Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Vision and Image Understanding 99 (2005) 1–57

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

A multi-Kalman filtering approach for video tracking of human-delineated objects in cluttered environments

Jean Gao*, Akio Kosaka, Avinash C. Kak

Robot Vision Lab, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA

Received 12 November 2002; accepted 27 October 2004
Available online 30 December 2004

Abstract

In this paper, we propose a new approach that uses a motion–estimation based framework for video tracking of objects in cluttered environments. Our approach is semi-automatic, in the sense that a human is called upon to delineate the boundary of the object to be tracked in the first frame of the image sequence. The approach presented requires no camera calibration; therefore it is not necessary that the camera be stationary. The heart of the approach lies in extracting features and estimating motion through multiple applications of Kalman filtering. The estimated motion is used to place constraints on where to seek feature correspondences; successful correspondences are subsequently used for Kalman-based recursive updating of the motion parameters. Associated with each feature is the frame number in which the feature makes its first appearance in an image sequence. All features that make first-time appearances in the same frame are grouped together for Kalman-based updating of motion parameters. Finally, in order to make the tracked object look visually familiar to the human observer, the system also makes its best attempt at extracting the boundary contour of the object—a difficult problem in its own right since self-occlusion created by any rotational motion of the tracked object would cause large sections of the boundary contour in the previous frame to disappear in the current frame. Boundary contour is estimated by projecting the previous-

* Corresponding author. Fax: +1 817 272 3784.
E-mail addresses: jgao@ecn.purdue.edu (J. Gao), kosaka@ecn.purdue.edu (A. Kosaka), kak@ecn.purdue.edu (A.C. Kak).

1077-3142/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2004.10.005

frame contour into the current frame for the purpose of creating neighborhoods in which to search for the true boundary in the current frame. Our approach has been tested on a wide variety of video sequences, some of which are shown in this paper.

Keywords: Tracking; Kalman filtering; Object tracking; Normalized cross-correlation; Perspective; Segmentation; Motion estimation; Recursive motion estimation; Feature extraction; Correspondence problem; Extended Kalman filtering; Boundary extraction; Region growing; Semi-automatic segmentation; Human-in-the-loop segmentation; Video surveillance; Video tracking

1. Introduction

Object tracking has received considerable attention during the past several years [28,40,17,16,23,6,30,36,38]. Applications of object tracking can be found in areas as diverse as video editing for publishing and entertainment, video surveillance, object-based coding for MPEG-4, query formation for MPEG-7, etc.

The approaches suggested so far for object tracking can be classified into automatic and semi-automatic categories. The fully automatic approaches, such as those proposed by [27,19,15,5,21], work mostly for simple objects executing simple motions against clutter-free backgrounds. An example would be a bright light source moving against a uniform dark background. Tracking under such conditions is relatively easy for the obvious reason that the object can be trivially segmented from the background. Automatic methods are not the focus of this paper, since we are specifically interested in complex objects executing complex motions against cluttered backgrounds.

The semi-automatic methods are all based on the rationale that if the human could help out with the initial segmentation of the object to be tracked, the computer could then be relied upon to track the extracted form in subsequent frames. This rationale underlies the many contributions in the semi-automatic category. The published literature on such semi-automatic methods uses two different approaches for motion estimation. While some researchers, such as [6,23,28], perform motion estimation by establishing feature correspondences between the frames of a video sequence, others do motion estimation by first calculating optical flows.

The optical-flow based and the feature-correspondence based methods for motion estimation have their own advantages and disadvantages. Optical flow based methods theoretically treat tracking as a segmentation of the flow field and group together the optical flow vectors that exhibit the same motion [34,41,1]. The tracking performance of these methods depends on the accuracy of the estimated motion field which is error-prone in the vicinity of intensity discontinuities in images [3,41,18,11]. Additionally, tracking can often require dense optical flow fields, which may result in burdensome computations.

With regard to the feature-correspondence based approaches for motion estimation, it is possible to use color, texture, contour, edge, illuminance, etc., for establishing correspondences between successive frames and to then determine the motion model parameters that best fit the entire set of observations [23,40,28,6].

There are, obviously, two issues here to deal with: the accuracy of feature correspondences, and the accuracy of motion estimates as obtained from the feature correspondences. The problem of feature correspondence is, in general, ill-posed due to the presence of either multiple candidates within a search region or no candidates because of occlusion and other factors. Using various assumptions—such as the corresponding features in two consecutive frames must be each other's nearest neighbors when one frame is projected into the other—several approaches have been proposed for establishing feature correspondences [43,10,45]. For example, Weng et al. [43] have proposed a multi-attribute image matching method that creates a vector of attributes for each pixel—in other words, every pixel becomes an image feature—and then seeks matches on the basis of the similarity of these vectors. Since this process is carried out at every pixel, the result is a dense motion map. For rigid motion, the attributes included in the vector are the intensity, the edgeness, and the cornerness. For another contribution related to feature correspondences, Cox [10] has surveyed different statistical data association methods, such as nearest neighbor, tracking-split, joint likelihood algorithms, etc., as strategies for establishing the correspondences.

After feature correspondences are established, motion estimation can be carried out using either a non-recursive approach or a recursive approach. Whereas the former processes all the correspondences all at once for the motion estimate, the latter processes one correspondence at a time that is then recursively updated by processing the next correspondence. It was shown by Tekalp [39] and Alon and Sclaroff [8] that non-recursive estimators are more stable and converge faster than recursive estimators especially when the motion model used is non-linear. On the other hand, recursive estimators tend to be computationally simpler, which can be a most important consideration for real-time motion estimation.

With regard to recursive methods, various authors have shown motion tracking results using the Kalman filter—more precisely, the extended Kalman filter (EKF). For some of the more prominent examples, see [7,23,29,8,45,20,9,46,47]. By using different motion and/or camera models, or by applying different constraints, the authors have shown different formulations of the EKF in the context of motion estimation. All of these contributions include a feature extraction stage, and the accuracy of the EKF algorithms depends considerably on the feature detection method used.

The work described in this paper is based on the feature-correspondence approach. Feature correspondence in our work takes place inside a feedback loop that projects uncertainties into the images to place bounds on where to look for a feature. Motion estimation then becomes an automatic by-product of the feature correspondence step.

Unlike previous work, we do not decompose the feature-correspondence based approach into two disjoint sub-problems—first establishing the feature correspondences and then estimating motions. On the other hand, we use a single integrated approach that uses feedback loops to simultaneously establish the feature correspondences and to perform motion estimation.

Many of the previously cited contributions on semi-automatic tracking use affine motion models, which implies (implicitly) that those methods are limited to the tracking of planar surface patches. Our work makes no such assumptions; the motions are allowed to be arbitrary rigid-body motions. Our framework is formulated to recover from the monocular image sequences the relative depth values of the features used for tracking at any given time. Furthermore, our system works for any previously recorded video because no camera calibration is needed [44]. Additionally, no restriction is applied to either the camera or the background movement.

Feature correspondences in our approach are established via normalized cross-correlation within bounded regions defined by the uncertainties associated with the estimated motion vector. A candidate feature X in the current frame is accepted as a correspondent for a feature Y in a previous frame provided the motion vector that incorporates the X-to-Y correspondence says that X is within one unit of Mahanalobis distance of Y. The problem of occlusion is taken care of by keeping track of the *lifetime* of each feature—from the frame in which it makes its first appearance to the frame in which it eventually disappears due to occlusion. The beginning frame for a feature is called its *genesis frame*. For motion estimation in the current frame, all of the features with the same genesis index are grouped together and used jointly for Kalman-based updating of the motion vector.

Since boundary contours play a critical role in humans' perception of objects, our system also makes a best attempt at extracting the object boundary during the tracking process. But, as is well known, when objects are allowed to undergo rotational motions or when the view angle changes significantly, the problem of boundary extraction is exceedingly difficult. For a best-attempt solution, our system carries out a previous-frame to next-frame projection of the bounding contour in order to create neighborhoods in which to search for the true boundary. The search process presented in this paper uses a combination of growing and shrinking of the inner and the outer bounds of such neighborhoods for the discovery of the real bounding contours. The procedure is not fool-proof.

The paper is organized as follows. Section 2 gives an overall description of our system. Then Section 3 presents feature correspondence matching through feature uncertainty modeling, prediction, and extraction. How Kalman filter is initially applied to motion estimation, and how motion estimation feedback is used in establishing feature correspondence are described in Section 4. Section 5 elaborates our multi-frame based motion estimation method. How motion estimation is achieved when some of the features get lost due to self-occlusion will also be described in Section 5. Section 6 discusses the topic of boundary extraction of an object that is being tracked. In Section 7, we test the accuracy of our approach, first with synthetic data, and then with real video sequences. Section 7 also presents the result of a comparison between our approach and a traditional implementation of Kalman filter based object tracking in which one simply uses correlation based feature selection, but no motion estimation feedback to improve on feature correspondences.

2. The motion tracking framework—an overview

As we mentioned earlier, ours is a semi-automatic approach to the tracking of objects in video sequences. To get around the difficulty of segmentation, the human must specify what to track in the first image of the sequence. Specification of what to track obviously calls for a graphical interaction module that is not demanding on the user. In our system, this problem was solved with a specially designed image editor module, called the color-interactive segmentation editor (CISE) that is described elsewhere [14]. The editor calls for minimal user interaction for specifying semantically significant boundaries. The degree of human interaction is reduced by giving the user tools to delete/merge the regions produced by a split-and-merge segmentor so that the final boundaries are appropriate for the object to be tracked. The editor incorporates smart heuristics for edge-linking, together with boundary smoothing through an energy-minimization algorithm. The role of the editor is made clear in the overview schematic of Fig. 1. The box labeled "Object Initial Definition" stands for the user using CISE to specify a Region of Interest for tracking. Since the focus of this paper is specifically on Kalman filtering for tracking, we will not delve any further into the details of CISE. The reader is referred to [14] for further details.

The rest of the flow diagram of Fig. 1 deals with the tracking of 3D objects. As implied by the figure, the tracking process begins with an automatic selection of feature points inside and on the boundary of the object, this function being performed by the *Feature Selection* module. As we will explain later, the boundary points are not used for tracking—since their visibility cannot be relied upon when objects are rotating—but only for maintaining an internal representation of the object being tracked. The interior points, on the other hand, are the ones that are tracked.

The interior points form a second internal representation of the object, this one purely for the purpose of tracking. We will show in the next section that all of the interior points taken together can be used to construct a motion vector (MV) and a shape vector (SV) that are subject to Kalman filtering for motion and pose prediction.

For the initially registered interior feature points, we keep tracking them until they disappear due to self-occlusion or feature mis-match, and in the meantime, new features are registered as time goes on. The basic tool for establishing a feature correspondence is the normalized-cross-correlation (*NCC*) within a neighborhood bounded by the Kalman-predicted motion of the object. The resulting feature correspondences are used to update the motion vector associated with the object. The updated motion vector is then used to test again the appropriateness of the feature

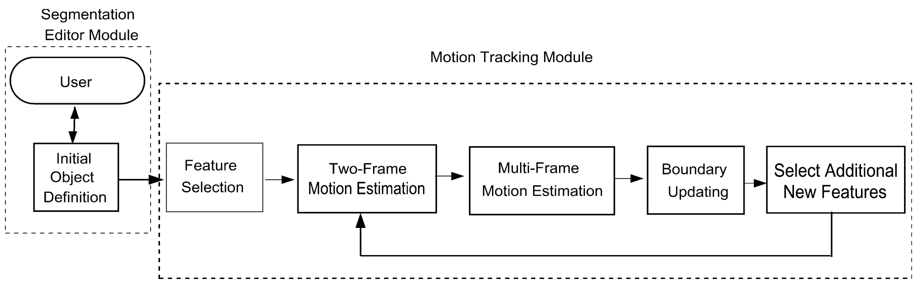

Segmentation Editor Module
User
Initial Object Definition
Motion Tracking Module
Feature Selection
Two-Frame Motion Estimation
Multi-Frame Motion Estimation
Boundary Updating
Select Additional New Features

Fig. 1. Overall architecture for motion tracking.

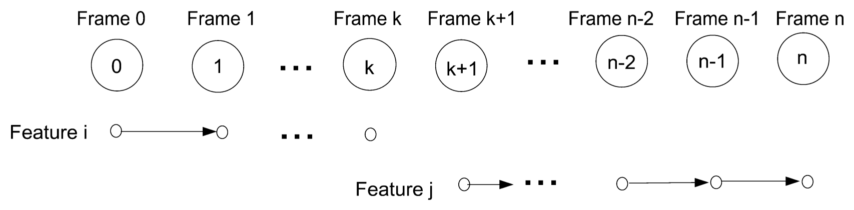


Fig. 2. Interior feature points flow, where feature i starts in frame 0 and disappears in frame $k+1$, and feature j starts in frame $k+1$ and is still active in current frame n.

correspondences, or to predict new neighborhoods in which to seek the corresponding features points.

Fig. 2 depicts a feature i that initially appears in frame 0, but disappears in frame $k+1$. Also depicted is feature j that first appears in frame $k+1$ and is still active in the current frame n. Frame 0 is the genesis frame for feature i and frame $k+1$ is the genesis frame for feature j. All of the features in the current frame are grouped on the basis of their individual genesis frame index. The *Two-Frame Motion Estimation* module carries out motion estimation using groups of features, all the features in the same group having the same genesis frame. Therefore, if a group of features in the current frame n has a genesis index k, the *Two-Frame Motion Estimation* module will update the motion vector of the object by applying the Kalman filter to the frame-k and frame-n data.[1]

While the *Two-frame Motion Estimation* module updates the motion vector in the current frame on a feature-grouping by feature-grouping basis, each grouping characterized with the same genesis index, we also need a mechanism to combine all of these updates into an overall single update for the motion vector of the object. This is accomplished by the *Multi-Frame Motion Estimation* module shown in Fig. 1.

Subsequently, the *Boundary Updating* module makes a best effort for delineating the boundary of the object. It is critical to note here that the output of this module only secondarily affects the overall tracking performance of our system. As was mentioned earlier, the boundary points are not used directly for tracking, since they are highly susceptible to occlusion in the presence of object rotations. Yet, humans witnessing the tracker like to see the boundary of the object being tracked. Hence this module.

The behavior of the *Boundary Updating* module affects the tracker in the sense that it circumscribes the region in which the system should look for additional new feature points if too many of the old disappear on account of occlusion.

[1] Note that this module carries out motion estimation *not* just from the last frame to the current frame (unlike many other implementations of Kalman filtering for tracking), but from the frame in which a feature first makes its appearance to the current frame. We believe that this approach results in more robust motion estimation. When a feature first pops up in a new frame, the system associates by default a large value of uncertainty with the feature. Subsequently, as this feature is tracked from frame to frame, the Kalman filter reduces that initial uncertainty.

The selection of the additional feature points when needed is carried out by the *Select Additional New Features* module shown in the figure.

3. Extraction of feature points, their representations, and uncertainty modeling

We will address in this section the core issue of how we find correspondences in frame n for the new features discovered in frame k, $k < n$. Fundamental to this process is the estimation of motion from frame k to frame n and the calculation of motion uncertainties in frame n. The uncertainty parameters in frame n will be obtained by a two-step procedure: prediction based on the uncertainty values in frame $n - 1$ and update based on the frame-k to frame-n correspondences deemed appropriate in frame n. So, on the one hand, the correspondences established will help us update the motion uncertainties, and, on the other, the estimated motion uncertainties will help us determine the correspondences. While on the face of it, this sounds circular, but, as we will show, when motion between successive frames is small, this logic yields good estimates of motion. Section 5 will then show how to use the frame-k to frame-n motion estimation framework of Section 4 to estimate motions in frame-n using all of the genesis frames. The frame-k to frame-n correspondence problem addressed in this section presents motion estimation formulas using the m interior feature points that make their first appearance in frame k, the current frame being frame n.

3.1. Automatic selection of feature points for tracking and for boundary description

Before we talk about feature correspondence for a group of features with genesis index k from frame-k to frame-n, we will first show how feature points are selected automatically in the very first frame at the beginning of tracking process. This corresponds to the *Feature Selection* module in Fig. 1. But before we do so, we need to mention that object feature points serve two distinct purposes in our work: (1) Those that are on the boundary of the object are used for maintaining a description of the object being tracked; and (2) Those that are interior to the object, in the sense of not being on its boundary as projected on the camera plane, are used for actual tracking. It should be obvious that in the presence of both translations and rotations, especially the latter, it would not be wise to use the boundary points for tracking, since the rotation would cause some of them to become invisible to the camera. However, since we do need to maintain a representation of what is being tracked, the boundary points are still important.

We will now describe how the feature points are chosen automatically in the first frame after a human has delineated an ROI. In what follows, we will first describe how the boundary points are chosen by the system. Next, we provide a description of how the interior points are chosen for tracking.

Boundary point selection is based on marking an initial point on the boundary by raster scanning the ROI and choosing for the first boundary point the first intersection of a raster line with the human-delineated boundary. The other points are extracted by the following recursive procedure:

Let the current point be denoted P_i. Now

1. Starting from P_i, visit each consecutive point on the ROI boundary. Let the current point be P'. Construct a chord from P_i to each P' and determine the maximal perpendicular distance between the chord and the boundary between P_i and P'.
2. If the perpendicular distance exceeds a threshold, make P' the next P_i and go back to step 1.
3. If P' goes past the original starting point, stop.

We denote the set of M boundary points thus extracted by $\mathscr{B} = \{\vec{b}_0, \ldots, \vec{b}_{M-1}\}$. Fig. 3A is a pictorial depiction of the algorithm. Of course, to achieve minimum boundary approximation error, a user can use pixel-wise representation of ROI boundary, i.e., select every single point on ROI boundary by restricting the maximum Euclidean distance between adjacent points to be 1.

The interior feature points are obtained by first calculating the edge maps of the R, G, B components of ROI by Canny edge detector. Next, an overall edge map is obtained by taking a logic OR of the three edge maps. Feature points are then selected by sub-sampling the edge points in the overall edge map. Let $\mathscr{E} = \{\vec{q}_0, \ldots, \vec{q}_{N-1}\}$ represent the set of selected interior feature points. Fig. 3B is the original image and Fig. 3C is the overall edge map. Fig. 3D shows the human-

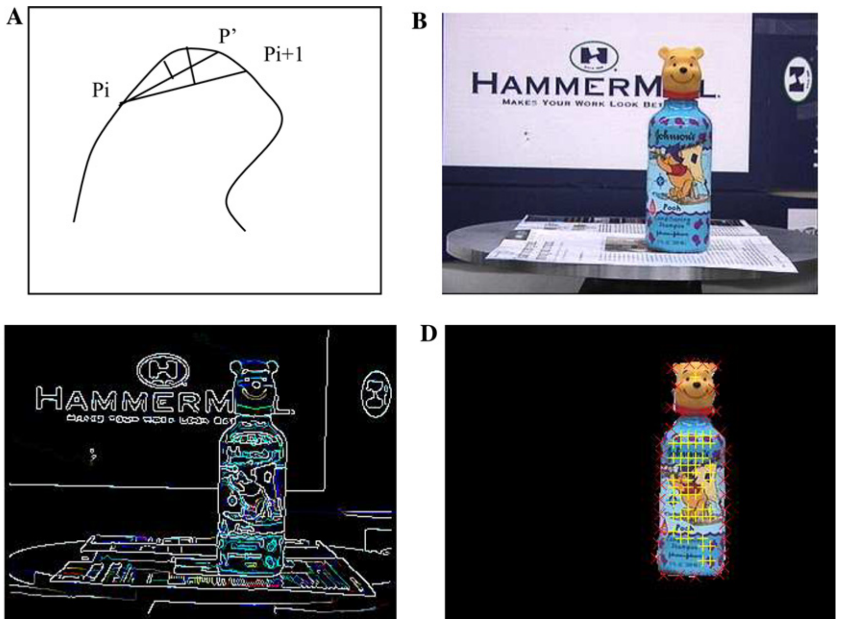


Fig. 3. (A) Selection of boundary points by thresholding chord lengths. (B) First image in the video sequence. (C) Edge map of the first image. (D) Segmented object region and selected feature points. The interior points are marked + and the boundary points ×.

delineated object using CISE. Within Fig. 3D, the "×"s are the selected boundary points, and the "+"s are the interior feature points.

3.2. Object representation for tracking and uncertainty modeling

In this subsection, we will show how the rigid-body motion constraint can be used to specify a unique representation for the interior points that works well for tracking. This representation consists of two vectors: a shape vector and a motion vector.

Using a pinhole camera model for a monocular sequence of images, we will assume that the center of projection coincides with the origin of the world coordinates and the XY plane of the world coordinates is parallel to the uv image plane. The relationship between an object point (X, Y, Z) in the world coordinates and its corresponding point $(\tilde{u}, \tilde{v})$ in the image plane is described by the following perspective transformation in homogeneous coordinates [12]:

$$\begin{bmatrix} \tilde{u}w \\ \tilde{v}w \\ w \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1}$$

where (u_0, v_0), α_u and α_v are the intrinsic parameters of the camera. Note that the $\tilde{u}$ axis is parallel to the X axis, and the $\tilde{v}$ axis is parallel to the Y axis.

By normalizing $(\tilde{u}, \tilde{v})$ using u_0, v_0, α_u, and α_v, i.e.,

$$u = \frac{\tilde{u} - u_0}{\alpha_u}, \quad v = \frac{\tilde{v} - v_0}{\alpha_v} \tag{2}$$

Eq. (1) becomes

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{3}$$

When the object is in motion and the transformation between the frame k and frame n is specified by the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{T}$, then

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{T}, \tag{4}$$

where (X', Y', Z') represent the new coordinates in the world coordinate frame, $\mathbf{R}$ is the rotation matrix with elements r_{ij}, $i, j = 0, 1, 2$ defined by rotation angles ϕ_x, ϕ_y, and ϕ_z with respect to X, Y, and Z axes and $\mathbf{T}$ is the translation vector defined by t_x, t_y, and t_z. We define the homogeneous transformation $\mathbf{H}$ as

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5}$$

If we observe this point (X', Y', Z') in the normalized image coordinates, then

$$Z' \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = Z\mathbf{R} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} + \mathbf{T}, \tag{6}$$

where (u', v') represents the image coordinates in the normalized camera image frame at time n. The above equation can be rewritten in the form of the following constraint that must be satisfied by each feature point i:

$$\mathbf{f}_i = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \mathbf{0}, \tag{7}$$

which is the same as saying

$$\mathbf{f}_i = \begin{bmatrix} u' - \frac{r_{11}u + r_{12}v + r_{13} + \frac{t_x}{Z}}{r_{31}u + r_{32}v + r_{33} + \frac{t_z}{Z}} \\ v' - \frac{r_{21}u + r_{22}v + r_{23}) + \frac{t_y}{Z}}{r_{31}u + r_{32}v + r_{33}) + \frac{t_z}{Z}} \end{bmatrix} = \mathbf{0}. \tag{8}$$

Combining the rotation angles, translation vector, and depth values, we now define a motion vector (MV) ${}_n\mathbf{p}_k$ and a shape vector (SV) $\mathbf{w}$ as

$${}_n\mathbf{p}_k = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}, \tag{9}$$

$$\mathbf{w} = (Z_0, Z_1, \ldots, Z_{m-1})^{\mathrm{T}}, \tag{10}$$

where m is the number of feature points first appeared in frame k. The vector ${}_n\mathbf{p}_k$ encodes the motion-induced change in the pose of the object from frame k to frame n. Central to our motion tracking scheme is the estimation of this vector. Since this vector is inherently a random entity, for the purpose of estimation it will be represented by its mean vector ${}_n\bar{\mathbf{p}}_k$ and by its covariance matrix ${}_n\mathbf{\Sigma}_k$. We represent the motion uncertainty from frames k to n as ${}_n\mathbf{U}_k : ({}_n\bar{\mathbf{p}}_k, {}_n\mathbf{\Sigma}_k)$.

While the motion vector represents a global property of the entire object that is being tracked, the shape vector is a collection of depth attributes, each local to a feature point. This calls for a different way of representing the uncertainty in the shape vector. Each element of the shape vector will be represented separately by its mean $\bar{Z}_i$ and its standard deviation σ_{Z_i}.

Without loss of generality, here we assume that the uncertainties in both the motion vector and the shape vector are Gaussian, implying that the means and the covariances are sufficient for their representation. In the context of uncertainty modeling, there is a respectable tradition for this assumption in computer vision, in general [37,7,8], and in vision-based motion tracking in particular [29].[2]

[2] As stated in [12], the Gaussian assumption is not necessary for Kalman based estimation to be optimum when the system measurements are linear functions of the state variables. This linearity assumption does not strictly apply in our case since our state equations are fundamentally non-linear, although we do linearize in the vicinity of the operating point for Kalman estimation.

3.3. Feature prediction and finding correspondences

The solution to the frame-k to frame-n correspondence problem is illustrated by part of the block diagram in Fig. 4, which is an expansion of the single block labeled *Two-Frame Motion Estimation* module of Fig. 1.

In Fig. 4, the prediction is carried out by the *Motion* ${}_n\boldsymbol{U}_k$ *Prediction* module that predicts the mean ${}_n\bar{\mathbf{p}}_k$ and error covariance matrix ${}_n\boldsymbol{\Sigma}_k$ of the motion vector (MV) associated with the m points using the frames k and n. The motion vector prediction is used by the *Feature Prediction and Extraction* module to first predict the locations of the m features points in the current frame n; to surround these locations with appropriate uncertainty regions; and, finally, to extract from the uncertainty regions thus delineated the new locations of the m feature points by using normalized cross-correlation (NCC). Within each uncertainty region, the system retains the feature point with the highest value yielded by NCC. If this highest value does not exceed a preset threshold, the feature point is discarded, in which case a given feature point in frame k may not possess a corresponding point in frame n. In the following paragraphs, we will technically present how the above steps are carried out.

3.3.1. Motion uncertainty ${}_n\boldsymbol{U}_k$ prediction

As defined already, ${}_n\mathbf{U}_k$ is the pair $({}_n\bar{\mathbf{p}}_k, {}_n\boldsymbol{\Sigma}_k)$, the first element of which is the mean value of the motion vector, meaning the mean change in the pose of the object from frame k to frame n, and the second element the covariance associated with this pose change. Given the m feature points in frame k, we want to estimate ${}_n\mathbf{U}_k$. What that means is that we want to use the m features of frame k and their corresponding points in frame n to estimate the new mean and the new covariance of the motion vector in frame n.

The estimate ${}_n\mathbf{U}_k$ is formed by first making its prediction ${}_n\mathbf{U}^{-}_k$, then using the predicted motion to find frame-n correspondences for the m features of frame-k, and finally updating the prediction into the desired estimate based on the correspondences.

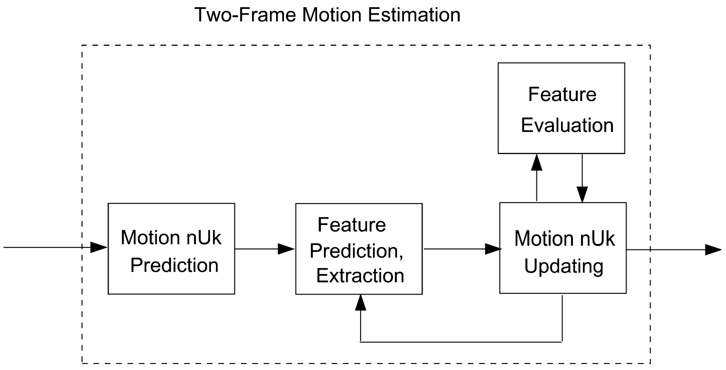


Fig. 4. Feature correspondence and motion estimation in frame-n for a group of feature points born in frame-k.

The prediction ${}_{n}\mathbf{U}^{\sim}{}_{k}$ can be obtained by using the previously computed estimate ${}_{n-1}\mathbf{U}_{k}$ and a reasonable initial guess about the motion uncertainty ${}_{n}\mathbf{U}_{n-1}$. The predicted (meaning guessed) pose change statistics from frame $n-1$ to n are given by ${}_{n}\mathbf{U}^{\sim}{}_{n-1} : ({}_{n}\bar{\mathbf{p}}^{\sim}{}_{n-1}, {}_{n}\boldsymbol{\Sigma}^{\sim}{}_{n-1})$ and the already computed estimate for frame k to frame $n-1$ by ${}_{n-1}\mathbf{U}_{k} : ({}_{n-1}\bar{\mathbf{p}}_{k}, {}_{n-1}\boldsymbol{\Sigma}_{k})$. Let ${}_{n}\mathbf{H}^{\sim}{}_{n-1}$ be the predicted homogeneous transformation from frame $n-1$ to frame n, and ${}_{n-1}\mathbf{H}_{k}$ be the one from frame k to frame $n-1$. We can therefore write the predicted homogeneous transform ${}_{n}\mathbf{H}^{\sim}{}_{k}$ from frame k to frame n as

$$ {}_{n}\mathbf{H}^{\sim}{}_{k} = {}_{n}\mathbf{H}^{\sim}{}_{n-1} * {}_{n-1}\mathbf{H}_{k}. \tag{11} $$

Based on the assumption that motion between two adjacent frames is small and bounded by some threshold, we can initialize ${}_{n}\bar{\mathbf{p}}^{\sim}{}_{n-1}$ by

$$ {}_{n}\bar{\mathbf{p}}^{\sim}{}_{n-1} = [0,0,0,0,0,0]^{\mathrm{T}}, \tag{12} $$

which says that mean change in pose between from frame $n-1$ to frame n can be expected to be zero for the purpose of initialization of the motion vector.

Again for the purpose of initialization, for the guessed error covariance matrix ${}_{n}\boldsymbol{\Sigma}^{\sim}{}_{n-1}$ we assume that the small standard deviation associated with the pose change from frame $n-1$ to frame n is the same for all three translational dimensions and is represented by σ_t. We make a similar assumption for the rotational dimensions of pose and represent that uncertainty by σ_ϕ. We will also assume that the uncertainties along the different dimensions are uncorrelated for the purpose of initialization. This implies the following diagonal structure for ${}_{n}\boldsymbol{\Sigma}^{\sim}{}_{n-1}$:

$$ {}_{n}\boldsymbol{\Sigma}^{\sim}{}_{n-1} = \begin{bmatrix} \sigma_\phi^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\phi^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_t^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_t^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_t^2 \end{bmatrix}. \tag{13} $$

The predicted mean value of motion vector ${}_{n}\bar{\mathbf{p}}^{\sim}{}_{k}$ can be obtained by directly expanding of Eq. (11). But the calculation for the predicted covariance matrix ${}_{n}\boldsymbol{\Sigma}^{\sim}{}_{k}$ is not straightforward. The mathematical details are presented in Appendix A.

For the video sequences we have experimented with, we have typically used 2° for σ_ϕ and $0.02m$ for σ_t. Obviously, the values chosen for these two standard deviations would depend on the nature of motion. If the motion between two consecutive frames is large, these values would need to reflect that.

3.3.2. Projecting predicted motion uncertainty into image space

So far we have talked about motion uncertainty modeling and prediction. Given the predicted motion uncertainty ${}_{n}\mathbf{U}^{\sim}{}_{k}$ in the current frame n, we will now discuss how this prediction can be used to place bounds on feature location uncertainties in the current frame n. This we will do by projecting the predicted motion uncer-

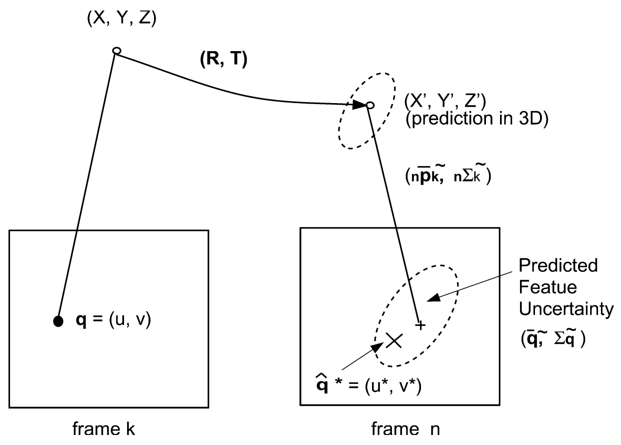

(X, Y, Z)
(R, T)
(X', Y', Z')
(prediction in 3D)
$({}_n\bar{\mathbf{p}}\tilde{}_k, {}_n\Sigma\tilde{}_k)$
Predicted
Featue
Uncertainty
$(\bar{\mathbf{q}}\tilde{}, \Sigma_{\tilde{\mathbf{q}}})$
q = (u, v)
$\hat{\mathbf{q}}$ * = (u*, v*)
frame k
frame n

Fig. 5. Uncertainty prediction for feature extraction.

tainty into the image frame and then using the uncertainty bounds in the image space for locating the correspondents of the feature points in frame k.

Calculation of the predicted feature location uncertainty in the current frame n is carried out by projecting $({}_n\bar{\mathbf{p}}\tilde{}_k, {}_n\mathbf{\Sigma}\tilde{}_k)$ into the image frame. More specifically, for each feature point $\mathbf{q}_i = (u_i, v_i)^{\mathrm{T}}$ in the image frame k, we have the prediction mean $\bar{\mathbf{q}}\tilde{}_i$ and prediction error covariance $\mathbf{\Sigma}_{\mathbf{q}\tilde{}_i}$ in the current frame n, which are obtained by:

$$\bar{\mathbf{q}}\tilde{}_i = \begin{bmatrix} \frac{\bar{Z}_i(r_{11}u_i+r_{12}v_i+r_{13})+t_x}{\bar{Z}_i(r_{31}u_i+r_{32}v_i+r_{33})+t_z} \\ \frac{\bar{Z}_i(r_{21}u_i+r_{22}v_i+r_{23})+t_y}{\bar{Z}_i(r_{31}u_i+r_{32}v_i+r_{33})+t_z} \end{bmatrix}, \tag{14}$$

$$\mathbf{\Sigma}_{\mathbf{q}\tilde{}_i} = E\{(\mathbf{q}\tilde{}_\mathbf{i} - \bar{\mathbf{q}}\tilde{}_\mathbf{i})(\mathbf{q}\tilde{}_\mathbf{i} - \bar{\mathbf{q}}\tilde{}_\mathbf{i})^{\mathrm{T}}\}, \tag{15}$$

$$= \left.\frac{\partial \mathbf{q}\tilde{}_i}{\partial({}_n\mathbf{p}_k)}\right|_{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}\tilde{}_k} * {}_n\mathbf{\Sigma}\tilde{}_k * \left[\left.\frac{\partial \mathbf{q}\tilde{}}{\partial({}_n\mathbf{p}_k)}\right|_{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}\tilde{}_k}\right]^{\mathrm{T}}, \tag{16}$$

$$= \left.\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}\right|_{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}\tilde{}_k} * {}_n\mathbf{\Sigma}\tilde{}_k * \left[\left.\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}\right|_{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}\tilde{}_k}\right]^{\mathrm{T}}. \tag{17}$$

When all pose random variables are in the vicinity of their means, the derivative matrix $\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}$ as computed at the expected values of the random variables is sufficient to convert the motion uncertainty into feature uncertainty in the image plane [29,37].

Fig. 5 shows the propagation of motion uncertainty into feature location uncertainty. In Fig. 5, the "+" at frame n represents the predicted mean value, $\bar{\mathbf{q}}\tilde{}$, and the ellipse represents the uncertainty region defined by $\mathbf{\Sigma}_{\mathbf{q}\tilde{}}$. This propagation greatly helps the feature extraction by reducing the search space for feature correspondence.

3.3.3. Feature extraction using predicted uncertainty

Feature extraction is done by template matching within the predicted uncertainty regions of the image space as bounded by the parameters $\bar{\mathbf{q}}\tilde{}$ and $\mathbf{\Sigma}_{\mathbf{q}\tilde{}}$. We search for the largest peak after we have correlated this uncertainty region with a 15×15 template, modified as described below, that defines the feature in frame k. The output of the correlator is subject to non-maximum suppression to sharpen up the peaks. We refer to the output of the correlator as the Normalized Cross Correlation (NCC) map of the predicted uncertainty region.

To elaborate, extraction of candidates in the current frame n for a particular feature in frame k consists of the following steps:

- Optimize in frame k the 15×15 template window used for NCC calculation by deleting pixels that are not on the object being tracked. As we mentioned earlier, in each frame we keep track of the boundary pixels that are used to modify the template in this manner. This is important for correlation based tracking in the presence of motion because of the ever-changing background. In Fig. 6A, the white polygons show the trimmed templates for the different feature points A and B represented by "+".
- Compute the NCC values within the predicted uncertainty region in frame n. The NCC is computed by

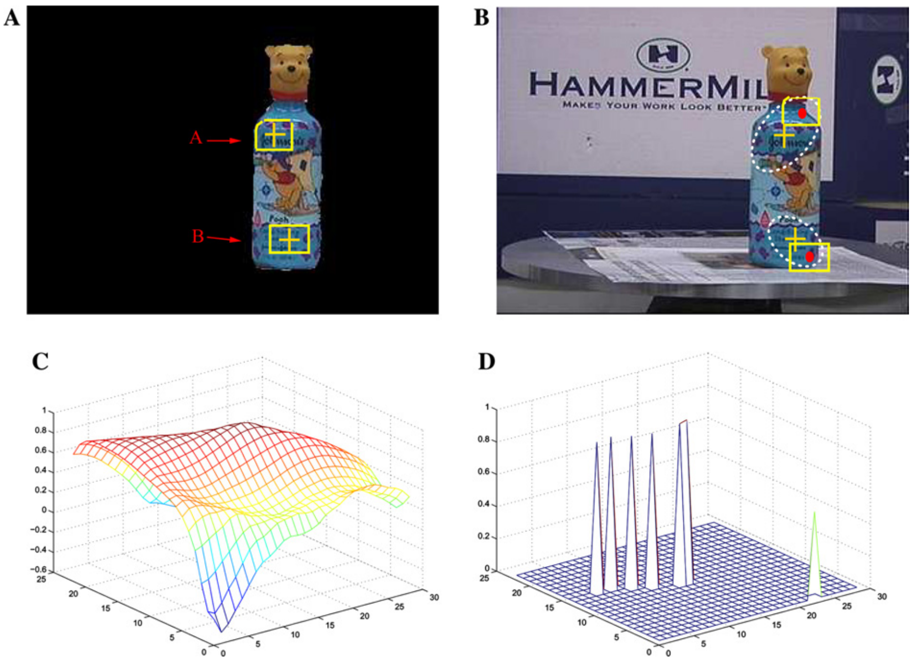


Fig. 6. (A) Original registered feature points represented by white "+"s in frame k, and the white polygons show the optimized matching templates. (B) Corresponding feature match in frame n, where the ellipses stand for predicted search regions and white polygons show NCC calculation at different candidate positions. (C) Calculated NCC map of the search region for point A. (D) NCC map after non-maximum suppression from (C).

$$NCC = \frac{\sum_{(u+\Delta u, v+\Delta v)\in \text{Template}} (I_k(u,v) - \bar{I}_k)(I_n(u+\Delta u, v+\Delta v) - \bar{I}_n)}{\sqrt{\sum_{(u,v)\in \text{Template}} (I_k(u,v) - \bar{I}_k)^2 \sum_{(u+\Delta u, v+\Delta v)\in \text{Template}} (I_n(u+\Delta u, v+\Delta v) - \bar{I}_n)^2}}, \tag{18}$$

where $I_k()$ represents the intensities in frame k, $I_n()$ the intensities in frame n, (u, v) the pixel coordinates of the original feature in frame k, and $(u + \Delta u, v + \Delta v)$ the coordinates of a possible candidate within the predicted region in frame n.

This can be seen in Fig. 6B where the white dotted ellipses depict the feature uncertainty regions. To simplify the computation, during implementation, the rectangular regions specified by the long and the short axes of the ellipses are used. Fig. 6C shows the calculated *NCC* map for feature point *A*.

- Apply eight neighborhood non-maximum suppression to the *NCC* map. The *NCC* map after non-maximum suppression will be composed of 0s and 1s where the 1s stand for the peaks. Fig. 6D is the result obtained after non-maximum suppression from Fig. 6C.
- If the highest *NCC* peak exceeds a certain threshold, record the corresponding position as a valid correspondent for the frame-*k* feature; otherwise, label that feature point as an outlier which means none of the peaks within the predicted uncertainty region can be used as possible candidates for the frame-*k* feature. In Fig. 5D, the point marked "$\hat{\mathbf{q}}^*$" symbolizes the largest peak that is a valid measurement in the *NCC* map.

4. Two-frame motion estimation

In the above section, we showed how to find in the current frame n the correspondents of the features that make their first appearances in frame k. We will now show how these correspondences can be used to update the predicted motion uncertainty ${}_n\mathbf{U}^{\sim}_k$ into a new estimate ${}_n\mathbf{U}_k$.

After previous *Feature Prediction and Extraction* submodule as shown in Fig. 4, the m interior features in frame n are classified into two categories: *Observable* and *Unobservable*. Only the features declared to be observable are retained. Using Kalman filtering, the observable features in frame n are used for updating the motion vector (MV). Unobservable features are simply discarded. (It is possible that an unobservable feature would become observable again in a later frame. If that happens, it can be treated like a new feature if chosen for tracking in that frame.) When a feature is discarded on grounds of observability, the system tries to find a new feature in order to keep the total number of features the same. This new feature gets used subsequently the way we are using the m features from frame k in this discussion.

Motion ${}_n\boldsymbol{U}_k$ *Updating* submodule in Fig. 4 uses the retained features to update ${}_n\mathbf{U}_k$ by extended Kalman filtering, as discussed in Section 4.1. The initially updated motion parameters are used to test the validity of the feature correspondences using

Mahanalobis distances, which is done in *Feature Evaluation* submodule; this is discussed in Section 4.2. If a correspondence gets invalidated, we then temporarily discard that feature and update the Motion uncertainty by using only those features which satisfy both NCC and Mahanalobis distance constraints. This constitutes the second visit of Kalman filtering.

For those invalid feature measurements produced by the Mahanalobis distance pruning process and the NCC thresholding, the system tries to regenerate correspondences as shown in the right branch of the flowchart of Fig. 7. To be more specific, the feature correspondences are re-selected based on NCC from a smaller uncertainty region that is the result of the second visit of Kalman filtering. The uncertainty is finally updated by incorporating these new correspondences. This constitutes the third visit of the Kalman filter.

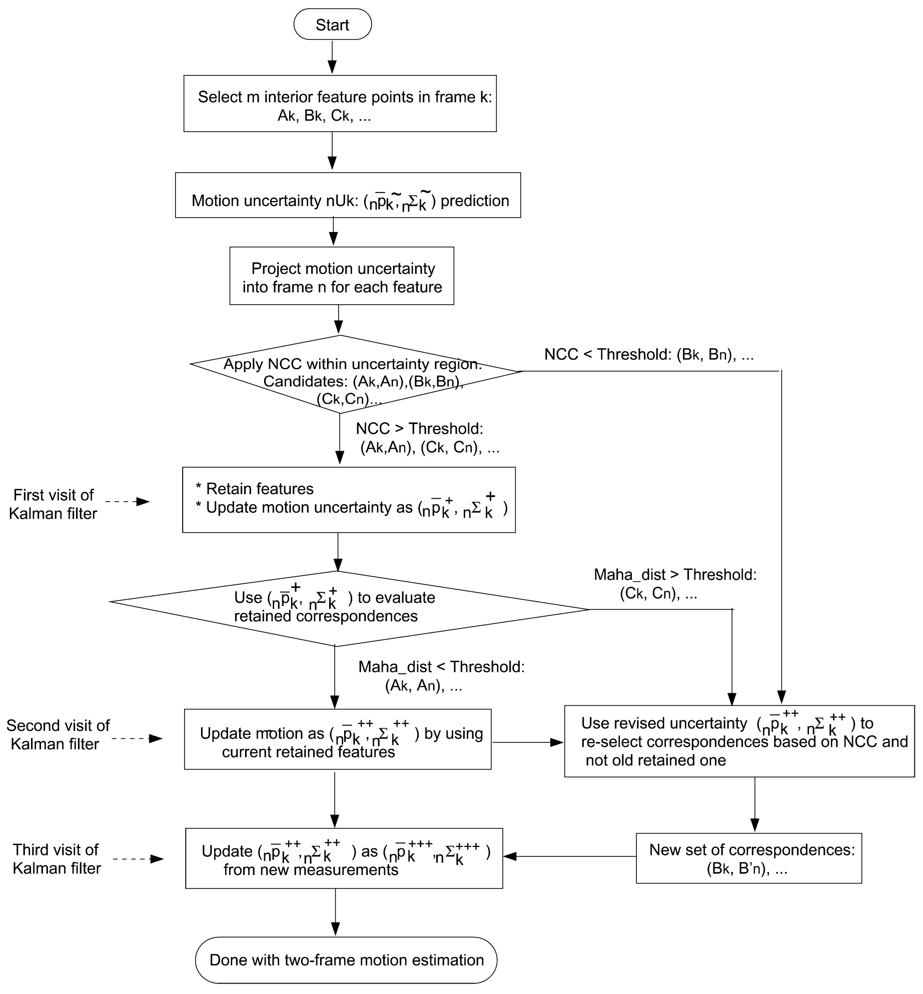

Start
Select m interior feature points in frame k: Ak, Bk, Ck, ...
Motion uncertainty nUk: ($_n\bar{p}_k^{\sim}$, $_n\Sigma_k^{\sim}$) prediction
Project motion uncertainty into frame n for each feature
Apply NCC within uncertainty region. Candidates: (Ak,An),(Bk,Bn),(Ck,Cn)...
NCC < Threshold: (Bk, Bn), ...
NCC > Threshold: (Ak,An), (Ck, Cn), ...
First visit of Kalman filter
* Retain features
* Update motion uncertainty as ($_n\bar{p}_k^+$, $_n\Sigma_k^+$)
Use ($_n\bar{p}_k^+$, $_n\Sigma_k^+$) to evaluate retained correspondences
Maha_dist > Threshold: (Ck, Cn), ...
Maha_dist < Threshold: (Ak, An), ...
Second visit of Kalman filter
Update motion as ($_n\bar{p}_k^{++}$, $_n\Sigma_k^{++}$) by using current retained features
Use revised uncertainty ($_n\bar{p}_k^{++}$, $_n\Sigma_k^{++}$) to re-select correspondences based on NCC and not old retained one
Third visit of Kalman filter
Update ($_n\bar{p}_k^{++}$, $_n\Sigma_k^{++}$) as ($_n\bar{p}_k^{+++}$, $_n\Sigma_k^{+++}$) from new measurements
New set of correspondences: (Bk, B'n), ...
Done with two-frame motion estimation

Fig. 7. Flowchart for feature correspondence matching.

As shown in Fig. 7, the updating of the motion uncertainty takes place at three different points during the processing of the pixel data in the current frame n. We use Kalman filtering for all three updates, and refer to the individual updates as the "First Visit of the Kalman Filter," the "Second Visit of the Kalman Filter," and the "Third Visit of the Kalman Filter." These three applications of the Kalman filter are clearly marked in Fig. 7.

4.1. Updating motion uncertainty from initial feature correspondences

The first Kalman updating of motion uncertainty is derived from the feature points in frame k and their initial correspondents discovered in frame n based on the NCC threshold. The problem addressed can be stated formally as: given image feature positions (u_i, v_i) $(i = 0,\ 1, \ldots, i \leqslant m-1)$ in frame k, their initial correspondents (u_i^*, v_i^*) in frame n, and the predicted motion uncertainty ${}_n\mathbf{U}^{\sim}{}_k : ({}_n\bar{\mathbf{p}}^{\sim}{}_k, {}_n\mathbf{\Sigma}^{\sim}{}_k)$, we want to find a first estimate ${}_n\mathbf{U}_k^+ : ({}_n\bar{\mathbf{p}}_k^+, {}_n\mathbf{\Sigma}_k^+)$.

As we mentioned earlier in Section 3.2, the Kalman estimate ${}_n\mathbf{U}_k^+$ can be derived from the motion constraint equation

$$\mathbf{f}_i({}_n\mathbf{p}_k, \mathbf{r}_i) = \mathbf{0} \tag{19}$$

by forming its linearized approximation around the current best available estimate of the state parameter ${}_n\mathbf{p}_k$—its best current estimate being the mean ${}_n\bar{\mathbf{p}}_k$—and the best available measurement $\hat{\mathbf{r}}_i^* = (u_i^*, v_i^*, \bar{Z}_i)^{\mathrm{T}}$ of $\mathbf{r}_i = (u_i', v_i', Z_i)^{\mathrm{T}}$ as follows:

$$\mathbf{0} \approx \mathbf{f}_i({}_n\bar{\mathbf{p}}_k, \hat{\mathbf{r}}_i^*) + \left.\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} ({}_n\mathbf{p}_k - {}_n\bar{\mathbf{p}}_k) + \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{r}_i}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} (\mathbf{r}_i - \hat{\mathbf{r}}_i^*). \tag{20}$$

Rewriting the above equation, we have

$$-\mathbf{f}_i({}_n\bar{\mathbf{p}}_k, \hat{\mathbf{r}}_i^*) + \left.\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} {}_n\bar{\mathbf{p}}_k = \left.\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} {}_n\mathbf{p}_k + \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{r}_i}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} (\mathbf{r}_i - \hat{\mathbf{r}}_i^*). \tag{21}$$

The linearized observation equation can be written as

$$\tilde{\mathbf{z}}_i = \mathbf{M}_i\, {}_n\mathbf{p}_k + \mathbf{v}_i, \tag{22}$$

where

$$\tilde{\mathbf{z}}_i = -\mathbf{f}_i({}_n\bar{\mathbf{p}}_k, \hat{\mathbf{r}}_i^*) + \left.\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} {}_n\bar{\mathbf{p}}_k, \tag{23}$$

$$\mathbf{M}_i = \left.\frac{\partial \mathbf{f}_i}{\partial({}_n\mathbf{p}_k)}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}}, \tag{24}$$

$$\mathbf{v}_i = \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{r}_i}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} (\mathbf{r}_i - \hat{\mathbf{r}}_i^*). \tag{25}$$

The mathematical calculation of matrix $\mathbf{M}_i$ can be seen in Appendix B. In the context of Eq. (22), the covariance matrix $\mathbf{G}_i$ of observation noise sequence $\mathbf{v}_i$ is given by

$$\mathbf{G}_i = \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{r}_i}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}} \Sigma_{\mathbf{r}_i} \left.\left[\frac{\partial \mathbf{f}_i}{\partial \mathbf{r}_i}\right]^{\mathrm{T}}\right|_{\substack{{}_n\mathbf{p}_k = {}_n\bar{\mathbf{p}}_k \\ \mathbf{r}_i = \hat{\mathbf{r}}_i^*}}, \tag{26}$$

where

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{r}_i} = \begin{bmatrix} \frac{\partial f_1}{\partial u_i'} & \frac{\partial f_1}{\partial v_i'} & \frac{\partial f_1}{\partial Z_i} \\ \frac{\partial f_2}{\partial u_i'} & \frac{\partial f_2}{\partial v_i'} & \frac{\partial f_2}{\partial Z_i} \end{bmatrix} \tag{27}$$

$$\Sigma_{\mathbf{r}_i} = \begin{bmatrix} \Sigma_{\mathbf{q}^{\sim}_i} & 0 \\ \mathbf{0} & \sigma^2_{Z_i} \end{bmatrix}. \tag{28}$$

Then the first estimated motion vector uncertainty ${}_n\mathbf{U}_k^+$ by Kalman filter can be written as:

$$\mathbf{K}_i = {}_n\Sigma^{\sim}_k \mathbf{M}_i^{\mathrm{T}} \left(\mathbf{G}_i + \mathbf{M}_i\,{}_n\Sigma^{\sim}_k \mathbf{M}_i^{\mathrm{T}}\right)^{-1}, \tag{29}$$

$${}_n\bar{\mathbf{p}}_k^+ = {}_n\bar{\mathbf{p}}^{\sim}_k - \mathbf{K}_i\mathbf{f}_i, \tag{30}$$

$${}_n\Sigma_k^+ = (\mathbf{I} - \mathbf{K}_i\mathbf{M}_i)\,{}_n\Sigma^{\sim}_k. \tag{31}$$

Each pair of corresponding feature points $((u_i, v_i), (u_i^*, v_i^*))$—including, of course, only those points in frame k for which correspondents were found in frame n—is used sequentially to update the uncertainty ${}_n\mathbf{U}_k$.

During our EKF formulation (29)–(31) for motion uncertainty updating, there are six unknown parameters and there are $2m$ measurements constituting the constraints on the parameters. To uniquely solve the problem, the number of internal features needed must satisfy $2m > 6$, $m > 3$. So theoretically we just need to register a certain number of feature points to solve the motion estimation problem. With the initialization of motion uncertainty, Eqs. (29)–(31) can update motion uncertainty using a single measurement but the price to pay will be large error covariance. In general, the larger the number of matched feature points in each frame, the better the performance of the filter will be in terms of convergence and accuracy. But when the filter reaches certain convergent point, extra feature points don't contribute to the motion estimation any more.

4.2. A second update of motion uncertainty

Motion uncertainty updating of the last section is based only on the correspondences established on the basis of maximum NCC values.

While *NCC* provides good candidates for the features of frame k, it cannot be relied upon completely for a final choice of correspondences. While it is possible to reduce the errors in template based matching by using large templates to describe the feature points, large templates imply greater computational load.

To reduce the errors in correspondences established on the basis of NCC—and to further improve the motion uncertainty estimate—we project the updated motion uncertainty $({}_n\bar{\mathbf{p}}_k^+, {}_n\mathbf{\Sigma}_k^+)$ into the current frame n and check the validity of each correspondence by applying a Mahalanobis distance based criterion to the location of the extracted feature in frame n and the center of the uncertainty ellipse, as explained in Fig. 8. If the Mahalanobis distance is larger than a certain threshold, we identify this point correspondence as an incorrect match, and eliminate this measurement for motion and shape estimation. Once all outliers are identified, we have a new smaller set of feature correspondences. We then re-update the motion vector from predicted $({}_n\bar{\mathbf{p}}^{\sim}{}_k, {}_n\mathbf{\Sigma}^{\sim}{}_k)$ by a second application of Kalman filter using only the reduced set of feature correspondences. We denote $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\mathbf{\Sigma}_k^{++})$ as the updated motion uncertainty from second application of Kalman filter.

Fig. 8 illustrates the above feature prediction and extraction procedure. For the two features shown in frame k, A, and B, the predicted uncertainty regions $(\bar{\mathbf{q}}^{\sim}, \mathbf{\Sigma}_{\mathbf{q}^{\sim}})$ in the current frame n as computed from $({}_n\bar{\mathbf{p}}^{\sim}{}_k, {}_n\mathbf{\Sigma}^{\sim}{}_k)$ are represented by solid line ellipses. After initial feature matching where the extracted features $\hat{q}^*$ are shown in $\times$, the first application of Kalman filtering yields updated motion parameters denoted by $({}_n\bar{\mathbf{p}}_k^+, {}_n\mathbf{\Sigma}_k^+)$. The updated motion uncertainty leads to the bold-dot ellipses for the new predicted feature location uncertainties $(\bar{\mathbf{q}}^+, \mathbf{\Sigma}_q^+)$. After applying the Mahanalobis distance constraint, it turns out that the measurement $\hat{\mathbf{q}}^*$ corresponding to feature B is an invalid measurement. After discarding the invalid measurement, we update motion uncertainty as $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\mathbf{\Sigma}_p^{++})$ by second visit of Kalman filter. The dash-dot ellipse in the figure will be explained in the next section.

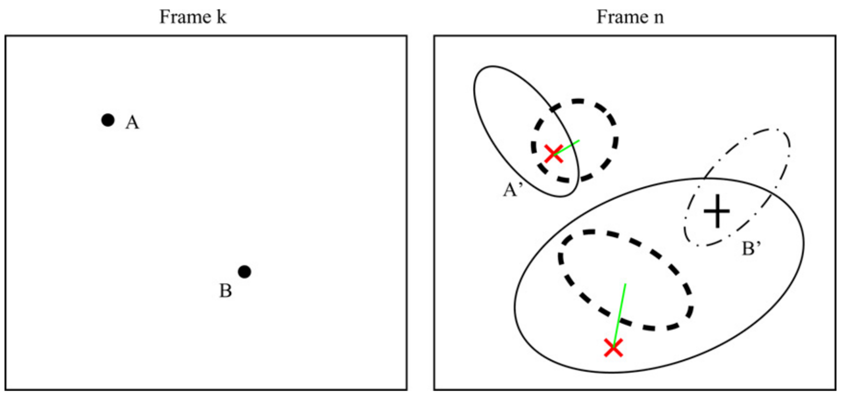


Fig. 8. Feature point prediction and extraction. The solid ellipses in frame n represent the predicted feature uncertainties $(\bar{\mathbf{q}}^{\sim}, \mathbf{\Sigma}_{\mathbf{q}^{\sim}})$ from $({}_n\bar{\mathbf{p}}^{\sim}{}_k, {}_n\mathbf{\Sigma}^{\sim}{}_k)$. The "×"s stand for the initial extracted features $\hat{\mathbf{q}}^*$. The bold-dot ellipses demonstrate the new feature uncertainties $(\bar{\mathbf{q}}^+, \mathbf{\Sigma}_{q^+})$ projected by $({}_n\bar{\mathbf{p}}_k^+, {}_n\mathbf{\Sigma}_k^+)$. The dash-dot ellipse illustrates the new feature uncertainty $(\bar{\mathbf{q}}^{++}, \mathbf{\Sigma}_{q^{++}})$ predicted by $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\mathbf{\Sigma}_k^{++})$, and the "+" shows the re-extracted feature point $\hat{\mathbf{q}}^{**}$ for the initial invalid measurement.

4.3. Seeking new matches for invalidated feature pairings

The processing described so far tries to find the frame n features that correspond to the frame k features. Establishing feature pairings gives us updated motion uncertainty in the form of $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\boldsymbol{\Sigma}_p^{++})$, while invalidating some of the correspondences as not being consistent with the updated estimates of the uncertainty.

We could simply discard the features in frame k whose initially selected correspondences are rejected during the formation of the $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\boldsymbol{\Sigma}_p^{++})$ estimate. But doing so reduces the pool of feature points that get tracked from frame to frame, making it necessary that we inject even more new feature points in each frame. This reduces the quality of the tracking process.

We therefore seek to find the new matches in the new frame n for those frame k features that lost their matches during the processing that led to $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\boldsymbol{\Sigma}_p^{++})$. In the example shown in Fig. 8, that means we must try to find a new match for feature B in frame k.

The feature correspondence re-match is done within a new locational uncertainty region $(\bar{\mathbf{q}}^{++}, \boldsymbol{\Sigma}_{q^{++}})$ obtained from the current updated motion uncertainty $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\boldsymbol{\Sigma}_k^{++})$. In some cases, as for point B in Fig. 8, this may result in a match for a frame-k feature that was otherwise unmatchable on the basis of the $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\boldsymbol{\Sigma}_k^{++})$ estimate. Within the new and smaller uncertainty region $(\bar{\mathbf{q}}^{++}, \boldsymbol{\Sigma}_{q^{++}})$, the measurement $\hat{\mathbf{q}}^{**}$ is extracted from the newly calculated NCC map NCC_{map2}. For obvious reason, if the point corresponding to the first peak value in NCC_{map2} is the same as the initial correspondent $\hat{\mathbf{q}}^{*}$ found in the NCC map NCC_{map1}, the second NCC peak value point within the new smaller NCC_{map2} is selected.

If the new NCC map yields new feature pairings for those points that lost their correspondences during the calculation of $(\bar{\mathbf{q}}^{++}, \boldsymbol{\Sigma}_{q^{++}})$, we re-apply the Kalman filter, using exactly the same formulation as described previously, and update the motion uncertainty using each of new feature pairings. The result of this process is the motion uncertainty $({}_n\bar{\mathbf{p}}_k^{+++}, {}_n\boldsymbol{\Sigma}_k^{+++})$.

This step is illustrated in Fig. 8 where the new locational uncertainty, drawn on the basis of $(\bar{\mathbf{q}}^{++}, \boldsymbol{\Sigma}_{\mathbf{q}^{++}})$, for the frame k feature point B is shown by the ellipse drawn dash-dot. The NCC peak inside this uncertainty region is marked as "+" and denoted $\hat{\mathbf{q}}^{**}$. This new point in frame n becomes the new feature correspondent for point B of frame k. Uncertainty $({}_n\bar{\mathbf{p}}_k^{++}, {}_n\boldsymbol{\Sigma}_k^{++})$, as updated by the new feature pairing for point B in frame k, is denoted $({}_n\bar{\mathbf{p}}_k^{+++}, {}_n\boldsymbol{\Sigma}_k^{+++})$.

The overall process of how feature correspondences are established between a prior frame k and the current frame n is summarized in Fig. 9. As mentioned previously, only observable features play a role in this process and in the related process of motion uncertainty updating. Feature points in frame k for which no correspondences can be found after all of the steps described so far are unobservable. The unobservable feature points are replaced by new feature points, as described in Section 6.

In the rest of this paper, in order to make the notation less cumbersome the motion uncertainty estimate $({}_n\bar{\mathbf{p}}_k^{+++}, {}_n\boldsymbol{\Sigma}_k^{+++})$ will be denoted as ${}_n\mathbf{U}_k : ({}_n\bar{\mathbf{p}}_k, {}_n\boldsymbol{\Sigma}_k)$.

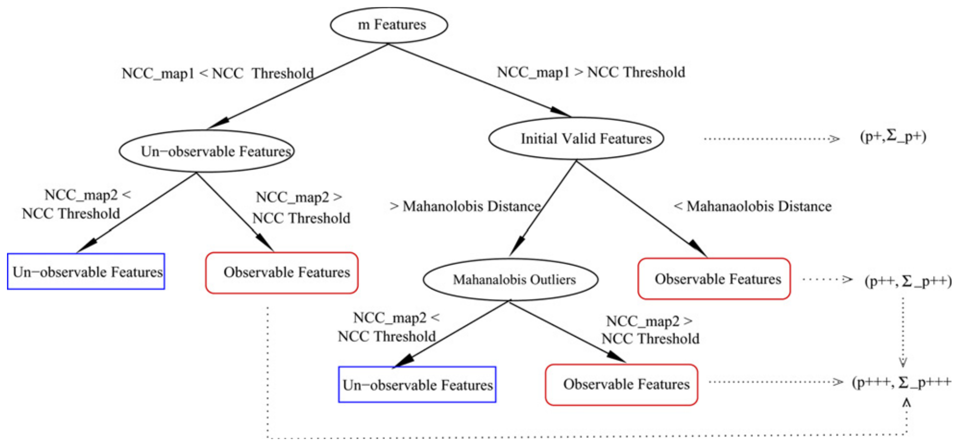


Fig. 9. Feature observability and unobservability for m features initially appeared in frame k.

5. Multi-frame based motion estimation

In the previous section, we talked about the motion estimation from frame k to n based on a set of feature points first appearing in the same frame k. In this section, we will present a framework for integrating all the motion estimates obtained for all previous values of k that contribute features to the current frame n.

In what follows, we first introduce the framework of motion estimation from sets of features with different starting points. For the N features initially registered in frame 0, as we have mentioned before, some, or maybe all, of them will disappear as time goes on due to self or external occlusion, and new features need to be registered. Our motion estimation frame work will introduce how to systematically estimate motion vector (MV) and shape vector (SV) by integrating multiple image frames, and to deal with new feature registration.

5.1. Motion vector estimation

5.1.1. Feature representation

In our motion tracking system, each individual feature in the current frame n is represented by a data structure whose various fields are:

- Starting frame k.
- Normalized image position (u, v) in frame k (normalization is carried out according to Eq. (2))
- Initialized depth value Z in frame k where Z will be updated based on motion vector (MV) updating.
- Transformation from frame 0 to k: ${}_k\mathbf{H}_0$ (already estimated in the previous frames).
- Transformation from frame k to $n-1$: ${}_{n-1}\mathbf{H}_k$ (already estimated in the previous frames).

- Transformation from frame k to n: ${}_n\mathbf{H}_k$ (estimated in current frame).
- Transformation from frame 0 to current frame n: ${}_n\mathbf{H}_0$ (which is what we want to estimate).
- Normalized feature correspondent (u', v') in current frame n.
- Observability status.

For the N initially registered internal features, we keep tracking them until they become unobservable in the image stream. The observability of feature points is defined during feature extraction based on the *NCC* thresholds and Mahanalobis distance constraints as introduced in Section 4. As we mentioned before, only the observable features are used for motion estimation.

5.1.2. Genesis frame based grouping of features in the current frame

Let n be the index of current frame, 0 be the first frame of image sequence, and k be the frame where feature i is first registered. Our goal is, of course, to estimate the object motion from frame 0 to n. Toward that end, we will group the features in the current frame on the basis of the *genesis frame* for the features. The genesis frame for a given feature is that frame in which it first makes its appearance. As was mentioned before, as one or more features become invisible, due to occlusion and possibly other phenomena, the system tries to use new features so that the overall number features in each frame remains constant and equal to N.

The N features in the current frame are grouped according to the index of the genesis frame for each. This is illustrated in Fig. 10. Let the S groupings thus formed be denoted $(\vec{G}_i,\ i = 0, \ldots, S-1)$. The set $\mathscr{E}$ of N feature points in the current frame n can now be represented as:

$$\mathscr{E} = \{\vec{q}_0, \ldots, \vec{q}_{N-1}\} = \vec{G}_0 \cup \cdots \cup \vec{G}_{S-1} \quad (S \leqslant N). \tag{32}$$

For features within a group $\vec{G}_i$ whose genesis index is k, we do feature extraction from frame k to n and motion estimation of ${}_n\mathbf{U}_k : ({}_n\bar{\mathbf{p}}_k, {}_n\mathbf{\Sigma}_k)$ by using the formulas presented in Section 4. In accordance with the discussion in Section 4.3, frame-k features in the current frame n are classified as observable or unobservable during this

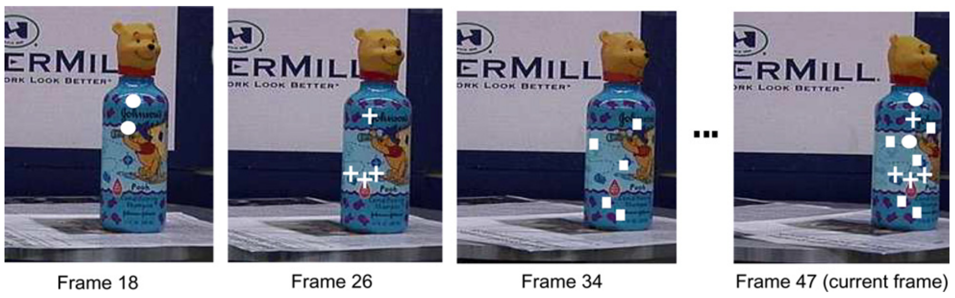


Fig. 10. Feature grouping based on genesis indices, where frame 47 is current processed frame with features initiated from frames 18, 26, 34,.... Different marks of features denote features with different genesis indices.

process. For each frame-k feature that is observable in frame n, we update its initialized depth value Z in the shape vector $\mathbf{w}$. Also, we use all the observable features in frame k to update the motion parameters in frame n as represented by ${}_n\mathbf{U}_k : ({}_n\bar{\mathbf{p}}_k, {}_n\mathbf{\Sigma}_k)$.

5.1.3. Final motion estimation

Given a motion estimate ${}_n\mathbf{U}_k$ in the current frame n for the different values of the genesis frame index k, our goal now is to estimate ${}_n\mathbf{U}_0$. From the estimated ${}_n\mathbf{U}_k$, we can then extract the transform ${}_n\mathbf{H}_k$ from its mean part. In terms of the genesis frame index k, and the transforms ${}_n\mathbf{H}_k$ and ${}_k\mathbf{H}_0$, the new transform ${}_n\mathbf{H}_0$ is

$$ {}_n\mathbf{H}_0 = {}_n\mathbf{H}_k\, {}_k\mathbf{H}_0. \tag{33} $$

Since ${}_k\mathbf{H}_0$ was already estimated when the current frame index was k, and since we have available to us ${}_n\mathbf{H}_k$ from the formulation presented in Section 4, we can use Eq. (33) to update the mean motion vector associated with the uncertainty ${}_n\mathbf{U}_0$.

But note that the above estimate ${}_n\mathbf{H}_0$ is only for a single genesis index k. The issue now is to somehow integrate all such estimates for the different values of k between 0 and the current frame index n. We propose a Kalman framework to bring about this integration (Fig. 11). In this framework, we treat the motion vector ${}_n\mathbf{p}_0$ constituting ${}_n\mathbf{H}_0$ as the "state vector" to be estimated from the "measurements" ${}_n\mathbf{p}_k$ and ${}_k\mathbf{p}_0$ for different values of k.

As before $\mathbf{H}$ is represented as

$$ \mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{34} $$

where rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$ are specified by corresponding $(\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}$.

By expanding Eq. (33), we have the following constraint equations relating the elements of the "state vector" with the "measurements":

$$ \mathscr{R}_{ij} = ({}_n\mathbf{R}_0)_{ij} - ({}_n\mathbf{R}_k\, {}_k\mathbf{R}_0)_{ij} = 0 \quad (i,j = 0,1,2), \tag{35} $$

$$ \mathscr{T} = {}_n\mathbf{T}_0 - {}_n\mathbf{R}_k\, {}_k\mathbf{T}_0 - {}_n\mathbf{T}_k = \mathbf{0}. \tag{36} $$

The Kalman estimation of ${}_n\mathbf{U}_0$ is achieved separately with respect to the two constraint equations shown above, as explained in the two cases presented below.

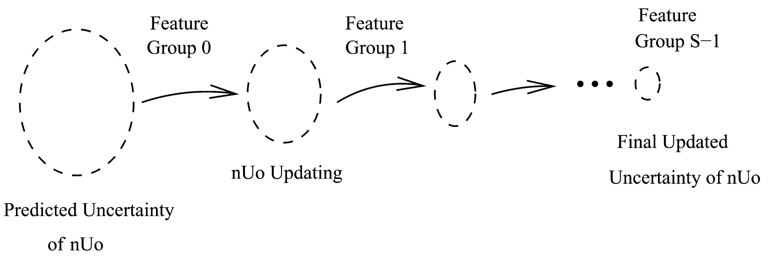

Feature Group 0
Feature Group 1
Feature Group S−1
Predicted Uncertainty of nUo
nUo Updating
Final Updated Uncertainty of nUo

Fig. 11. Sequential motion uncertainty updating by integrating multiple groups of features.

The extended Kalman filter shown below uses the same state vector notation as before—the symbol **p**. Recall, this vector represents the six-tuple $(\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)$. Estimation of this vector from the measurements ${}_n\mathbf{U}_k$ and ${}_k\mathbf{U}_0$, $k = 0,1,2,\ldots$, will yield the desired estimate ${}_n\mathbf{U}_0$. We will use the symbol **r** to represent these measurements, as expressed by the form $({}_n\mathbf{p}_k,\ {}_k\mathbf{p}_0)$. In other words, we have

$$\mathbf{r} = ({}_n\mathbf{p}_k, {}_k\mathbf{p}_0)^{\mathrm{T}}. \tag{37}$$

(1) *Case 1* ($\mathscr{R}_{ij} = 0$)

The constraint of Eq. (35) can be written as

$$\mathscr{R}_{ij} = \mathscr{R}_{ij}({}_n\mathbf{p}_0, \mathbf{r}) = 0. \tag{38}$$

By linearizing $\mathscr{R}_{ij}$ in a similar way as presented in Section 4.1, we can sequentially update $({}_n\bar{\mathbf{p}}_0, {}_n\mathbf{\Sigma}_0)$ by EKF as:

$$\mathbf{K} = {}_n\mathbf{\Sigma}_0\mathbf{M}^{\mathrm{T}}(\mathbf{G} + \mathbf{M}{}_n\mathbf{\Sigma}_0\mathbf{M}^{\mathrm{T}})^{-1}, \tag{39}$$

$$\mathscr{R}_{ij} = ({}_n\mathbf{R}_0)_{ij} - ({}_n\mathbf{R}_k\,{}_k\mathbf{R}_0)_{ij}, \tag{40}$$

$${}_n\bar{\mathbf{p}}_0 = {}_n\widetilde{\bar{\mathbf{p}}}_0 - \mathbf{K}\mathscr{R}_{ij}, \tag{41}$$

$${}_n\mathbf{\Sigma}_0 = (\mathbf{I} - \mathbf{K}\mathbf{M}){}_n\mathbf{\Sigma}_0^{\sim}, \tag{42}$$

where **K** is Kalman gain, **M** stands for the observation matrix, and **G** is the observation error covariance matrix. Here $({}_n\bar{\mathbf{p}}_0^{\sim}, {}_n\mathbf{\Sigma}_0^{\sim})$ is the last updated motion uncertainty. **M** and **G** are calculated by:

$$\mathbf{M} = \frac{\partial\mathscr{R}_{ij}}{\partial_n\mathbf{p}_0} \tag{43}$$

$$\mathbf{G} = \frac{\partial\mathscr{R}_{ij}}{\partial\mathbf{r}}\begin{bmatrix} {}_n\mathbf{\Sigma}_k & 0 \\ 0 & {}_k\mathbf{\Sigma}_0 \end{bmatrix}\left(\frac{\partial\mathscr{R}_{ij}}{\partial\mathbf{r}}\right)^{\mathrm{T}} \tag{44}$$

$$= \begin{bmatrix} \frac{\partial\mathscr{R}_{ij}}{\partial_n\mathbf{p}_k} & \frac{\partial\mathscr{R}_{ij}}{\partial_k\mathbf{p}_0} \end{bmatrix}\begin{bmatrix} {}_n\mathbf{\Sigma}_k & 0 \\ 0 & {}_k\mathbf{\Sigma}_0 \end{bmatrix}\begin{bmatrix} \frac{\partial\mathscr{R}_{ij}}{\partial_n\mathbf{p}_k} & \frac{\partial\mathscr{R}_{ij}}{\partial_k\mathbf{p}_0} \end{bmatrix}^{\mathrm{T}}. \tag{45}$$

(2) *Case 2* ($\mathscr{T} = \mathbf{0}$)

By linearizing

$$\mathscr{T} = \mathscr{T}({}_n\mathbf{p}_0, \mathbf{r}) = \mathbf{0} \tag{46}$$

the EKF related observation matrix **M** and observation error covariance matrix **G** for this case can be defined in a similar way as

$$\mathbf{M} = \frac{\partial\mathscr{T}}{\partial_n\mathbf{p}_0} \tag{47}$$

$$\mathbf{G} = \frac{\partial\mathscr{T}}{\partial\mathbf{r}}\begin{bmatrix} {}_n\mathbf{\Sigma}_k & 0 \\ 0 & {}_k\mathbf{\Sigma}_0 \end{bmatrix}\left(\frac{\partial\mathscr{T}}{\partial\mathbf{r}}\right)^{\mathrm{T}}. \tag{48}$$

The final updated motion uncertainty ${}_n\mathbf{U}_0$ from this constraint is then given by $\mathscr{T} = \mathbf{0}$ through the following formulization

$$\mathbf{K} = {}_n\mathbf{\Sigma}_0\mathbf{M}^{\mathrm{T}}(\mathbf{G} + \mathbf{M}_n\mathbf{\Sigma}_0\mathbf{M}^{\mathrm{T}})^{-1}, \tag{49}$$

$$\mathscr{T} = {}_n\mathbf{T}_0 - {}_n\mathbf{R}_{k\,k}\mathbf{T}_0 + {}_n\mathbf{T}_k, \tag{50}$$

$${}_n\bar{\mathbf{p}}_0 = {}_n\bar{\mathbf{p}}_0^{\sim} - \mathbf{K}\mathscr{T}, \tag{51}$$

$${}_n\mathbf{\Sigma}_0 = (\mathbf{I} - \mathbf{KM})_n\mathbf{\Sigma}_0^{\sim}. \tag{52}$$

Mathematical details underlying the Kalman filters for both cases above can be found in Appendix C.

The initialization of ${}_n\mathbf{U}_0 : ({}_n\bar{\mathbf{p}}_0, {}_n\mathbf{\Sigma}_0)$ is achieved by first initializing ${}_n\bar{\mathbf{p}}_0$ through

$${}_n\mathbf{H}^{\sim}{}_0 = {}_n\mathbf{H}^{\sim}{}_{n-1} * {}_{n-1}\mathbf{H}_k, \tag{53}$$

where motion uncertainty ${}_n\mathbf{H}^{\sim}{}_{n-1}$ is predicted as in Eq. (12), and ${}_{n-1}\mathbf{H}_k$ is stored in feature structure from the previous frame. The covariance matrix ${}_n\mathbf{\Sigma}_0$ for ${}_n\mathbf{U}_0$ is done by the method described in Appendix A.

To make sure the estimated motion parameter converge to the correct answer, whitening process is used during the implementation.

5.2. Shape vector estimation

The discussion so far dealt with the estimation of the motion vector (MV)—more particularly the uncertainty associated with the motion vector. This estimation is elaborate because the motion vector is a global property of the entire object and because this global property must be estimated from local features that appear and disappear in a video sequence.

On the other hand, the shape vector (SV) is associated with a group of features. It is a vector of values proportional to the depths corresponding to the feature points. Being specific to a set of features, as opposed to the entire object, the shape vector is estimated independently of the motion vector and its individual components estimated separately for each feature. The shape vector is updated separately for each of feature points that has the same genesis frame k (Fig. 12), as we will explain in the rest of this section. The shape vector is initialized by setting all elements equal to 1. Each element of $\mathbf{w}$ is updated separately according to the explanation below.

Before we update elements Z_i of the shape vector defined by $\mathbf{w} = (Z_0, Z_1, \ldots, Z_{m-1})^{\mathrm{T}}$, we need to re-estimate ${}_n\mathbf{U}_k : ({}_n\bar{\mathbf{p}}_k, {}_n\mathbf{\Sigma}_k)$ taking into account the latest estimate of ${}_n\mathbf{U}_0 : ({}_n\bar{\mathbf{p}}_0, {}_n\mathbf{\Sigma}_0)$. The previously estimated ${}_n\mathbf{U}_k$ uses feature correspondents in frame n and initialized depths Z_i as measurements, in which Z_i have relatively large uncertainties. The re-estimating of ${}_n\mathbf{U}_k$ by absorbing new information from ${}_n\mathbf{U}_0$ fine-tunes the estimate. The following formula can be used for this purpose:

$${}_n\mathbf{H}_k = {}_n\mathbf{H}_0 * {}_0\mathbf{H}_k = {}_n\mathbf{H}_0 * ({}_k\mathbf{H}_0)^{-1}. \tag{54}$$

The details of how this estimation is carried out are in Appendix D.

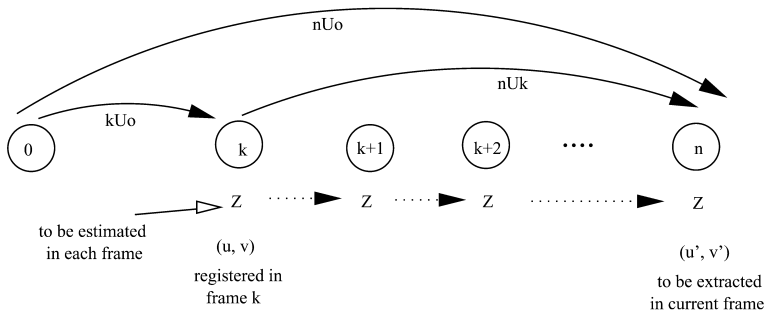

nUo
nUk
kUo
0
k
k+1
k+2
....
n
Z
Z
Z
Z
to be estimated in each frame
(u, v)
registered in frame k
(u', v')
to be extracted in current frame

Fig. 12. Motion vector and shape vector updating.

For a certain feature registered in frame k, once we have the fine-tuned motion estimate ${}_n\mathbf{U}_k$ from frame k to n, the initialized depth value Z in frame k can be based on the components of the frame-k to frame-n pose change vector ${}_n\mathbf{p}_k = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}$, as shown below:

$$u' = \frac{Z(r_{11}u + r_{12}v + r_{13}) + t_x}{Z(r_{31}u + r_{32}v + r_{33}) + t_z}, \tag{55}$$

$$v' = \frac{Z(r_{21}u + r_{22}v + r_{23}) + t_y}{Z(r_{31}u + r_{32}v + r_{33}) + t_z}, \tag{56}$$

where (u,v) are the pixel coordinates of the initially registered feature point in frame k, (u',v') the corresponding coordinates in current frame n, and r_{ij}, $i,j = 0, 1, 2$ the elements of the rotation matrix defined by ${}_n\mathbf{p}_k$. Here for simplicity of notation, we write Z_i as Z. Since the corresponding depth Z' in frame n can be obtained by

$$Z' = Z(r_{31}u + r_{32}v + r_{33}) + t_z, \tag{57}$$

we just need to consider the updating of Z for different features starting from different frame k.

We can rewrite Eqs. (55) and (56) as

$$h_1 = Z(r_{11}u + r_{12}v + r_{13}) + t_x - u'(Z(r_{31}u + r_{32}v + r_{33}) + t_z), \tag{58}$$

$$h_2 = Z(r_{21}u + r_{22}v + r_{23}) + t_x - v'(Z(r_{31}u + r_{32}v + r_{33}) + t_z). \tag{59}$$

To use EKF estimate $(\bar{Z}, \sigma_z)$, we define measurement $\mathbf{r}$, observation matrix $\mathbf{M}$, and measurement error covariance matrix $\mathbf{G}$ as

$$\mathbf{r} = \left[\phi_x, \phi_y, \phi_z, t_x, t_y, t_z, u', v'\right]^{\mathrm{T}}, \tag{60}$$

$$\mathbf{M} = \begin{bmatrix} \frac{\partial h_1}{\partial Z} \\ \frac{\partial h_2}{\partial Z} \end{bmatrix}, \tag{61}$$

$$\mathbf{G} = \left(\frac{\partial \mathbf{h}}{\partial \mathbf{r}}\right) \begin{bmatrix} {}^{\mathbf{n}}\boldsymbol{\Sigma}_{\mathbf{k}}^{\sim} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{q}_i}^{\sim} \end{bmatrix} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{r}}\right)^{\mathrm{T}}, \tag{62}$$

where $\boldsymbol{\Sigma}_{\mathbf{q}_i}^{\sim}$ is the measurement uncertainty for (u',v') and $\mathbf{h} = [h1,h2]^{\mathrm{T}}$. Now we can update depth value Z and standard deviation σ_Z for each feature point as follows:

$$\mathbf{K} = \sigma_Z^2 \mathbf{M}^{\mathrm{T}} (\mathbf{G} + \mathbf{M}\sigma_Z^2 \mathbf{M}^{\mathrm{T}})^{-1}, \tag{63}$$

$$\bar{Z} = \bar{Z}^{\sim} - \mathbf{K}\mathbf{h}, \tag{64}$$

$$\sigma_Z^2 = (\mathbf{I} - \mathbf{K}\mathbf{M})\widetilde{\sigma_Z^2}. \tag{65}$$

The mathematical expressions for $\mathbf{M}$ and $\partial\mathbf{f}/\partial\mathbf{r}$ are provided in Appendix E.

6. Object boundary updating by region-growing

As mentioned in the introduction, some of the boundary points on an object undergoing motion—especially rotation—will become occluded, while other points on the object surface will become the boundary points in the image captured by the camera. The goal of boundary updating is to make a best attempt at extracting the silhouette of the object in the current frame.

As was already mentioned in the Introduction, the problem of silhouette extraction of a moving object undergoing rotation is highly ill-posed [4,40,13,35,28,48,25]. All proposed solutions are based on the assumption that the object pixels in the vicinity of the boundary in the current frame possess texture and color properties similar to the object pixels in the vicinity of the boundary in the previous frame. But this assumption is often not satisfied by real-world objects. Shown in Fig. 13 are two frames of an object undergoing rotations. As is clear from the silhouettes

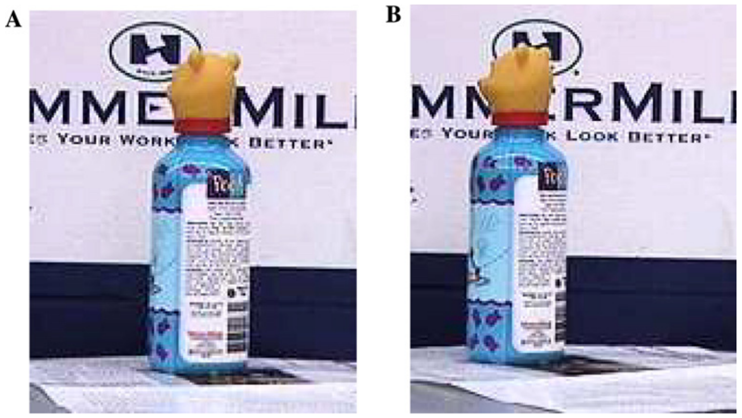


Fig. 13. An example of boundary occlusion due to self-rotation. The nose of the toy is not visible to the camera in the left image.

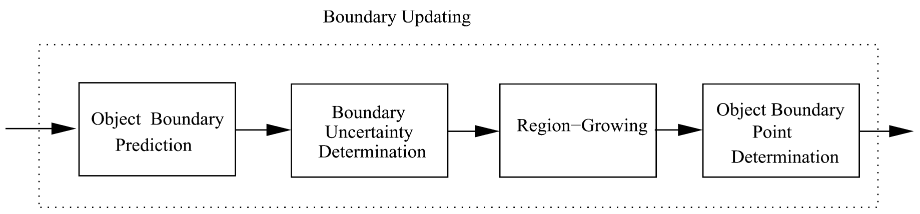

Boundary Updating
Object Boundary Prediction
Boundary Uncertainty Determination
Region–Growing
Object Boundary Point Determination

Fig. 14. ROI boundary refining processing diagrams.

shown for the frames, it would be very difficult to predict the deformation of the silhouette caused by the sudden appearance of the nose of the toy head.

In the rest of this section, we will describe our implementation of boundary updating. While the implementations of the other researchers are based on the assumption that the local properties of the boundary pixels in the current frame should be similar to the local properties of boundary pixels in the previous frame, our implementation is based on a stronger assumption that the local properties of the boundary pixels in the current frame should be similar to the local properties of the nearby interior pixels in the *same* frame. Fig. 14 is a block schematic of our implementation that we will describe in the rest of this section.

6.1. Boundary prediction and uncertainty field definition

Once we are done with motion estimation from frame 0 to frame n, as described in Section 5, the motion parameters between the two adjacent frames $n-1$ and n can be calculated through:

$$ {}_n\mathbf{H}_{n-1} = {}_n\mathbf{H}_0 ({}_{n-1}\mathbf{H}_0)^{-1}. \tag{66}$$

For the rest of the discussion in this section, we now define the following notation:

- $\mathscr{B}_{n-1} = \{\vec{b}_0^{n-1}, \ldots, \vec{b}_{M-1}^{n-1}\}^{\mathrm{T}}$: final extracted boundary in frame $n-1$ where $\vec{b}_i^{n-1}$ is the ith boundary point in the frame.
- $\mathscr{B}_n^{\sim} = \{\vec{b}_0^{n\sim}, \ldots, \vec{b}_{M-1}^{n\sim}\}^{\mathrm{T}}$: predicted object boundary in frame n.
- $\mathscr{B}_n = \{\vec{b}_0^{n}, \ldots, \vec{b}_{K-1}^{n}\}^{\mathrm{T}}$: final extracted boundary in frame n. Note that the number of boundary points, M and K, in frames $n-1$ and n, respectively, can be different due to scaling or self occlusion.

The initially predicted object boundary $\mathscr{B}_n^{\sim}$ in frame n can be obtained by perspectively transforming $\mathscr{B}_{n-1}$ from frame $n-1$ to the current frame n using the motion transform ${}_n\mathbf{H}_{n-1}$. In Fig. 15A, the white contour shows the initially predicted object boundary.

As we mentioned before, in the presence of rotations, the predicted object boundary $\mathscr{B}_n^{\sim}$ may not correspond to the real object boundary $\mathscr{B}_n$. We now define an uncertainty field around the predicted object boundary to search for the real boundary. Based on the predicted ROI boundary $\mathscr{B}_n^{\sim}$, we define $\mathscr{B}^{\mathrm{out}}$ as its dilation, and $\mathscr{B}^{\mathrm{in}}$ as its erosion,

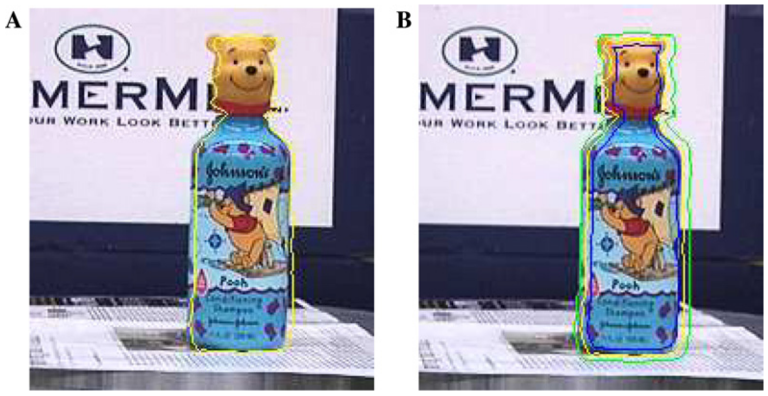


Fig. 15. (A) Predicted ROI boundary $\mathscr{B}_n^{\sim}$ is shown as the white contour. (B) Boundary uncertainty field is enclosed by the black and gray contours.

both by three pixels in our current implementation. Obviously, the extent of dilation and erosion depends on the frame sampling rates vis-a-vis the motion of the object. The uncertainty field is defined as the region between $\mathscr{B}^{\text{out}}$ and $\mathscr{B}^{\text{in}}$. Our goal is to locate the real object boundary $\mathscr{B}_n$ within this field. Fig. 15 illustrates the potential field definition. From the white predicted ROI boundary in Fig. 15A, the gray and the black contours in Fig. 15B correspond to the $\mathscr{B}^{\text{out}}$ and $\mathscr{B}^{\text{in}}$ obtained by morphological filters, and the uncertainty field is defined between the two contours.

6.2. Region-growing for boundary point detection

We will now present a framework that attempts to detect the true boundary points by growing outwards the eroded boundary $\mathscr{B}^{\text{in}}$. This growing process, bounded by the dilated boundary $\mathscr{B}^{\text{out}}$, expands into the uncertainty field using certain similarity and discontinuity measures which will be introduced in this section. The framework consists of the following three steps:

1. Recursively split-and-merge the eroded boundary $\mathscr{B}^{\text{in}}$ into multiple segments so that the variance of the pixel intensity in each segment is bounded. The pixel intensity is calculated from RGB using the usual transformation:
$$I(x,y) = \frac{R(x,y) + G(x,y) + B(x,y)}{3}. \tag{67}$$
2. Grow each segment outward on the basis of the averages and the variances of the intensity, the normalized R, and the normalized G. The normalized R and the normalized G at a pixel (x,y) are denoted $r(x,y)$ and $g(x,y)$. The averages associated with the ith segment are denoted $\hat{\mu}_i^I$, $\hat{\mu}_i^r$, and $\hat{\mu}_i^g$ for the intensity, for normalized R, and normalized G. The standard deviations of the same are denoted $\hat{\sigma}_i^I$, $\hat{\sigma}_i^r$, and $\hat{\sigma}_i^g$.
3. Two stopping criteria are used in the region-growing process:

(a) If no segment can be grown on the basis of the six parameters mentioned in the previous step, stop.
(b) If the growing process reaches the dilated boundary $\mathscr{B}^{\text{out}}$, stop.

4. After the region can no longer be grown further for all of the segments, smooth out the boundary with median filtering.

The following subsections provide further explanation.

6.2.1. Recursive partitioning of the eroded boundary

The eroded boundary $\mathscr{B}^{\text{in}}$ is subject to a recursive split-and-merge step to yield l segments, each denoted $\mathscr{C}^{\text{in}}$. We can therefore write:

$$\mathscr{B}^{\text{in}} = \{\mathscr{C}_i^{\text{in}},\ i = 0, \ldots, l-1\}. \tag{68}$$

The squared Fisher distance [49] is used as the similarity criterion to split the contour recursively:

$$D^2_{\text{Fisher}} = \frac{(n_1 + n_2)(\hat{\mu}_1 - \hat{\mu}_2)^2}{n_1\hat{\sigma}_1^2 + n_2\hat{\sigma}_2^2}, \tag{69}$$

where n_1, n_2, $\hat{\mu}_1$, $\hat{\mu}_2$, $\hat{\sigma}_1^2$, and $\hat{\sigma}_2^2$ are the sizes, intensity means, and intensity variances of the two adjacent contour segments on $\mathscr{B}^{\text{in}}$. During the splitting step, a segment gets bisected into two if the Fisher distance between the two halves exceeds a prespecified decision threshold, `split_threshold`. After the splitting is complete, the merging step takes over. Two adjacent segments are merged if the Fisher distance between them is below another threshold, `merge_threshold`. The `split_threshold` and the `merge_threshold` were specified by trial and error, with the former set to a value much smaller than that of the latter. In all our experiments, the `split_threshold` was set to 0.1 and the `merge_threshold` to 2.

The black and the white boundary segments in Fig. 16A show the segmentation of $\mathscr{B}^{\text{in}}$ after just the splitting step. The segments are shown alternately in black and white. Fig. 16B shows the segments after the merge step.

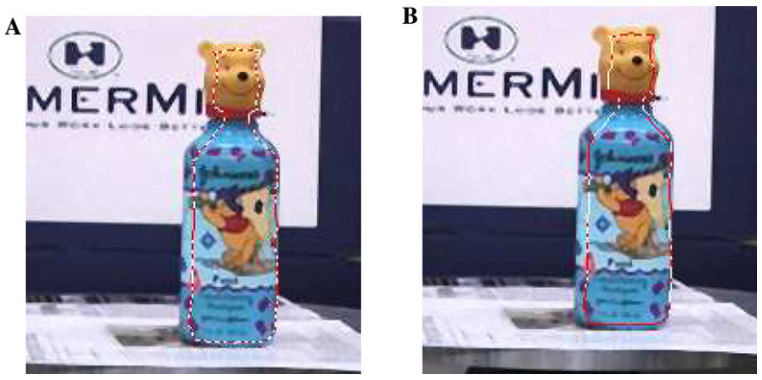


Fig. 16. (A) Splitting of the eroded boundary $\mathscr{B}^{\text{in}}$. The alternate segments of the decomposition are shown in black and white. (B) Boundary partitioning after the merging step.

6.2.2. Growing boundary segments

The segments created by the partitioning algorithm are grown into the uncertainty field by considering each pixel (x, y) that is an 8-neighbor of a pixel on a segment and applying the following criteria to it:

$$\begin{cases} \left|I(x,y) - \hat{\mu}_i^I\right| \leqslant 2\hat{\sigma}_i^I, \\ \left|r(x,y) - \hat{\mu}_i^r\right| \leqslant 2\hat{\sigma}_i^r, \\ \left|g(x,y) - \hat{\mu}_i^g\right| \leqslant 2\hat{\sigma}_i^g, \end{cases} \tag{70}$$

where the notation used was defined in the previous subsection.

To prevent the segment growing process from getting excessively biased by any outliers that may be present in the vicinity of the segment, the six-tuple of properties $\hat{\mu}_i^I, \hat{\mu}_i^r, \hat{\mu}_i^g, \hat{\sigma}_i^I, \hat{\sigma}_i^r, \hat{\sigma}_i^g$ is updated only after all the outward 8-neighbors of all the pixel on a segment have been examined for possible incorporation in the growing of the segment. This update takes place on the basis of all the new pixels accepted. This constitutes one iteration of the segment growing process. The process is repeated until the stopping criteria mentioned previously are met. This is illustrated in Fig. 17.

Fig. 18A shows in black the region grown by this process. This region was grown starting from the partitioned segments shown in Fig. 16B.

After region growing has come to a halt, its outer boundary is smoothed with a 5-point median filter. The outer points of the smoothed region then become the new boundary. The final updated object boundary is shown in red in Fig. 18B.

6.3. Selecting new features for tracking

As we mentioned earlier in the overview Section 2 and in more detail in Section 4, self-occlusions and other kinds of occlusions will cause a certain number of

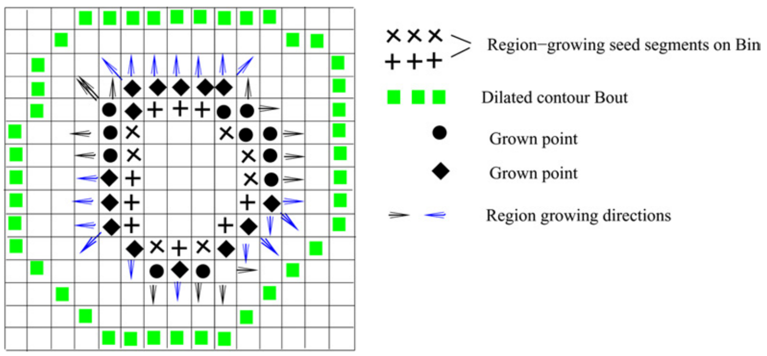

×××
+++
Region–growing seed segments on Bin
Dilated contour Bout
Grown point
Grown point
Region growing directions

Fig. 17. A visual demonstration of growing of the regions starting from the partitioned boundary segments. The points marked "×"s and "+" represent the eroded boundary $\mathscr{B}^{\text{in}}$. The different segments obtained by recursive partitioning are shown as "×"s and "+"s. The accepted 8-neighbors during one iteration of the growing step are shown as filled circles and diamonds.

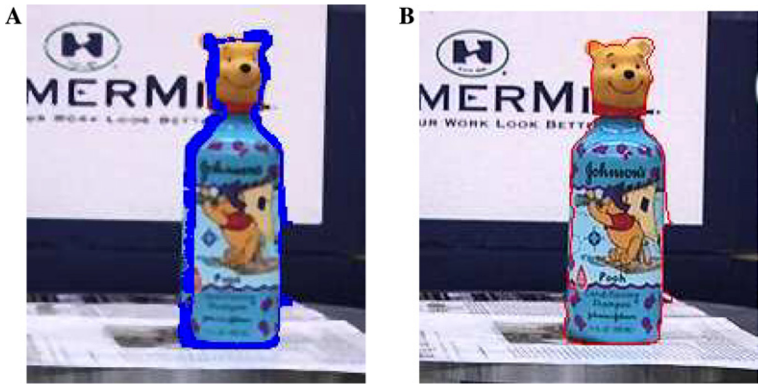


Fig. 18. (A) Grown region from the seed segments of Fig. 16B. (B) Final updated object boundary.

feature points to disappear as an object is being tracked frame to frame. It may therefore become necessary to replenish the the number of features that participate in the tracking process. When needed, the new feature points are selected from the region delineated by the latest estimate of the boundary of the object. These fresh feature points are selected using the same criteria as those mentioned in Section 3. In this manner, the system tries to use the same number of features for tracking at all times.

7. Experimental results

7.1. Experiments with synthetic data

While a tracking algorithm must be shown to work on real data, it is difficult to evaluate the accuracy of motion estimation on actual video sequences of real-world moving objects. If one is interested in testing the accuracy of motion estimation, one must use data that has objects moving with known speeds. Since it is difficult to create real data with objects executing precisely known motions, we have used a two-pronged approach for evaluating the tracking formalism of this paper. We use synthetic data to test the accuracy of motion estimation and then use real video sequences to demonstrate the overall tracking behavior. In this section, we will focus on demonstrating accuracy of our motion estimation approach using synthetic data.

Our synthetic data sequence consists of 100 frames, each of size 640×480. The beginning frame of the sequence has 49 evenly distributed feature points, the cluster situated in the middle of the frame. The feature points are spaced 32 pixels horizontally and vertically. The object region containing these 49 feature points is chosen to be of size 224×224. The object region is depicted as the white middle in Fig. 19A. The black dots in this figure are the initially selected 49 feature points.

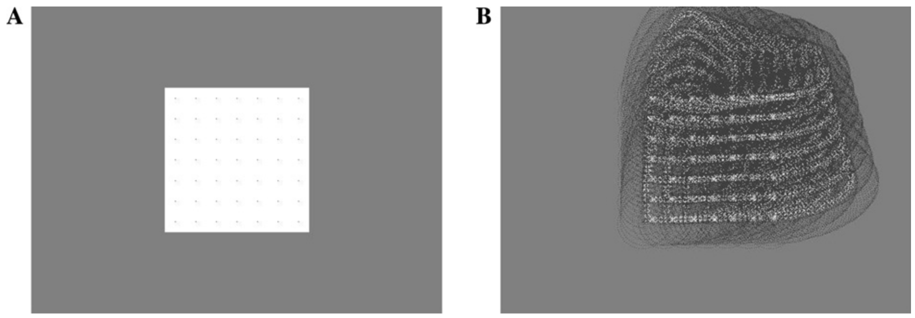


Fig. 19. Synthetic video sequence. (A) Frame 0. (B) Trajectories and uncertainty regions for all feature points.

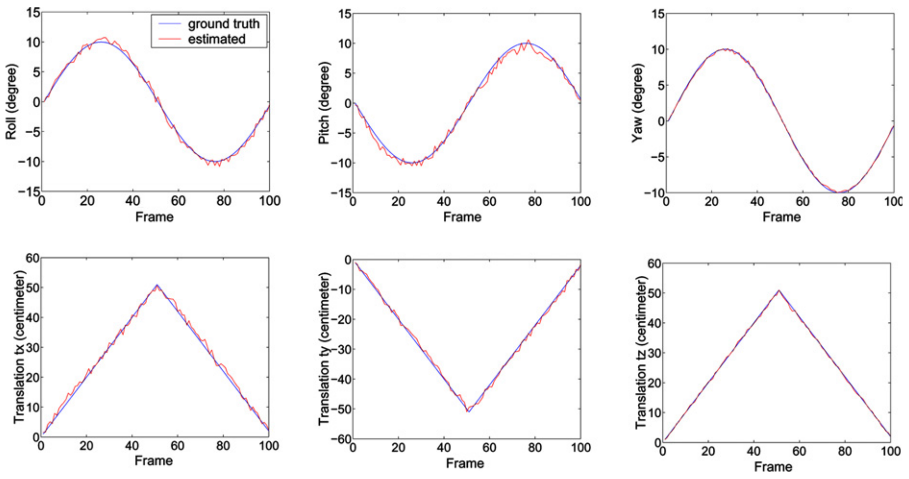
The object motion in the synthetic data is specified through a frame-to-frame motion vector for the object region for all 100 frames of the sequence. The motion specified for the object region includes rotations that vary sinusoidally with amplitude $\pm 10\,^{\circ}$ and translations that vary in a triangular fashion with maximum changes of ± 0.5 m. The motion change patterns for rotation and translation are shown as dark black lines in Fig. 20. During the tracking process, whenever the frame number is modulo 7, a set of five new feature points is added to the object region and five old feature measurements with the largest measurement errors are discarded. The feature measurements are obtained based on the true object motion, but with added zero-mean Gaussian noise of different standard deviations. Fig. 19B shows the trajectories and feature uncertainties of all the feature points during the tracking process when added Gaussian noise has two pixel standard deviation.

Fig. 20 demonstrates the accuracy of motion estimation at three noise levels of standard deviations 2, 4, and 10 pixels. For all the figures in Fig. 20, the light gray lines show one component of the estimated motion vector and the dark lines the ground truth. Table 1 illustrates the statistical performance of the estimator at different noise levels. The table displays the means and standard deviations of the rotation errors and the translation errors between the estimated values and the ground truth. We can see the gradual degradation of the performance with increased noise. Fig. 20 and Table 1 illustrate that the motion estimator performs stably and accurately. Even when the noise levels are high, it can still steadily track and follow the motion trajectories.

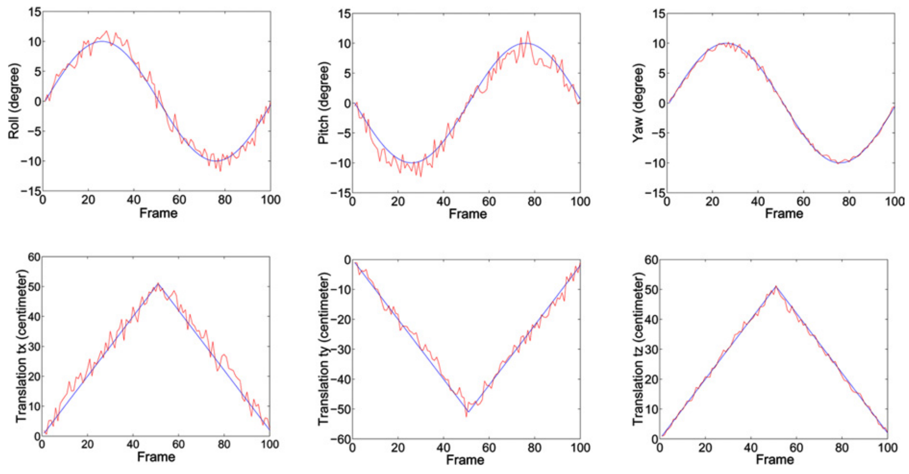
7.2. Experiments with real video sequences

This section will present tracking results using real video sequences. For the discussion in this section, we have chosen four different videos to show our results on:

Video 1: The object to be tracked in this video is of high contrast vis-a-vis the background and the object does not suffer from any self-occlusion.



(A) Estimated motion parameters added with Gaussion noise of 2 pixel standard deviation.



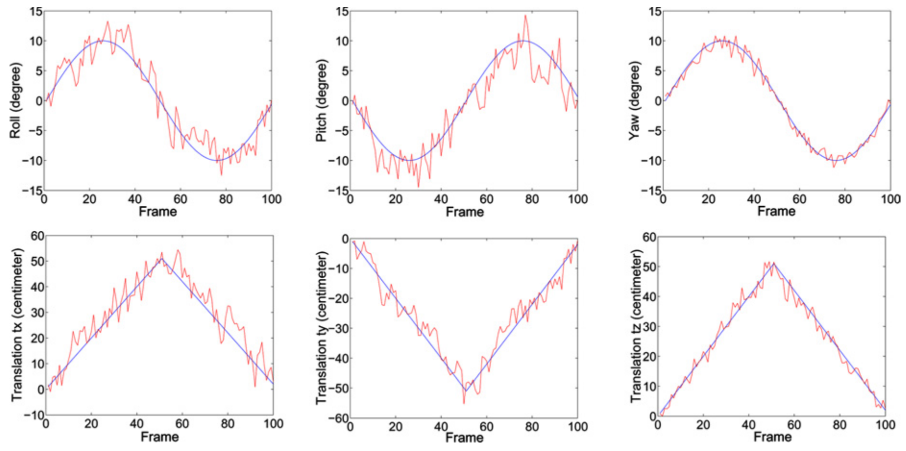
(B) Estimated motion parameters added with Gaussion noise of 4 pixel standard deviation.

Fig. 20. Motion estimation performance evaluation with different standard deviations of the Gaussian noise model.

Video 2: There is still no self-occlusion to be dealt with during tracking, but the colors and the textures on the object are rather similar to those of the background in the vicinity of the object.

Video 3: The object to be tracked undergoes noticeable self-occlusion on account of its rotation. The object background is complex.

Video 4: The object to be tracked exhibits greater self-occlusion than was the case with Video 3. The object is also getting further away from the camera, creating tracking issues related to scale.



(C) Estimated motion parameters added with Gaussion noise of 10 pixel standard deviation.

Fig. 20. (*continued*)

7.2.1. Video 1

As already mentioned, this is the simplest case we will be illustrating as it involves a high-contrast object that does not exhibit any self-occlusion.

This video consists of 150 images, of size 320×240, of a toy that looks like a fish. The video was recorded with a Sony TRV900 digital hand-held camcorder. The object undergoes a pendulum-like motion, with the plane of the motion roughly perpendicular to the camera line of sight. The camera is panned suitably to keep the object roughly in the center of the image frames.

Fig. 21A shows the object to be tracked in frame 0. Fig. 21B shows the human-delineated segmentation of the object in frame 0 using the CISE tool. Fig. 21C shows the feature points selected automatically by our system from the segmented region of Fig. 21B; these points are marked as "+"s. Fig. 21D shows the next frame, frame 1. Also shown in this frame are the ellipses that correspond to the predicted feature uncertainty regions for some of the feature points of frame 0. (We have not shown the ellipses for all the features points of frame 0, since that would clutter up the image excessively.) The system must look for the corresponding features within these uncertainty regions.

Shown in Fig. 22 are the tracking results in nine frames from this video sequence. The frame numbers are shown below the frames. The first, the third, and the fifth rows of this figure show the tracking of the object, in the form of the interior feature points used by the tracking process, and the estimated boundary of the object in the frames shown. The observable interior feature points are shown as "+"s. The second, the fourth, and the sixth rows show the tracked object extracted from its estimated boundary in the indicated frames. In the first, the third and the fifth rows, while the dark boundary is the computed boundary of the object, the white boundary is the eroded version of the predicted boundary of the object. Recall from our

Table 1
Estimation statistics for different noise levels

Guassian noise std (pixel)	Rotation error (°)						Translation error (cm)					
	m_{roll}	σ_{roll}	m_{pitch}	σ_{pitch}	m_{yaw}	σ_{yaw}	m_{tx}	σ_{tx}	m_{ty}	σ_{ty}	m_{tz}	σ_{tz}
2	0.000849	0.474467	−0.361466	0.542372	0.000772	0.152890	0.545231	0.998477	0.033059	0.841634	−0.093237	0.338433
4	0.191100	1.091025	−0.814444	1.265045	−0.049757	0.378270	1.342209	2.332323	0.380866	1.860176	−0.182235	0.890020
6	0.333433	1.636807	−1.029439	1.883077	−0.082283	0.573889	1.770572	3.473754	0.607204	2.700023	−0.263409	1.381075
8	0.386088	2.011334	−1.148336	2.261628	−0.106511	0.729688	2.017599	4.178912	0.665337	3.263815	−0.328758	1.800428
10	0.391593	2.238276	−1.251317	2.447540	−0.133897	0.866250	2.222713	4.518214	0.642090	3.615138	−0.393025	2.167590
12	0.377629	2.385670	−1.356780	2.526870	−0.162614	0.999835	2.423629	4.651280	0.585091	3.867647	−0.461561	2.503753

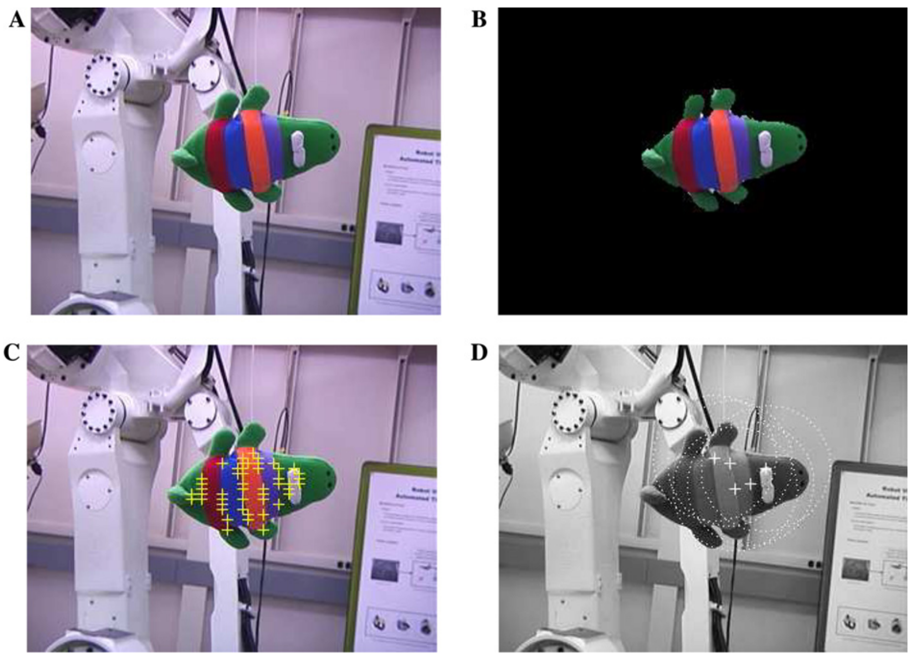


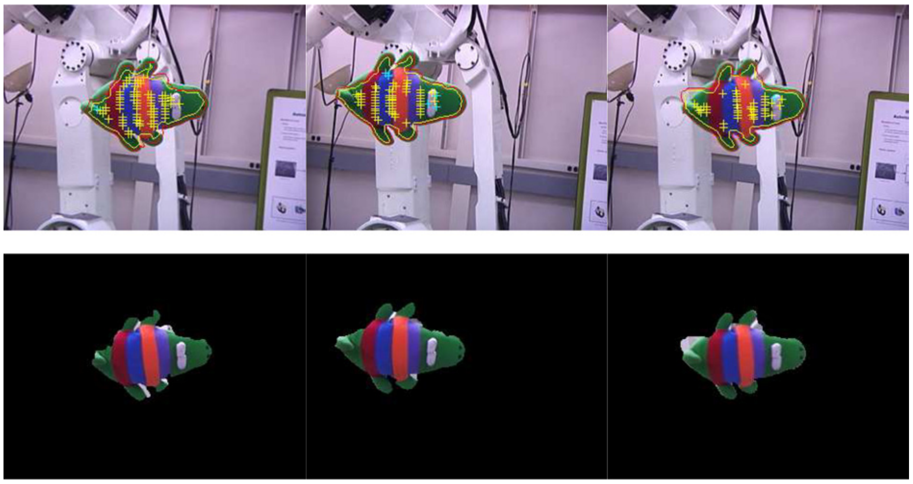
Fig. 21. (A) Frame 0 from the video sequence. (B) Human delineated toy object to be tracked. (This is the output of CISE.) (C) Automatically selected feature points in frame 0. (D) Predicted feature uncertainty regions for a few chosen features are displayed in red dotted ellipses for illustration.

discussion of Section 6, the new boundary in each frame is computed by first projecting into the current frame a motion-compensated boundary from the last frame; this projected boundary is eroded by 3 pixels, and then re-grown to presumably the true edge of the object using the algorithm described in Section 6.

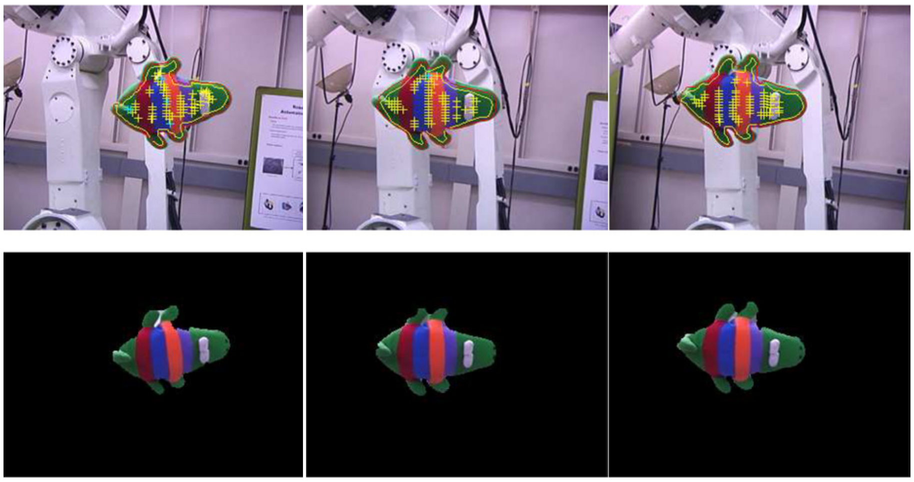
As can be seen in frame 85, it is interesting to note that the predicted boundary is much closer to the actual boundary of the fish in the lower side of the fish, but not on the upper side. This problem is caused by the interaction of the ambient illumination with the fish. In frame 85, the boundary pixels on the upper side of the fish show specularity effects, whereas those on the lower side—the side away from the ceiling-mounted illumination—show only diffuse reflection effects. So it is much more difficult for a region-growing algorithm to extend the eroded form of the predicted boundary to the true edge of the object on the upper side of the fish.

7.2.2. Video 2

This video, consisting of 60 frames, shows images of an accelerating car traveling down a straight road. The camera was panned as the video was being recorded so as to keep the car roughly in the center of the frames. The car is moving from the right to the left in the images. Initially, the car starts its movement while parked in the vicinity of a white van, as shown in (A) of Fig. 23. Shown in (B) of this figure is the human-delineated segmentation of the object to be tracked using the CISE tool.



(A) Frame 15. (B) Frame 30. (C) Frame 50.

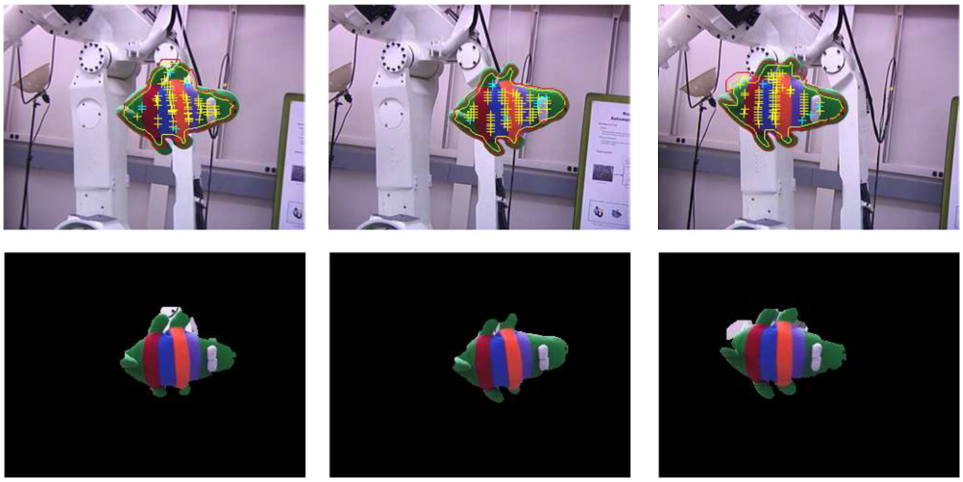


(D) Frame 70. (E) Frame 85. (F) Frame 100.

Fig. 22. Object tracking in video 1. The black contours in the first, third, and the fifth rows show the estimated object boundaries. The white contours are the eroded version of initially predicted boundaries, used for boundary updating. The second, fourth, and the sixth rows show the extracted object from indicated frames.

Part (C) of the figure shows the automatically selected features points for tracking; and part (D) the uncertainty ellipses for some of these points.

Tracking results at different time instances are shown in Fig. 25. The format of the results shown is the same as for Video 1 in Fig. 22. As before, the first and the third rows show the tracking process in the form of the extracted internal features points



(G) Frame 115. (H) Frame 130. (I) Frame 150.

Fig. 22. (*continued*)

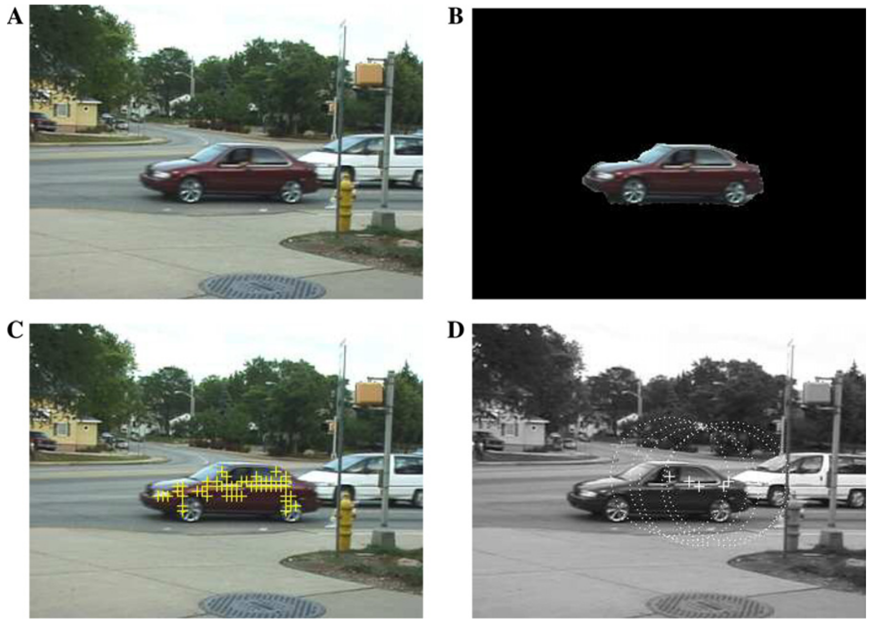


Fig. 23. (A) Frame 0 from the video sequence. (B) Human delineated car object to be tracked. (C) Automatically selected feature points in frame 0. (D) Predicted feature uncertainty regions for a few chosen features.

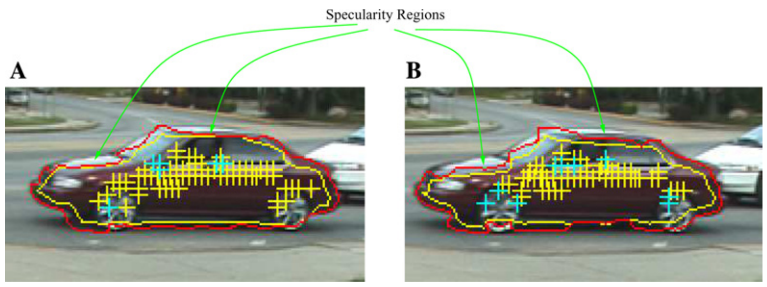


Fig. 24. Un-represented regions during region growing process.

and the computed object boundary in the indicated frames; and the second and the fourth rows the extracted objects. The meaning to be ascribed to the white and the black boundaries is the same as before.

Because the colors and the textures on the car surface are not too different from the background and from the shadow the car casts on the road, the boundary that is updated by region growing from its predicted version often "leaks" into pixels outside the car. Frame 10 and frame 50 are examples of this leakage; the updated boundaries in both these frames underneath the car leak into the shadow regions on the road surface.

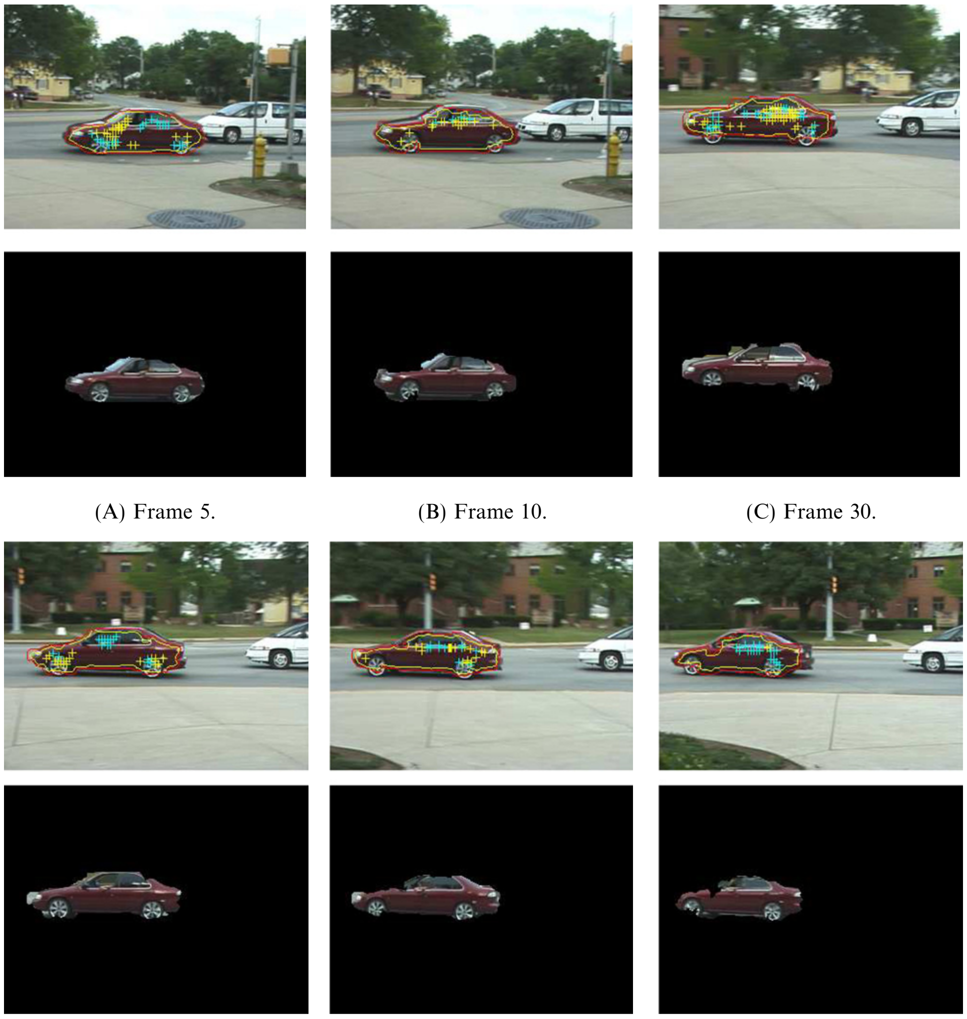
The specularities at the top of the car also cause problems with the updating of the boundaries. To highlight this phenomenon, we show the boundaries for frame 1 and 2 separately in Fig. 24. We have labeled the specular regions on the upper portions of the car. The region growing program is unable to extend the eroded version of the predicted boundaries into these regions.

7.2.3. Video 3

This video has a moving object undergoing some rotation that causes self-occlusion. The video consists of 100 frames taken with a stationary CCD camera. As the reader can see, the background in the vicinity of the moving object, a Folgers Coffee canister, is very cluttered. The contrast between the object and the background is rather indistinct. A person is holding the canister as the right-to-left motion is executed. During this motion, the canister is rotated left to create self-occlusion. *That the object is rotating is made evident by the fact that the letter 'S' of the name "Folgers" is at the right edge of the object in frame 1, but noticeably in the interior of the object, in frame 90.*

Using the same presentation format as for the images in Figs. 22 and 25, the tracking results for this video are shown in Fig. 26. The tracking performance is the same as for the previous two sequences.

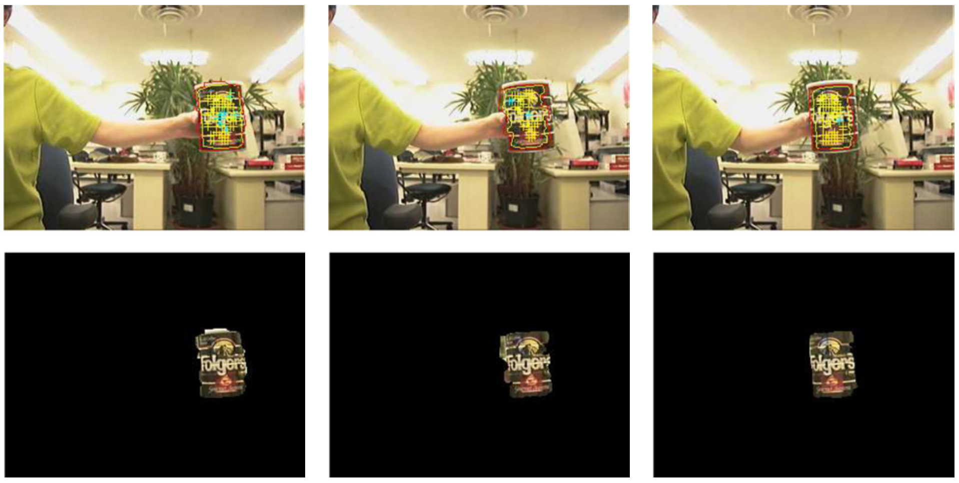
To point out specifically the challenges posed by self-occlusion, we show in Fig. 27 the tracking result obtained when we do not include boundary update through region growing. The specific frames shown in this figure match the frames shown in Fig. 26. Note the increasing error between the computed boundary and the actual boundary of the object. To appreciate the nature of this error, note that in frame 1 the left side of


(A) Frame 5. (B) Frame 10. (C) Frame 30.

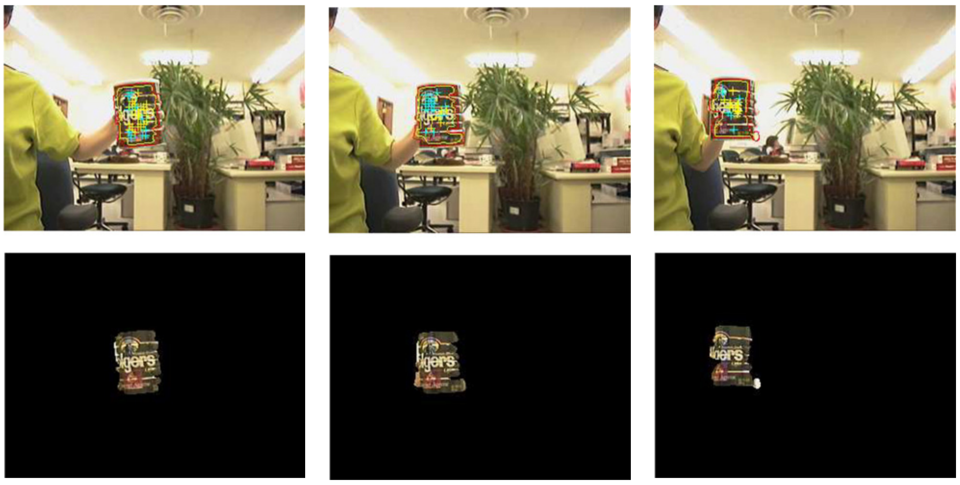
(D) Frame 40. (E) Frame 50. (F) Frame 60.

Fig. 25. Object tracking in video 2. The black contours in the first and the third rows show the estimat ed object boundaries. The white contours are the eroded version of initially predicted boundaries, used for boundary updating. The second and the forth rows show the extracted object from indicated frames.

the boundary passes through the letter ‘F’ of the name ‘‘Folgers’’ and the right side of the boundary passes through the letter ‘S.’ As time progresses, and as the object rotates toward left, the predicted boundary, which has motion compensation built into it, continues to pass through the letter ‘S’ on the right, indicated accurate motion compensation. But, because of object rotation, this predicted boundary is not the actual boundary. The actually boundary, shown in Fig. 26, can only be obtained by our region-growing approach of Section 6.



(A) Frame 1. (B) Frame 10. (C) Frame 20.



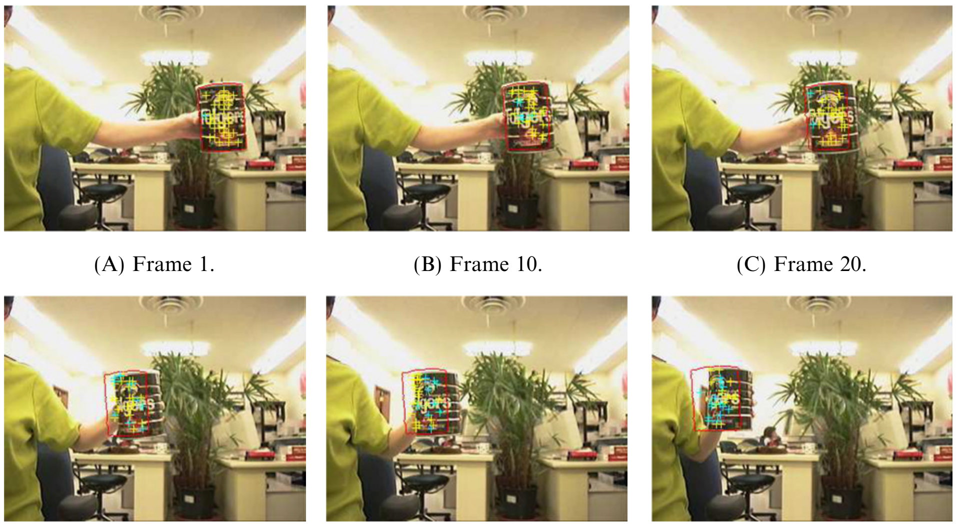
(D) Frame 50. (E) Frame 70. (F) Frame 90.

Fig. 26. Object tracking in video 3. The black contours in the first and the third rows show the estimated object boundaries. The white contours are the eroded version of initially predicted boundaries, used for boundary updating. The second and the fourth rows show the extracted object from indicated frames.

7.2.4. Video 4

This is the most complex of the four video sequences. The moving object, a car again, is turning around a corner, creating a motion that includes object rotation and that, at the same time, causes the object to change its apparent size in the camera image. As before, the rotation creates self-occlusion. The video consists of 103 images.

Using the same presentation format as for the last three videos, the tracking results for this video are shown in Fig. 28. Despite the scale change and the self-


(A) Frame 1. (B) Frame 10. (C) Frame 20.

(D) Frame 50. (E) Frame 70. (F) Frame 90.

Fig. 27. Video 3 sequence tracking without boundary updating.

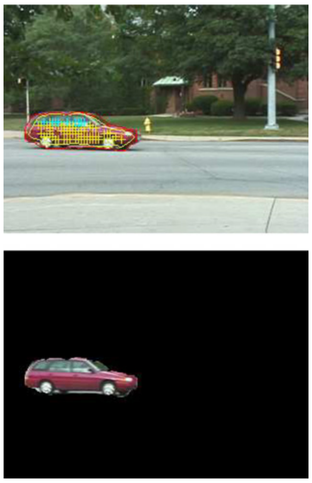
occlusions, the overall tracking performance is the same as for the previous three videos.

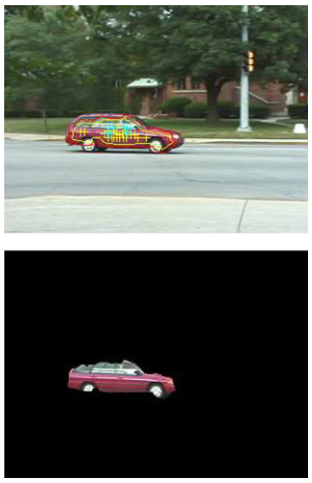
As we did for Video 3, the tracking results obtained when we do not include boundary update through region growing are shown in Fig. 29. The specific frames shown in this figure match the frames shown in Fig. 28. As was the case with the previous video, the extent of error between the computed boundary and the actual boundary of the object grows with time. Note that the predicted boundaries in the results shown in Fig. 29 maintain the shape of the car even as the car is shrinking in size as it moves away from the camera, indicating accurate motion estimation.

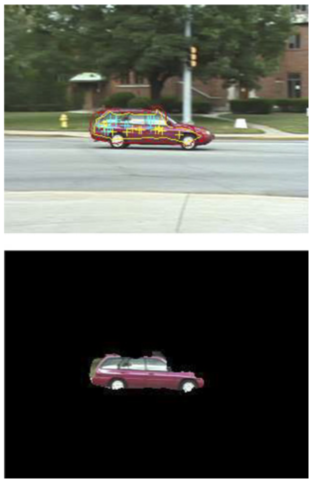
The dividends paid by boundary updating through region growing are evident in what is extracted for the tracked object in the second, the fourth, and the sixth rows of Fig. 28. The extracted pixels contain a majority of the car pixels even though the car is changing its size in the camera image. If we had carried out object extraction based on just the predicted boundaries shown in Fig. 28, the extracted regions would bear very little resemblance to the car.

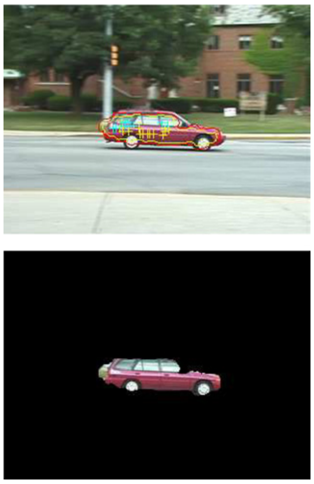
8. Concluding remarks

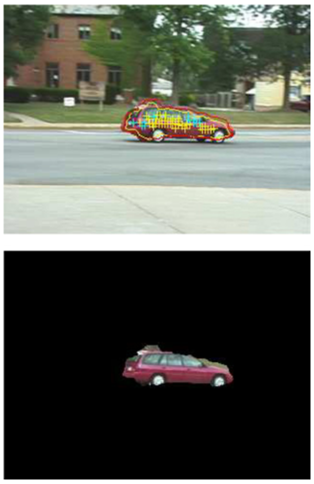
In this paper, we presented a 3D motion estimation scheme for object tracking with emphasis on solving the occlusion problem and providing a visually meaningful object delineation. For many object tracking methods proposed previously [17,28,16,6], the rigid object motion model is limited by the assumption that the


(A) Frame 1.


(B) Frame 10.


(C) Frame 20.


(D) Frame 50.


(E) Frame 70.

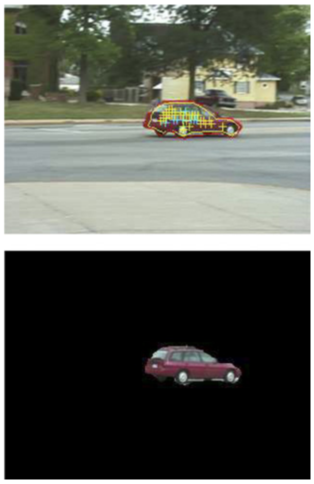
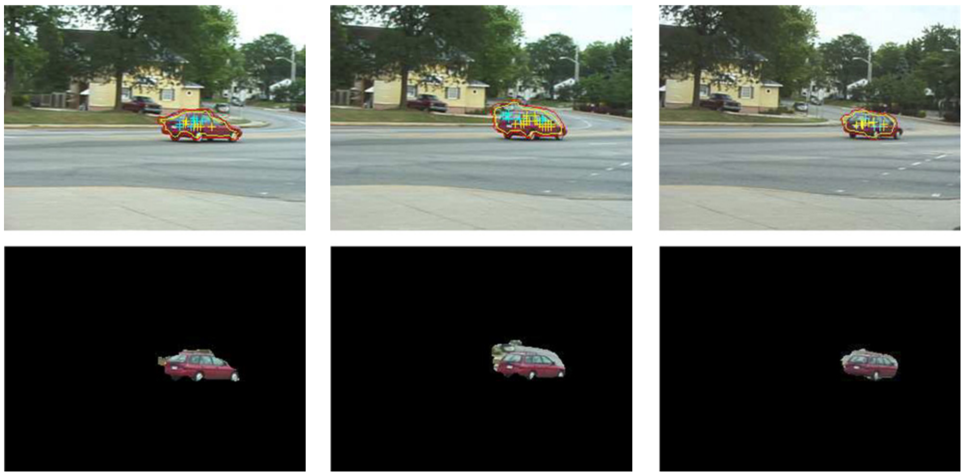

(F) Frame 90.

Fig. 28. Object tracking in video 4. The black contours in the first, third, and the fifth rows show the estimated object boundaries. The white contours are the eroded version of initially predicted boundaries, used for boundary updating. The second, fourth, and the sixth rows show the extracted object from indicated frames.

object consists of a 2D planar surface. Such a motion model fails to describe the motions of a 3D object, especially when the object is allowed to rotate vis-a-vis the camera.

Unlike previous contributions, our approach keeps track of the image feature points from the time they are born until they disappear due to occlusion. We do this



(G) Frame 80. (H) Frame 90. (I) Frame 103.

Fig 28. (*continued*)

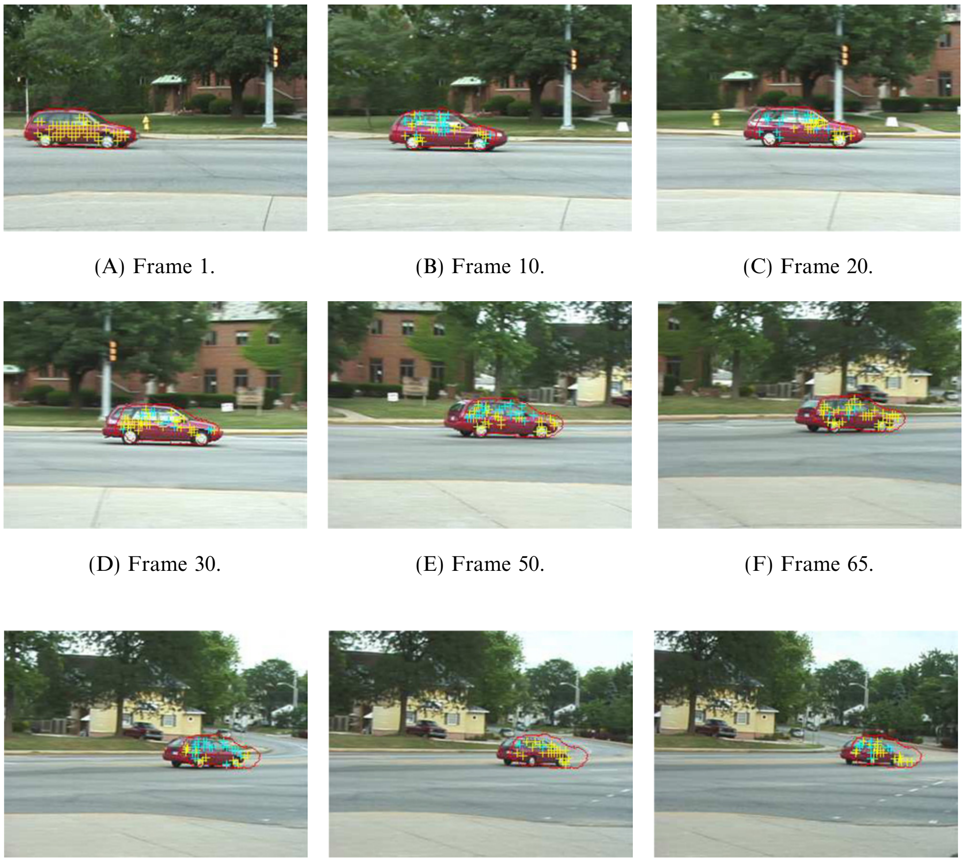
by associating a genesis-frame index with each feature point and bundling together all the feature points in the current frame that have the same genesis frame index. A two-frame Kalman filter is applied to each bundle for the updating of the motion uncertainty in the current frame. This, in our opinion, is the most novel aspect of the tracking formalism proposed in this paper.

There are several possible extensions and enhancements to our work. One obvious way to extend our system would be to couple it to a system for face and object detection. That would eliminate the need for a human to delineate the object boundaries in the first frame of a video sequence. But, then, the performance of the entire system would depend strongly on the performance of the face or the object detector. Since we wanted to focus solely on tracking in this work, it was a deliberate decision on our part to not show any results on such coupled systems. Another way to extend our work would be to apply it to articulated objects. This can be done by decomposing an articulated object into multiple rigid objects and using the geometric relationships as additional constraints for motion estimation.

Appendix A. Motion transform prediction from two transforms

We would like to predict the transform uncertainty $\mathbf{U}_a : (\bar{\mathbf{p}}_a, \mathbf{\Sigma_a})$, given two transforms uncertainties $\mathbf{U}_b : (\bar{\mathbf{p}}_b, \mathbf{\Sigma}_b)$ and $\mathbf{U}_c : (\bar{\mathbf{p}}_c, \mathbf{\Sigma}_c)$, and homogeneous transformation relationship $\mathbf{H}_a = \mathbf{H}_b * \mathbf{H}_c$.

For any given motion vector $\mathbf{p} = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}$, the rotation matrix $\mathbf{R}$, translation vector $\mathbf{T}$, homogeneous transformation $\mathbf{H}$, and their derivatives can be


(A) Frame 1. (B) Frame 10. (C) Frame 20.

(D) Frame 30. (E) Frame 50. (F) Frame 65.

(G) Frame 80. (H) Frame 90. (I) Frame 103.

Fig. 29. Video 4 sequence tracking without boundary updating.

$$c_x = \cos\phi_x, \quad s_x = \sin\phi_x, \tag{A.1}$$

$$c_y = \cos\phi_y, \quad s_y = \sin\phi_y, \tag{A.2}$$

$$c_z = \cos\phi_z, \quad s_z = \sin\phi_z, \tag{A.3}$$

$$\mathbf{R} = \begin{bmatrix} c_z c_y & c_z s_y s_x - s_z c_x & c_z s_y c_x + s_z s_x \\ s_z c_y & s_z s_y s_x + c_z c_x & s_z s_y c_x - c_z s_x \\ -s_y & c_y s_x & c_y c_x \end{bmatrix}, \tag{A.4}$$

$$\mathbf{T} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{O}^{\mathrm{T}} & 1 \end{bmatrix}, \tag{A.5}$$

$$\mathrm{d}\mathbf{R}_x = \frac{\partial \mathbf{R}}{\partial \phi_x} = \begin{bmatrix} 0 & c_z s_y c_x + s_z s_x & -c_z s_y s_x + s_z c_x \\ 0 & s_z s_y c_x - c_z s_x & -s_z s_y s_x - c_z c_x \\ 0 & c_y c_x & -c_y s_x \end{bmatrix}, \tag{A.6}$$

$$\mathrm{d}\mathbf{R}_y = \frac{\partial \mathbf{R}}{\partial \phi_y} = \begin{bmatrix} -c_z s_y & c_z c_y s_x & c_z c_y c_x \\ -s_z s_y & s_z c_y s_x & s_z c_y c_x \\ -c_y & -s_y s_x & -s_y c_x \end{bmatrix}, \tag{A.7}$$

$$\mathrm{d}\mathbf{R}_z = \frac{\partial \mathbf{R}}{\partial \phi_z} = \begin{bmatrix} -s_z c_y & -s_z s_y s_x - c_z c_x & -s_z s_y c_x + c_z s_x \\ c_z c_y & c_z s_y s_x - s_z c_x & c_z s_y c_x + s_z s_x \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.8}$$

So we can re-write relationship $\mathbf{H}_a = \mathbf{H}_b * \mathbf{H}_c$ as

$$\mathbf{R}_a = \mathbf{R}_b \mathbf{R}_c, \quad \mathbf{T}_a = \mathbf{R}_b \mathbf{T}_c + \mathbf{T}_b. \tag{A.9}$$

To be more specific, we have

$$(\mathbf{R}_a)_{ij} = (\mathbf{R}_b \mathbf{R}_c)_{ij} \quad (i, j = 0, 1, 2), \tag{A.10}$$

$$\mathbf{T}_a = \mathbf{R}_b \mathbf{T}_c + \mathbf{T}_b. \tag{A.11}$$

So the components of motion vector $\mathbf{p}$ can be obtained as

$$\phi_x = \operatorname{atan}\left[\frac{(\mathbf{R}_a)_{21}}{(\mathbf{R}_a)_{22}}\right], \tag{A.12}$$

$$\phi_z = \operatorname{atan}\left[\frac{(\mathbf{R}_a)_{10}}{(\mathbf{R}_a)_{00}}\right], \tag{A.13}$$

$$\phi_y = \operatorname{atan}\left[\frac{-(\mathbf{R}_a)_{20}}{(\mathbf{R}_a)_{00}\cos\phi_z + (\mathbf{R}_a)_{10}\sin\phi_z}\right], \tag{A.14}$$

$$t_x = \mathbf{T}_a[0], \quad t_y = \mathbf{T}_a[1], \quad t_x = \mathbf{T}_a[2]. \tag{A.15}$$

Take partial derivatives of Eqs. (A.10), (A.11), we have

$$\delta \mathbf{p}_a = \mathbf{F}\begin{bmatrix} \delta \mathbf{p}_b \\ \delta \mathbf{p}_c \end{bmatrix}. \tag{A.16}$$

Let $\mathbf{p}_a = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}$, $\mathbf{p}_b = (\theta_x, \theta_y, \theta_z, b_x, b_y, b_z)^{\mathrm{T}}$, $\mathbf{p}_c = (\psi_x, \psi_y, \psi_z, q_x, q_y, q_z)^{\mathrm{T}}$, we have

$$\begin{aligned} &\left(\frac{\partial \mathbf{R}_a}{\partial \phi_x}\right)_{ij} \delta\phi_x + \left(\frac{\partial \mathbf{R}_a}{\partial \phi_y}\right)_{ij} \delta\phi_y + \left(\frac{\partial \mathbf{R}_a}{\partial \phi_z}\right)_{ij} \delta\phi_z \\ &\quad = (\mathrm{d}\mathbf{R}_{ax})_{ij}\delta\phi_x + (\mathrm{d}\mathbf{R}_{ay})_{ij}\delta\phi_y + (\mathrm{d}\mathbf{R}_{az})_{ij}\delta\phi_z, \end{aligned} \tag{A.17}$$

$$
\begin{aligned}
&= \frac{\partial(\mathbf{R}_b\mathbf{R}_c)_{ij}}{\partial\theta_x}\delta\theta_x + \frac{\partial(\mathbf{R}_b\mathbf{R}_c)_{ij}}{\partial\theta_y}\delta\theta_y + \frac{\partial(\mathbf{R}_b\mathbf{R}_c)_{ij}}{\partial\theta_z}\delta\theta_z \\
&\quad + \frac{\partial(\mathbf{R}_b\mathbf{R}_c)_{ij}}{\partial\psi_x}\delta\psi_x + \frac{\partial(\mathbf{R}_b\mathbf{R}_c)_{ij}}{\partial\psi_y}\delta\psi_y + \frac{\partial(\mathbf{R}_b\mathbf{R}_c)_{ij}}{\partial\psi_z}\delta\psi_z, \qquad \text{(A.18)}
\end{aligned}
$$

$$
\begin{aligned}
&= (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}c)_{ij}\delta\theta_x + (\mathrm{d}\mathbf{R}_{by}\mathbf{R}c)_{ij}\delta\theta_y + (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}c)_{ij}\delta\theta_z \\
&\quad + (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{ij}\delta\psi_x + (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{ij}\delta\psi_y + (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{ij}\delta\psi_z, \qquad \text{(A.19)}
\end{aligned}
$$

$$
\begin{bmatrix} \delta t_x \\ \delta t_y \\ \delta t_z \end{bmatrix} = \frac{\partial\mathbf{R}_b}{\partial\theta_x}\mathbf{T}_c\delta\theta_x + \frac{\partial\mathbf{R}_b}{\partial\theta_y}\mathbf{T}_c\delta\theta_y + \frac{\partial\mathbf{R}_b}{\partial\theta_z}\mathbf{T}_c\delta\theta_z + \mathbf{R}_b\begin{bmatrix} \delta q_x \\ \delta q_y \\ \delta q_z \end{bmatrix} + \begin{bmatrix} \delta b_x \\ \delta b_y \\ \delta b_z \end{bmatrix}, \qquad \text{(A.20)}
$$

$$
= (\mathrm{d}\mathbf{R}_{bx}\mathbf{T}_c)\delta\theta_x + (\mathrm{d}\mathbf{R}_{by}\mathbf{T}_c)\delta\theta_y + (\mathrm{d}\mathbf{R}_{bz}\mathbf{T}_c)\delta\theta_z + \mathbf{R}_b\begin{bmatrix} \delta q_x \\ \delta q_y \\ \delta q_z \end{bmatrix} + \begin{bmatrix} \delta b_x \\ \delta b_y \\ \delta b_z \end{bmatrix}. \qquad \text{(A.21)}
$$

Combine above two equations, we have

$$
\begin{bmatrix}
(\mathrm{d}\mathbf{R}_{ax})_{00} & (\mathrm{d}\mathbf{R}_{ay})_{00} & (\mathrm{d}\mathbf{R}_{az})_{00} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{01} & (\mathrm{d}\mathbf{R}_{ay})_{01} & (\mathrm{d}\mathbf{R}_{az})_{01} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{02} & (\mathrm{d}\mathbf{R}_{ay})_{02} & (\mathrm{d}\mathbf{R}_{az})_{02} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{10} & (\mathrm{d}\mathbf{R}_{ay})_{10} & (\mathrm{d}\mathbf{R}_{az})_{10} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{11} & (\mathrm{d}\mathbf{R}_{ay})_{11} & (\mathrm{d}\mathbf{R}_{az})_{11} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{12} & (\mathrm{d}\mathbf{R}_{ay})_{12} & (\mathrm{d}\mathbf{R}_{az})_{12} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{20} & (\mathrm{d}\mathbf{R}_{ay})_{20} & (\mathrm{d}\mathbf{R}_{az})_{20} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{21} & (\mathrm{d}\mathbf{R}_{ay})_{21} & (\mathrm{d}\mathbf{R}_{az})_{21} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{ax})_{22} & (\mathrm{d}\mathbf{R}_{ay})_{22} & (\mathrm{d}\mathbf{R}_{az})_{22} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & I
\end{bmatrix}
\begin{bmatrix} \delta\phi_x \\ \delta\phi_y \\ \delta\phi_z \\ \delta t_x \\ \delta t_y \\ \delta t_z \end{bmatrix}
$$

$$
= \begin{bmatrix}
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{00} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{00} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{00} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{00} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{00} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{00} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{01} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{01} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{01} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{01} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{01} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{01} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{02} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{02} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{02} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{02} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{02} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{02} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{10} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{10} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{10} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{10} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{10} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{10} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{11} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{11} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{11} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{11} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{11} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{11} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{12} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{12} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{12} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{12} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{12} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{12} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{20} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{20} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{20} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{20} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{20} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{20} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{21} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{21} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{21} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{21} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{21} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{21} & \mathbf{0} \\
(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{22} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{22} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{22} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{22} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{22} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{22} & \mathbf{0} \\
\mathrm{d}\mathbf{R}_{bx}\mathbf{T}_c & \mathrm{d}\mathbf{R}_{by}\mathbf{T}_c & \mathrm{d}\mathbf{R}_{bz}\mathbf{T}_c & \mathbf{I} & 0 & 0 & 0 & \mathbf{R}_b
\end{bmatrix}
\begin{bmatrix} \delta\theta_x \\ \delta\theta_y \\ \delta\theta_z \\ \delta b_x \\ \delta b_y \\ \delta b_z \\ \delta\psi_x \\ \delta\psi_y \\ \delta\psi_z \\ \delta q_x \\ \delta q_y \\ \delta q_z \end{bmatrix}. \qquad \text{(A.22)}
$$

Define 6×12 matrix A and 12×12 matrix B as:

$$\mathbf{A} = \begin{bmatrix} (\mathrm{d}\mathbf{R}_{ax})_{00} & (\mathrm{d}\mathbf{R}_{ay})_{00} & (\mathrm{d}\mathbf{R}_{az})_{00} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{01} & (\mathrm{d}\mathbf{R}_{ay})_{01} & (\mathrm{d}\mathbf{R}_{az})_{01} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{02} & (\mathrm{d}\mathbf{R}_{ay})_{02} & (\mathrm{d}\mathbf{R}_{az})_{02} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{10} & (\mathrm{d}\mathbf{R}_{ay})_{10} & (\mathrm{d}\mathbf{R}_{az})_{10} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{11} & (\mathrm{d}\mathbf{R}_{ay})_{11} & (\mathrm{d}\mathbf{R}_{az})_{11} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{12} & (\mathrm{d}\mathbf{R}_{ay})_{12} & (\mathrm{d}\mathbf{R}_{az})_{12} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{20} & (\mathrm{d}\mathbf{R}_{ay})_{20} & (\mathrm{d}\mathbf{R}_{az})_{20} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{21} & (\mathrm{d}\mathbf{R}_{ay})_{21} & (\mathrm{d}\mathbf{R}_{az})_{21} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{ax})_{22} & (\mathrm{d}\mathbf{R}_{ay})_{22} & (\mathrm{d}\mathbf{R}_{az})_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix} \tag{A.23}$$

$$\mathbf{B} = \begin{bmatrix} (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{00} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{00} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{00} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{00} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{00} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{00} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{01} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{01} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{01} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{01} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{01} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{01} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{02} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{02} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{02} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{02} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{02} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{02} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{10} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{10} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{10} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{10} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{10} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{10} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{11} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{11} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{11} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{11} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{11} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{11} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{12} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{12} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{12} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{12} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{12} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{12} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{20} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{20} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{20} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{20} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{20} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{20} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{21} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{21} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{21} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{21} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{21} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{21} & \mathbf{0} \\ (\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{22} & (\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{22} & (\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{22} & \mathbf{0} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{22} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{22} & (\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{22} & \mathbf{0} \\ \mathrm{d}\mathbf{R}_{bx}\mathbf{T}_c & \mathrm{d}\mathbf{R}_{by}\mathbf{T}_c & \mathrm{d}\mathbf{R}_{bz}\mathbf{T}_c & \mathbf{I} & 0 & 0 & 0 & \mathbf{R}_b \end{bmatrix} \tag{A.24}$$

Then we have

$$\mathbf{F} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}(\mathbf{A}^{\mathrm{T}}\mathbf{B}). \tag{A.25}$$

So the mean value and covariance of motion vector $\mathbf{p}_a$ can be obtained as:

$$\bar{\mathbf{R}}_a = \bar{\mathbf{R}}_b\bar{\mathbf{R}}_c, \tag{A.26}$$

$$\bar{\mathbf{T}}_a = \bar{\mathbf{R}}_b\bar{\mathbf{T}}_c + \bar{\mathbf{T}}_b, \tag{A.27}$$

$$\mathbf{\Sigma}_{p_a} = \mathbf{F}\begin{bmatrix} \mathbf{\Sigma}_{p_b} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{\mathbf{p_c}} \end{bmatrix}\mathbf{F}^{\mathrm{T}}. \tag{A.28}$$

Appendix B. Jacobian matrix of perspective motion transform

The perspective transform $\mathbf{f_i}$ for a point (u, v) with depth value Z from frame k to point (u', v') in frame n is

$$\mathbf{f_i} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \tag{B.1}$$

$$= \begin{bmatrix} u' - \frac{r_{11}u+r_{12}v+r_{13}+\frac{t_x}{Z}}{r_{31}u+r_{32}v+r_{33}+\frac{t_z}{Z}} \\ v' - \frac{r_{21}u+r_{22}v+r_{23})+\frac{t_y}{Z}}{r_{31}u+r_{32}v+r_{33})+\frac{t_z}{Z}} \end{bmatrix}, \tag{B.2}$$

where r_{ij}, $(i,j = 0,\ 1,\ 2)$ are elements of rotation matrix $\mathbf{R}$ defined by motion vector ${}_n\mathbf{p}_k = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}$.

Define

$$\mathrm{d}\mathbf{R}_x = \frac{\partial \mathbf{R}}{\partial \phi_x}, \quad \mathrm{d}\mathbf{R}_y = \frac{\partial \mathbf{R}}{\partial \phi_y}, \quad \mathrm{d}\mathbf{R}_z = \frac{\partial \mathbf{R}}{\partial \phi_z}, \tag{B.3}$$

$$u_p = Z(r_{11}u + r_{12}v + t_x), \tag{B.4}$$

$$v_p = Z(r_{11}u + r_{12}v + t_x), \tag{B.5}$$

$$z_p = Z(r_{11}u + r_{12}v + t_x). \tag{B.6}$$

We can compute the Jacobian matrix of $\mathbf{f}_i$ as

$$\frac{\partial \mathbf{f}}{\partial({}_n\mathbf{p}_k)} = \begin{bmatrix} \frac{\partial f_1}{\partial \phi_x} & \frac{\partial f_1}{\partial \phi_y} & \frac{\partial f_1}{\partial \phi_z} & \frac{\partial f_1}{\partial t_x} & \frac{\partial f_1}{\partial t_y} & \frac{\partial f_1}{\partial t_z} \\ \frac{\partial f_2}{\partial \phi_x} & \frac{\partial f_2}{\partial \phi_y} & \frac{\partial f_2}{\partial \phi_z} & \frac{\partial f_2}{\partial t_x} & \frac{\partial f_2}{\partial t_y} & \frac{\partial f_2}{\partial t_z} \end{bmatrix}, \tag{B.7}$$

where

$$\frac{\partial f_1}{\partial \phi_x} = \frac{Z((\mathrm{d}\mathbf{R}_x)_{11}u + (\mathrm{d}\mathbf{R}_x)_{12}v + (\mathrm{d}\mathbf{R}_x)_{13})z_p - u_p Z((\mathrm{d}\mathbf{R}_x)_{31}u + (\mathrm{d}\mathbf{R}_x)_{32}v + (\mathrm{d}\mathbf{R}_x)_{33})}{z_p^2}, \tag{B.8}$$

$$\frac{\partial f_1}{\partial \phi_y} = \frac{Z((\mathrm{d}\mathbf{R}_y)_{11}u + (\mathrm{d}\mathbf{R}_y)_{12}v + (\mathrm{d}\mathbf{R}_y)_{13})z_p - u_p Z((\mathrm{d}\mathbf{R}_y)_{31}u + (\mathrm{d}\mathbf{R}_y)_{32}v + (\mathrm{d}\mathbf{R}_y)_{33})}{z_p^2}, \tag{B.9}$$

$$\frac{\partial f_1}{\partial \phi_z} = \frac{Z((\mathrm{d}\mathbf{R}_z)_{11}u + (\mathrm{d}\mathbf{R}_z)_{12}v + (\mathrm{d}\mathbf{R}_z)_{13})z_p - u_p Z((\mathrm{d}\mathbf{R}_z)_{31}u + (\mathrm{d}\mathbf{R}_z)_{32}v + (\mathrm{d}\mathbf{R}_z)_{33})}{z_p^2}, \tag{B.10}$$

$$\frac{\partial f_1}{\partial t_x} = \frac{1}{z_p}, \quad \frac{\partial f_1}{\partial t_y} = 0, \quad \frac{\partial f_1}{\partial t_z} = \frac{-u_p}{z_p^2}, \tag{B.11}$$

$$\frac{\partial f_2}{\partial \phi_x} = \frac{Z((\mathrm{d}\mathbf{R}_x)_{21}u + (\mathrm{d}\mathbf{R}_x)_{22}v + (\mathrm{d}\mathbf{R}_x)_{23})z_p - u_p Z((\mathrm{d}\mathbf{R}_x)_{31}u + (\mathrm{d}\mathbf{R}_x)_{32}v + (\mathrm{d}\mathbf{R}_x)_{33})}{z_p^2}, \tag{B.12}$$

$$\frac{\partial f_2}{\partial \phi_y} = \frac{Z((\mathrm{d}\mathbf{R}_y)_{21}u + (\mathrm{d}\mathbf{R}_y)_{22}v + (\mathrm{d}\mathbf{R}_y)_{23})z_p - u_p Z((\mathrm{d}\mathbf{R}_y)_{31}u + (\mathrm{d}\mathbf{R}_y)_{32}v + (\mathrm{d}\mathbf{R}_y)_{33})}{z_p^2}, \tag{B.13}$$

$$\frac{\partial f_2}{\partial \phi_z} = \frac{Z((\mathrm{d}\mathbf{R}_z)_{21}u + (\mathrm{d}\mathbf{R}_z)_{22}v + (\mathrm{d}\mathbf{R}_z)_{23})z_p - u_p Z((\mathrm{d}\mathbf{R}_z)_{31}u + (\mathrm{d}\mathbf{R}_z)_{32}v + (\mathrm{d}\mathbf{R}_z)_{33})}{z_p^2}, \tag{B.14}$$

$$\frac{\partial f_1}{\partial t_x} = 0, \quad \frac{\partial f_1}{\partial t_y} = \frac{1}{z_p}, \quad \frac{\partial f_1}{\partial t_z} = \frac{-v_p}{z_p^2}. \tag{B.15}$$

Appendix C. Motion estimation from two transforms

We would like to update the motion uncertainty from frame 0 to n associated with ${}_n\mathbf{U}_0 : ({}_n\bar{\mathbf{p}}_0, {}_n\mathbf{\Sigma}_0)$, given two transformations from frame 0 to k and from k to n respectively as ${}_k\mathbf{U}_0 : ({}_k\bar{\mathbf{p}}_0, {}_k\mathbf{\Sigma}_0)$ and ${}_n\mathbf{U}_k : ({}_n\bar{\mathbf{p}}_k, {}_n\mathbf{\Sigma}_k)$. In general case, given transformation uncertainties $(\bar{\mathbf{p}}_b, \mathbf{\Sigma}_b)$ and $(\bar{\mathbf{p}}_c, \mathbf{\Sigma}_c)$, and initial estimate of $(\bar{\mathbf{p}}_a, \mathbf{\Sigma}_a)$, we want to update $(\bar{\mathbf{p}}_a, \mathbf{\Sigma}_a)$ based on homogeneous transformation relationship $\mathbf{H}_a = \mathbf{H}_b * \mathbf{H}_c$, or

$$\mathbf{R}_a = \mathbf{R}_b\mathbf{R}_c, \quad \mathbf{T}_a = \mathbf{R}_b\mathbf{T}_c + \mathbf{T}_b. \tag{C.1}$$

Let $\mathbf{p}_a = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}, \mathbf{p}_b = (\theta_x, \theta_y, \theta_z, s_x, s_y, s_z)^{\mathrm{T}}, \ \mathbf{p}_c = (\psi_x, \psi_y, \psi_z, q_x, q_y, q_z)^{\mathrm{T}}$, then we have the following equations:

$$\mathscr{R}_{ij} = (\mathbf{R}_a)_{ij} - (\mathbf{R}_b\mathbf{R}_c)_{ij} = 0 \quad (i, j = 0, 1, 2), \tag{C.2}$$

$$\mathscr{T} = \mathbf{T}_a - \mathbf{R}_b\mathbf{T}_c + \mathbf{T}_b = \mathbf{0}. \tag{C.3}$$

Based on the above constraints, we can apply extended Kalman filter (EKF) to sequentially update $(\bar{\mathbf{p}}_a, \mathbf{\Sigma}_a)$ as following:

(1) *Case 1* ($\mathscr{R}_{ij} = 0$):

Let $\mathbf{p}$ and $\mathbf{r}$ be the system state vector and measurement vector for EKF separately, we have

$$\mathbf{p} = \mathbf{p}_a \quad \text{and} \quad \mathbf{r} = (\mathbf{p}_b, \mathbf{p}_c)^{\mathrm{T}}. \tag{C.4}$$

To apply Kalman filter, the relevant observation matrix $\mathbf{M}$ is

$$\mathbf{M} = \frac{\partial \mathscr{R}_{ij}}{\partial \mathbf{p}} = \begin{bmatrix} (\mathrm{d}\mathbf{R}_{ax})_{ij} & (\mathrm{d}\mathbf{R}_{ay})_{ij} & (\mathrm{d}\mathbf{R}_{az})_{ij} & 0 & 0 & 0 \end{bmatrix}. \tag{C.5}$$

The observation error covariance matrix $\mathbf{G}$ can be computed as

$$\frac{\partial \mathscr{R}_{ij}}{\partial \mathbf{r}} = \begin{bmatrix} \frac{\partial \mathscr{R}_{ij}}{\partial \mathbf{p}_b} & \frac{\partial \mathscr{R}_{ij}}{\partial \mathbf{p}_c} \end{bmatrix} \tag{C.6}$$

$$= \begin{bmatrix} -(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{ij} & -(\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{ij} & -(\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{ij} & 0 & 0 & 0 \\ -(\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{ij} & -(\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{ij} & -(\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{ij} & 0 & 0 & 0 \end{bmatrix} \tag{C.7}$$

$$\mathbf{G} = \frac{\partial \mathscr{R}_{ij}}{\partial \mathbf{r}} \begin{bmatrix} \mathbf{\Sigma}_b & 0 \\ 0 & \mathbf{\Sigma}_c \end{bmatrix} \left(\frac{\partial \mathscr{R}_{ij}}{\partial \mathbf{r}}\right)^{\mathrm{T}}. \tag{C.8}$$

So the updated $(\bar{\mathbf{p}}_a, \mathbf{\Sigma}_a)$ by EKF is:

$$\begin{aligned}
&\mathbf{K} = \mathbf{\Sigma}_a \mathbf{M}^{\mathrm{T}} (\mathbf{G} + \mathbf{M}\mathbf{\Sigma}_a\mathbf{M}^{\mathrm{T}})^{-1},\\
&\mathscr{R}_{ij} = (\mathbf{R}_a)_{ij} - (\mathbf{R}_b\mathbf{R}_c)_{ij},\\
&\bar{\mathbf{p}}_a = \bar{\mathbf{p}}_a - \mathbf{K}\mathscr{R}_{ij},\\
&\mathbf{\Sigma}_a = (\mathbf{I} - \mathbf{K}\mathbf{M})\mathbf{\Sigma}_a.
\end{aligned} \tag{C.9}$$

(2) *Case 2* ($\mathscr{T} = \mathbf{0}$):

Define system state vector $\mathbf{p}$ and measurement vector $\mathbf{r}$ for EKF same as above, we can compute observation matrix $\mathbf{M}$ as

$$\mathbf{M} = \frac{\partial \mathscr{T}}{\partial \mathbf{p}} = \frac{\partial \mathbf{T}_a}{\partial \mathbf{p}_a}. \tag{C.10}$$

Matrix $\mathbf{G}$ is calculated as

$$\frac{\partial \mathscr{T}}{\partial \mathbf{r}} = \left[-\mathrm{d}\mathbf{R}_{bx}\mathbf{T}_c \ \ -\mathrm{d}\mathbf{R}_{by}\mathbf{T}_c \ \ -\mathrm{d}\mathbf{R}_{bz}\mathbf{T}_c \ \ -\mathbf{I}_{3x3} \ \ 0 \ \ 0 \ \ 0 \ \ -\mathbf{R}_b \begin{bmatrix}1\\0\\0\end{bmatrix} \ \ -\mathbf{R}_b \begin{bmatrix}0\\1\\0\end{bmatrix} \ \ -\mathbf{R}_b \begin{bmatrix}0\\0\\1\end{bmatrix} \right], \tag{C.11}$$

$$\mathbf{G} = \frac{\partial \mathscr{T}}{\partial \mathbf{r}} \begin{bmatrix} \mathbf{\Sigma}_b & 0 \\ 0 & \mathbf{\Sigma}_c \end{bmatrix} \frac{\partial \mathscr{T}}{\partial \mathbf{r}}^{\mathrm{T}}. \tag{C.12}$$

And the final updated $(\bar{\mathbf{p}}_a, \mathbf{\Sigma}_a)$ by EKF is:

$$\mathbf{K} = \mathbf{\Sigma}_a \mathbf{M}^{\mathrm{T}} (\mathbf{G} + \mathbf{M}\mathbf{\Sigma}_a\mathbf{M}^{\mathrm{T}})^{-1}, \tag{C.13}$$

$$\mathscr{T} = \mathbf{T}_a - \mathbf{R}_b\mathbf{T}_c + \mathbf{T}_b, \tag{C.14}$$

$$\bar{\mathbf{p}}_a = \bar{\mathbf{p}}_a - \mathbf{K}\mathscr{T}, \tag{C.15}$$

$$\mathbf{\Sigma}_a = (\mathbf{I} - \mathbf{K}\mathbf{M})\mathbf{\Sigma}_a. \tag{C.16}$$

Appendix D. Motion estimation from inverse transforms

In this appendix, we are going to talk about the updating of motion uncertainty ${}_n\mathbf{U}_k : ({}_n\bar{\mathbf{p}}_k, {}_n\mathbf{\Sigma}_k)$ from frame k to n, given two uncertainties ${}_n\mathbf{U}_0 : ({}_n\bar{\mathbf{p}}_0, {}_n\mathbf{\Sigma}_0)$ and ${}_k\mathbf{U}_0 : ({}_k\bar{\mathbf{p}}_0, {}_k\mathbf{\Sigma}_0)$. Instead of using homogeneous transformation relationship ${}_n\mathbf{H}_k = {}_n\mathbf{H}_0 * {}_k\mathbf{H}_0^{-1}$, we update ${}_n\mathbf{H}_k$ by directly using ${}_n\mathbf{H}_0 = {}_n\mathbf{H}_k * {}_k\mathbf{H}_0$ based on EKF to avoid matrix inverse of known motion transform.

Let $\mathbf{p}_a = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)^{\mathrm{T}}$, $\mathbf{p}_b = (\theta_x, \theta_y, \theta_z, s_x, s_y, s_z)^{\mathrm{T}}$, $\mathbf{p}_c = (\psi_x, \psi_y, \psi_z, q_x, q_y, q_z)^{\mathrm{T}}$, then we have the following equations:

$$\mathscr{R}_{ij} = (\mathbf{R}_a)_{ij} - (\mathbf{R}_b\mathbf{R}_c)_{ij} = 0 \quad (i, j = 0, 1, 2), \tag{D.1}$$

$$\mathcal{T} = \mathbf{T}_a - \mathbf{R}_b\mathbf{T}_c + \mathbf{T}_b = \mathbf{0}. \tag{D.2}$$

Based on the above constraints, we can apply extended Kalman filter (EKF) to sequentially update $(\bar{\mathbf{p}}_b, \mathbf{\Sigma}_b)$ as following:

(1) *Case 1* ($\mathcal{R}_{ij} = 0$):

Let $\mathbf{p}$ and $\mathbf{r}$ be system state vector and measurement vector for EKF separately, we have

$$\mathbf{p} = \mathbf{p}_b, \quad \text{and} \quad \mathbf{r} = (\mathbf{p}_a, \mathbf{p}_c)^{\mathrm{T}}. \tag{D.3}$$

To apply Kalman filter, the relevant observation matrix $\mathbf{M}$ is

$$\mathbf{M} = \frac{\partial \mathcal{R}_{ij}}{\partial \mathbf{p}} = \begin{bmatrix} -(\mathrm{d}\mathbf{R}_{bx}\mathbf{R}_c)_{ij} & -(\mathrm{d}\mathbf{R}_{by}\mathbf{R}_c)_{ij} & -(\mathrm{d}\mathbf{R}_{bz}\mathbf{R}_c)_{ij} & 0 & 0 & 0 \end{bmatrix}. \tag{D.4}$$

The observation error covariance matrix $\mathbf{G}$ can be computed as

$$\frac{\partial \mathcal{R}_{ij}}{\partial \mathbf{r}} = \begin{bmatrix} \frac{\partial \mathcal{R}_{ij}}{\partial \mathbf{p}_a} & \frac{\partial \mathcal{R}_{ij}}{\partial \mathbf{p}_c} \end{bmatrix}, \tag{D.5}$$

$$\begin{aligned} = \big[&(\mathrm{d}\mathbf{R}_{ax})_{ij} \quad (\mathrm{d}\mathbf{R}_{az})_{ij} \quad (\mathrm{d}\mathbf{R}_{ay})_{ij} \quad 0 \quad 0 \quad 0 \\ &-(\mathbf{R}_b\mathrm{d}\mathbf{R}_{cx})_{ij} \quad -(\mathbf{R}_b\mathrm{d}\mathbf{R}_{cy})_{ij} \quad -(\mathbf{R}_b\mathrm{d}\mathbf{R}_{cz})_{ij} \quad 0 \quad 0 \quad 0 \big], \end{aligned} \tag{D.6}$$

$$\mathbf{G} = \frac{\partial \mathcal{R}_{ij}}{\partial \mathbf{r}} \begin{bmatrix} \mathbf{\Sigma}_a & 0 \\ 0 & \mathbf{\Sigma}_c \end{bmatrix} \left(\frac{\partial \mathcal{R}_{ij}}{\partial \mathbf{r}} \right)^{\mathrm{T}}. \tag{D.7}$$

So the updated $(\bar{\mathbf{p}}_a, \mathbf{\Sigma}_a)$ by EKF is:

$$\mathbf{K} = \mathbf{\Sigma}_b\mathbf{M}^{\mathrm{T}}(\mathbf{G} + \mathbf{M}\mathbf{\Sigma}_b\mathbf{M}^{\mathrm{T}})^{-1}, \tag{D.8}$$

$$\mathcal{R}_{ij} = (\mathbf{R}_a)_{ij} - (\mathbf{R}_b\mathbf{R}_c)_{ij}, \tag{D.9}$$

$$\bar{\mathbf{p}}_b = \bar{\mathbf{p}}_b - \mathbf{K}\mathcal{R}_{ij}, \tag{D.10}$$

$$\mathbf{\Sigma}_b = (\mathbf{I} - \mathbf{K}\mathbf{M})\mathbf{\Sigma}_b. \tag{D.11}$$

For each $\mathcal{R}_{ij}$, we go through above process sequentially.

(2) *Case 2* ($\mathcal{T} = \mathbf{0}$):

Define system state vector $\mathbf{p}$ and measurement vector $\mathbf{r}$ for EKF same as above, we can compute observation matrix $\mathbf{M}$ as

$$\mathbf{M} = \frac{\partial \mathcal{T}}{\partial \mathbf{p}} = \frac{\partial \mathbf{T}_a}{\partial \mathbf{p}_a} \tag{D.12}$$

$$= \begin{bmatrix} -\mathrm{d}\mathbf{R}_{bx}\mathbf{T}_c - \mathrm{d}\mathbf{R}_{by}\mathbf{T}_c - \mathrm{d}\mathbf{R}_{bz}\mathbf{T}_c - \mathbf{I}_{3\times3} \end{bmatrix}. \tag{D.13}$$

Matrix $\mathbf{G}$ is calculated as

$$\frac{\partial \mathcal{T}}{\partial \mathbf{r}} = \left[0 \quad 0 \quad 0 \quad \mathbf{I}_{3\times3} \quad 0 \quad 0 \quad 0 \quad -\mathbf{R}_b \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad -\mathbf{R}_b \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad -\mathbf{R}_b \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right], \tag{D.14}$$

$$\mathbf{G} = \frac{\partial \mathscr{T}}{\partial \mathbf{r}} \begin{bmatrix} \boldsymbol{\Sigma}_a & 0 \\ 0 & \boldsymbol{\Sigma}_c \end{bmatrix} \frac{\partial \mathscr{T}}{\partial \mathbf{r}}^{\mathrm{T}}. \tag{D.15}$$

So the final updated $(\bar{\mathbf{p}}_b, \boldsymbol{\Sigma}_b)$ by EKF is:

$$\mathbf{K} = \boldsymbol{\Sigma}_b \mathbf{M}^{\mathrm{T}} (\mathbf{G} + \mathbf{M} \boldsymbol{\Sigma}_b \mathbf{M}^{\mathrm{T}})^{-1}, \tag{D.16}$$

$$\mathscr{T} = \mathbf{T}_a - \mathbf{R}_b \mathbf{T}_c + \mathbf{T}_b, \tag{D.17}$$

$$\bar{\mathbf{p}}_b = \bar{\mathbf{p}}_b - \mathbf{K}\mathscr{T}, \tag{D.18}$$

$$\boldsymbol{\Sigma}_b = (\mathbf{I} - \mathbf{K}\mathbf{M})\boldsymbol{\Sigma}_b. \tag{D.19}$$

Appendix E. Jacobian matrices used in depth updating

During the process of estimating initialized depth uncertainty $(\bar{Z}, \sigma_Z)$, we are given ${}_n\mathbf{U}_k$ and feature correspondence pair $(u,v,z) \rightarrow (u', v')$. The relevant constraint equations and Jacobian matrices used during the updating process are defined as following: Constraint equations:

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} Z(r_{11}u + r_{12}v + r_{13}) + t_x - u'(Z(r_{31}u + r_{32}v + r_{33}) + t_z) \\ Z(r_{21}u + r_{22}v + r_{23}) + t_y - v'(Z(r_{31}u + r_{32}v + r_{33}) + t_z) \end{bmatrix}. \tag{E.1}$$

Jacobian $\mathbf{M}$:

$$\mathbf{M} = \begin{bmatrix} \frac{\partial h_1}{\partial Z} \\ \frac{\partial h_2}{\partial Z} \end{bmatrix} = \begin{bmatrix} (r_{11}u + r_{12}v + r_{13}) - u'(r_{31}u + r_{32}v + r_{33}) \\ (r_{21}u + r_{22}v + r_{23}) - v'(r_{31}u + r_{32}v + r_{33}) \end{bmatrix}. \tag{E.2}$$

Jacobian of $\mathbf{h}$:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{r}} = \begin{bmatrix} \frac{\partial h_1}{\partial \phi_x} & \frac{\partial h_1}{\partial \phi_y} & \frac{\partial h_1}{\partial \phi_z} & \frac{\partial h_1}{\partial t_x} & \frac{\partial h_1}{\partial t_y} & \frac{\partial h_1}{\partial t_z} & \frac{\partial h_1}{\partial u'} & \frac{\partial h_1}{\partial v'} \\ \frac{\partial h_2}{\partial \phi_x} & \frac{\partial h_2}{\partial \phi_y} & \frac{\partial h_2}{\partial \phi_z} & \frac{\partial h_2}{\partial t_x} & \frac{\partial h_2}{\partial t_y} & \frac{\partial h_2}{\partial t_z} & \frac{\partial h_2}{\partial u'} & \frac{\partial h_2}{\partial v'} \end{bmatrix}, \tag{E.3}$$

where

$$\frac{\partial h_1}{\partial \phi_x} = Z((\mathrm{d}\mathbf{R}_x)_{11}u + (\mathrm{d}\mathbf{R}_x)_{12}v + (\mathrm{d}\mathbf{R}_x)_{13}) - u'Z((\mathrm{d}\mathbf{R}_x)_{31}u + (\mathrm{d}\mathbf{R}_x)_{32}v + (\mathrm{d}\mathbf{R}_x)_{33}), \tag{E.4}$$

$$\frac{\partial h_1}{\partial \phi_y} = Z((\mathrm{d}\mathbf{R}_y)_{11}u + (\mathrm{d}\mathbf{R}_y)_{12}v + (\mathrm{d}\mathbf{R}_y)_{13}) - u'Z((\mathrm{d}\mathbf{R}_y)_{31}u + (\mathrm{d}\mathbf{R}_y)_{32}v + (\mathrm{d}\mathbf{R}_y)_{33}), \tag{E.5}$$

$$\frac{\partial h_1}{\partial \phi_z} = Z((\mathrm{d}\mathbf{R}_z)_{11}u + (\mathrm{d}\mathbf{R}_z)_{12}v + (\mathrm{d}\mathbf{R}_z)_{13}) - u'Z((\mathrm{d}\mathbf{R}_z)_{31}u + (\mathrm{d}\mathbf{R}_z)_{32}v + (\mathrm{d}\mathbf{R}_z)_{33}), \tag{E.6}$$

$$\frac{\partial h_1}{\partial t_x} = 1, \quad \frac{\partial h_1}{\partial t_y} = 0, \quad \frac{\partial h_1}{\partial t_z} = -u', \tag{E.7}$$

$$\frac{\partial h_1}{\partial u'} = -Z(r_{31}u + r_{32}v + r_{33}) - t_z, \quad \frac{\partial h_1}{\partial v'} = 0, \tag{E.8}$$

$$\frac{\partial h_2}{\partial \phi_x} = Z((\mathrm{d}\mathbf{R}_x)_{21}u + (\mathrm{d}\mathbf{R}_x)_{22}v + (\mathrm{d}\mathbf{R}_x)_{23}) - v'Z((\mathrm{d}\mathbf{R}_x)_{31}u + (\mathrm{d}\mathbf{R}_x)_{32}v + (\mathrm{d}\mathbf{R}_x)_{33}), \tag{E.9}$$

$$\frac{\partial h_2}{\partial \phi_y} = Z((\mathrm{d}\mathbf{R}_y)_{21}u + (\mathrm{d}\mathbf{R}_y)_{22}v + (\mathrm{d}\mathbf{R}_y)_{23}) - v'Z((\mathrm{d}\mathbf{R}_y)_{31}u + (\mathrm{d}\mathbf{R}_y)_{32}v + (\mathrm{d}\mathbf{R}_y)_{33}), \tag{E.10}$$

$$\frac{\partial h_2}{\partial \phi_z} = Z((\mathrm{d}\mathbf{R}_z)_{21}u + (\mathrm{d}\mathbf{R}_z)_{22}v + (\mathrm{d}\mathbf{R}_z)_{23}) - v'Z((\mathrm{d}\mathbf{R}_z)_{31}u + (\mathrm{d}\mathbf{R}_z)_{32}v + (\mathrm{d}\mathbf{R}_z)_{33}), \tag{E.11}$$

$$\frac{\partial h_2}{\partial t_x} = 0, \quad \frac{\partial h_2}{\partial t_y} = 1, \quad \frac{\partial h_2}{\partial t_z} = -v', \tag{E.12}$$

$$\frac{\partial h_2}{\partial u'} = 0, \quad \frac{\partial h_2}{\partial v'} = -Z(r_{31}u + r_{32}v + r_{33}) - t_z. \tag{E.13}$$

References

[1] Y. Altunbasak, P. Eren, A. Tekalp, Region-based parametric motion segmentation using color information, Graph. Models Image Process. 60 (No. 1) (1998) 13–23.

[2] G. Borshukov, G. Bozdagi, Y. Altunbasak, M. Tekalp, Motion segmentation by multistage affine classification, IEEE Trans. Image Process. 6 (11) (1997) 1591–1594.

[3] I. Celasun, A. Tekalp, M. Gokcetekin, D.M. Harmandi, 2-D mesh-based video object segmentation and tracking with occlusion resolution, Signal Process.: Image Commun. 16 (2001) 949–962.

[4] J. Chraskova, Y. Kaminsky, I. Krekule, An automatic 3D tracking system with a PC and a single TV camera, J. Neurosci. Meth. 88 (2) (1999) 195–200.

[5] R. Castagno, T. Ebrahimi, M. Kunt, Video segmentation based on multiple features for interactive multimedia applications, IEEE Trans. Circuits Systems Video Technol. 8 (5) (1998).

[6] A. Azarbayejani, A.P. Pentland, Recursive estimation of motion, structure, and focal length, IEEE Trans. Pattern Anal. Mach. Intell. 17 (6) (1995) 562–575.

[7] J. Alon, S. Sclaroff, Recursive estimation of motion and planar structure, in: IEEE Proc. Computer Vision and Pattern Recognition, North Carolina, 2000.

[8] T.J. Broida, S. Chandrashekhar, R. Chellappa, Recursive estimation of 3D motion from a monocular image sequence, IEEE Trans. Aerospace Electronic Syst. 26 (4) (1990) 639–656.

[9] I.J. Cox, A review of statistical data association techniques for motion correspondence, Int. J. Comput. Vision 10 (1) (1993) 53–66.

[10] N. Diehl, Object-oriented motion estimation and segmentation in image sequences, Signal Process.: Image Commun. 3 (1991) 23–56.

[11] O. Faugeras, Three-Dimensional Computer Vision, MIT press, Cambridge, MA, 1993.
[12] Fu. Yue, A.T. Erdem, A.M. Tekalp, Tracking visible boundary of objects using occlusion adaptive motion snake, IEEE Trans. Image Process. 9 (12) (2000) 2051–2060.
[13] J. Gao, A. Kosaka, A. Kak, Interactive color image segmentation editor driven by active contour model, in: IEEE Proc. Internat. Conf. on Image Processing, Japan, 1999.
[14] C. Gomila, F. Meyer, Automatic video object generation tool: segmentation and tracking of persons in real time, Ann. Telecommun.—Ann. Telecommun. 55 (3–4) (2000) 172–183.
[15] J. Guo, J. Kim, C. Kuo, An interactive object segmentation system for MPEG video, in: IEEE Proc. Internat. Conf. on Image Processing, Kobe, Japan, 1999.
[16] C. Gu, M. Lee, Semiautomatic segmentation and tracking of semantic video objects, IEEE Trans. Circuits Systems Video Technol. 8 (5) (1998) 572–584.
[17] S. Hsu, P. Anandan, S. Peleg, Accurate computation of optical flow by using layered motion representation, in: Proc. Internat. Conf. on Pattern Recognition, Jerusalem, Israel, 1994, pp. 743–746.
[18] K. Hata, J. Ohya, F. Kishino, R. Nakatsu, Automatic extraction and tracking of complex contours, Syst. Comput. Jpn. 30 (8) (1999) 40–50.
[19] Y.S. Hung, H.T. Ho, A Kalman filter approach to direct depth estimation incorporating surface structure, IEEE Trans. Pattern Anal. Mach. Intell. 21 (6) (1999) 570–575.
[20] M. Isard, A. Blake, Condensation-conditional density propagation for visual tracking, Int. J. Comput. Vision 29 (1) (1998) 5–28.
[21] D. Jang, H. Choi, Active models for tracking moving objects, Pattern Recogn. 33 (2000) 1135–1146.
[22] S. Kamijo, Y. Matsushita, K. Ikeuchi, M. Sakauchi, Occlusion robust tracking utilizing spatio-temporal Markov random field model, in: Proc. Internat. Conf. on Pattern Recognition, 2000.
[23] C. Kervrann, F. Heitz, Statistical deformable model-base segmentation of image motion, IEEE Trans. Image Process. 8 (4) (1999).
[24] M. Kim, J.G. Jeon, J.S. Kwak, M.H. Lee, C. Ahn, Moving object segmentation in video sequences by user interaction and automatic object tracking, Image Vision Comput. 19 (5) (2001) 245–260.
[25] A. Kosaka, A. Kak, Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties, Computer Vision, Graphics, Image Process. Image Und. 56 (3) (1992).
[26] M. Lee, W. Chen, B. Lin, C. Gu, T. Markoc, S. Zabinsky, R. Szeliski, A layered video object coding system using sprite and affine motion model, IEEE Trans. Circuits System Video Technol. 7 (1) (1997).
[27] F. Meyer, P. Bouthemy, Region-based tracking using affine motion models in long image sequence, CVGIP: Image Und. 60 (2) (1994) 119–140.
[28] N. Peterfreund, Robust tracking of position and velocity with Kalman snakes, IEEE Trans. Pattern Anal. Mach. Intell. 21 (6) (1999) 564–569.
[29] Y. Rui, T. Huang, S. Chang, Digital Image/video library and MPEG-7: Standardization and Research Issues, ICASSP, Seattle, 1998.
[30] R.C. Smith, P. Cheeseman, On the representation and estimation of spatial uncertainty, Int. J. Robotics Res. 5 (4) (1986) 56–68.
[31] S. Sun, D. Haynor, Y. Kim, Semiautomatic video object segmentation using vsnakes, IEEE Trans. Circuits Syst. Video Technol. 13 (1) (2003) 75–82.
[32] A. Tekalp, Digital Video Processing, Prentice Hall PTR, 1995.
[33] C. Toklu, A. Tekalp, A. Erdem, Semi-automatic video object segmentation in the presence of occlusion, IEEE Trans. Circuits Syst. Video Technol. 10 (4) (2000) 624–629.
[34] J. Wang, E. Adelson, Representing moving images with layers, IEEE Trans. Image Process. 3 (5) (1994) 625–638.
[35] J. Weng, T.S. Huang, N. Ahuja, Motion and Structure from Image Sequences (Series in Information Science), Springer, Berlin, 1993.
[36] J.J. Wu, R.E. Rink, T.M. Caelli, V.G. Gourishankar, Recovery of the 3-D location and motion of a rigid object through camera image (an extended kalman filter approach), Int. J. Comput. Vision 3 (1998) 373–394.
[37] Y. Yao, R. Chellappa, Tracking a dynamic set of feature points, IEEE Trans. Image Process. 4 (10) (1995) 1382–1395.

[38] Z. Zhang, O.D. Faugeras, Three-dimensional motion computation and object segmentation in a long sequence of stereo frames, Int. J. Comput. Vision 7 (1992) 211–241.

[39] Z.Y. Zhang, R. Deriche, O.D. Faugeras, Q.T. Luong, A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry, Artificial Intell. 78 (1–2) (1995).

[40] Y. Zhong, A.K. Jain, M.-P. Dubuisson-Jolly, Object tracking using deformable templates, IEEE Trans. Pattern Anal. Mach. Intell. 22 (5) (2000) 544–549.

[41] S.C. Zhu, A. Yuille, Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 18 (9) (1996) 884–900.