Kinematically Optimised Predictions of Object Motion

Dominik Belter*†, Marek Kopicki†, Sebastian Zurek†, Jeremy Wyatt†
*Institute of Control and Information Engineering, Poznan University of Technology, Poland
†School of Computer Science, University of Birmingham, UK

***Abstract*— Predicting the motions of rigid objects under contacts is a necessary precursor to planning of robot manipulation of objects. On the one hand physics based rigid body simulations are used, and on the other learning approaches are being developed. The advantage of physics simulations is that because they explicitly perform collision checking they respect kinematic constraints, producing physically plausible predictions. The advantage of learning approaches is that they can capture the effects on motion of unobservable parameters such as mass distribution, and frictional coefficients, thus producing more accurate predicted trajectories. This paper shows how to bring together the advantages of both approaches to achieve learned simulators of specific objects that outperform previous learning approaches. Our approach employs a fast simplified collision checker and a learning method. The learner predicts trajectories for the object. These are optimised post prediction to minimise interpenetrations according to the collision checker. In addition we show that cleaning the training data prior to learning can also improve performance. Combining both approaches results in consistently strong prediction performance. The new simulator outperforms previous learning based approaches on a single contact push manipulation prediction task. We also present results showing that the method works for multi-contact manipulation, for which rigid body simulators are notoriously unstable.**

I. Introduction

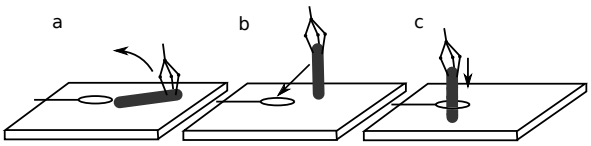

Fig. 1. Application scenario: a task which requires the model of the object and hand-object interaction

Planning of robot motion is a key topic in robotics. A particularly hard set of planning problems arise in robot manipulation of objects. One of the main requirements for manipulation planning is the existence of good simulators since in order to plan, a robot must first be able to predict the effects of its actions. In motion planning of the robot itself this is possible because the robot itself is a well modeled system, both kinematically and dynamically, and controllers are carefully tuned to ensure this. The objects the robot manipulates however, are inevitably not well modeled.

Imagine the manipulation problem presented in Fig. 1. The robot has to find the grasp and motion trajectory which causes the desired behaviour of the object. Efficient task execution requires knowledge about dynamic properties of the object. Parameters determining object behaviour such as frictional coefficients, mass and mass distribution are unknown and not directly observable. Moreover in real life these parameters vary between instances of the same object. Our goal is to create a method which enables learning the dynamics of the object during interaction with a robotic hand. We avoid incorporating explicit physics knowledge about dynamics into the system. Instead we apply a data driven methodology.

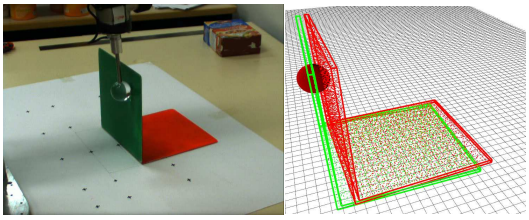

Fig. 2. Kinematicaly implausible output from the tracking system (red polyflap) and output from the predictor (green polyflap)

In this paper we will restrict ourselves to the problems of predicting the behaviours of rigid bodies. Rigid body physics engines, such as ODE, Bullet and PhysX, are good at making predictions that are physically plausible, although even this may require extensive hand tuning. Instead in robotics we need simulators that are not only plausible, but accurate. By accurate we simply mean that they produce predictions close to the behaviour of the real object being manipulated. To improve the accuracy of such rigid body simulators we can tune their parameters so that predicted behaviour matches real behaviour. As shown by [1] even this leaves predictions some way off reality.

A different approach is to use a machine learning method to learn to predict object behaviour from recorded trajectories of objects under manipulation [1]. This requires that a separate model is learned for each object. The predictions are more accurate in terms of matching the trajectory of the real object than those of physics simulation, and can be generalised to new actions and shapes. But they can also produce predicted trajectories that are physically implausible, such as predicting interpenetrations between rigid objects (Fig. 2). This is because learning approaches suffer significantly from even small amounts of noise in the training data (object tracks).

The main contribution of this paper is to show how to combine the best properties of rigid body engines with the best properties of learners. We present two new methods. In our first approach we show how a learned predictor's

predictions can be improved by optimisation post prediction. The optimisation uses a simplified collision checker to detect interpenetrations in predictions. An optimisation method minimises a cost function that penalises interpenetrations, and also penalises deviations from the predicted trajectory. We show empirically that this not only results in predicted trajectories that are more plausible, but also in trajectories that are more accurate. In our second method we show how the same optimisation procedure can be used to clean training data prior to learning. We show that when combined with the post prediction optimisation this leads to good generalisation performance with respect to actions, and accurate and physically plausible predictions. Moreover we demonstrate empirically that this approach can be extended to previously unmodelled objects, and to multi-contact manipulations. By acquiring a point cloud model of the object to be manipulated on the fly the method is not restricted to cases where there is already an accurate shape model.

II. Related work

Physics engines are used to predict the motions of interacting rigid bodies. But some predictions may not be possible due to inherent limitations of the model employed, for example when modelling friction [2]. In addition while physics engines have been used for prediction for control [3] or tasks such as visual tracking [4], [5] they must be carefully tuned to the particular objects used, and so lose some of their generality. Some machine learning approaches can classify object classes, e.g. rolling versus non-rolling objects [6], [7], or liftable versus non-liftable objects [8]. The predictions in those methods are not, however, generalisable to new objects, pose or push direction, and explicit 6-DoF rigid body motions are not predicted. Stoytchev [9] showed learning of the relationship between the robot's action with a particular tool, and the resulting 2D motions of puck-like discs sliding across a 2D table-top, under four discrete possible actions (pushing away from or towards the robot, and in left and right directions). Work has also been done by [10] on using physics simulators together with logical rules to support naive physics predictions.

The goal there is not to provide accurate predictions of the precise motion, but to assess the possible qualitative outcomes (e.g. stability of the grasp). Finally work on learning generalisable predictions of body motion has been carried out by [11], [1]. This work uses a similar core predictor as here, but tries to learn the kinematic constraints. These are encoded using a small number of experts attached to object parts, which then contribute to a product of densities model. The advantage of that approach compared to that here is its potential flexibility, but the method fails to prevent implausible predictions in many cases, and is affected by noise in the training data. Combining learned outputs with a predictor based on the principle of virtual work prevents this [12], but is computationally more expensive, evaluating the work done by complete trajectories.

Most recently, a method for kinematically plausible online object tracking was presented by Chalon et al. [13]. A kinematicaly plausible object pose is found using a particle filter. In our work we solve the same problem but we show how to gain from object's pose optimization strategy during tracking as well as during the prediction stage. In addition ours is an approach for learning to predict.

III. Motion predictor

In our previous work [1] we showed how to learn to predict the motions of interacting rigid bodies. The behavior of an object which is in contact with other objects is significantly different than the behaviour of an object which doesn't collide. Moreover, the behaviour at the point at which the object starts to collide is discontinuous and nonlinear. This causes difficulties for regression-based learning methods. We avoid this situation by learning two separate models and switching between them. A free motion model is created when the object doesn't collide or collides with the ground plane. A collision model is created for situations when the object collides with one or more of the robot fingers. During prediction we detect the number of collisions and then we apply proper model.

Collision detection and model switching are controlled by a simple collision checking procedure. If the object is given by a mesh we create a grid of points located on the object's surface, where the points' locations are additionally perturbed by random noise (Fig. 2). We found it more computationally effective than using a regular mesh of points. With randomly perturbed grid of points we observe less inter-body penetrations with the smaller number of points. When available, we directly use the point cloud model of the object obtained from the robot's sensory system. To detect collision a sphere-to-point collision checking procedure is employed. In order to detect collisions, we compute the distance between the centre of the sphere and the considered point. The procedure is fast and allows us to check collisions multiple times during the optimisation procedure without a significant increase in computation cost.

A. Learning a predictive model

The prediction problem is formulated here as a regression problem. We learn the relation between a set of input variables and the object motion at the output. We use the current state of the object and robotic hand as an input. We also use the previous state of the system to encode the object dynamics. We don't use acceleration as an input because it's noisy and it increases the dimensionality of the problem. We also don't incorporate explicitly friction from the surface, friction distribution of the surface, friction between the interacting objects and contact normal forces in the model. These properties are hidden in the model and obtained by learning.

The goal is to predict the pose change of the object $T^{O_t,O_{t+1}}$ using information about the desired motion of the finger/fingers $T^{F_t,F_{t+1}}$. The $T^{X_t,X_{t+1}}$ is the homogeneous transformation from coordinate system X_t at time t to X_{t+1} coordinate system at time $t+1$. We repeat the procedure sequentially to predict the whole trajectory of the object.

The prediction of the object pose change is a problem of finding a function

$$F_{\text{free}} : O_t^{\text{rot,z}}, T^{O_{t-1},O_t} \longrightarrow \quad T^{O_t,O_{t+1}} \tag{1}$$

for a free motion model, where $O_t^{\text{rot,z}}$ is the orientation of the object (set of quaternions) and distance to the ground (z coordinate) and T^{O_{t-1},O_t} is the pose change of the object. We use the limited representation of the object's pose (orientation and distance to the ground) due to the fact that behaviour of the objects does not depend on its horizontal position (x,y).

For a collision model we have to find a function:

$$F_{\text{coll}} : O_t^{\text{rot}}, T^{F_t^{i=1},O_t}, ..., T^{F_t^{i=N},O_t}, T^{G_t,O_t}, T^{O_{t-1},O_t}, \\ T^{F_{t-1}^{i=1},F_t^{i=1}}, ..., T^{F_{t-1}^{i=N},F_t^{i=N}} \longrightarrow \quad T^{O_t,O_{t+1}}, \tag{2}$$

where $T^{F_t^i,O_t}$ is the transformation between the i-th finger and the object, $T^{F_{t-1}^i,F_t^i}$ is the pose change of the i-th finger, N is the number of fingers and G_t is the coordinate system located on the ground surface below the object's coordinate system.

Functions F_{free} and F_{coll} should be able to describe interactions between the object O and robotic finger F. They encode unobservable physical properties of the object (frictional coefficient, centre of mass) as well as the shape of the object. Furthermore, they can be learned from observations of the real trajectories registered when the robot manipulate the object.

To create the input to the predictor we parametrize each transformation. Each homogeneous transformation $T^{X,Y}$ (4×4 matrix) is represented by 7-dimensional vector $\overrightarrow{T}^{X,Y}$ which contains four quaternions and euclidean coordinates x, y and z.

In our approach we recast (1) and (2) as a conditional probability density (CPD) p_{free} and p_{coll} over possible object motions $T^{O_t,O_{t+1}}$:

$$p_{\text{free}}(T^{O_t,O_{t+1}} | O_t^{\text{rot,z}}, T^{O_{t-1},O_t}), \tag{3}$$

$$p_{\text{coll}}(T^{O_t,O_{t+1}} | O_t^{\text{rot}}, T^{F_t^1,O_t}, ..., T^{F_t^N,O_t}, T^{G_t,O_t}, \\ T^{F_{t-1}^1,F_t^1}, ..., T^{F_{t-1}^N,F_t^N}, T^{O_{t-1},O_t}). \tag{4}$$

To find the conditional probability density we estimate the joint density $p(T^{\text{out}}, T^{\text{in}})$, and marginal density $p(T^{\text{in}})$ using (5), and (6) [14]:

$$p(T^{\text{in}}, T^{\text{out}}) = \frac{1}{n} \sum_{i=1}^{n} c_i K_{h_2}(T^{\text{in}} - T_i^{\text{in}}) K_{h_1}(T^{\text{out}} - T_i^{\text{out}}), \tag{5}$$

$$p(T^{\text{in}}) = \frac{1}{n} \sum_{i=1}^{n} c_i K_{h_2}(T^{\text{in}} - T_i^{\text{in}}), \tag{6}$$

where T_i^{in} and T_i^{out} are input and output vectors of the training set, respectively. K_{h_1} and K_{h_2} are Gaussian kernels, c_i is normalisation factor (note that we don't have to normalise the probability because we use the obtained probabilities to find extreme not the probability value) and n is the number of samples.

The conditional density function is computed as follows:

$$p(T^{\text{out}} | T^{\text{in}}) = \frac{p(T^{\text{in}}, T^{\text{out}})}{p(T^{\text{in}})}. \tag{7}$$

To find the predicted motion T^{out} which corresponds to the given value of the input T^{in} we solve the optimisation problem:

$$p = \arg\max_{T^{\text{out}}} p(T^{\text{out}} | T^{\text{in}}) \tag{8}$$

For this purpose we use Particle Swarm Optimization (PSO) [15]. We found it slower but more robust than the mean-shift algorithm which we compared it to.

IV. Kinematic optimization

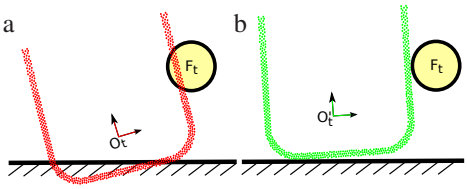

Fig. 3. Pose of the object at time t before optimization O_t (a) and the feasible pose after optimization O'_t (b). The transformation from O_t to O'_t is T^{O_t,O'_t}. Pose of the robot's finger is F_t

The object tracking system as well as the predictor produce erroneous results in terms of kinematic feasibility. Because of measurement noise or prediction inaccuracies the finger penetrates the object (Fig. 2) or sinks into the ground. The predictor also mimics this behavior when it's learned using noisy data. During the prediction stage the error accumulates and implausible behaviours occur, e.g. the finger might go through the object.

The goal of kinematic optimisation is to find a feasible pose of the object which satisfies all kinematic constraints (minimises interpenetrability) and is as close as possible to the current pose of the object. We are looking for a transformation T^{O_t,O'_t} which moves the object from the current estimated pose O_t at time t to a new kinematically feasible pose O'_t (Fig. 3). The objective function is:

$$E = \sum_{i=1}^{N_k} c_1 e^{(-\frac{d_i}{2r_i})} + \sum_{n=1}^{7} c_n ln(1 + \overrightarrow{T}_{N,n} - \overrightarrow{T}_n^{O_t,O'_t}) + \\ + O'_t(z) + \sum_{m=1}^{7} c_m | \overrightarrow{T}_m^{O_t,O'_t} - \overrightarrow{T}_m^{pred} | \tag{9}$$

where N_k is the number of interpenetrating points, d_i is a distance between the i-th point and the centre of the finger or ground (the ground is represented as a sphere with large radius to be consistent with finger-to-object collision checking), r_i is a radius of the finger/ground, $\overrightarrow{T}_n^{O_t,O'_t}$ is the n-th

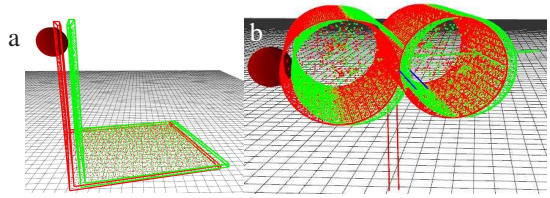

Fig. 4. Experiment: pose of the object before optimisation (red colour represents output from the tracking system or output from the predictor) and after optimisation (green) for polyflap (a) and double cylinder (b)

element of the vector representing transformation, $\overrightarrow{T}_N$ is a neutral transformation (vector – parametrised homogenous identity matrix) which doesn't move the object, $O'_t(z)$ is a distance from the new pose of the object to the ground, T^{pred} is an output from the predictor (predicted pose change of the object) or transformation between previous and current pose of the object from tracking system and c_1, c_n, c_m are constants.

The first element of (9) is the penalty for the finger penetrating the object. For each point on the object's surface penetrating the finger we compute the value of the Gaussian function. It forces the optimisation procedure to take the object out of the collision area. The second term is the penalty for moving the object away from the current pose (inertia equivalent). We are looking for a solution which is as close as possible to the current pose of the object. Third we penalise solutions which are against gravity. We don't use this element of the (9) for multi-finger contact. In this case the preference for motion convergent with the gravity vector means that the robot loses the object. The last element prefers motions which are in similar direction to that given by the predictor or tracking system.

To find the optimal value of (9) we use the PSO algorithm. Example results from the optimization procedure are presented in Fig. 4. The kinematically implausible poses of the objects (red polyflap in Fig. 4a and red double cylinder in Fig. 4b) are moved to new penetration-free poses (green objects).

V. Results

For real experiments, we used a 5-axis Katana robotic manipulator [16] equipped with a single rigid finger and a Kuka arm equipped with DLR-Hit II hand. The tracking system uses a single camera and a visual tracking algorithm [17] for pushing experiments. For experiments with the DLR hand and multi-contact prediction we use two Kinect sensors. The first Kinect is mounted on the robotic arm. We use it to acquire a point cloud reconstruction of the scene from various positions. The obtained point clouds are used to extract the point cloud model of the manipulated object. The second external Kinect is used to track the object during the experiment using this obtained point cloud object model. In this second experiment the tracking method instead tries to best fit the previously obtained model of the object to the scene using Generalized ICP algorithm [18].

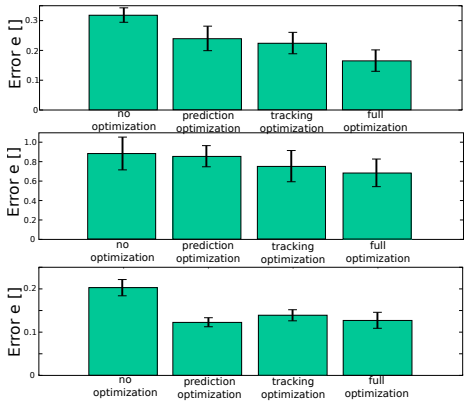

Fig. 5. Prediction error in verification experiments. Top row-double cylinder. Middle row-cylinder. Bottom row-polyflap.

During experiments the robot performs a random pushing movements towards the object or performs a reach to grasp of the object followed by a pick up action. The tracking systems capture object pose every 1/30th of a second. For pushing experiments we stored a set of trajectories, 50 of them are used for learning and a further 10 are used as a test set for verification of the prediction method. For the experiment with the DLR hand we use data from only 11 experiments to predict next (12th) trajectory of the object. Note that during the prediction stage we don't use additional feedback from the tracking system to refine the predictions, i.e. there is no filtering, so we are testing the power of pure prediction. To achieve this the prediction each step is fed back to produce the prediction for the following step, for more than 100 iterations to reach the end of the sequence. In the case of the pushing experiment the system predicts the trajectory for 11 seconds and for the grasping experiment the predicted trajectory is 15 seconds into the future. The error for the predicted N-step trajectory on a test case is:

$$e = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=n}^{7}|\overrightarrow{O^{\text{ref}}_{t,n}} - \overrightarrow{O_{t,n}}| \tag{10}$$

where: $\overrightarrow{O^{\text{ref}}_{t,n}}$ and $\overrightarrow{O_{t,n}}$ are the position vectors (pose and orientation) of the object at time t from the tracking system and the predicted pose of the object (7D vector), respectively. Note that we compare prediction results to the trajectory obtained from the tracking system without using optimization method proposed in the paper (this may be a trajectory which is kinetically implausible). Nevertheless this represents the best raw estimate of object pose through the entirety of any test trajectory, and thus is the best ground truth obtainable without resort to another sensing method. Critically it therefore allows us to fairly distinguish the effect of different optimisation methods on prediction accuracy.

The experiments were performed using three types of objects: a double cylinder (Fig. 6), a cylinder (Fig. 7) and an L shaped polyflap (Fig. 8). For each object we performed four combinations of methods. First, we used a Kernel

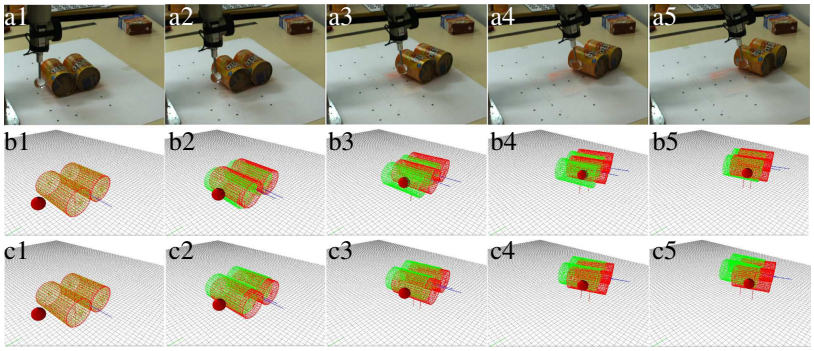

Fig. 6. Pushing experiment (a): prediction results at 0, 80, 160, 240 and 320th frame for double cylinder using the KDE predictor without optimisation (b) and with the proposed optimisation method applied during both the tracking and prediction stages (c)

Density Estimation (KDE) predictor (3) (4). In the second condition we added the optimisation procedure (9) post the prediction stage. The third condition tested the effect of the optimisation procedure when used solely to kinematically clean the tracking data prior to learning. The last condition used the optimisation procedure (9) in both the tracking and prediction stages. The results are presented in Fig. 5. A two-tailed t-test was used to determine whether there were significant differences between the treatments with and without kinematic optimisation. The two-tailed P values for the unpaired t-test were 0.0041, 0.0021 and 0.0081 for the experiments with the double cylinder, cylinder and polyflap, respectively, and thus statistically significant.

We also verified the proposed method in a multi-contact prediction experiment (Fig. 9). The manipulated object is a stapler lifted by the robot using three fingers. The goal of the experiment is to predict the trajectory of the object given the reference trajectory of the robot's hand. The obtained accuracy of the predictor is lower than for the pushing experiment. We observe a latency between the reference and predicted trajectory. The latency is caused by the properties of the regression method (the variance of Gaussians). If the variance is high we observe the latency. On the other hand, when the latency is too small we lose generalization properties of the predictor. However, despite the fact that the prediction error accumulates we can still predict the pose of the object quite well 15 seconds into the future (cf. the pose of the objects at the end of motion presented in Fig. 9). It is notable that most rigid body simulators modeling multi-contact force closure grasps have the well-known problem that they are unable to produce stable trajectories for more than a few seconds unless carefully tuned for objects with detailed models known a priori.

The first finding is that kinematically optimizing during the tracking stage with the training data reduces the subsequent prediction error on test cases. It can be seen in Fig. 8b and Fig. 7b that without optimization the finger starts to penetrate the object during subsequent predictions on the test cases (Fig. 8b3 and Fig. 7b4). In these examples because the error accumulates the finger finally goes right through the object. When we introduce the optimization during the prediction stage we can also reduce the prediction error. Prediction error falls even when the tracking data are not filtered (Fig. 5). When we use optimization during both the tracking and prediction stages the mean prediction error falls further except in the case of the polyflap, where they are comparable. The second finding is a demonstration that the prediction method works both for single contacts (when the object can twist and roll away from the manipulator contact point) and for multi-contact manipulation. In addition the data for our multi-contact case is not a precise model of the object shape, but one acquired on-line using an inaccurate depth camera. Thus the method is shown to be robust to significant noise in the reference shape model. In our future work we will extend the method to allow changing numbers of contacts by learning a predictor for each number of contacts, and then building a switching model.

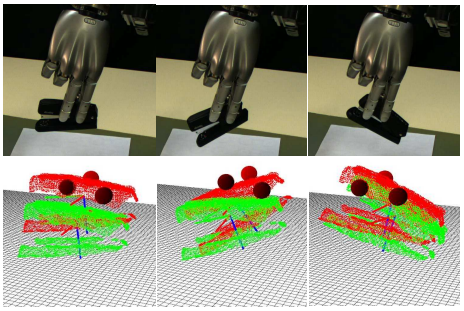

Fig. 9. Predicted state of the object (green point cloud) at the end of the multi-contact lift experiment. The reference pose of the object is represented by the red point cloud. Red spheres correspond to the last fingers' links.

VI. Conclusions

In this paper an extension to previous methods for learning to predict object motion has been presented. Previous published results already show that learning methods outperform physics simulation in terms of the accuracy of the predicted trajectory. But those learning methods still produce trajectories which are kinematically infeasible. In this paper we have described a method that optimises any prediction to make it kinematically plausible. The key side effect of this is that the predicted trajectory also becomes more accurate. The same approach can be applied prior to learning, kinematically cleaning trajectories prior to learning.

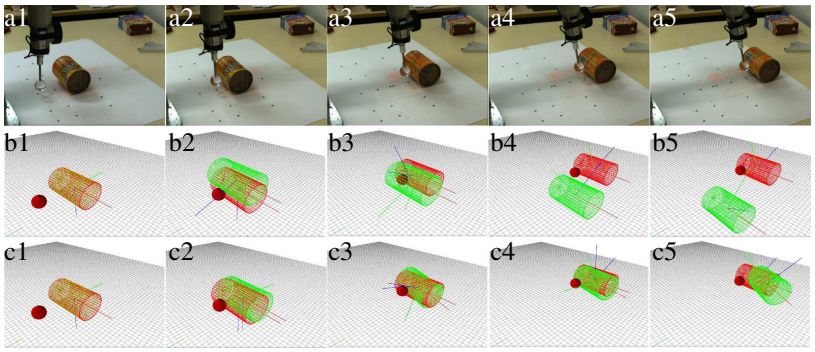

Fig. 7. Pushing experiment (a): prediction results at 0, 80, 160, 240 and 320th frame for cylinder using KDE predictor without optimization (b) and with the proposed optimization methods during both the tracking and prediction stages (c)

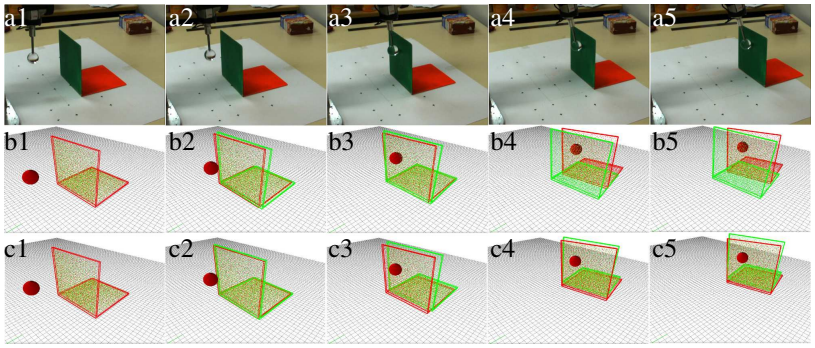

Fig. 8. Pushing experiment (a): prediction results at 0, 80, 160, 240 and 320th frame for polyflap using the KDE predictor without optimization (b) and with the proposed optimization methods applied during both the tracking and prediction stages (c)

We showed that by performing both optimisations an error which is consistently low can be achieved. Finally we also demonstrated empirically that the proposed method can be applied to predict the motion of previously unknown object manipulated by a multi-finger robotic hand. Despite the fact that currently the computation time is few times longer that simulation time (to predict 1 second of trajectory takes a few seconds) the relationship is a linear and so simple code optimizations should enable real time motion prediction.

VII. Acknowledgments

We gratefully acknowledge the support of EU-FP7-IST grants Nos 248273 (GeRT) and 600918 (PaCMan).

References

[1] M. Kopicki, M. Zurek, R. Stolkin, T. Morwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 5722–5729, 2011.

[2] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[3] D. J. Cappelleri, J. Fink, B. Mukundakrishnan, V. Kumar, and J. C. Trinkle, "Designing open-loop plans for planar micro-manipulation," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 637–642, 2006.

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

[5] D. Song, N. Kyriazis, I. Oikonomidis, C. Papazov, A. Argyros, D. Burschka, and D. Kragic, "Predicting human intention in visual observations of hand/object interactions," in *Robotics and Automation, 2010 IEEE International Conference on*, pp. 1600–1607, 2013.

[6] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *IEEE Int. Conf. on Robotics and Automation*, vol. 3, 2003.

[7] B. Ridge, D. Skocaj, and A. Leonardis, "Towards learning basic object affordances from object properties," in *Proceedings of the 2008 International Conference on Cognitive Systems*, 2008.

[8] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner, "Learning to perceive affordances in a framework of developmental embodied cognition," in *IEEE 6th International Conference on Development and Learning, 2007. ICDL 2007*, pp. 110–115, 2007.

[9] A. Stoytchev, "Learning the affordances of tools using a behavior-grounded approach," in *Springer Lecture Notes in Artificial Intelligence (LNAI) 4760 (*E. Rome et al.*)*, pp. 140–158, 2008.

[10] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz, "Simulation-based temporal projection of everyday robot object manipulation," 2011.

[11] M. Kopicki, J. Wyatt, and R. Stolkin, "Prediction learning in robotic pushing manipulation," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1–6, 2009.

[12] M. Kopicki, R. Stolkin, S. Zurek, T. Mörwald, and J. Wyatt, "Predicting workpiece motions under pushing manipulations using the principle of minimum energy," in *Proceedings of the RSS workshop on Representations for object grasping and manipulation in single and dual arm tasks*, 2010.

[13] M. Chalon, J. Reinecke, and M. Pfanne, "Online in-hand object localization," in *Intelligent Robots and Systems (IROS), 2013 IEEE International Conference on*, pp. 2977–2984, 2013.

[14] E. Hansen, "Nonparametric conditional density estimation," *Neural Computation*, vol. 21, no. 2, pp. 533–559, 2004.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. on Neural Networks*, 1995.

[16] Neuronics AG, "Katana user manual and technical description." http://www.neuronics.ch, 2004.

[17] T. Mörwald, M. Zillich, and M. Vincze, "Edge tracking of textured objects with a recursive particle filter," in *Proceedings of the Graphicon 2009*, (Moscow, Russia), 2009.

[18] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proceedings of Robotics: Science and Systems*, (Cambridge, USA), 2005.