ABSTRACT

This paper presents a novel approach to track multiple articulated objects in a video sequence. The key idea is to define a model of the object using a set of geometrical primitives linked by physical constraints, and exploit the physics engines to solve these constraints while the model adapts to the object under the influence of local mean-shift processes. This novel approach to object tracking has numerous advantages: the model provides rich geometric information about the object at the articulation level; multiple touching objects are implicitly distinguished using collision detection strategies; physics engines are able to efficiently manage both image-based and model-based constraints simultaneously for a negligible computational cost, suggesting their potential interest for many more image processing applications.

Index Terms— Tracking, Physics engines, Mean-shift

1. INTRODUCTION

Single and multiple object tracking in video sequences are popular topics in the computer vision community, with numerous applications in scene monitoring [1–3]. Throughout the literature, two sets of approaches have been studied. Two-step approaches perform a preliminary detection of the objects on each sequence frame, and then reconstruct the objects trajectories using various association techniques [4–6]. Such methods are however not adapted to track articulated objects, as they focus more upon the object trajectory rather its geometrical conformation. Direct approaches perform both detection and tracking simultaneously by iteratively adapting user-defined models to match the objects, and repeat this process on each frame to follow all targets. Popular candidates in this approach include mean-shift [2, 7] and active contour [8, 9] techniques. These methods require a higher acquisition frame rate to allow each models to adapt locally to the new object position, and are well suited for objects undergoing frequent shape or pose changes. Unfortunately, these methods only provide global localization and conformation of the objects, and fail to provide shape information at the articulation level. This is a very limiting factor when tracking multiple touching objects in behavioral studies, where indeep shape information is necessary to extract objects interactions events. In [10], a mean-shift-based method was proposed to track multiple cells observed in phase-contrast microscopy, exhibiting a particular two-phase intensity profile (bright center and dark boundary). This pattern was modeled using multiple mean-shift kernels stitched together in a rigid way, and the displacement of the global model was driven by the combination of all the underlying local mean-shift processes. While the method could distinguish touching cells efficiently, the model rigidity failed to provide shape information about the tracked cells.

This paper proposes a novel and original approach to track multiple articulated objects in video sequences using physics engines, in order to quantify shape and orientation of all objects at the articulation level even upon contact. In short, our approach consists in defining a model for each object in the scene using geometrical primitives that are a) linked through physical constraints and b) attracted toward image features through a data-attachment process, which in our case exploits the mean-shift approach. The physics engine solver is then used to combine and solve model-based and image-based constraints and deform the model accordingly. As a result, this approach is able to a) track objects in a very fast and robust way, b) maintain the identity of multiple touching objects, and c) provide local shape information at the articulation level through each of its geometrical primitives. The remainder of this paper is organized as follows: we first recall below the principle of physics engines, and then describe the proposed method for tracking articulated objects in section 2. Section 3 discusses the performance of the method on two sample applications. Finally, section 4 concludes the paper by discussing the genericity of the proposed approach and its potential applications to other types of objects.

2. TRACKING WITH PHYSICS ENGINES

2.1. Physics engines and models

Physics engines are software or hardware toolkits dedicated to the realistic simulation of complex physical systems governed by Lagrangian dynamics, and are extensively used in mechanical design [11] and in the computer-gaming industry. This paper shows how to exploit such engines to track articulated objects in video sequences. By representing the objects using physically constrained models, and by linking these models to the image data, the engine solver is able to track the evolution of all objects simultaneously, even if they come into contact over time.

For each object to track, we define a model \( M_{O,C} \) by a set \( O = \{o_i\}_{i=1..n} \) of primitives (also called bodies), and a set \( C \) of constraints or joints linking the bodies together. Each body is characterized by a mass, a damping (i.e. friction) factor, and an anchor where displacement forces will be applied. Primitives include rectangles, circles and other convex polygons, and more complex models can then be constructed as combinations of these primitive bodies. Bodies are then linked pair-wise using a specific joint describing the degree of freedom of the articulation considered. In classical physics engines, one can use joints of at least four types: distance, angle, elastic and slider. Collision between bodies of a given model, and between bodies of distinct models are both handled by the physics engine, using various built-in algorithms that either favor computation time or physical realism.
Fig. 1. Physics model of a mouse. (a): description of the model bodies (solid lines) and the gate kernel of their attractive maps (dashed lines). To prevent overlapping of other instances of mouse model, grey bodies are not allowed to overlap with grey bodies of another model. (b): final model matching a mouse in a sequence frame.

2.2. From image data to physics engines

In order to attract the model toward the object in the image, we compute an image-driven attraction force for each of its individual bodies, and let the physics engine solver combine these forces with the physical constraints imposed by the model joints in order to compute the final displacement. We first create a pool of attraction maps \( M = \{m_i \}_{i=1, \ldots, n} \) which can be either computed from the original image using a filter-like method, or as in traditional mean-shift, given by the look up table of a predefined feature space. Here we only use two attraction maps: a binary mask of the image is used to attract peripheral bodies toward the object center. An edge map is used to attract peripheral bodies toward the object boundary. Note that the maps are computed once for each sequence frame and are not updated during the model deformation. We now need to convert the attraction maps data into physical forces and apply them to the model through its anchors. We start by assigning to each anchor \( a \) (defined by its position \( p_a \)) an appropriate attraction map \( m \) taken from \( M \). Then, this map is restricted to a subset of pixels centered on the anchor (noted \( Q_a \)). In our case, we define this subset as a simple disk of arbitrary radius around the anchor position, although more complex gating kernels can be employed. Finally, for each anchor \( a \), one computes the resulting attraction force \( \vec{f}(a) \) as follows:

\[
\vec{f}(a) = \sum_{q \in Q_a} q \vec{p}_a \vec{q}, \quad (1)
\]

where \( q \) describes all pixels within attractive map.

Finally, once all forces have been computed for each anchor, they are sent to the physics engine, which will take care of combining these forces with the physical constraints of the model. The deformation is an iterative process, therefore attraction forces are recomputed at every iteration until a stop condition is reached. Typically, a classical criterion is to detect when the system reaches equilibrium, i.e.,

\[
\left\| \sum_{a \in M} \vec{f}(a) \right\| < \epsilon, \quad (2)
\]

where \( \epsilon \) is a small constant.

\[
\forall a \in M, \quad \vec{v}(a_t) = \frac{\vec{a}_t - \vec{a}_{t-1}}{\epsilon}, \quad (3)
\]

2.3. A physically constrained mean-shift approach

The principle of mean-shift tracking [2] is to move a model toward a local maximum defined by a kernel. In our method, the model corresponds to a body of our physics model, and the associated kernel is analogous to the attraction map associated to the body anchor. The iterative process that seeks for the local maximum lead to a translation vector that we transform into a force for the physics system. The object will then be moved by the physics engine which will consider the whole forces and constraints of the model nor than the simple translation vector given at the mean-shift level. It is this step which gives the control of the mean-shift results to the physics engine. Our approach incorporates physical constraints between bodies of a same physics model, thereby improving object detection and tracking using a higher level of semantic information.

2.4. Motion prediction using the physics engine

The motion prediction used in this paper is original since it is performed under the supervision of the physics engine. We aim at finding the next position \( f + 1 \) of the target knowing its position at time \( t \) and \( t - 1 \) (fig. 1). We seek for the new object position considering the object prolongates its previous motion (i.e., we do not attempt to create a probability map of the next position). To do so, we take advantage of mass-aggregate engines. We recall that in such engines the whole object undergoes complex movements as a result of the constrained linear motion of each component. Fig. 2ab illustrate rotation prediction and Fig. 2cd illustrate neck elongation caused by the head motion prediction which creates a pull force on the neck. We exploit this motion information by computing the speed vector for each anchor from frame \( t - 1 \) to frame \( t \), and apply this speed to the model at time \( t \) to obtain the constrained predicted position on frame \( t + 1 \). Physics engines offer the ability to bypass the speed computation made with forces and allow one to set directly a computed speed to anchors (i.e. which find its motivation in impulse). At the end of the convergence iterations (i.e. which find its motivation in impulse), the formula is applied to each anchor, then the physics engine is iterated once. To ensure the current inertia will not interfere with the next image to process, each anchor’s inertia is set to null vector.

\[
\forall a \in M, \quad \vec{v}(a_t) = \frac{\vec{a}_t - \vec{a}_{t-1}}{\epsilon}, \quad (3)
\]

2.5. Dealing with multi-object tracking

The appealing interest of the physics model approach is that an arbitrary number of models can be defined in the same world and deformed simultaneously using the physics engine solver. One can thus perform concurrent multi-object tracking, where each model follows a given target in the image according to the displacement of each of
Fig. 3. Sequence of 8 images taking advantage of the motion prediction. On each frame, the red model is the current matching model for the frame, and the white model is the initialization which will be used for the next frame.

Fig. 4. Summarizing view of all steps involved in the tracking.

its constitutive objects. In case two objects come into contact, ambiguities related to model overlapping are automatically detected and handled by the physics engine through efficient collision detection and feedback strategies.

2.5.1. Dealing with object proximity

While model overlapping is forbidden, it may occur that anchors of same nature are sufficiently close such that their attraction maps do overlap. Consequently, an identical portion of image information is exploited to compute the movement of concurrent anchors, potentially attracting the models toward wrong positions (as illustrated in Fig. 5). We remove the ambiguity by weighting the influence of each computed vector by introducing $\delta(q)$ such as it represents the number of anchor sharing $q$ and having the same attractive map nature. Eq. 1 thus becomes:

$$\mathbf{f}(a) = \sum_{q \in Q_a} q \mathbf{p}_a \mathbf{g} / \delta(q) \quad (4)$$

Illustrating movies can be watched online at the following address: [www.bioimageanalysis.org/8722](http://www.bioimageanalysis.org/8722). Two movies are available: the first illustrates all the steps of convergence for each frame, and illustrates the motion prediction step (slowed down for viewing purpose). The second is the final tracking of the whole sequence.

Fig. 5. Matching mice in close contact without (a,b) and with proximity handling (c,d). With proximity handling, the overlap between attractive maps of same nature is removed from the computations, yielding a correct adjustment of the models.

3. EXPERIMENTS & RESULTS

Our goal is to extract behavioral statistics from the co-habitation of two mice evolving in a rectangular cage observed with a webcam. A 42 seconds-long sequence was acquired at 12.5 frames per second, yielding 530 frames. We focus on labeling three types of events: oral-oral contact and oral-genital contacts (in both directions), highlighting the need to determine at all times the position and orientation of each mouse. This quantification was previously done manually with a timer, with an extensive use of playbacks on the same sequence to label the different events. Moreover, these events having a very short duration, they are often labeled in over-sized time frames. We describe the mouse physics model below and then present quantitative results on a sample sequence.

Fig. 6. Mouse with a contracted posture (a) and with the head on the side (c). (b,d): final mouse model super-imposed on the original images.

Fig. 7. Force vectors computed for each model anchor. (a): original image. (b,c): force vectors (cf. Eq. 4) super-imposed on a gradient map (b, green vectors) and binary map (c, pink vectors) of the image. (d): vectors and boundaries of their attractive maps.

3.1. Sample physics model

Our mouse model is illustrated in Fig. 11 and is defined as follows: the bodies H, B and N are the head, the belly and the neck of the mouse, respectively. To define the degree of freedom, we allow the neck to slide along the belly, constrained between parts C and D. A sliding joint is added to bound the distance between the belly and the neck. The head is a simple circle and is linked to the neck by a distance joint. Our model thus has two degrees of freedom: a) one for the sliding neck simulating mouse elongation or retraction, and b) one for the head rotating at the end of the neck. In order to
prevent overlap between different mice models, we impose that all bodies colored in gray may not overlap with similar bodies of another model (see Fig. 7 for an illustration of mouse contact situation). Anchors H and A1 through A4 exploit an attractive map based on intensity gradients, while B exploits an attractive map that is a binary image highlighting dark image regions (i.e., reflecting mice). Dashed circles depict the boundary of the attractive maps. The final model is shown super-imposed on a mouse image in Fig. 7.

### 3.2. Validation

As mentioned earlier, convergence is detected on each frame when the model(s) reach a steady-state. It is however possible with mean-shift approaches that the model converge to a wrong position, e.g., right in the middle of two local maxima. To evaluate this behavior, we define for each model a matching score function $s(M)$ ranging from 0 to 1 by:

$$s(M) = \frac{\sum_{a \in M} Q_a}{N_a(M)}$$

where $Q_a$ is the average value of the pixels forming $Q_a$, and $N_a(M)$ is the number of anchors in the model. In case the model matches perfectly with the object, then $s(M) = 1$, this value decreasing otherwise. The average matching score for our 530 frames sequence is 0.9 for each mouse, with a standard deviation of 0.1, showing that the mice were correctly followed all along the sequence. We stress that the physical constraint layer increases the robustness of the mean-shift process, since more information is combined during the deformation process, thereby reducing the risks of driving the model towards a wrong position. Additionally, note that if the attractive map linked the anchor is the look up table of a feature space, then $s(M)$ can also be computed as the model and reference distributions, e.g., using the Bhattacharyya distance [2].

### 3.3. Events quantification

Tracking results are illustrated in Figs. 5, 6, and 7. The method is able to automatically detect the position and orientation of the mice even when they are in contact, and compute the proximity of the head and posterior of each mouse to label the events in an efficient way. The final timeline of events detected in this sequence is presented in Fig. 8.

![Timeline of labeled events](image)

Fig. 8. Timeline of the labeled events found on a 40-second sequence (530 frames). In this particular sequence, one can note that the ‘head 1 - genital 2’ event never occurs.

### 3.4. Computational performance

Thanks to its computational efficiency, the method is able to process the whole sequence in 55 seconds, i.e., yielding a processing rate around 10 fps, which is very close to real-time in the present case. In order to compute the overhead of the physics layer on the overall performance, we measured the proportion of processing time spent, and found that only 3.5% of the time was consumed by the physics engine to solve all constraints and deform the models. The main bottleneck of the method is the iterative computation of the forces to apply to each anchor. Therefore, the complexity of the problem is almost solely dependent on the number of mean-shift processes to solve (i.e., function of the number of models, of anchors per model, and of pixels per local attraction map). We may conclude that the physics constraint layers adds a neglectable computational overhead with respect to its numerous advantages.

### 4. CONCLUSION

In this paper we presented a novel approach to track multiple articulated objects in 2D video sequences using geometrically constrained models and physics engines. The method describes the tracking problem a set of multiple local mean-shift processes applied to individual bodies, linked together toward a set of user-defined degrees of freedom and supervised by a physic engine. The physics engine is used to solve all these local processes while handling user-defined constraints such as collision between different models in a fast and fully automated fashion. We believe that the combination of efficient mean-shift approaches with the power and flexibility of physics engines provides a novel and powerful tracking tool with numerous potential applications involving articulated objects. Our future work will focus on improving the computation time using hardware acceleration, in order to reach real-time computation and allow videos to be processed and labeled directly during their acquisition.

### 5. REFERENCES


