Predicting the Unobservable
Visual 3D Tracking with a Probabilistic Motion Model

Abstract—Visual tracking of an object can provide a powerful source of feedback information during complex robotic manipulation operations, especially those in which there may be uncertainty about which new object pose may result from a planned manipulative action. At the same time, robotic manipulation can provide a challenging environment for visual tracking, with occlusions of the object by other objects or by the robot itself, and sudden changes in object pose that may be accompanied by motion blur. Recursive filtering techniques use motion models for predictor-corrector tracking, but the simple models typically used, often fail to adequately predict the complex motions of manipulated objects. We show how statistical machine learning techniques can be used to train sophisticated motion predictors, which incorporate additional information by being conditioned on the planned manipulative action being executed. We then show how these learned predictors can be used to propagate the particles of a particle filter from one predictor-corrector step to the next, enabling a visual tracking algorithm to maintain plausible hypotheses about the location of an object, even during severe occlusion and other difficult conditions. We demonstrate the approach in the context of robotic push manipulation, where a 5-axis robot arm equipped with a rigid finger applies a series of pushes to an object, while it is tracked by a vision algorithm using a single camera.

I. INTRODUCTION

This paper describes a novel approach to visual tracking of an object undergoing pushing manipulation operations, in which a robot arm, equipped with a single rigid finger, applies a series of pushes or pokes to the object, causing it to move from one pose to another. It may seem somewhat esoteric to focus on pushing, however there are good reasons to do so. Pushing is a very fundamental form of manipulation. More complex activities, such as combinations of many simultaneous single-finger actions. More practically, even industrial pick and place operations can be viewed in-hand manipulation with multiple fingers, can be viewed as combinations of many simultaneous single-finger actions. More practically, even industrial pick and place operations with a simple two jawed gripper often result in a pushing phase, where uncertainties in both object and robot pose lead to one jaw contacting the object before the other. This can even lead to gross grasp failures when the object topples over under pushing from the first jaw, before the second jaw makes contact. Therefore it is important to solve the problems of robotic pushing, and controlled pushing will typically be reliant on tracking the object pose with a vision system.

II. RELATED WORK

There is a limited body of literature describing vision algorithms tailored specifically for robotic manipulation tasks. For example, Drummond and Cipolla [1] incorporate knowledge of kinematic constraints into tracking, to better track articulated chains of rigid bodies, with a view to tracking robotic arms for visual servoing. However, it is more usual for researchers to simply take a generic tracking algorithm and incorporate it with an existing manipulation planning system, e.g. [2]. Typically the vision algorithms are drawn from the model based tracking literature, for example [3–5], which predominantly track by choosing candidate poses of the tracked body, whose projected wire-frame edges best match edges extracted from images. More recently, the ability to make use of advanced graphics cards for high speed projective calculations, means that such techniques can be applied to tracking with robust particle filters, e.g. [6].

Particle filters rely on motion models, to propagate particles from one predictor-corrector step to the next. In practice, little may be known about the motion of the tracked object, and so predominantly these motion models must be very blunt instruments. It is typical to simply apply Gaussian noise to particles to propagate them, assuming no real understanding of how the object might move at the next time step. However, if the tracked object is an object, subject to robotic manipulation, we should be able to make use of our knowledge of the planned manipulative action, to make a much more informed prediction of the next phase of the objects motion. In the case of an object which is rigidly held in the jaws of a hand or gripper, the motion prediction problem becomes trivial since the object is exactly constrained to follow the motion of the manipulating arm. However, in pushing manipulation, the motion of an object which will result from an applied single-finger push or poke is much more uncertain.

Early approaches to predicting the effects of robotic pushes on object motion, [7–9], attempted analytical solutions of physical constraints. These approaches did not progress beyond anything more complex than the simple 2D case, with flat polygonal objects, constrained to slide on a planar surface. More recently, Cappelleri [10] used physics simulation software to plan manipulative pushes, but again...
This was limited to a 2D problem, with a small, flat rectangular object which was constrained to slide while floating on a film of oil to simplify frictional interactions. We know of little in the way of literature which specifically addresses the prediction problem in robotic push manipulations of real 3D objects, which are subject to complex 6-dof motions such as tipping and toppling. It is possible to use physics simulators to predict the motions of interacting rigid bodies, however this approach is reliant on explicit knowledge of the objects, the environment and key physical parameters which can be surprisingly difficult to effectively tune in practice. \[11\]. Furthermore, once a physics simulator has been set up for a particular scenario, it is not generalizable to new objects or novel situations.

In contrast, our recent work, \[12\] proposes a system which can learn to predict the explicit 3D rigid body transformations that will result when an object in an arbitrary orientation is subjected to an arbitrary push. The system does not make use of any physics simulation, or any hard coding of Newtonian physics equations or physical constraints. Instead, a statistical relationship between applied pushes and resulting object motions is trained, by simply having the robot apply a series of random pushes to the object, proprioceptively recording the finger trajectories, and observing the resulting object motions with a vision system.

In this paper, we provide an overview of the "learning-to-predict" architecture, and then show how it can be conveniently incorporated into a particle filter-based vision algorithm to propagate particles from one frame to the next. We demonstrate the effectiveness of the technique, for tracking pushed objects past large occlusions and other difficult circumstances, where attempting vision without adequate prediction would fail.

### III. Overview

The paper proceeds as follows. Section \[IV\] provides an overview of our system for learning to predict the outcomes of manipulative pushes. We describe how the motions of rigid bodies are represented by coordinate frames and transformations. We show how objects and their motions can be decomposed and how a variety of probabilistic experts can be trained to predict various aspects of these motions. We show how to combine the opinions of these experts as a product of densities, which is capable of significant generalization to new objects with different shapes and different push directions which have not been encountered during training. In Section \[V\] we describe our algorithm \[13\] for visual tracking of 3D objects using edges, colour and texture features. We then show how tracking can be improved by incorporating a well trained predictor as described in Section \[IV\]. Section \[VI\] presents results of this work, providing examples of how the enhanced tracker copes with difficult situations such as occlusion and motion blur. Section \[VII\] summarizes the results, and discusses ongoing and future work.

### IV. Prediction

This section is just a brief overview of the work in \[12, 14\] to show how rigid body movement can be described in a probabilistic form.

![Fig. 1. A system consisting of three interacting bodies with frames A and B and some constant environment with Frame O.](image)

A system consisting of three interacting rigid bodies can be described by coordinate frames \(A, B\) and \(O\) and by six transformations between the bodies and different time steps \((t-1, t\) and \(t + 1)\), with respect to a constant environment \(O\) as shown in Figure 1. \(A\) and \(B\) change in time and are observed at discrete time steps \(t, t+1\), every non-zero \(\Delta t\). As stated in \[12\], a triple of transformations \(T_{A_t , O} , T_{A_{t-1} , A_t} \) and \(T_{A_t , A_{t+1}}\) provide a complete description of a state of a rigid body \(A\) in terms of classical mechanics. Of course the same is true for some body \(B\). The prediction problem can be stated as: given we know or observe the starting states and the motion of the pusher, \(T_{A_t , A_{t+1}}\), predict the resulting motion of the object, \(T_{B_t , B_{t+1}}\). This is a problem of finding a function:

\[
f : T_{A_t , B_t} , T_{B_t , O} , T_{A_{t-1} , A_t} , T_{B_{t-1} , B_t} , T_{A_t , A_{t+1}} \rightarrow T_{B_t , B_{t+1}} \tag{1}
\]

In many robotic applications manipulations are slow, so we can assume quasi-static conditions and it is often possible to ignore all frames at time \(t - 1\). This conveniently reduces the dimensionality of the problem, giving:

\[
f : T_{A_t , B_t} , T_{B_t , O} , T_{A_{t+1} , A_{t+1}} \rightarrow T_{B_t , B_{t+1}} \tag{2}
\]

Prediction learning using Functions \(1\) or \(2\) is limited with respect to changes in shape (see Chapter 5.3 of \[14\]). The problem can be expressed by a product of several probability densities over the rigid body transformation, encoding global as well as local contact configurations. Figure 2 shows the frames representing two different experts \[13\].

\[
p_{\text{global}}(T_{B_t , B_{t+1}} | T_{A_{t+1} , A_{t+1}} , T_{A_t , B_t} , T_{B_t , O}) \tag{3}
\]

\[
p_{\text{local}}(T_{B_t , B_{t+1}} | T_{A_t , A_{t+1}} , T_{A_t , B_t} ) \tag{4}
\]
In addition to learning how an object moves in response to a push, it is desirable if we can also incorporate learned information about the inherent tendencies of parts of an object to move in various directions with respect to the environment or any other objects, but regardless of whether it is being pushed or not. This additional information may help when predicting the motions of previously unseen objects, because it provides some prior knowledge about what kinds of motions are possible and which are not.

The subsequent motion of the object in the inertial frame can now be described as:

\[
p(T_{Bi}, B_{i+1} | K) = p_{\text{local}}(T_{Bi}, B_{i+1} | T_{Ai}, A_{i+1}, T_{Ai}', B_{i}) \times p_{\text{global}}(T_{Bi}, B_{i+1} | T_{Ai}, A_{i+1}, T_{Ai}', B_{i}, T_{Bi}, O) \tag{4}
\]

where \( K \) stands for the known conditions \( T_{Ai}, A_{i+1}, T_{Ai}', B_{i}, T_{Ai}, B_{i}, O \). The density product (4) is maximised using the differential evolution optimization algorithm [15].

Now the prediction system can be trained by training each of the density terms of equation (4) on data derived simply by observing the outcomes of random robotic pushes, and extracting the resulting object motions with the tracker presented in the next Section.

V. Tracking

Visual observation of the trajectory of the object is the problem of finding the pose \( T_{Bi}, O \) given an image \( I \) where the camera is assumed to be fixed to the global environment \( O \). We are proposing a probabilistic framework and therefore searching for the conditional probability distribution \( p(T_{Bi}, O | Z) \). Estimation is essentially based on Bayes’ theorem:

\[
p(T_{Bi}, O | Z) = \frac{p(Z | T_{Bi}, O) p(T_{Bi}, O)}{p(Z)} \tag{5}
\]

where \( Z \) is a set of observations. \( p(Z) \) and \( p(T_{Bi}, O) \) are assumed to be uniformly distributed, which means that the unconditioned probability of any observation or pose is the same, and thus equation (5) simplifies to

\[
p(T_{Bi}, O | Z) \propto p(Z | T_{Bi}, O) \tag{6}
\]

For estimating this distribution we are using the methods proposed in [13] [16]. The basic idea is to project a texture based representation of the object into the current camera image and to maximise the match using a particle filter.

A. Image processing

We project the geometric model of the object (described by vertices, faces and textures) into the image space using the transformation \( T_{Bi}, O \) and standard techniques of computer graphics such as perspective transformation and texture mapping. In image space we compute the edge gradients of the model \( \mathbf{g}_M \) and of the image captured by the camera \( \mathbf{g}_I \).

B. Matching

Now it is possible to find a measure for the match, or weight \( w \) of a pose \( T_{Bi}, O \). For each point \((u,v)\) on the model \( M \) in image space we can compute the deviation of the edge gradients by superimposing the projected model over the image. The sum of the difference of the gradients is computed as:

\[
w_i \propto \frac{1}{e_i} \int_{(u,v) \in M} \left| \mathbf{g}_M (u,v) - \mathbf{g}_I (u,v) \right| \tag{7}
\]

A second approach for matching is similar to (7), with respect to pixel-wise comparison of the model and the image. But instead of computing the difference of gradients, the difference of the colour with respect to the hue in HSV (Hue, Saturation, Value) colour space is used.

\[
w_i \propto \frac{1}{M} \int_{(u,v) \in M} \left| h_M (u,v) - h_I (u,v) \right| \tag{8}
\]

The advantage of using a colour based tracker is increased robustness against edge based clutter as in Figure 8. Of
course it is less robust against changing lighting but the combination of both kinds of cues can significantly improve the overall performance. How to combine the two methods in an optimal way is an open issue and remains as future work.

C. Particle filter

A particle filter, such as the SIR (Sequential Importance Resampling) as explained in [17], estimates the current state $x_{t+1}$ based on the previous state $x_{t}$ and the current observation $z_{t+1}$, where the system state is perturbed by system noise $n_{t+1}$ and the observation by observation noise $v_{t+1}$. The particle filter maintains a probability distribution over the system state represented as a set of samples.

$$x_{t+1} = f(x_{t}, u_{t}) + n_{t+1}$$
$$z_{t+1} = h(x_{t+1}) + v_{t+1}$$

In our case the system state to be estimated is the transformation $T^{B_{t+1},O}$. $f(x_{t}, u_{t})$ corresponds to function (2) where $u_{t} \equiv K$. In the case of not using prediction at all equation (9) simplifies to

$$x_{t+1} = x_{t} + n_{t+1}$$
$$z_{t+1} = h(x_{t+1}) + v_{t+1}$$

i.e. $f(.)$ reduces to a static motion model.

The observation is given by equation (7) and (8) respectively, i.e. the probability of the observation of a sample $i$ is given as

$$p(z_{i}|T^{B_{t+1},O}) \propto \exp (w_{i})$$

Note that the term sample is equivalent to particle, pose and transformation with respect to $T^{B_{t+1},O}$.

System noise $n$ is given as

$$N(0, \sigma)$$
$$\sigma = (1 - w_{i})\sigma_{0}$$

Given the requirements for tracking accuracy and speed for a typical table top scenario we chose a basic standard deviation $\sigma_{0}$ of 0.03 m for the translational and 0.5 rad for the rotational degrees of freedom. The standard deviation is then scaled for each particle individually based on the particle’s weight. This means that as the weight of a particle increases its noise level decreases, leading to faster convergence.

L.e. the noise level follows the system’s current confidence. Observation noise $v$ is modeled in a similar manner as in equation (10) with a $\sigma_{0}$ of 4 pixels.

The particle filter always tries to find the local maximum in observation space. In the case of occlusion, as shown in Figure 6 and 8, this leads to drifting of the tracker away from the real pose of the object. To cope with this problem we introduce a threshold $c_{th}$ for the confidence value $c = \text{mean}(w_{i})$ setting the system noise level such that

$$n = N(0, \sigma) \quad \text{if} \quad c > c_{th}$$
$$n = 0 \quad \text{if} \quad c \leq c_{th}$$

This means that below a certain confidence $c \leq c_{th}$ the tracker relies completely on the output of the predictor. We currently set this threshold to an empirically determined value between 0.3 and 0.5.

Figure 3 gives an overview of our method. As proposed in [13] and [16], we use iterative particle filtering for better computational performance and accuracy of the tracker. The initialisation of the pose is implemented very similar to the algorithm described in Section 4.2 in [18].

VI. Results

In all our experiments we are using the tracking system as described in Section IV and compare it against the tracker without prediction which is the same system but without the motion prediction step as in Figure 3. Other than motion prediction we are using the same configuration for each of the tracker, respectively non-iterative particle filtering with 100 particles for each frame. The following experiments are designed to illustrate the differences in performance between tracking with and without incorporating a learned prediction system. The poses are drawn as wireframe models with the following colour-code:

- **White**: Ground truth. This is not actually ground truth, but it is the best estimate of ground truth that we can generate, by running the tracker slowly with many iterations and many particles (4 and 200 respectively).
- **Red/Magenta**: Tracking with prediction as proposed in this paper.
- **Blue**: Pure motion prediction without visual feedback by the tracker.

A. Experiment - Accuracy

In this experiment we show how accuracy increases using the proposed methods. We compare the poses of tracking with and without prediction against ground truth. To evaluate the error in both cases we used the non-normalized error measure described in Section IV.B of [19].

![Fig. 4. Overlaid tracking results with and without prediction](image-url)
For a fair comparison we only used pushing examples, where also the tracker without prediction was able to maintain tracking throughout the whole sequence, as shown in Figure 4. In the upper-left image both trackers are initialised at the same pose. The upper-right and lower-left show the mismatch of the tracker without prediction (red) while the object is moving. At the end of the sequence, lower-right image, both trackers converge to the ground-truth pose as to be expected.

Figure 5 shows the result of the evaluation of 20 pushes. For both, average error and standard deviation, the tracker which takes advantage of information from the predictor performs significantly better.

The following experiments show robustness to various events in a qualitative manner. For these cases a quantitative evaluation against the tracker without prediction is meaningless, as the latter loses the object at some point altogether.

**B. Experiment - Occlusion**

Typically visual tracking algorithms fail when the object is partially or completely occluded. The prediction model of Section IV allows us to overcome such situations. Figure 6 shows a pushing sequence which suffers from heavy occlusion. At the beginning both of the trackers perform very well, since at least parts of the object are visible (top row). At the point where the occluder is completely hiding the object, the tracker without predictor (magenta wireframe) is drifting away to a visually more likely position (e.g. it is attracted to the robot hand which introduces clutter with respect to edges as well as colour) and fails to keep track of the object pose.

Pure prediction (blue wireframe) does not use any visual feedback and produces the whole trajectory from the initial pose. The finger is pushing very near to the centre of the object which is a very unstable position. Given this push, the object in some cases might slide to the left or to the right, or topple over depending on slightly different initial positions. However, since the predictor used for tracking gets updated by the visual observation it is possible to handle such difficult situations.

**C. Experiment - Motion blur**

Another example of a difficult situation is fast movement of the object relative to the camera. Figure 7 illustrates such a case, where a box is pushed forward causing it to tilt until it reaches an unstable pose and finally toppling over. This is a very critical situation for visual observation. The falling object moves quite fast, causing the image to blur.

Again the tracker with (green) and without prediction (red), and the pure predictor (blue) are initialised at the same starting pose. During the first phase of the sequence the predictor proposes an erroneous rotation of the object, while the vision system extracts the correct pose relatively accurately (top row of Figure 7). However, by the time of the unstable pose shown in the lower-left image the tracker with prediction is already better than the tracker without prediction. The object is moving fast during the next frames causing the effects mentioned above. The tracker without prediction can not follow the fast movement, loses track and gets trapped in a local maximum. The predictor on the other hand proposes the right pose and the corresponding tracker refines the result.

**D. Experiment - Occlusion with motion blur**

The hardest case for a visual observation system is the combination of fast movement and occlusion. We tested this case by applying a pushing manipulation where the object is hidden behind an occluder where it topples over, as shown in Figure 8.

In the top-left image enough parts of the object are visible and both of the trackers produce good results. The top-right image shows the object behind the occluder already in the phase of falling down as the blur suggests. The lower-left is the subsequent frame and illustrates the large change of the pose, which causes the tracker without prediction to fail, whereas the tracker with prediction overcomes this difficult
The pose of the pure predictor is also very close to the real one, but suffers from integrating error over the trajectory.

Note that for this experiment we placed a virtual occluding object in the scene. This allowed us to vary the size and texture of the occluder and most importantly to position it right in front of the toppling object.

VII. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this work we demonstrated how objects can be tracked in 3D under visually challenging situations such as occlusion, motion blur and fast movement. We summarized the main ideas of probabilistic prediction and explained edge and colour based 3D tracking using a Monte Carlo particle filter. We show how the use of probabilistic prediction as motion model for the tracker leads to clearly improved accuracy as well as robustness. There are cases where erroneous prediction can degrade tracking performance (e.g. in unstable contact configurations), but these are outweighed by the majority of cases where prediction is correct and especially for occlusion.

B. Future Work

Although the robustness of the tracker is already improved, there are several points which remain open.

First of all the confidence of a visual observation is not very distinctive and it is very hard to find a good measure indicating whether the pose it suggests is wrong or not. As described in Section VI we manually set a threshold. Obviously this is not the optimal solution since this threshold depends on the visual model of the object and the clutter in the environment, especially introduced by the occluder. I.e. for an edge based tracker it is very hard to tell if it is correct or not in case of edge rich clutter as shown in Figure 8, whereas the colour based tracker is strongly influenced by objects which are coloured similar to the object, with respect to equation (8).

Although the predictor is trained using visual tracking, this still happens in an offline stage for now. For an automated system it is desirable to learn the predictor online. This leads to a chicken-and-egg problem. Since at the beginning the predictor is not trained, the tracker produces bad results for challenging situations, which can thus not be learned by the predictor. As mentioned above, the tracker is not able to reliably measure the correctness of the pose suggested and the predictor will thus learn those wrong poses.

Furthermore, at the moment the predictor provides only the most likely pose in equation (4) to the tracker. This means that we do not make full use yet of the probabilistic framework inside the predictor.

VIII. ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

REFERENCES