# A Backmapping Approach for Graph-based Object Tracking

Thiago Meireles Paixão Ana Beatriz V. Graciano Roberto M. Cesar Jr.
Roberto Hirata Jr.
Institute of Mathematics and Statistics - University of São Paulo
Rua do Matão, 1010, 05508-090, São Paulo, Brazil
{thiagopx,abgv,cesar,hirata}@ime.usp.br

## Abstract

*Model-based methods play a central role to solve different problems in computer vision. A particular important class of such methods rely on graph models where an object is decomposed into a number of parts, each one being represented by a graph vertex. A graph model-based tracking algorithm has been recently introduced in which a model is generated for a given frame (reference frame) and used to track a target object in the subsequent ones. Because the view of an object changes along the video sequence, the solution updated the model using affine transformations. This paper proposes a different approach and improves the previous one in several ways. Firstly, instead of updating the model, each analyzed frame is backmapped to the model space, thus providing more robustness to the method because model parameters do not have to be modified. A different method for model generation based on user traces has also been implemented and used. This model generation approach is much simpler and user-friendly. Finally, a graph-matching algorithm that has been recently proposed is used for object tracking. This new algorithm is more efficient and leads to better matching results. Experimental results using synthetic and real sequences from the CAVIAR project are shown and discussed.*

## 1 Introduction

Structural pattern recognition is based on the concept that each pattern is composed by a certain number of parts and the relations between them. For instance, a house may be decomposed as a roof on top of walls. A wall may have a door in it, which in its turn has a window beside it. The house parts are: roof, walls, door and window. The relations are *on top, in, beside*. There has been intense research on structural pattern recognition methods, particularly for computer vision applications [1, 4]. The present paper addresses the problem of tracking an object in a video sequence while recognizing its parts in each frame. A structural pattern recognition approach based on inexact graph matching is adopted.

There is a large literature on parts detection and recognition using graphs [3, 5]. The method discussed herein is based on a recent object tracking scheme [6] which, in turn, derives from the inexact matching method described in [2]. In this approach, the object of interest is decomposed into parts and represented as a graph. Each part of the model is associated to a graph vertex. Spatial relations are measured from the parts of the image and represented as graph edges linking the corresponding vertices. A similar graph is extracted from the image where the object parts should be segmented and recognized. These tasks are carried out by matching the vertices from the model to the input graph.

The present paper starts from the method described in [6] and improves it in a number of different ways. The tracking and recognition results are better and obtained in a much more efficient way. Model generation is carried out from imprecise traces made by a user through a specially-designed GUI. The model update strategy has been replaced by a model-space oriented concept: the frame to be recognized is backmapped to the model space, which leads to more robust results since the model parameters do not have to be re-estimated for each frame. A new matching algorithm, recently introduced in [7] for static image segmentation, has been adapted and employed, leading to better and faster results. Finally, different pre-processing steps based on motion estimation and connected component analysis have been created. These aspects constitute the main original contributions of the present paper.

This paper is organized as follows. Section 2 reviews the main concepts of the approach, thus providing an overview of the method. The main contributions of the current paper are presented in Section 3, which is followed by the experimental results in Section 4. The paper is concluded in Section 5.

## 2 Graph-based methodology overview

This section reviews the mathematical modeling used in [6] for object tracking. The basic idea behind the method is to identify the target object in a given frame (reference frame) and then to track it by recognizing and mapping its parts in the subsequent frames. Therefore, object identification in the reference frame is equivalent to identifying its parts. This task may be carried out interactively (user-oriented) or by means of automatic detection procedures (e.g. face detection methods for face tracking and recognition applications). This step is known as *model generation* and is addressed in Section 3.1. Both object model and subsequent frames are similarly represented by attributed relational graphs (ARG) [9]. An ARG is actually a graph in which attribute vectors are assigned to vertices and to edges. Such vectors are responsible for adding relevant problem information to a graph data structure, since they hold symbolic properties and features related to the nodes and edges they are assigned to. We will henceforth refer to the *model graph* simply as *model*. On the other hand, the *input graph* is associated to the target frame where the object of interest should be tracked / recognized. Figure 1 provides an overview of the method. A model $G_m$ is generated in the reference frame ($F_0$). Subsequent frames $F_1, F_2, \ldots, F_n$ are analyzed so that input graphs $G_i^1, G_i^2, \ldots, G_i^n$ are also generated. Model and input graphs are matched, thus resulting in the tracked objects with recognized parts (indicated as $R_i$ in the figure).

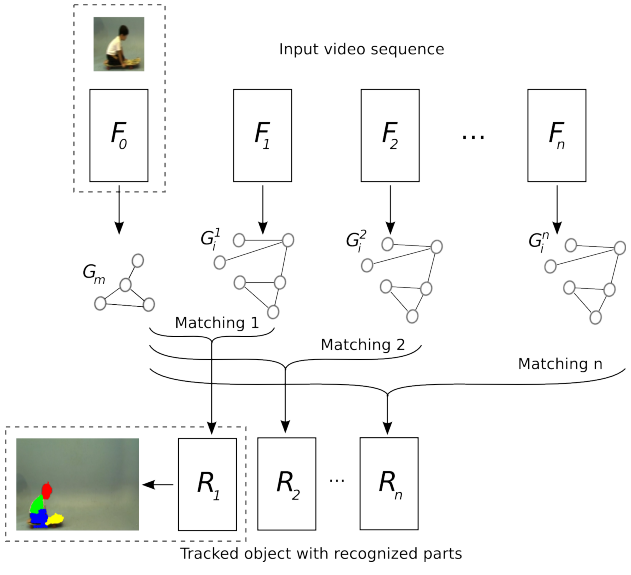



**Figure 1. Overview of tracking process.**

More formally, an ARG is a graph with feature vectors defined by a tuple $G = (V, E, \mu, \nu)$, where $V$ denotes the vertices and $E$ the edges. $V$ is associated to object parts, whereas edges represent spatial relations between them. $\forall v \in V$, there is an associated *object attribute vector* $\mu : V \rightarrow \mathbb{R}^p$. Analogously, $\forall e \in E$, there is an associated *relational attribute vector* $\nu : E \rightarrow \mathbb{R}^q$. The values $p$ and $q$ indicate the number of vertex and edge attributes, respectively. Adjacency graphs [8] have been used to define the graph topology.

Let $G = (V, E, \mu, \nu)$ be an ARG, $v, w \in V$ be two vertices and $a = (v, w) \in E$ and $a' = (w, v) \in E$ be two edges. The *object* attribute vector $\mu(v)$ is defined as:

$$\mu(v) = (RGB(v), c(v)). \tag{1}$$

The term $RGB(v)$ is a tuple $(r(v), g(v), b(v))$ of the normalized average RGB-levels of the image region associated to vertex $v$, whereas $c(v)$ indicates the centroid coordinates of the region. The *relational* attribute vector is defined as:

$$\nu(v, w) = (\overrightarrow{v}), \text{ with } \overrightarrow{v} = \frac{\overrightarrow{(c(w) - c(v))}}{d_{max}}. \tag{2}$$

where $d_{max}$ is the maximum distance between two points in the image.

The input ARG $G_i$ is generated from the watershed partition [10] of an input frame. Each watershed region is associated to an input ARG vertex and its object attributes vector is calculated from the corresponding watershed region (i.e. average RGB tuple and centroid). The watershed partition also helps to define the ARG topology, which is created as an adjacency graph based on the watershed basins neighborhood. It is also necessary to calculate the model $G_m$, which is explained in Section 3.1.

A solution to the tracking/recognition problem is a mapping from $V_i$ to $V_m$ [6]. Because $|V_i|$ is usually much larger than $|V_m|$, a suitable homomorphism between $G_i$ and $G_m$ should map distinct vertices of $G_i$ onto a single vertex of $G_m$, which corresponds to merging coherent input regions. An *association graph* $\tilde{G}_A$ between $G_i$ and $G_m$ is defined as the complete graph $\tilde{G}_A = (V_A, E_A)$, where $V_A = V_i \times V_m$ and $E_A = E_i \times E_m$. Thus, $\tilde{G}_A$ is a graph representation of all possible mappings from $G_i$ to $G_m$. Particularly, homomorphisms between $G_i$ and $G_m$ are a family of such possible mappings. A graph homomorphism $h$ between $G_i$ and $G_m$ is a mapping $h\colon V_i \rightarrow V_m$ such that $\forall a_1 \in V_i, \forall b_1 \in V_i, (a_1, b_1) \in E \Rightarrow (h(a_1), h(b_1)) \in E_m$. This definition assumes that all vertices in $G_i$ should be mapped to $G_m$.

As proposed in [2], a solution for finding a homomorphism between $G_i$ and $G_m$ may be defined as a complete subgraph $\tilde{G}_S = (V_S, E_S)$ from the association graph $\tilde{G}_A$, in which $V_S = \{(a_1, a_2), a_1 \in V_i, a_2 \in V_m\}$ such that $\forall a_1 \in V_i, \exists a_2 \in V_m, (a_1, a_2) \in V_S$, and $\forall (a_1, a_2) \in$

$V_S, \forall(a_1', a_2') \in V_S, a_1 = a_1' \Rightarrow a_2 = a_2'$, assuring that each vertex from the input ARG corresponds to exactly one vertex of the model ARG and $|V_S| = |V_i|$. Such a solution only considers the structures of $G_i$ and $G_m$, giving rise to many possible homomorphisms between both graphs. In order to find a suitable solution, a cost function is defined, so that the search may be expressed as an optimization problem.

Consider again $G_i$ and $G_m$, as well as vertices $a_1, b_1 \in V_i$, $a_2, b_2 \in V_m$ and edges $e_1 \in E_i$, and $e_2 \in E_m$. Let $\tilde{G}_S$ be a suitable subgraph of the association graph $\tilde{G}_A$ that represents a valid solution. The adopted cost function is defined as:

$$f(\tilde{G}_S) = \frac{\alpha}{|V_S|} \sum_{(a_1,a_2)\in V_S} c_V(a_1, a_2) + \frac{(1-\alpha)}{|E_S|} \sum_{e \in E_S} c_E(e). \quad (3)$$

$f$ is a weighted sum of object ($c_V$) and structural ($c_E$) properties. The cost functions $c_V$ and $c_E$, defined in [7] and adopted in this paper, consist of dissimilarity measures between vertices and edges, respectively. Thus, compatible pairs of vertices or edges present small dissimilarity value and contribute to minimizing $f$.

## 3 Proposed approach

This section presents the improved methodology to object tracking using graph matching. We first present the new tool for model generation, and the strategy to facilitate the segmentation algorithm. We propose a way to approximate the object for detection and then a new paradigm that maps the input graph back to the model space.

### 3.1 Model generation

The model is an ARG that represents object parts and background of an object neighborhood. In Section 2, we described the ARG generation process. Model generation is a particular case of ARG generation. In [7], the model generation process is presented to segment static images. We adapted this procedure by introducing background labeling, which is fundamental for object tracking. Basically, a model is generated by an interactive process in which the user makes some traces associating object parts to specific labels. If a labeled trace intersects a watershed region, a correspondent new labeled vertex is added to the ARG. Figure 2 depicts this step.

A specific tool including a GUI (Figure 3) to control this process has been developed using the platform C++/GTKmm [1]. This tool is integrated with Intel OpenCV

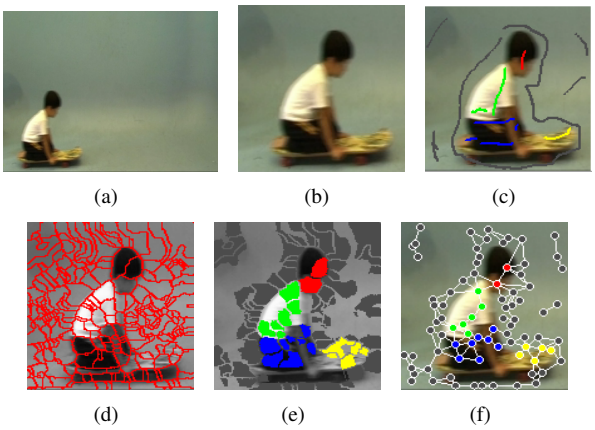



**Figure 2. Model generation steps: (a) reference frame; (b) region of interest; (c) user-defined traces; (d) watershed partition; (e) intersection of watershed regions with user-defined traces; (f) model superposed to original image.**

library [2], which implements the tracking methodology.

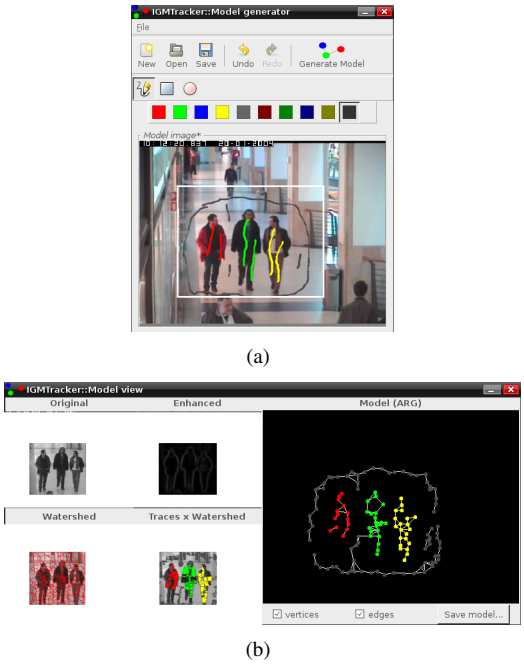



**Figure 3. Model generator: (a) ROI and traces defined by user; (b) resulting model.**

[1] http://www.gtkmm.org

[2] http://sourceforge.net/projects/opencvlibrary

### 3.2 Pre-processing

Yilmaz et al. [11] list four ways to detect objects: point detectors, background subtraction, segmentation and supervised learning methods. We based our method in a restricted area of segmentation, i.e. segmentation is performed inside a region of interest (ROI) for the sake of computational savings and object segmentation quality. The pre-processing step aims at obtaining a dilated object silhouette as a ROI and control points to calculate object motion.

Initially, the user defines a rectangular ROI, which may not represent the object shape properly. Therefore, we compute an input ARG from the original ROI and segment this region by matching against the model. The result includes classified object parts as well as the classified background region. The object silhouette is the union of object parts whereas the strategic control points are the centroids of parts. This process is illustrated in Figure 4.

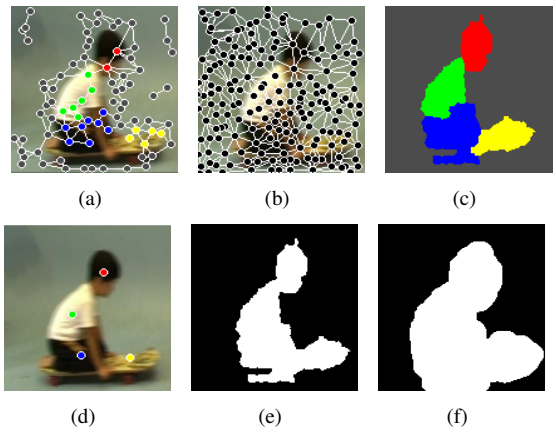



**Figure 4. Pre-processing steps: (a) model superposed to rectangular ROI; (b) input graph computed inside a rectangular ROI; (c) segmentation; (d) centroids of object parts for motion estimation purposes; (e) object silhouette; (f) dilated object silhouette.**

### 3.3 Approximate object detection

Before the final object detection through model-based segmentation, we have proposed an *approximate object detection* pre-stage to simplify the input graph extraction and the graph-matching computing. We estimate an affine transform $T_i$ that maps an object of an input frame $i$ onto the object in reference frame. $T_i^{-1}$ denotes the inverse of $T_i$. Let $S$ be a binary image of the dilated object silhouette obtained in the pre-processing step. Therefore, $T_i^{-1}(S)$ denotes the region of interest of the input frame $i$. In fact, the transformation $T_i$ can only be estimated after segmentation of frame $i$ because the segmentation provides control points that match those in pre-processing step. These control points allow us to calculate a proper affine transformation (see Section 3.4).

We used a dilated version of the silhouette because even if $T_i$ is not accurate, $T_i^{-1}(S)$ still will include the whole object. At this point, we emphasize that the object is not fully detected, because the region may include non-object parts that need to be classified. $T_i(S)$ plays the role of a mask for input graph generation in order to save computational processing. It helps to eliminate other moving objects that may lie on any part of the scene.

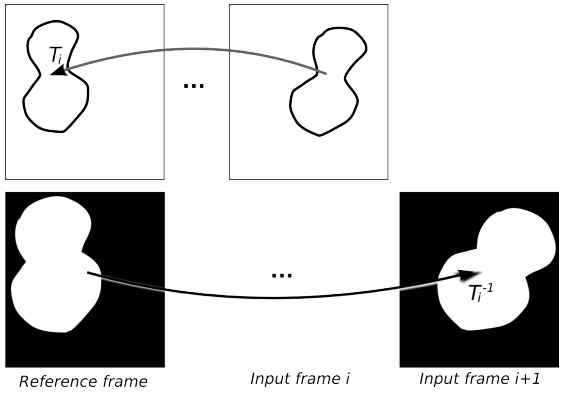


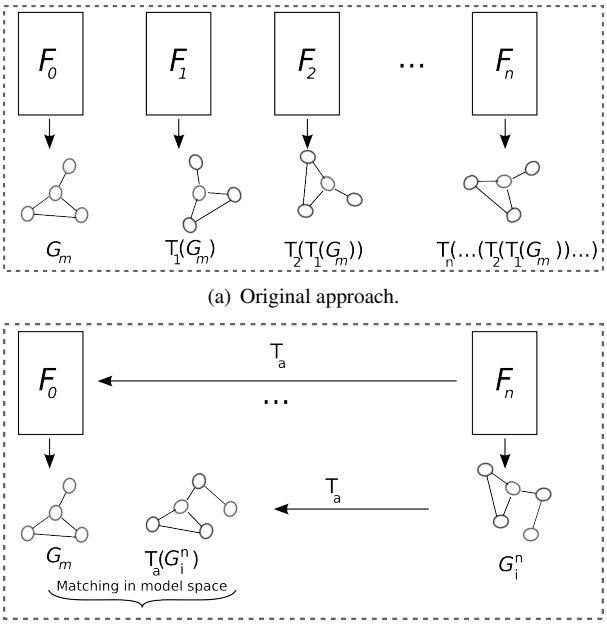

**Figure 5. Approximate object detection. In the first row, the transformation $T$ maps the object of input frame $i$ onto the object of the reference frame. In the second row, we use $T^{-1}$ to map the dilated object silhouette onto the approximate dilated silhouette of input frame $i+1$.**

### 3.4 Backmapping

The key idea behind our tracking methodology is to backmap an object onto the reference frame for segmentation. In this work, backmapping is defined as the task of estimating an affine transformation that maps an input graph back to the model space. This procedure and the segmentation step are strongly interdependent. During the second input frame processing, the respective input graph is not classified and is slightly misaligned with respect to the model. Thus, the segmentation step precedes the backmapping. After segmentation, the backmapping is estimated and applied to the input graph of the next frame. A well-performed backmapping depends on a well-performed segmentation and vice versa.

This approach differs from the original method which is based on *model update*, i.e. the model is mapped onto the

object of the current frame, whereas our approach holds the model fixed in order to prevent accumulation of errors in backmapping estimation and therefore model degeneration. Figure 6 confronts other two approaches.

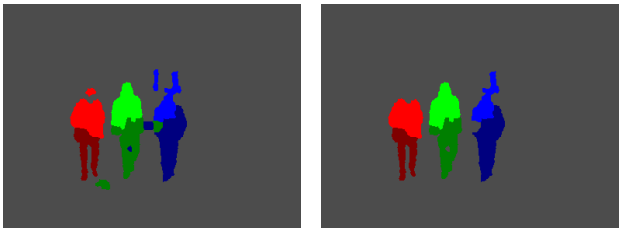



(a) Original approach.

(b) New approach.

**Figure 6. Original x new approach. In the original approach the model suffers sequential transformations that may degenerate the model. In the new approach the model is fixed and input graph is backmapped.**

We assume that an object part is a connected component. After the segmentation step, we filter the resulting image of selecting the largest connected component for each part (Figure 7).



(a) (b)

**Figure 7. Connected components heuristic: (a) raw segmentation; (b) segmentation after connected component filtering.**

The last step is to estimate the backmapping affine transformation. We take the centroids of each classified object part and we build a linear system to recover the transformation matrix. The affine transformation is given by six parameters, so the segmented input frame, and also the reference frame, must present at least three object parts.

## 4 Experimental results

The algorithms have been implemented using C++ and Intel OpenCV library. The timings reported here refer to a Celeron machine (1.73GHz) with 1GB of RAM. The dataset has three MPEG sequences ($320 \times 240$ pixels/frame) and frame rate of 30fps. Segmentation was performed with $\alpha = 0.4$ (weight for vertex cost, see Section 2) and $\delta = 0.6$ (weight for modulus edge cost [7]). Results present tracked sets with recognized parts and approximate object detection. The time consumption for the three cases is about 10 seconds for model generation, 13 seconds for the preprocessing step and 20 seconds per processed frame.

In the first sequence, a boy performs a translation from left to right of scene. The boy's body, skate board and some background have been labeled according to the Figure 8(a). The result (Figure 9) shows 5 sequence frames sampled in 5-spaced intervals, from a total of 36 frames. We achieved a good visual performance in this sequence, with just very small variations along the frames due to excessive blur and low resolution image.

The second sequence is from a rotation of a doll. The doll's body and a background region have been marked according to Figure 8(b). In this case, the watershed gives rise to only one background region. This is a problem because the whole background is represented by just a single vertex. If that vertex is misclassified, all the background is misclassified. Thus, new vertices are generated around object by splitting of background into a $p \times q$-square grid ($p = q = 25$ pixels in this sequence). The result (Figure 10) shows 5 sequence frames sampled in five 15-spaced intervals, from a total of 60 frames (rotation of 60 degrees). In column 4 of Figure 10(b), the doll's right arm (reference of the reader) was misclassified as background due to a post-processing filter. If on the one hand this post-processing helps to maintain a good visual performance, on the other, objects may disappear if they disconnect from the main part. In the next frame, the tracking algorithm recovers from the mistake.

The third sequence is from the CAVIAR project [3]. There are three people walking from top to bottom of the image causing a zoom effect. We modeled the three people as a unique object (Figure 8(c) depicts object and background modeling). This sequence includes challenging elements such as other moving people, shadows, heterogeneous background and poor object resolution in reference frame. Fig-

[3] EC Funded CAVIAR project/IST 2001 37540, found at URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

ure 11 shows some results. Our approximate object detection strategy eliminates non-target objects. In some frames, dark strips of the floor was merged to dark pants of the three targets, a limitation due to the way that the watershed algorithm is being used. The advantage of structural approach is that artifacts, like other people walking on scene, do not introduce errors in the tracking if they are not very close to the tracked object. On the fourth column of the second row, approximate object detection estimates a rotation that does not occur in video sequence. The cause is the loss of few object parts, therefore a bad estimation of motion control points (object parts centroids). Common effects of bad estimation is rotation and, in some cases (depending on object symmetry), scaling.

(a) (b) (c)

**Figure 8. Regions of interest and traces for three video sequences.**

## 5 Concluding remarks

Recent advances on video analysis show that Structural Pattern Recognition may be a promising way to approach the object tracking problem. A previous work models the objects to be tracked as a graph with attributes and, for each frame, the model is updated using affine transforms. In this paper, we proposed a new paradigm where the model is not updated for the subsequent frames but it is the object that is backmapped to the original model, simplifying the algorithm. The graph-matching algorithm involved in this process has also been changed to the one presented in [7]. In relation to the previous one, this one has decreased the runtime complexity from $O(|V_g|^{|V_i|})$ to $O(|V_g| * |V_i|)$, drastically dropping the amount of time taken to process a given video. In practice, when tested[4] against the goose sequence [6], it has taken 5 minutes (in comparison to 78 minutes from the previous work), with visually similar results. Finally, the initial model generation is done now with a user-friendly interface. The approach has been implemented, tested (to certify against implementation mistakes) and experimented in some video sequences, three of them were reported in this work (one of them is a surveillance video used to test motion segmentation algorithms). The results are very promising and show significant improvements in relation to the previous approach.

[4] experiment available on www.vision.ime.usp.br/ thiagopx/sibgrapi/

**Acknowledgements.**

The authors are grateful to FAPESP, CNPq and CAPES for partial (Roberto M. Cesar Jr. and Roberto Hirata Jr.) and full financial support (Thiago M. Paixão and Ana B. V. Graciano).

## References

[1] H. Bunke. Recent developments in graph matching. In *ICPR*, pages 2117–2124, 2000.

[2] R. Cesar-Jr, E. Bengoetxea, P. Larranaga, and I. Bloch. Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms. *Pattern Recognition*, 38(11):2099–2113, November 2005.

[3] Y. Chen, L. Zhu, C. Lin, A. Yuille, and H. Zhang. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 289–296. MIT Press, Cambridge, MA, 2008.

[4] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004.

[5] P. F. Felzenszwalb and J. D. Schwartz. Hierarchical matching of deformable shapes. In *CVPR*. IEEE Computer Society, 2007.

[6] A. B. V. Graciano, R. M. Cesar-Jr., and I. Bloch. Graph-based object tracking using structural pattern recognition. In *SIBGRAPI '07: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*, pages 179–186, Washington, DC, USA, 2007. IEEE Computer Society.

[7] A. Noma, A. B. V. Graciano, L. A. Consularo, R. M. Cesar-Jr, and I. Bloch. A new algorithm for interactive structural image segmentation. 2008.

[8] T. Pavlidis. Algorithms for shape analysis of contours and waveforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(4):301–312, 1980.

[9] R. J. Schalkoff. *Pattern recognition: statistical, structural and neural approaches*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[10] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 13(6):583–598, 1991.

[11] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13, 2006.

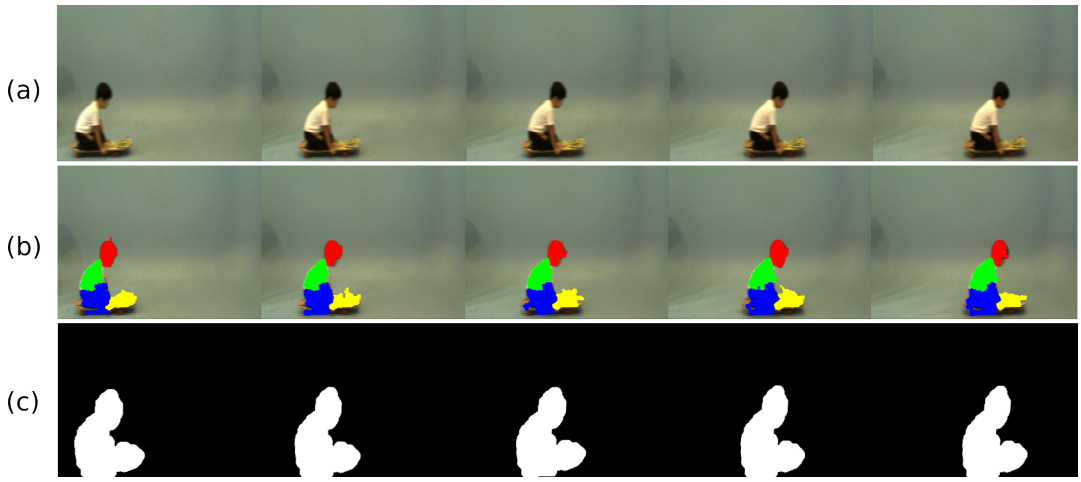



**Figure 9. Results of the first experiment.**

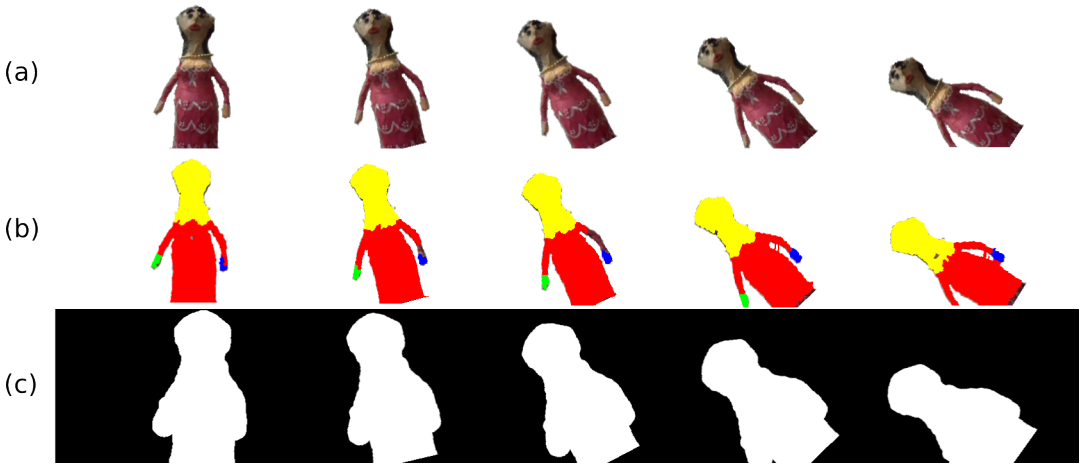



**Figure 10. Results of the second experiment.**

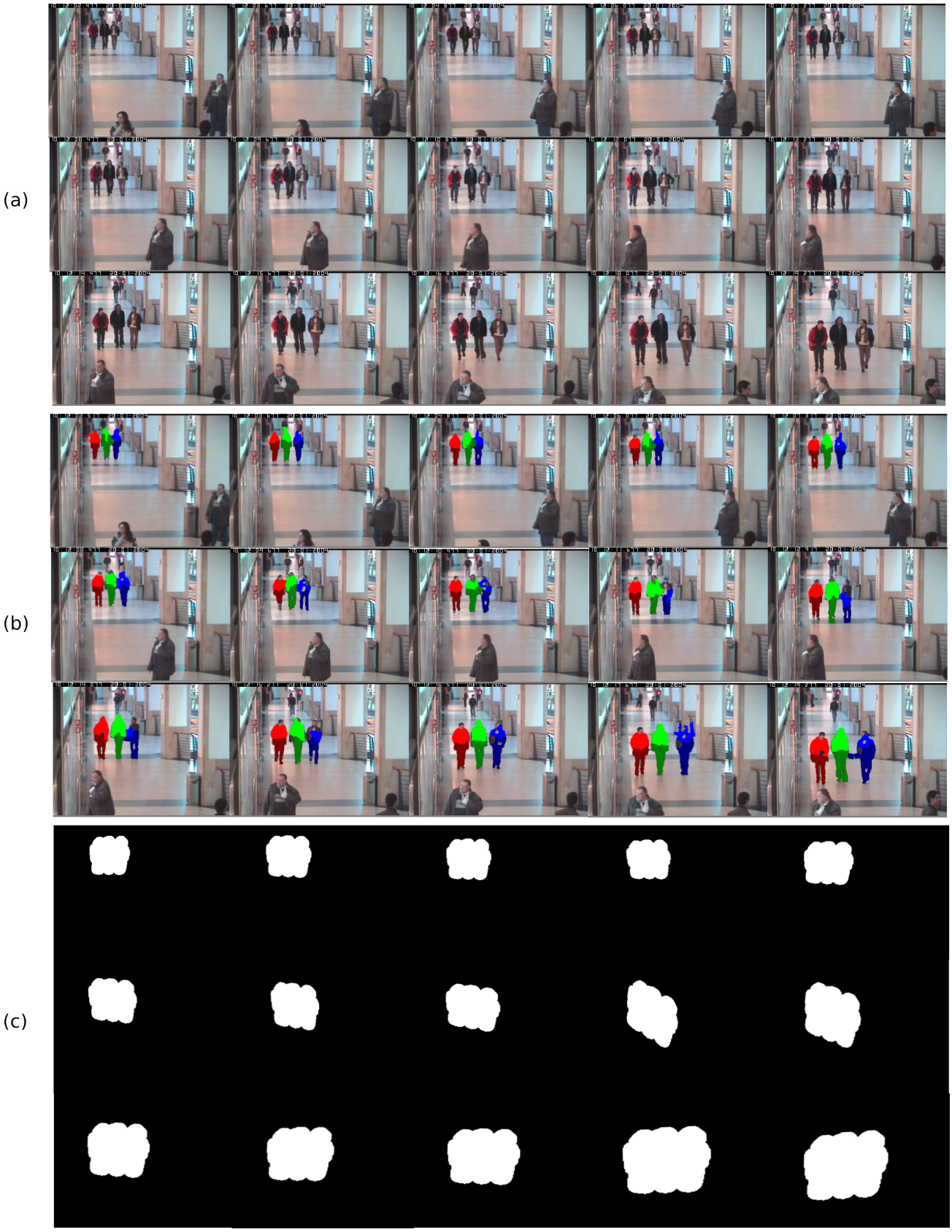



**Figure 11. Results of the third experiment.**