Reconstructing Depth from Spatiotemporal Curves

Rui Rodrigues rpr@tom.di.uminho.pt
António Fernandes af@di.uminho.pt
Universidade do Minho, Portugal

Kees van Overveld overv@natlab.research.philips.com
Fabian Ernst fabian.ernst@philips.com
Philips Research, Eindhoven

Abstract

We present a novel approach for 3D reconstruction based on multiple video frames taken from a static scene. Our solution emerges from the spatiotemporal analysis of video frames. The method is based on a best fitting scheme for spatiotemporal depth curves, which allows us to compute 3D world coordinates of the objects within the scene. As opposed to a large number of current methods, our technique deals with random camera movements in a transparent way, and even performs better in these cases than with pure translation. Robustness against occlusion and aliasing is inherent to the method as well.

1 Introduction and Context

Structure from Motion (SfM) is a computer vision field still in progress. It deals with the problem of recovering the 3D structure of a scene from different perspective projections (e.g. video frames taken with a moving camera) [1]. The range of applications of SfM techniques includes 3D-scene modelling, virtual view generation, 3D TV, image/video synthesis and autonomous navigation.

The availability of various perspective projections allows us to estimate depth (the distance to the camera) of objects by comparing the projections' relative displacement of such objects in different frames, and using knowledge about camera motion.

Several problems arise in this seemingly simple process. The first problem is related to camera information, which often is not readily available and has to be estimated as well. This is the camera calibration problem.

Second, it is not trivial to determine which parts of a set of digital images correspond to the same object or local feature in 3D space. This is known as the correspondence problem [2]. Some factors that contribute to the correspondence problem are image noise, periodic textures and the occlusion of objects.

A third problem arises due to numerical or geometric instability: the stability problem.

The techniques described in the extensive literature available on SfM range from block matching algorithms to stochastic techniques, texture-based to feature-based. Many of the concepts are inherited from motion estimation research.

A large number of techniques analyse the case of consecutive frame pairs or triplets (e.g. trilinear tensor), estimating motion (depth) for each pair or triplet, and integrating that estimated data overtime as a post-processing operation. These techniques face stability problems in the fusion of the estimated data.

The reader is referred to [1] for an overview and references on SfM methods.

In this paper we focus on fusion. One way to perform this fusion still at the stage of motion/depth estimation is to regard video data as 3D information, time being the third dimension [3]. In this context, a sequence with a given number of frames can be represented as a colour distribution on the spatiotemporal domain, resulting in a spatiotemporal volume – the VideoCube. If the camera motion between consecutive frames is relatively small, the similarity between frames allows identifying spatiotemporal curves and surfaces, corresponding to the temporal path of objects throughout the scene.

The shape of these spatiotemporal entities is related to both the camera motion and the position of identifiable points in world space.

We propose a SfM method which exploits the VideoCube assuming that camera parameters (both intrinsic and extrinsic) are known, or at least well estimated.

The method estimates the depth of a set of points chosen from the video images based on a set of depth candidates and a best-fit metric of the spatiotemporal curves corresponding to those candidates. It is assumed that the scene is static with little or no highlights.

The main features of our approach are:

-The ability to deal with arbitrary (including non-smooth) motion paths

-The potential to combine benefits of frame-to-frame coherence (meaning few occlusion differences on short time scale) with large baseline (meaning that due to the large total camera paths, geometric stability can be achieved). Furthermore, all frames can be used simultaneously.

- The correspondence problem is tackled with a stochastically stable matching technique

- Robustness to occlusion, noise and aliasing is inherent to the method

The paper is organised as follows: The VideoCube is introduced in section 2, along with a brief overview of the literature in the area of video spatiotemporal analysis, followed by the concept of spatiotemporal curves in Section 3. In section 4, a technique for estimating spatiotemporal curves is derived and later summarised in an algorithm. Section 5 contains the results obtained with a current implementation of the algorithm. Section 6 compares the technique with other SfM and spatiotemporal based techniques, showing the main differences. Finally, conclusions and future work close this paper in sections 7 and 8, respectively.

2 The VideoCube

Assume that each point that belongs to an object is identified by its 3D co-ordinates and can be mapped to one colour. The set of all points P_i with a colour defines the world w:

$$w = \{ \langle x,y,z \rangle \rightarrow \textbf{color} \mid \forall \langle x, y, z \rangle \in \textbf{OBJECTS} \} \quad (1)$$

A perspective camera placed in this world can be described by the camera parameters cp

$$cp = \langle e, h, v, k, f \rangle \quad (2)$$

where e is the camera position, h, v, k are the normal vectors (horizontal, vertical and look) defining camera orientation and f is the focal length of the camera.

Consider a finite plane π lying at distance f from the eye and perpendicular to k – the *projection plane* (Figure 1.a). There is a subset of w that can be projected on this plane.

The projection $P' = \langle i, j \rangle$ of a world point P in π is the intersection of the projection line eP with π.

The depth of a point and its projection are defined as follows:

$$\textbf{Pdepth}(P, cp) = cp.k \bullet (P - cp.e) \quad (3)$$
$$\textbf{Pproj}(P, cp) = \langle cp.h \bullet (P - cp.e), cp.v \bullet (P - cp.e) \rangle * f / \textbf{Pdepth}(P, cp)$$

There is a large set of world points that project to P' (all the world points that lie on eP). Due to occlusion, only the point closest to the camera is registered in π.

$$\textbf{Cproj}(\langle i, j \rangle, cp) = w[P] \mid \text{MIN}_{P \in \text{dom}(w) \wedge \textbf{Pproj}(P, cp) = \langle i, j \rangle} (\textbf{Pdepth}(P, cp)) \quad (4)$$

An image is therefore the set of world point colours projected in all its pixels, considering occlusion and given a set of camera parameters:

$$\textbf{image}(cp) = \{ \langle i, j \rangle \rightarrow \textbf{Cproj}(\langle i, j \rangle, cp) \mid \forall \langle i, j \rangle \in \pi \} \quad (5)$$

Consider now that the camera is moving, following a path cpt, associating a new set of camera parameters at each time instant when an image is recorded:

$$cpt[t] = \{ t \rightarrow \langle e_t, h_t, v_t, k_t, f_t \rangle \} \quad (6)$$

A video sequence is defined as the set of images obtained along the camera path:

$$\textbf{vs}(cpt) = \{ \textbf{image}(cpt[t]) \mid \forall t \in \textbf{dom}(cpt) \} \quad (7)$$

This video sequence however also can be seen as tri-dimensional data, namely bi-dimensional data varying along the time dimension. This leads to the VideoCube concept (Figure 1.b): the spatiotemporal volume representing projected colour as a function of position in π and time (see Figure 2 for an example):

$$\textbf{vc}(cpt) = \{ \langle i, j, t \rangle \rightarrow \textbf{Image}(cpt[t])[i, j] \mid \forall \langle i, j \rangle \in \pi, \forall t \in \textbf{dom}(cpt) \} \quad (8)$$

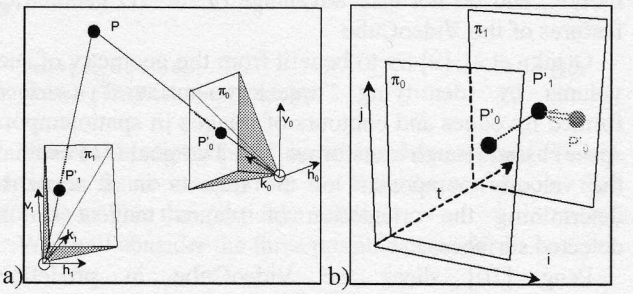
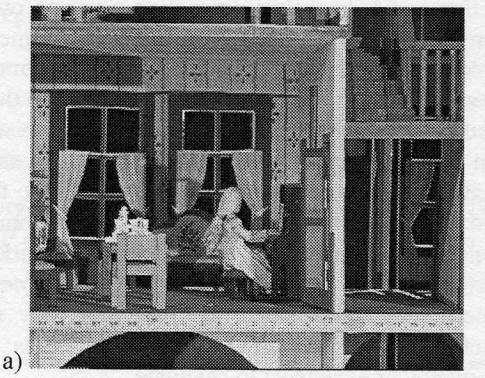


Figure 1 *Two frames of a sequence when viewed in:*
a) 3D-world space
b) Spatiotemporal space



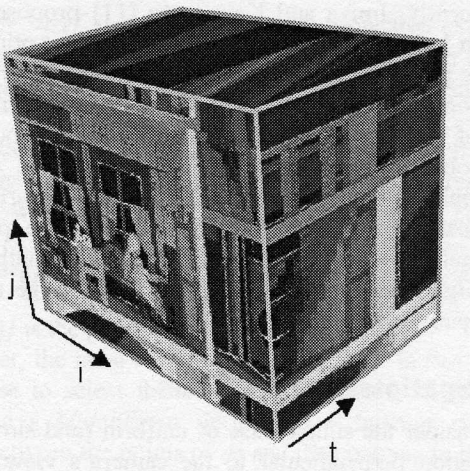


Figure 2 *A VideoCube example*
a) One frame of the "doll house" scene
b) The corresponding VideoCube

2.1 Previous VideoCube-related works

One of the papers that first mentioned the VideoCube, and the "motion as orientation" effect of the paths formed in the t direction, was done by Adelson and Bergen [3]. It is

oriented to visual perception, and proposes to detect motion models based on energy and impulse response filters.

Although the works by Duc et al. [5], Wang et al. [6] and Moschenni et al. [7] are based on spatiotemporal analysis, they are oriented to segmentation based on motion, and do not take advantage of the 3D geometrical features of the VideoCube.

Otsuka et al. [9] try to benefit from the geometry of such volume by identifying "trajectory surfaces" (surfaces formed by edges and contours of images in spatiotemporal space) using Hough transforms [12]. The goal is to estimate the velocity component of the objects in a scene by determining the orientation of planes tangent to the detected surfaces.

Peng [10] slices the VideoCube in predefined orientations, and divides such slices into strips to detect line orientations that next are converted to optic flow.

Kim's work [8] on spatiotemporal analysis for edge detection and optical flow estimation gives an overview of the problems with spatiotemporal analysis. He mentions the under-sampling in time dimension on common video sequences, and the lack of continuity inside the volume caused by that under-sampling and by image noise.

On a more probabilistic approach, Caplier and Luthon [4] extended Markov Random Fields (MRF) to the spatio-temporal model, defining a Markov Random Volume (which is an extension of concepts such as Markov chains and MRF).

These techniques share the fact that they are either restricted to simple camera motion models or small movements, or expensive to calculate.

Recently, Imiya and Kawamoto [11] proposed another Hough transform based approach. It uses a voting scheme to rate point correspondences over a series of frames, and to reconstruct world points. The authors randomly select a pair of points from the spatiotemporal data and check if they obey to the epipolar constraint. If so, a vote is accumulated to the corresponding world point. Reconstruction takes place by choosing the points that accumulated a larger number of votes. This method results in a high computational load, due to the large number of point pairs required.

3 Spatiotemporal Curves

Consider the simple case of uniform (and slow) camera translation perpendicular to the camera's view direction. Assume that the camera moves in a horizontal world plane $y = S$.

In the plane $j = S$ in the VideoCube, one can easily identify (nearly straight) line patterns (see Figure 3 and Figure 4). These lines are related to the relative apparent displacement of the objects, due to camera motion.

The slope of these lines is a function of the distance of the objects to the camera. Lines that are nearly parallel to the time axis correspond to objects more distant (small apparent motion - large depth) and lines with sharper angles correspond to closer objects (large apparent motion – small depth).

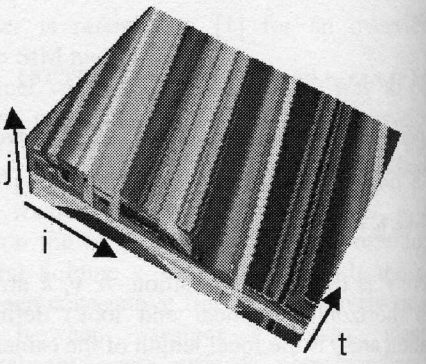


Figure 3 *A sliced VideoCube*

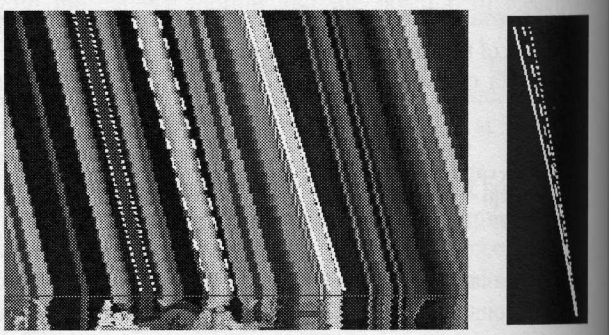


Figure 4 *Spatiotemporal lines and slope differences*

Hence, if all camera parameters are known, depths can be estimated from the slopes of these lines, and a reconstruction is possible.

In this simple case, a properly parameterised Hough transform can detect the lines, and compute accurate estimates of the slopes (see [9] for an example).

However, if we consider a more complex camera motion model, such as a piecewise rotation or a superposition of rotation and translation, creating parameterised motion models is not feasible. In addition, the practical implementation of Hough transforms with a parameter space of more than two dimensions also becomes unfeasible.

Nevertheless, it is clear that a video sequence resulting from a given camera movement yields spatiotemporal curves whose shape is related to the camera motion.

4 Spatiotemporal Curve Estimation

4.1 Goal

We aim to solve the following problem: given a VideoCube containing a set of implicit spatiotemporal

curves, obtain the set of 3D points that originated such curves.

4.2 Requirements

The scene must contain only static objects. This assures that any apparent motion in frames is only due to camera motion. For this same reason, lighting must also be constant, and there should be little or no highlights in the scene. The position of a highlight varies with camera motion, yielding the same result as with a moving object.

The method assumes that camera parameters such as focal length, trajectory, and orientation are well estimated.

4.3 Defining the Set of Interest Points *(sip)*

Estimating depth for all points (and respective spatiotemporal curves) in the *vc* is not feasible. Our approach reduces the set of points to be estimated by considering only points that lie on contours in the individual images.

For this purpose, a transformation of the VideoCube is performed applying an edge detection filter to each image. The points to be used in the depth estimation are those which are present in a transformed VideoCube (*tvc*), defined as:

$$tvc = \{ \langle i, j, t \rangle \rightarrow contours(vc, i, j, t) \mid \forall \langle i, j, t \rangle \in dom(vc) \} \quad (9)$$

where *vc* is the original VideoCube. Points that are successfully detected as part of a contour are assigned TRUE; all other points are assigned FALSE.

The set of interest points (*sip*) is therefore

$$sip = \{ P' \mid P' \in dom(tvc) \wedge tvc[P'] = TRUE \} \quad (10)$$

4.4 Estimating depth for SIP

Depth estimates can be obtained by trying to trace the spatiotemporal curves that exist on the VideoCube. However, the development of a tracing algorithm for spatiotemporal curves is not trivial due to aliasing.

We propose an alternative approach: to search depths for a given point $P' \in sip$ by matching the implicit VideoCube spatiotemporal curves with a set of candidate depth curves for P'. A depth curve is defined as a spatiotemporal curve generated based on a candidate depth.

A depth curve can be generated in two steps. First, we define a reverse projection of a chosen spatiotemporal point *P'* of coordinates $\langle ir, jr, tr \rangle$, with given camera parameters *cpt[tr]* and an attributed depth *d* as follows:

$$Dproj(\langle ir, jr \rangle, d, cpt[tr]) = cp.e + d * (ir * cp.h + jr * cp.v + cp.f * cp.k) \quad (11)$$

This reverse projection gives us a point in world coordinates. The corresponding depth curve *stc* is defined as:

$$stc(\langle ir, jr, tr \rangle, d, cpt) = \{ t \rightarrow \langle Pproj(Dproj(\langle ir, jr \rangle, d, cpt[tr]), cpt[t]) , t \rangle \mid \forall t \in dom(cpt) \} \quad (12)$$

The set of candidate depth curves $scdc_{P'}$ is thus defined as:

$$scd = \{ d_0, d_1, \ldots, d_n \} \quad (13)$$
$$scdc_{P'} = \{ stc(P', d, cpt) \mid d \in scd \}$$

The real spatiotemporal curve *rsc* that contains *P'* in a particular frame is unknown. Let P be the world point that projects as P' in that particular frame.

We shall consider the three possible cases for P:

a) *P* has a projection in all frames and its projections are all identified as contour points.

$$\forall t, \langle Pproj(P, cpt[t]), t \rangle \in dom(tvc) \wedge \forall t, tvc[Pproj(P, cpt[t]), t] = TRUE \quad (14)$$

b) *P* has a projection in all frames but in at least one frame it is not identified as a contour. This can be due to the occlusion of *P*, aliasing problems, faulty edge detection, or noise in the original VideoCube.

$$\forall t, \langle Pproj(P, cpt[t]), t \rangle \in dom(tvc) \wedge \exists t : tvc[Pproj(P, cpt[t]), t] = FALSE \quad (15)$$

c) The projection of *P* lies outside *tvc* in at least one frame.

$$\exists t : \langle Pproj(P, cpt[t]), t \rangle \notin dom(tvc) \quad (16)$$

Let $P' = \langle iref, jref, tref \rangle$ be a point projection existing in a reference frame *tref*, and $dc \in scdc_{P'}$. The depth curve *dc* includes *P'* and is built based on a given depth estimate d_e for *P'*.

In the ideal case a), *rsc* intersects a contour in all frames in *tvc*. If the depth estimate for *P'* in the reference frame is correct, *dc* will also intersect a contour in all frames, i.e.:

$$\forall t, tvc[dc[t]] = TRUE \quad (17)$$

where *dc[t]* is the point intersection of *dc* at frame *t*.

However, the most common situation is b). In this case, we propose to select the best depth for *P'* from *sdc* by minimizing a *MatchError* defined as a function of the distance between the points of *dc* and the contours in *tvc*. A distance transform [13] of *tvc* is used to provide the required distances.

$$dtvc(tvc) = \{ \langle i, j, t \rangle \rightarrow distanceToContour(tvc, [i, j, t]) \mid \forall \langle i, j, t \rangle \in dom(tvc) \} \quad (18)$$

$$MatchError(stc, dtvc) = \sum_{t} dtvc[stc[t], t] \quad (19)$$

The metric *MatchError* will have a theoretical minimum when d_e matches the real depth d_r. We refer to it as "theoretical minimum" because, due to aliasing problems and noise, the depth corresponding to the actual minimum may be different from d_r. We shall come back to this issue later, for now we will accept the theoretical minimum as the real minimum.

The problem of depth curve estimate for a point $P' \in sip$, where P' belongs to a frame t can now be stated as:

$$\mathbf{MIN_d\ MatchError(\ stc(<P', t>, d, cpt),\ dtvc\ )} \qquad \mathbf{(20)}$$

A search in depth space can be performed in order to find the minimum of *MatchError*. Using this approach, even with aliasing and noise, we can expect to find a depth d with a *MatchError* value fairly close to zero, i.e. below an error threshold. The minimisation process uses the Brent minimisation algorithm as described in [14], so that for each point, the initial *sdc* is extended with new depth guesses provided by the algorithm.

However, degenerate cases may occur. *MatchError* can have multiple well-separated local minima below a minimum threshold for different depths (Figure 5.a). This happens, for instance, when a depth curve intersects by coincidence different contours in different frames.

In this case, we have a number of clearly distinct depth approved candidates for a point. Points close to the original point can be tested and if the situation persists then the point should be dismissed.

Another possible degenerate case is when the camera path is a simple translation and some contours are aligned with the direction of the camera movement. In this case we can expect an interval of depths with a fairly low error (Figure 5.b). Random camera movements will eliminate these situations.

In general, random camera movement means more information about the scene can be extracted as opposed to simple translations or rotations of the camera. The method treats random camera movements in a transparent way.

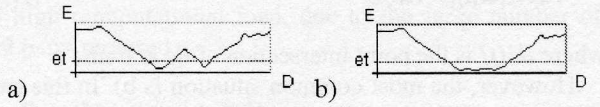


Figure 5 *Match Error as a function of Depth:*
a) multiple well separated local minima
b) Interval of depths with a low error

Let us consider now what happens when occlusion occurs (Figure 6.a). In such cases, the search is unlikely to achieve *MatchError* values below the error threshold for any depth value. An analysis of the individual frame errors for each depth should be performed to look for frame error sequences of values above and below a frame error threshold. If such sequences do exist then we can infer that the point is indeed occluded, and individual frame errors above the frame error threshold should be dismissed. A minimum length for sequences where the frame error values are below the threshold is required in order to have confidence in the estimate.

Finally, case c) occurs when the depth curve estimate lies outside the *tvc* in some frames (Figure 6.b). It doesn't make sense to compute a *MatchError* for these frames and therefore they should be disregarded. If the number of frames that can be used to compute *MatchError* is below a threshold, then the resulting value should be given a low level of confidence.

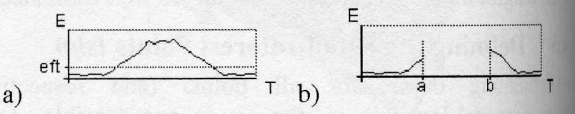


Figure 6 *Match Error as a function of Time (Frames)*
a) A possible occlusion case.
b) The estimates for frames in the interval [a,b] are outside the VideoCube.

4.5 The Algorithm

An algorithm that applies the concepts discussed so far to the reconstruction of video scenes is now outlined.

```
Build vc from video frames
Build dtvc
Build sip
For each P' ∈ sip
   sdc = Initial depth candidates
   While MatchError not minimised
      Generate sdc_P' from sdc
      Read match values from dtvc
      Add new values to MatchError function
      Generate new sdc using Brent's Alg.
   Analyze errors to decide P' validity
   Attribute final depth (world coord) to P'

/* For visualization*/
Build connectivity graph from valid points
in the transformed reference frame
Triangulate and texture
```

5 Results

In this section, we present some results obtained with the current implementation of the algorithm.

Sub-section 5.1 uses a simple synthetic scene to illustrate situations where occlusion and extra-VideoCube depth curves exist. Sub-sections 5.2 and 5.3 present the reconstruction of a synthetic and a real scene, respectively.

5.1 Occlusion and Extra-VideoCube ST curves

For this section, we built a synthetic scene consisting of a camera rotating 180° around a set of three objects (Figure 7) – a torus and two thin blocks.

Figure 7 *Some frames of the sequence*

The *tvc* corresponding to this sequence can be seen in Figure 8, where the thick contours correspond to the TRUE values. A *sip* was chosen from the points of the first (left) frame, and the lines crossing tvc are the depth curves corresponding to the selected depth for each point in *sip*.

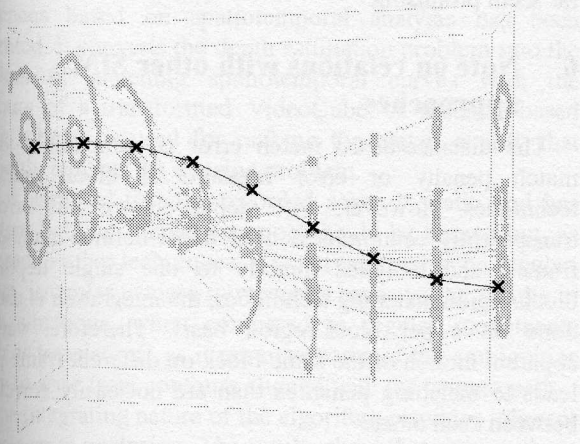


Figure 8 *A representation of a tvc corresponding to 9 frames (the thick contours correspond to the TRUE values) and estimated depth curves for a sip*

Notice the outlier depth curve (darkest line) extending from the inner circle of the torus. This is an example of a depth curve that, although it doesn't match any real spatiotemporal curve, intersects all contours by coincidence. Notice also how most points in the two blocks have good depth curves, despite being occluded by the torus on the frames corresponding to the centre of the VideoCube. Figure 9.a) illustrates a *scdc* for a single point, and Figure 9.b) shows the selected depth curve.

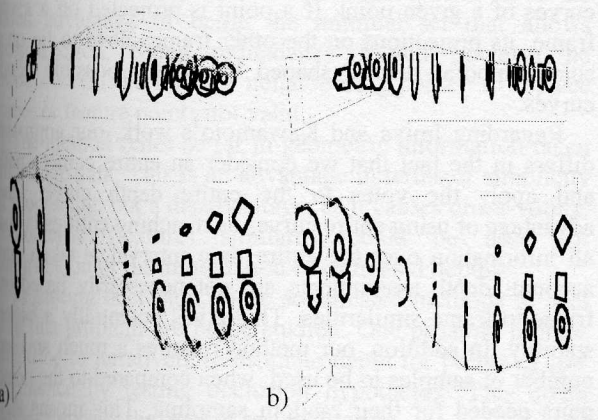


Figure 9 *Side and top views of a)* scdc *for a point, and b) the selected* dc

One can see that the point is well traced outside the VideoCube, although the section of the *dc* outside the VideoCube is not used in the matching. This occurs because there is enough evidence in the remaining frames to get a good estimate.

5.2 Synthetic scene

This scene is made up of 20 coloured boxes, arranged in a circle. The camera moved with random changes within a small interval in translation, rotation angle and axis. The original scene can be seen in Figure 10. The camera positions are represented by small dots (bottom left of figure). The small line segments extending from each dot represent the camera's orientation, and the line connecting them is the camera path.

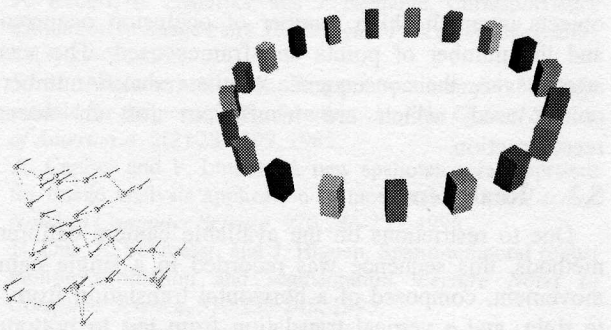


Figure 10 *The synthetic scene, including the camera positions*

Figure 11 shows three frames from the 40 used for the reconstruction.

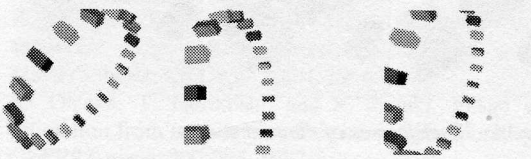


Figure 11 *Three frames of the synthetic scene*

A sip of 400 points was processed, from which 279 where considered valid. Figure 12 shows the valid reconstructed points (black) superimposed on the ground truth data (grey).

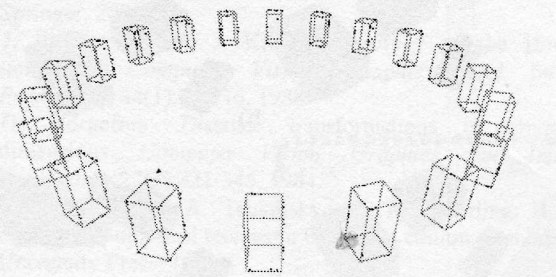


Figure 12 *Point reconstruction vs. ground truth*

The final reconstruction, after triangulation and texturing, can be seen in Figure 13.

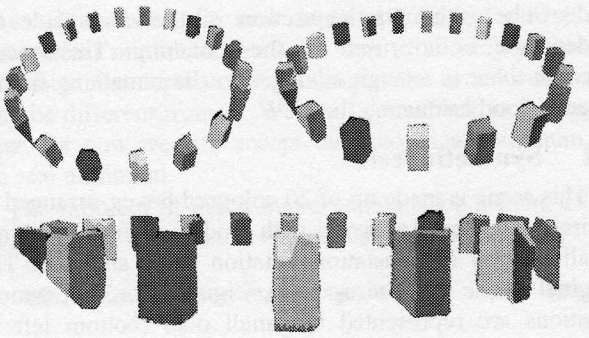


Figure 13 *Three views of the reconstruction*

The reconstructed boxes lie in a circumference as in the original frame. These results show good localisation of objects given the high number of occlusion occurrences and the number of points and frames used. The visible artefacts are the consequence of the reduced number of points used, which are insufficient for an accurate reconstruction.

5.3 Real scene

Due to restrictions on the available camera calibration methods, this sequence was recorded in a single camera movement, composed of a horizontal translation from left to right, and a vertical translation from top to bottom. It contains three objects, as can be seen in Figure 14, where a subset of the 40 frames used is shown.

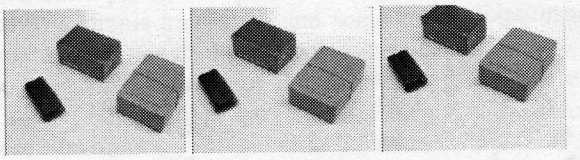


Figure 14 *Three frames of the real scene*

In the scene reconstruction of Figure 15, a sip of 300 points was built, from which 155 where considered valid.

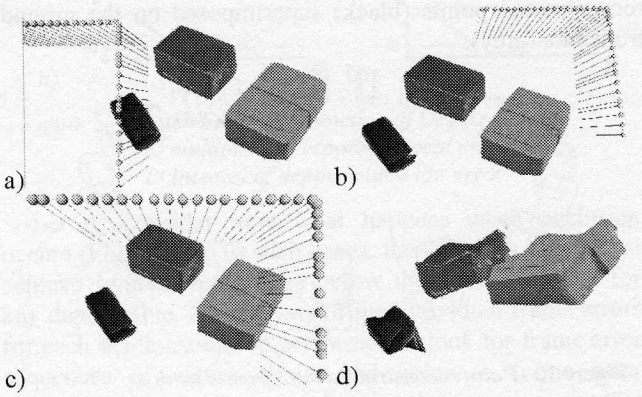


Figure 15 *Four views of the reconstructed scene*

Regarding Figure 15, a), b) and c) show the reconstruction from viewpoints close but outside the original camera path. In these three cases, the reconstruction preserves the shape and texture of the original.

In d) a view from a position and orientation radically different from the original cameras was used. Although some distorting artefacts can be seen, the results are still good in terms of localisation. If we take into account the fact that no regularisation is in use, the results obtained so far seem promising.

6 Note on relations with other SfM approaches

The distance-based match error is comparable to the match penalty or error measure of block matching techniques. However, with depth curves, this error is transparently accumulated over a pre-defined number of frames, and assigned directly to one single depth. In block/region matching techniques, the integration is usually done on a per block/region basis. Therefore, varying apparent motion of the same block on different frame pairs leads to matching penalties than are not easily correlated between each other.

Our method does not rely on camera's simple motion models of translation or rotation, as opposed to some of the classical SfM methods, or spatiotemporal methods, as the ones presented in section 2. The assumption of irregular camera motion on our spatiotemporal framework is one of its strong points. In fact, as seen in the previous sections, an irregular motion of the camera can disambiguate some situations where scene objects would be aligned with a regular camera motion. Furthermore, the technique does not rely on either small or smooth camera motion between successive frames.

Implicit constraints to reduce the search space found on other matching techniques, such as epipolar geometry, also have their dual on the surface swept by the possible depth curves of a given point. If a point is projected on a given frame, its projections on the other frames have to lie on a spatiotemporal surface shaped by all the possible depth curves.

Regarding Imiya and Kawamoto's work, our approach differs in the fact that we consider an entire camera path and apply the votes to the entire depth curve. One advantage of using entire curves as matching entities is that all information over the entire time interval is taken into account: depth assignments are not necessarily based on frame-to-frame similarities. This gives potentially a better stability. In addition, our method requires a much smaller number of samples to be used, when compared to the point pairs needed for their random sampling. This means that the computational load of our method is substantially lower.

Finally, classical point tracking methods have the disadvantage of losing track of points when they are occluded for some frames, being able to track them later again, but not able to link the two tracks. In addition, although they do not require special camera motion, they require small steps between adjacent camera positions. Our method overcomes both of these problems transparently.

7 Conclusions

A new technique for depth reconstruction from video sequences based on spatiotemporal analysis has been presented. It converts the depth estimation problem into the problem of matching spatiotemporal curves with the contours of a transformed VideoCube. A distance-based rating scheme is used for ranking the match quality that implicitly reduces the aliasing problem.

It is assumed that the recorded scene is static and has constant lighting. Camera motion should be known but, as opposed to other techniques, it is not required to be regular. In fact, irregular camera movements improve the results in some situations, when compared to regular movements (such as translation), as it removes some of the ambiguities likely to arise from contours aligned with the camera path.

The integrating nature of the algorithm provides inherent robustness to occlusion. The results show that even without regularisation, depth estimations with consistent localisation and reconstruction can be achieved.

8 Future Work

It is known that regularisation plays an important role in 3D reconstruction. It is expected that a careful implementation will improve this method as well. Such regularisation should remove outliers resulting from the degenerate cases presented in 4.4, while preserving meaningful depth discontinuities, as the ones between objects in the foreground and in the background.

The set of interest points *sip* can be based on the set of corners and feature points in *tvc*. These points have the added advantage of being the minimal set that allows reconstruction. Some redundancy should however be introduced (in the form of neighbouring points) to replace corner or feature points not valid.

The method relies so far in geometric information taken from edge data. This is an advantage in terms of simplicity of processing, but a disadvantage, as it causes cross-voting. Photometric (texture) similarity could be used for confirmation or disambiguation of estimated depths.

Regarding occlusion, more sofisticated methods to deal with this problem are being studied. One possible process would be to iteratively re-estimate depth for points of low confidence by testing occlusion hypotheses. This could further improve the overall reconstruction.

Other common problems such as noise in the source video data, inaccurate camera calibration and sensitivity of edge detection algorithms also require further testing.

Acknowledgements

The work presented here is supported by Grant PRAXIS XXI/BD/20322/99, sponsored by Fundação para a Ciência e Tecnologia (the Portuguese Foundation for Science and Technology).

References

[1] T. Jebara, A. Azarbayejani and A. Pentland. 3D structure from 2D motion. IEEE Signal Processing Magazine, 16(3), 1999.

[2] A. Redert, E. Hendriks, and J. Biemond. Correspondence estimation in image pairs. *IEEE Signal Processing Magazine*, 16(3):29-46, 1999.

[3] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *J. of the Optical Society of America A*, 2(2):284-299, 1985.

[4] A. Caplier and F. Luthon. A new spatiotemporal approach for image analysis application to motion detection. *Lecture Notes in Computer Science*, 970:246-253, 1995.

[5] B. Duc, P. Schroeter, and J. Biguen. Spatio-temporal robust motion estimation and segmentation. *Lecture Notes in Computer Science*, 970:238-245, 1995.

[6] J. Y. A. Wang and E. H. Adelson. Spatio-temporal segmentation of video data. Technical Report 262, MIT Media Lab Vismod, 1994.

[7] F. Moscheni, S. Bhattacharjee and M. Kunt. Spatiotemporal segmentation based on region merging. *IEEE TPAMI*, 20(9):897-915, September 1998.

[8] Z. Kim and K. Wohn. Spatio-temporal analysis of image sequence: Edge detection and optical flow estimation. Master's thesis, CS Dept., Kaist, 1996.

[9] K. Otsuka, T. Horikoshi, and S. Suzuki. Image velocity estimation from trajectory surface in spatiotemporal space. In *CVPR97*, pages 200-205, 1997.

[10] S. Peng. Temporal slice analysis of image sequences. In *CVPR'91*, pages 283-288, 1991.

[11] A. Imiya, K. Kawamoto. Random sampling and voting method for three-dimensional reconstruction. In *International Workshop RobVis 2001, Proceedings*, volume 1998 of *Lecture Notes in Computer Science*, pages 193-200. Springer, 2001.

[12] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87-116, 1988.

[13] G. Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321-345, 1984.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. Numerical recipes in C, second edition. Cambridge University Press, 1993.