Segmentation-Based Depth Propagation in Videos*

Nicole Brosch, Christoph Rhemann and Margrit Gelautz

Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria
{*nicole.brosch,rhemann,gelautz*}*@ims.tuwien.ac.at*

Abstract
In this paper we propose a simple yet effective approach to convert existing 2D video content into 3D. We present a new semi-automatic algorithm that propagates sparse user-provided depth information over the whole monocular video sequence. The advantage of our algorithm over previous work comes from the use of spatio-temporal video segmentation techniques. The segmentation preserves depth discontinuities, which is challenging for previous approaches. A subsequent refinement step enables smooth depth changes over time and yields the final depth map. Quantitative evaluations show that the proposed algorithm is able to produce good quality and temporal-coherent 3D videos.

1. Introduction

With 3D displays being commercially available the need for 3D media increases. Consequently, generating footage for 3D displays is of academic and commercial interest. A 3D display works by rendering different viewpoints, which can be generated by recording the scene by multiple cameras or a single stereoscopic camera. Another possibility is to synthesize the different views from a single viewpoint according to the depth of the scene. A special sensor, e.g. a time-of-flight sensor, can record the required depth information. In addition to the necessary equipment, the main disadvantage of these methods is that the need for 3D content has to be known before capturing the video. Hence, existing monocular videos can not be processed by these methods. This is, on the contrary, possible with semi- or fully automatic 2D to 3D conversion techniques, including [4, 7, 9, 10, 11] or ours.

Semi-automatic conversion techniques, like the proposed algorithm, incorporate user input [4, 6, 9, 10]. With such techniques, the user defines depth values for pre-segmented objects [10], scribbles [4] or every pixel [6, 9] in keyframes. This information is then propagated to all frames of the video. Even though the integration of user input allows flexibility in choosing the video material, arbitrary scenes still pose several challenges, including the preservation of object outlines and temporal-coherence.

This paper presents a semi-automatic conversion approach which addresses the problems mentioned before by using segmentation techniques. Segmentation refers to partitioning of scenes into regions, which are homogenous in a certain feature space (e.g. color). Likewise, depth propagation attempts to assign similar depth values to pixels which are similar in terms of color and position [4, 6, 9, 10]. Therefore, we propagate depth values given in the form of scribbles with a region-growing-based segmentation process that assigns spatio-temporal neighboring pixels which are similar in color to the

*This work was supported by the Doctoral College on Computation Perception at TU Vienna and the Austrian Science Fund (FWF) under project P19797. Christoph Rhemann received funding from the Vienna Science and Technology Fund (WWTF) under project ICT08-019.

same depth value. This way depth is propagated on the entire video. To interpolate depth variations over time and to refine our depth map, we employ an edge- and motion-preserving smoothing scheme.

Similar to our approach, Guttmann et al. [4] use user scribbles in the first and last frame. They propagate depth values by solving a linear system of equations. Its main drawback is that their quadratic smoothness term tends to over-smooth depth discontinuities, which leads to visible artifacts. In contrast, our approach preserves depth discontinuities by taking segmentation into account.

Additionally, approaches relying on image segmentation were introduced [6, 10]. In Wu et al.'s approach [10] the user assists the algorithm in segmenting objects in keyframes and assigning their depths. Then the objects are tracked from frame to frame. Depths in non-keyframes are assigned according to the depths of keyframes. Li et al. [6] attempt to improve this approach by using a more robust tracking algorithm. However, the user has to annotate objects in a large number of keyframes to obtain satisfactory results. This labor-intensive procedure is even more time consuming for small objects or fine details. Our conversion technique differs from these methods. We make direct use of a video segmentation algorithm to obtain spatio-temporal segments. This approach is, to the best of our knowledge, new for depth propagation. In contrast to [6, 10], our approach requires less annotated keyframes. Moreover, by applying a local filter, we preserve thin structures.

Varekamp and Barenburg [9] propagate depth values by bilateral filtering. As the authors point out, the quality of their depth videos decreases with the distance from the keyframe. Therefore, users again have to provide depth information for a large number of frames.

In addition to semi-automatic approaches, fully automatic methods have been proposed. Facing a task which generally is not invertible, namely recovering scene geometry from one position in the image plane, fully automatic methods have to include additional information, such as motion (e.g. [7]). These methods assume that depth is proportional to the motion's magnitude. This is, however, only true for static scenes taken by a moving camera. Another approach is to synthesize the stereo case by selecting the second view for a frame from the same video (e.g. [11]). This is even more restrictive. Apart from the limitation to stationary scenes, at most horizontal camera motion is accepted.

The rest of the paper is organized as follows. Section 2. describes the proposed algorithm. In Section 3. it is evaluated with reference data obtained by a stereo camera. Section 4. concludes our discussion.

2. Algorithm Description

The proposed propagation approach consists of three steps. To begin, the user annotates the first and the last frame of a video shot with scribbles, which encode depth for the marked areas (see Figure 1 a)-b) and Section 2.1.). Then a joint segmentation and propagation algorithm identifies spatio-temporal regions in the video sequence and assigns depth information to each pixel (Section 2.2.). In a subsequent step we interpolate depth values over time and refine the depth map by applying an edge and motion-preserving smoothness scheme (Section 2.3.). Below, we discuss these steps in detail.

2.1. Disparity Scribbles

As suggested in [4], we propagate disparity values rather than depth values. While the latter refers to the absolute distance of real-world objects from the center of projection, disparity, a proportional quantity, expresses the relative depth. For each shot, the user draws color scribbles in the first and last frame. The scribble's hue encodes the disparities for the pixels it contains (see Figure 1 a)-b)).

Figure 1. Input and output example. The first frame *a)* and the last frame *b)* of the input video are annotated with scribbles, whose hues encode depth. The depth map *c)* for an intermediate frame, generated by our approach, encodes depth by gray values (white: foreground, black: background). Copyright of original video: Warner Bros.

2.2. Joint Segmentation and Propagation

To propagate the given disparity values to the remaining pixels of the video, we adopt the segmentation algorithm of Grundmann et al. [3] for our task. It produces temporal-coherent segmentations, with consistent boundaries, and can be applied to video shots, containing motion, partial occlusions and illumination changes. Another benefit of this algorithm, over e.g. global segmentation, is that it can be implemented efficiently, allowing to process long videos (>40 seconds) in reasonable time [3]. The segmentation algorithm comprises two steps. First a generalized version of a graph-based image segmentation algorithm [2] is applied. Then neighboring segments are merged according to their similarity [3]. Below, we give a review of this algorithm and adopt it for disparity propagation.

To begin, the given data is represented as a graph. Each pixel in the video sequence is considered as a vertex, which additionally stores a disparity value if it crosses a scribble. Vertices are connected to their spatial and temporal neighbors by edges e. Temporal edges either connect direct neighbors of a pixel in an adjacent frame or, if making use of optical flow information (e.g. [8]), the neighbors along the backward flow. In order to express the similarity of two connected pixels, each edge is associated with a weight e_{wp}, i.e. their normalized color difference [2]. In the following, the vertices are grouped into regions. Initially, each vertex is considered as region of its own. Subsequently, we traverse edges that connect two regions in ascending order of the edge weights. Following this fixed merging order, the regions connected by an edge e are merged if the *internal variations* [2] of both regions are larger than the edge weight e_{wp}. Thereby, the internal variation $Int(R)$ of a region R is defined as [2]:

$$Int(R) = \max_{e \in MST} e_{wp} + \frac{\tau}{|R|} \tag{1}$$

In the first term the maximal edge weight of the Minimum Spanning Tree (MST), which spans a region R, is used to express a region's internal differences. Consequently, variations inside a region are tolerated. The second term makes this expression dependent on the region size $|R|$. This has to be done in order to handle regions which consist only of one vertex and therefore no edges. Here, τ is a constant parameter which influences the precision of the segmentation result (larger τ produces larger regions, but less accurate results [3]). Finally, the number of regions is reduced by merging low-cost edges of regions containing less than 100 pixels (in ascending order of their weights).

We use the algorithm described above to simultaneously propagate disparities when merging two regions. Specifically, if a region with unknown disparity is merged with a region with known disparity, the known pixel disparity is propagated to the other region (i.e. assigned to its pixels). Secondly, merging two regions without disparity information yields a region without disparity information.

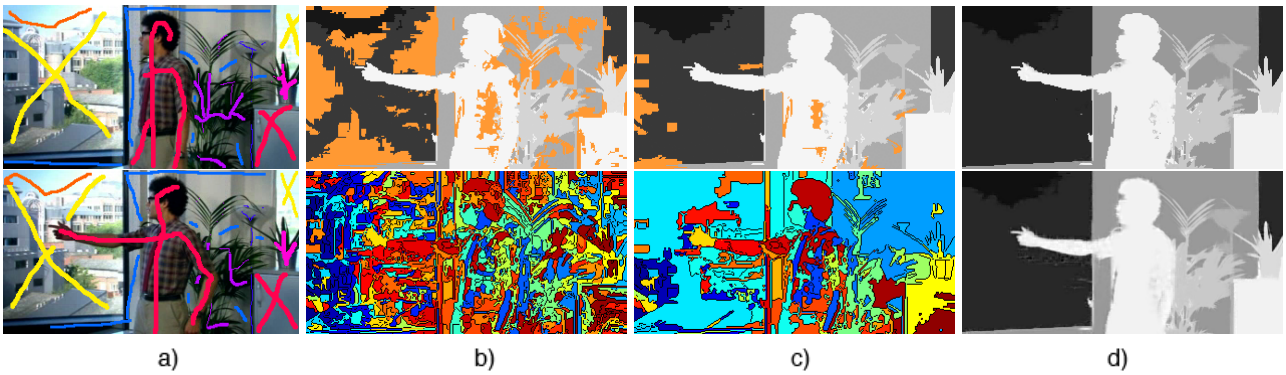

Figure 2. Segmentation and propagation. ***a)*** **Scribbles in the first and last frame.** ***b)*** **Pixel-wise over-segmentation of a middle frame (bottom) and propagation result (top). Orange pixels (top) have unknown disparities.** ***c)*** **Region-graph segmentation (bottom) and propagation result (top).** ***d)*** **Assignment of disparities to the missing regions and inter-segment smoothing (top). Refinement step (bottom). Copyright of original video: FremantleMedia Limited.**

Thirdly, if regions with conflicting disparities are merged, their respective disparities are preserved. As a result, spatio-temporal segments may contain vertices with different disparities. An example of this case is shown in Figure 2 c), where the blue segment (2nd row, top right) is split into subsegments (e.g. wall, lamp) in the disparity map (1st row). Note that this property is of particular importance for regions that represent slanted surfaces or change their disparity over time. It enables us to interpolate disparity variations within a region and obtain smooth disparity changes over time (Section 2.3.).

The result of the procedure described above is an over-segmentation of the video sequence, in which segments are assigned to disparity values (Figure 2 b)). Note that some segments are not assigned to a disparity value yet (Figure 2 b), 1st row, orange regions) and have to be merged applying our propagation rule set. To this end, we define a region-graph, in which each boundary pixel of a region, derived in the previous step, is a vertex. Neighboring boundary pixels (vertices) that belong to different regions are connected by edges e. Edges have two weights, the color similarity of the connected boundary pixels e_{wp} and a region edge weight e_{wr}. Region weights are derived from a region's normalized *LAB* histogram and if optical flow is used, its per frame-flow histograms (see [3] for details). In the latter case, the region weights e_{wr} are a combination of the χ^2 distances of both histograms (d_c distance of normalized *LAB* histograms, d_f distance of normalized per-frame-flow histograms) [3]:

$$e_{wr} = (1 - (1 - d_c)(1 - d_f))^2 \tag{2}$$

Hence, the resulting weights are close to zero for regions with similar motion and color properties and else close to one. The algorithm traverses edges in acceding order of their weights and merges neighboring regions, which may contain various disparities (e.g. Figure 2 c)). In order to ensure that the disparity of the most similar border pixel is propagated, edges are firstly sorted by their region weights e_{wr} and secondly by their pixel weights e_{wp}. We iteratively merge regions and jointly propagate disparities by applying the previously described merging process. As suggested in [3], in each iteration the minimum region size and τ are scaled by the factor 1.1. Finally, regions without disparity are assigned a value by iteratively merging low-cost edges (in ascending order of their weights). Alternatively, an interface for assigning or changing certain disparities can be implemented.

2.3. Disparity Interpolation and Refinement

Having applied our joint segmentation and propagation algorithm, every pixel in the video is associated with a disparity value. However, the current disparity map does not capture fine details like hair

Figure 3. Disparity propagation examples. ***a)*** **User scribbles in the first and last frame of a video shot.** ***b)*** **Example frames of obtained disparity map. Copyright of original video:** ***1st row:*** **Universal Pictures** ***3rd row:*** **Warner Bros.**

and contains abrupt temporal disparity changes. For instance, a region with a high disparity in the first and a low disparity in the last frame consists only of pixels with these two disparities. Moreover, the disparity defined in the first frame is dominant in the first frames and vice versa. To interpolate disparities over time we apply a spatio-temporal filtering on the video volume. In particular, we extend the recently proposed guided filter [5], which was originally developed to process images, to perform video filtering. It smoothes the input image, but preserves edges. Hence, it behaves similarly to a bilateral filter, but has runtime properties independent of the filter size. However, in order to perform edge- and motion-preserving smoothing of video content, we use three-dimensional instead of two-dimensional kernels. The extended filter is still locally applied and can process in linear time.

To interpolate disparity variations over time, we apply the guided video filter on each spatio-temporal segment independently, excluding the remaining pixels of the video sequence. Due to the edge preserving properties of the guided filter, we smooth the disparity map in regions that have similar colors in the input video and preserve disparities at edges in the input video. Choosing the diameter of the filter kernel as large as half of the number of frames contained in a particular segment, we are able to obtain smooth disparity changes over time. Note that in order to preserve edges at region boundaries, despite the potentially large kernel size, only disparities within such segments are smoothed.

In a second step, we refine the disparity map, i.e. object outlines, by applying the guided video filter (smaller kernel, radius i.e. three pixels) to the entire disparity map. The disparity map is filtered under the guidance of the input video. This means that the disparities are smoothed, while preserving edges caused by spatial and temporal changes in the input video. Textured surfaces of constant disparity keep their disparities, while homogenous surfaces with different disparities are smoothed. Furthermore, the guided filter is sensible to fine image structures. Hence, it can reveal fine details in the disparity map that have not been captured before (Figure 2 d)).

3. Experimental Results

We applied our propagation algorithm to a variety of video shots, including sport scenes, shots from broadcast videos and shots filmed with a conventional camcorder. We obtained temporal-coherent disparity maps, which reflect disparity changes due to object's motion (Figure 3, 1st row). As shown in Figure 3, our results adapt well to the corresponding scenes/objects. They contain homogenous

MSE	with OF	without OF	method [4]
Palace	1.01	1.39	8.51
Parade	0.23	0.23	6.87
City	0.32	0.56	10.30
Soccer	0.23	0.33	4.95
Stairs	0.19	0.31	1.56

Table 1. Quantitative evaluation of our algorithm with and without optical flow (OF) edges (left) in comparison to our implementation of Guttmann et al.'s method [4] (right). The table lists the mean squared error (MSE) (i.e. averaged over all frames) of the disparities. The given numbers are scaled by 100.

regions with hard disparity edges as well as plausible disparities in slanted surfaces. We are able to capture fine structures and small objects (Figure 3, 2nd row). We see that our algorithm deals with partial occlusions (Figure 3, 3rd row) and motion (Figure 3, 2nd row). In case of full occlusions, our algorithm uses neighboring regions to guess the disparity. However, a single occlusion of an object with constant disparity in the entire video sequence still leads to plausible results, assuming the respective reference data in the first and last frame are available. We observed limitations in cases when the segmentation algorithm fails and noticed halos near some edges (disadvantage of [5]).

We quantitatively compare our algorithm to Guttmann et al.'s semi-automatic stereo extraction method [4] on five different test sequences, recorded by a stereoscopic camera. To obtain reference solutions for these scenes, we apply a state-of-the art stereo method [1], which derives a disparity map for each sequence. (Note, that we only use one of the views for evaluating our algorithm.) The recorded videos introduce complexities such as object and camera movement as well as partial occlusions, shadows and slanted surfaces. Similar to the regular user input, we draw scribbles in the first and the last frame of the video shots. Again the scribbles define which disparity values should be propagated. To make a comparison with the reference solution possible, the disparity values for the marked pixels are defined as the disparities of the reference data at the scribble positions. Table 1 lists the mean squared error (MSE) averaged over all frames of a video sequence for our propagation algorithm and our implementation of Guttmann et al.'s method [4]. We evaluate our method in two versions, with and without making use of optical flow edges. When employing optical flow edges, we use the same flow fields as [4], i.e. dense flow fields obtained by Ogale and Aloimonos' estimation method [8]. Therefore the comparison is independent of the flow fields's quality. Figure 4 exemplarily shows the results for two video sequences of our dataset (City (top), Stairs (bottom)). It can be seen (Figure 4, Table 1) that our propagation algorithm produces high-quality disparity maps, which are near to the reference data. More importantly, we outperform the previous work by Guttmann et al. [4] using the same user input (Figure 4, Table 1). Our algorithm adapts better to the underlying scenes and generates disparity maps in which disparity discontinuities are preserved. As an additional advantage over [4], we are able to spare optical flow fields and still obtain similarly good results (Table 1). However, when we examine the respective differences in detail we noticed limitations at region borders of moving objects (Figure 4 b)-d)) and in cases when the segmentation algorithm encounters problems, such as objects similar in color. Considering the former limitation, guided video filtering quantitatively and visibly reduces the error at motion discontinuities. Figure 4 b)-d) further illustrates that our approach is able to reflect disparity changes due to object's motion in its propagation result. Note that the difference in the error images results from the different velocities of the disparity changes in the reference data and our disparity maps.

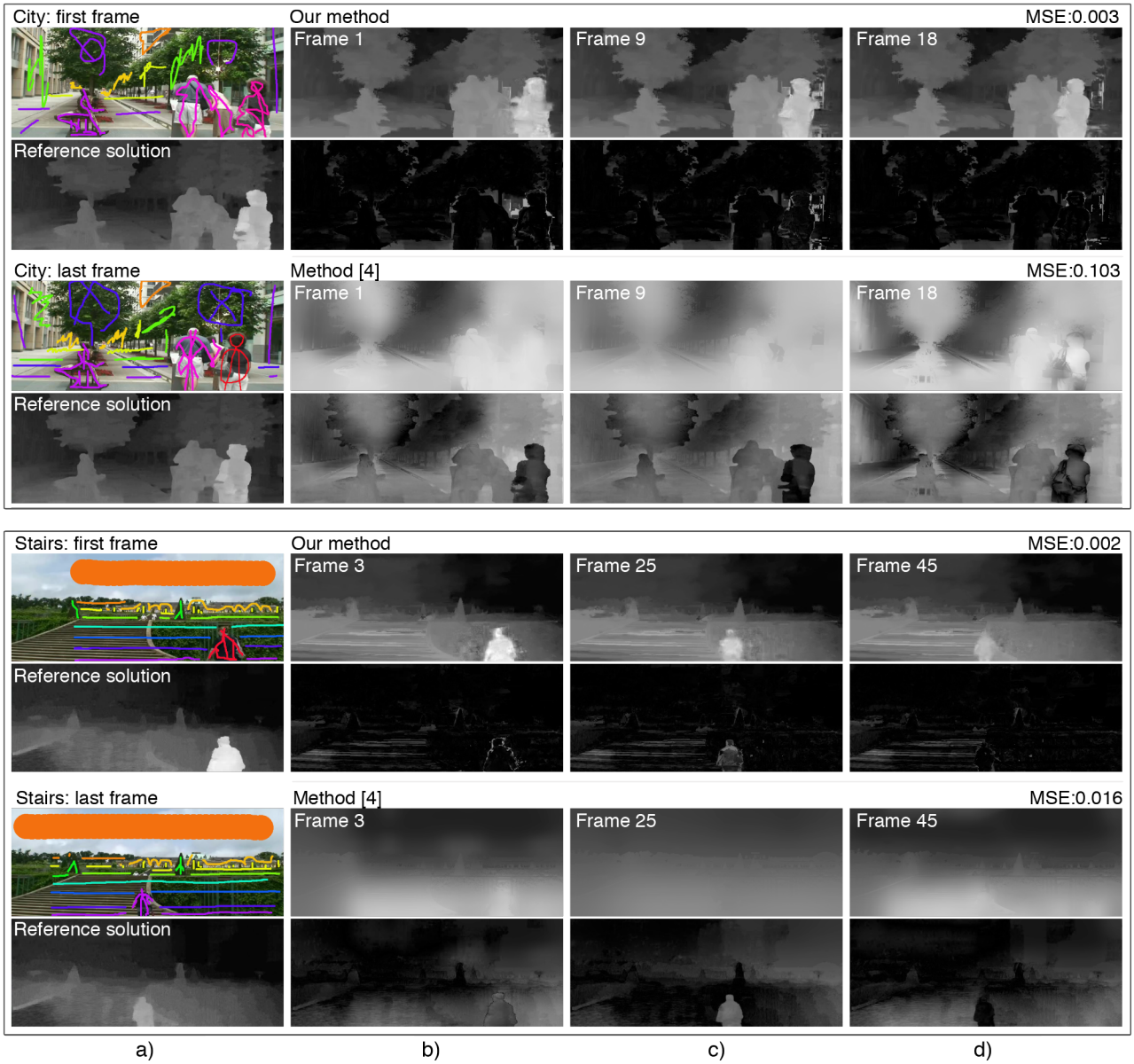

Figure 4. Evaluation results. Per sequence: *a)* Scribbles and reference data of first and last frame. *b)-d)* Our propagation result (1st row) and corresponding error (2nd row). In the latter case, dark and white pixels denote low and high errors, respectively. Disparities (3rd row) obtained by our implementation of [4] and corresponding error (4th row). Our result: hard edges, homogenous regions. Result [4]: over-smoothed.

4. Conclusion

We proposed an approach to propagate disparities in dynamic video shots, which outperforms a previous method over a set of real-world videos. Our contribution was to adopt a robust segmentation algorithm for disparity propagation, which enables us to process video shots containing camera and object motion. A subsequent step interpolates disparity values over time and refines the disparity map, enforcing disparity edges to be consistent with the input video. As our propagation algorithm relies on video segmentation, it has to cope with its inherent challenges including occlusions and similar appearances. Future work could concentrate on tackling these challenges.

References

[1] M. Bleyer and M. Gelautz. Simple but effective tree structures for dynamic programming-based stereo matching. In *VISAPP '08: International Conference on Computer Vision Theory and Applications*, pages 415–422, 2008.

[2] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *IJCV'04: International Journal of Computer Vision*, 59:167–181, 2004.

[3] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR '10: Conference on Computer Vision and Pattern Recognition*, pages 1–14, 2010.

[4] M. Guttmann, L. Wolf, and D. Cohen-Or. Semi-automatic stereo extraction from video footage. In *ICCV '09: International Conference on Computer Vision*, pages 136–142, 2009.

[5] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV '10: European Conference on Computer Vision*, pages 1–14, 2010.

[6] Z. Li, X. Xie, and X.D. Liu. An efficient 2d to 3d video conversion method based on skeleton line tracking. In *3DTV-CON '09: Conference on The True Vision - Capture, Transmission and Display of 3D Video*, pages 1–4, 2009.

[7] K. Moustakas, D. Tzovaras, and M.G. Strintzis. Stereoscopic video generation based on efficient layered structure and motion estimation from a monoscopic image sequence. *IEEE Transactions on Circuits and Systems for Video Technology*, 15:1065–1073, 2005.

[8] A.S. Ogale and Y. Aloimonos. A roadmap to the integration of early visual modules. *IJCV'07: International Journal of Computer Vision*, 72:9–25, 2007.

[9] C. Varekamp and B. Barenbrug. Improved depth propagation for 2d to 3d video conversion using key-frames. In *IETCVMP '07: European Conference on Visual Media Production*, pages 1–7, 2007.

[10] C. Wu, G. Er, X. Xie, T. Li, X. Cao, and Q. Dai. A novel method for semi-automatic 2d to 3d video conversion. In *3DTV-CON '08: Proceedings of the 2nd Conference on The True Vision - Capture, Transmission and Display of 3D Video*, pages 65–68, 2008.

[11] G. Zhang, W. Hua, X. Qin, TT. Wong, and H. Bao. Stereoscopic video synthesis from a monocular video. *IEEE Transactions on Visualization and Computer Graphics*, 13:686–696, 2007.