Visual object tracking by an evolutionary self-organizing neural network

José Everardo B. Maiaa,∗, Guilherme A. Barretoa and André L.V. Coelhoc

aDepartment of Statistics and Computing, State University of Ceará (UECE), Fortaleza, Ceará, Brazil
bDepartment of Teleinformatics Engineering, Federal University of Ceará (UFC), Fortaleza, Ceará, Brazil
cGraduate Program in Applied Informatics, University of Fortaleza (UNIFOR), Fortaleza, Ceará, Brazil

Received: 23 August 2010

Abstract. In this paper, a recently proposed evolutionary self-organizing map is extended and applied to visual tracking of objects in video sequences. The proposed approach uses a simple geometric template to track an object executing a smooth movement represented by affine transformations. The template is selected manually in the first frame and consists of a small number of keypoints and the neighborhood relations among them. The coordinates of the keypoints are used as the coordinates of the nodes of a non-regular grid defining a self-organizing map that represents the object. The weight vectors of each node in the output grid are updated by an evolutionary algorithm and used to locate the object frame by frame. Qualitative and quantitative evaluations indicate that the proposed approach present better results than those obtained by a direct method approach. Additionally, the proposed approach is evaluated under situations of partial occlusion and self-occlusion, and outliers, also presenting good results.

1. Introduction

Visual tracking is the act of consistently locating a region in each image of a video sequence that matches the given object [40]. It is a critical step in many machine vision applications such as surveillance [14], driver assistance systems [2], remote sensing, defense systems [9], human-computer interactions [29]. Usually, the object visual tracking problem is formulated as one of the two following approaches: Bayesian tracking or detection-based tracking. In Bayesian tracking, a motion dynamic model of the object must be known in advance and the tracking algorithm is composed of two steps: one-step-ahead prediction of the object position and updating of its position based on sensory measurements using Bayes theorem. Typical algorithms used by this approach are Kalman filter [7], extended Kalman filter [42] and particle filters [12]. The second approach does not require a movement model to be specified in advance. Instead, it performs detection and recognition of the object in each frame based on features or an object model that makes it possible to distinguish the object from the background and other irrelevant objects.

An important feature extraction method used in many benchmarking studies is the scale-invariant feature transform (SIFT) [20]. Using SIFT, descriptor vectors computed from local histograms of gradients have been shown to be robust and discriminative enough even when few data vectors are available and when the plain nearest-neighbor approach is used [26]. SIFT distinctive power is the result of a mixing of crudely localized information and the distribution of gradient related features, while fending off the effects of localization errors in terms of scale or space. The use of relative strengths and orientations of gradients also reduces the effect of photometric changes.

∗Corresponding author. E-mail: jmaia@uece.br (J.E.B. Maia), guilherme@deti.ufc.br (G.A. Barreto), acoelho@unifor.br (A.L.V. Coelho).

1064-1246/11/$27.50 © 2011 – IOS Press and the authors. All rights reserved
The Speeded Up Robust Features (SURF) descriptor [1] is based on discriminative properties similar to those of SIFT, but with reduced complexity. The SURF method is implemented in two steps. The first step consists of fixing a reproducible orientation based on information from a circular region around the interest point. Then, the method constructs a square region aligned to the selected orientation, and extract the SURF descriptor from it.

Randomized Trees (RTs) [19] rely on statistical learning techniques to model the set of possible appearances of a patch. Since the set of possible patches around an image feature under changing perspective and lighting conditions can be seen as a class, it is possible to train a set of RTs to recognize feature points by feeding it on samples of their possible appearances, synthesized by warping the patches found in a training image using randomly chosen homographies. This approach is fast and effective to matching a planar target. Note that, unlike traditional classification problems, a close-to-perfect method is not required, since it is enough to recognize some features successfully and to use a robust estimator such as the RANSAC (Random Sample Consensus) algorithm [10] to detect the object. However, a scalable approach is still highly desirable for practical applications since the number of feature points might become very large (typically >400). In such situations, RTs performances tend to deteriorate.

The aforementioned methods belong to the category of “feature-based methods” since they first extract a sparse set of distinct features from each image separately, and then recover and analyze their correspondences in order to determine the motion of the object. Direct (i.e. pixel-based) methods, on the contrary, recover unknown motion parameters directly from image quantities at each pixel in the image, such as image brightness, brightness-based cross-correlation, among others [17]. A shortcoming of these frameworks is that structural information, such as geometric and topological relations among points, is not used by them. The idea of using structural information to improve the detection of keypoints has been successfully used by many authors (see, for example, [13, 16, 28, 30, 35–39]).

The work of Schmid and Mohr [30] used geometric constraints to refine keypoint classification, while the work in [37] obtained substantial improvements in the results by using topological constraints. A probabilistic modeling scheme is proposed in [28] where small groups of keypoints are considered by using non-hierarchical structures, referred to as ferns, to classify the patches within the classical naive Bayes framework. Each fern consists of a small set of binary tests and returns the probability that a patch belongs to any one of the classes that have been learned during training. These responses are then combined in a Naive Bayesian way. As in [19], the classifier is trained by synthesizing many views of the keypoints extracted from a training image as they would appear under different perspective or scale.

Since graphs are mathematical objects that naturally model relations, some studies have been using graphs built from keypoints to detect objects. The works in [35, 36] shift away from the classification approach and tries to solve the problem with graph matching, achieving successful results with dynamic graphs defined over SIFT points. In [13], the authors use attributed relational graphs (ARGs) to represent objects, which carry both local and relational information about them. The recognition is performed by inexact graph matching, which consists in finding an approximate homomorphism between ARGs derived from a target image and a model image.

In the approach by [16] the problem is reduced to supervised classification, which is more efficient than graph matching. Instead of classifying single keypoints, it classifies sets of keypoints using both appearance and structural information. The entities are graphs of keypoints, referred to as keygraphs. The weakness of this approach is that keygraphs are application dependent and difficult to build.

Finally, an adaptive geometric template-based method for robust motion recovery from features is proposed in [39]. A geometric template consists of nodes containing salient features (e.g. corner features). The spatial configuration of the features is modeled using a spanning tree. The authors proposed an iterative data association method to estimate the template structure in conjunction with its individual features deforming the template with target motion due to translation. The structure is altered (i.e. adapted) by new features added to the target or the removal of untracked features.

Self-organizing neural networks were used in [33, 34] in the task of attributed relational graphs (ARGs) matching. The application focus is shaping indexing for image recovery in database querying. In this method, each shape is represented with line segments. For each pair of line segments inside a given neighborhood, a number of scale, rotation and translation invariant features are extracted. The algorithm in [34] is based on Self-Organizing Map (SOM) network and in [33] is based on Neural Gas network. The main drawback of
both is that the construction of the ARG is preprocessing intensive for segmentation and extraction of image features which reduces the potential applications of the algorithms for tracking purposes.

In this paper, we propose a strategy that consists in using simple representations for the patch centered on keypoints and computationally efficient measures of matching to compare patches. For this purpose, we build a self-organizing map on the object model image with the nodes located in the keypoints using distance and the neighborhood relations to connect them. A common phenomenon in patch matching based tracking is the drift of the points along successive frames. The neighborhood relations imposed by the topographic map impose constraints that prevent the drift of the points in successive iterations. In our approach the quality of a solution considers the matching of all patches and also the correlation of distances between points in the center of each patch. Unlike other approaches [13, 16, 35, 36], which build classes of small graphs on the different regions of the object, we build only a self-organizing map to track it in its evolution frame by frame. It is worth emphasizing that in the proposed approach perfect matching is not required to track the object. To the best of our knowledge, the proposed approach differs considerably from previous works.

The remainder of this paper is organized as follows. In Section 2 and its subsections the problem is defined and the proposed approach is introduced. In Section 3 the simulation results are presented and discussed. The article is concluded in Section 4.

2. The proposed approach

There are three major issues in feature-based visual tracking of objects: 1) how to represent an object of interest so that it can effectively be discriminated from background and other non-relevant objects; 2) how to locate and/or recognize the object in each frame in a frame sequence; 3) how to dynamically update the representation (feature- or model-based) to take into account changes in object’s appearance and structure due to modifications in the surroundings of the tracked object. Although there are several approaches to address each of these tasks separately, they are closely related in the sense that the choice of a representation for the object limits the range of appropriate measures of matching and update mechanisms. The following sections detail the object representation used in this work and the proposal of a joint solution to the problems of locating the object and updating its representation in a frame-by-frame basis.

2.1. Problem definition

Firstly, we need to define the reference, current and candidate templates. Let \( I = \{I_0, I_1, \ldots, I_i\} \) be a sequence of indexed images and \( T_0, T_1, \ldots, T_i \) are gray-level intensities of templates defined on these images. The template (or patch) defined in the first frame, \( T_0 \), is referred to as the reference template (or the reference patch). When tracking from frame \( i \) to frame \( i+1 \), we refer to frame \( i \) as the current frame, and the template within this frame, \( T_i \), as the current template. The frame \( i+1 \) is referred to as the target frame, and a template within this frame, \( T_{i+1} \), as a candidate template.

Additionally, let the Sum of Squared Differences (SSD) to be used as a measure of matching between templates. Also, let \( x \in T_0 \) be a feature point in the corresponding template. Thus, the problem of finding a transformation parameter vector \( \hat{p} \) between \( T_0 \) and \( T_i \) is formulated using SSD as

\[
\hat{p} = \arg \min_{p} \sum_{x \in T_0} [T_i(x') - T_0(x)]^2, \quad (1)
\]

\[
= \arg \min_{p} \sum_{x \in T_0} [T_i(w(x, p)) - T_0(x)]^2, \quad (2)
\]

where \( x' = w(x, \hat{p}) \) is the projection of the feature point \( x \in T_0 \) onto the current frame \( i \).

The SSD-based tracking problem can thus be stated as the task whose goal is to select and track feature points from images \( I_0 \) to \( I_{i+1} \). Assuming that the transformation \( w(x, \hat{p}) \) from frame 0 to the current frame \( i \) is known, the problem reduces to finding an increment \( \Delta p \) for the transformation parameter vector between \( T_i \) and \( T_{i+1} \) through an iterative method that solves

\[
\Delta \hat{p} = \arg \min_{\Delta p} \sum_{x \in T_i} [T_{i+1}(w(x', \Delta \hat{p})) - T_i(x')]^2. \quad (3)
\]

Then, by function composition, we find the whole transformation imposed to the feature point \( x \in T_i \) from frame 0 (reference) to frame \( i+1 \) (target):

\[
x' = w(x', \Delta \hat{p}) \circ w(x, \hat{p}) \quad (4)
\]

where the feature point \( x' \) belongs to the current frame \( i \) (i.e. \( x' \in T_i \)).

The transformation \( w : R^2 \rightarrow R^2 \) is the warp function corresponding to a transformation whose para-
ters are specified in \( p \). Usually, it is given by an affine transformation:

\[
\begin{pmatrix}
    x' \\
y'
\end{pmatrix} = \begin{pmatrix}
    s \cos \theta & s \sin \theta \\
-s \sin \theta & s \cos \theta
\end{pmatrix} \begin{pmatrix}
x \\
y
\end{pmatrix} + \begin{pmatrix}
b_x \\
b_y
\end{pmatrix}
\]

(5)

(6)

where the matrix \( A \in \mathbb{R}^{2 \times 2} \) accounts for rotations and scaling, since \( \theta \) is an angle of rotation and \( s \) is a scale factor, while \( x = (x, y) \), \( x' = (x', y') \) and \( b = (b_x, b_y) \) denote respectively the original positions, the transformed ones and a translation vector.

By formulating the tracking problem as in Eq. (1), we assume that the object is executing a smooth movement whose evolution from one frame to the next is represented by affine transformations whose parameter vectors \( p = (b_x, b_y, \theta, s) \). This condition is approximately satisfied when the object is rigid, the camera is stationary and the object’s movement is slow relative to the video frame rate such that location, scalar velocity and direction of motion of a given point change little from one frame to the next. Despite these simplifying assumptions, this approach is representative of a wide range of applications in surveillance, human-computer interaction and visual servoing.

2.2. Object representation

In this paper, we tackle the tracking problem as a problem of matching detected keypoints between successive frames. By keypoint we denote a point in an image that is sufficiently different from its neighbors so that it can be easily distinguished from other similarly extracted points in the same or another image. Also, the same point must be detected as a keypoint even when the image undergoes some changes due to illumination, viewpoint, random noise, image blur or compression.

It is assumed that a small neighborhood is also moving together with the point and therefore a small image patch around the point, called model patch, can be considered for analysis. Although a number of methods to extract keypoints in images has been developed [15, 25], in this work the keypoints are manually marked by the user on the object image in the first frame. Examples of keypoints that have been successfully used by previous studies are L-corners, T-junctions, white dots in black background, among others.

By taking the keypoints selected as vertices, a template in the form of an undirected graph (or grid) is built to represent the object (see Fig. 1). A planar grid is built by establishing links between some of the keypoints, which impose significant constraints on the geometric appearance of the object. Although the links can be arbitrary, distances and neighborhood relations between the keypoints should be considered to define the links. The vertices of the template can then be interpreted as the coordinates of a non-regular grid defining a self-organizing feature map (SOFM) that represents the object. The weight vectors of each node in the output grid are updated by an evolutionary algorithm and used to locate the object in a frame by frame basis. For this hypothetical example, we defined five keypoints (and, hence, five patches) and eight links to build the SOFM.

It is worth emphasizing the rationale behind the use of the term “non-regular grid”. For the standard SOM network [18], the output grid is regular, in the sense that it has a well-defined geometric structure, e.g. rectangular, cylindrical or toroidal, and the coordinates of the nodes are located at equally-spaced positions, so that the distances between neighboring coordinates are equal. In the proposed approach, the coordinates of the nodes correspond to the position of the chosen keypoints, which do not need to be equally-spaced one from another. The only constraint is that, once the coordinates of the keypoints have been chosen, the neighborhood relations between them should be preserved.

Figure 2 shows typical aspects and positions that the ‘kite’-shaped template shown in Fig. 1 can assume when subjected to affine transformations. This synthetic template is used in one of our experiments to represent the object to be tracked. In the next section, we summarize the theory of the evolutionary self-organizing
where \( \tilde{w} \) denotes the number of data samples, \( w \) weight vectors, and the parameters data set. Mathematically, QE is defined as

\[
\text{QE} = \left( \frac{1}{L} \sum_{l=1}^{L} \| x_l - w_{l} \| \right),
\]

where \( i(x_l) \) is the winning neuron for pattern vector \( x_l \), being determined as

\[
i(x_l) = \arg \min_{j} \| x_l - w_{j} \|
\]

where \( \| \cdot \| \) denotes the euclidean norm.

Since \((r_{m}, r_{n})\) are the coordinates of pairs of nodes in the output grid and \((w_{m}, w_{n})\) are the corresponding pairs of weight vectors, the PCC index is the cross-correlation of the distances between nodes in the output space, \(d(r_{m}, r_{n})\), and the distances between weight vectors in the input space, \(d(w_{m}, w_{n})\). Mathematically, we have

\[
\text{PCC}(\tilde{W}) = \frac{\sum_{m=1}^{N} \sum_{n=1}^{N} d(r_{m}, r_{n}) d(w_{m}, w_{n})}{(N-1) \sum_{m=1}^{N} \sum_{n=1}^{N} d(r_{m}, r_{n}) d(r_{m}, r_{n})},
\]

where \( S_r \) and \( S_w \) are respectively the standard deviations of \([d(r_{m}, r_{n})] \) and \([d(w_{m}, w_{n})] \), \( m, n = 1, \ldots, N \).

The larger the value of PCC, the higher the correlation between distances in the output space, \(d(r_{m}, r_{n})\), and distances in the input space, \(d(w_{m}, w_{n})\). The smaller the value of \(QE\), the better the quantization of the input space. By reversing the sign of the \(QE\) index, this multi-objective optimization problem reduces to the maximization of the single index PCC.

The evolutionary algorithm comprises the following usual steps [27]: 1. Generate the initial population, 2. Evaluate the fitness of each individual in the population, 3. Select probabilistically the best-ranked individuals to mate, 4. Allow the selected individuals to mate at a given crossover probability, 5. Allow the generated offspring to have their genes mutated at a given probability, 6. Evaluate the fitness values of the offspring, 7. Replace worst ranked part of previous population with the generated offspring, 8. Repeat 3–7, until termination. Some specific operators were developed for the problem to speed up the convergence of the algorithm. In the next section a variant of the EvSOM algorithm is developed and applied to the tracking problem.

2.4. Object location and representation updating

In this paper we introduce an object tracking algorithm that uses a variant of the EvSOM algorithm to update the object template. A parameter vector is associated with each node of the output grid which projects the corresponding model patch in the current frame \(i\) onto the target frame \(i+1\). A possible strategy for evolutionary search for a solution to the problem of tracking is to search for the best fitness in some range...
of values within the parameter vector space \( p \). A similar approach is used in [4, 8, 23] to handle the problem of medical image registration. In this paper, however, we develop a different two-stage approach, for which the smooth motion between successive frames is an important assumption.

The input to each tracking stage is the updated template resulting from the previous stage. This template indeed defines the EvSOM topology, i.e., the keypoints correspond to the coordinates \( v_f, j = 1, \ldots, N \), of the nodes comprising the EvSOM output grid (see Subsection 1). The weight vector \( w_j = p_j = (b_j^0, b_j^1, \ldots, \beta_j^0, \beta_j^1) \) represents the parameters of an affine transformation that projects the coordinates of the \( j \)-th keypoint in the current frame onto the next frame. The coordinates of the projected keypoints correspond to the new coordinates of the nodes comprising the output grid of the EvSOM for the next frame.

At the first stage, the algorithm selects a set of candidate patches at each keypoint. These candidate patches are randomly searched in the current frame (i.e., frame \( i \)) in the vicinity of the \( j \)-th model patch of the frame \( i \) (see Fig. 3). Thus, for a template with \( N \) keypoints, the result of the search process is \( N \) sets of candidate patches. It is worth pointing out that the first stage of the proposed algorithm aims at transforming the search space into a discrete set of candidate solutions. For a ‘candidate solution’ we mean a set of new node coordinates for the EvSOM, which is equivalent to new positions for the template keypoints.

The maximum number of candidate patches per keypoint is a prespecified value. Additionally, each candidate patch and the corresponding model patch must satisfy a measure of matching whose value must be smaller than a given threshold \( \lambda_{th} \). Just as an example, assuming that the number of candidate patches per keypoint is \( M \) (all of them satisfying the required measure of matching), then for \( N \) keypoints there are \( MN \) potential solutions.

At the second stage, the proposed procedure for dealing with the joint task of locating the object and updating its representation consists in evolving one EvSOM per frame. For the initial frame (i.e., frame 0), the keypoints of the manually selected initial template defines the coordinates of the nodes of the EvSOM for the frame 0. From frame 1 onwards, we initialize the coordinates of the nodes of the EvSOM for the frame \( i + 1 \) with the coordinates of the nodes of the EvSOM for the frame \( i \). At each frame, the complete set of candidate patches defines a discrete search space within which the best solution is searched for.

By evolving the EvSOM for the frame \( i \) we mean finding, using an evolutionary algorithm, the optimum weight vector of the \( j \)-th node that encodes the parameters of the affine transformation that maps the coordinates of that node from frame \( i \) to frame \( i + 1 \). It is worth mentioning that learning the mappings from the keypoints of frame \( i \) to the keypoints of frame \( i + 1 \) is equivalent to locating (or tracking) the moving object.

In order to evaluate the degree of similarity between regions in two images, a measure of matching between the reference patch and a candidate patch is required by the fitness function. In principle, this measure could be any (dis-)similarity measure between two images, such as the root-mean-square error [21], the mutual information [11], the squared correlation coefficient [41] or the sum of the absolute differences [6]. In this paper, we use a SSD-related index, defined as

\[
SSD = \sum_k \sum_f (T_{i+1}(k, j) - T_i(k, j))^2, \tag{11}
\]

where \( T_{i+1}(k, j) \) and \( T_i(k, j) \) are, respectively, the intensities of gray levels in the target and current templates. By introducing the SSD index into the EvSOM fitness function, we get

\[
Fitness(W) = \alpha \cdot PCC(W) - \beta \cdot SSD(W). \tag{12}
\]

In a sum, the goal of the EvSOM-based tracking algorithm is to determine iteratively the candidate template which represents better the evolution of the object from the previous frame to the current frame. The mapping is determined by jointly optimizing the SSD index and the PCC index. In this paper, the PCC index is a measure...
of correlation for the distances among neighborhood interest points of the two images. The pseudo-code of the proposed EvSOM-based tracking algorithm is given below. In this algorithm the parameters $FIT_{best}$, $FIT_{max}$ and $G_{max}$ denote, respectively, the best fitness value for the current generation, the maximum fitness value found until the current generation and the maximum number of generations.

At this point, four comments are necessary. To begin with, a major feature of the proposed algorithm is that it takes into account in a very natural, inherent way the topological constraints of the template used for tracking the object of interest. This is not easily done by traditional tracking methods, as mentioned in the introduction. It becomes natural for the proposed algorithm because it is based on a topology-preserving self-organizing neural network. These topological constraints are taken into account via the PCC index included in the fitness function shown in Eq. (11). If such topological constraints are not present, the algorithm does not work suitably. This can be easily verified if one removes the PCC index from the fitness function and tries to optimize the SSD index solely.

As we emphasized in the 3rd paragraph of the Section 2.2, the grid of nodes comprising the EvSOM-based tracking algorithm is a non-regular one, unlike e.g. the regular grid of Kohonen map, since the distances from one node to the others need not to be equal. However, the links between the nodes do define their neighborhood (i.e. topological) relationships, and must be preserved, i.e. maintained as the tracking proceeds. In this respect the proposed algorithm differs considerably from standard self-organizing neural network structures, such as the Kohonen map [18] or the Topology-Representing network (TRN) [22], either by the use of a non-regular grid of nodes, either by establishing neighborhood links in the output space (not in the data input space, as the TRN).

Another important point in object tracking is the template update problem. The template drift due to accumulation of some sort of memory about the object in the template update rule. In [24] the proposed strategy is to update the template in the frame $i+1$ by solving Eq. (2) using gradient descent minimization starting from the accumulated value of the parameter vector $p$ at the frame $i$ and then using a heuristic rule to decide for the use or not of the outcome. In this paper we solve Eq. (2), for each $p_j$, through the hill-climbing search with a fixed number of iterations $N_{hc}$. The resulting template is then considered as the updated template.

Finally, since the components of the weight vector $w_j = p_j$, $j = 1, \ldots, N$, are real numbers, the projection of the keypoints of frame $i$ onto frame $i+1$ leads to new keypoints with real-valued coordinates. Since the coordinates of the keypoints in the image can assume only integer values, we have to quantize and interpolate the values of the coordinates of the projected keypoints.

Algorithm 1 EvSOM-based tracking algorithm
1. Set $i = 0$. Then, manually extract a template with $N$ keypoints and $L$ links. This is the updated template for the frame $0$.
2. For all frame $i + 1$ do
3. Set the number of EvSOM nodes equal to the number of keypoints of the updated template in frame $i$, with the coordinates of the keypoints assigned as the coordinates $r_j$ of the nodes in the output grid, following the topological constraints established by the distance links. Then, set the EvSOM weight vectors to $w_j = p_j = (0, 0, 0, 1, j = 1, 2, \ldots, N)$.
4. Perform a random search in the neighborhood of the $j$-th node, in order to find a set $C_j$ containing at most $M_j$ candidate patches which must satisfy $SSD_j \leq \Delta h_0$, for $j = 1, \ldots, N$. The neighborhood of the $j$-th node is defined as $\Delta p_j = (\Delta b^{(1)}, \Delta b^{(0)}, \Delta x^{(1)}, \Delta x^{(0)})$. For the $k$-th candidate patch associated with the $j$-th node, store its transformation vector $p_j^k$ and its $SSD_j^k$ value, for $k = 1, \ldots, M_j$ and $j = 1, \ldots, N$.
5. Build the initial population of candidate templates by randomly taking a candidate patch from each set $C_j$, $j = 1, \ldots, N$, and assessing its fitness.
6. While $FIT_{best} \leq FIT_{max}$ and generation $\leq G_{max}$ do
7. Generate the offspring and compute the fitness values;
8. Build the next population and assess its fitness;
9. End while
10. To avoid template drift, update each $p_j$, $j = 1, \ldots, N$, by solving Eq. (2) through the hill-climbing algorithm.
11. Compute the transformations $w(r, p_j) = w(r, b^{(1)}, b^{(0)}, \theta^{(1)}, \theta^{(0)})$, compute the resulting RMSE and present the solution.
12. End for
3. Results and discussion

In this section, we report experiments with synthetic and real-world films in order to evaluate the performance of the proposed EvSOM-based approach in object tracking. We compare the performance of the proposed approach with the direct tracking method described in [31, 32]. For all the experiments to be described in the next sections, the fitness function parameters were set to $\alpha = \beta = 1$, since the PCC and SSD indexes are rescaled to the range $[0, 1]$. For the computation of RMSE values, the ground-truth for reference trajectories of the object of interest were manually established. All the reported experiments were developed in Matlab, version R2009a, running under Microsoft Windows Vista, in a desktop PC with an Intel Core 2 Duo processor, clock of 1.8 GHz and 4 GB RAM.

3.1. Experiments with synthetic films

In the first set of experiments, as a proof of concept, we use a synthetic film, from now on called clip 1, that lasts 15 seconds, with a rate of 30 frames per second, each frame with size $256 \times 256$ pixels. The algorithm must track a geometric kite-shaped template (see Fig. 1), which moves smoothly along two arcs of sine with different periods. The parameters of the proposed algorithm are the following: $\Delta p = [\{-10, 10\}, \{-10, 10\}, \{-5, 5\}, [0.98, 1.02\}, N = 5, M = 32$ and $\lambda_{th} = 0.05$. Model and candidate patches are of size $15 \times 15$ pixels for all the experiments with this video clip.

Figure 4 shows superimposed snapshots of the tracking during the motion of the object. The rectangle in dashed line in each snapshot represents the decision taken by the algorithm on the location of the object. Figure 5 shows the actual path of the centroid of the object (‘dots’) during its motion and the trajectory (‘+’ signs) estimated by the proposed tracking algorithm. For both figures, the object follows an arc sine trajectory $(b_x, b_y)$ and rotates at a constant rate of two degrees per frame ($\theta$) on a gray-scale irregular background.

Figure 6 shows the performance of the proposed approach in RMSE values for the clip 1, compared to the performance of a direct tracking method. The RMSE value is computed between the true and estimated key-points for the whole sequence of frames. One can note that the performance of the proposed approach is consistently better than that of the direct method throughout the film. The proposed method achieved an average RMSE value of 1.74 with standard deviation of 1.35, while the direct method achieved an average RMSE value of 2.94 with a standard deviation of 2.12.

3.2. Partial and self-occlusion and outliers

Additionally, we performed an empirical assessment of the performance of the proposed approach in situations of partial occlusion, partial self-occlusion and in the presence of outliers. Partial occlusion (Fig. 2b)
occurs when one part of the template is hidden in the
image by an obstacle. Partial self-occlusion (Fig. 2c)
occurs when the template itself ‘hides’ one or more of
its keypoints during the movement. By outliers (Fig. 2d)
we mean candidate patches with appearance similar to
the model patch which are generated by the background
in the vicinity of the search region. Outliers cause the
tracking algorithms to make wrong decisions. The most
serious consequence of long occlusions and repeated
errors due to outliers is the accumulated deviation from
the correct trajectory which may eventually lead to the
loss of the tracking ability of the algorithm.

The experiment to evaluate the resilience of the pro-
posed tracking algorithm to partial occlusion consisted
in crossing a black spot on the trajectory of the template
so that 40% of keypoints (i.e. 2 keypoints) remained
hidden for a period of time corresponding to 10 frames.
The experiment on partial self-occlusion involved the
rotation of the template on itself to hide up to 40% of
keypoints. In the third experiment outliers were intro-
duced artificially in the background of the search region
for candidate patches. The outliers were reproductions
of each model patch in the search space for candi-
date patches. This type of test is conservative because
it ensures that the individual outlier patch will have
the best matching possible. The effect of outliers is
observed as a distortion in the template grid and increas-
ing the deviation of the path. The robustness of the

Fig. 6. Evolution of the RMSE values between true and estimated keypoints for each frame of the clip 1.

Fig. 7. Recovery of the trajectory path by the tracking algorithm in the presence of outliers for 20 frames.

Fig. 8. Example in which the proposed algorithm loses its tracking ability when the speed of the object is increased 5 times by sampling the clip every 5 frames.
proposed approach is verified when it is able to re-track the trajectory when the object leaves the region of ‘uncertainty’ resulting from the presence of outliers. Uniformly distributed random noise of up to 6% of the magnitude of gray level applied to 5% of the pixels of the image did not affect the results presented here.

The maximum RMSE values resulting from the experiments were 27.5% (partial occlusion experiment), 39.1% (self-occlusion experiment) and 21.4% (outliers experiment). To quantitatively measure the robustness to outliers we also use the rate of false positives (FP), which measures the percentage of all true
keypoints that were erroneously taken as outliers. For this experiment, we got

\[ \frac{8 \text{ outliers detected}}{5 \text{ keypoints per frame} \times 10 \text{ frames}} = 16\% \] (13)

The percentage of outliers present in the data was 20%. Figure 7 illustrates the effect and the resilience of the algorithm to outliers.

Figure 8 shows an example in which the algorithm loses its tracking ability. In this case, the speed of the object was increased five times with respect to the value used in the previous experiments. The other parameters remained the same. The loss of tracking ability occurs primarily because when the velocity increases the keypoint begins to approach the border of the neighborhood search for candidate patches. The tracking performance can be improved by increasing the neighborhood search. However, the immediate consequence is an increase in the computational demands of the algorithm.

3.3. Experiments with real-world films

In order to evaluate the efficiency of the proposed algorithm, for the second set of experiments we used two real-world films, named clip 2 (200 frames) and clip 3 (362 frames). Each frame in these films has 480 × 640 pixels (clip 2) and 512 × 512 pixels (clip 3).

![Fig. 11. Evolution of the RMSE values between true and estimated keypoints for each frame of the studied clips. Upper figure: clip 2, lower figure: clip 3.](image-url)
respectively. The parameters of the proposed algorithm are the following: \( \Delta \propto \{[-15, 15], [-10, 10], [-5, 5], [0.98, 1.02]\} \), \( N = 5 \), \( M = 32 \) and \( \lambda_\theta = 0.15 \) (clip 2) or \( \lambda_\theta = 0.20 \) (clip 3). Model and candidate patches are of size \( 21 \times 21 \) pixels for all the experiments reported in this section. Figure 9 shows the initial template for the clips 2 and 3. Clip 2 was produced by the authors of this paper and it is available upon request, while clip 3 is publicly available\(^1\).

Figure 10 shows a sequence of four frames of the clips 2 and 3, within which a rectangle delimits the region corresponding to the decision of the algorithm on the location of the objects (a man and a cork). For the clip 2, it is worth mentioning that the object to be tracked (i.e., the man) is not a rigid object, since the head, hands and legs move considerably from frame to frame. Thus, the selection of the initial template is essential for a good performance of the tracking algorithm. For this case, the keypoints of the template are selected in the the torso area. Furthermore, the background of the clip 2 is very noisy, adding additional difficulties to the choice of the optimal solution at each stage of the tracking algorithm if no topological constraints are taken into consideration. For the clip 3, it is worth emphasizing that the object of interest experiences changes in the illumination level along its motion. In spite of this changes in illumination, the EvSOM-based algorithm is able to track the object successfully.

Figure 11 shows the performances of the proposed approach in RMSE values regarding clips 2 and 3, compared to the performance of a direct tracking method. One can note that for both clips the proposed approach consistently outperformed the direct method in average. For the clip 2, the proposed method achieved an average RMSE value of 1.61 (3.92) with standard deviation of 1.01 (3.11), while the direct method achieved an average RMSE value of 2.56 (4.54) with a standard deviation of 1.88 (3.63). A conclusion that can be drawn from these results is that structural information, present in the EvSOM-based approach but not in the direct tracking method, indeed improves the tracking performance.

As a final remark, the processing times for one run of the proposed algorithm, implemented in a non-optimized Matlab code, are 13 ms (clip 1), 27 ms (clip 2) and 46 ms (clip 3), excluding the loading time of the images. Considering that the proposed approach involves the training of a grid-like self-organizing neural network through an evolutionary algorithm these times are quite remarkable. This can be explained by the fact that, at each run of the algorithm, the initial solution for the template position is already close to the final solution since the frame rate is much higher than the object speed. With a suitable code optimization those processing times can be diminished further.

4. Conclusions

An extension of the Evolutionary Self-Organizing Map algorithm (EvSOM) [3, 5] was developed and applied to object tracking. The main characteristic of the proposed approach is the inclusion of geometric or topological constraints in the determination of parameters of affine transformations that map template keypoints from one frame to the next one. Simulation results using synthetic and real-world films have shown that the proposed approach consistently outperformed a direct tracking method, even in situations that includes tracking of non-rigid objects and changes in the illumination level of the object of interest. Qualitative evaluation in situations of partial occlusion, partial self-occlusion and when subject to outliers also presented satisfactory performance.

References


\(^1\) Website: http://esm.gforge.inria.fr/ESMdownloads.html