Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2008, Article ID 328052, 14 pages
doi:10.1155/2008/328052

Research Article

Tracking of Moving Objects in Video Through Invariant Features in Their Graph Representation

O. Miller,[1] A. Averbuch,[2] and E. Navon[2]

[1] *ArtiVision Technologies Pte. Ltd., 96 Robinson Road #13-02, Singapore 068899*
[2] *School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel*

Correspondence should be addressed to A. Averbuch, amir@math.tau.ac.il

Received 21 July 2007; Accepted 10 July 2008

Recommended by Fatih Porikli

The paper suggests a contour-based algorithm for tracking moving objects in video. The inputs are segmented moving objects. Each segmented frame is transformed into region adjacency graphs (RAGs). The object's contour is divided into subcurves. Contour's junctions are derived. These junctions are the unique "signature" of the tracked object. Junctions from two consecutive frames are matched. The junctions' motion is estimated using RAG edges in consecutive frames. Each pair of matched junctions may be connected by several paths (edges) that become candidates that represent a tracked contour. These paths are obtained by the k-shortest paths algorithm between two nodes. The RAG is transformed into a weighted directed graph. The final tracked contour construction is derived by a match between edges (subcurves) and candidate paths sets. The RAG constructs the tracked contour that enables an accurate and unique moving object representation. The algorithm tracks multiple objects, partially covered (occluded) objects, compounded object of merge/split such as players in a soccer game and tracking in a crowded area for surveillance applications. We assume that features of topologic signature of the tracked object stay invariant in two consecutive frames. The algorithm's complexity depends on RAG's edges and not on the image's size.

1. INTRODUCTION

Object tracking is an important task for a variety of computer vision applications such as monitoring [1], perceptual user interfaces [2], video compression [3], vehicular navigation, robotic control, motion recognition, video surveillance, and many more. These applications require reliable object tracking techniques that satisfy real-time constraints. For example, object tracking is a key component for efficient algorithms in video processing and compression for content-based indexing [4, 5]. In MPEG-4, the visual information is organized on the basis of the video object (VO) concept, which represents a time-varying visual entity with arbitrary shape. Tracking these video objects along the scene enables individual manipulation of its shape and combines it with other similar entities to produce a scene.

A robust tracking system should meet the following challenges: tracking of multiple objects that are partially covered (occluded) by other objects, tracking a compounded object of merge and split as players in a soccer game, tracking of slow disappearance of an object from a scene and then its reappearance after several frames, tracking in a crowded area for surveillance applications. Many of the tracking techniques utilize the model shape [6] of the tracked object to overcome the errors produced by the use of an estimated motion vector. Unfortunately, an object may translate and rotate in three-dimensional space while its projection on the image plane undergoes projective transformations that cause substantial deformations in its two-dimensional shape. Furthermore, an object may change its real (original) shape in physical space, for example, in the case of a human object changing its body position. In this situation, it is necessary to decide whether the model that describes the object has to be updated, or whether the change in its shape will be considered as a transitory event. In other cases, the first frame of the sequence may consist of two different objects that are very close to each other, and the object extraction process considers them as a single object. Then, the "single" object may split, after a few frames, into two objects. Depending on the adopted based technique, the object's shape model will have to be reinitialized according to the new tracked objects in successive frames.

Producing an automatic robust tracking system is a significant challenge. There have been a lot of efforts that produce solutions to handle the tracking problem. The existing techniques in the literature can be roughly classified into the following groups of trackers: region-based, contour-based, and model-based.

1.1. Region-based trackers

An adaptive algorithm, which updates the histogram of the tracked object in order to account for changes in color, illumination, pose, and scale, was presented in [7]. The tracked object is represented as "blobs" in the back-projection [8]. This technique has a low complexity, and thus suits real-time systems. Other techniques, which utilize structural features such as texture, color and luminance, are proposed in [9–11]. For example, [9] suggested a moving object tracking algorithm that is based on a combination of adaptive texture and color segmentation. They use Gibbs-Markov random field to model the texture, and a two-dimensional Gaussian distribution to model the color. The tracked target is obtained by a probabilistic framework of the texture and color cues at region level and by adapting both the texture and color segmentation over time. Similar techniques were used earlier in [10, 11]. However, when fast motion of the object and significant change in behavior are allowed, color- or texture-based techniques are not sufficient to detect a long-term tracking. Region-based algorithms to deal with the above were suggested in [12, 13]. The method in [12] is based on a modified version of Kalman's filter. A four-step tracking technique using motion projection, marker extraction, clustering and region merging was suggested in [13]. The motion projection represented by a linear motion model is calculated on the reliable parts of the projected object. These parts are called marker points and their extractions are based on the assumption about the relationship between the projected and real objects. Then, starting from these markers, a modified watershed transformation followed by a region-merging algorithm produces the complete segmentation of the next frame. The advantage of the techniques in [12, 13] is in their abilities to deal with significant changes in the object's regions.

1.2. Contour-based trackers

These algorithms detect and track only the contour of the object. For example, the snake algorithm [6] does it by using a converged snake in the reference frame as the initial snake in the successive frame. However, the algorithm blurs the image to disperse gradient information. Therefore, it has a limited search area and it is unable to track changes in the contour that lie outside the range of the blurred gradient operator. As a consequence, this method is effective only when the motions and changes in the object's shape between consecutive frames are small. To overcome this limitation, some use dynamic programming that increases the search area of the snake. Temporal information to bias the snake toward the shape of the segmentation in the previous frame is used in [14]. This method improves the shape memory of the snake but does not adequately track large-scale object movements or significant changes in its shape. Conversely, [15] suggested a tracker, which handles significant changes in the shape. It partitions the object's contour into several curves and estimate the motions of each curve independently of the others. Then, the predicted location of the complete contour is obtained by using dynamic programming. A particle filter-based subspace projection method, which neither causes blur nor demands an expensive dynamic programming, is presented in [16].

1.3. Model-based trackers

A generic solution to object tracking is still a challenging problem. Therefore, model-based techniques, which demand a priori information regarding the object's shape and type, were developed for suitable applications. For example, [17] suggested a model, which is based on edge detector, to extract moving edges that are grouped together by a predefined model shape. A similar concept was used in [18], which groups the edges by using the Hough transform. In addition, the use of deformable templates has been very common in model-based techniques. A deformable template enables to track large movements and changes, and also provides global shape memory for the tracked object. For example, a known model of a key is defined in [19]. This approach allows both affine transformations of the entire key and local deformations in the form of variations in the notches of the key. A similar approach of deformable templates was used in [20] for tracking a human hand by global and local deformation templates, which were defined for movement changes and for finger motions, respectively. Recently, a dynamic-model-based technique [21], which decreases the constraint limitation by having a predefined model, was suggested. It uses a hierarchy of separate deformation stages: global affine deformations, local (segmented) affine deformations, and snake-based continuous deformations. This approach provides a shape memory update of the object's model and thus, enables the snake algorithm to be used for final contour refinement after each frame.

Our proposed algorithm is classified as a contour-based technique. The algorithm gets as an input the contour (curve) of the moving object to be tracked and two consecutive frames I_t and I_{t+1} in times t and $t+1$, respectively. Initially, we apply a still-segmentation process on I_t and I_{t+1}. Region adjacency graphs (RAGs), denoted by G_t and G_{t+1}, are the constructed data structures from the segmentation of I_t and I_{t+1}, respectively. Then, the contour of the extracted object in G_t is segmented into subcurves while interior junctions are marked. Each subcurve represents a different homogeneous region in G_t. The interior junctions connect between two different subcurves. These junctions are called important (easy) points to track. Motion estimation of these junctions is performed where the search area is the edges in G_{t+1}. For each pair of tracked junctions in G_t, there exists a small number of "candidate paths," which connect the estimated pair in G_{t+1}. These paths are obtained by an algorithm that finds the k shortest path between two nodes in weighted and directed graph. We claim that only one of

the candidate paths accurately represents the single-tracked subcurve between a pair of junctions. Then, the construction of the tracked object is actually a process that matches between a single edge (signature) in G_t and a set of candidate paths in G_{t+1}.

The matching process has a low complexity due to the limited number of candidate paths. The tracked contour is constructed from all the matched paths between pairs of estimated junctions. We show that the probability of matching the right path is proportional to the number of points that participate in the matching process.

In contrast to other contour-based techniques such as in [6, 14, 15], our algorithm does not use the snake algorithm [6] to obtain the predicted curve of the object. Therefore, the suggested algorithm does not have limitations on the object's motion or changes in the object's pose. Furthermore, it does not require a priori information about its shape, type, or motion as in [17, 19, 20].

The rest of the paper is organized as follows. Section 2 introduces the notation and outlines the main steps of the algorithm. Section 3 describes the object's contour segmentation. The estimation of the junction's motion and the algorithm for finding the candidate paths is given in Section 4. Section 5 presents the matching process that constructs the tracked contour. Implementation and complexity analysis is given in Section 6. Experimental results are given in Section 7.

2. NOTATION AND OUTLINE OF THE ALGORITHM

The following notations are used; see Figure 1 for illustration. The flow of the algorithm is given in Figure 2. Let I_t and I_{t+1} be the inputs of the source frame and its consecutive frame, respectively. We denote by $\{R_i^t\}$, $i = 1,\ldots,n_t$ and $\{R_j^{t+1}\}$, $j = 1,\ldots,n_{t+1}$ the sets of n_t and n_{t+1} non-overlapping regions that are generated by a still image segmentation (Section 3.1 using [22–24]) of I_t and I_{t+1}, respectively. Region adjacency graphs (RAGs) $G_t = (V_t, E_t)$ and $G_{t+1} = (V_{t+1}, E_{t+1})$ are the data structures that represent the segmentation of I_t and I_{t+1}, respectively. The nodes of the RAGs represent the regions R_i^t, $i = 1,\ldots,n_t$. An edge $e(i,j) \in E_t$ represents a shared common boundary of R_i^t and R_j^t such that $(x,y) \in e(i,j)$, for all $(x,y) \in R_i^t \cap R_j^t$. The same is true for $G_{t+1} = (V_{t+1}, E_{t+1})$.

Let C_t be the closed input curve that belongs to the object's contour to be tracked in I_t such that $C_t \subseteq E_t$ and let C_{t+1} be the tracked contour in I_{t+1}. We assume that $C_{t+1} \subseteq E_{t+1}$. The "important" points in C_t are called *junctions* (see Section 3.2). They are denoted by J_k, $k = 1,\ldots,N$, which is the set of the N junctions on the curve C_t. Thus, C_t (source) is segmented into subcurves, denoted by $P_t^{k,k+1}$, $k = 1,\ldots,N$, where the indices that enumerate the junctions are $k = 1,\ldots,N-1$ and the Nth junction is connected to 1. Each subcurve $P_t^{k,k+1}$ is actually the edge that connects the pair J_k and J_{k+1}. The corresponding points of J_k, $k = 1,\ldots,N$ in I_{t+1} are denoted by S_k, $k = 1,\ldots,N$, such that each matched point in S_k corresponds to a junction in J_k. In other words, a set of matched points in J_k, $k = 1,\ldots,N$, in I_{t+1} will be detected by a block-matching process (Section 4.1) that is based on SAD minimization of the matched squared errors between J_k, $k = 1,\ldots,N$, and a searched area in I_{t+1}. The set of matched points in I_{t+1} is S_k, $k = 1,\ldots,N$.

The construction of C_{t+1} is achieved by the application of the matching procedure (Section 5.1) that measures the similarity between the edges of I_t and I_{t+1}. We will show (Section 5.1) that each subcurve in C_t (the edge between J_k and J_{k+1}) has a single matched edge in C_{t+1} (the edge that connects S_k and S_{k+1}). However, several edges may connect the matched pointsS_k and S_{k+1}. We call this set of edges "candidate paths" (destination), denoted by $P_{t+1}^{k,k+1}$, $r = 1,\ldots,R_k$, where R_k is the number of candidate paths between each J_k and J_{k+1}. We say that $C_{t+1} \subseteq P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$. There exists r_k, $1 \le r_k \le R_k$, such that a path $P_{t+1,r_k}^{k,k+1}$ that satisfies $P_{t+1,r_k}^{k,k+1} \in C_{t+1}$ is called a *matched path*, denoted by $mp_{t+1}^{k,k+1}$. Then, $C_{t+1} = \cup_{k=1,\ldots,N-1} mp_{t+1}^{k,k+1}$. In other words, the new object's curve C_{t+1} is formed by sets of matched paths $mp_{t+1}^{k,k+1}$ between S_k and S_{k+1}, $k = 1,\ldots,N$.

3. CURVE SEGMENTATION

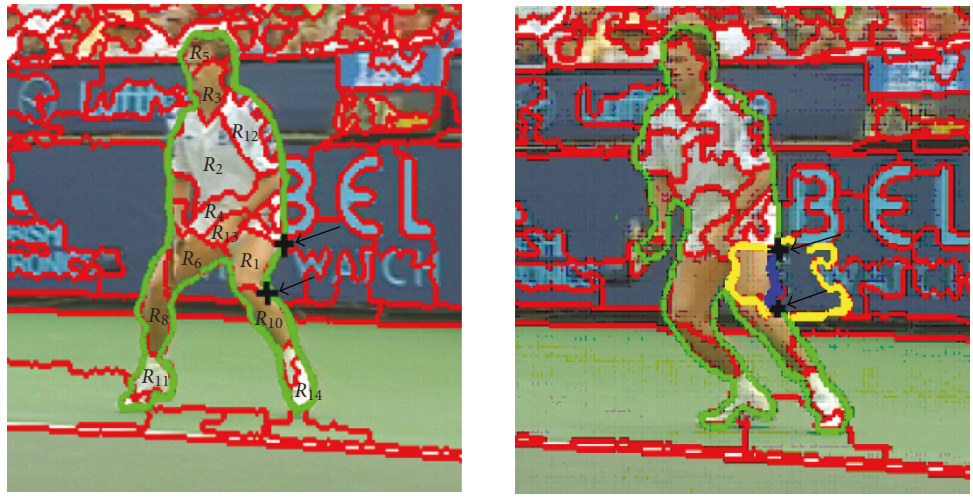
Assume we have the two input frames I_t and I_{t+1} and an object's contour C_t. Segmentation of the object's contour (curve) C_t into subcurves is required since the object may translate and rotate in three-dimensional space, while its projection onto the image plane causes substantial deformations in its two-dimensional shape. Therefore, it is better to estimate separately the motion of each subcurve rather than estimating the motion of the entire object's contour. The segmentation of the object's contour into subcurves is achieved by finding a set of junctions, which connect the subcurves between themselves and separate between homogenous regions. In (Section 3.2), we justify why these junctions are well-tracked points that faithfully represent the rest of the points that belong to C_t. These junctions comprise the *signature* of the tracked object that will characterize it uniquely.

In order to partition the object's curve into subcurves, we utilize the still-segmentation results of I_t. We require from the still segmentation to produce homogeneous regions such that only the boundaries of the regions overlap each other.

3.1. Image segmentation

The performance of the algorithm is directly affected by the number of segments from the two input frames I_t and I_{t+1}. We want produce a minimal number of segmented regions while preserving the homogeneity criteria. For this purpose, we apply the segmentation algorithm in [22–24].

This algorithm uses the watershed algorithm [25] followed by an iterative merging process, which generates local thresholds. The watershed algorithm gets the gradients of a gray-scale image of the input color image. The image is considered as a topographic relief. By flooding the topographic surface from its lowest altitude, the algorithm defines lakes and dams. Dams are watershed lines that separate adjacent lakes. When the whole surface is immersed,



(a) (b)

Figure 1: (a) Still segmentation of I_t. C_t (source) is the green closed curve. The red contours are the segmented boundaries. The black arrows point to two different junctions J_1 and J_2. (b) Still segmentation of I_{t+1}. C_{t+1} is the tracked green closed curve and the black arrows point to the matched points S_1 and S_2 that correspond to the junctions J_1 and J_2, respectively. The yellow curves are the paths of $P^{1,2}_{t+1,r}$, $r = 1,\ldots,3$ and the blue curve is the matched path denoted by $mp^{1,2}_{t+1}$.

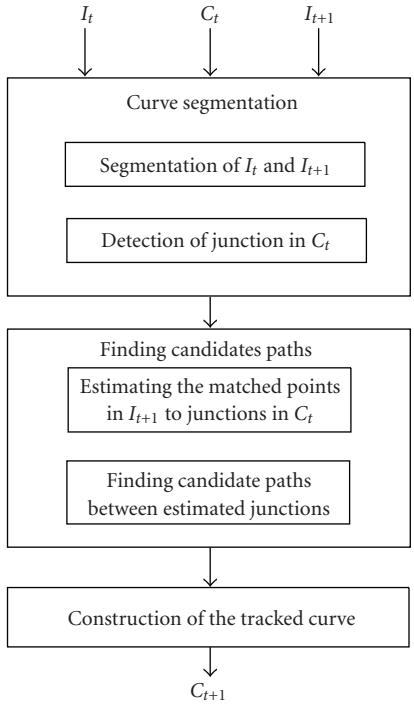


Figure 2: Flow diagram of the proposed algorithm where I_t and I_{t+1} are two consecutive input frames, and C_t is the input object's contour to be tracked. C_{t+1} is the tracked contour in I_{t+1}.

the image is divided into lakes. Each lake represents a region. However, due to its sensitivity to weak edges, the watershed algorithm generates oversegmented output. Therefore, its output is used as an initial guess for the merging process phase, which aims at reducing the number of segments. The merging process is done iteratively. At any iteration during the merging process, the most similar adjacent regions are merged. The similarity between any pair of adjacent regions is measured by the outcome of a dissimilarity function [22–24]. Local thresholds are derived by an automatic process, where local information is taken into consideration. Any threshold refers to a specific region and its surroundings. The number of thresholds that defines the final regions is known only when the process is terminated. The output of the segmentation is represented by R^t_i, $i = 1,\ldots,n_t$, which partitions I_t. These partitions construct the initial data structures that we use during the entire duration of the algorithm. Recall that $G_t = (V_t, E_t)$ (Section 2) is an undirected graph that represents the partition of I_t. The region R^t_i is represented by a node $v_i \in V_i$. An edge $e(i, j)$ exists only if R^t_i and R^t_j are adjacent, where adjacent regions share a boundary. The edge $e(i, j)$ contains all the pixels that lie between R^t_i and R^t_j. Hence, all the pixels of the segmented boundaries are represented by E_t. The same is true for the graph $G_{t+1} = (V_{t+1}, E_{t+1})$.

3.2. Detection of a junction

The proposed algorithm first tracks the "important" points (junctions) J_k, $k = 1,\ldots,N$. We will show here that these points are sufficient to produce reliable tracking without the need to have the rest of the points in C_t.

We assume that pixels in homogeneous areas are difficult to track while pixels in high-textured areas, which are characterized by having more content, are more likely to be well tracked. A method to identify good features to track was introduced in [26]. This method derives a set of interesting points by simultaneously tracking them. The points that optimize the trackers' accuracy are chosen as good feature.

The algorithm mainly detects corners as well-tracked points. Hence, corners are considered important points to track.

A corner is defined as a meeting point of two (or more) straight edge lines [26–29]. Following this definition, each corner contains at least two different edges and at least two different regions, which together create two-dimensional structures. Since real images do not contain geometric shapes only or rigid objects, corners are related to curvature on the contour. L-junction, T-junction, and X-junction are different corner types, where the number of regions that meet in the corner determines the corner type. Examples of a T-junction profile and corners in a synthetic image are illustrated in Figure 3.

The corners, as a source for rich information, are used as an anchor points for tracking. Corners can be tracked with high accuracy as it is demonstrated in [26, 30]. In [31, 32] corner tracking is used for robot homing and low bit rate video codec, respectively.

Since corners are well-tracked points, we are motivated to associate the important points in C_t with corners. Hence, the important points (which are called junctions) are defined as follows.

Definition 1. Let C_t be the contour of a given object. Let R_k^t, $k = 1,\ldots,N$, be the subset of R_i^t, $i = 1,\ldots,n_t$, which is represented by $G_t = (V_t, E_t)$. Assume N ($N \le n_t$) regions of the object intersect C_t. The N regions are renumbered $1,\ldots,N$ such that R_k^t is adjacent to R_{k-1}^t and R_{k+1}^t for all $k = 1,\ldots,N$. A junction J_k, $k = 1,\ldots,N$, on C_t is defined as $J_k \triangleq (x, y)$, where $(x, y) \in C_t$ and $(x, y) \in e(k, k+1)$, $e \in E_t$.

In other words, a junction is defined as a point on C_t where two interior segments of the object meet. Junctions are distinguished from corners by the fact that they are not necessarily represented as a meeting point of straight lines and do not necessarily represent a curvature in C_t (see Figure 4). However, like corners, junctions contain at least two different edges (lines). They represent a meeting point of at least two different homogenous regions of the object, while a corner may represent only homogeneous region of the object. They are considered as good features and are used in our application as "important" anchor points to base the tracking on them.

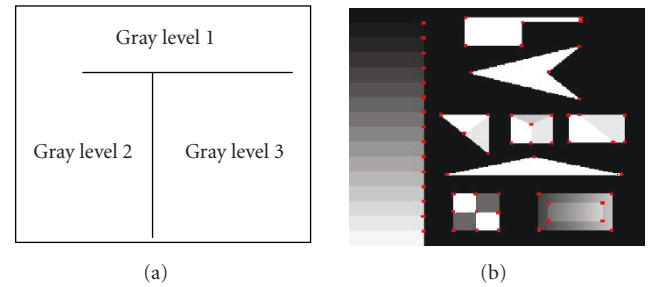


FIGURE 3: (a) Profile of T-junction. (b) Corner map (red dots) of a synthetic image.

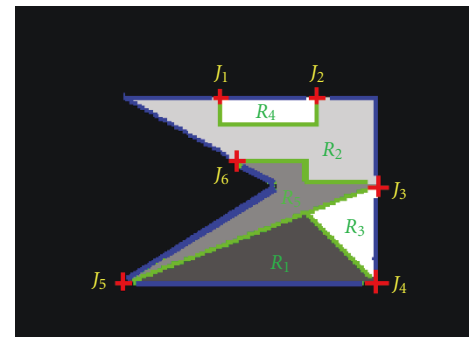


FIGURE 4: Syntactic illustration of detected junctions in C_t. The blue curve is the object contour C_t. Each homogeneous region in C_t is bounded by a green curve. The red crosses on C_t represent a set of six junctions J_k, $k = 1,\ldots,6$. Each connection between two consecutive junctions J_k and J_{k+1}, represents a homogeneous region. Each junction J_k, $k = 1,\ldots,6$ is a connection between two (or more) segmented regions R_i and R_j, $i, j = 1,\ldots,5$ and $i \neq j$. The two top corners are the curvature regions of C_t that do not satisfy Definition 1.

4. FINDING THE CANDIDATE PATHS

In Section 3, we discussed the advantages of having junctions in C_t as well-tracked points. The junctions enable to find a set of matched points S_k, $k = 1,\ldots,N$ in I_{t+1} and the candidate paths $P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$ for the construction of C_{t+1}.

The construction of the object's contour C_{t+1} in I_{t+1} relies on the fact that the set J_k, $k = 1,\ldots,N$, is defined as connected points between homogeneous regions in C_t (see Figure 4). Any pair of consecutive junctions J_k and J_{k+1} is connected by a subcurve $P_t^{k,k+1}$. This subcurve is an edge in G_t that represents the boundary of a homogeneous region. As defined in Section 2, S_k, $k = 1,\ldots,N$, is a set of matched points (in G_{t+1}) to set J_k, $k = 1,\ldots,N$ (in G_t). Each consecutive pair S_k and S_{k+1} can be connected by several paths which are edges in G_{t+1} (see Figure 1). However, based on our image segmentation assumption, the contour of the tracked object C_{t+1} is an integral part of the segmentation outputs such that $C_{t+1} \subseteq E_{t+1}$. Since all possible paths between S_k and S_{k+1}, $k = 1,\ldots,N$, satisfy $E_{t+1} = \cup_{k=1,\ldots,N,\, r=1,\ldots,R_k} P_{t+1,r}^{k,k+1}$ we get that $C_{t+1} \subseteq \cup_{k=1,\ldots,N, r=1,\ldots,R_k} P_{t+1,r}^{k,k+1}$. After these sets of paths (candidate paths) between each pair in S_k are obtained, the construction of C_{t+1} is done by the application of the matching process between a single edge $P_t^{k,k+1}$ to the set $P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$. These paths mean that we actually transform the tracking of the entire curve into a problem of how to identify match between edges.

This section describes how to find the matched points S_k for J_k, $k = 1,\ldots,N$ in E_{t+1} (Section 4.1). Then, we describe the algorithm that finds candidate paths $P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$ between any pair of matched points S_k and S_{k+1}.

4.1. Finding the corresponding junctions

The match between regions in consecutive frames is done by the application of a block-matching process on the set of junctions $J_k,\ k = 1,\ldots,N$. This set contains information about the connected points between the object regions $R_i^t,\ i = 1,\ldots,n_t$, that intersect C_t. Although we assume nothing about the object's motion, the overall structure of the object is generally preserved between two consecutive frames. The interior regions of the object do not radically change and so their connecting points. Therefore, the information that is stored in J_k, which is contained in E_t, should also exist in the object's boundaries in E_{t+1}.

Unlike traditional motion estimation techniques, where the search area is usually a predefined squared window, we utilize the data structure of G_{t+1} to adaptively determine the search area of each point in $J_k,\ k = 1,\ldots,N$. The still segmentation of I_{t+1} enables to define the set of pixels in the edges E_{t+1} of the RAG G_{t+1} as the search area for finding the corresponding matched point S_k. Thus, only a small portion of the whole image participates in the search process (block matching), which reduces the complexity ($O(|E_{t+1}|)$) to be dependent only on the graph edges.

The center of the search window is located in the coordinates of each $J_k,\ k = 1,\ldots,N$. Our goal is to find the motion vector that minimizes the matching error for a given junction J_k. It is being done through a common block matching procedure. The matching error between the block that is centered in $(x_0^k, y_0^k) \in J_k$ and has a matched point S_k is

$$\mathrm{SAD}_{(x,y)}(x_0^k, y_0^k) = \sum_{j=-B/2}^{B/2} \sum_{i=-B/2}^{B/2} \left| \left(I_t(x_0^k, y_0^k) - I_{t+1}(x+i, y+j)\right) \right|, \quad (1)$$

where $B \times B$ is the block size. Among all the searched points in $(x, y) \in E_{t+1}$, the matched point is assigned to $S_k,\ k = 1,\ldots,N$ that minimizes the matching error score

$$S_k \triangleq \underset{(x,y)\in E_{t+1}}{\arg\min} \mathrm{SAD}_{(x,y)}(x_0^k, y_0^k). \quad (2)$$

The $\mathrm{SAD}_{(x,y)}(x_0^k, y_0^k)$ is computed for each $J_k,\ k = 1,\ldots,N$. Then, we obtain the set of *matched points* $S_k,\ k = 1,\ldots,N$. Figures 8(c) and 8(d) illustrate the sets $J_k,\ S_k,\ k = 1,\ldots,N$, respectively. We anticipate that $S_k \subset C_{t+1}$ for $k = 1,\ldots,N$. Therefore, the curve C_{t+1} of the tracked object is assumed to pass through the matched points. If only one curve passes through the matched junctions, the algorithm is terminated and this curve is considered as the object's contour C_{t+1}. However, this is not always the case in real and inhomogeneous images. Several curves may pass through each pair of the matched points (see Figure 1). Section 4.2 describes how to find these curves, based on the algorithm that finds the k shortest paths between two nodes in a given graph.

4.2. Finding the k shortest paths (candidate paths)

The RAG G_{t+1} is transformed into a connected and undirected weighted graph $G'_{t+1} = (V'_{t+1}, E'_{t+1})$ as follows. Initially, any pixel $(x, y) \in E_{t+1}$ is represented by a node in G'_{t+1}. In other words, each pixel on the boundaries of the segmentation is a node in G'_{t+1}. Every two nodes in G'_{t+1} are connected by an edge $e \in E'_{t+1}$ if they are adjacent. Thus, G'_{t+1} is another representation of the segmentation while G_{t+1} focuses on regions and their connections, and G'_{t+1} focuses on the boundaries of the segmentation. Since the matched points satisfy $S_k \subseteq E_{t+1}$, they are nodes in G'_{t+1}. Consequently, all the paths $P_{t+1,r}^{k,k+1},\ r = 1,\ldots,R_k$, between S_k and S_{k+1}, that have to be found, are exactly the paths in G'_{t+1} between S_k and S_{k+1}.

Assume that G'_{t+1} is a connected graph. Then, at least one path exists between any pair of nodes. The number R_k of paths between two given nodes S_k and S_k in G'_{t+1} can be big. Since the length of $mp_{t+1}^{k,k+1}$ has to be approximately the length of the source path $P_t^{k,k+1}$, we define $L_{k,k+1}$ to be the maximal length of the candidate paths $P_{t+1,r}^{k,k+1},\ r = 1,\ldots,R_k$. $L_{k,k+1}$ is initialized to be twice the length of $P_t^{k,k+1}$. Hence, finding $P_{t+1,r}^{k,k+1},\ r = 1,\ldots,R_k$, which are shorter than $L_{k,k+1}$, is equivalent to the problem that lists all the paths that connect a given source-destination pair in a graph shorter than a given length.

The proposed algorithm is based on [33] that finds the k shortest paths. The k shortest path algorithm lists k paths, which connect a given source-destination pair in a directed graph with minimal total length. The main idea is to construct a data structure from which the sought after paths can be listed immediately by focusing on their unique representation. The data structure is constructed by handling separately the edges of the shortest-path-tree from those which are not. By differentiating between the edges, any path can be represented only by edges, which are not in the shortest-paths-tree. In addition to the implicit representation of a path, any path can be represented by its father and an additional edge. The father of a path differs from it only by one edge, which is the last edge among all the edges that are not in the shortest-paths-tree. An order-path-tree is constructed from this representation and the sought-after paths are constructed through its use. Before we describe the construction of the data structure we transform G'_{t+1} by this technique, which is described in [34], to become a directed graph denoted by $G''_{t+1} = (V'_{t+1}, E''_{t+1})$. The direction of the sought-after paths is from S_k to S_{k+1}.

We denote by $head(e)$ and $tail(e)$ the two endpoints of an edge $e \in E''_{t+1}$, which is directed from $tail(e)$ to $head(e)$. The length of an edge e is denoted by $l(e)$. An example of a directed graph with lengths attached to its edges is shown in Figure 5(a). The length of each edge in G''_{t+1} is initialized to 1. The length of a path $p \in G''_{t+1}$, which is the sum of its edge lengths, is denoted by $l(p)$. The length of the shortest path from S_k to S_{k+1} is denoted by $\mathrm{dist}(S_k, S_{k+1})$. We find the shortest path from each vertex $v \in G''_{t+1}$ to S_{k+1}. The set of all these paths generates a single-destination shortest path tree, denoted by T (see Figure 5(b)). From the construction of T, the edges in G''_{t+1} are divided into two groups. The first group consists of all the edges in T while the second contains all the edges that are not in T. Each edge $e \in G''_{t+1}$ is assigned a value, denoted by $\delta(e)$, that

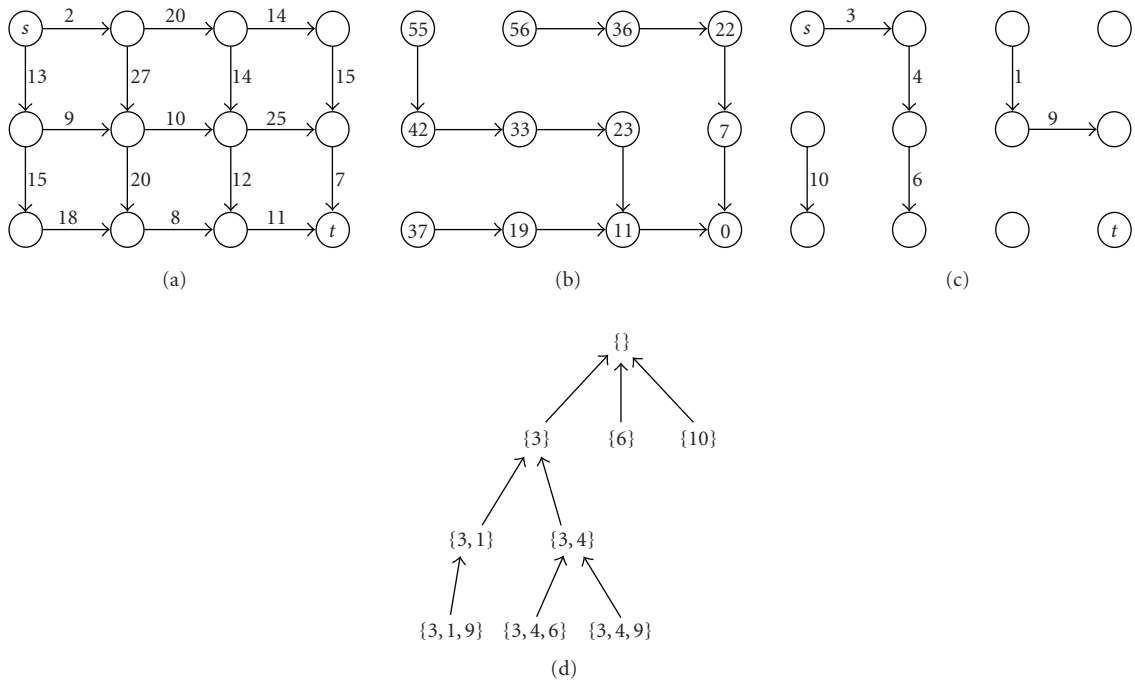


Figure 5: (a) A directed graph G with different edge lengths. The nodes which are marked by "s" and "t" are the source s_k and destination s_{k+1}, respectively. (b) The single-destination shortest path tree T. (c) The values of $\delta(e)$ of all the edges in $G - T$. (d) The order path tree is constructed by the father-son representation while only sidetracks are used.

measures the difference between $\text{dist}(tail(e), S_{k+1})$ and the shortest path from $tail(e)$ to S_{k+1} that contains e. In other words, $\delta(e)$ measures the lost distance caused by adding e to the shortest path. $\delta(e)$ is defined as

$$\delta(e) \triangleq l(e) + \text{dist}(head(e), S_{k+1}) - \text{dist}(tail(e), S_{k+1}). \quad (3)$$

Then, for any $e \in T$, we get that $\delta(e) = 0$ and for any $e \in G''_{t+1} - T$, $\delta(e) > 0$. We call the edges with $\delta(e) > 0$ (the edges in the second group) "sidetrack" edges. Examples of all $e \in G''_{t+1} - T$ and their $\delta(e)$ values are shown in Figure 5(c).

Any path $p \in G''_{t+1}$ is described by a sequence of edges that contains edges in T and sidetrack edges. Listing only the sidetracked edges was found to be a unique representation of p since every pair of nodes, which are the endpoints of two successive sidetrack edges in p, is uniquely connected by the shortest path between them (from edges in T). If p does not contain sidetracks, p is the shortest path in T. Consequently, the length of p, which connects the nodes S_k and S_{k+1}, can be computed as the sum of $\text{dist}(S_k, S_{k+1})$ and the length of its sidetracks. From the fact that any path can be represented only by its sidetracks, an additional interpretation can be used. Let $\text{prepath}(p)$ be the sidetracks of p except for the last one. We call the path, which is defined by the set of $\text{prepath}(p)$, the father of p. Any path p with at least one sidetrack can be represented by its $\text{prepath}(p)$ and by its last sidetrack. From the father-son relation, we can construct an order-path-tree (see Figure 5(d)) in which all the paths in G''_{t+1} are represented. The order path tree contains only the edges in $G''_{t+1} - T$ since the edges in T are already stored in an appropriate data structure (which is T itself) that fits the implicit representation. This data structure, which is described next, is constructed from different heaps into a final graph. All the paths $P^{k,k+1}_{t+1,r}$, $r = 1, \ldots, R_k$, in G''_{t+1} will be represented by paths from the final graph.

We denote each vertex $v \in G''_{t+1}$ by $\text{out}(v)$ of the edges in $G''_{t+1} - T$ with a tail in v. For each $v \in G''_{t+1}$, we construct a heap $H_{\text{out}}(v)$ from the edges in $\text{out}(v)$. Each node in $H_{\text{out}}(v)$ has at least two sons, while the root has only one. $H_T(v)$ is a heap of v that contains all the roots of $H_{\text{out}}(w)$, where w is on the path from v to S_{k+1}. $H_T(v)$ is built by merging the root of $H_{\text{out}}(v)$ into $H_T(\text{next}_T(v))$, where $\text{next}_T(v)$ is the next node that follows after v on the path from v to S_{k+1} in T (Figure 6). We formed $H_{G''_{t+1}}(v)$ by connecting each node w in $H_T(v)$ to the rest of the heap $H_{\text{out}}(w)$ except for the two nodes it points to in $H_T(v)$. By merging all $H_{G''_{t+1}}(v)$ for each $v \in G''_{t+1}$, we get a direct acyclic graph $D(G''_{t+1})$ (see Figure 7(a)). We denote by $h(v)$ the root of $H_{G''_{t+1}}(v)$ and use $\delta(v)$ instead of $\delta(e)$, where e is the edge in G''_{t+1} that corresponds to v.

$D(G''_{t+1})$ is augmented to the *path graph*, that is denoted by $P(G''_{t+1})$ (see Figure 7(b)). The nodes of $P(G''_{t+1})$ belong to $D(G''_{t+1})$ with the root $r = r(S_k)$ as an additional node. The nodes of $P(G''_{t+1})$ are unweighted but the edges are. Three types of edges exist in $P(G''_{t+1})$. (1) The first type is the edges in $D(G''_{t+1})$. Each (u, v) has a length $\delta(v) - \delta(u)$. (2) The second type is the edges from v to $h(w)$, where $v \in P(G''_{t+1})$ corresponds to an edge $(u, v) \in G''_{t+1} - T$. These edges are called *cross-edges*. (3) A single edge between r and $h(w)$ with length $\delta(h(S_k))$represents the third type. From the construction process, there is one-to-one correspondence between the paths starting from $r \in P(G''_{t+1})$ and all the

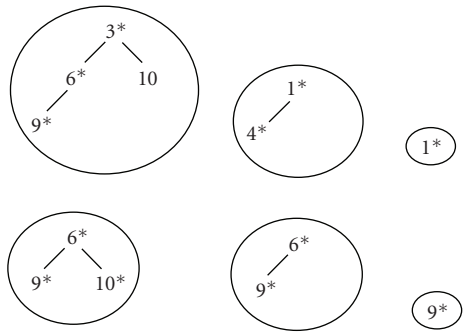


FIGURE 6: $H_T(v)$ of each node in G in Figure 5(a), where out(v) is not empty. For any $v \in G$ in Figure 5, out(v) contains only one node, therefore, $H_G(v)$ is equal to $H_T(v)$.

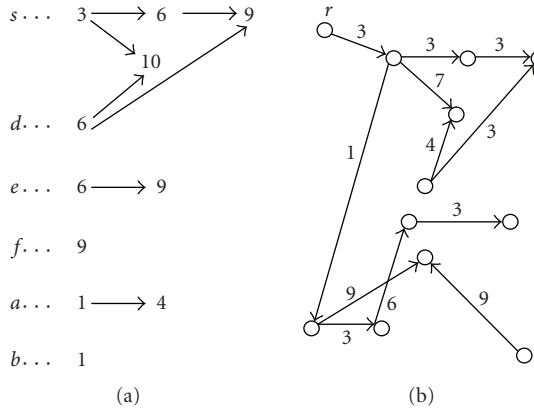


FIGURE 7: (a) $D(G)$. It has a node for each node that is marked by $(*)$ in Figure 6. The nodes that are marked by $(*)$ are the nodes that were updated by the insertion of the root of out(v) into $H_T(\text{next}_T(v))$. (b) $P(G)$.

paths from S_k to $S_{k+1} \in G''_{t+1}$. To prove this claim, we have to show that any path $p \in G''_{t+1}$ corresponds to a path $p' \in P(G''_{t+1})$ that starts from r, and any path p' from $r \in P(G''_{t+1})$ corresponds to a path $p \in G''_{t+1}$, which is represented by its sidetracks.

Next we outline the proof that any path p' from $r \in P(G''_{t+1})$ corresponds to a path $p \in G''_{t+1}$. We list for any path $p' \in P(G''_{t+1})$ a sequence of sidetracks, which represents its corresponding path $p \in G''_{t+1}$, as follows. For any cross-edge in p', the edge in G''_{t+1}, which corresponds to the tail of the cross edge, is added to the sequence. The last edge added is the edge in G''_{t+1} that corresponds to the last vertex in p'. Since each node in $P(G''_{t+1})$ corresponds to a unique edge in $G''_{t+1} - T$, the sequence of edges, which is formed to represent p, consists only of sidetrack edges.

Given the representation of all paths in G''_{t+1}, we construct the heap $H(G''_{t+1})$ in order to list only the sought-after paths, which are shorter than the given threshold. $H(G''_{t+1})$ is constructed by forming a node for each path in $P(G''_{t+1})$ rooted at R. The parent of a path is the path that is achieved by the removal of the last sidetrack. The weights of the nodes are the length of the paths. From the construction, each son is shorter than its father, and the weights are heap-ordered.

Finally, we apply the length-limited depth first search (DFS) [27] on $H(G''_{t+1})$. The length-limited DFS is the regular DFS algorithm with the following modification. If the weight of the current node is bigger than $L_{k,k+1}$, the process is regressed and continues with the node that was reached before the current node. The search results are the set of nodes that reached weights smaller than $L_{k,k+1}$. Then, we translate the search results to a full description of the paths with the representation discussed above (any path is described as a sequence of edges in T and edges in $G''_{t+1} - T$). This translation generates all the candidate paths $P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$ between S_k and S_{k+1}, that are shorter than $L_{k,k+1}$.

5. CONSTRUCTION OF THE TRACKED CONTOUR

We claim that there exists a strong similarity between $mp^{k,k+1}_{t+1} \subseteq P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$, and the original (source) path $P^{k,k+1}_t$ in C_t. This similarity exists due to the fact that the segmentation of the contour C_t represents homogeneous subcurves (edges) $P^{k,k+1}_t$, $k = 1,\ldots,N$ (Section 3.2), by the homogeneity criteria. Homogeneous subcurves in C_t are likely to preserve their contents between consecutive frames. Furthermore, the candidate path $P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$, represents different homogeneous edges (see Section 3.1). Thus, even if the content of the edge is changed such that $P^{k,k+1}_t$ becomes different from $mp^{k,k+1}_{t+1}$, it is unreasonable that $P^{k,k+1}_t$ will transform its content to be similar to one in $P^{k,k+1}_{t+1,r} - mp^{k,k+1}_{t+1}$, $r = 1,\ldots,R_k$.

We are given the candidate paths $P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$, $k = 1,\ldots,N$. The construction of the tracked curve C_{t+1} is transformed into a process that matches between the edges. The tracked contour C_{t+1} will be constructed by the matched paths $mp^{k,k+1}_{t+1}$, $k = 1,\ldots,N$, which are independent of each other. All its candidate paths $P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$ between each pair of successive points S_k and S_{k+1} are listed. Then, the path which is the most similar path to $P^{k,k+1}_t$ (the connection between J_k and J_{k+1}) among all $P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$, is considered as the *matched path* $mp^{k,k+1}_{t+1} \in C_{t+1}$.

5.1. Match between paths

The process that finds $P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$, $k = 1,\ldots,N$, is based only on the segmented boundaries. On the other hand, the detection of the matched paths $mp^{k,k+1}_{t+1}$ is performed using the original input contour C_t. Since $P^{k,k+1}_t$ is composed of one homogeneous region, not all its points are needed in the matching procedure.

Let $SP^{k,k+1}_t$ be a subset of β independent points in $P^{k,k+1}_t$. A search for the best match between any pixel $(x, y) \in SP^{k,k+1}_t$ and all the pixels in $P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$, is performed. A matched grade is computed for every $(x, y) \in P^{k,k+1}_{t+1,r}$, $r = 1,\ldots,R_k$, using (1). The matched value is assigned to the pixel that minimizes the SAD (2) among all the searched

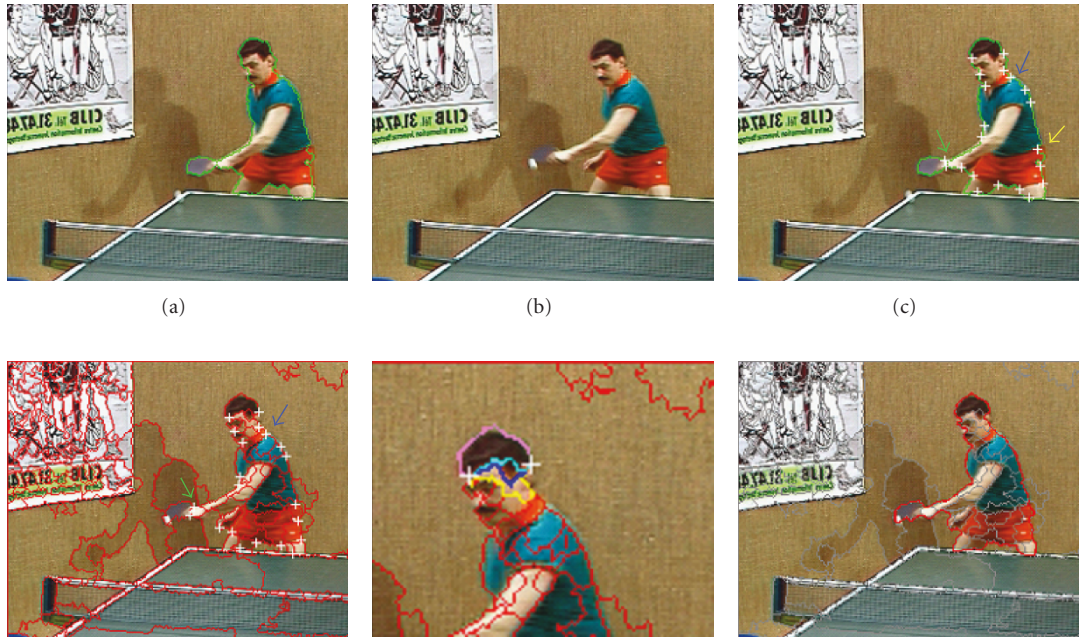


FIGURE 8: Step-by-step illustrations of the main steps of the algorithm. (a) and (b) are the input frames. (c) and (d) illustrate the junction detection with their matched points, respectively. (e) illustrates a single set of candidate paths, and (f) is the final tracked object.

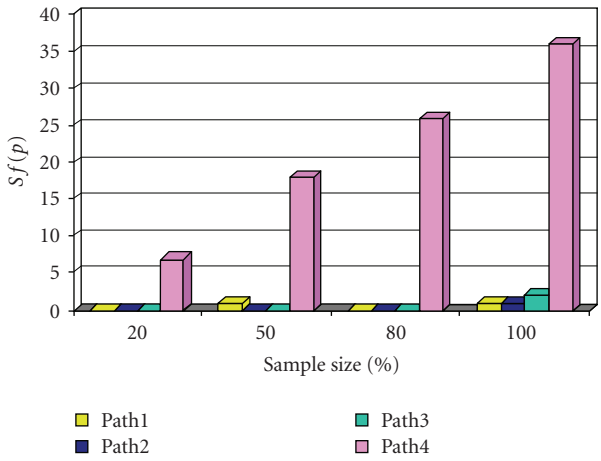


FIGURE 9: The values of $Sf_k(p)$ (6) as function of β (sample size) for the four paths in Figure 8(e). β is assumed to be 20%, 50%, 80%, and 100%.

pixels $(x, y) \in P_{t+1,r}^{k,k+1}, r = 1, \ldots, R_k$. We call this point the *best point* and denote it by $bp_{t+1}^{k,k+1}(x, y)$. In addition, $BP_{t+1}^{k,k+1}(x, y)$ denotes the set of all the *best point* that corresponds to the subset $SP_t^{k,k+1}$. Then, for $k = 1, \ldots, N$, we have

$$BP_{t+1}^{k,k+1} = \{bp_{t+1}^{k,k+1}(x, y) \mid (x, y) \in P_{t+1,r}^{k,k+1},\ r = 1, \ldots, R_k\}. \tag{4}$$

If $BP_{t+1}^{k,k+1}$ belongs entirely to a single path p, $p \in P_{t+1,r}^{k,k+1}$, $r = 1, \ldots, R_k$, such that $BP_{t+1}^{k,k+1} \subseteq p$, then this path is considered as the matched path $mp_{t+1}^{k,k+1}$ in C_{t+1}. Otherwise, we differentiate between the paths by giving a grade to each $p \in P_{t+1,r}^{k,k+1}, r = 1, \ldots, R_k$ as follows. For $k = 1, \ldots, N$ and for each $(x, y) \in P_{t+1,r}^{k,k+1}, r = 1, \ldots, R_k$, the characteristic function $f_k(x, y)$ is defined as follows:

$$f_k(x, y) \triangleq \begin{cases} 1, & (x, y) \in BP_{t+1}^{k,k+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

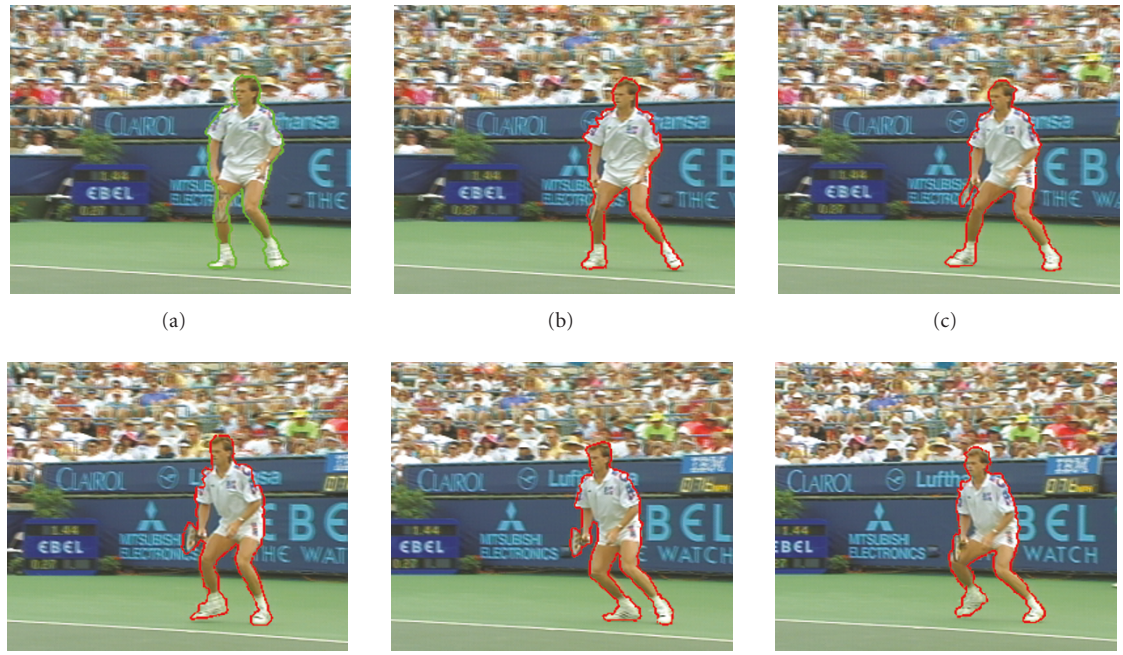
Each path $p \in P_{t+1,r}^{k,k+1}, r = 1, \ldots, R_k$ is assigned with a matched grade $Sf_k(p)$ by

$$Sf_k(p) \triangleq \sum_{(x,y)\in p} f_k(x, y). \tag{6}$$

Then, we consider the path that maximizes $Sf_k(p)$ as the matched path $mp_{t+1}^{k,k+1}$:

$$mp_{t+1}^{k,k+1} \triangleq \underset{p\in P_{t+1,r}^{k,k+1}, r=1,\ldots,R_k}{\arg\max} Sf_k(p). \tag{7}$$

Although not all the best points satisfy $BP_{t+1}^{k,k+1} = mp_{t+1}^{k,k+1}$, most of them do. The use of a set of best points in each subcurve $P_t^{k,k+1}$ is aimed to offset and correct the errors produced by the minimization of the SAD measurements of a single point. However, since the searched area $\{(x, y) : (x, y) \in P_{t+1,r}^{k,k+1}, r = 1, \ldots, R_k\}$ is characterized by different homogeneous edges, it reduces the probability of a mismatch between similar closed points. In other words, using the data structure $G_{t+1} = (V_{t+1}, E_{t+1})$ as a limited search area, enforces



(a) (b) (c)

(d) (e) (f)

Figure 10: Final tracking results for five successive frames taken from "Stefan" video sequence.

$bp_{t+1}^{k,k+1}(x,y)$ to reside either in C_{t+1} or in a relatively far and dissimilar region. Note that the grades, which distinguish between the path $mp_{t+1}^{k,k+1}$ and the other paths $P_{t+1,r}^{k,k+1} - mp_{t+1}^{k,k+1}$, $r = 1,\ldots,R_k$, are affected by the subset size β. Decreasing the size of β reduces the computational time, especially for longer paths. Subset paths of $\beta = 30\%$ from the original subcurve $P_t^{k,k+1}$ are sufficient to reliably produce the contour C_{t+1}.

6. IMPLEMENTATION AND COMPLEXITY

Our method stresses the accuracy in finding the object's contour and not the area where it is located. An efficient implementation is achieved while preserving the accuracy of the final results.

6.1. Implementation

input: I_t, C_t, I_{t+1}
output: C_{t+1}
process:

(1) Still segmentation in Section 3.1 that uses [22–24] is applied on I_t and I_{t+1}. R_i^t, $i = 1,\ldots,n_t$ and R_j^{t+1}, $j = 1,\ldots,n_{t+1}$ are the segmentations of I_t and I_{t+1}, respectively.

(2) $G_t = (V_t, E_t)$ and $G_{t+1} = (V_{t+1}, E_{t+1})$ are constructed. They represent the segmentation of I_t and I_{t+1}, respectively.

(3) For every $(x,y) \in C_t$, we check whether (x,y) satisfies the junction definition. If it does, we add (x,y) to the set of junctions J_k, $k = 1,\ldots,N$.

(4) C_t is segmented into $P_t^{k,k+1}$, $k = 1,\ldots,N$ subcurves that correspond to the set of junctions J_k, $k = 1,\ldots,N$.

(5) For every junction J_k, $k = 1,\ldots,N$, do the following.

(a) Its SADs (1) are calculated in the corresponding searched area in E_{t+1}.

(b) The point with the minimal SAD 2 is added to S_k list.

(6) The graph G''_{t+1} is constructed from G_{t+1}.

(7) For every S_k and S_{k+1}, $k = 1,\ldots,N$, do the following.

(a) All the candidate paths $P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$ that are shorter than twice the length of $P_t^{k,k+1}$ (Section 4.2) are found.

(b) The matched path $mp_{t+1}^{k,k+1} \in P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$, among all the paths $P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$ are found as follows.

(i) The subset $SP_t^{k,k+1}$ of β independent points from $P_t^{k,k+1}$ is constructed.

(ii) We find for each $(x,y) \in SP_t^{k,k+1}$ a point (using (1) and (2)) that belongs to one set in $P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$, and assign it to $BP_{t+1}^{k,k+1}$.

(iii) For all $(x,y) \in P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$, $f_k(x,y)$ is computed.

(iv) For all $p \in P_{t+1,r}^{k,k+1}$, $r = 1,\ldots,R_k$, $Sf_k(p)$ (using (6)) is calculated.

(v) $mp_{t+1}^{k,k+1}$ is the path that maximizes (7) $Sf_k(p)$, $r = 1,\ldots,R_k$.

(a) (b) (c)

(d) (e) (f)

Figure 11: Final tracking results for five successive frames taken from "soccer" video sequence.

6.2. Complexity analysis

We present here the complexity analysis of the implementation in Section 6.1. The numbers refer to the steps in Section 6.1.

(1) Image segmentation is done in $O(N + K \cdot |E| \log |E|)$ operations, where K is the number of iterations (see [14, 22–24]).

(2) The construction of the RAG requires one scan on the boundaries of the segmentation. Therefore, the construction of $G_t = (V_t, E_t)$ and $G_{t+1} = (V_{t+1}, E_{t+1})$ requires $O(|E_{t+1}| + |E_t|)$ operations, where $|E_t|$ is the number of pixels that is stored in E_t.

(3) Let n be the number of pixels on the object's contour C_t. Finding junctions on C_t is done in one scan of C_t, which requires $O(n)$ operations. The following steps are performed for each $(x, y) \in C_t$: (i) its eight neighbors are extracted; (ii) the label of (x, y) and the labels of its neighbors are compared. $O(1)$ operations are needed to accomplish step (i), since $(x, y) \in C_t$, $(x, y) \in R_k$, $k = 1, \ldots, N$. Step (ii) checks whether there exists at least one neighbor (x', y') of (x, y) such that $(x', y') \in R_{k-1}$ or $(x', y') \in R_{k+1}$. If $k = 1$, then R_{k-1} is R_N. If $k = n$, then R_{k+1} is R_1. If (x, y) has such a neighbor, then it is a junction. Otherwise, it is not. This step requires $O(1)$ operations when segmentation labeling is used. Since the size of C_t in the worst case is $O(|E_t|)$, then finding these junctions requires $O(|E_t|)$ operations.

(4) The segmentation of a closed curve C_t on a given set of junctions J_k, $k = 1, \ldots, N$, into $P_t^{k,k+1}$, $k = 1, \ldots, N$, subcurves is done by one scan of C_t. This requires N operations. Thus, in the worst case, $N = |E_t|$. The segmentation of C_t requires $O(|E_t|)$ operations.

(5) The number of required operations in the matching process of any junction depends on the number of candidate points in the search area and on the number of operations required to find the minimal SAD. Since the size of the SAD window is predefined, the SAD operations are considered to be constant. In the worst case, the number of candidate points in the search area is $O(|E_{t+1}|)$. The number of junctions (in the worst case) is $O(|E_t|)$. Therefore, the matching process for all the junctions requires $O(|E_t \cdot |E_{t+1}|)$ operations.

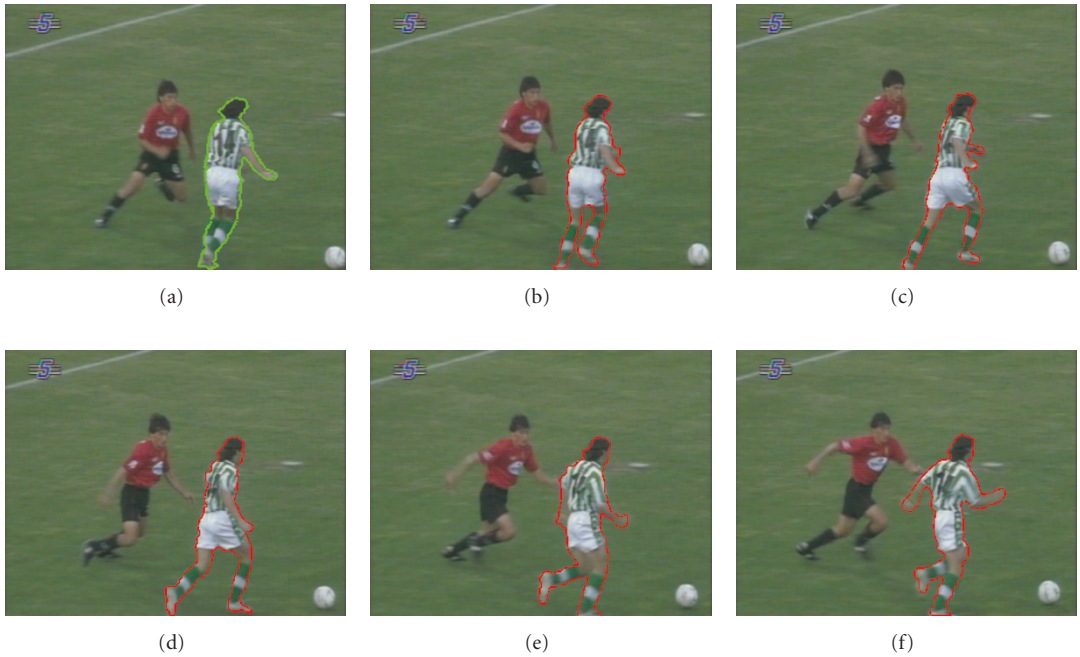
(6) The construction of G''_{t+1}, which is linear in the graph size, requires $O(|E''_{t+1}|)$ operations.

(7) We estimate here the cost of the two main procedures: finding candidate paths (Section 4.1) and the matching process (Section 5.1). In Section 4.1, the algorithm that finds the k shortest path between a pair of matched points is used. It requires $O(|E''_{t+1}| + |V'_{t+1}| + k)$ operations, where k is the number of paths (see the analysis in [35]). The number of pair of junctions in the worst case is $O(|E_{t+1}|/2)$. Therefore, the overall complexity of Section 4.1 is $O(|E_{t+1}||E''_{t+1}| + |V'_{t+1}| + k)$. The match process inc Section 5.1 between β points on C_t and all the candidate paths is done in $O(|E_t \cdot |E_{t+1}|)$ operations since β is equal (at most) to $|E_t|$ and the size of all the candidate paths in the worst case is $O(|E_{t+1}|)$.

The total number of operations after we sum each step in the above operations is

$$(O(|E_t|) + |E_t| \cdot |E_{t+1}| + |E_{t+1}|(|E''_{t+1}| + |V'_{t+1}| + k)). \tag{8}$$

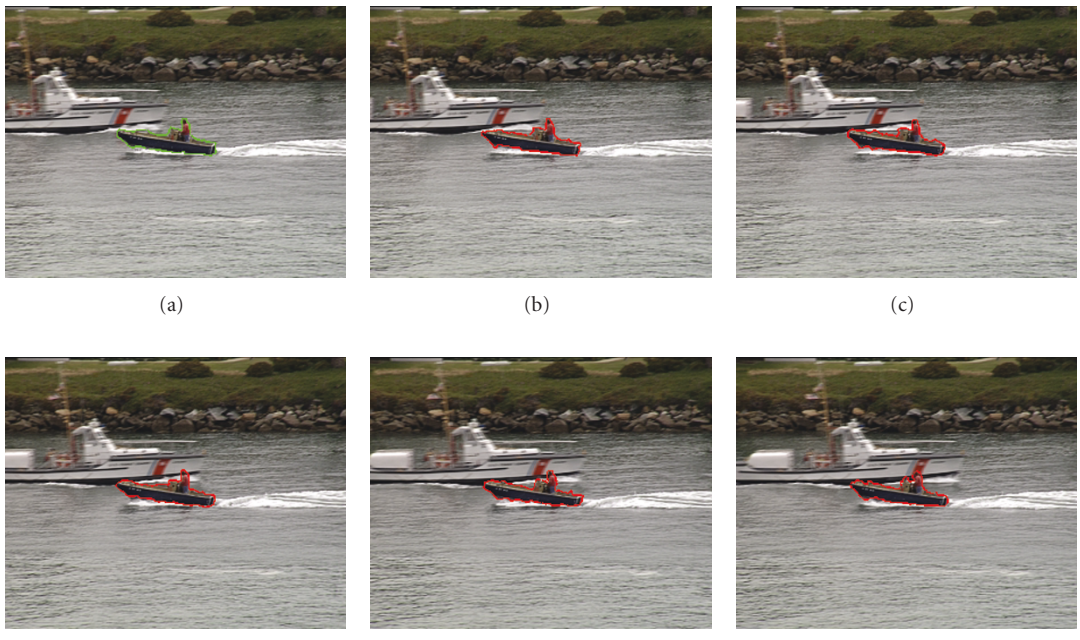
Since consecutive frames have similar content, we assume that $O(|E_t|) \approx O(|E_{t+1}|)$. From the construction of G'_{t+1},



(a) (b) (c)

(d) (e) (f)

Figure 12: Final tracking results for five successive frames taken from another part of the "soccer" video sequence in Figure 11. On the top most left image, the legs are tracked as one object and in the top middle image the legs are separated and tracked correctly.



(a) (b) (c)

(d) (e) (f)

Figure 13: Final tracking results for five successive frames taken from the "Coast Guard" video sequence. It demonstrates the capability of the algorithm to track merge and split objects.

we get that $O(|V'_{t+1}|) \approx O(|E_{t+1}|)$. In addition, for the worst case $|E'_{t+1}| = 8|V'_{t+1}|$, each pixel has maximum of eight neighbors. Thus, $O(|E'_{t+1}|) \approx O(|V'_{t+1}|)$. From the construction of G''_{t+1}, we have $O(|E''_{t+1}|) \approx O(|V'|)$. Therefore, $O(|E''_{t+1}|) \approx O(|E_{t+1}|)$. Consequently, the overall complexity of the tracking algorithm is $O(|E_t|^2)$.

7. EXPERIMENTAL RESULTS

A variety of video sequences with different motion types were examined. In Section 7.1, we present a step-by-step illustration of a single example. The final results of tracked contours are presented in Section 7.2.

7.1. Step-by-step illustration of the algorithm

Figure 8 is a step-by-step evolution of the tracking algorithm and how the final tracked contour is constructed. Figures 8(a) (I_t) and 8(b) (I_{t+1}) are frames 1 and 3, respectively, from the "Tennis" sequence. The input contour of the object to be tracked is surrounded by a green curve in Figure 8(a). The segmented curve of the first frame is shown in Figure 8(c) (I_{t+1}). Each pair of consecutive junctions J_k and J_{k+1}, $k = 1,\ldots,N$, is marked by white crosses. The junction that is marked with a yellow arrow in Figure 8(c) represents the intersection of two different homogeneous regions (the red pants and the blue shirt of the player), which is obtained from its still image segmentation process. The two other arrows (blue and green), marked in Figure 8(d) (I_{t+1}), represent the corresponding matched points (blue and green) from Figure 8(c) (I_t), respectively. As shown, all the detected junctions in Figure 8(c) are located on the boundaries between different homogeneous areas, and thus, are classified as "important" and well-tracked points that will be used in the next steps of the algorithm.

A particular example of a curve construction of a pair of matched points S_1 and S_2 is given in Figure 8(e). For this pair (marked by two white crosses in Figure 8(e)), a set consisting of four different candidate paths $P^{1,2}_{t+1,r}$, $r = 1,\ldots,4$ (marked by four different colors) is found. The pink path, in this example, represents the matched path $mp^{1,2}_{t+1}$. This path is located after the application of the matching procedure between the set $P^{1,2}_{t+1,r}$, $r = 1,\ldots,4$ and $P^{1,2}_t$ (shown by the final constructed contour in Figure 8(f)). The matched grades $Sf_1(p)$ of $P^{1,2}_{t+1,r}$, $r = 1,\ldots,4$, are given in Figure 9 as a function of four different β values (20%, 50%, 80%, and 100%). As shown in Figure 9, the maximal $Sf_1(p)$ was obtained for the pink path, for all β values. Different values of β affect only the ratio between the right path (pink) and the other paths.

7.2. Final results

The examples in Figures 10, 11, 12, and 13 demonstrate the final results of tracked contours in four different video sequences. All the experiments were performed without tuning of any parameters. The input object to be tracked in all the examples is marked by a green contour. Red contour represents the algorithm's final output in the current frame.

8. CONCLUSIONS

In this paper, we propose a novel contour-based algorithm for tracking a moving object. Based on the RAG data structure, accurate results are achieved while preserving a low complexity. In the initialization step, two consecutive input frames are segmented with respect to a semantic homogeneous criterion. Their corresponding RAGs are constructed to represent the partitions of the frames. The object's contour is segmented into subcurves according to the detected junctions that reside on the contour. The subcurves of the object's contour are the basis for the construction of the new tracked contour. A corresponding point for each junction in I_{t+1} is searched only in the RAG edges of the consecutive frame. Then, each pair of matched points is connected by a set of candidate paths. Among all the candidate paths, the path that is most similar to its corresponding subcurve is considered to be a part of the tracked contour. Hence, only one of the RAG's edges represents the tracked contour. Consequently, the new object's contour is accurately constructed by the matched paths, and the overall complexity of the algorithm is proportional to the edges of the RAG. Note that representation of RAG edges usually consists of 10% of the entire image.

REFERENCES

[1] Y. Cui, S. Samarasekera, Q. Huang, and M. Greiffenhagen, "Indoor monitoring via the collaboration between a peripheral sensor and a foveal sensor," in *Proceedings of IEEE Workshop on Visual Surveillance (VS '98)*, pp. 2–9, Bombay, India, January 1998.

[2] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV '98)*, pp. 214–219, Princeton, NJ, USA, October 1998.

[3] A. Vetro, H. Sun, and Y. Wang, "MPEG-4 rate control for multiple video objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 186–199, 1999.

[4] M. R. Naphade, I. V. Kozintsev, and T. S. Huang, "Factor graph framework for semantic video indexing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 40–52, 2002.

[5] M. R. Naphade and T. S. Huang, "A probabilistic framework for semantic video indexing, filtering, and retrieval," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 141–151, 2001.

[6] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[7] J. I. Agbinya and D. Rees, "Multi-object tracking in video," *Real-Time Imaging*, vol. 5, no. 5, pp. 295–304, 1999.

[8] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[9] E. Ozyildiz, N. Krahnst"over, and R. Sharma, "Adaptive texture and color segmentation for tracking moving objects," *Pattern Recognition*, vol. 35, no. 10, pp. 2013–2029, 2002.

[10] D. Yang and H.-I. Choi, "Moving object tracking by optimizing active models," in *Proceedings of the 14th International Conference of Pattern Recognition (ICPR '98)*, vol. 1, pp. 738–740, Brisbane, Australia, August 1998.

[11] R. Murrieta-Cid, M. Briot, and N. Vandapel, "Landmark identification and tracking in natural environment," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '98)*, vol. 1, pp. 179–184, Victoria, Canada, October 1998.

[12] D.-S. Jang, S.-W. Jang, and H.-I. Choi, "2D human body tracking with structural Kalman filter," *Pattern Recognition*, vol. 35, no. 10, pp. 2041–2049, 2002.

[13] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 539–546, 1998.

[14] C. L. Lam and S. Y. Yuen, "An unbiased active contour algorithm for object tracking," *Pattern Recognition Letters*, vol. 19, no. 5-6, pp. 491–498, 1998.

[15] Y. Fu, A. T. Erdem, and A. M. Tekalp, "Occlusion adaptive motion snake," in *Proceedings of IEEE International Conference on Image Processing (ICIP '98)*, vol. 3, pp. 633–637, Chicago, Ill, USA, October 1998.

[16] J. Shao, F. Porikli, and R. Chellappa, "A particle filter based non-rigid contour tracking algorithm with regulation," in *Proceedings of IEEE International Conference on Image Processing (ICIP '06)*, Atlanta, Ga, USA, October 2006.

[17] D. G. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 113–122, 1992.

[18] P. Wunsch and G. Hirzinger, "Real-time visual tracking of 3-D objects with dynamic handling of occlusion," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '97)*, vol. 4, pp. 2868–2873, Albuquerque, NM, USA, April 1997.

[19] K. F. Lai and R. T. Chin, "Deformable contours: modeling and extraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 11, pp. 1084–1090, 1995.

[20] C. Kervrann and F. Heitz, "Hierarchical Markov modeling approach for the segmentation and tracking of deformable shapes," *Graphical Models and Image Processing*, vol. 60, no. 3, pp. 173–195, 1998.

[21] T. Schoepflin, V. Chalana, D. R. Haynor, and Y. Kim, "Video object tracking with a sequential hierarchy of template deformations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 11, pp. 1171–1182, 2001.

[22] E. Navon, *Image segmentation based on minimum spanning tree and computation of local threshold*, M.S. thesis, Tel-Aviv University, Tel-Aviv, Israel, October 2002.

[23] E. Navon, O. Miller, and A. Averbuch, "Color image segmentation based on adaptive local thresholds," *Image and Vision Computing*, vol. 23, no. 1, pp. 69–85, 2004.

[24] E. Navon, O. Miller, and A. Averbuch, "Color image segmentation based on automatic derivation of local thresholds," in *Proceedings of the 7th Digital Image Computing, Techniques and Applications Conference (DICTA '03)*, pp. 571–580, Sydney, Australia, December 2003.

[25] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[26] J. Shei and C. Tomasi, "Good feature to track," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pp. 593–600, Seattle, Wash, USA, June 1994.

[27] R. Laganière, "A morphological operator for corner detection," *Pattern Recognition*, vol. 31, no. 11, pp. 1643–1652, 1998.

[28] J.-S. Lee, Y.-N. Sun, and C.-H. Chen, "Multiscale corner detection by using wavelet transform," *IEEE Transactions on Image Processing*, vol. 4, no. 1, pp. 100–104, 1995.

[29] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference (AVC '88)*, pp. 147–151, Manchester, UK, August-September 1988.

[30] P. Smith, D. Sinclair, R. Cipolla, and K. Wood, "Effective corner matching," in *Proceedings of the 9th British Machine Vision Conference (BMVC '98)*, pp. 545–556, Southampton, UK, September 1998.

[31] A. A. Argyros, K. E. Bekris, and S. C. Orphanoudakis, "Robot homing based on corner tracking in a sequence of panoramic images," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 2, pp. 3–10, Kauai, Hawaii, USA, December 2001.

[32] G. Srivastava, G. Agarwal, and S. Gupta, "A very low bit rate video codec with smart error resiliency features for robust wireless video transmission," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '02)*, vol. 4, p. 4191, Orlando, Fla, USA, May 2002.

[33] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 652–673, 1998.

[34] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[35] A. Zisserman, A. Blake, and R. Curwen, "A framework for spatio-temporal control in the tracking of visual contours," in *Real-Time Computer Vision*, C. M. Brown and D. Terzopoulos, Eds., pp. 3–33, Cambridge University Press, Cambridge, UK, 1995.