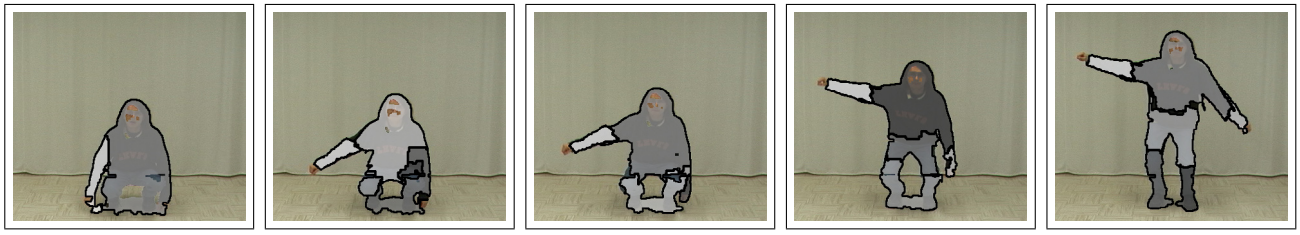
Incremental Discovery of Object Parts in Video Sequences

Stéphane Drouin, Patrick Hébert and Marc Parizeau
Computer Vision and Systems Laboratory, Department of Electrical and Computer Engineering
Laval University, Sainte-Foy, QC, Canada, G1K 7P4
{sdrouin, hebert, parizeau}@gel.ulaval.ca



(a) Frame #3 (b) Frame #11 (c) Frame #14 (d) Frame #22 (e) Frame #29

Figure 1. Discovered object parts at selected frames of video sequence C (see Figure 9 for open-loop segmentation).

Abstract

This paper addresses the fundamental problem of automatically discovering an unknown moving deformable object in a monocular video sequence. No prior model of the object is used; it is only assumed that the object is composed of a set of apparently rigid parts that are not necessarily visible simultaneously, making it possible to circumvent the typical constraint of model initialization. A set of rigid parts describing the object is incrementally extracted in a modeling-tracking loop with reinforced memory. In this framework, low-level segmentation is considered as a necessary but non reliable process that helps initiating hypotheses. Motion-based layer segmentation from feature points and edges is applied only when and where no modeled parts can be tracked. Using the quantity of motion measure, it is further shown how to deal with temporal scale. The interest for this approach in applications such as human tracking is demonstrated for a set of various sequences including a rapidly evolving shape.

1. Introduction

Tracking of deformable objects in video sequences is a challenging computer vision problem. Model-based approaches to this problem require the construction of a model and its initialization [8]. In practice, models are often handcrafted when object types are known a priori and initialization, when not performed manually, relies on strong hypotheses concerning the initial pose of the object. Once the model is initialized, tracking may be conducted in several ways [6, 12].

In this paper, we address the problem of discovering the parts of a target object without making any assumption on the topology of its model. The number of parts and their shapes are initially unknown. They are learned (discovered) in an incremental and continuous processing of video sequence images. The system is simply initialized with a single part corresponding to the initial image foreground. Parts are then extracted through a motion-based layer segmentation, but segmentation is only activated locally when current parts can no longer be tracked. Hence new sets of parts (models) are dynamically created and may coexist during tracking. While re-observed models are reinforced, those models that have not been tracked for a long period of time are eventually forgotten.

For example, Figure 1 illustrates an application of the approach on a rapidly evolving shape. A human is initially in a squat position before raising himself and waving his arms. The system starts with the foreground area where segmenting a human body using existing model-based approaches would be almost impossible. As soon as the first arm starts to rise, the system dynamically discovers a sec-

ond part. Then, when the whole body starts to move, the main part is reconsidered (re-segmented) several times to produce a final model in 6 parts at the end of the sequence.

Since no a priori model is assumed, the approach described hereafter is applicable to the motion analysis of any moving deformable object, including the motion of human bodies. It is assumed that deformable objects are composed of rigid parts whereby a deformation refers to a relative displacement between parts. This type of object encompasses but is not limited to articulated objects. For human motion, the idea is to stay away from strong hypotheses like standing pedestrians, and to allow the tracking of people carrying objects. Our main current restriction is that motion, to be trackable and stabilized, should be mostly 2D, that is the apparent shape (projected aspect in images) of rigid parts should not change too much during motion.

Related work includes offline approaches, where a video sequence is processed as a whole [5], as opposed to our online approach. In this case, an optimal model is extracted from the analysis of the complete set of images of a volume sequence. In [15], 2D rigid parts are learned sequentially from the video sequence, starting with the background, and optimized using EM in a layer-based framework. Tracking allows for 2D transformation of known parts and is used in order to accelerate the search of new parts. Another relevant approach is used in [10], where a model including articulations is first segmented into piecewise rigid segments and assembled, then tracked in subsequent images. However, all parts of the model must be visible and in motion in the keyframes selected for segmentation. Recent work [9] added the tracking of 3D models in 2D images, but the issue of selecting keyframes for segmentation is still open.

The main contribution of this paper is to use object modeling at the heart of the system (Figure 2), minimizing the impact of segmentation which is a complex operation that is not reliable in all situations. Anticipating that valid segmentations will be obtained in some situations, the system reinforces low-level segmented regions through high-level tracking and memory of models. Model memory is a second contribution of this paper that stabilizes the overall system by making it more robust to segmentation and tracking errors, as well to inevitable wrong decisions because of the many parameters which are mainly present in segmentation algorithms. This paper also contributes to motion-based segmentation in two ways: the measure of quantity of motion is used in order to deal with temporal scale and the combination of feature points and edge templates allows the segmentation of less textured images.

The paper is organized as follows. Section 2 describes the proposed approach. Details of the feedback loop that involves model management, tracking, and segmentation are included. Section 3 provides results and analysis for three different video sequences.

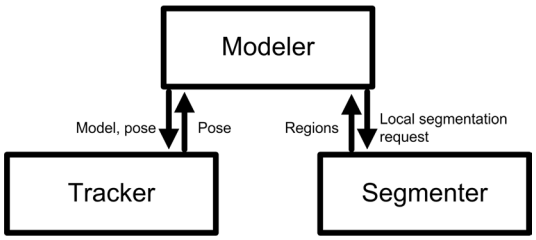


Figure 2. Data flow in the system.

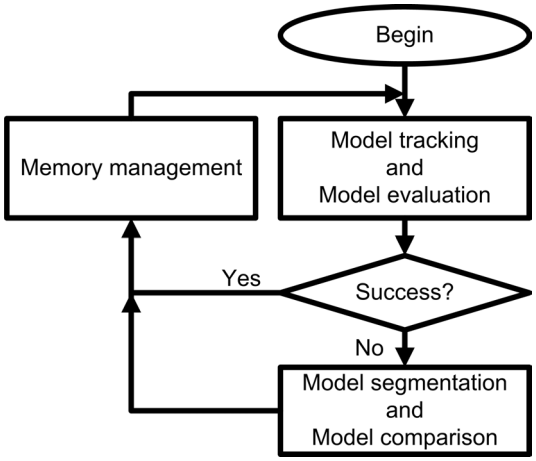


Figure 3. Modeler processing structure.

2. Modeling object parts

The *Modeler* is based on a constant feedback loop that gives priority to tracking over segmentation (Figure 3). At any given time, the *Modeler* tracks a number of model candidates M_j, each one being a set of rigid parts: $M_j = \{P_i^j\}$. Rigid parts are first segmented as regions R_i based on their apparent motion in the images, then assembled into models as parts. The system tracks the available models in the video sequence based on their appearance. Since the *Segmenter* and the *Tracker* produce occasional bad results, the *Modeler* processes models as hypotheses in an effort not to depend on a single segmentation or tracking result. Therefore, it is constantly assessing the quality of segmented regions and their subsequent tracking as models. It also detects the apparition of new moving regions in the sequence. By doing so, it acts as a buffer between the *Segmenter* and the *Tracker*, which do not communicate directly with each other (Figure 2). This constant feedback makes the *Modeler* robust to erroneous segmentations, departure in tracking and also to its own bad decisions. In order to favor the emergence of better models, the confidence of models that are successfully tracked is continuously reinforced while models that cannot be tracked are eventually forgotten.

The model hypotheses that are incrementally built by the system are composed of a variable number of rigid parts.

We shall discuss the case where a single viewpoint is available and where the 2D projection of the observed object is stable over time. A 2D model is composed of a set of support maps representing the geometry of its parts, a set of images representing their appearance and a set of 2D rigid transformations representing their time-variant pose. In the following, it is assumed that the foreground mask F_t of the image at time t is extracted in a pre-processing step.

2.1. The Modeler

The *Modeler* manages the feedback between the *Tracker* and the *Segmenter* by requesting results from these modules when necessary. Its decisions are based on its memory of model candidates.

Model tracking estimates the best pose of all of the model candidates when a new image of the sequence is fed to the system. Each model candidate, which is a possible explanation for the scene, is tracked independently by the *Tracker* in order to obtain its pose parameters and its appearance in the current image. The *Modeler* then proceeds to evaluate models based on the result of this operation.

Model evaluation has to determine if at least one model has been successfully tracked in the current image. Three complementary error measures are used: an SSD error for each part to evaluate if the modeled appearance matches with the image; a measure of the subsethood of each part with respect to the foreground mask, which determines if its support map is still within the moving portions of the image and a measure of the subsethood of the foreground mask with respect to each model to determine if all moving regions of the image are modeled:

$$E_{\text{SSD}}(P_i^j) = \frac{\sum_{p \in P_i^j} D(p)^T D(p)}{\text{area}(P_i^j)}, \quad (1)$$

$$E_{\text{part}}(P_i^j) = 1 - \frac{\text{area}(F_t \bigcap P_i^j)}{\text{area}(P_i^j)}, \quad (2)$$

$$E_{\text{model}}(M_j) = 1 - \frac{\text{area}(F_t \bigcap \cup P_i^j)}{\text{area}(F_t)}, \quad (3)$$

where p are image pixels and $D(p)$ is the vector difference of all color channels between the fitted part appearance and the underlying image pixels. Each error measure is compared to a threshold: the tracking of model M_j is successful if $E_{\text{SSD}}(P_i^j) < \tau_{\text{SSD}}$ and $E_{\text{part}}(P_i^j) < \tau_{\text{part}}, \forall P_i^j \in M_j$ and $E_{\text{model}}(M_j) < \tau_{\text{model}}$. If tracking is successful for more than one model, then the one with the most reinforced memory (see memory management below) is chosen to be the successfully tracked model for the current image.

Thresholds are adaptive and computed from past errors that were found to be acceptable. Given the history of a measure $H_x = \{E_x^k\}$ where $k < t$, which is the set of error measures $x \in \{\text{SSD}, \text{part}, \text{model}\}$ of successfully tracked models before time t, the threshold at time t is given by $\tau_x^t = \text{mean}(H_x) + 2\sigma_{H_x}$ where σ_{H_x} is the standard deviation of H_x. The system keeps a history of E_{SSD} and E_{part} for each rigid part and a global history of E_{model}. Every time a model is accepted, its error measures are added to their corresponding histories. Thresholds are initially set to infinity before a minimum number of τ_H observations have been available. In practice, τ_H is set to 2.

Model segmentation is necessary when the evaluation of all models have failed. However, the system first tries to salvage valid parts of existing models. A part is deemed valid if $E_{\text{SSD}}(P_i^j) < \tau_{\text{SSD}}$ and $E_{\text{part}}(P_i^j) < \tau_{\text{part}}$. The existing model with the most reinforced memory and with at least one such valid part is chosen as a start model M_s.

If a start model exists, only the portions of the image foreground mask (F_t) intersecting parts for which $E_{\text{SSD}}(P_i^s)$ or $E_{\text{part}}(P_i^s)$ is too large have to be resegmented, in addition to portions of F_t not intersecting any part if $E_{\text{model}}(M_s)$ is too large. The new model M_n is then assembled using the regions of the new segmentation and the valid parts of M_s. If no start model exists, the whole foreground needs to be resegmented and M_n is assembled with the regions of this new segmentation.

Model comparison is performed to decide if a new model M_n is in fact an update of an existing model M_j. It is an update if M_n and M_j have the same number of rigid parts and their matched parts overlap above a threshold. The matching of M_n and M_j optimizes the proportion of support map pixels in both models that belong to matched rigid parts:

$$O(M_n, M_j) = \frac{\sum_{p \in (M_n \cup M_j)} \delta(p)}{\text{area}(M_n \cup M_j)} \quad (4)$$

where $\delta(p)$ is set to 1 if the part of M_n overlaying p is matched to the part of M_j overlaying p and set to 0 otherwise. The overlap is large enough if it is above a threshold τ_{overlap} which is set to 80% in the experiments. When a new model is an update, the newly segmented parts are used to update their matched existing parts and the updated model is chosen as the best explanation for the current image. An existing part is updated by replacing its support map and appearance with resegmented values. If it is not an update, the new model is accepted as the best explanation for the current image.

Memory management ensures that not all model candidates are tracked indefinitely and avoid accumulating hypotheses. To this end, each model accumulates an amount of memory credits C_j. At the end of each cycle, the *Modeler* gives a memory credit to the model M_m that is the best explanation for the current image: $C_m \leftarrow C_m + 1$. One credit

is taken off all other models: $C_j \leftarrow C_j - 1, \forall j \neq m$. As soon as a model has no credit ($C_j = 0$), it is removed from the memory. Therefore, models that are often re-observed will remain active for a longer period of time even if they are sometimes not tracked successfully (because of occlusions or erroneous segmentation for example) and transition models which are not re-observed will be quickly forgotten by the system.

2.2. The Segmenter

Image segmentation based on object motion is carried out in the same spirit as in [10]. Special attention was given to make it less dependent on image texture and velocity of motion with respect to the frame rate. To handle less textured images, the *Segmenter* uses edge templates in addition to feature points. Edge templates do not need textured images, but need a large neighborhood in order to provide a reliable motion estimate based on the geometry of the scene. Feature points give a more local motion estimate, but rely on textured images and are also less stable. Both are used since local texture-less regions are frequent in real images.

Since segmentation is based on motion between images, the temporal scale must be carefully chosen so there is enough motion within the considered images. This is an important difficulty not addressed in other systems. The *Segmenter* uses the Quantity of Motion concept [4], which is a global measure of the amount of detected motion. It is defined on the foreground mask F_t of the image to segment at time t and that of its temporal neighbors F_k at times $t - N \leq k < t$:

$$\text{QoM}(N) = \frac{\text{area}(\bigcup_{k=t-N}^{t} F_k)}{\text{area}(F_t)} - 1. \quad (5)$$

The temporal scale N is chosen so that $\text{QoM}(N)$ is above a threshold that is fixed to $\tau_{QoM} = 10\%$ in the experiments. Therefore, the temporal neighborhood of the segmented image is not selected by absolute frame indexes, but rather by a selected QoM value.

Figure 4 outlines the layer-based segmentation algorithm. This algorithm aims at segmenting the regions of the image based on their motion. Motion cues are obtained by using a set of images $\{I_k\}$ in the temporal neighborhood of the currently segmented image I_t. Each pair (I_k, I_t) is first processed independently to segment selected features in I_t, then the results are combined to compute motion parameters of regions. These parameters are then used to segment I_t in a global optimization framework. All processing is done in a mask (region of interest) defined on I_t.

Feature segmentation extracts transformations describing the motion of regions between I_t and I_k. Since we are only interested in the transformations, any feature can be used.

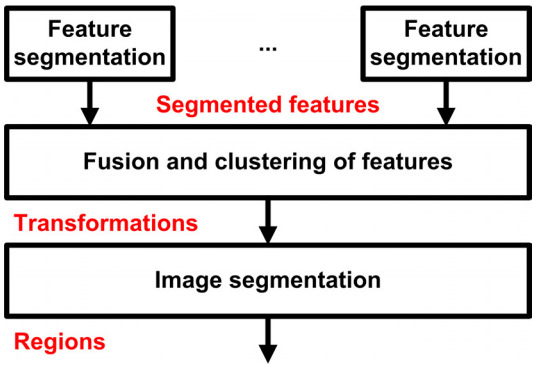


Figure 4. Segmenter processing structure.

Feature points and edge templates are segmented independently and then combined in a later stage.

Feature points [14] are selected in I_t and then a set of templates centered at feature points is used to match them in I_k [2]. The resulting motion vectors are clustered: motion fields are estimated as described in [16] except that we use a fuzzy partition of feature points instead of a crisp one.

Edge template masks are obtained with the Canny algorithm, both in I_t and I_k. The rigid transformation minimizing the error between the transformed edges of I_t and those of I_k is then obtained using an algorithm which is implemented as the well known Generalized Hough Transform [1]. The angular error ϵ between corresponding edges of the templates are calculated, then these error maps are thresholded. To automatically define a threshold, a subset S_ϵ which includes only good matches by hypothesis is defined on the smaller errors. In the experiments, we choose the lowest 40% errors to be part of S_ϵ. The threshold is given by the largest error in that subset plus one robust standard deviation, σ_{rS_ϵ}, computed in the subset:

$$\tau_{\text{edges}} = \max(S_\epsilon) + \sigma_{rS_\epsilon}. \quad (6)$$

The two thresholded error maps give the unmatched edges in I_t and I_k. These thresholded maps become the new edge masks and the algorithm is repeated. The iterations stop when there is not enough unmatched edge in the thresholded error maps (we set this minimum to 10 in the experiments).

Fusion and clustering of features is performed on results arising from the segmentation of all I_k and all feature types. Each segmentation corresponds to a set of transformations from I_t to a particular temporal neighbor. A membership value $\mu_{T_l^{I_k}}(f)$ to each possible transformation T_l in I_k is associated with each feature f. These values are collected in a vector $\mu(f)$ and all features are clustered to obtain the final segmentation of features that will combine feature types and temporal neighbors.

The membership value of a feature f to a transformation $T_l^{I_k}$ is a function of the error that stems from using that

transformation to move the feature from I_t to I_k. We use the sum of squared differences (SSD) over a neighborhood of the feature to estimate the error $E_{\text{rr}}(T_l^{I_k}, f)$. The membership value is a ratio between the error coming from all of the transformations $T_j^{I_k}$:

$$\mu_{T_l^{I_k}}(f) = \frac{\tilde{E}_{\text{rr}}(T_l^{I_k}, f)}{\sum_j \tilde{E}_{\text{rr}}(T_j^{I_k}, f)}, \quad (7)$$

$$\tilde{E}_{\text{rr}}(T_l^{I_k}, f) = \frac{\sum_j E_{\text{rr}}(T_j^{I_k}, f)}{\max(E_{\text{rr}}(T_l^{I_k}, f), \epsilon)}.$$

The membership vector of a feature is then the collection of all membership values associated with segmented transformation $T_j^{I_k}, \forall I_k$.

Features are clustered using a fuzzy *k*-means algorithm [7]. The initial prototype vectors are obtained through a binarization of the membership vectors: $\mu_{T_l^{I_k}}(f)$ is replaced by 1 in the binary vector if $\mu_{T_l^{I_k}}(f) = \max_j\{\mu_{T_j^{I_k}}(f)\}$ and by 0 otherwise, for each I_k. The occurrence of binary vectors are counted and the most common ones are used as initial prototype vectors. The clustering algorithm is run with different K numbers of prototype vectors. The Bayesian Information Criterion [13] selects the best K_{BIC} among the resulting clusterings, typically selecting values between 1 and 5 in the experiments. Given the partitions, a transformation $T_c^{I_k}$ from I_t to each I_k is computed for each of the K_{BIC} clusters (c) using features belonging to that cluster.

Image segmentation is performed by a Graph Cut algorithm [3] which optimizes class assignment given dynamic and static costs. A dynamic cost is attributed to all of the image pixels p located in the foreground F_t for all transformations $T_c^{I_k}$ (all clusters c). This dynamic cost is the SSD error between a neighborhood of p in I_t and a neighborhood of the corresponding pixel $p(T_c^{I_k})$ in I_k. The SSD error is computed for all images I_k. Errors are added, then normalized so that the total dynamic costs associated to p is 1. A static cost is also attributed to neighboring pixels that do not belong to the same class, this cost is also limited to 1.

The creation of a new model from the region segmentation consists in creating a rigid part for each region, which is used to define support maps in the segmented image. The appearance of each part is extracted from the image texture, and the initial pose parameters are set to identity.

2.3. The Tracker

Given a model and its parameters (pose, appearance, support map) up to time $t-1$, the *Tracker* estimates a new set of pose parameters such that the model optimally fits I_t. For each part of the model, the 3 parameters of a rigid 2D transformation (rotation and translation) are estimated so that the SSD between its appearance and I_t is minimized [11]. The rigid transformation is not a restriction, the *Tracker* could estimate a similarity or an affinity, but a rigid transformation works well for the kind of images that we use and the reduced set of parameters is easier to optimize. Unless a self-occlusion of the model is predicted, a penalty is added for overlapping parts. This penalty replaces the SSD error for pixels overlapping other parts. Ideally, the *Tracker* would optimize all free parameters of the model at once, but in order to lower the processing time, the parameters of the model are optimized sequentially: the largest parts of the model (measured by their area) are tracked first (this strategy is also used in [15]). Since perfect results are not expected from the *Tracker*, this shortcut should not be seen as a problem for the system. For each part P of the model, the final pose π_t is attained when the total error is minimized:

$$\pi_t = \arg\min_{\pi} \sum_{p \in P(\pi)} (\text{SSD}_e(p) + O_e(p)) \quad (8)$$

where $P(\pi)$ is the part with pose π, p are the pixels in the support map of $P(\pi)$, and $\text{SSD}_e(p)$ and $O_e(p)$ are SSD and overlap errors, respectively. The search is carried out on discrete values of π chosen near the previous pose π_{t-1}. $O_e(p)$ is set to a constant penalty ϵ_O (set to a very large value, half of the maximum possible $\text{SSD}_e(p)$ in the experiments) if there is an overlap at p and $\text{SSD}_e(p) = D(p)^T D(p)$. After the minimization, the appearance of the rigid part is updated using the current image and positioned support map [11].

3. Results

We demonstrate our system on three video sequences. The input and output sequences are available on our Web page[1]. All system parameters remain the same for all sequences, and the thresholds are computed in the same way. No fine-tuning of the many thresholds and parameters is necessary. For instance, τ_H is set to 2, $\tau_{overlap} = 80\%$ and $\tau_{QoM} = 10\%$. The only difference in parameters, from one sequence to the other, is the background subtraction, which has been adjusted to give a satisfactory foreground when using a single reference background image for sequences A and C, and was manually obtained for sequence B. Processing time for a sequence is linear with the number of images and is constant, $O(C)$, for any given frame, thus preventing process overload with time. For all of the test sequences, the *Segmenter* uses a past neighborhood chosen with QoM and a future neighborhood fixed to 3 frames; therefore the first available result is for frame #3.

Figure 5 shows the estimated time scale (equation (5)) for sequence A. The length N of the past neighborhood

[1] `http://vision.gel.ulaval.ca/~sdrouin/tracking/`

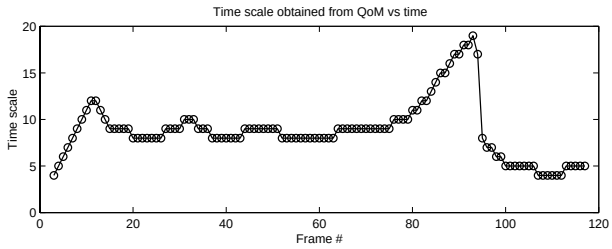


Figure 5. Time scale obtained from QoM for sequence A.

used by the *Segmenter* is plotted for each frame of the sequence. A higher value of N means less motion, as at the begin of the sequence, where N extends the past neighborhood up to the very first frame. A smaller N, as at the end of the sequence, means more motion. In this case, both arms move at the same time.

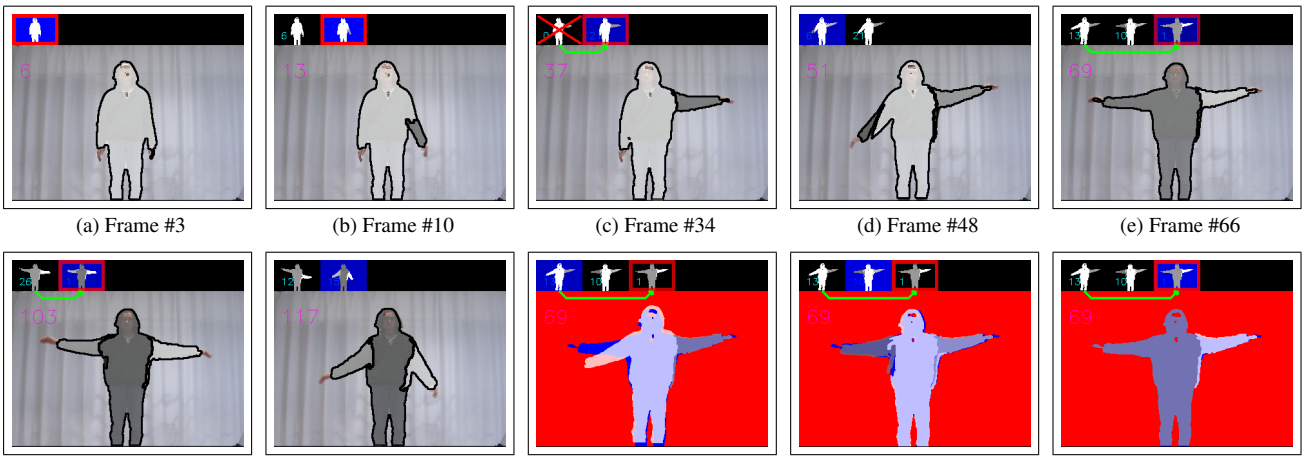
Figure 6 shows the output of the *Modeler* for selected frames of sequence A. In these results, the parts are shown in different gray levels according to their label and are superimposed on the original image. The external contours of each part is shown for visualization. At the top of each image, a toolbar shows the model candidates in the memory of the system. In sequence A, the subject moves his left arm, then his right arm. For the first segmentation (Figure 6a), there is not enough motion for the arm, so the whole foreground is modeled as a single part. As the arm moves, this model becomes invalid when E_{SSD} becomes too large. A new model with two parts is created in Figure 6b; this new model is successfully tracked until the second arm starts moving (Figure 6c). At this point, the system starts generating a series of new model hypotheses that always include the already discovered arm (Figures 6d to 6f). These new models compete with the already strong two-part model as can be seen in Figure 7a, which shows the memory credits attributed to each model as a function of time. New models are created starting shortly after frame #40, but it is not until frame #100 that a three-part model is created (Figure 6f), that will eventually surpass (in memory credits) the two-part model. Its final state is shown in Figure 6g.

Additional insights in the process of model evaluation can be obtained from Figure 7b, which plots E_{SSD} and τ_{SSD} for the first two models (Figures 6a and 6b). Each value is shown for the single-part model and the two-part model, from their creation frame until they are forgotten by the system (near frame #20 and #80, respectively). The first part of Model 3 has high E_{SSD} and τ_{SSD} since its boundary (where outlier errors are more frequent) counts for a larger proportion of the whole part. Thresholds are automatically adapted independently for both parts, which results in a smaller threshold for the main body part (with a value closer to the first model). Also, even after the two-part model stops being successfully tracked as a whole (after frame #40), E_{SSD} for the arm always remains below its thresholds, which suggests that this part of the model was successfully learned. Therefore, when the final three-part model is created by the system at frame #100, this part, which was first segmented in frame #10, is kept. This demonstrates the ability of the system to incrementally discover the parts of an object.

Figures 6h to 6j show the result of the *Tracker* at frame #66 of sequence A. The first model (Figure 6h) is the two-part object which accumulated many memory credits in the first half of the sequence. The system is still tracking it, but only the part corresponding to the arm is successfully tracked. The second model (Figure 6i) is the three-part model that was segmented near frame #40. At the selected frame, the arm on the left side of the image cannot be successfuly tracked because its part includes a portion of the torso, which causes too much error. The *Modeler* then requests a new segmentation and creates a new model (Figure 6j) which is based on Model #3 (because it has more memory credits than Model #12, which could also have been used as a base model). The unmodeled portions of the foreground (body and arm on the right side of the image) are resegmented from scratch and result in a single part because the *Segmenter* cannot identify two parts from motions in that particular frame. This new model, however, is the first of a series of transition models which will result in the final three-part object of Figure 6f.

Figure 8 shows the output for sequence B, which is a robotic arm moving in a plane nearly parallel to the image. This sequence is interesting because there are very few feature points on the robot, thus the *Segmenter* mainly relies on edges. The robot first moves as a single rotating rigid part, then other joints are progressively rotated until all three sections move relatively to each other. This progression can be seen in the results of Figures 8a to 8c, where parts corresponding to the three sections of the robot are sequentially added to the active model. The results illustrate the use of model memory at frames #41 and #42 where the system decides to use a model with a part overlapping two of the sections of the robot (Figure 8d) and then comes back to the previous model in the subsequent frames (Figure 8e). It is the role of model memory to reinforce valid models over time. Models that are often re-observed (tracked) will remain active and transition models which are not re-observed will be quickly forgotten by the system. The final model is correctly segmented despite poorly textured images through the use of edge templates. The system was able to add parts as they started moving in the sequence, and then continue to track them at the end of the sequence even when one joint stopped rotating.

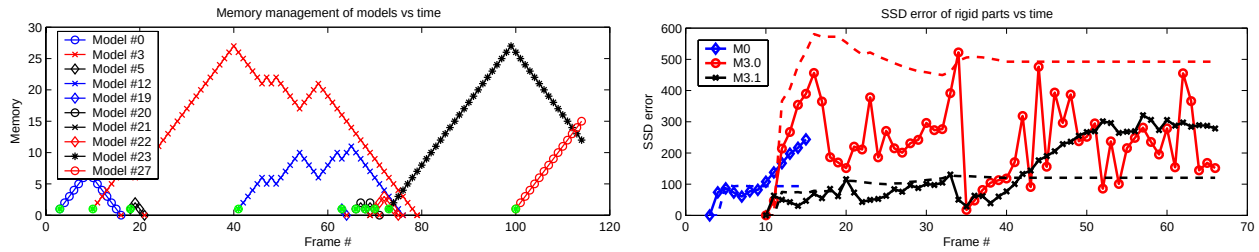
Figure 1 shows the output for sequence C where a human subject moves from a crouching to a standing position.



(a) Frame #3 (b) Frame #10 (c) Frame #34 (d) Frame #48 (e) Frame #66

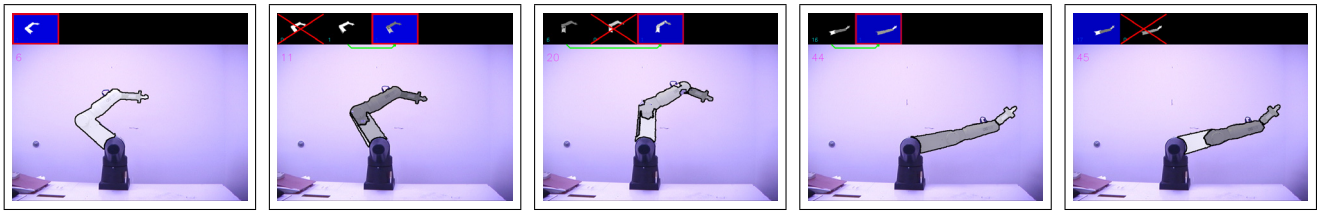
(f) Frame #100 (g) Frame #114 (h) Model #3 in frame #66 (i) Model #12 in frame #66 (j) Model #20 in frame #66

Figure 6. Discovered object parts at selected frames of sequence A and tracked model candidates.



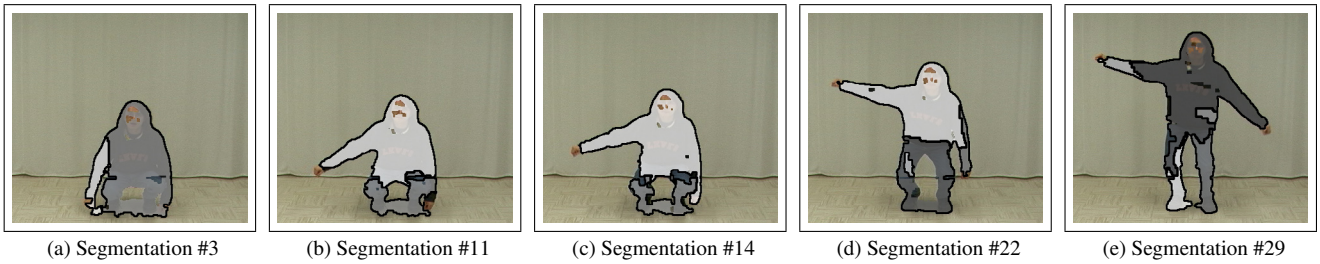
(a) Memory management of models (b) SSD error of rigid parts and adaptive threshold (dotted lines)

Figure 7. Memory management of models and SSD error of rigid parts for sequence A.



(a) Frame #3 (b) Frame #8 (c) Frame #17 (d) Frame #41 (e) Frame #42

Figure 8. Discovered object parts at selected frames of sequence B.



(a) Segmentation #3 (b) Segmentation #11 (c) Segmentation #14 (d) Segmentation #22 (e) Segmentation #29

Figure 9. Open-loop segmentation for selected frames of sequence C.

Since this motion of the subject does not result in rigid 2D transformations, we expect the system to resegment quite often and to sometimes find strange models in the transition. This is shown in the results, where a first rigid model composed of two rigid parts (Figure 1a) is progressively transformed into a six-part model (Figure 1e). During the transition, the non-rigid 2D motion produces parts which are quickly forgotten, for instance in the lower portion of the model in Figures 1b and 1c. However, the right arm of the subject, which moves rigidly in the plane, is always modeled in the same way throughout the sequence. This result shows that the system can learn parts of the model at different points in time, as the observed shape evolves.

Figures 9a to 9e show the results of applying the motion-based segmentation algorithm (in open loop) to the selected frames of the same sequence C. These results show the unreliability of the segmentation which does not always identify the arm as a region (Figures 9b to 9d) or, because it cannot build upon multiple observations, identifies regions which overlap multiple sections of the subject (torso-arm-leg region in Figure 9e). It is worth noting that this motion-based segmentation, as opposed to a colour-based segmentation, allows a part to be composed of multiple colours or multiple adjacent parts to share the same colour as shown in Figures 1, 6 and 9. From these results, it would appear that the modeling-tracking with segmentation on demand produces a more stable overall segmentation.

4. Conclusion

This paper addressed the fundamental problem of automatically discovering the parts of an unknown moving object in a monocular video sequence. The *Modeler* lies at the heart of the system and minimizes the impacts of erroneous segmentations and departures in tracking. It was shown how the memory of models stabilizes the detection by reinforcing valid models and managing motion-based segmentation of rigid parts of a same model at different points in time.

Further work needs to be done to generate articulated models from the discovered object parts. This additional information would improve tracking by imposing model-wide constraints on the pose of the parts. The system could also be improved by generalizing the approach to 3D motions [9] but without the need for keyframes.

Acknowledgements

This work was supported by NSERC-Canada and Precarn. The authors also express their gratitude to A. Schwerdtfeger for proofreading the manuscript.

References

[1] D. Ballard and C. Brown. *Computer Vision*. Prentice Hall, 1982.

[2] J.-Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. Technical Report included into OpenCV distribution, Intel Corporation, 2000.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

[4] A. Camurri, I. Lagerlöf, and G. Volpe. Recognizing emotion from dance movement: comparison of spectator recognition and automated techniques. *International journal of human-computer studies*, 59:213–225, 2003.

[5] C.-W. Chu, O. C. Jenkins, and M. J. Matarić. Markerless kinematic model and motion capture from volume sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 475–482, June 2003.

[6] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 126–133, June 2000.

[7] J. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[8] D. M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, January 1999.

[9] N. Krahnstoever and R. Sharma. Articulated models from video. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 894–901, June 2004.

[10] N. Krahnstoever, M. Yeasin, and R. Sharma. Automatic acquisition and initialization of articulated models. *Machine Vision and Applications*, 14(4):218–228, September 2003.

[11] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, June 2004.

[12] K. Nickels and S. Hutchinson. Model-based tracking of complex articulated objects. *IEEE Transactions on Robotics and Automation*, 17(1):28–36, February 2001.

[13] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, March 1978.

[14] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

[15] M. K. Titsias and C. K. I. Williams. Fast unsupervised greedy learning of multiple objects and parts from video. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, page 179, June 2004.

[16] J. Wills, S. Agarwal, and S. Belongie. What went where. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 37–44, June 2003.