Voting-based Simultaneous Tracking
of Multiple Video Objects

Aishy Amer

Abstract

This paper proposes an automatic object tracking method based on both object segmentation and motion estimation for real-time content-oriented applications. The method focuses on the issues of speed of execution and reliability in the presence of noise, coding artifacts, shadows, occlusion, and object split.

Objects are tracked based on the similarity of their features in successive frames. This is done in three steps: feature extraction, object matching, and feature monitoring. In the first step, objects are segmented and their spatial and temporal features are computed. In the second step, using a non-linear two-step voting strategy, each object of the previous frame is matched with an object of the current frame creating a unique correspondence. In the third step, object changes, such as objects occlusion or split, are monitored and object features are corrected. These new features are then used to update results of previous steps creating module interaction.

The contributions in this paper are the real-time two-step voting strategy and the monitoring of object changes to handle occlusion and object split. Experiments on indoor and outdoor video shots containing over 6000 frames, including deformable objects, multi-object occlusion, noise, and coding and object segmentation artifacts have demonstrated the reliability and real-time response of the proposed method.

Keywords

Video object tracking, multiple objects, non-linear feature voting, object occlusion, object split, region merging, object prediction, object segmentation.


Aishy Amer is with the Department of Electrical and Computer Engineering, Concordia University, Montréal, Quebéc, Canada. Send correspondence to amer@ece.concordia.ca.

This work was supported, in part, by the Natural Sciences and Engineering Research Council (NSERC) of Canada.
I. Introduction

Object tracking has many applications. It can be used, for example, to interpret temporal behavior of objects [1], [2], [3], [4], [5], to assist estimation of coherent motion trajectories, and to support object segmentation [6]. Solving the correspondence (or matching) problem, i.e., finding a unique correspondence between two objects in ambiguous conditions is the challenge in object tracking where it is important not to lose object during tracking. Finding a unique correspondence in real scenes is a difficult task due to (a) image changes, such as noise, shadows, light changes, reflection, and clutter, that can obscure object features to mislead tracking; (b) the presence of non-rigid or articulated objects and their non-uniform features; (c) the presence of multiple moving objects, especially when objects have similar features or when their paths cross; (d) ambiguous matches, e.g., one object corresponds to several objects, for example, when objects split or merge; (e) erroneous segmentation of objects;
(f) changing features due to varying object appearance, e.g., deformation, scale, light, or viewpoint changes; and (g) application related requirements, such as real-time processing.

To address these difficulties, this paper proposes a tracking scheme that accounts for (1) feature changes by monitoring features over time (Sec. VI), (2) multiple matches by introducing a confidence measure and explicit comparison of multiple matches (Sec. V-B), (3) local changes by adaptation to noise and to previous frames (Sec. IV), and (4) real-time processing by avoiding complex operations.

The remainder of the paper is organized as follows: Section II discusses related work; Section III gives an overview of our tracking method; Section IV presents our choice of features used; Section V proposes our non-linear voting strategy; Section VI shows how to adapt the tracking to characteristics of the input video (VI-C), how to handle multi-object occlusion (VI-A), and how to relate split objects (VI-B); Section VII discusses experimental results; and Section VIII contains a conclusion.

II. Related work

While object tracking has been extensively studied, limited work has been done to real-time tracking of multiple objects. In this section, we review fast or real-time methods as to compare with the proposed method. Techniques for object tracking are numerous [2], [6], [7], [8], [9], [10], [11], [12], [13], [14] and can be distinguished into two main strategies: the first uses correspondence to match segmented objects between successive frames (e.g., [13], [9], [14]) and the second performs explicit tracking using position prediction or motion estimation (e.g., [6], [9], [12], [15]).

Explicit tracking approaches model occlusion implicitly but have difficulties detecting entering objects without delay and simultaneously tracking multiple objects. Furthermore, they assume that object features remain invariant over time [12]. Most of these methods have high computational costs and are not suitable for real-time applications.

Tracking based on correspondence uses object prediction to handle, for instance, occlusion. Prediction techniques can be based on Kalman filters or on motion compensation. The use of a Kalman filter
[7], [13], [10], [14] relies on an explicit trajectory model. The definition of an explicit trajectory model is difficult in complex scenes and cannot be easily generalized [9]. Furthermore, Kalman filtering has difficulties recovering its target when lost [9]. Advanced Kalman filters can estimate trajectories in some occlusion cases but have difficulty when occlusion, the number of objects, or artifacts increases.

Few methods have considered both real-time response and multiple objects. In addition, limited solutions to the occlusion problem exist (e.g., [2], [13]). These solutions track objects after and not during occlusion. In addition, many methods are designed for specific applications (e.g., using object models) [11], [14], [12] or impose constraints on camera or object motion (e.g., upright motion) [2], [13]. Furthermore, many tracking approaches based on feature extraction assume that the object topology is fixed throughout the image sequence.

This paper contributes a real-time solution to tracking of multiple object including heavy occlusion. No prior knowledge, initialization, or object models are assumed. The main contribution is the non-linear feature voting scheme that solve the occlusion problem by establishing unique correspondences using two steps: object voting to match two objects and correspondence voting to resolve ambiguities matches in case of multiple matches. This paper contributes, furthermore, a solution to explicitly accounts for changing object features, e.g., due to object fusion (occlusion), object splitting, and object deformation due to temporally varying shape. This is done by adaption to image noise, by explicitly detecting object occlusion or erroneous object segmentation, and monitoring features over time. In addition, we introduce plausibility rules to handle errors and confirm temporal reliability and consistency of object matching. The proposed method is designed for real-time content-oriented applications such as video surveillance, e.g., to support high-level video content extraction.

III. PROPOSED APPROACH - AN OVERVIEW

Our approach is oriented to requirements of applications where speed of execution and temporal reliability are of more concern than precise object features (e.g., boundaries). It aims at tracking
multiple objects in the presence of heavy occlusion or split where it is important not to lose objects.
We assume that object motion is smooth and objects do not suddenly disappear or change direction.
No other assumptions are imposed on the motion of objects or on the camera position.

In our approach, objects are tracked based on similarity of their features using three steps: feature
extraction, feature matching, and feature monitoring (Fig. 1). In the first step, object segmentation [16]
and motion estimation [17] modules extract objects and their spatial and temporal features (Sec. IV).
The output of this step is $O(n)$ and $O(n-1)$, lists of objects in the current $I(n)$ and previous $I(n-1)$
frames, respectively, with $n$ the time instant. Note that the cardinalities of both lists may differ.

Fig. 1. Framework of the proposed tracking method. $O(n), O(n-1)$ are object lists of the frames $I(n)$ and $I(n-1)$.

In the second step, object matching is activated once an object enters (or appears) where the seg-
mentation module keeps looking for objects entering (or appearing). For object matching we combine
single feature using a two-step non-linear voting scheme where each object $O_{lp} \in I(n-1)$ is matched
with at least one object $O_{ic} \in I(n)$ creating a correspondence (or a temporal link) $M_{li} : O_{lp} \rightarrow O_{ic}$.

An object $O_{ic} \in I(n)$ that has no correspondence in $I(n-1)$ is assumed to be entering or appearing.
Once an object enters it is assigned a unique identity (ID) that is kept throughout the input video.
If an object $O_{ic} \in I(n)$ appears the object matching looks if this object can be matched to objects
without correspondence in $I(n-1)$. If an object $O_{lp} \in I(n-1)$ cannot be matched (found) in $I(n)$ it
is assumed to have exited or disappeared.
The third step, feature monitoring, supports finding a unique correspondence by detecting and correcting errors resulting from the first or second step. In this paper, we focus on segmentation errors, such as when objects are erroneously merged (e.g., due to occlusion) or split (e.g., due to noise). Correction of merging or split produces new object data which are then used to update the output of previous steps (see the feedback loops in Fig. 1). For example, when object merging due to occlusion is corrected, new objects are produced. In this case, data (such as shape) of old objects in $I(n-1)$ need to be updated and motion estimation and tracking need to be performed for the new objects. In the case of multiple object occlusion, i.e., more than two objects merge, the occlusion detection module first detects occluded and occluding objects and then continues to track both types of objects even if objects are completely occluded. This is important in the case when objects reappear.

When object tracking between $I(n)$ and $I(n-1)$ is done, information is passed to the next frame $I(n+1)$ through a linked list of objects and their features. These features are set or updated throughout the three step tracking algorithm. The features used are identity (ID), age (the number of frames since the object occurred), perimeter, area, minimum bounding box (MBB), start position, split data (e.g., current status and ID of the object that was split from), occlusion data (e.g., status and ID of the object that occludes, or is occluded by, this object), recent motion vectors, corresponding confidence, and the link to the corresponding object.

IV. STEP 1: FEATURE EXTRACTION

In this paper, we select the following simple but efficient object features which when combined together aim at balancing real-time and reliability requirements:

- Size is represented by the area $A_{ic}$ of an object $O_{ic}$, perimeter $P_{ic}$ of $O_{ic}$, maximum horizontal extent $W_{ic}$ of $O_{ic}$, and maximum vertical extent $H_{ic}$ of $O_{ic}$.

- Shape is represented by easy to compute and relatively robust features: the minimum bounding box (MBB) $B_{O_{ic}}$, i.e., the smallest rectangle that includes $O_{ic}$, extent ratio $e_{ic} = \frac{H_{ic}}{W_{ic}}$ if $H_{ic} < W_{ic}$ (or $e_{ic} =\ldots$}
\[ \frac{W_{ic}}{H_{ic}} \] if \( W_{ic} < H_{ic} \), \( e_{ic} \) is rotation- and scale invariant), compactness \( c_{ic} = \frac{A_{ic}}{W_{ic}H_{ic}} \), and irregularity \( r_{ic} = \frac{P_{ic}^2}{4\pi A_{ic}} \), \( c_{ic} \) increases when the shape becomes irregular and \( r_{ic} \) is invariant to various transformations [18] as the squared perimeter makes the ratio independent of the object size. These simple shape descriptors are a compromise between matching quality and computational complexity [19].

- Motion is represented by the current displacement \( w_{ic} = (w_x, w_y) \) and direction \( \delta_{ic} = (\delta_x, \delta_y) \) of \( O_{ic} \).
- Center-of-gravity (centroid) of \( O_{ic} \) is represented by the center of \( B_{O_{ic}} \).
- Distance is represented by the Euclidean distance between the centroids of two objects.

To extract spatial object features, we use an object segmentation [16] based on three steps: change detection, morphological edge detection, and contour analysis. The change detection is spatio-temporal adaptive to estimated noise variance and to previously detected motion by using a memory component that examines if there has been a significant, little, or no temporal change in the current frame compared to previous frames.

To extract motion features, we apply the method in [17] that is based on two steps: initial coarse estimation, to find a single displacement for an object using the four sides of the minimum bounding box of the segmented object, and estimation of non-translational motion, to detect the type of object motion and to eventually estimate different displacement values per object.

V. STEP 2: VOTING-BASED OBJECT MATCHING

When matching two objects using several features, the question is how to combine these features for reliable tracking. Many methods combine features linearly using a weighting function. A linear combination does not, however, 1) take into account the non-linear properties of the human visual system (HVS), 2) consider the distinguishing power of single feature, and 3) monitor the effectiveness, that can vary over time, of a feature.

In this paper, we combine object features using a non-linear voting scheme consisting of two steps (Fig. 2): voting between features of two objects (object voting, Sec. V-A) and voting between features...
of two established correspondences in case of multiple matches (correspondence voting, Sec. V-B). Each voting step is first divided into \( v \) sub-votes with \( v \) features. Since features can become misleading, \( v \) varies spatially (objects) and temporally (throughout the video) depending on a feature monitoring strategy (Sec. VI). Then each sub-vote is performed separately using an appropriate voting function. When the voting function is applied either a similarity measure \( s \) or a dissimilarity measure \( q \) is increased. Depending on the number of features in a sub-vote \( s \) or \( q \) may increase by one or more votes. Finally, a majority rule compares the two measures and decides about the final vote.

Note that in the object voting step, each object \( O_{lp} \in I(n-1) \) is matched with at least one object
$O_{ic} \in I(n)$, $M_{li} : O_{lp} \rightarrow O_{ic}$. If no object in $I(n - 1)$ can be matched to $O_{ic} \in I(n)$, i.e., zero match $M_{li} : \nrightarrow O_{ic}$, $O_{ic}$ is declared entering or appearing depending on its location. If no object in $I(n)$ can be matched to $O_{lp} \in I(n - 1)$, i.e., reverse zero match $M_{l0} : O_{lp} \nrightarrow$, $O_{lp}$ is declared disappearing or exiting depending on its location.

A. Object voting

To establish initial object correspondences, the votes for shape, size, and motion are applied (cf. Appendix). A correspondence $M_{li} : O_{lp} \rightarrow O_{ic}$ between $O_{lp}$, the $l^{th}$ object of the previous frame $I(n - 1)$, and $O_{ic}$, the $i^{th}$ object of the current frame $I(n)$, is accepted (rejected) if

$$M_{li} : (d_{li} < t_r) \land (w_{x_{ic}} < w_{max}) \land (w_{y_{ic}} < w_{max}) \land (\zeta_{li} != q_{sv})$$

(1)

where

- $M_{li} : O_{lp} \rightarrow O_{ic}$ is a non-correspondence between $O_{lp}$ and $O_{ic}$,
- $d_{li}$ is the distance between $O_{lp}$ and $O_{ic}$,
- $t_r = \frac{500}{R} + \text{MAX}(W_{lp}, H_{lp})$ is the radius of a search area centered at $O_{lp}$,
- $W_{lp}$ and $H_{lp}$ are the width and height of $O_{lp}$,
- $w_{ic} = (w_{x_{ic}}, w_{y_{ic}})$ is the displacement of $O_{lp}$ if matched to $O_{ic}$,
- $w_{max}$ is the maximum displacement adaptive to the frame rate $R$ (Sec VI-C, $15 < w_{max} < 32$),
- $\zeta_{li}$ is the confidence of $M_{li}$, and
- $s$ and $q$ are the similarity and dissimilarity measures between $O_{lp}$ and $O_{ic}$.

The conditions $w_{x_{ic}} < w_{max}$ and $w_{y_{ic}} < w_{max}$ are relaxed with occlusion. To compute $s$ and $q$ five voting functions are applied as in Eqs. 10-13 (cf. Appendix). Eq. 1 states that $M_{li}$ is accepted if $O_{ic}$ is close to $O_{lp}$, the displacements of $O_{ic}$ is not too large, and both objects are sufficiently similar (i.e., $\zeta_{li}! = \frac{q_{sv}}{v}$).
\( \zeta_{li} \) measures the degree of certainty of \( M_{li} \) as follows:

\[
\zeta_{li} = \begin{cases} 
\frac{1}{v} & s = q \\
\frac{s-q}{v} & \frac{s}{q} > t_v \\
\frac{q-s}{v} & \frac{s}{q} < t_v 
\end{cases}
\]  

(2)

where \( v \) is the total number of feature votes and \( 0 < t_v < 1 \) is the confidence threshold which is set lower if \( O_{lp} \) experience occlusion or split.

Note that following Eqs. 1 and 2, \( M_{li} \) may be accepted even if \( s < q \) (depending on \( t_v \)). This is important when features get significantly dissimilar due to erroneous segmentation (e.g., occlusion or split) and might thus cause the rejection of a good correspondence. Note further that if a set of disjoint regions in \( I(n-1) \) merge into \( O_{ic} \in I(n) \), \( O_{ic} \) is matched to the most similar region of these. In addition, if \( O_{lp} \in I(n-1) \) splits into two regions in \( I(n) \), \( O_{lp} \) is matched to the most similar region. Recall that object voting is followed by correspondence voting and thus possible faulty object voting would be corrected.

### B. Correspondence voting

In the previous step, an object in \( I(n-1) \) is matched against at least one object in \( I(n) \), i.e., object voting may output multiple object correspondences. For example, \( \{ M_{li} : O_{lp} \rightarrow O_{ic} \text{ and } M_{lj} : O_{lp} \rightarrow O_{jc} \} \) or \( \{ M_{li} : O_{lp} \rightarrow O_{ic} \text{ and } M_{mi} : O_{mp} \rightarrow O_{ic} \} \) with \( O_{lp}, O_{mp} \in I(n-1) \) and \( O_{ic}, O_{jc} \in I(n) \). The following majority voting rule is applied to resolve such ambiguities,

\[
M_{li}: \ s_{li} > s_{lj} \\
M_{lj}: \ s_{li} < s_{lj}
\]  

(3)

where \( s_{li} \) and \( s_{lj} \) are the similarity measures representing \( M_{li} \) and \( M_{lj} \). To compute \( s_{li} \) and \( s_{lj} \) 7 voting functions in Eqs. 14-20 (see Appendix) are applied. The first 6 functions are based on the features distance, confidence, size, shape, motion direction, and displacement. If after applying these first six votes \( s_{li} = s_{lj} \) an additional motion-based voting function is applied (Eq. 20).
For temporally consistent tracking, if during the matching process (e.g., after object separation) a better correspondence is found, the matching is revised as follows:

1. if two features of the two correspondences are similar, then this feature is excluded from voting;
2. if an object splits into disjoint regions and $s_{ti}$ and $s_{tj}$ are similar, then the correspondence with the larger object area is selected;
3. if two objects occlude and $s_{ti}$ and $s_{tj}$ are similar, then the correspondence with the oldest (i.e., larger age) object is selected;
4. if two objects occlude and the confidence measures of two correspondences are similar, then the correspondence with the oldest object is selected; and
5. if $O_{jc}$ and $O_{lp}$ of the new correspondence are dissimilar, $O_{lp}$ was occluded, and the two objects are very close, then this correspondence is selected.

VI. STEP 3: FEATURE MONITORING

Low-level video processing methods such as object segmentation are likely to output erroneous results, e.g., object features. A good tracking technique must account for such possible errors. In this section, we propose plausibility rules to adapt the tracking to segmentation errors (e.g., occlusion, Sec. VI-A and split, Sec. VI-B) and to parameters of the input video (Sec. VI-C).

A. Monitoring erroneous object fusion (occlusion)

Object fusion or occlusion is given when two or more objects get connected, i.e., the minimum bounding boxes of the objects physically overlap in the frame. In this case, the object segmentation module outputs one region consisting of all occluded objects. Object occlusion can be a result either of real object occlusion (i.e., objects physically get connected) or of segmentation errors (i.e., objects overlap although being distant).
A.1 Detecting occlusion

When objects get connected, at least one side of the object’s four MBB sides will experience a significantly large outward displacement (Fig. 3(a)). Define:

- $O_{tc} \in I(n)$, the occlusion of $O_{kp} \in I(n-1)$ and $O_{lp} \in I(n-1)$ in $I(n)$ where $M_{ki}: O_{kp} \rightarrow O_{tc}$;
- $d_{kl}$, the distance of the centroids of $O_{kp}$ and of $O_{lp}$;
- $w_{kp} = (w_x, w_y)$, the current displacement of $O_{kp}$, i.e., between $I(n-2)$ and $I(n-1)$.
- $w_{vl}$, the vertical displacement of the lower MBB row of $O_{kp}$;
- $w_{vu}$, the vertical displacement of the upper MBB row of $O_{kp}$;
- $w_{hr}$, the horizontal displacement of the right MBB column of $O_{kp}$;
- $w_{hl}$, the horizontal displacement of the left MBB column of $O_{kp}$; and
- $t_w, t_d$, the displacement and distance thresholds (inversely proportional to $R$, see Sec. VI-C).

Object occlusion is declared (Eq. 4, Fig. 3(a)) if 1) the current displacement and the displacement of one of the four side of the MBB of $O_{kp}$ significantly differ, 2) the displacement of that MBB side is outward, and 3) there exists an object $O_{lp} \in I(n-1)$ close to $O_{kp}$.

\[
\begin{align*}
&\left[ (|w_x - w_{hr}| > t_w) \land (w_{hr} > 0) \land (d_{kl} < t_d) \right] \lor \\
&\left[ (|w_x - w_{hl}| > t_w) \land (w_{hl} > 0) \land (d_{kl} < t_d) \right] \lor \\
&\left[ (|w_y - w_{vl}| > t_w) \land (w_{vl} > 0) \land (d_{kl} < t_d) \right] \lor \\
&\left[ (|w_y - w_{vu}| > t_w) \land (w_{vu} > 0) \land (d_{kl} < t_d) \right].
\end{align*}
\]
If occlusion is detected then all (occluding and occluded) objects are labeled. This labeling allows to continue monitoring objects in subsequent frames even if they become completely occluded (i.e., the occlusion conditions in Eq. 4 are not met). This is important since objects might reappear. Note that the labeling is important to help detect occlusion even if the occlusion conditions in Eq. 4 are not met.

A.2 Correcting fusion errors

If occlusion is detected, the occluded object $O_{ic}$ is separated into two or more objects depending on how many objects were connected. Separation is done by predicting related objects of $I(n - 1)$ onto $I(n)$ based on their estimated displacements.

Assume $O_{kp} \in I(n - 1)$ and $O_{lp} \in I(n - 1)$ occlude in $I(n)$ giving $O_{ic}$ and $M_{ki} : O_{kp} \rightarrow O_{ic}$, i.e., the segmentation module will not list $O_{lp} \in I(n)$. To predict $O_{kp}, O_{lp}$ in $I(n)$, their displacements are estimated as follows:

$$
\begin{align*}
  w_{lp} &= (f(w_{cxl}, w_{pxl}, w_{uxl}), f(w_{cyl}, w_{pyl}, w_{uyl})) \\
  w_{kp} &= (f(w_{cxm}, w_{pxm}, w_{uxm}), f(w_{cym}, w_{pym}, w_{uym}))
\end{align*}
$$

(5)

where

- $f(\cdot)$ is a function that can be defined in different ways such as the mean or the median,
- $w_{cxl}, w_{pxl}, w_{uxl}$ are the current, previous, and past-previous horizontal displacements of $O_{lp}$,
- $w_{cyl}, w_{pyl}, w_{uyl}$ are the current, previous, and past-previous vertical displacements of $O_{lp}$,
- $w_{cxm}, w_{pxm}, w_{uxm}$ are the current, previous and past-previous horizontal displacement of $O_{kp}$, and
- $w_{cym}, w_{pym}, w_{uym}$ are the current, previous and past-previous vertical displacement of $O_{kp}$.

After prediction, the lists of objects and their features in $I(n)$ and $I(n - 1)$ are updated, for example, by adding $O_{lp}$ to $I(n - 1)$. If new objects are added to the object lists of $I(n), I(n - 1)$ both steps, object matching and feature monitoring, are applied recursively for these objects (Fig. 1).
A.3 Tracking during occlusion

When two objects $O_{lp}^a, O_{kp}^b \in I(n - 1)$ with IDs $a, b$ occlude in the current frame $I(n)$, the object segmentation module produces one region, say $O_{ic} \in I(n)$ where $O_{ic} \sim (O_{lp}^a + O_{kp}^b)$. Since $O_{ic}$ is similar to either $O_{lp}^a$ or $O_{kp}^b$, the object-voting step matches $O_{ic}$ to either $O_{lp}^a$ or $O_{kp}^b$, say with $M_{ki} : O_{kp}^b \rightarrow O_{ic}$. In this case, the list of objects in $I(n)$ is missing $O_{kp}^b$. In addition, $O_{ic}$ includes only the visible parts of the two occluded objects, i.e., for example, the area $A_{ic} \leq A_l + A_m$. After prediction and separation of occluded objects, the occlusion component first updates the list of objects and features both in $I(n)$ and $I(n - 1)$ and then recursively performs the matching, the monitoring, and the motion estimation steps for $O_{lp}^a$ and $O_{kp}^b$ (Fig. 1).

This simultaneous update and the subsequent recursive processing facilitate the tracking of objects during heavy occlusion. Assume $O_{lp}^a$ and $O_{kp}^b$ occlude further in the frames $\{I(n+1), \ldots, I(n+f), I(n+f+1), \ldots, I(n+f+g), I(n+f+g+1), \ldots, I(n+f+h)\}$ where $1 < f < g, g < h$. To handle complete occlusion, our method stores the recent past of objects. (Note that object experiencing occlusion are matched only if they are spatially close.) Assume $O_{kp}^b$ is completely occluded in $I(n + f)$ and $O_{lp}^a$ is matched to $O_{l}^a \in I(n + f)$. No object for $O_{kp}^b$ can be thus found in $I(n + f)$. If the recent past of $O_{kp}^b$ shows that it was recently, i.e., in $I(n + f - 1)$, present and occluded, complete occlusion is then assumed and the method continues to carry $O_{kp}^b \in I(n + f)$. If $O_{kp}^b$ becomes later, e.g., in $I(n + f + g)$, visible it should be visible close to an object $O_{ig} \in I(n + f + g)$ where $M_{ii} : O_{lp}^a \rightarrow O_{ig}$. In such case $O_{lp}^a$ and $O_{kp}^b$ are predicted (Eq. 5) in $I(n + f + g)$ and $O_{ig}$ is separated into two objects, $O_{lp}^a, O_{kp}^b$.

Examples of object tracking during occlusion are shown in Figs. 4 and 5. The scene in Fig. 4 shows two objects moving before they occlude. The object segmentation module outputs one segment for both objects when they occlude but the tracking module tracks the two objects during occlusion. Note that in the original frames of these examples (Fig. 9), the objects appear very small and pixels are missing or misclassified due to non-accuracy of the object segmentation used. However, most pixels of the two objects are correctly classified and tracked.
Fig. 4. Examples of tracking objects during occlusion.

Fig. 5. Examples of tracking during occlusion of multiple objects. Upper figures display the original frames and the lower figures the tracked objects.

Fig. 5(a) shows three objects, $O_{id1}$, $O_{id2}$, and $O_{id5}$ where $O_{id1}$ and $O_{id2}$ occlude and the tracking module successfully predicts and separates them. (Note that the displayed segments are not the output of the segmentation but the output of the object separation.) In Fig. 5(b) $O_{id1}$, $O_{id2}$, and $O_{id5}$ occlude and the tracking separates them. In Fig. 5(c), the three objects still occlude but are separated by the tracking module. In Fig. 5(d), only $O_{id1}$ and $O_{id5}$ occlude while $O_{id2}$ is correctly segmented. In Fig. 5(e), the three objects move without occlusion are correctly segmented and tracked.

Note that we do not assume perfect segmentation, but we handle possible segmentation errors at a higher level where more information is available. As the goal of this paper is to track objects context-independently, no application dependent information such as camera position was integrated.

B. Monitoring erroneous object splitting

B.1 Detecting splitting

A segmentation method may split an object into several regions. When an object split, at least one side of the object’s four MBB sides will experience significantly large inward displacement (Fig. 3(b)). Assume $O_{kp} \in I(n - 1)$ is erroneously split in $I(n)$ into $O_{ic1}$ and $O_{ic2}$ and $M_{ki1} : O_{kp} \rightarrow O_{ic1}$. Define
the distance between $O_{ic1}$ and $O_{ic2}$; and $w = (w_x, w_y)$, the current displacement of $O_{kp}$ between $I(n - 2)$ and $I(n - 1)$. (Recall the displacement $t_w$ and the distance $t_d$ thresholds.)

An object split is declared (Eq. 6, Fig. 3(b)) if 1) the current displacement and the displacement of one of the four sides of the MBB of $O_{kp}$ significantly differ, 2) the displacement of that MBB side is inward, and 3) there is an object $O_{ic2}$ close to $O_{ic1}$ in $I(n)$.

\[
\begin{align*}
&\left[(|w_x - w_{hr}| > t_w) \land (w_{hr} < 0) \land (d_{i12} < t_d)\right] \lor \\
&\left[(|w_x - w_{hl}| > t_w) \land (w_{hl} < 0) \land (d_{i12} < t_d)\right] \lor \\
&\left[(|w_y - w_{vl}| > t_w) \land (w_{vl} < 0) \land (d_{i12} < t_d)\right] \lor \\
&\left[(|w_y - w_{vu}| > t_w) \land (w_{vu} < 0) \land (d_{i12} < t_d)\right].
\end{align*}
\] (6)

B.2 Correcting splitting: Region merging

If splitting is detected, then the two regions $O_{ic1}$ and $O_{ic2}$ are merged into one object $O_{ic}$. Region merging is desirable because it reduces the total number of regions to process and thus improve algorithm performance. Regions can be merged based on i) spatial homogeneity features such as texture, ii) temporal features such as motion, or iii) spatial relationships such as inclusion [20], [21].

This paper proposes a merging strategy based on spatial relationships, temporal coherence, and object matching. Assume $O_{kp} \in I(n - 1)$ is erroneously split in $I(n)$ into two sub-regions $O_{ic1}$ and $O_{ic2}$ and the object matching outputs $M_{ki1} : O_{kp} \rightarrow O_{ic1}$. Then $O_{ic2}$ and $O_{ic1}$ are merged, giving $O_{ic}$, if

- $M_{ki} : O_{kp} \rightarrow O_{ic}$, i.e., object voting matches $O_{kp}$ to $O_{ic}$.
- $O_{ic1}$ is spatially close to $O_{ic2}$ and $O_{ic2}$ spatially close to $O_{kp}$.
- The size of the merged object $O_{ic}$ is similar to the size of $O_{kp}$.
- The motion direction of $O_{kp}$ does not significantly change if matched to $O_{ic}$.
- If a split is found on one side of the MBB, then all the displacements of the three other MBB sides of $O_{kp}$ should not change significantly when the two objects are merged.

After merging, the lists of objects in $I(n)$ and $I(n - 1)$ are updated accordingly (e.g., $O_{ic}$ is added to
The good performance of the proposed merging is due to the cooperation between the matching and
merging processes. For example, the matching supports merging by detecting split whereas the merging
supports matching by providing complete (merged) objects for which matching is done recursively. The
advantage of the proposed merging strategy compared to known merging techniques (cf. [20], [21]) is
that it is based on temporal coherence and does not assume fixed object topology.

C. Tracking adaptation

C.1 Threshold selection

The proposed voting scheme requires the definition of some parameters to allow adaptation in case
of feature estimation errors or input video characteristics (e.g., object size, frame rate, or frame size).
We adapt most of these thresholds to parameters of the input video such as frame rate $R$, frame size
$F$, and object size $A$. We propose the following plausibility rules to adapt these parameters:

1. Spatial thresholds are adapted to the frame and object sizes: if objects are large, then a larger error
can be tolerated than when they are small. This adaptation allows better distinction at smaller sizes
and stronger matching at larger sizes and makes thus the thresholds used useful for the entire sequence
where the size of the object varies. As a consequence, the spatial thresholds used in Eqs. 10, 12, 16,
and 17 are adapted following Eq. 7.

$$
t = \begin{cases} 
  t_{\text{min}} & : A \leq A_{\text{min}} \\
  t_{\text{min}} + \frac{(t_{\text{max}} - t_{\text{min}})}{A_{\text{max}}} A + \frac{t_{\text{min}}}{F_{\text{max}}} F & : A_{\text{min}} < A \leq A_{\text{max}} \\
  t_{\text{max}} & : A > A_{\text{max}}
\end{cases} 
$$

(7)

where $A$ represents the object size, $A_{\text{min}}$ ($A_{\text{max}}$) the minimum (maximum) size, $F$ the frame size, and
$F_{\text{max}}$ the maximum (e.g., HDTV) frame size. $t_{\text{min}}$ and $t_{\text{max}}$ were set experimentally. $A_{\text{min}}$ and $A_{\text{max}}$ are
linearly dependent of $F$ as follows (with typical values for $\alpha$ between 0.1 and 0.25):

$$
A_{\text{max}} = \alpha \cdot F, \quad A_{\text{min}} = \beta \cdot F, \quad \beta = 0.0001 \cdot \alpha.
$$

(8)
2. Temporal thresholds used to vote on distances and displacements (Eqs. 1, 4, 14, and 19) are adaptive (Eq. 9) to the frame rate of the input video. This makes the algorithm more sensitive when objects between frames are more distant, i.e., low $R$.

$$t_i = \eta \frac{1}{R}, \quad 190 < \eta < 300.$$  

3. In interlaced video, estimation of vertical motion can be distorted because of the different rasters. To compensate for such distortion, current vertical displacements are compared to estimates from previous fields; if they slightly deviate, then the minimum displacement is selected.

C.2 Temporal consistency

We propose the following plausibility adaptation aiming at temporal tracking consistency:

1. Objects are tracked once they enter (or appear) and also during occlusion.

2. Two objects are matched only if estimated motion direction is consistent to those in previous frames.

3. If, after applying the two-steps voting scheme, two objects in $I(n-1)$ are matched with the same object in $I(n)$, the match with the oldest object (i.e., with the longer trajectory) is selected.

4. If, due to splitting, two objects in $I(n)$ are matched with the same object in $I(n-1)$, the match with the largest size is selected.

5. If, during the matching (e.g., after merging), a better correspondence than a previous one is found, the matching is revised, i.e., the previous correspondence is removed and the new one is established.

6. An object of disjoint regions in $I(n-1)$ and connected in $I(n)$ is tracked and matched with the region in $I(n-1)$ most similar to the newly formed object in $I(n)$.

7. To monitor the change of features over time, the dissimilarity of features between two successive frames is examined and when it grows large this feature is not included in the matching process.

8. If the feature votes of two correspondences ($M_{li}$ and $M_{lj}$) of the same object $O_{lp}$ are equal, then this feature is excluded from the voting process. For example, if shape irregularity features of both correspondences do not differ, this feature is excluded from the voting (e.g., Eq. 17).
9. Consistent object size over time:

(a) If an object $O_i \in I(n)$ has not been tracked (i.e., after both matching and monitoring) for a short period, i.e., its age is larger than an adaptive threshold $t_a = c \cdot R$ with $R$ the frame rate and $0 < c < 1$ then $O_i$ is considered as an artifact object. $O_i$ is then removed regardless of its size. This rule is important if an ”artifact object” was detected, e.g., due to local illumination changes or image clutter.

(b) When an object $O_i \in I(n)$ moves away from the camera, its size will shrink and when it is sufficiently far from the camera its estimated motion is close to zero. In this case we keep $O_i$ even if its size is small as long as $O_i$ has a corresponding object in $I(n - 1)$.

VII. Results

Experiments on more than 3800 indoor and 2200 outdoor frames (of the video sequences ‘Hall’, ‘Stair’, ‘Floor’, ‘FloorDeposit’, ‘FloorRemoval’, ‘Urbicande’, ‘Highway1’, ‘Highway2’, and ‘Survey’) have demonstrated the reliability and real-time response of our method. These sequences include multi-object occlusion, noise, local light changes, and shadow. To test the reliability in the presence of heavy noise, we have applied the proposed method on sequences overlaid with 20, 25, and 30 dB PSNR white noise. As shown in Fig. 6, the method is robust to video noise.

The three steps of our method requires on average 0.14 seconds on a SUN-SPARC-5 for a CIF(352x288) frame using the C programming language. This means that the method process 10 frames per second. The main computational costs is needed for the separation of occluded objects. In the worse case our method requires 0.35 seconds when multiple large objects occlude. The non-linear voting scheme is very fast requiring 0.0001 seconds.

A. Tracking with skipped frames

The reliability of the proposed tracking can be highlighted when frames are skipped, i.e, processing is done on a subset of the original frames. As can be seen in Fig. 7, the objects are reliably tracked
Tracking in $I_{\text{org}}(81)$, in $I_{\text{org}+30\text{dB}}(81)$, and in $I_{\text{org}+25\text{dB}}(81)$

Fig. 6. Noise robustness of the proposed method applied to ‘Hall’ sequence.

Tracking in $I_{\text{org}}(166)$, in $I_{\text{org}+30\text{dB}}(166)$, and in $I_{\text{org}+25\text{dB}}(166)$

Fig. 7. Reliable tracking in the shot ‘Highway1’ with skipped frames.

when five frames are skipped which demonstrates the reliability of the selected thresholds. Note that despite object deformation, i.e., object shape vary over time, the tracking works reliably.

B. Temporal reliability of estimated trajectories

An object trajectory is approximated by that of the centroid of the object. To illustrate the temporal reliability of the proposed algorithm, the estimated trajectory of each object is plotted as a function of the frame number. In Figs. 8-10, the left sub-figure depicts the trajectory of the objects in the image plane while the two other sub-figures show the trajectories for vertical and horizontal directions
Fig. 8. The trajectories of the objects in the sequence ‘Highway1’.

separately. In these figures, ‘StartP’ represents the starting point of a trajectory. Such a plot illustrates the reliability of the tracking method (and allows the interpretation of the behavior of objects [22].)

Fig. 8 shows that objects enter the scene at different times. Some objects move quickly while the others are slower. Objects are moving in both directions: away from the camera and towards the camera. $O_4$ and $O_2$ are moving fast (the trajectory curve increases rapidly). $O_2$ starts at the left of the frame and moves and stops at the edge of the highway. The figure illustrates that $O_2$ is available throughout the whole shot while the other objects start and disappear within the shot.

Fig. 9 shows the trajectories of the ‘Urbicande’ shot where many persons enter and leave the scene. As can be seen, the proposed method is reliable in the presence of very small objects, illumination changes, multiple occluding objects ($O_1$ & $O_6$, $O_1$ & $O_8$), and temporally varying object size. (Note that one person $O_1$ is moving around throughout the whole sequence.

In the ‘Floor’ shot (Fig. 10), an object is being first deposited and then removed. This is a long sequence with complex movements, interlace artifacts, illumination changes and shadows. As can be seen, the trajectory of the objects is complex but our method is able to track the object correctly.

C. Samples of tracking results throughout the video

Our method is reliable in the case of occlusion (e.g., Fig. 12), of object scale variations (e.g., Fig. 9 and 12), of local illumination changes and noise (e.g., Fig. 11) Figs. 11-12 show samples of tracking results where each object is marked by its ID and enclosed by its minimum bounding box.
Fig. 9. The trajectories of the objects in the sequence ‘Urbicande’. The original sequence is rotated by 90° to the right to comply with the CIF format.

Fig. 10. The trajectories of the objects in the sequence ‘Floor’.

Fig. 11 shows how our method differentiates between deposit and split (right image in second row). When an object deposit another object the method interprets this as fault object split as long as the distance is too close (even when the segmentation gives two different regions). Only when the distance is large enough no split is detected. As can be seen from the figure, as soon as the distance between the person and the bag is large enough the bag is detected. This shot includes local illumination changes and object shadows but the proposed algorithm works correctly.

In Fig. 12, our method tracks three objects that occlude at some point. Two objects enter the scene in the first frame while the third object enters around the 70th frame. $O_1$ moves horizontally to the left and vertically down, $O_2$ moves horizontally right and vertically up, and $O_5$ moves fast to left. Despite the multi-object occlusion ($O_1$, $O_2$ and $O_5$), light changes, and reflections (e.g., car surfaces) the algorithm stays reliable. Because of the static traffic sign, the change detection divides the object into two regions, the tracking recovers properly, however.
D. Reliability and limitations of the proposed approach

In this paper, we tested our method on frames without global motion and global illumination changes. If the appearances of an object over time significantly differ due to heavy global light or viewpoint changes, it cannot be matched. We tested our method, however, on frames with wide local illumination changes (e.g., ‘Stair’ and ‘Survey’ shots).

Due to various conditions such as noise, feature deviations are possible. Our method uses thresholds to compensate for possible deviations. The thresholds used depend on parameters of the input video such as frame rate, frame size, object size, and maximum displacement (Sec. VI-C). For example, if objects (or frame) are large then a larger error allowance is tolerated. This adaptation allows a better distinction at smaller sizes and a stronger matching at larger sizes. This adaptation makes the thresholds useful also for the entire sequence and for different sequences.

To detect occlusion or split, our method monitors changes at all sides of the object bounding box, i.e., we use both the vertical and horizontal displacements, and uses the distance between objects.
We have tested our method with video shots overlaid with heavy white noise (Fig. 6) or including MPEG-2 artifacts. No significant tracking errors were found. Some object boundaries deviate, however, by few pixels not affecting the overall performance of the method.

The focus of our method is on real-time response and reliability where errors of object segmentation are compensated for. The compensation of errors of low-level steps is done at higher steps when more reliable information is available. For example, the voting corrects object segmentation through merging or splitting of erroneously segmented regions. We have shown that even if the segmentation delivers faulty results the tracking is able to recover (e.g., Figs. 12 and 9).

To test our method, we used test sequences commonly referenced in related work. Our method has been applied to track multiple objects but it was not tested in crowded scenes. Fig. 9 shows, however, how our method successfully tracks very small objects in a busy street scene. Our method successfully handles objects of various sizes, and also situations when object size changes throughout the video shot. In most sequence used there are multiple objects and/or occlusion. Object fusion or splitting occurs in all sequences used.

Our simulations show that the features used, although simple, are adequate to solve the occlusion problem even over an extended period of time (e.g., the sequences ‘Urbicande’ or ‘Stair’). First, the shape features used are reliable as they include different aspects of the object shape. Second, the feature monitoring step corrects feature errors when the segmentation module fails. This monitoring allows to track and update objects over time. Third, our voting strategy uses two steps to check the tracking correctness: matching of objects and matching of correspondences. Finally, the thresholds used are adaptive to the object size, frame sizes, and frame rate making the features reliable over time. Examples include the ‘Urbicande’ and the ‘Stair’ sequences where some objects remain in the scene for an extended period of time, experience occlusion, and have temporally varying shapes.

The method reliability is due to (a) the non-linear two step (object and correspondence) voting, (b) the parameter adaptation and plausibility rules for temporal reliability and consistency, and (c) the
explicit detection of object occlusion and segmentation errors.

VIII. Conclusion

This paper proposes a method for tracking multiple moving objects reliably in the presence of noise, occlusion, and segmentation errors. No template or model matching is used but rather rules that are independent of specific object appearance. No constraints regarding object motion and camera position are imposed. The method is based on a non-linear voting scheme that accounts for multiple matches by a two step approach. In the first step, objects are matched whereas in the second step established correspondences are examined and eventually reset. In addition, a correspondence confidence measure is maintained until the method decide about the correct correspondence.

The problem of occlusion and segmentation error is solved by monitoring object features over time. If strong deviations are detected, occlusion or other segmentation errors are assumed and an correction scheme is then activated. Plausibility rules for temporal consistency and error tolerance are proposed for reliable tracking over long periods of time. The thresholds used are adaptive to the input video characteristics such frame rate and object size.

Experimental results have shown that our approach simultaneously tracks multiple objects, also those with temporally varying shapes. Our approach can handle heavy occlusion, object appearance, entrance, re-appearance, disappearance, and exit.

We are currently testing our method with video shots of moving camera using global motion compensation. We are also introducing a module to handles object re-entrance.

Appendix

I. Object voting

To compute the similarity and the dissimilarity measures $s$ and $q$ between two objects $O_{ic} \in I(n)$ and $O_{lp} \in I(n - 1)$ (Sec. V-A), the following feature votes for shape, size, and motion are applied independently. Note that the total number of votes $v$ is $3 + 3 + 2$. 
A. Distance vote

Note that in Eq. 1, \(d_{li} < t_r\), where \(d_{li}\) is the distance between \(O_{lp}\) and \(O_{ic}\). This means \(O_{lp}\) and \(O_{ic}\) must be spatially close (within a search range of radius \(t_r\)) for the following size, shape, and motion votes to take place.

B. Size vote

Recall that the size of an object \(O_{ic}\) is represented by the area \(A_{ic}\), perimeter \(P_{ic}\), maximum horizontal extent \(W_{ic}\), and maximum vertical extent \(H_{ic}\). The similarity (dissimilarity) measure increases depending on the size features as follows:

\[
\begin{align*}
    s_{++} & : (p_{ali} > t_s); \\
    s_{++} & : (p_{hi} > t_s); \\
    s_{++} & : (p_{wi} > t_s); \\
    q_{++} & : (p_{ali} \leq t_s); \\
    q_{++} & : (p_{hi} \leq t_s); \\
    q_{++} & : (p_{wi} \leq t_s);
\end{align*}
\]

where

- \(s_{++}\) and \(q_{++}\) represent an increase of the similarity and dissimilarity measures, \(s\) and \(q\), by one vote;
- \(0 < t_s < 1\) the size threshold (see adaptation in Sec. VI-C); and

\[
p_{ali} = \begin{cases} 
    \frac{A_{lp}}{A_{ic}}; & A_{lp} \leq A_{ic} \\
    \frac{A_{ic}}{A_{lp}}; & A_{lp} > A_{ic}
\end{cases}
\quad p_{hi} = \begin{cases} 
    \frac{H_{lp}}{H_{ic}}; & H_{lp} \leq H_{ic} \\
    \frac{H_{ic}}{H_{lp}}; & H_{lp} > H_{ic}
\end{cases}
\quad p_{wi} = \begin{cases} 
    \frac{W_{lp}}{W_{ic}}; & W_{lp} \leq W_{ic} \\
    \frac{W_{ic}}{W_{lp}}; & W_{lp} > W_{ic}
\end{cases}
\]  

(11)

C. Shape vote

Recall that the object shape is represented the MBB \(B_{O_{ic}}\), extent ratio \(e_{ic}\), compactness \(c_{ic}\), and irregularity \(r_{ic}\). The similarity (dissimilarity) measure of \(O_{lp}\) and \(O_{ic}\) increases as follows:

\[
\begin{align*}
    s_{++} & : (d_{e_{li}} \leq t_p); \\
    s_{++} & : (d_{c_{li}} \leq \frac{t_p}{2}); \\
    s_{++} & : (d_{r_{li}} \leq 2t_p); \\
    q_{++} & : (d_{e_{li}} > t_p); \\
    q_{++} & : (d_{c_{li}} > \frac{t_p}{2}); \\
    q_{++} & : (d_{r_{li}} > 2t_p);
\end{align*}
\]

where

- \(0 < t_p < 1\) the shape threshold which is a function of the frame and object sizes (Sec. VI-C).
- \(e_{lp}, c_{lp}, r_{lp} (e_{ic}, c_{ic}, r_{ic})\) are the extent ratio, compactness, and irregularity of \(O_{lp} (O_{ic})\) shape.
- \(d_{e_{li}} = |e_{lp} - e_{ic}|; \quad d_{c_{li}} = |c_{lp} - c_{ic}|; \quad d_{r_{li}} = |r_{lp} - r_{ic}|.\)
D. Motion vote

Depending on the motion direction, the similarity measures $s$ increases if $O_{ic} \in I(n)$ would not reverse its direction if matched to $O_{lp} \in I(n-1)$ as follows:

\[
\begin{align*}
    s_{++} & : \left( (w_{cxi} \geq 0 \land w_{xil} \geq 0) \lor (w_{cxi} \leq 0 \land w_{xil} \leq 0) \right) \\
    s_{++} & : \left( (w_{cyi} \geq 0 \land w_{yil} \geq 0) \lor (w_{cyi} \leq 0 \land w_{yil} \leq 0) \right) \\
    q_{++} & : \left( (w_{cxi} \geq 0 \land w_{xil} < 0) \lor (w_{cxi} < 0 \land w_{xil} \geq 0) \lor (w_{cxi} < 0 \land w_{xil} < 0) \right) \\
    q_{++} & : \left( (w_{cyi} \geq 0 \land w_{yil} < 0) \lor [(w_{cyi} < 0 \land w_{yil} < 0) \lor (w_{cyi} < 0 \land w_{yil} \geq 0)] \right)
\end{align*}
\]

where

- $w_{cxi}, w_{cyi}$, the current horizontal and vertical displacements of $O_{ic}$ and
- $w_{xil}, w_{yil}$, the horizontal and vertical displacements of $O_{ic}$ if matched to $O_{lp}$.

II. Correspondence voting

To resolve matching ambiguities (Sec. V-B), a vote between two correspondences $M_{li} : O_{lp} \to O_{ic}$ and $M_{lj} : O_{lp} \to O_{jc}$ is applied to compute the similarity measures $s_{li}$ and $s_{lj}$ between these two correspondences. This correspondence vote uses the feature vote functions independently for distance, confidence, size, shape, direction, displacement, and direction-displacement. For stable voting, if the feature votes of two correspondence ($M_{li}$ and $M_{lj}$) of the same object $O_{lp}$ are equal, then this feature is excluded from the voting process. For example, the difference of shape irregularity feature should be $d_{rij} = |r_{li} - r_{lj}| > t_{pc}$ to include it in the vote (Eq. 17).

A. Distance vote

The similarity correspondence measures $s_{li}$ and $s_{lj}$ increase as follows:

\[
\begin{align*}
    s_{li++} & : (d_{d_{ij}} > t_{dc}) \land (d_{li} < d_{lj}) \\
    s_{lj++} & : (d_{d_{ij}} > t_{dc}) \land (d_{li} > d_{lj})
\end{align*}
\]

where $d_{li}$ is the distance between $O_{lp}$ and $O_{ic}$; $d_{lj}$ is the distance between $O_{lp}$ and $O_{jc}$; $d_{d_{ij}} = |d_{li} - d_{lj}|$ is the distance difference; and $t_{dc} > 1$ is the distance threshold (see adaptation in Sec. VI-C).
The condition $d_{ij} > t_{dc}$ ensures that only if the two features differ significantly the vote is applied; if the features do not differ significantly then neither $s_{li}$ nor $s_{lj}$ are increased. Thus the total number of votes $v$ varies depending on feature deviations.

B. Confidence vote

The similarity correspondence measure increases as follows:

$$
\begin{align*}
  s_{li} & \leftarrow \begin{cases} 
    \left[ (d_{\zeta} > t_{\zeta c}) \land (\zeta_{li} > \zeta_{lj}) \right] & 
  
  s_{lj} & \leftarrow \begin{cases} 
    \left[ (d_{\zeta} > t_{\zeta c}) \land (\zeta_{li} < \zeta_{lj}) \right] & 
\end{align*}
$$

where $d_{\zeta} = |\zeta_{li} - \zeta_{lj}|$ is the confidence difference; and $0 < t_{\zeta c} < 1$ is the confidence threshold. $d_{\zeta} > t_{\zeta c}$ means only if the two features significantly differ the vote is applied.

C. Size vote

Depending on the size features, the similarity correspondence measures increase as follows:

$$
\begin{align*}
  s_{li} & \leftarrow \left[ (d_{a_{ij}} > t_{sc}) \land (p_{a_{li}} < p_{a_{lj}}) \right]; \\
  s_{li} & \leftarrow \left[ (d_{h_{ij}} > t_{sc}) \land (p_{h_{li}} < p_{h_{lj}}) \right]; \\
  s_{li} & \leftarrow \left[ (d_{w_{ij}} > t_{sc}) \land (p_{w_{li}} < p_{w_{lj}}) \right]; \\
  s_{lj} & \leftarrow \left[ (d_{a_{ij}} > t_{sc}) \land (p_{a_{li}} > p_{a_{lj}}) \right]; \\
  s_{lj} & \leftarrow \left[ (d_{h_{ij}} > t_{sc}) \land (p_{h_{li}} > p_{h_{lj}}) \right]; \\
  s_{lj} & \leftarrow \left[ (d_{w_{ij}} > t_{sc}) \land (p_{w_{li}} > p_{w_{lj}}) \right].
\end{align*}
$$

where (see Eq. 11) $d_{a_{ij}} = |p_{a_{li}} - p_{a_{lj}}|$ is the area difference; $d_{h_{ij}} = |p_{h_{li}} - p_{h_{lj}}|$ is the height difference; $d_{w_{ij}} = |p_{w_{li}} - p_{w_{lj}}|$ is the width difference; and $0 < t_{sc} < 1$ is the size threshold (see adaptation in Sec. VI-C). Note that if the features do not differ significantly then neither $s_{li}$ nor $s_{lj}$ are increased.
D. Shape vote

Following the shape vote, $s_{li}$ and $s_{lj}$ increase as follows:

\[
\begin{align*}
    s_{li} & : [ \{(d_{ei_{ij}} > t_{pc}) \land (p_{eic} < p_{ejc})\}] ; \\
    s_{lj} & : [ \{(d_{cj_{ij}} > t_{pc}) \land (p_{cic} < p_{cjc})\}] ; \\
    s_{ri} & : [ \{(d_{ri_{ij}} > t_{pc}) \land (p_{ric} < p_{rjc})\}] ; \\
    s_{ri} & : [ \{(d_{ei_{ij}} > t_{pc}) \land (p_{eic} > p_{ejc})\}] ; \\
    s_{ri} & : [ \{(d_{cj_{ij}} > t_{pc}) \land (p_{cic} > p_{cjc})\}] ; \\
    s_{ri} & : [ \{(d_{ri_{ij}} > t_{pc}) \land (p_{ric} > p_{rjc})\}] .
\end{align*}
\]

where (see Eq. 12) $d_{ei_{ij}} = |d_{ei_{i}} - d_{ei_{j}}|$ is the extent difference; $d_{ci_{ij}} = |d_{ci_{i}} - d_{ci_{j}}|$ is the compactness difference; $d_{ri_{ij}} = |d_{ri_{i}} - d_{ri_{j}}|$ is the irregularity difference; and $0 < t_{pc} < 1$ the shape threshold (see Sec. VI-C). If the features do not differ significantly then neither $s_{li}$ nor $s_{lj}$ are increased.

E. Motion direction vote

The similarity correspondence measures increase based on the motion direction as follows:

\[
\begin{align*}
    s_{li} & : [ \{(\delta_{x_{li}} = \delta_{x_{cl}}) \land (\delta_{x_{li}} = \delta_{x_{pl}}) \land (\delta_{y_{li}} = \delta_{y_{ul}})\} \lor [(\delta_{y_{li}} = \delta_{y_{cl}}) \land (\delta_{y_{li}} = \delta_{y_{pl}}) \land (\delta_{y_{li}} = \delta_{y_{ul}})]] ; \\
    s_{lj} & : [ \{(\delta_{x_{jc}} = \delta_{x_{cl}}) \land (\delta_{x_{jc}} = \delta_{x_{pl}}) \land (\delta_{y_{jc}} = \delta_{y_{ul}})\} \lor [(\delta_{y_{jc}} = \delta_{y_{cl}}) \land (\delta_{y_{jc}} = \delta_{y_{pl}}) \land (\delta_{y_{jc}} = \delta_{y_{ul}})]] .
\end{align*}
\]

where

- $\delta_{cl} = (\delta_{x_{cl}}, \delta_{y_{cl}})$ is the current direction of $O_{lp}$;
- $\delta_{pl} = (\delta_{x_{pl}}, \delta_{y_{pl}})$ is the previous direction of $O_{lp}$;
- $\delta_{ul} = (\delta_{x_{ul}}, \delta_{y_{ul}})$ is the past-previous direction of $O_{lp}$;
- $\delta_{li} = (\delta_{x_{li}}, \delta_{y_{li}})$ is the direction of $O_{lp}$ if it is matched to $O_{ic}$; and
- $\delta_{lj} = (\delta_{x_{lj}}, \delta_{y_{lj}})$ is the direction of $O_{lp}$ if matched to $O_{jc}$.
F. Displacement vote

Since motion is an important feature, the displacement vote may contribute one, two, or three votes to the matching process depending on the displacement difference $d_w$ as follows

\begin{align}
  s_{li+1}: & \quad (d_w > t_{dw}) \land (d_w < t_{dw_{min}}) \land (w_i < w_j) \\
  s_{li+2}: & \quad (d_w > t_{dw}) \land (t_{dw_{min}} < d_w < t_{dw_{max}}) \land (w_i < w_j) \\
  s_{li+3}: & \quad (d_w > t_{dw}) \land (d_w > t_{dw_{max}}) \land (w_i < w_j) \\
  s_{lj+1}: & \quad (d_w > t_{dw}) \land (d_w < t_{dw_{min}}) \land (w_i > w_j) \\
  s_{lj+2}: & \quad (d_w > t_{dw}) \land (t_{dw_{min}} < d_w < t_{dw_{max}}) \land (w_i > w_j) \\
  s_{lj+3}: & \quad (d_w > t_{dw}) \land (d_w > t_{dw_{max}}) \land (w_i > w_j)
\end{align}

where

- $w_i = (w_{xli}, w_{yli})$ is the displacement of $O_{lp}$ if it is matched to $O_{lc}$;
- $w_j = (w_{xlj}, w_{ylj})$ is the displacement of $O_{lp}$ if it is matched to $O_{jc}$;
- $d_w = ||w_i| - |w_j||$ is the displacement difference;
- $t_{dw}$ is the displacement threshold which decreases with higher $R$ (Sec. VI-C) and increases with segmentation errors;
- $t_{dw_{min}}$ and $t_{dw_{max}}$ are adapted 1) to segmentation error, i.e., they increases in case of occlusion and 2) to $w_{max}$, the maximum possible displacement (see Eq. 1) with $t_{dw_{min}} = \frac{u_{max}}{\gamma}$, $t_{dw_{max}} = u_{max}$ where $\gamma$ increases in case of occlusion.

Here $d_w > t_{dw}$ means that the displacements have to differ significantly to be considered for voting.

G. Extra motion vote

If after applying the previous votes $s_{li} = s_{lj}$, the measures increase as follows:

\begin{align}
  s_{li++}: & \quad (w_i < w_j) \land (\delta_{xlc} = \delta_{xcl}) \land (\delta_{ycl} = \delta_{ycl}) \\
  s_{lj++}: & \quad (w_i > w_j) \land (\delta_{xjc} = \delta_{xcl}) \land (\delta_{ycl} = \delta_{ycl})
\end{align}
REFERENCES


