Physical simulation for monocular 3D model based tracking

Damien Jade Duff, *Student Member, IEEE,* Thomas Mörwald, Rustam Stolkin and Jeremy Wyatt

***Abstract*— The problem of model-based object tracking in three dimensions is addressed. Most previous work on tracking assumes simple motion models, and consequently tracking typically fails in a variety of situations. Our insight is that incorporating physics models of object behaviour improves tracking performance in these cases. In particular it allows us to handle tracking in the face of rigid body interactions where there is also occlusion and fast object motion. We show how to incorporate rigid body physics simulation into a particle filter. We present two methods for this based on *pose* and *force* noise. The improvements are tested on four videos of a robot pushing an object, and results indicate that our approach performs considerably better than a plain particle filter tracker, with the *force* noise method producing the best results over the range of test videos.**

I. Introduction

Model-based tracking of pose is one of the most widely addressed computer vision problems, with a particular importance in contemporary cognitive robotics. Model-based tracking typically fails, though, in a number of key situations that are frequently encountered. These include failure due to occlusion, or rapid target motion, which in turn causes motion blur and is challenging for local search methods. At this point the tracking often falls back on the dynamics model, the component that specifies how the object can move through space. Yet in most tracking methods the dynamics model is simple and often wrong. One way to improve it is to use a physics informed dynamics model. This is what we do in this paper.

The main contribution of this paper is that we show how to incorporate a dynamics model based on physical simulation into an existing recursive monocular 3D model-based particle-filter based tracker. We make the specific prediction that by doing this we will improve the performance of the tracker in situations where image information is insufficient to disambiguate object motion using existing methods alone (particularly when occlusion and motion occur).

Such improvements will have a positive impact in scenarios involving robotic manipulation of objects as well as human-robot interaction where objects are frequently obscured from view or knocked about. More broadly, we see computer vision as potentially bound up with many other problems in robotics, such as motion planning, which often are addressed using simulation-based approaches too.

D.J.Duff, J.Wyatt & R.Stolkin are at the School of Computer Science, University of Birmingham, B15 2TT, Birmingham, United Kingdom

T.Mörwald is at the Institute of Automation and Control, Vienna University of Technology, Gusshausstrasse 27-29 / E376 1040 Vienna, Austria

Emails: {D.J.Duff,J.L.Wyatt,R.Stolkin}@cs.bham.ac.uk, moerwald@acin.tuwien.ac.at

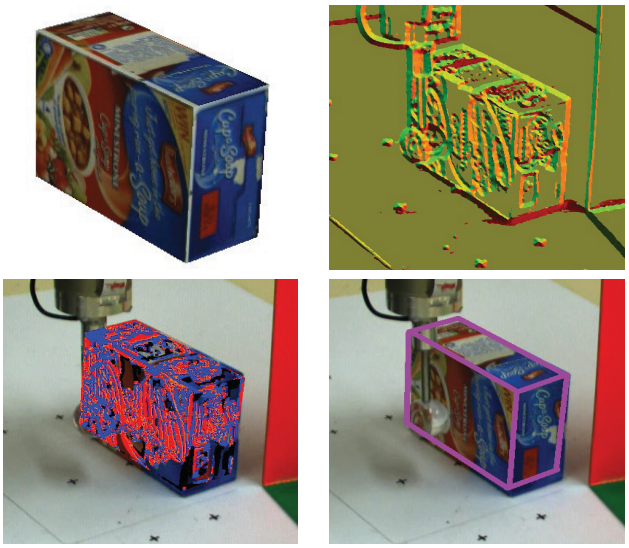

Fig. 1. **Top left**: A view of the textured model tracked in experiments. **Top right**: A view of the edge image extracted from a candidate image. **Bottom left**: A projection of the edges of the textured model onto the image (obtained by first projecting the texture from a key pose, extracting edges, projecting the result onto the model surface and then reprojecting the edges onto the image). **Bottom right**: A reconstruction of the pose of the object with respect to the camera.

After reviewing related work, we briefly describe the initial tracking framework that we build apon, as well as the fundamentals of physical simulation. We show two ways to incorporate a physics model into the particle filter. Lastly, we present results detailing the differential performance of the basic tracker and these different approaches to incorporating physical simulation into that tracking framework.

Related work.: There has been some work on using dynamics models of people while tracking them [1], [2]; it remains unclear however as to whether the physical models can provide an improvement over the use of motion capture data alone; the scenario considered in the present paper allows us to start investigating improvements with a probabilistic dynamics based only on a standard rigid physical model.

Using physical models of deformable objects has also been demonstrated from motion capture data [3] and as a way of dealing with noise while tracking rigid objects [4]. In the area of motion synthesis, related techniques may be found [5][6] but these techniques need adapting to make them robust when input poses are substituted for image data, which is what is being described in this paper.

Finally, physics models have been used in two-dimensional motion estimation of moving and colliding objects [7]. In the present paper we consider real-time object tracking in 3D.

II. Overview of approach

The model based object tracker extended in the present paper is described more fully in [8], [9][1]. The object tracker is provided with a three dimensional object model. It uses geometric edges to maintain initial track on the object while it recovers the textures on surfaces of the object. Consequently it is able to track both geometry edges and texture edges. Fig. 1 shows a three dimensional object model and illustrates how texture edges are projected to an image for matching.

A particle filter framework is employed to recursively sample candidate poses over time, which are accepted or rejected probabilistically depending on an edge match score. The edge matching process is accelerated using a GPU programmed using OpenGL Shader Language.

In this paper we alter the probabilistic dynamics model used by the particle filter by running candidate poses through the physics simulator. We add noise to this process in one of two ways - either by perturbing the pose after simulation, or by introducing probabilistic forces into the simulation.

III. Texture tracking framework

A. Model

In this paper, we are not concerned with the texture recovery phase, rather the consequent object tracking phase. As such we make use of a previously acquired textured object model as seen in Fig. 1. The model consists of a quad- and tri-mesh with texture stored as raster images.

B. Particle filtering

Particle filtering is sequential or recursive importance sampling of the probability distribution that governs the probability of an object state x_{t_i} at a time t_i given all preceding observations of the object $z_{t_0:t_i}$.

The form of that probability distribution is well known in the context of recursive estimation. We state it here and then describe it briefly:

$$p(x_{t_i}|z_{t_{0:i}}) = \frac{p(z_{t_i}|x_{t_i})}{p(z_{t_i})} p(x_{t_i}|z_{t_{0:i-1}}) \tag{1}$$

$$= \frac{p(z_{t_i}|x_{t_i})}{p(z_{t_i})} \int_{t_0}^{t_i} p(x_{t_i}|x_{t_{i-1}}, t_\Delta) p(x_{t_{i-1}}|z_{t_{0:i-1}}) dx_{t_{i-1}}$$

This form separates the observation likelihood model from the model for the evolution of state by assuming that the current state depends on all previous states and observations only through the previous state and that the current observations depend on all states and observations only through the current one.

This allows us to express the probability of any current pose in terms of the previous pose estimate $p(x_{t_{i-1}}|z_{t_{0:t-1}})$, the state evolution (or "dynamics") model $p(x_{t_i}|x_{t_{i-1}}, t_\Delta)$ and likelihood of the current image $p(z_{t_i}|x_{t_i})$. Note that in this paper we make the unusual step of allowing the dynamics model to depend on the time elapsed between timesteps $t_\Delta = t_i - t_{i-1}$, which allows us to use the same notation as when discussing simulation, and to define extra recursions.

We maintain an imperfect representation of the above distribution as a set of weighted hypothesis "particles" $\langle x^j, w^j \rangle$:

$$p(x_{t_i}|z_{t_{0:i}}) \approx \sum_{\langle x^j_{t_i}, w^j_{t_i} \rangle} w^j_{t_i} \cdot \delta(x - x^j_{t_i}) \tag{2}$$

We use importance sampling to calculate the succeeding probability distribution after time has passed and new observations have been made. In effect, importance sampling allows us to approximate any probability distribution $p(x)$ by sampling from a different probability distribution (the proposal distribution) and weighting the resulting samples. Much of the time, analogous to Kalman filtering, the proposal distribution is the prior over x_{t_i} as calculated from the previous distribution over pose (the second part of equation 1 above), via the probabilistic dynamics model. This is the case in the particle filter employed in the present paper. Samples are consequently weighting by the image likelihood (the first part of equation 1 above).

Subsequently, resampling is performed to make all the weights equal by sampling a number of particles proportional to their weight. For each particle we use its weight to calculate its number of successors N^j (where J_b is the target number of particles):

$$N^j = \frac{w^j}{\sum_k w^k} J_b \tag{3}$$

The union of the successor particles from the current distribution make up the resampled distribution. The algorithm in outline is therefore:

1. Resample $\{\langle x^0, w^0 \rangle ... \langle x^{J_a}, w^{J_a} \rangle\} \rightarrow \{\langle x^0, 1 \rangle ... \langle x^{J_b}, 1 \rangle\}$.
2. Sample from dynamics $x^j_{t_i} \sim p(x_{t_i}|x^j_{t_{i-1}}, t_\Delta)$.
3. Reweight by likelihood $w^j = p(z_{t_i}|x^j_{t_i})$.
4. If processing time remins, set $t_\Delta = 0$ and go to step 1.

Step 4 has been called recursive particle filtering [8] and is a heuristic feature of the tracker that we are making use of and enables the tracker to spend more time evaluating potential hypotheses if time is available.

The dynamics model used in the current tracker is the simplest possible, assuming a Gaussian distribution around the previous pose, where the variance Σ is provided (and for our purposes is a diagonal covariance matrix):

$$p_1(x_{t_i}|x_{t_{i-1}}, t_\Delta) = \mathcal{N}(x_{t_{i-1}}, \Sigma) \tag{4}$$

Note that sampling from a Gaussian in rotation space is not defined, since it is not a Euclidean space, but we take the ad-hoc step of sampling the vector of the unit quaternion representation of a rotation and renormalising.

More details about particle filtering can be found in [10], [11], [12], [13].

[1] See also http://cogx.eu/data/cogx/publications/moerwald2009edge.pdf

C. Edge likelihood calculation & GPU acceleration

Details of the likelihood calculation can be found in [8], [9]. In short, the number of matching edge pixels is counted with respect to the predicted and actual number of edge pixels in the image frame. See Fig. 1 for a broad illustration. In order to do this calculation efficiently, GPU acceleration is used to project texture edges into image space. But even so, for efficiency's sake a single key pose is chosen as a heuristic average of the particles with highest likelihood, from which the texture is projected and edges calculated. These edges are then reprojected onto the object surface (this step is needed to prevent thinning of edges) before the edge pixels are reprojected back into the image. This means that match calculations for hypotheses away from this key pose tend to be less accurate - encouraging the tracked distribution to become unimodal.

The number of recursions of the particle filter between time steps (which coincide with the reception of new image frames) although in theory a variable parameter, is in practice fixed in advance for experiments. In typical application, for stability it is fixed at 2, but in this work we vary it because different numbers of recursions work better with different dynamics models.

D. Multiple dynamics models I: Likelihood-rank selection

As mentioned previously, although the motivation for particle filters is probabilistic, in practice they employ imperfect representations (samples) of uncertainty in the true pose. One consequence of this is that good hypotheses, once found, may be lost when passed through the dynamics model probability distribution due to scattering - leading to jitter and tracking instability. The problem is amplified if the likelihood distribution is very peaky. The solution employed in the tracking framework we use is to keep back some particles from step to step based on the ranking r^j of their likelihood weights with respect to the size of the full distribution J (with a parameter R to determine the cutoff rank). Further, if more than one instance of a particle was created during resampling, only one instance is kept back.

The way that we model this is as a distribution incorporating multiple probabilistic dynamics models - the Gaussian dispersion model already mentioned in equation 4 and the Dirac delta distribution $\delta()$:

$$p_0(x_{t_i}|x_{t_{i-1}}, t_\Delta) = \delta(x_{t_{i-1}}) \tag{5}$$

The actual probability distribution used is a kind of combination of these two, which distribution being used depending on the rank and resampling status of the particle in question:

$$p^j(x_{t_i}|x_{t_{i-1}}, t_\Delta) = \begin{cases} p_0(x_{t_i}|x_{t_{i-1}}, t_\Delta) \text{ if } r^j < R, \text{1st sample} \\ p_1(x_{t_i}|x_{t_{i-1}}, t_\Delta) \text{ otherwise} \end{cases} \tag{6}$$

As such, the nature of the combination is procedural rather than probabilistic, but it was found to work in practice in the original tracking framework.

IV. Integration of physical simulation

A. Physical simulation

The existing textured object model, by adding a mass and density, as well as friction and restitution coefficients, can become a dynamic rigid body that can be simulated in any off-the-shelf physics simulator: in the case of the present paper, PhysX [14]. We also include a ground plane in the simulation. Physics simulators, given a starting state of the physical system $x_{t_{i-1}}$ and an elapsed time t_Δ provide the state of a physical system after that elapsed time x_{t_i}.

The way that these simulators work, in principle, is by specifying the physical system as a set of differential equations:

$$\frac{d}{dt}x_t = F(t, x_t, u) \tag{7}$$

And solving for x_t. Since in general we have no closed form, we numerically integrate the above equation to produce a "simulation function":

$$x_{t_i} = \mathcal{S}(x_{t_{i-1}}, t_\Delta, u) = x_{t_{i-1}} + \int_0^{t_\Delta} F(s, x_s, u)ds \tag{8}$$

(u here represents any specified external forces acting on the system).

In practice, however, physical simulation is much more procedural. We need work to adapt the above method to solve through collisions since the required timestep would need to be exceedingly small to maintain stability (since even if it were continuous, the function F must change very rapidly with respect to x_s between, for instance, states where collisions occur or do not occur). As such in practice a mixture of methods is often used. For instance, something like the above method can be used to solve for free-flight and constrained motion but collision detection and resolution routines to deal with discrete world events involving new collisions [15]. Indeed, constraints and collision detection/resolution are often used interchangeabely for parts of the simulation, but with different properties. However, we may still treat the procedures as a black-box simulation function $\mathcal{S}$.

B. Noise models

1) Simulation with pose noise: The most straightforward way of incorporating a physical simulator into our probabilistic sampling tracker is to use it to extend the dispersion model. This method of sampling from the dynamics model takes a previous pose hypothesis, puts it through the physical simulator, and then adds Gaussian noise to its location:

$$p_2(x_{t_i}|x_{t_{i-1}}, t_\Delta) = \mathcal{N}(\mathcal{S}(x_{t_{i-1}}, t_\Delta, 0), \Sigma) \tag{9}$$

It is important to note that our x contains only a 6-dimensional representation of pose. Particles do not carry an estimate of velocity with them. It would be possible to add velocity to the state vector x, or to estimate it anew for each particle. However, we treat the velocity as 0 at the start of each run of the simulator (as a minimal iteration on the existing method and to reduce the dimensionality of the filter). This restricts the range of things that we can model well (moving or bouncing objects, as found in [7],

are less likely to see success). However, the simulator is still able to model stable, almost stable, or multistable physical interactions. Indeed, the dynamics model tends to strongly prefer transitions into lower energy states (for instance, a transition from a box lying on its end to a box lying on its side) since the Gaussian noise acts to perturb the physical system while the dynamics acts to find the lower energy state.

2) Simulation with force noise: A different approach to sampling is to randomize the input into the simulator rather than the output. In our case, we perform sampling of an external force acting on the object. We sample these forces by uniformly sampling points on the surface of the object model, M_{surf}, and creating a force applied at that point whose direction and magnitude is acquired by sampling uniformly from within the bounds of an ellipsoid A_Σ.

$$p_3(x_{t_i}|x_{t_{i-1}}, t_\Delta) = \mathcal{S}(x_{t_{i-1}}, t_\Delta, u_i) \tag{10}$$

$$u_i = \overrightarrow{(u_i^{head} + u_i^{tail})(u_i^{head})} \tag{11}$$

$$u_i^{tail} \sim \mathcal{U}(M_{surf}), u_i^{head} \sim \mathcal{U}(A_\Sigma) \tag{12}$$

This approach has both the advantage and disadvantage of allowing only feasible successor states to be sampled. By sampling more inputs into the model and freeing up more simulation parameters, any state is conceivably achievable. In the present paper, since only forces are sampled, however, behaviours such as rotating through the ground plane are unachievable. Moreover, the effect of sampling transitions only between stable states mentioned above is exaggerated since physically impossible behaviour produces more ways of transitioning between stable states.

3) Multiple dynamics models II: Mixture model: In order to provide a probabilistically better grounded and more general approach to combining dynamics models than the likelihood-rank selection approach described above, we introduce a mixture model approach to combining dynamics models, of which we now have at least four to choose from (No noise p_0, Gaussian dispersion p_1, Simulation with Gaussian dispersion p_2, force noise p_3):

$$p_M(x_{t_i}|x_{t_{i-1}}, t_\Delta) = \sum_k \pi_k p_k(x_{t_i}|x_{t_{i-1}}, t_\Delta) \tag{13}$$

Practically speaking this means that, while sampling, for each particle a sub-model is chosen probabilistically according to the mixture parameters π_k. Likelihood-rank selection and mixture approaches can also be combined, by assigning a precedence to each sub-model and checking for them in order of precedence.

V. Experimental setup

In order to test the above-described changes and compare them to the pre-existing framework (which we consider to be amongst the state of the art in object trackers), we ran the various algorithms on 4 videos of an object being pushed by a robotic finger, each about 10 seconds long at 30fps. We analyse interesting time-slices in the tracking, particularly those normally associated with loss of track in the pre-existing framework. For example time-slices from the four videos, see Figs. 2, 3, 4 and 5.

Three conditions without physics and seven conditions with are presented here. The default tracking behaviour is found in condition A, where Gaussian dispersion is used and likelihood rank selection is performed to propogate particles without noise, with a rank selection parameter $R = 0.25$. Condition B is identical except that particles are propogated without noise with a probability of 0.1 (i.e. the mixture approach). Condition C has all particles undergoing Gaussian noise with none kept back. The remaining conditions involve simulation. Condition D uses simulation with Gaussian dispersion, and likelihood-rank selection to keep back particles without simulating or adding noise. Condition E uses a mixture approach to keep back particles, while F has no retention. Condition G is the force noise approach with likelihood-rank selection for retaining particles without noise. Condition H uses the mixture approach to propagate particles without noise and I has no retention. Finally, condition J is a mixture of Gaussian dispersion noise, simulation with Gaussian dispersion noise, and simulation with force noise ($\pi = 0.33$ for each of these models), with likelihood-rank selection retaining particles without adding noise with a higher precedence.

The number of particles was set to the default of 100, Σ for the translation component of pose was $diag(0.04, 0.04, 0.04)$ and for the rotation component was $diag(40, 40, 40)$, the defaults for the tracker. During Gaussian dispersion, additional dispersion is performed in the camera's z-axis with a Σ_z of 0.06, since this dimension is the most ambiguous visually. Image matching parameters were set to the default.

Experiments showed that the control conditions (using Gaussian dispersion of particles) obtained better results when the number of recursions is 2. For the novel simulation with Gaussian noise method, one recursion sometimes produced more accurate results, and subsequent recursions can only reduce performance for the force-noise approach since subsequent recursions do not invoke the simulation necessary to observe move the particles. As such, we present the results coming from using 2 recursions for conditions A-F, and one recursion for the force-noise conditions G-J.

In order to evaluate the resulting videos numerically we cannot simply count the number of successful tracks because the choice is often a subjective one since a track can be on a continuum anywhere from less to more successful. In the absence of ground-truth there is no point attempting to evaluate the values produced by the pose directly. Instead, we opt to "label" videos by producing by hand a track that matches the image and then evaluating tracker output by calculating the distance between the vertices projected onto the image by labeled poses and the vertices projected by a candidate track. Since the tracker is probabilistic, its behaviour is nondeterministic so we illustrate the distribution of peformance over 40 trials in Fig. 6.

VI. Results

A. Distracting edges, occlusion

In video 1, Fig. 2, the target object is pushed away from the camera by a robot finger while a coloured occluding

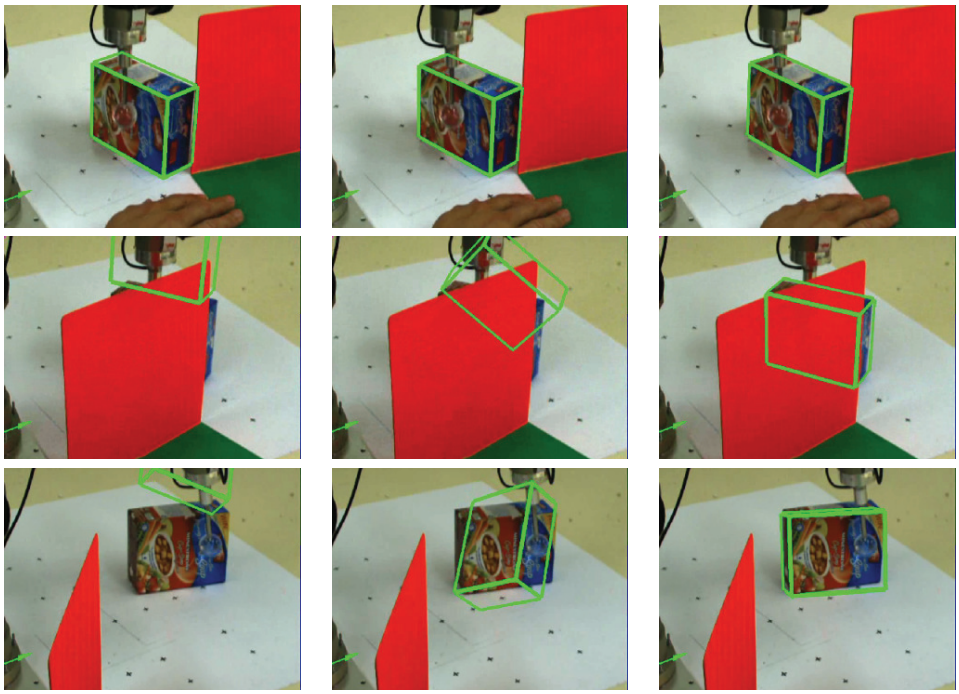

Fig. 2. **Video 1**, phases 1.1,1.2 and 1.3 from top to bottom. **Left**: Gaussian perturbation, likelihood-rank retention, 2 recursions (condition A). **Middle**: Simulation with Gaussian perturbation, likelihood-rank retention, 2 recursions (condition D). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition G).

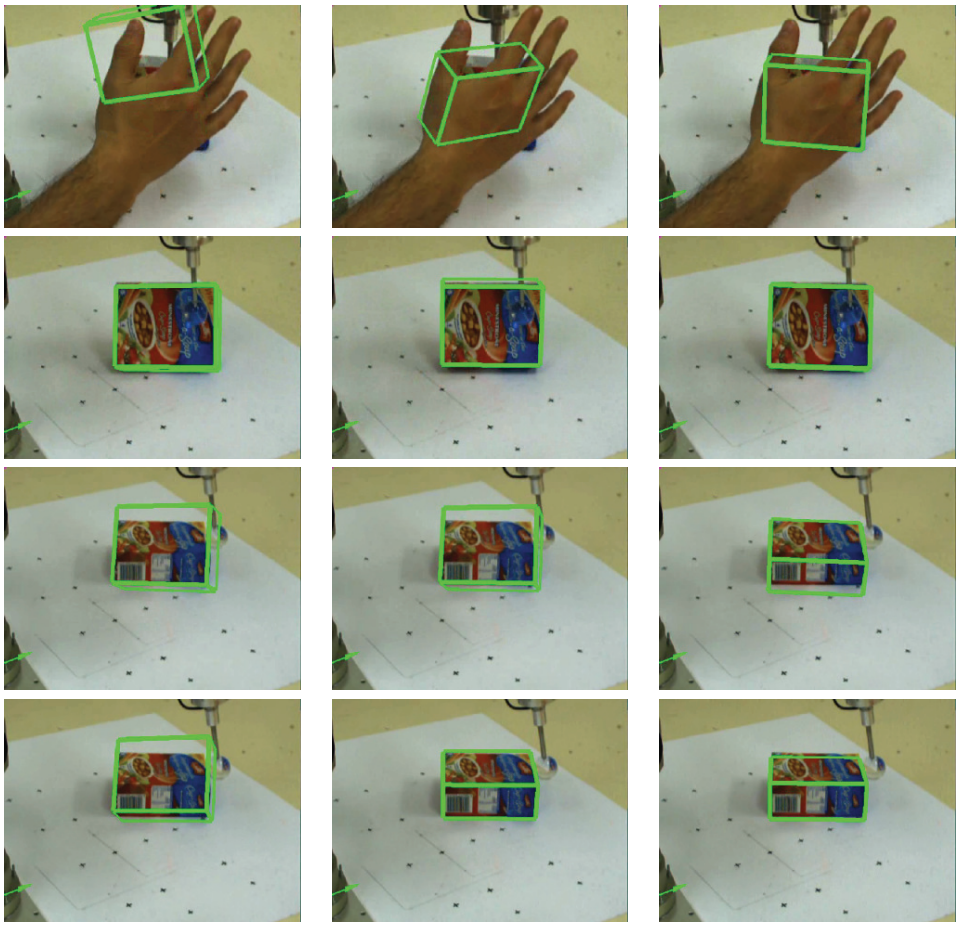

Fig. 3. **Video 2**, phases 2.1,2.2, 2.3 and 2.4 from top to bottom. **Left**: Gaussian perturbation, likelihood-rank retention, 2 recursions (condition C). **Middle**: Simulation with Gaussian perturbation, likelihood-rank retention, 2 recursions (condition D). **Right**: Simulation with force noise, likelihood-rank retention, 2 recursions (condition G).

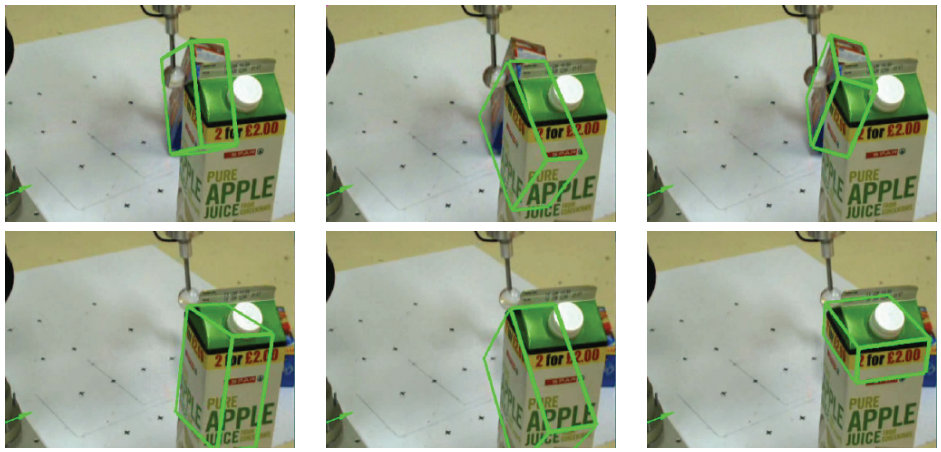

Fig. 4. **Video 3**, phases 3.1 and 3.2 from top to bottom. **Left**: Gaussian perturbation, likelihood-rank retention, 2 recursions (condition A). **Middle**: Simulation with Gaussian perturbation, likelihood-rank retention, 2 recursions (condition D). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition G).

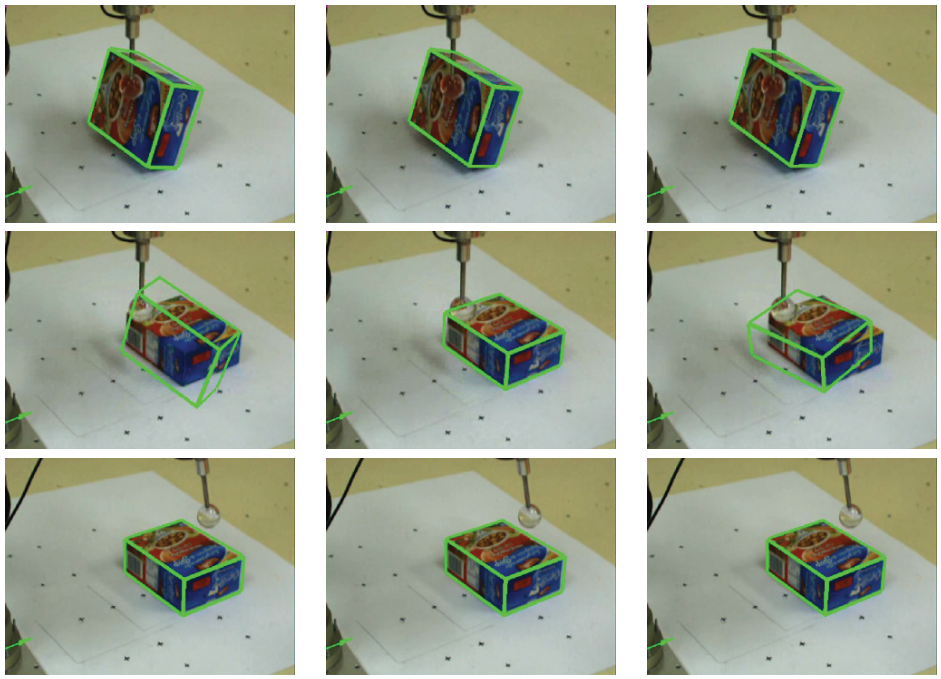

Fig. 5. **Video 4**, phases 4.1,4.2 and 4.3 from top to bottom. **Left**: Gaussian perturbation, likelihood-rank retention, 2 recursions (condition A). **Middle**: Simulation with Gaussian perturbation, likelihood-rank retention, 2 recursions (condition D). **Right**: Simulation with force noise, likelihood-rank retention, 1 recursion (condition G).

object (providing a very strong distracting edge) is passed in front of it. We can see that allowing only feasible motions to occur by using the force noise model prevents the track from passing to other nearby edges, which is possible since in this tracker occlusions are not considered explicitly. This effect is also observed in the numerical results seen in Fig. 6, chart 1.2. However, allowing the simulator to be followed by Gaussian perturbation results in physically inconsistent motions being hypothesised and track subsequently locking onto higher likelihood distractors. Thus, the simulation with Gaussian noise condition is not much better than without simulation in this case.

Furthermore, the numerical results show that the Gaussian noise typically allows the tracker to recover track (1.3) on this object after the occluder is removed, with the retention scheme having a complex effect. But the effect of recovery of track after it is lost is not robust, and, clearly, considering the success of the force-noise condition, maintaining the correct hypothesis throughout the tracking period leads to better behaviour here.

B. Occlusion, tipping

Video 2, Fig. 3, shows a hand occluding the target object (2.1), followed by the object being tipped away from the camera (2.2), being pushed over (2.3) and later resting (2.4). Again, force noise allows track to be maintained through the occlusion. Simulation with dispersion noise also does slightly better than the basic dispersion noise across all retention conditions. However, after the occlusion, simulation with Gaussian noise occasionally does not recover.

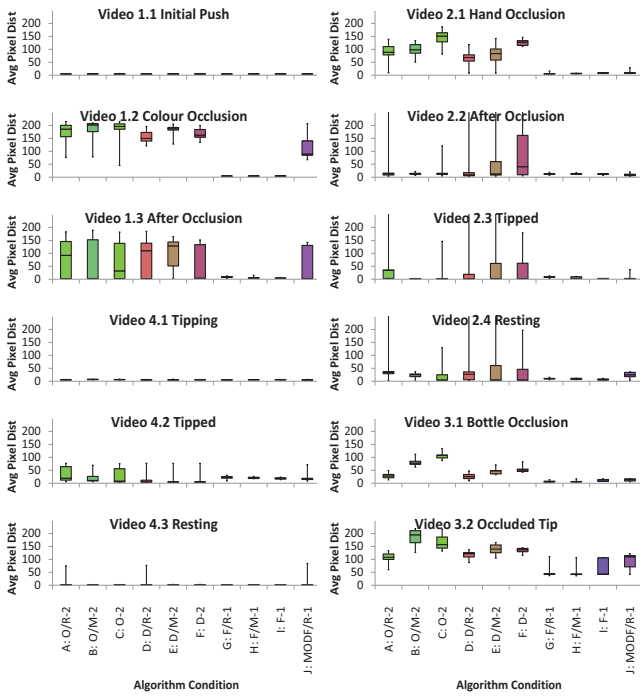

Fig. 6. **Numerical results.** For each of 10 treatments, including 3 control conditions, accuracy of track in image frame with respect to hand-labeled poses for 12 time points from 4 videos, shown using box-whisker plots (plotting 0th, 25th, 50th, 75th and 100th percentiles). Conditions are, from left to right: **A** (O/R-2): Constant pose Gaussian dispersion with rank-based retention of poses, 2 recursions. **B** (O/M-2): The same with mixture model retention. **C** (O-2): The same, no retention. **D** (D/R-2): Simulation + Gaussian dispersion with rank-based retention of poses, 2 recursions. **E** (D/M-2): The same, mixture model retention. **F** (D-2): The same, no retention. **G** (F/R-1): Simulation with force noise, rank-based retention. **H** (F/M-1): The same, mixture model retention. **I** (F-1): The same, no retention. **J** (MODF/R-2): Mixture of all three dynamics models with rank retention, 2 recursions. Accuracy is expressed as average of Euclidean distance between labeled and estimated vertices in image plane (**unit: pixels**), averaged over 40 trials.

It is also found that the use of one particle filter recursion rather than two often tracks better through occlusions for the simulation with Gaussian dispersion noise conditions (results not shown here), but this effect is often canceled by a reduction in effectiveness when the object is visible due to the inability of the filter to sample the likelihood distribution as well.

With respect to the subsequent tipping of the object, there is a small interaction between retention scheme and dynamics model in the Gaussian noise conditions (with and without simulation). However, not losing track at all, as with the force noise condition, leads to the best behaviour.

C. Occluded tipping

In video 3, Fig. 4, the robot finger pushes the target behind a bottle (3.1) and tips it over while it is occluded there (3.2). Again, physics simulation based approaches do better at maintaining track under occlusion with force-noise methods doing best. The tipping is exceedingly difficult to track for conventional methods because of the combination of occlusion and fast movement. However, the force-noise condition is able to track it. The tracker does not actually lock on to the tipped object, however, but some nearby edges on the occluder; it's success is down to, first, the ability of the tracker to track the mostly occluded object starting to tip, and its subsequent tendency to sample physically plausible movements.

D. Tipping

Finally, in video 4, Fig. 5 the robot finger tips (4.1) the target over (4.2). We also investigate the behaviour of the tracker after the object has stayed at rest for six seconds (4.3).

We can see here, that immediately after the tip, simulation-based methods again perform better, and the force noise methods consistently so. The techniques that don't use a physics model are sometimes not able to track this transition but after sampling for several seconds are able to recover track (4.3).

The better performance of both simulation conditions under tipping is attributed to more directed sampling increasing the probability of finding the new mode in the observation likelihood when the object and consequently likelihood mode moves very fast. Since the robot finger is not modeled, the success of this effect does depend on the tracker being able to maintain an unstable pose (i.e. with the object only slightly tipped), since, if only stable poses are tracked, the sampler may never be able to jump over the intervening space of unstable poses into the correct neighbouring stable pose. It is thought that the rank-retention heuristic contributes to this stability, but in any case the noise models used are here demonstrated to track unstable poses. However, much more unstable poses than those explored in these experiments are possible in practice. For example, if an object is lifted by an modeled human hand, it might appear to the simulator to "float" and so tracking it would require the sampler sampling the necessary forces or displacements to keep it floating.

A further difficulty that the use of physical dynamics overcomes in this scenario is that poses away from the previous best pose have a tendency towards a lower observation likelihood because the edge matching algorithm obtains edge surfaces from texture surfaces by reprojecting via that pose.

E. Mixture of dynamics models

The mixture approach (condition J, numerical results in fig. 6), unexpectedly, seems to perform somewhere between all of the other methods across all videos. It is important to remember that the number of recursions of the particle filter in this condition is only one, so that distractors already have a weaker effect. However, even with 2 recursions, this observation remains, on the whole, true. While the success of the physics-based methods is often dependent on the exclusion of physically implausible states, like the simulation with Gaussian noise, the mixture model does sample physically unlikely states; it is merely less likely to.

Beyond this analysis, the effect of different particle retention systems is smaller than the effect of different dynamics models. Indeed, on the particular videos in these experiments there is no clear winner amongst retention methods.

VII. Summary and future work

A. Summary

As hypothesised, improvement is possible in tracking behaviour under occlusion and fast object movement by incorporating simulation into the dynamics sampling process. By restricting dynamics sampling to physically plausible dynamics (force noise conditions) we eliminate false track under occlusion and fast movement, but predict that it will be difficult to track anything that can't be simulated in our closed-world model. Conversely, using physics only to guide sampling allows us to track fast movement better, but remains susceptible to distractors.

B. Future Work

More importantly, this method needs to be tested on a wider variety of scenarios, particularly those that violate the assumptions made here. There are three routes that may be taken to mature this approach and deal with more scenarios so that the benefits of including simulation are not mutually exclusive between either tracking under occlusion or robustness to physical model failure.

1) More realistic handling of simulation: In the first route, the force noise used is incorporated in a way such that each candidate simulation can experience the kind of forces that they may occur in every-day interaction - such as stable or stabilising forces, indeed intentional forces too - rather than the perturbing forces currently used. This route would expand the range of applicability of the more strict simulation style, while still relying on the accuracy of the simulation model. It would not require vast changes and can work well in controlled conditions.

In robotic manipulation scenarios, there is accurate proprioceptive and intentional information regarding the location of manipulators, which would come in useful during the tipping scenario above, for instance. Experiments currently in progress suggest that incorporating efferent information about the finger location produces a small improvement in the cases considered here, but not comparable to the improvement obtained by using force noise. Propagating the velocity may lead to improvements in some scenarios.

2) Change to sampling strategy: In the second route, we would instead handle multiple dynamics models better by allowing the tracker to sample physically implausible dynamics, either by reconstructing trajectories retrospectively in light of the different models, or structuring the filter to maintain different discrete hypotheses better.

With respect to multiple hypothesis maintenance, there are many existing methods for explicitly maintaining multiple hypotheses and targeting sampling (e.g. [16]). With respect to retrospective trajectory reconstruction, it is possible to calculate both physical and observational feasibility scores for potential trajectories from a range of previous timepoints using efficient sampling techniques, similar to the motion estimation problem [7].

3) Learning: An alternative approach is to do away with the fragile simulator and use learning to find a better dynamics model [9]. However, learning does still tend to produce fragile models and the same issues addressed in the current paper need to be dealt with, in particular the problems of generalisation and robustness. Contemporary machine learning techniques are by themselves unfortunately not yet capable of the degree of generalisation necessary to deal with the same range of situations as human-designed simulators, but that is the aim.

VIII. Acknowledgments

We gratefully acknowledge the material support of the EU FP6 IST Cognitive Systems Integrated Project (CoSy), FP6-004250-IP, the EU FP7 IST Project CogX FP7-IST215181, University of Birmingham School of Computer Science and the UK Overseas Research Students Awards Scheme.

References

[1] C. Wren and A. Pentland, "Dynamic models of human motion," in *Proc. IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, Nara, Japan, 1998, pp. 22–27.

[2] M. Vondrak, L. Sigal, and O. C. Jenkins, "Physical simulation for probabilistic motion tracking," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 1–8.

[3] D. Metaxas and D. Terzopoulos, "Shape and nonrigid motion estimation through physics-based synthesis," *IEEE Trans. Pattern Anal. Machine Intell. (PAMI)*, vol. 15, no. 6, pp. 580–591, 1993.

[4] M. Chan, D. Metaxas, and S. Dickinson, "Physics-Based tracking of 3D objects in 2D image sequences," in *Proc. International Conference on Pattern Recognition (CVPR)*, Seattle, 1994, pp. 432–436.

[5] J. Popovic, S. M. Seitz, M. Erdmann, Z. Popovic, and A. Witkin, "Interactive manipulation of rigid body simulations," in *Proc. ACM SIGGRAPH Annual Conference*, New Orleans, 2000, pp. 209–218.

[6] K. S. Bhat, S. M. Seitz, J. Popovic, and P. K. Khosla, "Computing the physical parameters of Rigid-Body motion from video," in *Proc. European Conference on Computer Vision (ECCV) - LNCS*, vol. 2350, Copenhagen, 2002, pp. 551–565.

[7] D. J. Duff, J. Wyatt, and R. Stolkin, "Motion estimation using physical simulation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Alaska, May 2010, pp. 1511–1517.

[8] T. Moerwald, M. Zillich, and M. Vincze, "Edge tracking of textured objects with a recursive particle filter," in *International Conference on Computer Graphics and Vision (GraphiCon)*, Moscow, Russia, Oct. 2009.

[9] T. Mörwald, M. Kopicki, R. Stolkin, J. Wyatt, S. Zurek, M. Zillich, and M. Vincze, "Predicting the unobservable: Visual 3D tracking with a probabilistic motion model," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, 2011, in these proceedings.

[10] M. Isard and A. Blake, "CONDENSATION – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[11] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "Object tracking with an adaptive Color-Based particle filter," in *Proc. of the 24th DAGM Symposium on Pattern Recognition - LNCS*, vol. 2449, Zurich, 2002.

[12] S. Thrun, "Particle filters in robotics," in *Proc. Annual Conference on Uncertainty in AI (UAI)*, Alberta, Canada, 2002.

[13] G. Klein and D. Murray, "Full-3d edge tracking with a particle lter," in *Proc. 17th British Machine Vision Conference (BMVC)*, Edinburgh, 2006.

[14] "PhysX features," May 2010. [Online]. Available: http://developer.nvidia.com/object/physxfeatures.html [Accessed: 2010-05-07 12:13:05]

[15] A. Witkin and D. Baraff, "Physically based modeling: Siggraph 1997 course notes," 1997. [Online]. Available: http://www.cs.cmu.edu/~baraff/sigcourse/ [Accessed: 2010-08-24 14:18:47]

[16] C. Chang, R. Ansari, and A. Khokhar, "Multiple object tracking with kernel particle filter," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, 2005.