Motion Segmentation in Long Image Sequences

Steven Mills and Kevin Novins
Department of Computer Science
University of Otago
PO Box 56, Dunedin, New Zealand
{mills_sj,novins}@cs.otago.ac.nz

Abstract

Long image sequences provide a wealth of information, which means that a compact representation is needed to efficiently process them. In this paper a novel representation for motion segmentation in long image sequences is presented. This representation – the feature interval graph – measures the pairwise rigidity of features in the scene. The feature interval graph is recursively computed, making it a compact representation, and uses an interval model of uncertainty. The feature interval graph forms the basis for new algorithms for motion segmentation and occlusion analysis. Results of these algorithms are presented on synthetic and laboratory scenes.

1 Introduction

The analysis of long image sequences is a growing area of research in computer vision. Long sequence analysis is important for visual surveillance, mobile robotics, and other areas where a dynamic scene is observed over a long period of time. Long image sequences provide a wealth of information, but this raises the problem of efficient representation. It is not feasible to store an entire sequence, and so a compact representation is needed which can be efficiently computed, and does not grow with the length of the sequence.

In this paper the particular problem of motion segmentation in long image sequences is addressed. In motion segmentation the goal is to cluster the scene, or features extracted from it, into regions having a common motion. The clusters correspond to independently moving objects in the scene and so are useful for tracking and navigation.

Section 2 presents some previous approaches to the task of motion segmentation in long image sequences. These approaches make use of the Kalman filter [2], which provides a compact and robust representation of information gathered from the image sequence. The Kalman filter, however, relies on a Gaussian model of uncertainty, and assumes that the variances of these distributions are known. In Section 3 a new representation for motion segmentation is presented. This representation, the feature interval graph, measures pairwise rigidity information directly and shares many of the advantages of the Kalman filter, but requires less a priori knowledge.

The feature interval graph is then used to develop new algorithms for motion segmentation and the analysis of partial occlusions in the scene in Sections 4 and 5. An analysis of these algorithms is given in Section 6, followed by some concluding remarks.

BMVC 2000 doi:10.5244/C.14.17

2 Previous Work

Much of the previous work in long sequence analysis makes use of the Kalman filter [2]. The Kalman filter is used to combine information from a series of observations and is based on a pair of model equations

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{u}_{t-1}, \text{and} \tag{1}$$

$$\mathbf{y}_t = M\,\mathbf{x}_t + \mathbf{w}_t, \tag{2}$$

where $\mathbf{x}_t$ is the state of the system at time t, $\mathbf{y}_t$ is a measurement made to determine the state at time t, and $\mathbf{u}_{t-1}$ and $\mathbf{w}_t$ are independent Gaussian noise vectors with known covariance matrices.

The Kalman filter makes an iterative estimate of the state, $\hat{\mathbf{x}}_t$, and its covariance matrix, P, using a recursive estimation procedure. The estimate at each time is based on the previous estimate and the current measurement. The Kalman filter is an attractive model for long sequence analysis for a number of reasons. The Kalman filter is efficient, since all of the computations are linear, and only use information from the current and previous frames. Since the filter is a recursive procedure, historical information does not need to be stored. The Kalman filter is also robust with respect to measurement errors, which are modelled using covariance matrices.

Zhang and Faugeras [11] use the Kalman filter to track line segments through an image sequence. Three-dimensional line features are extracted from a scene using stereo, and an extended Kalman filter is used to estimate their kinematic parameters – rotational velocity, and translational velocity and acceleration. An 'extended' Kalman filter is used since the relationship between a feature's location and kinematics is non-linear. The Kalman filter is extended by adding a linearisation stage, which approximates this relationship by a linear one, and then applying a standard Kalman filter to the approximation. If a feature is not tracked from one frame to the next, then its location is estimated using its estimated kinematics. Occluded features tracked like any other until they reappear or are not observed for a long period of time, at which stage they are deactivated.

Once the kinematic parameters of a set of features have been determined, they are grouped into objects. If two features lie on the same rigid object, then their kinematic parameters should be the same. To account for uncertainty, the Mahalanobis distance, which may be thought of as a Euclidean distance weighted by uncertainty [1, page 75], is used to compare features. If the distance between two features' parameters is small then they are determined to lie on the same rigid object.

Another approach to motion segmentation in long image sequences is ASSET-2 [10]. Rather than three-dimensional line features, ASSET-2 uses two-dimensional point features. These have the advantage that they may be found in a wide variety of images and need only a monocular image sequence. Features are tracked through the sequence, and a Kalman filter is used to estimate their two-dimensional motion parameters. This information is then used to cluster the features into rigid objects.

The motion of each feature over the last N frames (typically 10) is used to reduce the effects of noise. This motion, $\vec{u}$ is compared to that predicted by a motion model, $\vec{u}_m$, by computing a normalised distance

$$D = \frac{|\vec{u} - \vec{u}_m|}{\frac{|\vec{u}|+|\vec{u}_m|}{2} + \sigma}, \tag{3}$$

where σ is used to reduce the effects of noise when matching small vectors. The motion model is initially that of constant motion, which can be initialised with a single vector, but is replaced by an affine model once sufficient information is available. A feature is added to a cluster if the distance defined by Equation 3 is less than some threshold.

The clusters are then tracked over time, again using a Kalman filter to estimate motion parameters. Occlusion in the scene is analysed by the motion of these clusters, rather than individual features. Occlusion is detected when two clusters overlap in the image, or when the number of features in the cluster decreases rapidly. The shape of the occluded cluster is frozen, and its motion is estimated from the visible portion. If a cluster disappears entirely then the motion parameters are also frozen, and the location of the cluster is predicted by extrapolation.

3 The Feature Interval Graph

In this section we present a novel representation for long sequence segmentation. This representation, the feature interval graph, shares many properties with the Kalman filter. The feature interval graph is recursively computed, relying on only the current and previous frames; is efficient to compute; and is robust with respect to measurement uncertainty.

The feature interval graph is constructed from three-dimensional features identified in each frame. Such features may be located and tracked using established techniques in feature detection, stereo, and motion correspondence. For the examples presented here, the Harris feature detector [6], and stereo and motion correspondences were established using bipartite graph matching techniques [3, 4]. Measurement uncertainty is accounted for by using interval ranges rather than precise values to represent features' locations. Uncertainty is propagated through subsequent computations using interval arithmetic [8].

The feature interval graph is initialised with the first observation. A vertex is added to the graph for each feature in the scene, and an edge links each pair of distinct features. Associated with each edge is a measurement of the three-dimensional distance between the features, computed using interval arithmetic. Each subsequent frame of the sequence gives another set of observations and distance measurements. Over time, as more measurements are made, the graph evolves and compactly represents the information that has been gathered over the sequence. There are two main tasks to be made – to combine information from multiple observations, and to account for missing observations.

At each frame a new measurement may be made for the distance between each pair of points. These distances are represented as intervals, and are combined using an intersection operator. The new distance stored in the graph is the intersection of the old distance and the latest observation. This represents the set of distances which are consistent with all of the measured distances. If this intersection is ever empty then the distance between the two features must have changed, and so the edge is removed from the graph.

The use of intervals to represent the locations of features and the distances between them has the effect of making the edges of the graph slightly 'elastic'. A small amount of motion between features is allowed, and is considered the effect of measurement errors, but motion which is too large to be accounted for by the estimated error bounds causes the edge to be removed from the graph.

The second issue that needs to be accounted for is that of points which appear, disappear, and reappear in the scene. The problem of distinguishing reappearance from

appearance is deferred until Section 5, but making this distinction is important so that information in the graph may be retained across periods of occlusion. For now, we will consider the effects of the appearance, disappearance, and reappearance on the feature interval graph.

When a new feature appears in the scene, a new vertex is added to the graph. This new vertex is linked to all other visible features' vertices and the edge distances are initialised with the current distance measurement. When a feature disappears the portion of the graph relating to it is frozen. The location of the feature is estimated, as discussed in Section 5, but the edges incident on the corresponding vertex are not changed. Features which disappear are not immediately removed from the graph, since they may reappear. Finally, a previously occluded feature may reappear. In this case information gathered prior to the occlusion is combined with the latest observation. Edges observed only before the occlusion retain their distance information, and those observed only after the occlusion use the new information. Finally if an edge is observed both before and after the occlusion the intersection of the old and new distance estimates is taken. In any case, edges which have been removed prior to the occlusion are never restored.

An example of the feature interval graph construction from a synthetic sequence is shown in Figure 1. A synthetic scene is used since it produces clearer results to illustrate the process. Examples on laboratory sequences will be shown in the following sections.

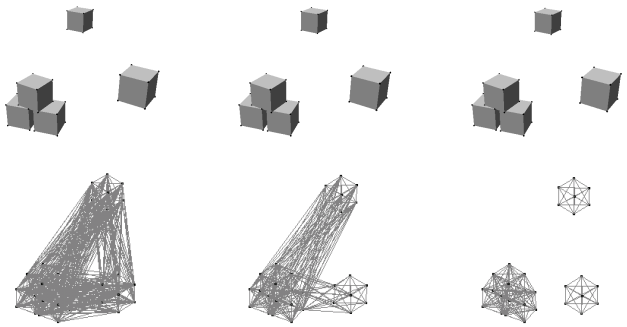

Figure 1: Frames from a synthetic scene (top) and the feature interval graphs (bottom) after (from left to right) 1, 2, and 3 frames of motion. As more frames are observed edges linking points on different objects are removed, while those within an object remain.

4 Segmenting the Graph

If all of the edges linking separate objects were removed from the feature interval graph, then the objects could be found as connected components of the graph. The use of components, however, is very sensitive to extraneous edges in the graph. Such edges are often present since the graph is initially assumed to be complete, and strong evidence is needed to remove an edge.

Conversely if none of the edges within objects are removed, then cliques in the graph – sets of vertices, each connected to all the others – could be used to find objects. The use of cliques has the opposite problem to a component-based segmentation in that it is sensitive to missing edges. Of more concern is the computational cost of a clique-based segmentation. The cliques used to define objects in the scene should be as large as possible, and the problem of finding the largest clique in a graph is NP-complete [5].

To overcome the problems associated with simple segmentation schemes we propose an object definition based on triangles in the feature interval graph. Triangles, or 3-cycles, in a graph are sets of three vertices, each connected to both of the others. Triangles correspond to small rigid substructures in the graph, and lead to a segmentation method which is robust with respect to a small number of missing or spurious edges in the graph, and which can be efficiently computed.

To construct a triangle-based segmentation an auxiliary graph, the triangle graph, is constructed from the feature interval graph. The vertices of the triangle graph correspond to triangles in the feature interval graph, and two vertices are linked by an edge if the corresponding triangles share an edge (or equivalently two vertices) in the feature interval graph. A simple component analysis is then applied to segment the triangle graph, and the segmentation is transferred back to the vertices of the feature interval graph.

Figure 2 (a) shows a graph which contains a number of problematic features. There is a missing edge, E, a spurious edge, F, and a cut vertex, V. The graph has a single component but intuitively seems to contain three 'objects'. Figure 2 (b) shows the triangle graph constructed from Figure 2 (a). The components of this graph correspond to the desired object segmentation of the original graph.

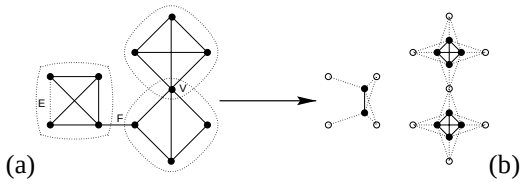

Figure 2: The computation of the triangle graph. The feature interval graph (left) appears to have contain three objects, indicated by the dashed lines. The segmentation is complicated by the missing edge, E,the spurious edge, F, and the cut vertex, V. The triangle-based segmentation (right) has vertices (filled circles) constructed from connected triples of vertices in the feature interval graph (hollow circles). The components of the triangle graph correspond to the desired segmentation.

Once the triangles in the graph have been labelled, the segmentation is transferred back to the vertices of the feature interval graph. In most cases this is a straightforward process, but some vertices in the feature interval graph, such as the vertex, V in Figure 2, contribute to triangles on two or more objects. Such vertices' classification is ambiguous, and so they are labelled as such.

Problem arises when new points enter the scene. New features are connected to all other visible features, and so can disrupt the segmentation. The edges associates with the new features, however, have less significance than those which have been observed for many frames, and so have undergone greater scrutiny. In order to overcome, a temporal

extension to the segmentation scheme outlined above is proposed. Each triangle is assigned an age, being the minimum age of its component vertices. The oldest triangles are segmented first, and the remaining are added in order of decreasing age. If a new triangle disrupts the segmentation of the older ones by merging two or more clusters then it is disregarded in favour of the older, more trustworthy, information.

Figure 3 shows the results of the triangle based segmentation applied to the scene from Figure 1. After 3 frames the objects have become distinct and are identified.

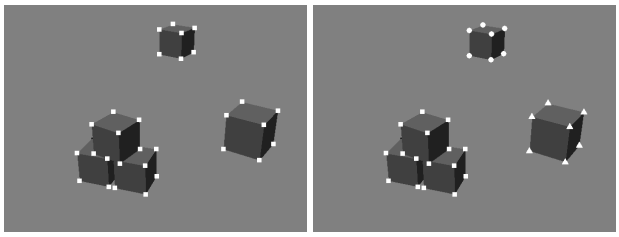

Figure 3: The object segmentation after 2 (left) and 3 (right) frames of motion in the scene from Figure 1. The segmented features have been projected back onto the image, and different shaped symbols represent different objects.

A more complicated case of motion segmentation is shown in Figure 4. This example is a laboratory sequence filmed using stop-motion techniques. There are two objects in the scene, composed of plastic blocks, and one moves across from the left. After 3 frames of motion the feature interval graph appears confused, due to features appearing in the scene, but the temporal analysis allows the triangle-based segmentation to identify the two objects.

5 Occlusion Analysis

The final problem remaining is reasoning about occlusion. There are two main tasks to be solved – estimating the location and detecting the reappearance of occluded features. A good solution to the first problem will make the second problem much easier. If the location of occluded features can be reliably estimated, then reappearance can be detected when a ‘new’ feature appears where an occluded one is expected to lie.

Previous approaches to occlusion reasoning have either been feature- or object-based. In feature-based approaches, such as the work of Zhang and Faugeras [11], the motion parameters of individual features are estimated, and used to predict their locations should they disappear. This approach relies on the motion parameters being constant, which generally holds only for short periods of occlusion. This is taken into account by Zhang and Faugeras by deactivating features which are occluded for long periods of time.

An alternative approach is to track higher-level objects rather than features. ASSET-2 [10], for example, tracks clusters of features which are partially occluded using the motion of the visible portion, and the invisible portion is assumed to follow the same motion. ASSET-2’s approach allows for the motion of partially occluded objects to change, but does not directly estimate the locations of individual occluded features, and this estimation is needed to detect reappearance.

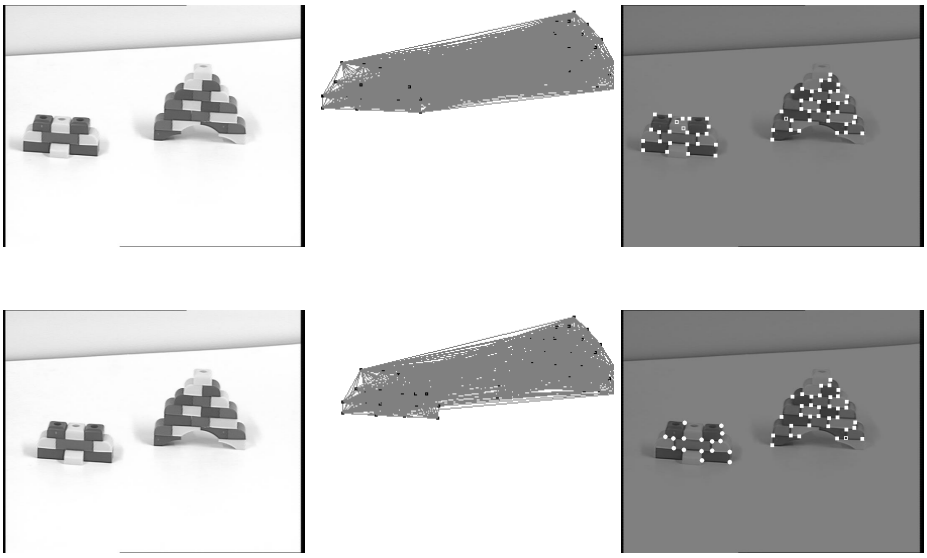

Figure 4: Frames (left), feature interval graphs (center) and motion segmentations (right) for a laboratory scene after 2 (top) and 3 (bottom) frames of motion. Note that the feature interval graph remains strongly connected due to new features appearing in the scene, but the temporal analysis identifies the two objects.

We propose a hybrid approach for the analysis of partial occlusion. The rigid-body transform (rotation and translation) of each object is computed using a least-squares technique [7], and then this transform is applied to occluded vertices associated with that object. This means that information available from the visible portion is used, as in ASSET-2, and the locations of individual features are predicted, as in Zhang and Faugeras' work.

To apply this technique, it is necessary to associate occluded features with objects in the scene. This may be achieved using the triangle-based segmentation strategy from the last section. Triangles containing occluded features are taken to have an age of 0, and so are grouped last. This means that information based on occluded features, which has no direct evidence, is given the least weight in the temporal segmentation.

Once the locations of the occluded points are predicted, they are compared with features which have just appeared in the scene. If a new feature appears where the occluded feature is predicted to lie then they are considered the same, and are merged as described in Section 3. Since features' locations are represented as intervals, two features are in the same place if the interval bounds on their locations intersect.

Figure 5, shows an example of the occlusion analysis in the example from Figure 4. The region shown in the lower left corner of the rear object, which is occluded when the other object moves in front of it.

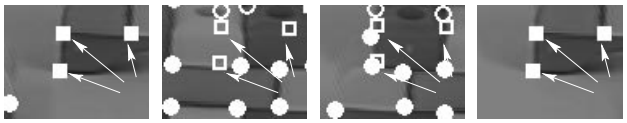

Figure 5: Detail of some occluded features in the image sequence from Figure 4. The three features indicated by arrows are occluded by an object which passes in front of them. Their locations are updated over time, shown as hollow symbols, and they are recognized when they reappear, allowing them to be correctly classified (right).

6 Analysis

Algorithms for long image sequence analysis need to be efficient in terms of both computational cost and storage requirements. The algorithms presented in the previous sections satisfy these requirements. The feature interval graph update requires that every vertex and every edge in the graph needs to be updated, and so requires $O(m+n)$ computations, where m is the number of edges in the graph, and n is the number of vertices, which is also the number of features in the scene. In the worst case $m \approx n^2$ so the graph update procedure is $O(n^2)$. Similarly, information must be stored for each vertex and edge in the graph, and so the storage requirements for the feature interval graph are also $O(n^2)$.

The object segmentation is a little more expensive. The feature interval graph may have up to $O(n^3)$ triangles, and so the triangle graph has $O(n^3)$ vertices. The triangle graph is, however, a sparse graph – each triangle has at most n adjacent triangles since adjacent triangles differ by at most one vertex in the feature interval graph. This means that the triangle graph has at most $O(n^4)$ triangles. Finding the components of a graph with v vertices and e edges requires $O(v+e)$ time [9, pages 426,437–438]. This means that the time required to identify objects in a scene with n features is $O(n^3+n^4)$, or $O(n^4)$, in the worst case. The space requirements for the triangle segmentation are $O(n^3)$ to store the triangles, and the edges of the triangle graph may be generated as needed.

Figures 6 and 7 show the time and space required to process the examples used in Sections 3 to 5. These timings are based on a C++ implementation of the algorithms running on a 200 MHz Pentium-based machine under the Linux operating system. These figures show that the computational time and cost depends primarily on the number of features in the scene, and does not increase as more frames are observed.

The computation times indicate that for real time processing is possible for scenes with up to 100 features or so, with moderate improvements in processor speed. Off-line processing is expected to be feasible for scenes with several times this number of features, but the $O(n^4)$ nature of the segmentation algorithm means that processing time increases rapidly past this point. The storage requirements are modest, about 200 Kb in total for the laboratory scene. By comparison, a stereo pair of PAL format colour images, such as those used in this sequence, requires 2,340 Kb to store.

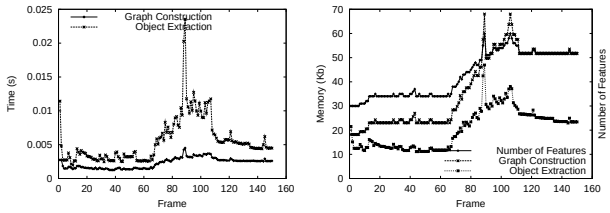

Figure 6: Computational time in seconds (left) and storage requirements in kilobytes (right) for a long synthetic sequence. The sharp rise in cost around Frames 88 and 89 is due to the introduction of several new features, with a corresponding increase in the number of vertices in the graphs.

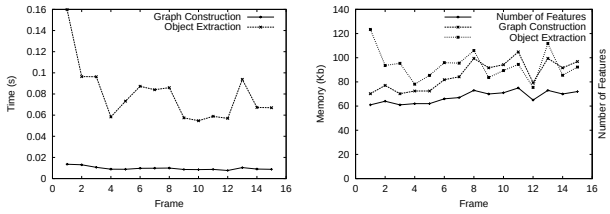

Figure 7: Computational time in seconds (left) and storage requirements in kilobytes (right) for the laboratory sequence.

7 Conclusion

In this paper a novel representation for motion segmentation in long sequences has been presented. This representation, the feature interval graph, uses an interval model of uncertainty and measures pairwise rigidity between features in the scene. Like the Kalman filter, the feature interval graph is recursively computed so that a complete history of the scene does not need to be stored, takes uncertainty and measurement errors into account, and combines a sequence of uncertain measurements over time.

The feature interval graph differs from the Kalman filter in two main respects: an interval, rather than statistical, method of reasoning about uncertainty is used and no model of the motion is needed. The Kalman filter uses a Gaussian model of uncertainty and a linear motion model. These both introduce parameters to the system which need to be known, or estimated.

Using the feature interval graph representation algorithms for motion segmentation and partial occlusion analysis have been developed. The motion segmentation algorithm is based on finding small rigid substructures in the graph. This approach means that a small number of errors in the graph can be overcome, and a temporal extension makes the segmentation stable over time. The occlusion analysis is a hybrid of concepts from two earlier approaches. The motion of the visible portion of a partially occludes object is used to estimate its rigid body motion parameters, and these are used to predict the location of occluded features.

The feature interval graph construction is an efficient process. Since the feature in-

terval graph is recursively defined, it is not necessary to store a complete history of the scene. The fact that edges removed from the graph cannot be restored, however, means that key information about the pairwise rigidity of features is not forgotten. The object segmentation method is more computationally expensive, being $O(n^4)$ for a scene with n features, but is still practical for scenes with a modest number of features.

The feature interval graph can take several frames to converge. This is not a problem from a computational viewpoint since in long sequences it is possible to wait for more information before making a decision. Practically, however, this may be of concern. The convergence of the feature interval graph could be accelerated by applying heuristics to remove edges from the graph, or to reduce features' initial connectivity. For example, each vertex in the graph could be connected only to its nearest m neighbours. This would also decrease the computational cost, since for a scene with n features the graph would have only $O(mn)$ edges and up to $O(m^n)$ triangles in the graph, each having at most m neighbours. The triangle graph would, therefore, have $O(m^n)$ vertices and $O(m^3 n)$ edges. Since m is a constant, this approach would yield linear computational costs for both the graph construction and object segmentation processes.

References

[1] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley and Sons, New York, NY, second edition, 1984.

[2] S. M. Bozic. *Digital and Kalman Filtering*. Edward Arnold, London, 1979.

[3] Y.-Q. Cheng, V. Wu, R. T. Collins, and E. M. Riesman. Maximum-weight bipartite matching technique and its application in image feature matching. In *SPIE Conference on Visual Communication and Image Processing*, pages 463–462, 1996.

[4] G. Fielding and M. Kam. Applying the Hungarian method to stereo matching. In *IEEE Conference on Decision and Control*, pages 549–558, 1997.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Fransisco, CA, 1979.

[6] C. Harris and M.J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–152, 1988.

[7] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America, A*, 4(4):629–642, 1987.

[8] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[9] R. Sedgewick. *Algorithms*. Addison-Wesley, Reading, MA, second edition, 1988.

[10] S. M. Smith and J. M. Brady. ASSET-2: Real-time motion segmentation and shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):814–820, 1995.

[11] Z. Zhang and O. D. Faugeras. Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *International Journal of Computer Vision*, 7(3):211–241, 1992.