Image and Vision Computing 29 (2011) 190–205

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Video-object segmentation and 3D-trajectory estimation for monocular video sequences

Feng Xu [a,b], Kin-Man Lam [a,*], Qionghai Dai [b]

[a] Centre for Signal Processing, Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong
[b] TNList and Department of Automation, Tsinghua, University, Beijing 100086, China

ARTICLE INFO

Article history:
Received 19 February 2010
Received in revised form 30 July 2010
Accepted 2 September 2010

Keywords:
2D-to-3D video conversion
3D trajectory estimation
Video-object segmentation

ABSTRACT

In this paper, we describe a video-object segmentation and 3D-trajectory estimation method for the analysis of dynamic scenes from a monocular uncalibrated view. Based on the color and motion information among video frames, our proposed method segments the scene, calibrates the camera, and calculates the 3D trajectories of moving objects. It can be employed for video-object segmentation, 2D-to-3D video conversion, video-object retrieval, etc. In our method, reliable 2D feature motions are established by comparing SIFT descriptors among successive frames, and image over-segmentation is achieved using a graph-based method. Then, the 2D motions and the segmentation result iteratively refine each other in a hierarchically structured framework to achieve video-object segmentation. Finally, the 3D trajectories of the segmented moving objects are estimated based on a local constant-velocity constraint, and are refined by a Hidden Markov Model (HMM)-based algorithm. Experiments show that the proposed framework can achieve a good performance in terms of both object segmentation and 3D-trajectory estimation.

1. Introduction

The proposed method jointly achieves video-object segmentation and 3D-trajectory estimation. Video-object segmentation, an important technology for MPEG-4, virtual reality, video surveillance and many other high-level applications, plays a vital role in computer vision. 3D-trajectory estimation from a monocular video is also a fundamental problem for many applications, such as behavior recognition, 3D-object reconstruction, video surveillance, etc. When these two tasks can be jointly achieved, both the appearance and the behavior of objects in a video are acquired, so more high-level applications can be established. For example, 3DTV has become an increasingly popular research topic, and has many applications in the advertising, entertainment, and 3D visualization markets, and so on. For 3DTV, one popular 3D content format is called 2D-plus-depth, which has a grey-scale depth map associated with each 2D frame. Ordinary 2D videos cannot be directly used in 3DTV because they have no depth information. In order to achieve 3D visualization with these 2D videos, we need to estimate the depth map for each frame. For this 2D-to-3D conversion application, segmentation is employed to identify the pixels present in moving objects, and 3D trajectories can provide useful information about the depth change of the objects. As the semi-automatic 2D-to-3D conversion methods proposed in [1,2] do not efficiently model depth changes in non-key frames, our method can be used to handle the problem directly. As our work is related to both video-object segmentation and 3D object reconstruction, we will review these state-of-the-art techniques in the following.

To perform segmentation, both spatial (color, texture, edge) and temporal (motion) information should be fully explored. As we know, segmentation in images is a much more difficult problem than in videos, as images contain spatial information only. The use of motion information in videos becomes a key issue in video segmentation. Many methods have been proposed for this problem. One popular method is to incorporate motion information into video-object segmentation by means of optical flow [3–8]. Although optical flow can provide a dense motion field, it has a limited ability to handle overlapped motion fields and large inter-frame motion. Another popular method is by means of tracking [9–13], which can use motion information to determine object positions and boundaries in non-key frames, based on the segmentation results in those key frames. Generally, this approach has two major drawbacks: first, the performance depends on the quality of key-frame segmentation, which is still a challenging task; and second, errors in segmentation will accumulate during the tracking procedure. In addition to these two approaches, there are many other methods: [14] treats videos as 3D signals and uses the Bayesian approach to achieve segmentation; [15] first transforms videos into the frequency domain and then performs segmentation. Using a similar concept, both [16,17] segment videos in the compressed domain.

3D reconstruction is also an important research topic in computer vision. Most of the work in the literature has focused on reconstructing one 3D scene from several images with different viewpoints

* Corresponding author. Tel.: +852 27666207.
E-mail addresses: xufeng2003@gmail.com (F. Xu), enkmlam@polyu.edu.hk (K.-M. Lam), qhdai@mail.tsinghua.edu.cn (Q. Dai).

0262-8856/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2010.09.001

(structure from motion, and shape from stereo) [18–21], different illuminations (photometric stereo) [22–24], or different focus planes (shape from focus) [25]. For 2D-to-3D video conversion, as the 3D property of a scene changes with time and is captured by one camera, 3D reconstruction has to be performed for every frame. As this 2D-to-3D conversion is an ill-posed problem, constraints on the motions of objects are imposed in many methods. Yuan and Medioni [26] used the assumption that objects are moving on a plane to reconstruct the scene and calibrate the camera. Avidan and Shashua [27] focused on the case where an object moves either on a line or conically. Han and Kanade [28] assumed that objects move on a line at a constant speed. Ozden et al. [29] have also used the planarity constraint and the heading constraint to find the relative scales among the reconstructed objects. These methods all utilize global motion assumptions, so they can handle only a certain kind of motion. In contrast, our proposed method employs a local constant-velocity constraint, which allows for linearly, and locally temporal motion and overall nonlinear motion. In other words, this local constraint can be satisfied by more general global motions; and especially in high frame-rate videos, the application scope can be wider. Experimental results will also demonstrate the performance of our proposed method when moving objects are under linear and nonlinear motions.

In our proposed approach, we use a feature-based method for both video-object segmentation and 3D-trajectory estimation. Our approach utilizes the correspondences in consecutive frames, determined by the Scale-invariant feature transform (SIFT) features [30], to estimate motion information. As we know, motion vector and optical flow are two popular ways to represent the motion information in a video. However, motion vector is developed for compression, not for obtaining accurate, and true motions, while optical flow cannot handle large occlusive regions and large disparity motions accurately. Consequently, feature-correspondence-based methods are adopted in our algorithm, which can produce accurate rather than dense motions.

Our method can be described as follows. The SIFT feature points in consecutive frames are first extracted, and their correspondences are determined to represent their motions. Meanwhile, over-segmentation is performed on these frames using a graph-based image segmentation method [35]. As the 2D motions of the feature points belonging to the same moving object should be the same or at least related to each other, the estimated 2D motions can help merge the over-segmented regions, and hence a more accurate segmentation result can be achieved. The feature points of a moving object can then be jointly used to estimate the motion of the object more accurately. Since the motion estimation and the segmentation can refine each other, we propose a hierarchical algorithm to iteratively perform these two tasks so as to achieve a good segmentation and accurate motion estimation. Next, we detect the static background based on the segmentation results, and calibrate the camera concerned. Then, based on the computed 2D motion of each object and the use of the local constant-velocity constraint, we devise an efficient method to estimate the 3D motion of the objects in three consecutive frames. Finally, an HMM-based algorithm integrates the information about the whole video sequence for estimating the 3D trajectories of the moving objects. Fig. 1 illustrates the flowchart of our proposed framework.

2. Feature motion estimation

In order to achieve an accurate 2D motion estimation, feature points in every two consecutive frames and their correspondences are determined by using the SIFT method [30]. Unlike Wills et al. [31] using the Förstner feature [32] and Xiao and Shah [33] using Harris corners in the determination of feature-point correspondences between successive frames, our algorithm adopts the SIFT feature, which has two major advantages. First, SIFT can locate the feature points with sub-pixel accuracy, which is crucial to the 3D-trajectory estimation that follows. Second, as demonstrated in [34], the SIFT descriptors are very robust and distinctive, and are most suited for feature matching, even if the feature points are under rotation, blurring, scale change, or illumination change. Therefore, using SIFT can determine the correspondence between feature points more robustly and accurately. The details will be given in the following subsection. The motions of these feature points will be employed for object segmentation, and will be refined at the same time to achieve a more accurate motion and segmentation result. These will be described in Section 4.

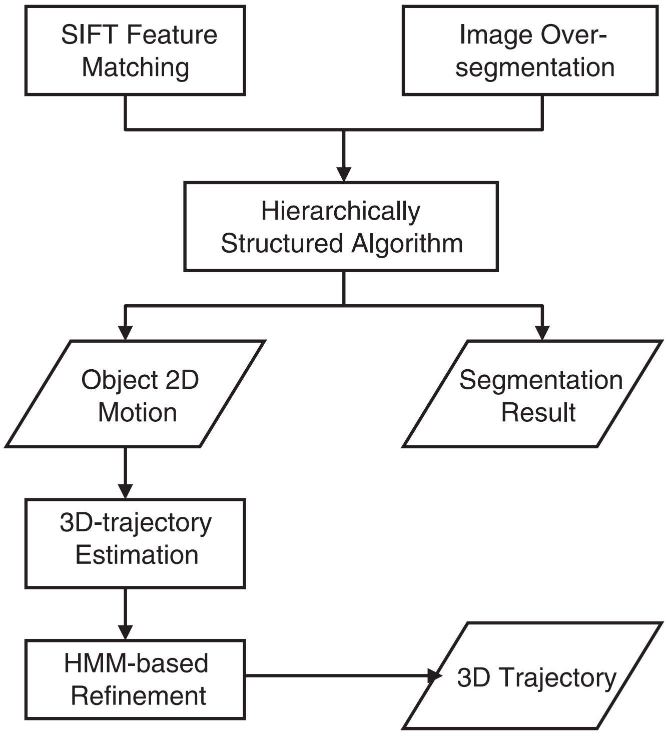


Fig. 1. Flowchart of the proposed framework.

In our approach, the correspondences between the feature points in every two consecutive frames are determined. Finding the correspondences between two successive frames is achieved by comparing the SIFT features using the L_2 norm. Each feature point in the first frame is compared to all the feature points in the next frame. A point in the next frame is selected as the primary corresponding feature point if two conditions are satisfied. The first condition is that the feature point is the nearest one in the second frame, and the second is that the distance between the two features is less than T percent of the second nearest feature point. A list of primary correspondences, from the first frame to the second frame, is produced. A similar list will be obtained by considering the matching of feature points from the second to the first frames. If two feature points match each other in both lists, correspondence between them will be built. Thus, a matching list L of the correspondences is obtained, as follows: $L=\{(x_a^i,y_a^i,x_b^i,y_b^i)|i=1,\ldots,N\}$,where N is the total number of correspondences between the two frames, and (x_a^i,y_a^i) and (x_b^i,y_b^i) are the coordinates of the i^{th} correspondence in the two consecutive frames, respectively.

The parameter T controls the number of feature-point pairs and the accuracy of the correspondences. If T is set at a large value (e.g. 80 to 100), many correspondences are built between the frames. However, some of them may be incorrect. If T is set at a small value (e.g. 30 to 50), nearly all the correspondences are correct. However, the number of feature-point pairs may be insufficient for estimating the trajectory of moving objects. From our experimental results, it is found that $T=60$ is a good trade-off, which can result in a sufficient number of accurate corresponding feature points. Feature correspondences of two frames in the "jeep" and "container" videos are shown in Fig. 2.

3. Image over-segmentation

In the previous section, the preliminary motions of the matched feature points have been estimated. However, our goal is to calculate the 3D motions of moving objects for object segmentation and 3D-trajectory estimation. To achieve this, we have to segment moving objects from videos, and then the motion of an individual object can be estimated by integrating the motions of its features. But, unfortunately, image segmentation is a challenging problem in computer vision. There is no approach that can handle this problem perfectly without involving additional information. In this paper, we propose an efficient segmentation method which needs only an over-segmentation result to achieve the integration of feature motions in a hierarchically structured manner. Furthermore, the motions in each hierarchy level can be used in merging the over-segmented regions. The details of this will be presented in Section 4. In this section, we will describe the method for image over-segmentation.

In our algorithm, a graph-based method [35] is used for image over-segmentation. This method has three main advantages. First, it is computationally efficient, with a computational complexity of $O(n \log n)$, where n is the number of pixels in the image. Second, it can capture the perceptually important non-local properties of an image, so objects with high variability in intensity can also be segmented. Third, as we aim to perform over-segmentation, it is necessary to avoid segmenting pixels from different moving objects into the same segment. This can be achieved by setting a proper value to one parameter only in this graph-based method. To perform over-segmentation, a graph model is firstly built. Let $G=(V, E)$ be an undirected graph with vertices $v_i \in V$, and edge $e_{ij} \in E$ connects the pair of neighboring vertices (v_i, v_j). Each edge e_{ij} is associated with a weight w_{ij}, which is a non-negative value measuring the dissimilarity between the neighboring elements v_i and v_j. For image over-segmentation in our case, the elements in V are pixels, and the weight of an edge is the L_2 distance between the gray-level intensities of the two pixels connected by that edge. After using the graph-based method, a segmentation S will be achieved, which is a partition of V into components such that each component (or region) $C \in S$ corresponds to a connected component in a graph $G'=(V, E')$, where $E' \subseteq E$. There are different ways to measure the quality of a segmentation result, but in general, a high-quality segmentation means that the elements in one component are similar, while elements in different components are dissimilar. In the algorithm, there is an important parameter, k_c, which controls the merging of the nodes, i.e. the segmentation of the image. In particular, it controls the size of the components. Setting k_c at a small value will produce components of a small size; a large k_c will result in large components. Thus, over-segmentation can be achieved by setting an appropriate value to k_c. In our experiments, after normalizing each frame to a specific size (about one million pixels in our algorithm), experiments show that setting $k_c=500$ can always give satisfactory over-segmentation results. Notice that different settings of k_c may lead to different over-segmentation results, but after applying our proposed hierarchically-structured algorithm, the final segmentation results will have almost no difference.

4. Segmentation and object motion estimation

4.1. Problem formulation

In previous sections, we have estimated preliminary 2D motions of the feature points on video frames, and have performed over-segmentation for the frames using a graph-based method. In this section, we will present an algorithm for merging the segments into different motion layers and for calculating the integrated 2D motion of each motion layer from all the feature points' motions. This is achieved by using an iterative method based on a hierarchical structure.

Two tasks will be performed in this section. The first one is to merge the over-segmented regions into accurate motion layers. The 2D motions are the key information used for this purpose. Our idea is based on the fact that the motions of those points belonging to the same object should be either the same or closely related. By comparing the motions of two segments, we can decide whether merging should take place or not. The second task is to refine the estimated 2D motions. As a moving object may have a number of feature points, an accurate motion of the object can therefore be estimated by integrating all the respective feature points' motions based on the structure information of the object. Consequently, the respective refinements of the segments and the motions are related to each other in an iterative manner. We propose an efficient algorithm with a hierarchical structure to perform these two tasks.

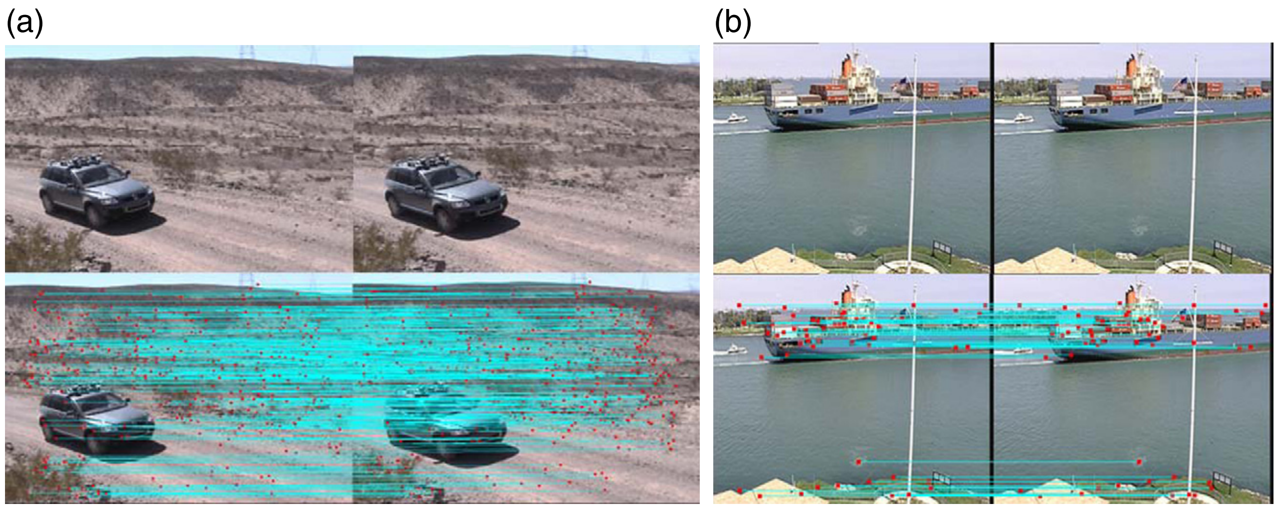


Fig. 2. Feature matching between two frames: (a) results for frames 105 and 109 from the "jeep" sequence. The frame rate is 29 frames per second, and the frame size is 720×480, and (b) results for frames 20 and 40 from the "container" sequence. The frame rate is 25 frames per second, and the frame size is 352×288.

4.2. Hierarchically structured algorithm

The segments, obtained directly by using the over-segmentation algorithm, are used to form a graph, which represents the lowest or finest layer of the hierarchical structure. In the graph, each node corresponds to one segment, and if two segments are neighbors, the corresponding nodes are connected by an edge. Then the nodes are assembled into small aggregates according to the weights of the edges (the weight will be discussed later). The small aggregates form the nodes of the next graph, and are assembled in the same manner. As the nodes are aggregated at different levels, a hierarchical structure is formed. In the hierarchy, each node in a new graph is an aggregation of a number of nodes in the previous graph. (These nodes are called "child nodes" of the node in the new graph.) Therefore, the nodes in every new graph represent a set of merged segments, and the coarsest layer in the hierarchical structure will represent the final segmentation result. To illustrate this method, Fig. 3 presents some major layers of the hierarchical structure which is established for a frame of the video "jeep". In this figure, we randomly choose one pixel in each segment to present the node, and draw the corresponding graph. We can see that, with the new graphs generated, the over-segments can be merged to produce a proper segmentation result.

The details of the merging procedures are as follows. First, for the graph of a certain layer, two measurements are computed for each node: the weighted average $\vec{\mu}_k$ and the weighted deviation $\vec{D}_k$, computed as follows:

$$\vec{\mu}_k^{[t]} = \sum_{p=1}^{N_k} d_p \, \vec{\mu}_p^{[t-1]}, \quad \vec{D}_k^{[t]} = \sum_{p=1}^{N_k} d_p \left(\vec{\mu}_p^{[t-1]} - \vec{\mu}_k^{[t]} \right)^2, \tag{1}$$

where $[t]$ denotes the tth layer in the hierarchical structure, k denotes the kth node (i.e. the kth segment) in the new layer, and N_k is the number of child nodes in the previous layer (i.e. layer $t-1$) of the current node k. $\vec{\mu}_p^{[t-1]}$, where $p=1, \ldots, N_k$, represent the respective averages of those child nodes for node k at layer t. d_p is the normalized distance between the center of the segment p in the previous layer and the center of the newly-aggregated segment (denoted by k in (1)), i.e. $\sum_{p=1}^{N_k} d_p = 1$. When $t=1$, $[t]$ presents the finest layer of the hierarchy, and $\vec{\mu}_p^{[0]}$, which is used to calculate the two measurements in the finest layer, represents the motion of the corresponding feature calculated in the previous stage, i.e. $\vec{\mu}_p^{[0]} = (x_b^p - x_a^p, y_b^p - y_a^p)$. It should be noted that both $\vec{\mu}_k^{[t]}$ and $\vec{D}_k^{[t]}$ are 2-dimensional, as motion in the image domain has 2 components. Although the two measurements are defined in a simple manner, they can capture the motion information about a scene. First, the average can represent the global motion of a segment or an object, while the deviation is influenced by rotation or local non-rigid motions. Furthermore, the motions in a real scene are usually highly structured: feature motions exhibit strong dependencies, especially when they are in close spatially proximity. The two measurements, when calculated in accordance with the hierarchical structure, can exploit the structural information.

With these measurements, the weight of an edge in the graph is defined as follows:

$$w_{ij} = W\left(\left(V_i, V_j\right)\right)^{[t]} = \left\| \vec{\mu}_i^{[t]} - \vec{\mu}_j^{[t]} \right\|^2 + \lambda \left\| \vec{D}_i^{[t]} - \vec{D}_j^{[t]} \right\|^2, \tag{2}$$

where V_i and V_j are the two nodes connected by the edge, and λ determines the relative importance of the two terms. λ is set to 0.15, which is determined by experiments. Then, the nodes are merged again using the graph-based method [35] with a parameter k_c', which has the same meaning as k_c in the initial over-segmentation step, but has its value increased in a new layer (the value will be identified later). The aggregated nodes will form a node of a new graph, and this new graph is the next, coarser layer of the hierarchical structure. In the course of merging the nodes, the segments become larger and larger. The two measurements in (1) of a newly formed segment are computed, which is important for finding the new motion information about the new segments. These procedures are repeated in order to produce other coarser layers until no more nodes aggregate under the terminal parameter k_c'. Thus the whole hierarchy is achieved, and the segmentation result is presented by the coarsest layer. This parameter k_c', which decides the termination of the iteration, increases by 50% for each iteration from the value of k_c used in the over-segmentation procedure. When the value reaches 100 times that of the initial value, k_c' will stay constant until no more nodes aggregate (i.e. the iteration procedure terminates). The 2D motion of each final segment can be represented as the weighted average, which captures the structural information of the segment. The algorithm presented is described by the pseudo-code, as shown in Fig. 4.

4.3. Camera calibration

The motion in a video is determined not only by moving objects but also by the camera. To estimate objects' motions from the video (which will be elaborated on in the following section), camera parameters are also needed. To achieve this, we utilize the feature correspondences in the static background to calibrate the camera. Since we have achieved motion segmentation, we can manually select a segment which represents the static background in a certain frame. Then, the corresponding background in other frames can be automatically determined by feature correspondences. After the detection, all the feature correspondences in the segment can be used to estimate the intrinsic and extrinsic parameters for each frame by the method introduced in [36], which first estimates the projective camera matrix of each frame, and then the parameters by linear and nonlinear approaches.

5. 3D-Trajectory estimation

In our approach, the 3D trajectories of moving objects in a video are estimated based on their respective 2D positions in successive frames. As only two-dimensional (2D) projections of 3D motions are

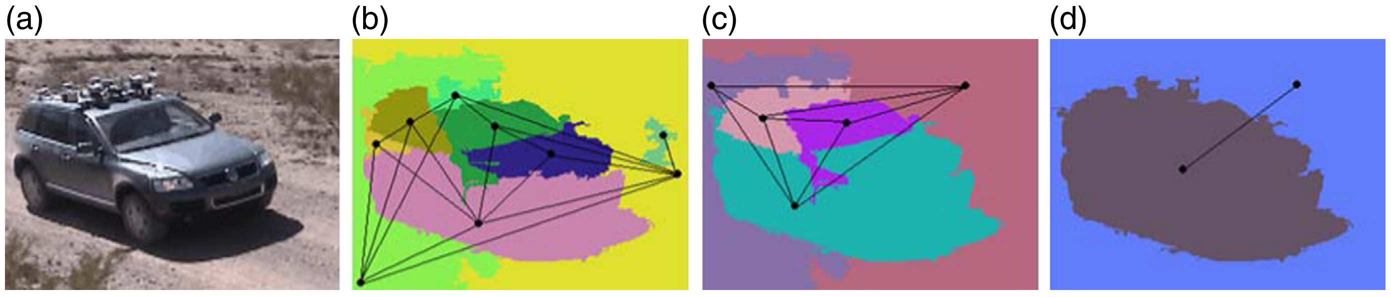


Fig. 3. The hierarchical structure: (a) the video frame, and (b)–(d) the three fine-to-coarse layers in the hierarchical structure.

```
Input: feature correspondences $L = \{(x_a^i, y_a^i, x_b^i, y_b^i) \mid i = 1,\ldots,N\}$, with over-segmentation of the frame
flag ← 0
Map over-segmented frame to a graph A
Calculate the two measurements of the nodes by using (1) and the weights of edges (using Eq 2) in graph A
based on the feature points' motions $\vec{\mu}_p^{[0]} = (x_b^p - x_a^p, y_b^p - y_a^p)$
while flag = 0
    Aggregation of nodes in graph A to form larger segments
    Treat the new segments as nodes to produce a new graph B
    if B = A
        flag ← 1
    else
        Calculate the two measurements of the nodes and the weights of edges in graph B
        A ← B
    end-if
end-while
Obtain the segmentation result and the 2D motions of segments based on graph A
```

Fig. 4. The pseudo-code of our segmentation algorithm.

available in a monocular video, the estimation is therefore an ill-posed problem. In order to tackle this problem, certain constraints must be applied so as to make it tractable. In our algorithm, we use the constraint that objects are moving with constant velocities within consecutive frames of a video. In fact, if the video frame-rate is high enough, this constraint will always be satisfied, regardless of whether the real motion is constant or not. Using this local constant-velocity constraint, we propose an efficient method to calculate the global 3D trajectories of moving objects in a monocular video.

The method will first be illustrated in the case of a fixed camera. Then, we extend it to a moving camera with varying internal camera parameters. A 3D scene is shown in Fig. 5(a), where the point O is the camera centre, and $S1$, $S2$, and $S3$ denote the respective 3D positions of the point, moving at a constant speed, at three time instants with the same interval. The plane in Fig. 5(a) denotes the image plane, where the points labeled $I1$, $I2$, and $I3$ are the corresponding projections of the three 3D points $S1$, $S2$, and $S3$. The detailed 2D positions of the point in the image plane are shown in Fig. 5(b). In our algorithm, these three 2D coordinates are used to calculate the 3D motion of the point.

The relationship between a 3D position and its 2D projection on the image plane can be described as follows:

$$\begin{pmatrix} x/z \\ y/z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{3}$$

where $\boldsymbol{X} = (X, Y, Z)$ denotes the 3D coordinates of the point in the world coordinate, and $\boldsymbol{x} = (x/z, y/z)$ is the 2D projection on the image plane. The matrix

$$P = K[R|t] = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4}$$

is the projection matrix of the camera, where

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

is the intrinsic matrix, and

$$[R|t] = [I|0] \tag{6}$$

is the external matrix. This camera model and the camera intrinsic parameters f, p_x, p_y are explained in [37]. As the camera coordinate is used as the world coordinate in this formulation, the external matrix has its form, as shown in (6).

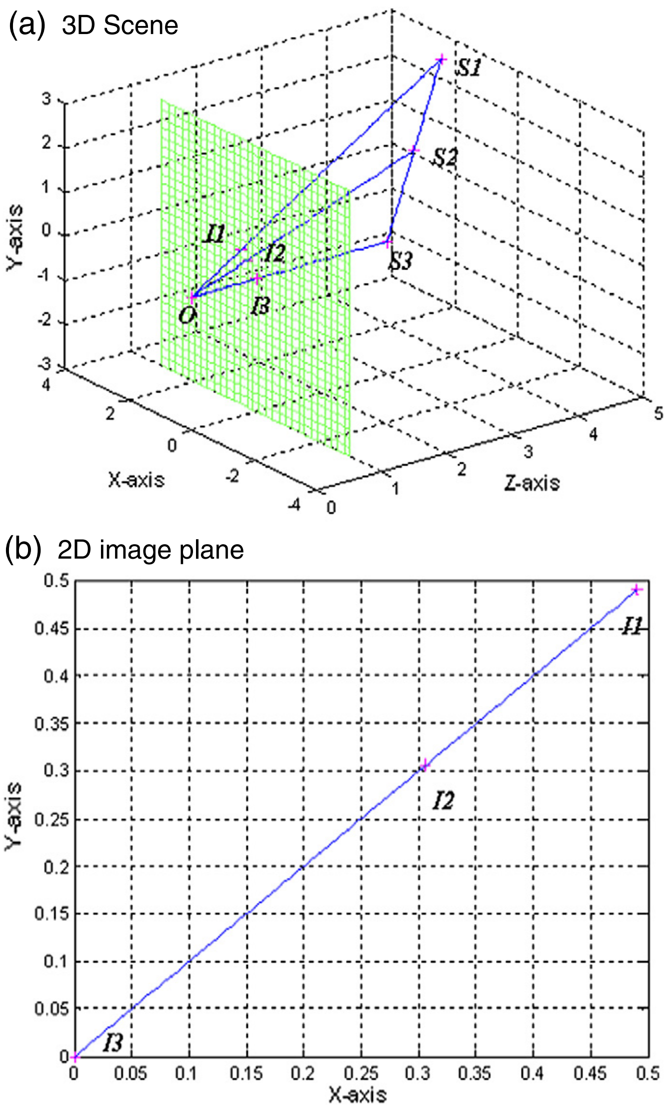


Fig. 5. Three 3D points and their respective 2D projections.

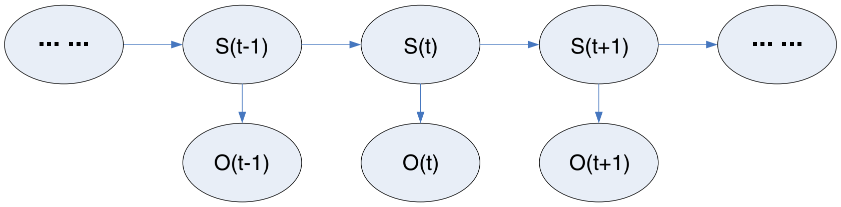


Fig. 6. The architecture of the HMM used in the algorithm.

Based on the constraint and the relationship of the projections, we can obtain the following equations:

$$X_{S1}+X_{S3}=2X_{S2},Y_{S1}+Y_{S3}=2Y_{S2},Z_{S1}+Z_{S3}=2Z_{S2},$$
$$x_{I1}=\frac{X_{S1}}{Z_{S1}}f+p_x,x_{I2}=\frac{X_{S2}}{Z_{S2}}f+p_x,x_{I3}=\frac{X_{S3}}{Z_{S3}}f+p_x, \tag{7}$$
$$y_{I1}=\frac{Y_{S1}}{Z_{S1}}f+p_y,y_{I2}=\frac{Y_{S2}}{Z_{S2}}f+p_y,y_{I3}=\frac{Y_{S3}}{Z_{S3}}f+p_y,$$

where (X, Y, Z) and (x, y) are the 3D and 2D coordinates of the feature point, respectively. The subscripts in (7) denote the corresponding 2D and 3D points, as described previously. The first three equations in (7) are obtained using the local constant-velocity constraint, and the other equations are derived from the relationship of the projections.

In (7), we have a total of nine variables $(X_{s1},Y_{s1},Z_{s1},X_{s2},Y_{s2},Z_{s2},X_{s3},Y_{s3},Z_{s3})$ and nine equations. It seems that we can solve the problem completely, but in fact the projections of the three 3D positions on the image plane will be on the same straight line, which means that:

$$\frac{y_{I3}-y_{I1}}{x_{I3}-x_{I1}}=\frac{y_{I2}-y_{I1}}{x_{I2}-x_{I1}}. \tag{8}$$

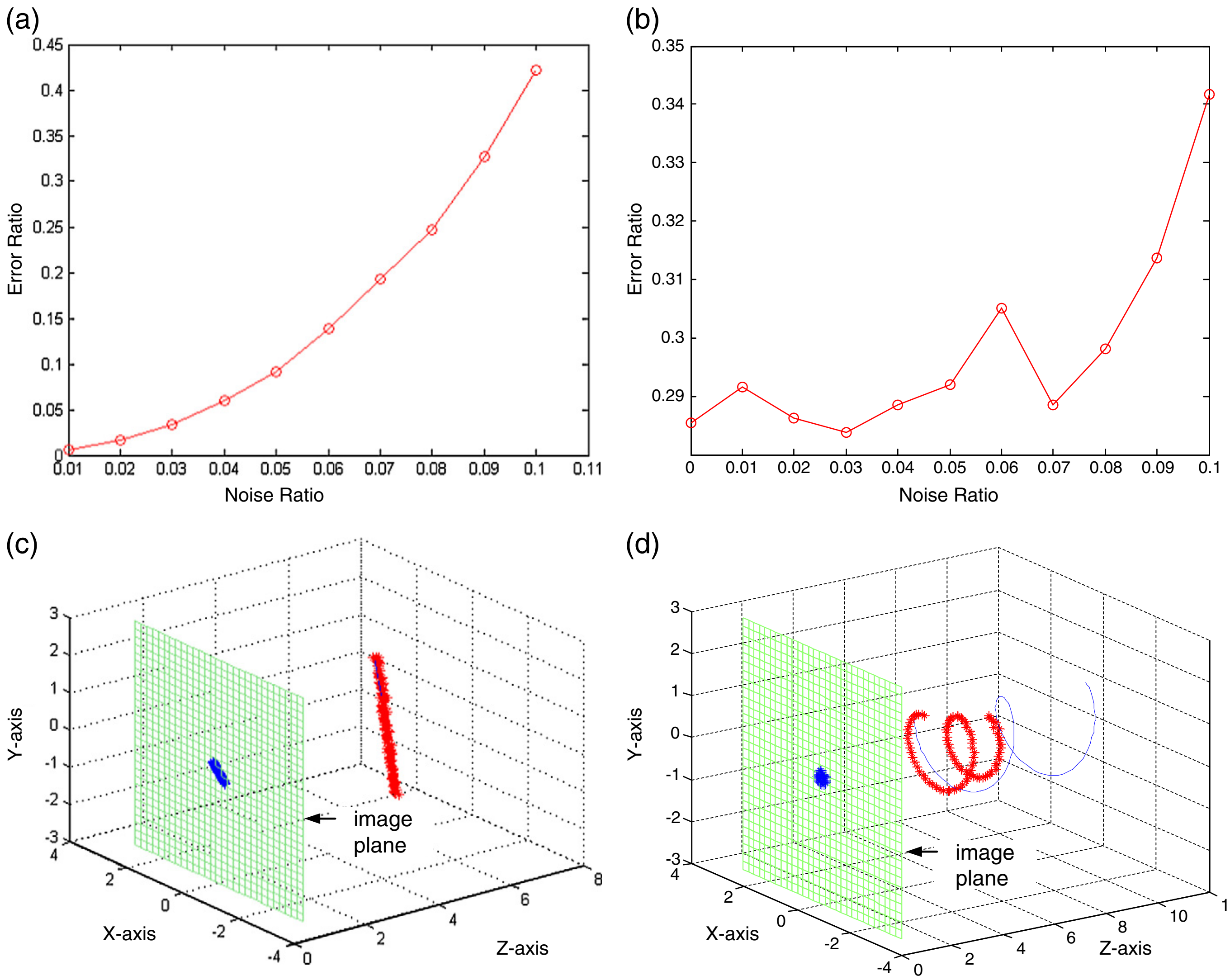


Fig. 7. Estimation errors under different levels of noise based on synthetic videos with: (a) linear motions, and (b) non-linear motions. The 3D and 2D trajectories of the synthetic sequences with: (c) linear motions, and (d) non-linear motions. The blue line and the red line in the 3D space represent the synthetic and the estimated 3D trajectories, respectively.

In other words, any eight of the equations can be used to derive the remaining one. Thus, only eight independent equations are available. In this case, we cannot obtain the 3D coordinates of each position without other additional information. Nevertheless, we can still calculate the 3D position by treating one of the variables as already known. Then, the relative position of the moving object in the 3D space can be calculated. For convenience, the depth of $S2$ is treated as known already; we can then obtain the complete solution as follows.

$$\begin{cases} Z_{S1} = \dfrac{2(x_{I3}-x_{I2})Z_{S2}}{x_{I3}-x_{I1}} \\ Z_{S3} = \dfrac{2(x_{I2}-x_{I1})Z_{S2}}{x_{I3}-x_{I1}} \end{cases} \text{or} \begin{cases} Z_{S1} = \dfrac{2(y_{I2}-y_{I3})Z_{S2}}{y_{I1}-y_{I3}} \\ Z_{S3} = \dfrac{2(y_{I1}-y_{I2})Z_{S2}}{y_{I1}-y_{I3}} \end{cases} \tag{9}$$

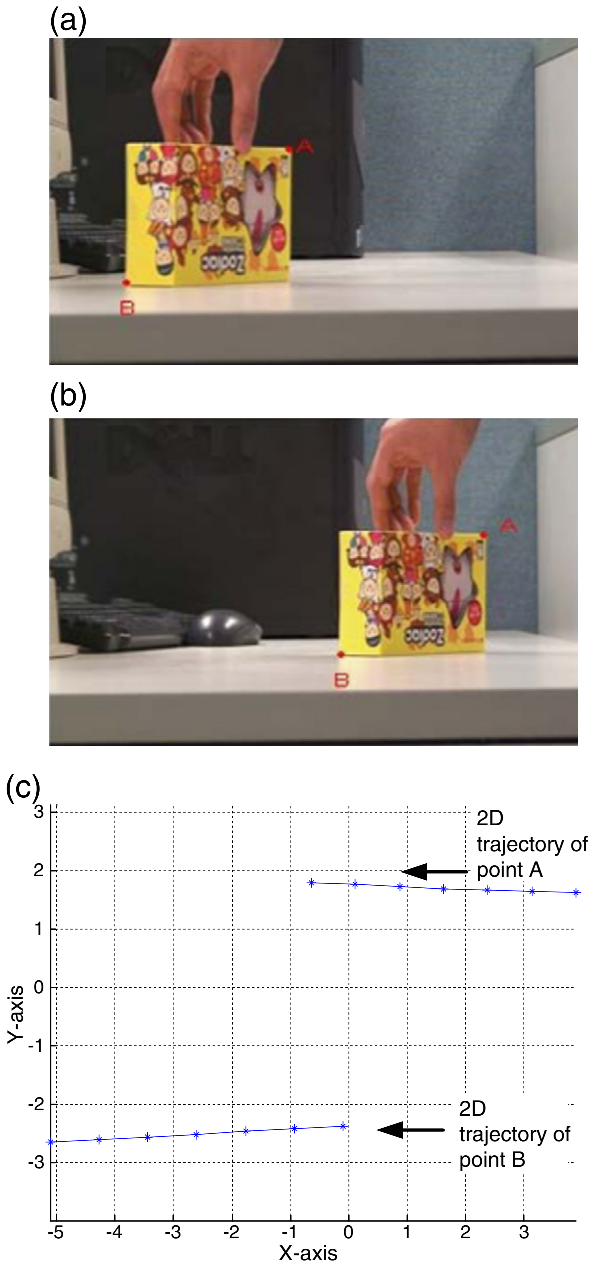


Fig. 8. (a) The first frame (frame number 1), (b) the last frame (frame number 121) of the “cube1” sequence used in our experiment, and (c) the 2D trajectories of the two points marked “A” and “B”. The frame rate is 25 frames per second, and the frame size is 1440×1080.

The relationship between the variables in (9) and the result in the previous section can be formulated as follows:

$$\vec{\mu}_k^{[t_{\max}]} = (x_{I2}-x_{I1}, y_{I1}-y_{I2}). \tag{10}$$

where $\vec{\mu}_k^{[t_{\max}]}$, estimated from frames $I1$ and $I2$, represents the weighted average motion of the segment k in the coarsest layer . By further utilizing $\vec{\mu}_k^{[t_{\max}]}$ calculated from frames $I2$ and $I3$, other variables in (9) can also be obtained. In (9), the values of X and Y are not shown, because they can be computed easily by using the value of

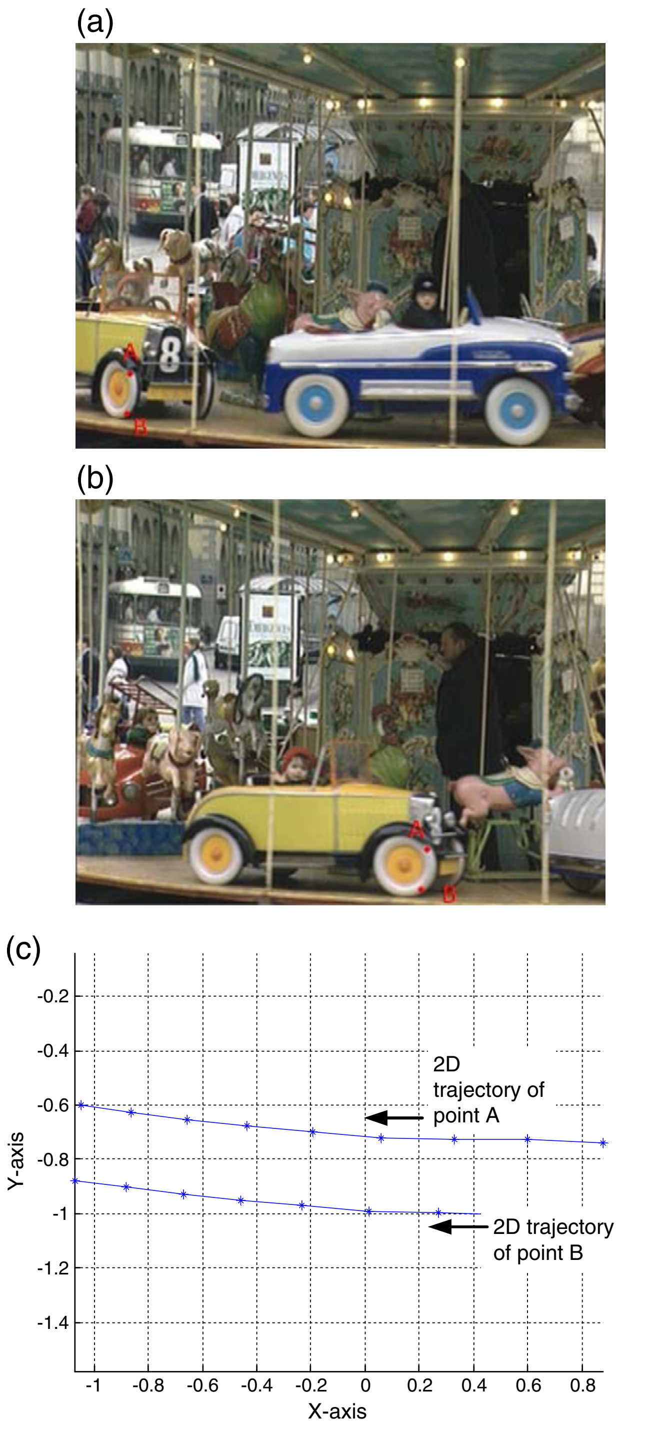


Fig. 9. The first frame (frame number 161), (b) the last frame (frame number 193) of the “funfair” sequence used in our experiment, and (c) the 2D trajectories of the two points marked “A” and “B”. The frame rate is 25 frames per second, and the frame size is 352×288.

Z from the system of equations in (7). By performing this procedure repeatedly, the 3D coordinates of the point in all the frames can be estimated. As a consequence, our method achieves 3D-trajectory estimation of an object up to a certain scale. Besides this, from (9), we can find that when the depth of the point at a certain time instant is known, the respective depths in the consecutive frames can then be calculated. Generally speaking, if we have partitioned a video into video shots and the depth information about one of the frames in a shot (usually the key-frame) has been predetermined, our proposed method can estimate the depths of all the moving objects within the shot. Consequently, in combination with an image depth estimation algorithm, our method can be applied to convert a 2D video to a 3D video.

The extension of the method to a moving camera with varying internal parameters is discussed in the following. First, the equations used to solve the 3D trajectory (for a fixed camera, illustrated in (7)) have the following formulation:

$$\begin{aligned}
&X_{S1} + X_{S3} = 2X_{S2}, Y_{S1} + Y_{S3} = 2Y_{S2}, Z_{S1} + Z_{S3} = 2Z_{S2},\\
&x_{I1} = \frac{X'_{S1}}{Z'_{S1}} f_1 + p_{x1}, x_{I2} = \frac{X'_{S2}}{Z'_{S2}} f_2 + p_{x2}, x_{I3} = \frac{X'_{S3}}{Z'_{S3}} f_3 + p_{x3},\\
&y_{I1} = \frac{Y'_{S1}}{Z'_{S1}} f_1 + p_{y1}, y_{I2} = \frac{Y'_{S2}}{Z'_{S2}} f_2 + p_{y2}, y_{I3} = \frac{Y'_{S3}}{Z'_{S3}} f_3 + p_{y3}.
\end{aligned} \tag{11}$$

f_n, p_{xn}, and p_{yn} denote the intrinsic parameters of the camera when capturing frame n. $\boldsymbol{X}_{\text{cam}} = (X', Y', Z')$ is the coordinates of the feature point in the camera coordinates, while $\boldsymbol{X} = (X, Y, Z)$ is in the world coordinates. (x, y) and the subscripts have the same meaning as (7). Notice that in the first three equations, the feature points are in the world coordinates, while in the following six equations they are in the camera coordinates.

We consider the relationship of the camera coordinates and the world coordinates. After calibration, this relationship can be expressed as follows:

$$X_{\text{cam}} = R_{\text{cam}}(X - \mathbf{C}_{\text{cam}}), \tag{12}$$

where $\boldsymbol{X}_{\text{cam}}$ is in the camera coordinates, R_{cam} and $\mathbf{C}_{\text{cam}}$ are the rotation matrix and the translation vector of this camera, respectively, and $\boldsymbol{X}$ is in the world coordinates, and is the same as the camera

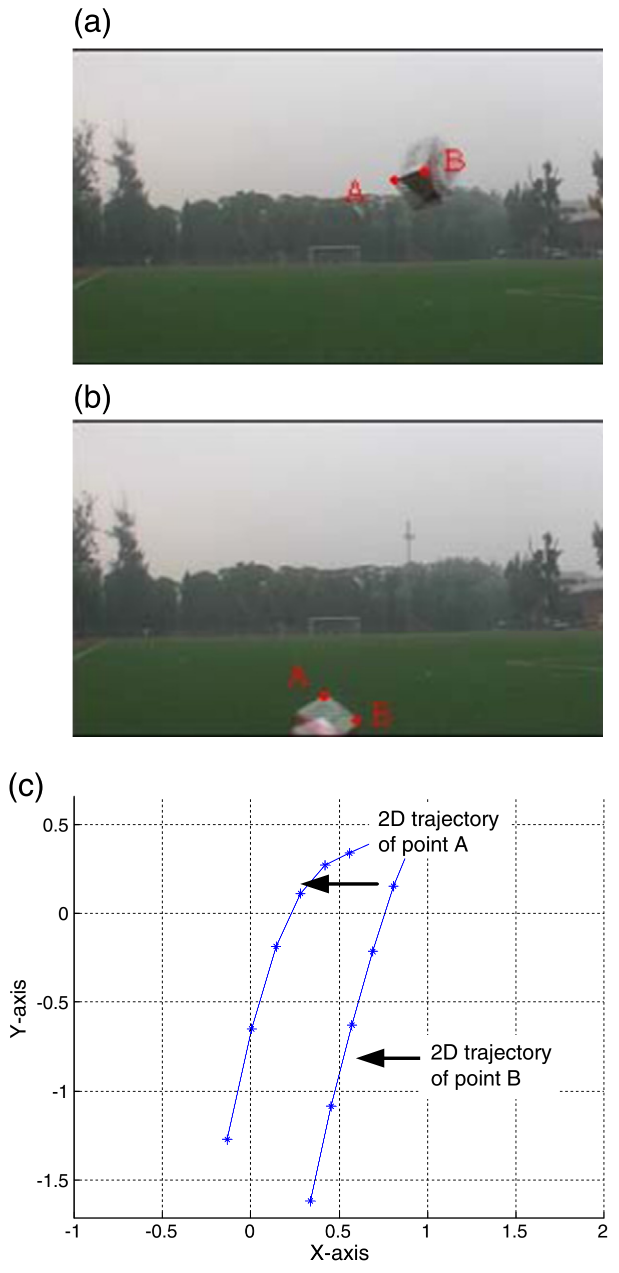


Fig. 10. The first frame (frame number 1), (b) the last frame (frame number 7) of the "cube2" sequence used in our experiment and (c) the 2D trajectories of the two points marked "A" and "B". The frame rate is 25 frames per second, and the frame size is 720×576.

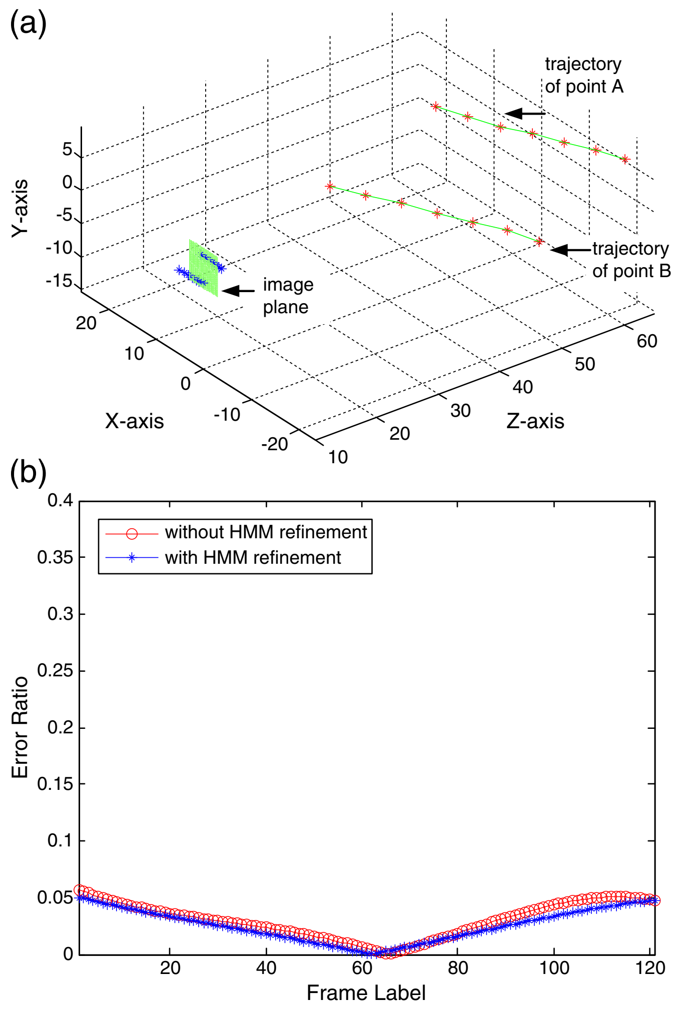


Fig. 11. (a) The 2D and estimated 3D trajectories for the "cube1" sequence, and (b)the reconstruction errors for the "cube1" sequence.

coordinates of the first frame, whose R_{cam} equals the identity matrix and $\boldsymbol{C}_{cam}$ equals the zero matrix $\mathbf{0}$. As R_{cam} and $\boldsymbol{C}_{cam}$ can be calculated as described in Section 4.C, the relationship can be fully determined.

When referring to (11), as f_n, p_{xn}, and p_{yn} can also be calculated as described in Section 4.C, only $\boldsymbol{X}=(X, Y, Z)$ are unknown variables, and the formulation is similar to the Eq. (7). So (11) can be solved in the same manner, and the 3D-trajectory estimation under a moving camera can be fully solved by integrating the camera calibration method.

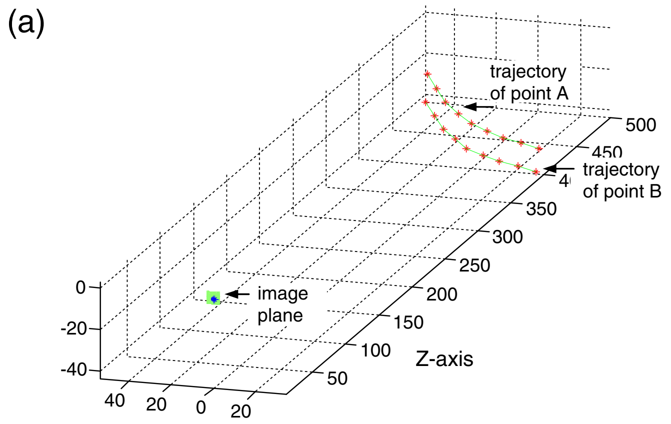
6. HMM-based Refinement

To estimate 3D trajectories accurately, in addition to the hierarchically structured method which utilizes intra-frame information, we have also developed a Hidden Markov Model (HMM)-based method for estimating 3D trajectories in a large number of video frames by utilizing inter-frame motion information. HMM [38] has been used in many applications, such as cryptanalysis, speech recognition, machine translation, and gene prediction. Fig. 6 shows the architecture of the HMM used in our application, whereby each oval represents a continuous random variable. The random variable $S(t)$ denotes the hidden state at time t, and the random variable $O(t)$ is the observation at time t. The arrows in the graph represent the conditional dependencies of the different variables. For a moving object, its true 3D motion at time t is treated as the hidden random variable $S(t)$, and the observation random variable $O(t)$ is used to represent the 3D motion at time t, calculated by using our proposed 3D trajectory algorithm. The goal of this stage of our algorithm is to compute the most likely state sequence $S(1)$, $S(2)$, …, $S(T)$, with a given observation $O(1)$, $O(2)$, …, $O(T)$ and the HMM parameter λ, as shown in Fig. 6, where T denotes the number of frames. The HMM is modeled as follows. The observation probability distribution is assumed to be a Gaussian distribution centered at the value of the corresponding hidden state with a standard deviation σ_1. The state-transition probability distribution is also modeled as a Gaussian distribution, whose mean is the value of the previous hidden state; and the standard deviation is σ_2, because the motion of an object will not change abruptly. The formulations are therefore as follows:

$$p(O(t)|S(t)) = \frac{1}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{(O(t)-S(t))^2}{2\sigma_1^2}\right), \tag{13}$$

and

$$p(S(t+1)|S(t)) = \frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{(S(t+1)-S(t))^2}{2\sigma_2^2}\right). \tag{14}$$



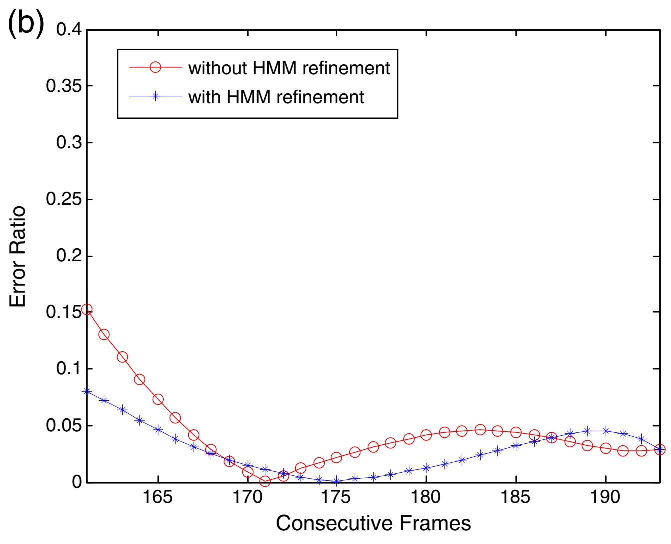
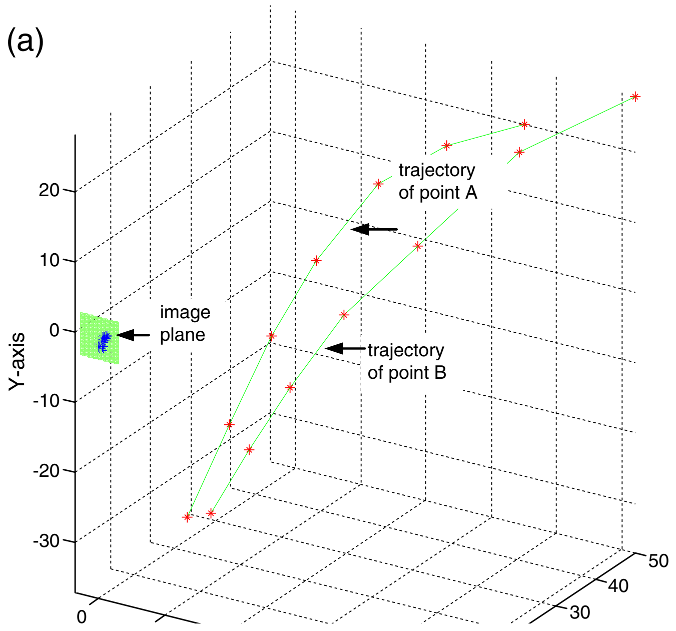


Fig. 12. (a) The 2D and estimated 3D trajectories for the "funfair" sequence, and (b) the reconstruction errors for the "funfair" sequence.



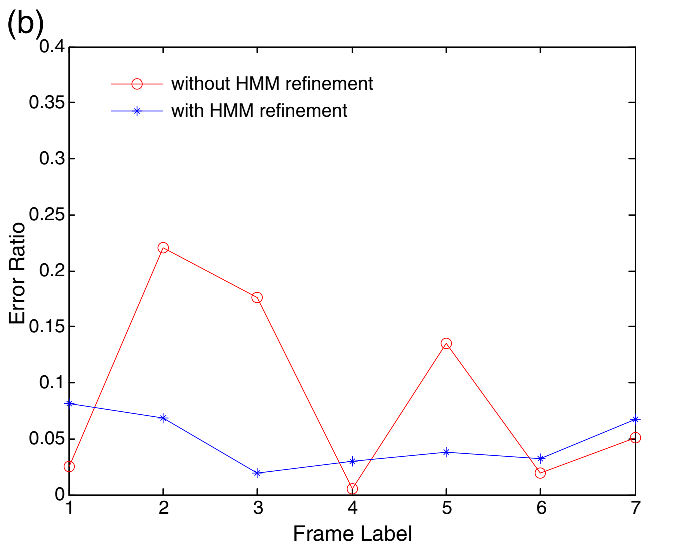


Fig. 13. (a) The 2D and estimated 3D trajectories for the "cube2" sequence, and (b) the reconstruction errors for the "cube2" sequence.

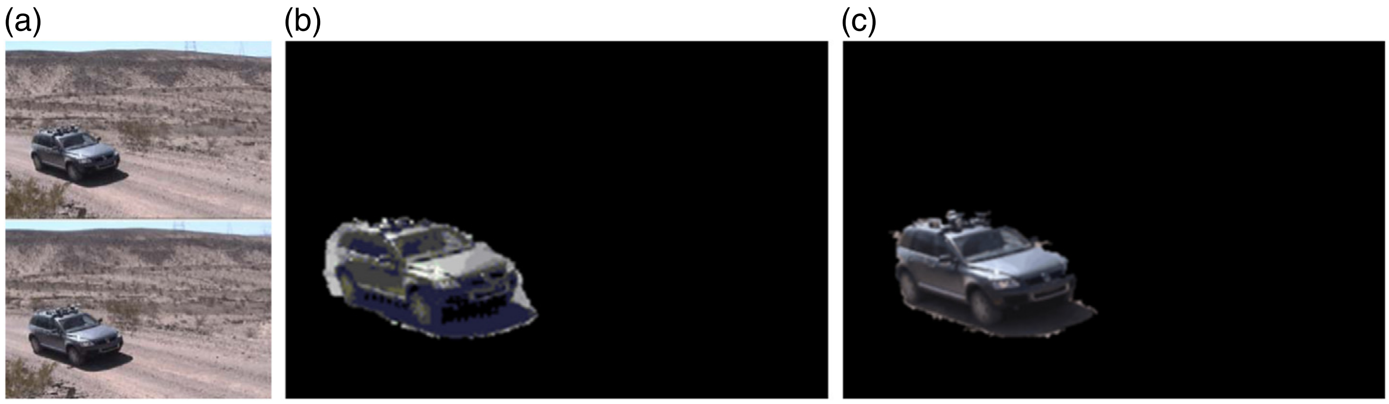


Fig. 14. Segmentation results for the "jeep" video: (a) the two frames (frame numbers 105 and 109) used for motion segmentation, (b) the segmentation result using Wills's method, and (c) the segmentation result using the proposed method. The frame rate is 29 frames per second, and the frame size is 720×480.

After obtaining the observation and the HMM parameter λ, the following probability is to be maximized:

$$p(S(1), S(2)\cdots S(T) \mid O(1), O(2)\cdots O(T), \lambda), \tag{15}$$

which can be expressed by the conditional distribution as follows:

$$p(O(1) \mid S(1)) \prod_{i=1}^{T-1} [p(S(i+1) \mid S(i)) p(O(i) \mid S(i))]. \tag{16}$$

As maximizing the probability is equivalent to maximizing its log value, we have

$$\arg \max_{S(1), S(2)\cdots S(T)} \log\left(p(O(1) \mid S(1)) \prod_{i=1}^{T-1} [p(S(i+1) \mid S(i)) p(O(i) \mid S(i))]\right), \tag{17}$$

and this function can be further simplified as follows:

$$\begin{aligned} &\log\left(\frac{1}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{(O(1)-S(1))^2}{2\sigma_1^2}\right)\prod_{i=1}^{T-1}\left[\frac{1}{2\pi\sigma_1\sigma_2}\exp\left(-\frac{(S(i+1)-S(i))^2}{2\sigma_2^2}\right)\right.\right. \\ &\qquad \left.\left.\times \exp\left(-\frac{(O(i)-S(i))^2}{2\sigma_1^2}\right)\right]\right) \\ &\qquad = \left(-\sum_{i=1}^{T}\frac{(O(i)-S(i))^2}{2\sigma_1^2} - \sum_{j=1}^{T-1}\frac{(S(j+1)-S(j))^2}{2\sigma_2^2} + k\right), \end{aligned} \tag{18}$$

where k is related to σ_1 and σ_2, but does not affect the solution. As shown in (18), only the relative value of σ_1 and σ_2 will affect the solution, so we can further simplify the formulation as follows:

$$\arg \min_{S(1), S(2)\cdots S(T)} \left(\sum_{i=1}^{T}(O(i)-S(i))^2 + \alpha \sum_{i=1}^{T-1}(S(j+1)-S(j))^2\right), \tag{19}$$

where α denotes the trade-off between the two terms. If the motion in the video frames changes rapidly, α should be a small number, e.g. 0.5 to 1. To solve the minimization problem, we set the deviation of the function to 0, and the formulation can be expressed as follows:

$$\begin{bmatrix} 1+2\alpha & -\alpha & & & & \\ -\alpha & 1+2\alpha & -\alpha & & & \\ & -\alpha & 1+2\alpha & -\alpha & & \\ & & \ddots & \ddots & \ddots & \\ & & & -\alpha & 1+2\alpha & -\alpha \\ & & & & -\alpha & 1+2\alpha \end{bmatrix} \begin{bmatrix} S(1) \\ S(2) \\ S(3) \\ \vdots \\ S(T-1) \\ S(T) \end{bmatrix} = \begin{bmatrix} O(1) \\ O(2) \\ O(3) \\ \vdots \\ O(T-1) \\ O(T) \end{bmatrix}. \tag{20}$$

This matrix equation can be solved even for a large value of T. In our application, $O(t)$ is the motion vector $\boldsymbol{X} = (X, Y, Z)$ of a certain object in frame I_t, and $S(t)$ is the output motion vector, which forms the final 3D trajectory. Thus, with the use of the HMM-based method, we can incorporate the information in all the video frames to achieve an optimal trajectory estimation.

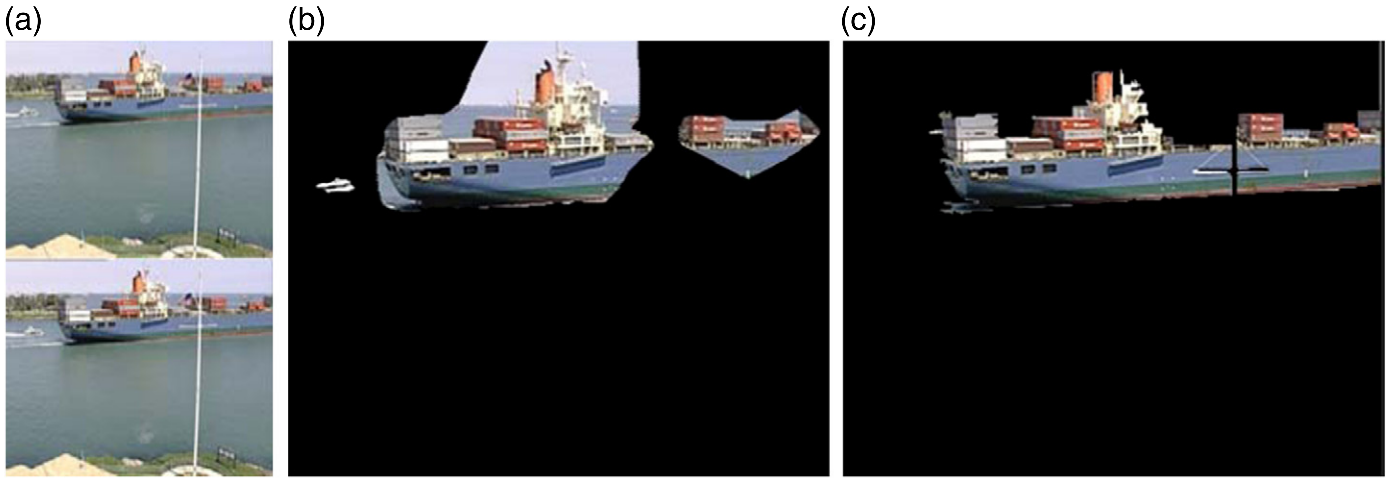


Fig. 15. Segmentation results for the "container" video: (a) the two frames (frame numbers 20 and 40) used for motion segmentation, (b) the segmentation result using Wills's method, and (c) the segmentation result using the proposed method. The frame rate is 25 frames per second, and the frame size is 352×288.

7. Experimental result

7.1. 3D Trajectories of moving objects

The 3D-trajectory estimation algorithm is a major contribution of this paper. In this subsection, we will evaluate the performance of the algorithm using both synthetic data and real data, with a point's 2D position in consecutive frames given. With the use of synthetic data, we evaluate the robustness of our algorithm via the addition of noise. With the real data and the ground truth results, we can measure the accuracy of our algorithm. Notice that all the experiments conducted in this subsection are based on individual points only. Our whole method actually integrates all the feature motions with structural information in the extraction of objects' 2D motions, so that it can be more robust to noise and can therefore achieve a better performance in 3D estimation. The overall performance will be shown in Subsection *C*.

7.1.1. Experiments using synthetic data

The first synthetic video used in our experiment is generated by projecting a moving object, which moves with a linear, constant velocity in the 3D space, onto a 2D image plane. A feature point on the object in the synthetic video frames is used to estimate the 3D trajectory. To evaluate the robustness of our algorithm to noise, we add a zero-mean Gaussian noise to the position of the feature point under consideration. We measure the algorithm's accuracy when the Gaussian noise has different standard deviations. Fig. 7(a)-(b) shows the accuracy of our algorithm under different standard deviations of Gaussian noise, where the error ratio is the error distance between the estimated 3D positions and the corresponding true 3D positions divided by the distance moved by the object between the corresponding two successive frames. The results shown in Fig. 7(a) are obtained by taking the average of 100 such simulations. From Fig. 7(a), we can see that the error ratio will increase quickly when the noise increases. When the noise standard deviation is equal to 5%, the corresponding error ratio is only 0.1. The error ratio is equal to zero if the standard deviation is zero. Fig. 7(c) shows the estimation result when the noise standard deviation is set to 2.5%, which may be considered noisy for real data. The line on the image plane denotes the locations of the feature point in the synthetic video. The thick red line in the 3D space is the estimated 3D trajectory of the moving object, while the thin blue line represents the true trajectory of the object. (Only a small part of the blue line can be seen because most part of the line coincides with and is covered by the red line.). In addition to linear motions, we have also synthesized local non-linear motion to test our method. In Fig. 7(d), the synthetic 3D trajectory (blue line) and the corresponding reconstructed 3D trajectory (red line) under a noise with 2.5% standard deviation are illustrated. Fig 7(b) shows the error ratio under different noise ratios. Although the estimated results have more deviations from the true motions as compared to the linear-motion case, the shape of the 3D trajectory is correctly estimated. This implies that our method can handle non-linear motion to a certain extent.

Table 1
Comparison of the segmentation performances of Wills's method and the proposed method in terms of ROCs.

	The “jeep” sequence		The “container” sequence	
	Frame 105	Frame 141	Frame 21	Frame 121
Wills's method	41.12	8.13	28.00	5.83
Our method	346.90	155.50	39.05	173.85

7.1.2. Experiments using real data

Three different video sequences, with two of the sequences having their objects moving at varying velocities, were used to evaluate our proposed algorithm. The “cube1” sequence, as shown in Fig. 8, contains a cube moving away from the camera at a nearly constant velocity. Thus, the scale of the moving object is changing. The “funfair” sequence, as shown in Fig. 9, contains a toy car moving in a curved trajectory. The motion of this object is not linear. The “cube2” sequence, as shown in Fig. 10, contains a cube moving in rotation and under the gravitational force. The moving object here is under a complex motion.

To test the performance of our 3D-trajectory estimation method, the 2D trajectory of the object concerned is assumed to be known. Thus, in each of these three videos, two points, denoted as *A* and *B*, are manually selected and tracked, as shown in Figs. 8–10. These two points are a fixed distance apart. Although the true trajectories of the objects are unknown, we can measure the variations of the distances between points *A* and *B* in the estimated 3D trajectories as an indication of estimation accuracy. If the estimation is accurate, the variations should

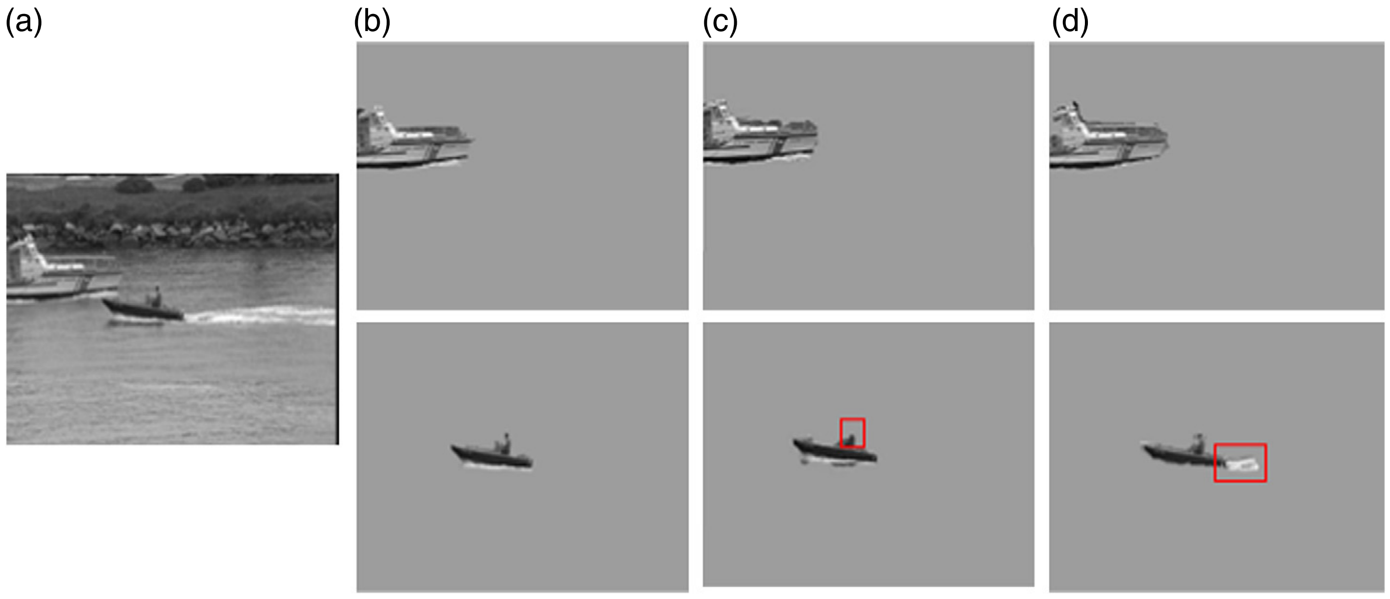


Fig. 16. Comparison results for the 20th frame of the “coastguard” sequence: (a) the original frame, (b) the ship and boat segments using our proposed method, (c) the results of Wang's method, and (d) the results of Tasi's method. (Wang et al. did not indicate which frame is used for the result as shown. From our observation, it is about the 20th frame, so we have used this frame for comparison.).

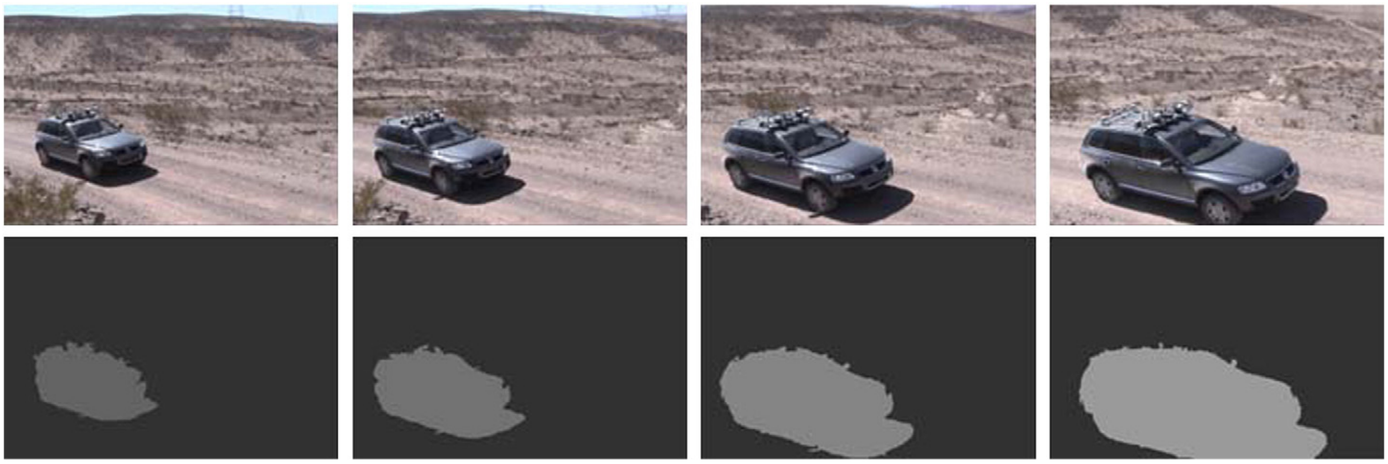


Fig. 17. Video frames and the corresponding segmentation results for the “jeep” sequence using our proposed method. The four columns show the results for frames 105, 117, 129 and 141, respectively.

be close to zero, and vice versa. As the car in the “funfair” sequence and the box in the “cube2” sequence both move with nonlinear motions, the measured variations are expected to be larger than that for the “cube1” sequence. Nevertheless, our locally linear, constant speed assumption is applied to only a small number of successive frames, so the overall nonlinear motions of the objects can be captured.

The 3D-trajectory reconstruction results and errors for the “cube1”, “funfair”, and “cube2” sequences are shown in Figs. 11—13, respectively. The estimation error in one frame is the absolute deviation of the measured distance between the two feature points *A* and *B* from the corresponding mean distance over the whole sequence. Furthermore, the error ratio here is defined as the error divided by the mean distance. From Figs. 11(b), 12(b) and 13(b), we can see that our algorithm with the HMM-based refinement can achieve a better performance than without using the refinement scheme. Furthermore, the improvement achieved by using the refinement is more obvious when a video contains complex motions. This can be seen by comparing the results shown in Fig. 13(b) for a complex-motion video with those shown in Fig. 11(b) for a simple-motion video. Consequently, the HMM-based method can make the local-constant velocity constraint used in our algorithm more adaptive to varying motions. From the results, the error ratio of each of these sequences with different motions is less than 8%. Notice that, for the “cube1” sequence (Fig. 11(b)), one frame in the middle of the sequence achieves the lowest error ratio. This is because the trajectories of Points *A* and *B* are nearly straight lines. The distance of the two trajectories changes linearly. As the error ratio is defined as the distance divided by the mean, there should, in theory, be at least one frame in between with a zero error ratio.

7.2. Experiments on our segmentation algorithm

Segmentation plays an important role in 2D-to-3D conversion. In our proposed algorithm, an accurate segmentation is achieved by using the 2D motion estimation in a hierarchical structure. In this subsection, we will evaluate the performance of our segmentation algorithm. Since our method utilizes feature correspondences as motion information, the feature-based motion-segmentation method proposed by Wills et al. [31] will be compared to our proposed algorithm. Their method first utilizes the Förstner operator to detect interesting points. Then, by comparing the features at these points, initial correspondences are achieved. Based on the RANSAC method, the scene's motion fields can be estimated. Finally, a smooth assignment of pixels to motion fields can be achieved by using a fast approximate graph-cut algorithm based on the Markov random field. Figs. 14 and 15 show the respective motion-

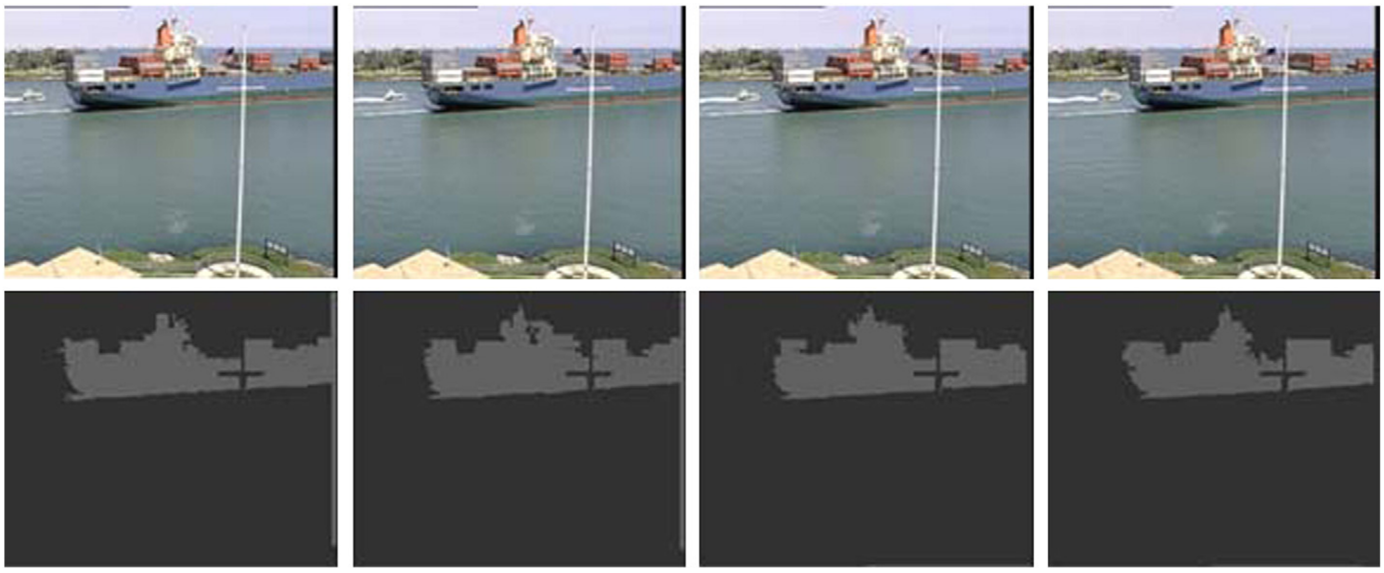
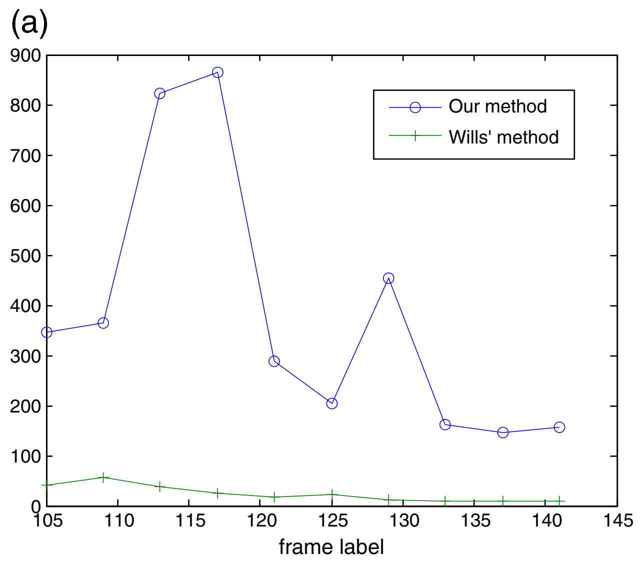


Fig. 18. Video frames and the corresponding segmentation results for the “container” sequence using our proposed method. The four columns show the results for frames 21, 61, 81 and 121.



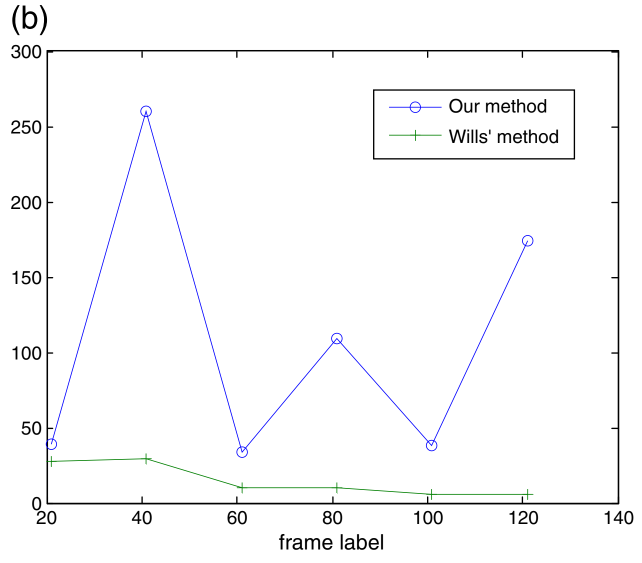


Fig. 19. ROC for (a) the "jeep" sequence, and (b) the "container" sequence.

segmentation results based on the two methods. We can see that, with the "jeep" sequence, our proposed method can achieve a more accurate segmentation performance than Wills's method, which also segments some background regions, like those to the left of and behind the jeep, as part of the foreground object. With the "container" sequence, our proposed method can still achieve an accurate segmentation. In particular, the flagpole which occludes the ship can be segmented correctly even though it is very thin. Using Wills's method, because of the uniform and similar color of the sky, the ship, and the sea, the segmentation result is not accurate. Wills's method can detect the small boat, but then wrongly segments it into the ship layer.

As mentioned above, besides feature-based methods, there exist other methods for video segmentation. In our experiment, we compare the performance of our proposed algorithm with two other methods. One method, proposed by Wang et al. [39], utilizes a probabilistic framework for spatio-temporal segmentation. The other, proposed by Tsai et al. [14], first partitions a video into 3D watershed volumes, and then merges the partitions to obtain the required objects. Fig. 16 shows the results based on the different methods. We can see that Wang's method wrongly segments the head of the man on the boat into the background, while Tsai's method assigns some parts of the bow wave to the boat layer.

In order to quantify the performance of a segmentation method, the Receiver Operating Characteristic (ROC) is used, which is defined as follows:

$$ROC = \frac{TP\,/\,Total \text{ number of foreground pixels}}{FP\,/\,Total \text{ number of background pixels}}, \tag{21}$$

where TP is the number of foreground pixels which are correctly segmented into the foreground segment, and FP is the number of background pixels which are wrongly segmented into the foreground segment. Table 1 tabulates the ROCs for both the Wills's method and our proposed method. We can see that, for both sequences, our proposed method can achieve a higher ROC than Wills's method can.

To evaluate the stability of the segmentation method, we perform segmentation over a number of video frames (the "jeep" sequence has 37 frames, and the "container" sequence has 101 frames). Some of the segmented frames are shown in Figs. 17 and 18. We can see that the segmentation results using our proposed algorithm are stable and consistent in the frames. Furthermore, Fig. 19(a) and (b) show the ROCs of our method and Wills's method, based on the "jeep" sequence and the "container" sequence, respectively. We can see that, in both of these sequences, our method achieves a better performance than Wills's does.

To illustrate the performance of our algorithm with multiple objects, two video sequences, "Akko&Kayo" and "race", were used, both of which contain two moving objects. The two segmentation results are shown in Figs. 20 and 21, respectively. We can notice that

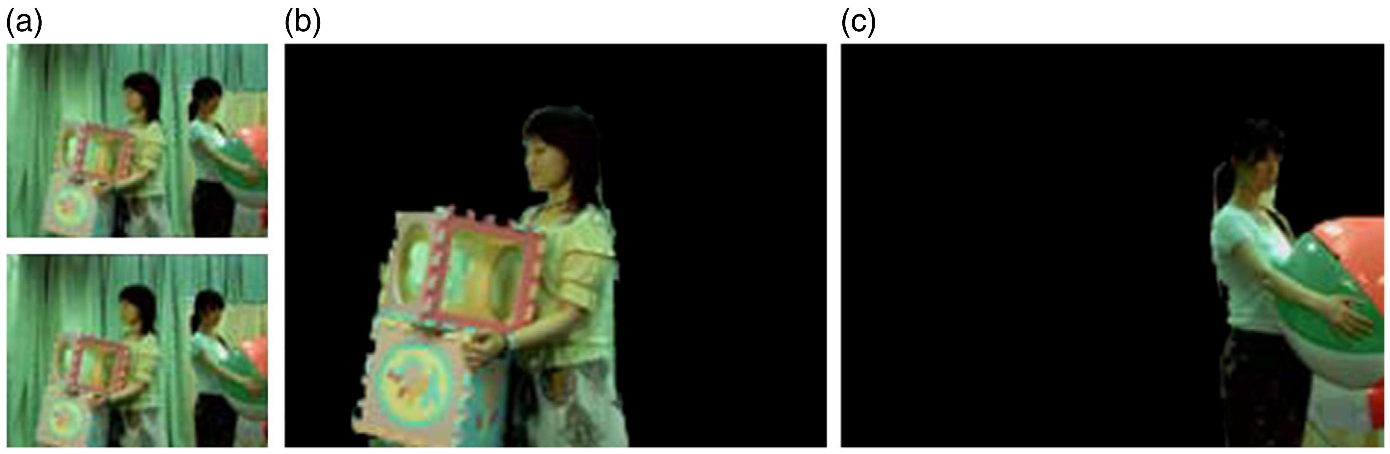


Fig. 20. Segmentation results for the "Akko&Kayo" video: (a) two frames (numbers 51 and 52) used for motion segmentation, and (b) and (c) the two girls segmented by the proposed method. The frame size is 640 × 480.

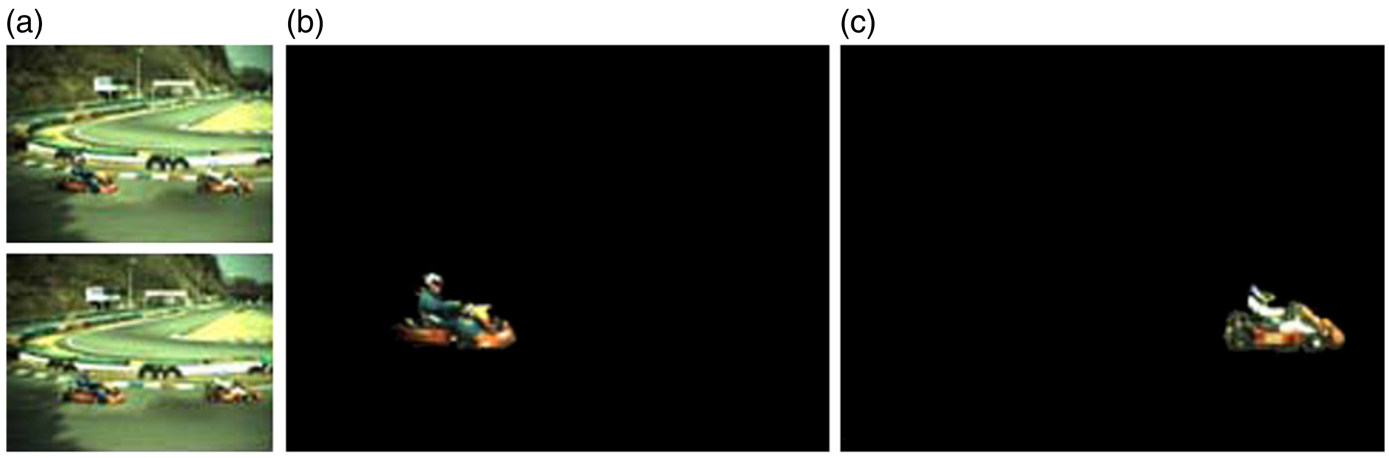


Fig. 21. Segmentation results for the "race" video: (a) two frames (numbers 30 and 31) used for motion segmentation, and (b) and (c) the two karts segmented by the proposed method. The frame size is 640 × 480.

for multiple objects, our method can still perform satisfactorily. However, in Fig. 21, the head of the right-hand kart driver is segmented into the background layer. This is due to the fact that the background and the helmet are both white. The color-based over-segmentation method cannot distinguish these two parts.

7.3. Object's 3D trajectories

In Section 7.1, to evaluate the performance of our 3D-trajectory estimation method, the 2D trajectory is assumed to be known. In this subsection, we will present the estimated 3D trajectory obtained

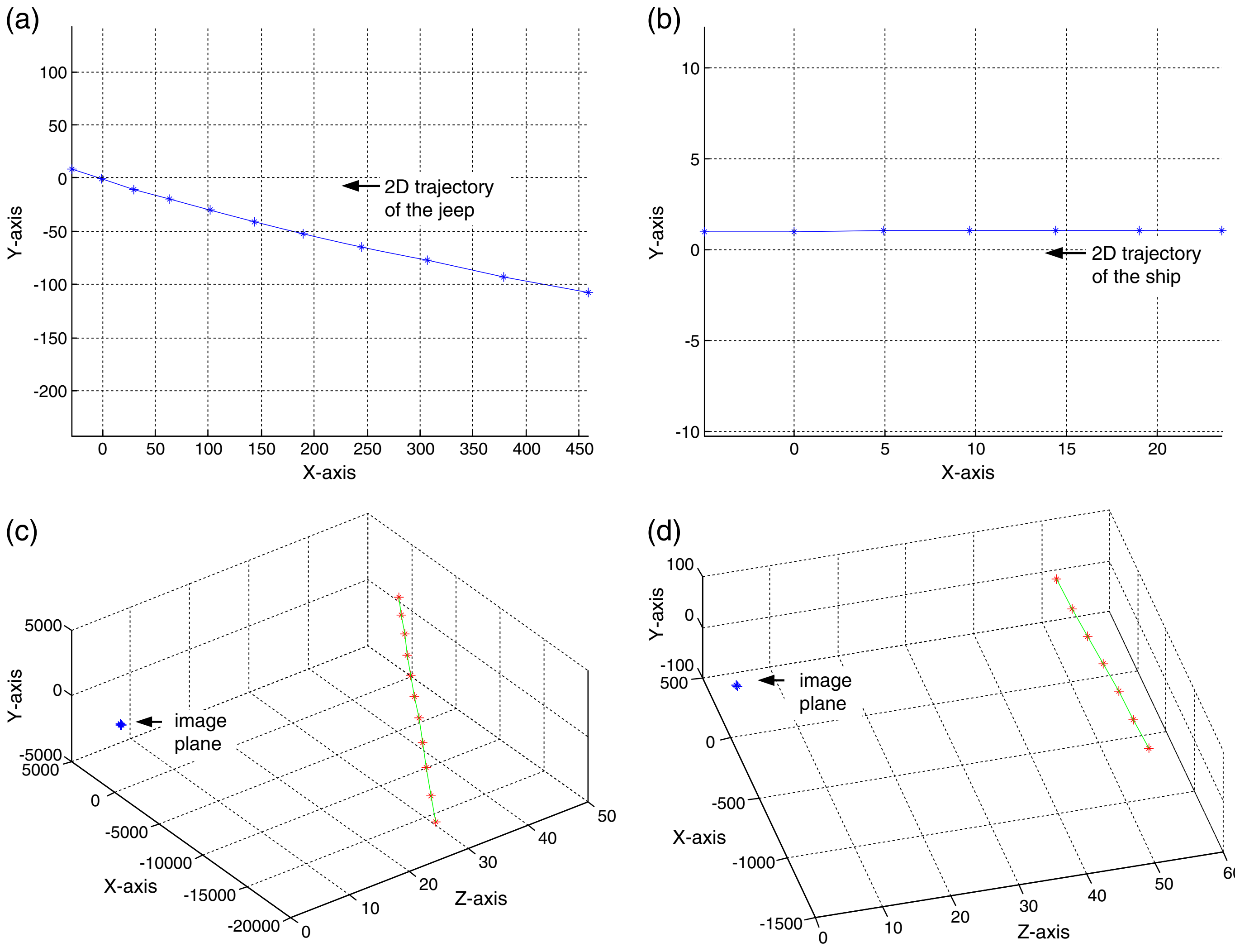


Fig. 22. The 2D trajectories of (a) the jeep and (b) the container in the image plan. The estimated 3D trajectories of (c) the jeep and (d) the container.

using our whole scheme. That means the 2D trajectory used in the estimation is estimated by the previous steps of our whole method. First, we will show the results of the "jeep" sequence, captured with a moving camera; and the "container" sequence, captured with a fixed camera. For the "jeep" sequence, as the camera is moving, we have to perform camera calibration. Then, we obtain the 2D trajectory of the jeep at the coordinates of the first frame (as shown in Fig. 22(a)). Fig. 22(c) shows the estimated 3D trajectory for the "jeep" sequence. Although the camera is moving, the proposed method can still achieve an accurate trajectory estimation. As the jeep moves along the road, the corresponding trajectory passes from the left to the right (see the *x* direction) of the camera, with its depth decreasing (see the *z* direction). Fig. 22(b) and (d) show the corresponding results for the "container" sequence. As the ship sails from left to right, the trajectory is nearly parallel to the image plane (see the *x*–*y* plane). From the two long-term trajectories of both sequences, we can see the stability and accuracy of the proposed method.

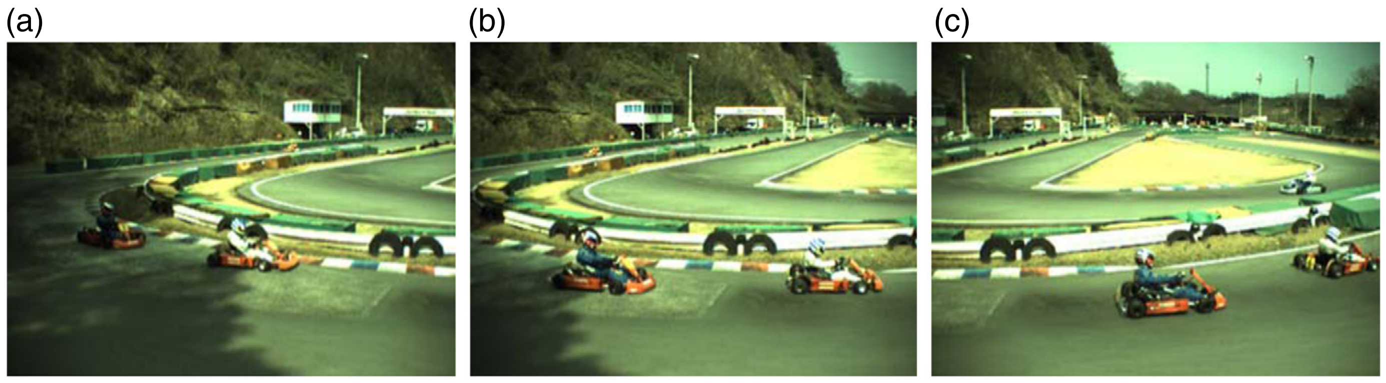
Our method can also handle the sequence of multiple objects with nonlinear motions (also captured with a moving camera). Fig. 23 shows the 3D trajectories of the "race" sequence. Consider the three consecutive frames shown in Fig. 23(a)–(c), two karts are running on a curved racetrack. The estimated 3D trajectories are illustrated in Fig. 23 (e). The *z*-axis presents the principal axis of the camera. As there is - little motion on the *y* axis (the vertical axis in the world coordinates), we only show the *x*–*z* plane of the 3D space. From Fig. 23, we can see that the shapes of the two 3D trajectories are correctly estimated; the difference in the curvature of the two trajectories can also be seen.

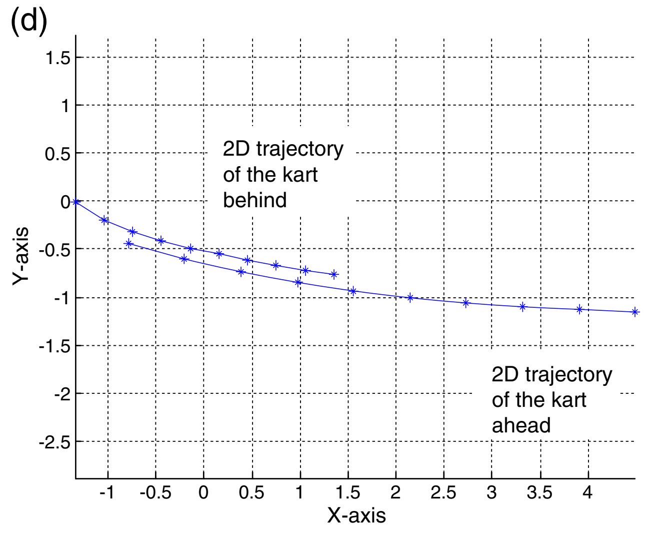
Table 2
The processing time of the proposed method.

	Feature motion estimation	Over-segmentation & Hierarchically structured algorithm	3D trajectory estimation & HMM-based refinement	Total estimated
Race (10 frames)	0.781 s	0.469 s	0.031 s	1.281 s
Container (10 frames)	0.234 s	0.313 s	0.016 s	0.563 s
Jeep (10 frames)	0.313 s	0.328 s	0.016 s	0.657 s

Note that our method cannot achieve accurate estimation when occlusion, objects splitting, or object merging happens. In these situations, the detected 2D motion of each object is no longer accurate, because the segment representing one object may change significantly in the successive frames. Failure also happens when handling articulated objects with different moving parts, like human bodies, because the motions of the different parts may not have a definite relation to the corresponding global 2D motion.

The runtimes required by our proposed method for ten frames of the three sequences: "race", "container", and "jeep" are tabulated in Table 2. The experiments were conducted on a 2.4-GHz Inter(R) Core™2 Duo computer with 2 Gbyte RAM using the unoptimized MATLAB 2006b for implementation. The MATLAB function "cputime",





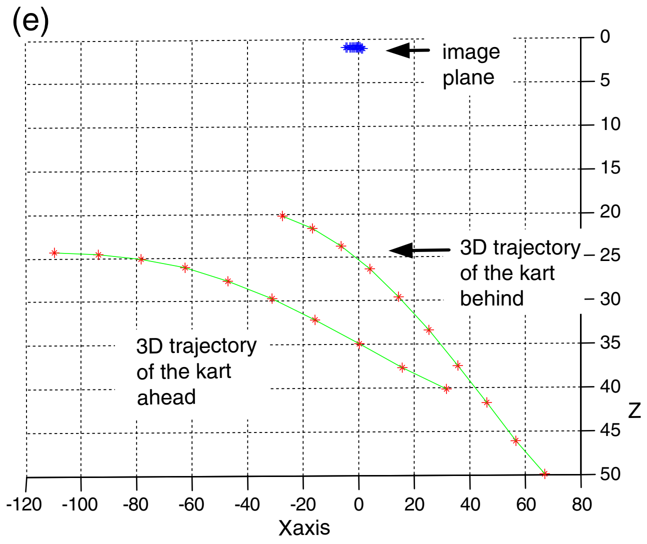


Fig. 23. The 3D trajectories of the karts: (a)–(c)Three frames (numbers 20, 30 and 40) in the sequence, (d) the 2D trajectories in the image plane, and (e) the 3D trajectories in the *x*-*z* plane.

which measures the CPU time in seconds for a process, is used to measure the processing time.

8. Conclusion

In this paper, we have presented an accurate approach to video-object segmentation and 3D-trajectory estimation of a dynamic scene from a monocular, uncalibrated view. In our proposed method, 2D feature motion estimation and image over-segmentation are achieved separately. Next, a hierarchically structured framework is developed to jointly use feature motion and over-segmentation, which can achieve an accurate segmentation result and object motion estimation. Then, a 3D-trajectory estimation method utilizing the local constant-velocity constraint is proposed for 2D-to-3D video conversion. Finally, a HMM-based method is devised, which integrates inter-frame information to calculate the trajectories of all the moving objects in the video frames.

Experiments have shown that our proposed framework can achieve a good performance in both motion segmentation and 3D-trajectory estimation for a dynamic scene from a monocular, uncalibrated view. This has a wide range of applications. In order to achieve fully automatic 2D-to-3D video conversion, some future work is needed. The 3D trajectories of an object can be estimated using our algorithm, but the exact 3D structure is still unknown. Therefore, if the depth information about a frame in a video shot is estimated using existing image depth estimation algorithms, the conversion can then be more automatic. With our framework, segmentation has been achieved and feature correspondences have been established, so the "shape from motion" (SFM) method can also be employed to handle the structure problem. In addition, our algorithm mainly utilizes the global motion of each segment. Local motions may also be considered for non-rigid body segmentation and trajectory estimation. This can result in a more accurate 3D reconstruction of video scenes.

Acknowledgement

The work described in this paper was supported by an internal grant from the Centre for Signal Processing, The Hong Kong Polytechnic University, Hong Kong, China (Project No. 1-BB9C), and the grants from the National Basic Research Project (No. 2010CB731800) and the Key Project of NSFC (No. 61035002).

References

[1] C.L. Wu, G.H. Er, X.D. Xie, T. Li, X. Cao, Q.H. Dai, A novel method for semi-automatic 2D to 3D video conversion, in 3DTV Conference, 2008, pp. 65–68.
[2] Z. Li, X.D. Xie, X.D. Liu, An efficient 2D to 3D video conversion method based on skeleton line tracking, in 3DTV Conference, 2009, pp. 1–4.
[3] R. Piroddi, T. Vlachos, A simple framework for spatio-temporal video segmentation and delayering using dense motion fields, IEEE Signal Processing Lett. 13 (7) (July 2006) 421–424.
[4] Z. He, Dynamic programming framework for automatic video object segmentation and vision-assisted video pre-processing, IEEE Proceedings-Vision, Image and Signal Processing 152 (5) (October 2005) 597–603.
[5] G.D. Borshukov, G. Bozdagi, Y. Altunbasak, A.M. Tekalp, Motion segmentation by multistage affine classification, IEEE Trans. Image Process. 6 (11) (November 1997) 1591–1594.
[6] D. Zhong, S. Chang, An integrated approach for content-based video object segmentation and retrieval, IEEE Trans. Circuits Syst. Video Technol. 9 (8) (December 1999) 1259–1268.
[7] J. Min, R. Kasturi, O. Camps, Extraction and temporal segmentation of multiple motion trajectories in human motion, Image Vis. Comput. 26 (12) (2008) 1621–1635.
[8] D. Wang, Unsupervised video segmentation based on watersheds and temporal tracking,, IEEE Trans. Circuits Syst. Video Technol. 8 (5) (May 1998) 539–546.
[9] V. Mezaris, I. Kompatsiaris, M.G. Strintzis, Video object segmentation using bayes-based temporal tracking and trajectory-based region merging, IEEE Trans. Circuits Syst. Video Technol. 14 (6) (June 2004) 782–795.
[10] T. Gevers, Robust segmentation and tracking of colored objects in video, IEEE Trans. Circuits Syst. Video Technol. 14 (6) (June 2004) 776–781.
[11] H.L. Li, K.N. Ngan, Face segmentation in head-and-shoulder video sequences based on facial saliency map, Proc. IEEE Int. Symp. Circuits Syst., Island of Kos, Greece, 2006, pp. 2681–2684.
[12] R.C. Verma, C. Schmid, K. Mikolajczyk, Face detection and tracking in a video by propagating detection probabilities, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (October 2003) 1215–1228.
[13] Y. Li, H. Ai, Y. Takayoshi, S. Lao, K. Masato, Tracking in low frame rate video: a cascade particle filter with discriminative observers of different life spans, IEEE Trans. Pattern Anal. Mach. Intell. 30 (10) (October 2008) 1728–1740.
[14] Y.P. Tsai, C.C. Lai, Y.P. Hung, Zen-Chng Shih, A Bayesian approach to video object segmentation via merging 3-D watershed volumes, IEEE Trans. Circuits Syst. Video Technol. 15 (1) (January 2005) 175–180.
[15] A. Briassouli, N. Ahuja, Integration of frequency and space for multiple motion estimation and shape-independent object segmentation, IEEE Trans. Circuits Syst. Video Technol. 18.no. 5 (May 2008) 657–669.
[16] W.Q. Wang, J. Yang, W. Gao, Modeling background and segmenting moving objects from compressed video, IEEE Trans. Circuits Syst. Video Technol. 18 (5) (May 2008) 670–681.
[17] R. Venkatesh Babu, K.R. Ramakrishnan, S.H. Srinivasan, Video object segmentation: a compressed domain approach, IEEE Trans. Circuits Syst. Video Technol. 14 (4) (April 2004) 462–474.
[18] T. Jebara, A. Azarbayejani, A. Pentland, 3D structure from 2D motion, IEEE Signal Process Mag. 16 (3) (May 1999) 66–84.
[19] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch, Visual modeling with a hand-held camera, Int. J. Comput. Vision 59 (3) (October 2004) 207–232.
[20] M. Lhuillier, L. Quan, A quasi-dense approach to surface reconstruction from uncalibrated images, IEEE Trans. Pattern Anal. Mach. Intell. 27 (3) (March 2005) 418–433.
[21] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int. J. Comput. Vision 47 (1-3) (April-June 2002) 7–42.
[22] R.J. Woodham, Photometric method for determining surface orientation from multiple images, Opt. Eng. 19 (1) (January/February 1980) 139–144.
[23] A. Georghiades, Recovering 3-D shape and reflectance from a small number of photographs, Eurographics Symposium on Rendering, 2003, pp. 230–240, Leuven, Belgium.
[24] N. Joshi, D. Kriegman, Shape from varying illumination and viewpoint, Proc. IEEE Int. Conf. Comput. Vis, 2007, pp. 1–7.
[25] S.K. Nayar, Y. Nakagawa, Shape from focus, IEEE Trans. Pattern Anal. Mach. Intell. 16 (8) (August 1994) 824–831.
[26] C. Yuan, G. Medioni, 3D Reconstruction of background and objects moving on ground plane viewed from a moving camera, Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognition, New York, USA, 2006, pp. 2261–2268.
[27] S. Avidan, A. Shashua, Trajectory triangulation: 3D reconstruction of moving points from a monocular image sequence, IEEE Trans. Pattern Anal. Mach. Intell. 22 (4) (2000) 348–357.
[28] M. Han, T. Kanade, Multiple motion scene reconstruction from uncalibrated views, Proc. IEEE Int. Conf. Comput. Vis., Vancouver, Canada, 2001, pp. 163–170.
[29] K.E. Ozden, K. Cornelis, L. Van Eycken, L. Van Gool, Reconstructing 3D independent motions using non-accidentalness, Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognition, Washington DC, USA, 2004, pp. 819–825.
[30] D. Lowe, Object recognition from local scale-invariant features, Proc. IEEE Int. Conf. Comput. Vis, 1999, pp. 1150–1157.
[31] J. Wills, S. Agarwal, S. Belongie, A Feature-based approach for dense segmentation and estimation of large disparity motion, Int. J. Comput. Vision 68 (2) (June 2006) 125–143.
[32] W. Förstner, E. Gülch, A fast operator for detection and precise location of distinct points, corners and centres of circular features, Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, 1987, pp. 281–305.
[33] J. Xiao, M. Shah, Motion layer extraction in the presence of occlusion using graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (October 2005) 1644–1659.
[34] K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (October 2005) 1615–1630.
[35] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vision 59 (2) (Sept. 2004) 167–181.
[36] M. Pollefeys, R. Koch, L. Van Gool, Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters, Proc. IEEE Int. Conf. Comput. Vis, 1998, pp. 90–95.
[37] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, CUP, Cambridge, 2000.
[38] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (February 1989) 257–286.
[39] Y. Wang, K.F. Loe, T. Tan, J.K. Wu, Spatiotemporal video segmentation based on graphical models, IEEE Trans. Image Process. 14 (7) (July 2005) 937–947.