixel-wise depth map, annotated with depth uncertainty intervals.
Postprocessing. This *relaxation* algorithm is discussed in section 2.5 and allows to improve the accuracy of the depth maps with information from neighboring segments.

2.2 Segment Matching

The core of our dense structure-from-motion algorithm (DSFM) is the matching of segments in the first image I_0 to find their location in the next image I_1. To each segment S, a depth $\hat{d}$ is assigned which minimizes the following error norm, the so-called *match penalty* [5]:

$$E(d) = \sum_{\mathbf{x} \in S} |I_1(\mathbf{x} + \Delta\mathbf{x}(d)) - I_0(\mathbf{x})|, \tag{1}$$

i.e., we estimate for all pixels $\mathbf{x}$ in a segment S their location in I_1 based on the proposed depth, and compute the absolute difference between the colors at their positions in I_0. The camera transformation relates the depth value d to a motion vector $\Delta\mathbf{x}$. Note that we do *not* match a segment with a segment of frame I_1, since this would put high requirements on the time-consistency of the segmentation.

This matching approach comes from the domain of motion estimation, for instance for scan-rate conversion [5] or MPEG compression [7]. The key difference between our method and existing matching approaches is that in our case the region S is an arbitrarily shaped segment, instead of a square region. Another difference is that the minimization is 1-D (over depths d), instead of 2-D (the two components $(\Delta x, \Delta y)$ of a motion vector). In case of non-integer pixel displacements, we use bilinear interpolation to estimate color values at inter-pixel positions.

Solving equation (1) not only results in a depth per segment, but can give us also information on its accuracy by considering the error curve: the error as function of the depth (see figure 1 for an illustration). We define the boundaries of the depth uncertainty interval as:

$$d_l = \max\{d : d < \hat{d} \wedge E(d) > E(\hat{d}) + \Delta E\}, \tag{2}$$

$$d_u = \min\{d : d > \hat{d} \wedge E(d) > E(\hat{d}) + \Delta E\}, \tag{3}$$

i.e., the depth values closest to $\hat{d}$ for which the match penalty is still above a predefined threshold.

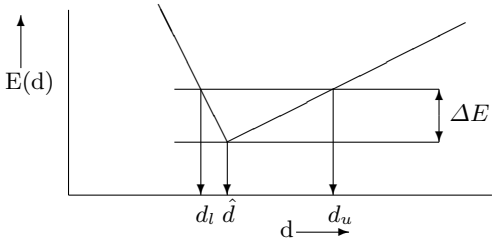


Fig. 1. Illustration of the depth accuracy. The error as function of depth has a global minimum in $\hat{d}$. The lower and upper depth bounds are given by d_l and d_u according to expressions (2) and (3).

2.3 Solving the Minimization Problem: The "Candidates" Concept

A naive version of matching would do a full search over all possible depths d. For applications requiring real-time performance, full search is computationally too expensive. Hence the number of depths to be evaluated has to be reduced, however, without affecting the quality of the result. For block-based motion estimation, a fast method for solving minimization problem (1) has been proposed [5] which uses for each block only motion vectors $\Delta\mathbf{x}_i$ coming from a small candidate set (CS). This CS is composed of motion vectors of some well-selected neighbors of the block under consideration, and some random vectors. We use a modified version of this algorithm by assigning a depth value d to a segment as a result of minimizing the match error $E(d)$ over a limited set of candidate depth values. It is assumed that the candidates sample the function $E(d)$ sufficiently dense. Since we deal with arbitrarily shaped segments, the number of neighboring segments is a priori unknown and might be different for each segment. To keep the notion of a small CS, we might reduce the number of candidates by taking only neighboring segments into account with similar color or reliable depth.

Before the first iteration the depth values are initialized in a random way. During the iterations the candidate depth with the minimal value of the match penalty is taken as the new estimate. The iteration procedure is halted as soon as all the depths of the segments have converged to their final value; this typically takes about four iterations. An estimate of the depth accuracy (expressions (2) and (3)) is obtained by storing the results for the tested depth candidates.

2.4 Segmentation

Segmentation groups pixels into a number of (4-connected) segments, such that every pixel is part of exactly one segment. For our DSFM algorithm, membership of a pixel to a segment should be decided based on depth. However, before the matching we do not know the depth yet, but we assume that depth discontinuities coincide with color discontinuities. Since depth discontinuities can only occur at

segment boundaries (because of the matching process), our main requirement is that color discontinuities should coincide with segment boundaries, and that the positioning of these boundaries is *pixel-accurate*. Image segmentation methods can be in general divided in feature-based and region-based methods (see [14] for an overview of state-of-the-art techniques). The aim of feature-based methods is to detect and then combine features (edges, gradients, ...), to obtain structures. Region-based methods use region measures (texture, color, ...) to find zones where certain properties are (nearly) constant. In this paper, we use for stability reasons a region-based method, similar to a watershed method [20,21].

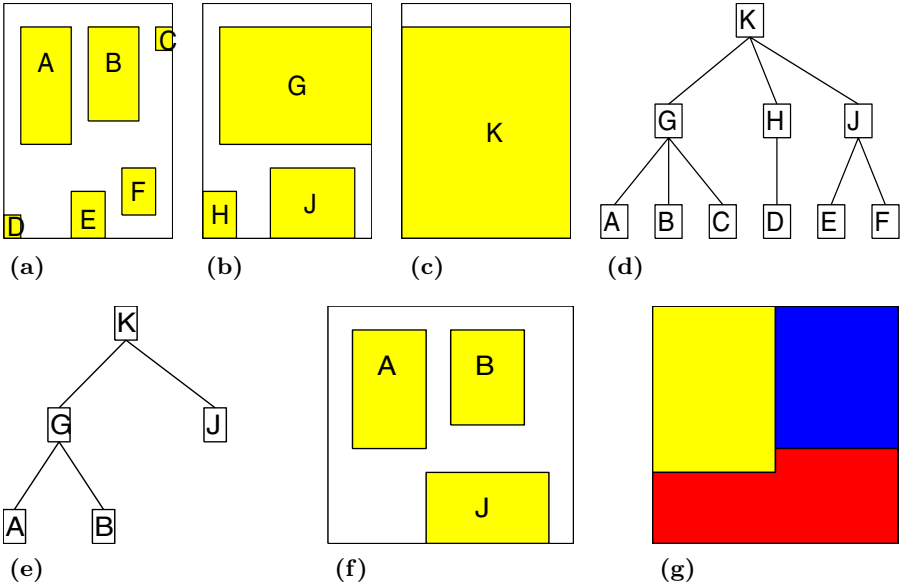


Fig. 2. Illustration of the hierarchical segmentation. **(a),(b),(c)**: The resulting cores at three different levels of p: p_0, p_1, p_2. **(d)** The tree structure arising from the core segments. **(e)** The pruned tree. Note that different parts of the tree have a different value for p. **(f)** The final cores. **(g)** The segmentation after growing the cores.

Many segmentation techniques suffer from the fact that there is no clear connection between the parameters specified by the user (commonly some kind of threshold p on the level of homogeneity) and the final segmentation. For our application, a priori we do not know the complexity nor content of the scene, and it is hard to specify this threshold p in advance. Furthermore, to accommodate for variations in the image, the threshold should be region-dependent. This would increase the amount of freedom in the choice of the parameters, which is undesired. Therefore we study the behavior of the segmentation when p is increased continuously. For a given value of p, let us denote a segmentation by $\mathcal{S}(p)$, where

$$\mathcal{S}(p) = \{S_i(p), i = 1, \ldots, S\}\,. \tag{4}$$

S_i is the i^{th} segment in the image and S is the number of segments. The parameter p can be any kind of threshold. We define C_i as the *core* of segment i, such

that $C_i(p) \subset S_i(p)$. For our hierarchical approach we now require that

$$\forall i \;\; \forall q \geq p \;\; \exists j \text{ such that } C_i(p) \subset C_j(q). \tag{5}$$

This requirement means that we can build a tree structure for the cores. If we take the cores as homogeneous regions in the image, they form the inner parts of the final segments (hence the name 'core').

Our segmentation algorithm now consists of the following steps:

1. A homogeneity criterium satisfying expression (5) is defined to decide whether a pixel should be part of a core at homogeneity level (threshold) p.
2. We generate a family of cores by letting this threshold vary ($p = p_0, p_1, \ldots, p_N$). For each value of p, we compute the resulting cores.
3. A tree structure can be built for the cores, due to requirement (5). If p increases, cores grow larger, and cores which are separated for a low value for p might merge at a higher level of p (see figure 2).
4. To create a segmentation, the tree is *pruned*. We define a binary *validity constraint* $V(C_i, m)$, which indicates whether a core C_i satisfies some desired property. It has one free parameter m. For instance, $V(C_i, m)$ may check whether core C_i has a color variance smaller than m. The tree is pruned such that every core satisfies the validity constraint. Note that we have in this manner replaced the original threshold p with an alternative threshold m. Pruning the tree such that each core satisfies $V(C_i, m)$ is equivalent to setting p differently in different regions of the image. However, where in general a region-dependent threshold increases the amount of freedom in the parameter choice, here it is a *result* of a fixed requirement. Moreover, a clear requirement on the segmentation (such as, say, the color variance of a segment) can not be expressed directly in terms of an initial threshold p, but can now be specified through $V(C_i, m)$.
5. The segments are grown from the cores by assigning non-core pixels to a core. The growing is based on the color gradient. The distance within the cores is zero by definition, for a non-assigned pixel the distance to the cores is calculated as the sum of the absolute value of the color difference on any path connecting the pixel with the core, using a distance transform algorithm. The pixel is then assigned to the core for which this distance is minimal. This means that if there is a color discontinuity between two cores, the segment border is coincident with this discontinuity.

A simple illustration of this concept is shown in figure 2.

2.5 Relaxation: Using Depth Accuracy for Post Processing

The accuracy of the depth estimate depends on the amount of information available in the region of interest, and as a result noisy depth values may occur in places of low texture. The accuracy also depends on the camera motion: for instance, in two images with purely horizontal camera displacement, a texture consisting of horizontal stripes gives a large depth uncertainty.

In this section, we present a method to improve the quality for depth maps. The method is designed to correct outliers and other noise, but to preserve

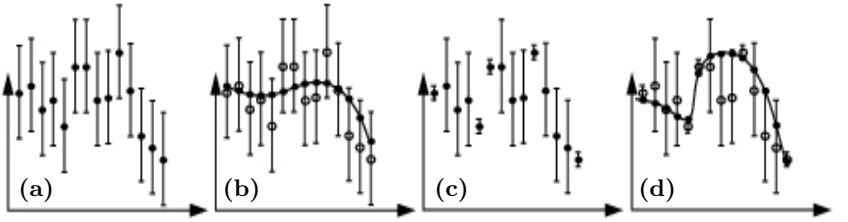


Fig. 3. Illustration of the relaxation algorithm for a 1-D signal. **(a)** The initial signal $\{d_i\}$ (black circles). **(b)** The smooth signal $\tilde{d}$ is given by the solid line and the black circles; the original signal d_i is drawn as open circles. Since the error bounds are large, the final $\tilde{d}_i$ can deviate arbitrarily far from the initial d_i. **(c)** Now some d_j have small error bounds. **(d)** In regions with large depth uncertainties, the resulting $\tilde{d}_i$ values are smooth; however, if the error intervals are small the smoothing is restricted by the error bounds and depth discontinuities are preserved.

real depth discontinuities. Conventional regularization methods typically add a term λE_s to the match penalty which accounts for smoothness. If $\lambda = 0$, a rather instable solution results; for larger λ the depth map is smoother but it may deviate more from the actual values. Also, depth discontinuities may get washed out. Anisotropic operators E_s do not smooth across discontinuities, so that edges are preserved. However, their computation is expensive. Moreover, anisotropic smoothing suffers from a chicken-and-egg problem: The smoothing operator assumes to have detected a large gradient, and preserves this large gradient. So effectively, it only smoothes those areas that already are in some sense 'smooth' (i.e., have low gradient values). But if a noise feature happens to have a (locally) coherent nature, it may get mistaken for a gradient by the anisotropic smoother, and hence it is not removed.

If there is no further information available, it is not clear how to do better than anisotropic smoothing. But in the case of segment-based depth estimation, there are two additional information channels: the depth uncertainty (expressions (2) and (3)), and the assumption that depth discontinuities coincide with color discontinuities. Here we discuss a smoothing algorithm for depth maps that takes advantage of these two additional sources of information. It is based on an extension of an iterative relaxation method, which has been studied earlier in computer vision [17,10]. Applications and the study of some special cases have been published in [19]. Let N_i be the set of indices j of d_i's neighbors. Then we set

$$\tilde{d}_i \leftarrow \tilde{d}_i + \alpha \frac{|N_i|}{|N_i|+1}((\frac{1}{|N_i|}\sum_{j\in N_i}\tilde{d}_j) - \tilde{d}_i), \text{for all } i; \tag{6}$$

$$\tilde{d}_j \leftarrow \tilde{d}_j - \alpha \frac{1}{|N_i|+1}((\frac{1}{|N_i|}\sum_{j\in N_i}\tilde{d}_j) - \tilde{d}_i), \text{for } j \in N_i, \tag{7}$$

where all assignments occur simultaneously.

Consider the case (figure 3c) where for some indices j, the initial signal d_j has a small error interval. Then, clearly, the value of $\tilde{d}$ should not be allowed outside this interval. This can be achieved during the iteration by clamping any new value $\tilde{d}_j$ against the borders of its error interval. Given for each pixel its estimated depth $d(\mathbf{x})$ and associated uncertainty $(d_l(\mathbf{x}), d_u(\mathbf{x}))$, we search for a smooth function $\tilde{d}(\mathbf{x})$ such that

$$d_l(\mathbf{x}) \leq \tilde{d}(\mathbf{x}) \leq d_u(\mathbf{x}), \quad \forall \mathbf{x}. \tag{8}$$

In this way, depth discontinuities survive smoothing if there is sufficient evidence for a discontinuity in terms of the uncertainties. For clarity, we show the concept of the relaxation algorithm on a 1-D signal in figure 3.

For a regular 2-D grid (blocks), where $|N_i| = 4$ for all i, it can be verified that the process converges (if it converges) to a state where the resulting 2-D arrangement of samples lies on a bi-cubic polynomial surface, and hence is optimally smooth, provided that no clamping against error intervals occurs. For an arbitrary topology we cannot give an analytic description for the converged state. Still, since the filter is a direct generalization of the 2-D regular tesselation-case, we may expect that again a smooth surface results.

The second information channel to constrain the relaxation is the color. Consistent with the earlier assumption that depth discontinuities and color discontinuities coincide, we propose to take only neighboring segments into account in equations (6) and (7) with small color differences with the actual segment. This preserves discontinuities even at places where the error bounds are too large to clamp the new d-values.

This entire procedure is iterated over all segments until the updates are sufficiently small. This typically takes 3-4 iterations.

3 Results

We have applied our algorithm to numerous different scenes. In this section we show some typical results, and highlight some of the characteristics of the dense structure-from-motion algorithm.

The first example applies DSFM to a scene depicting a doll house. In this case, the camera trajectory was known, and consisted of a horizontal translation. The typical parallax between subsequent images in the sequence is of the order of 5 pixels. In figure 4a-b we show the images for which we do DSFM. The segmentation is shown in figure 4c. All pixels which satisfy

$$\max(|C(x', y') - C(x, y)|, |x - x'| \leq 1, |y - y'| \leq 1) \leq p \tag{9}$$

are taken as core pixels on level p. Here, C is the color of a pixel. For noise reduction reasons, in the segmentation step a low-pass filtered version of the image is used. It can easily be seen that cores based on equation (9) satisfy the tree criterium specified in equation (5), since pixels which are core pixels at level p are also core pixels at all higher levels. As a validity constraint, we require that the color variation in a core should be smaller than 8% of the dynamic color range.

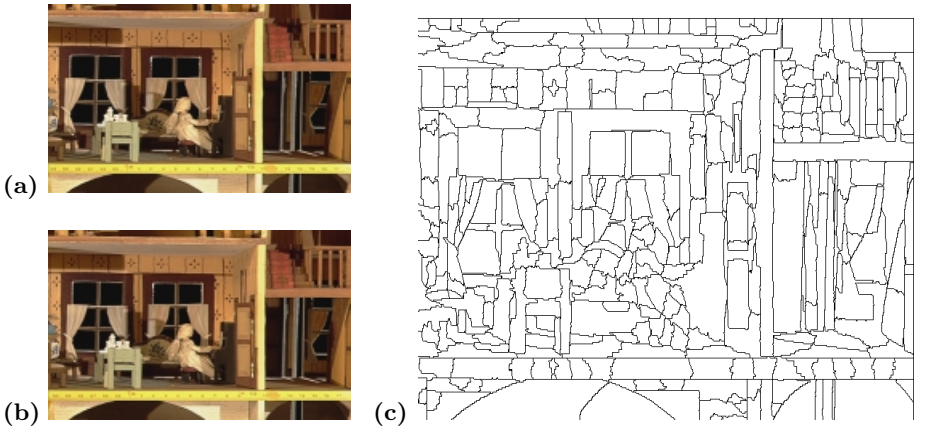


Fig. 4. **(a), (b)** Two subsequent frames from the doll house sequence. The parallax between the frames is on the order of 5 pixels. **(c)** The segmentation of the frame in (a). There are segment boundaries at all depth discontinuities; some oversegmentation is visible.

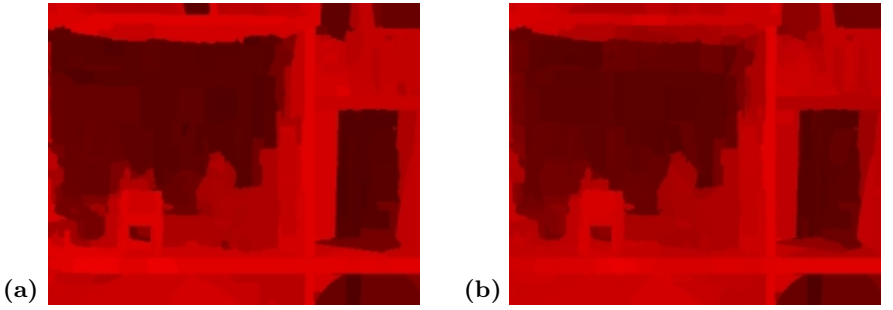


Fig. 5. **(a)** Raw depth map for the doll house. Bright color means small depth, dark color means large depth. Some outliers are visible, for instance near the middle of the upper boundary, and below the doll. **(b)** Depth map after the relaxation procedure. The depth map is smoothed at the location of the discontinuities, but the depth discontinuities between doll and background and between the legs of the chair are not affected. The color scale is the same as in (a).

We have pruned the tree by removing all cores with a higher color variation. It can be clearly seen that there is some form of oversegmentation. However, for depth recovery this is not a problem: an object which is split into multiple segments can still have a single depth. After applying the DSFM algorithm, the raw depth map is shown in figure 5a. The overall quality is quite good, however, some depth outliers are present at the top of the image in the middle, and below the doll. Depth discontinuities around the doll and on the edge of the piano are recovered well. The resulting depth map after application of the relaxation algorithm is shown in figure 5b. The 'real' depth discontinuities have not been affected by the smoothing, whereas the outliers have been removed. The depth discontinuities around the tea pot on the table have even been sharpened, be-

cause the uncertainty of those segments allows an assignment of the background depth to those segments. To indicate the accuracy of the matching, we show error curves for segments containing the head of the doll and the curtain adjacent to it in figure 6. Here, the value of the match penalty as a function of segment displacement (parallax, and thus depth) is displayed. There is a clear and pronounced global minimum for both segments, and the depth difference between the two segments is clearly visible. Note that it is very well possible to find the parallax with a subpixel accuracy. This error curve is typical for pairs of images containing sufficient texture and having a good camera calibration. From our experience, parallax differences up to $1/8^{th}$ of a pixel can be detected reliably.

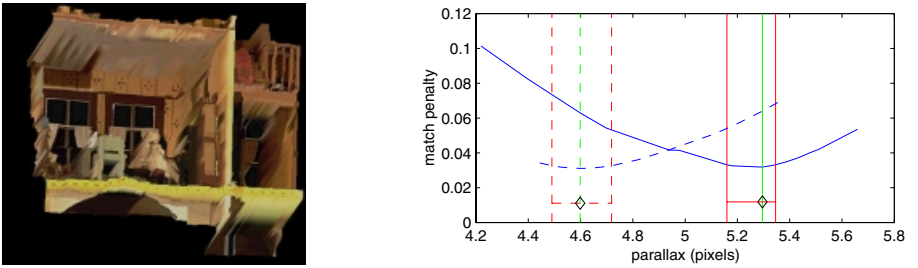
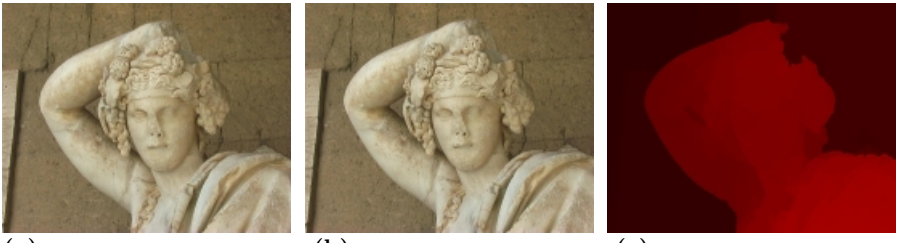


Fig. 6. **(a)** Screen shot from the 3-D reconstruction of the doll house, where the viewpoint has been moved to the lower left. **(b)** Error curve for the doll's head (solid line) and the right curtain of the right window (dashed line) as a function of the parallax (in this case corresponding to reciprocal depth). The optimal value and error bounds surrounding it are denoted by the vertical bars. A pronounced global minimum can be discerned, and it can be estimated to high sub-pixel accuracy (up to $1/8^{th}$ of a pixel).



(a) **(b)** **(c)**

Fig. 7. **(a), (b)** Frames of the Dionysos sequence. **(c)** The resulting depth map after applying the DSFM algorithm. A pixel-accurate distinction between foreground and background is made; however, also depth differences within the statue are reconstructed (see also figure 8).

The second example is based on a hand-held video sequence of a statue of Dionysos (see figure 7a-b). In this case, the camera position and orientation were not known, but have been estimated with a camera calibration algorithm.

In this case, the camera motion consisted of both a rotation and a translation. The final depth map is shown in figure 7c. We see a pixel-accurate distinction between foreground and background. However, to show that also details in the face can be recovered, we have plotted error curves of the segments containing the nose and the eyes in figure 8. The very small parallax difference (of the order of 0.25 pixels) has been recovered; for comparison, the background is at a depth of 3.4 units, and maximal parallax differences are on the order of 7 pixels. This means that DSFM does not only have a high accuracy, but also a high dynamic range.

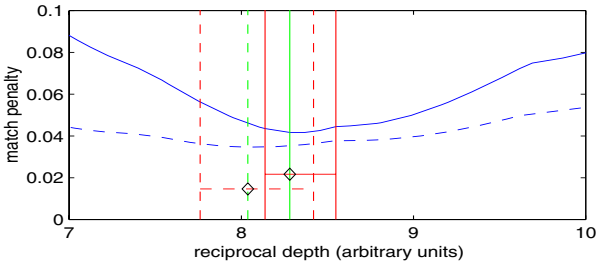


Fig. 8. Error curve for the nose of Dionysos (solid line) and the right eye (dashed line). The optimum values and its error bounds are denoted by the vertical bars. Even the small depth difference between eyes and nose (corresponding to a difference in segment motion of approximately 0.25 pixels) can be discerned. For comparison, the background has a reciprocal depth of approximately 3.4 units.

4 Handling Occlusions in Depth Estimation

In the computation of the match penalty (1), we estimate for all pixels in a segment their location in the next image, based on the proposed depth, and compute the absolute difference between the colors at the positions in the first (I_0) and second (I_1) image. However, it may occur that the point X corresponding to a pixel $\mathbf{x}$ which is visible in the first image, is not visible in the second image, because another part of the scene occludes it (for instance due to camera movement, see figure 9). In that case, the contribution of this pixel to the match penalty has no relation to the depth of the segment. Therefore, we propose a modification of the match penalty, where we only sum over those pixels of a segment which are also visible in the second image:

$$E(d) = \sum_{\mathbf{x} \in S} v(\mathbf{x}) |I_1(\mathbf{x} + \Delta\mathbf{x}(d)) - I_0(\mathbf{x})|, \tag{10}$$

where $v(\mathbf{x})$ is defined as

$$v(\mathbf{x}) = \begin{cases} 1, & \text{if } X, \text{ corresponding to } \mathbf{x} \text{ in } I_0, \text{ is visible in } I_1 \\ 0, & \text{if } X, \text{ corresponding to } \mathbf{x} \text{ in } I_0, \text{ is not visible in } I_1 \end{cases} \tag{11}$$

The function $v(\mathbf{x})$ is called the *visibility map*. In order to compute the visibility map, the following steps have to be carried out for each pixel $\mathbf{x}$:

1. Set $v(\mathbf{x}) = 1$ everywhere.
2. Estimation of the depth for the segment containing $\mathbf{x}$ in I_0 using expression (10).
3. Computation of the point $\mathbf{X}$ in 3-D space which projects onto $\mathbf{x}$.
4. Computation of the resulting location and depth of $\mathbf{X}$ in I_1.
5. Set $v(\mathbf{x}) = 1$ if this is either the only pixel at that location in I_1, or the pixel with the smallest depth of all pixels which end up at that location. Else set $v(\mathbf{x}) = 0$.
6. If the process has not converged yet, go to step 2.

The correct handling of occlusion removes a source of distortion in the computation of the match penalty, resulting in a more accurate depth estimate.

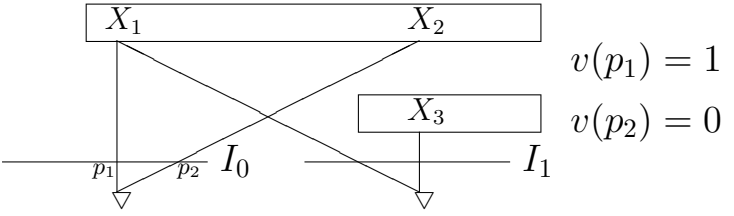


Fig. 9. Illustration of the visibility map concept. The location X_1, corresponding to p_1, is visible in the first (left) and second (right) image. The location X_2, however, is only visible in the first image (as pixel p_2), but not in the right image, since it is occluded by the shape containing X_3.

To show the effect of occlusion handling, we have applied DSFM to a stereo sequence depicting a farm scene. Due to the large camera baseline, there are large occlusion areas. We focus especially at the area around the child's head, where parts of the fence are occluded (see figure 10). The segmentation of that area is shown in figure 11a. Figure 11b shows the error curve for the shaded segment (one of the posts of the fence). Without taking occlusion into account, there are two pronounced minima, of which - of course - only one corresponds to the real depth. The other minimum corresponds to a parallax where this post is mapped onto another post, resulting in a wrong depth (see figure 12a). With the visibility map, the adapted error curve still has two minima, but the global minimum is now much lower (see figure 11b). The final depth map is shown in figure 12b.

5 Conclusion and Discussion

We have presented a dense structure-from-motion algorithm (DSFM) based on segment matching. Its major advantage is its robustness in providing dense depth maps, since region instead of point correspondences are found between images. After segmentation, the depth and depth accuracy of each segment is estimated by minimizing a match penalty. Post processing removes outliers and does a depth-discontinuity preserving smoothing of the depth field. Occlusions are explicitly taken into account through an adaptation of the match penalty. In con-

Fig. 10. Part of the left and right image of a frame of the 'Farm' stereo sequence. Parts of the posts of the fence in the left image are occluded by the child's head in the right image, which may hamper the depth estimation.

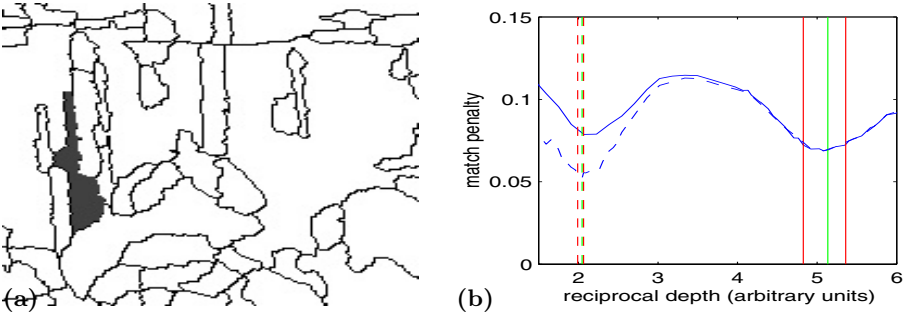


Fig. 11. **(a)** Segmentation of the left image of figure 10. The shaded segment is partly occluded in the right image. **(b)** Error curves for the shaded segment. The solid curve is the error curve for the basic match penalty (1). The minimum corresponding to an incorrect depth is lower than the correct minimum due to occlusion effects. If the occlusion effect is taken into account with expression (10) (the dashed line), the minimum corresponding to the correct depth is lower and hence taken.

trast to stereovision methods, this method can be used for arbitrary camera motions (not only horizontal translation).

In the examples, we have demonstrated the robustness and accuracy of the algorithm. A depth accuracy corresponding to motion differences of $1/8^{th}$ of a pixel is obtainable, also in low-textured regions. DSFM is especially suited for the case of small motions and small motion differences. Because regions are matched, the correspondence problem typically encountered in feature-based methods can to a large extent be avoided. Since segments are relatively large and hence contain more information, DSFM is more robust than feature-based approaches. Increasing the segment size does not decrease the resolution, however, since the segment borders are aligned with color, and hence depth, discontinuities.

Although the DSFM algorithm gives a quite satisfactory depth estimation for (static or stereo) video sequences, there is still some room for improvement, e.g. in the parameterization of the depth within a segment. A challenging re-

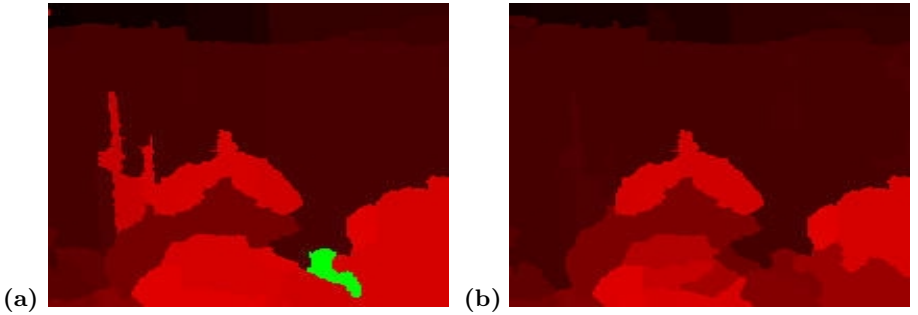


Fig. 12. **(a)** Depth map without taking occlusion into account. Parts of the fence are too close to us. **(b)** Depth map when the visibility is taken into account. The fence now has the correct depth.

search subject is still the estimation of depth from dynamic video sequences (the "independent motion" problem).

DSFM has several possible applications, from scene modeling to object extraction and tracking. In the domain of digital television, 3-D reconstruction is a necessary element to convert existing 2-D video for future 3D TV applications. Depth reconstruction algorithms, if implemented in hardware, may be efficient enough to do the computation in real time. The present algorithm is very regular, and due to the candidate-based minimization algorithm also very efficient. A slight change of the algorithm allows for segment-based motion estimation, which has prominent applications in scan-rate conversion and MPEG encoding.

Acknowledgement. We would like to thank Marc Pollefeys (KU Leuven) for providing the Dionysos sequence. Financial support from the ITEA BEYOND project is gratefully acknowledged.

References

1. M. Accame, F.G.B. De Natale, and D. Giusto. Hierarchical block matching for disparity estimation in stereo sequences. In *ICIP95*, pages 374–377, 1995.
2. G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. PAMI*, 7:384–401, 1985.
3. S.T. Barnard and W.B. Thompson. Disparity analysis of images. *IEEE Trans. PAMI*, 2:333–340, 1980.
4. H.A. Beyer. Some aspects of the geometric calibration of CCD cameras. In *ISPRS Intercomm. Conf. on Fast Processing of Photogrammetic Data*, Interlaken, 1987.
5. G. de Haan and P. Biezen. Sub-pixel motion estimation with 3D recursive search block matching. *Signal Processing: Image Communication*, 6:229–239, 1994.
6. R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge University Press, 2000.
7. MPEG-4 Video group ISO WG11. MPEG-4 overview (Maui Version). Technical Report ISO/IEC/JTC1/SC29/WG11 N3156, ISO, 1999.
8. J.R. Jain and A.K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Trans. Comm.*, 29:1799–1808, 1981.

9. Tony Jebara, Ali Azarbayejani, and Alex Pentland. 3D structure from 2D motion. *IEEE Signal Processing Magazine*, pages 66–84, May 1999.
10. J.L. Mallet. Discrete smooth interpolation in geometric modelling. *Computer Aided Design*, 24:178–191, 1992.
11. M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Metric 3D surface reconstruction from uncalibrated image sequences. In *Proc. SMILE Workshop (post-ECCV'98), LNCS 1506*, pages 138–153. Springer-Verlag, 1998.
12. P.A. Redert, E.A. Hendriks, and J. Biemond. Correspondence estimation in image pairs. *IEEE Signal Processing Magazine*, 16:29–46, 1999.
13. R. Rodrigues, K. van Overveld, and P. Wilinski. Depth reconstruction based on irregular patches. In *Proc. EPCG no 9*, Marinha Grande, Portugal, 1999.
14. P. Salembier and F. Marques. Region-based representations of image and video: segmentation for multimedia services. *IEEE Trans. CSVT*, 9:1147–1169, 1999.
15. S. Soatto and P. Perona. Reducing "Structure from Motion": A general framework for dynamic vision. part 1: Modeling. *IEEE Trans. PAMI*, 20:933–942, 1998.
16. H. Tao, H.S. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *Proc. ICCV*, pages 532–539, Vancouver, Canada, 2001.
17. D. Terzopoulos. The computation of visible-surface representations. *IEEE Trans. PAMI*, 10:417–438, 1988.
18. R.Y. Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV camera lenses. *IEEE Journal on Robotics and Automation*, RA-3:323–344, 1987.
19. C.W.A.M. van Overveld. The application of relaxation and optimisation methods in computer aided geometric design. In B. Özgüc and V. Akman, editors, *Proc. of First Bilent Comp. Graphics Conf.*, pages 161–180, Ankara, Turkey, 1993.
20. L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. PAMI*, 13:583–598, 1991.
21. D. Wang. Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Trans. CSVT*, 8:539–546, 1998.
22. A. Yezzi and S. Soatto. Stereoscopic segmentation. In *Proc. ICCV*, pages 59–66, Vancouver, Canada, 2001.