Learning kinematic forward models for the motion of manipulated objects

Marek Kopicki, Sebastian Zurek, Rustam Stolkin, Thomas Mörwald, Jeremy L. Wyatt

***Abstract*—An important problem in robotic manipulation is the ability to predict how objects behave under manipulative actions. This is necessary to enable many manipulations to be planned. Physics simulators are used, but model many kinds of object interaction poorly. An alternative is to learn a motion model from data. In this paper we address the problem of learning to predict the kinematic interactions of rigid bodies, and demonstrate the results in the domain of robotic push manipulation. A robot arm applies pushes to various objects and observes the resulting motion with a vision system. The relationship between push actions and object motions is learned, and enables the robot to predict the motions that will result from new pushes. The learning does not make explicit use of physics knowledge. We show how to pose the problem in both a regression and a density estimation framework, and show that both frameworks exhibit good interpolative generalisation performance. Additionally, we show how factorising prediction can enable extrapolative generalisation with respect to object shapes and robot actions. Performance is evaluated through a combination of virtual experiments in a physics simulator, and real experiments with a 5-axis arm equipped with a simple, rigid finger.**

***Index Terms*—Learning, forward model, manipulation, prediction.**

I. Introduction

An essential ability for a robot that can plan future activity is the ability to predict the effects of its actions. In this paper we describe a new approach to prediction of the behaviour of rigid objects under manipulation. Essentially, rather than using a single, general engine informed by the laws of mechanics, we apply machine learning techniques to obtain many different learned predictors. These algorithms do not rely on any prior encoding of Newtonian mechanics, impenetrability, gravity, or kinematic relations between objects, all these being learned from data. This paper shows that these learners are able to learn good predictive models (forward models) for a variety of objects, in the case when the behaviour can be adequately described by a kinematic model.

We make several contributions. First, we show how the problem of prediction can be posed in two different ways, within regression and density estimation frameworks, and study the performance of each approach. Second, we show how the kinematic prediction problem for objects can be naturally factorised and hence can provide a degree of generalisation with respect to making predictions about objects with novel shapes not seen during learning, and objects subjected to previously unencountered manipulative actions.

M. Kopicki, S. Zurek, R. Stolkin and J. L. Wyatt are with the Department of Computer Science, University of Birmingham, UK e-mail: msk@cs.bham.ac.uk. Kopicki and Zurek are identified as joint first authors for this paper.

T. Mörwald is with the Automation and Control Institute, Vienna University of Technology, AT.

The problems we tackle also go beyond what is addressed in previous work. Predictions are made about the trajectories of a manipulated object rather than of the robot itself, and these objects are not in free flight but are in contact with both a robot manipulator and a table surface. This means that the shape of an object and its contact relations are significant in determining how it moves. One of the advantages we have shown of a learning approach over using a physics engine for prediction is that the learners can also capture the effects of unobserved variables such as mass distribution and frictional coefficients on object motion. We do not consider grasping actions, but instead the problem of prediction under pushing actions. These problems appear simple in the sense that there is only a single finger contact between the robot and the object, but are harder in that now the object is free to move quite differently to and even independently of the manipulator, such as when an object tips or rolls. Finally the prediction problem is particularly challenging because of the fact that the predictors are trained for one step prediction, but are tested for predictions over multiple steps (in this paper around 100-150 steps) into the future by feeding back the predictions as the inputs for the next time step.

This paper extends our previous work [1], [2], [3], [4] in several ways. First, most of our previous results were in simulation, and here all our results (except for one experiment) are given in terms of both simulation experiments and with real experiments in which a robot arm pushes real objects. Second we show that good models can be learned using both regression and density estimation for a variety of real objects, and that these object-specific models are significantly better at prediction than a physics engine. Third we show on real data that a factorised version of the density estimation approach enables extrapolative generalisation to novel actions. Finally we show that it enables extrapolative generalisation to objects of novel shape. We also show results for interpolative generalisation with respect to shape, but it is important to note that extrapolative generalisation is a much harder problem than interpolative generalisation.

In summary, the work presented here tests three hypotheses. The first hypothesis (Hypothesis 1) is that a learning approach can outperform physics engines for prediction of rigid body motion for a variety of objects. The second hypothesis (Hypothesis 2) is that the learned predictions generalise both interpolatively and extrapolatively with respect to the actions performed on the object. The third hypothesis (Hypothesis 3) is that the learned predictions generalise both interpolatively and extrapolatively with respect to the shape of the objects.

The remainder of this paper is laid out as follows. Section II reviews various literature including early investigations of robotic pushing manipulation, and related uses of machine

learning to make predictions about objects. Section III introduces mathematical methods for representing and reasoning about the motion of pushed objects. The prediction problem is stated in Section IV. Section V suggests how to improve generalisation performance by including extra information about the learned behaviours of small parts of objects of various shapes. Section VI discusses several possibilities for posing the prediction problem in terms of statistical machine learning, and compares and contrasts some of the different possible probabilistic methods that can be employed. Section VII presents our method of combining this extra information using a factored representation and discusses why this may have some useful advantages. Section VIII explains the details of how we have implemented several different approaches to learning to predict the motions of pushed objects. Section IX provides details of our experimental studies to evaluate, compare and contrast the benefits of the various possible approaches. Section X presents the results of our experimental studies. Section XI discusses what can be observed in these results and provides some concluding remarks.

II. Background

Although much work has been done on push manipulation in robots [5], [6], [7], [8] it is restricted to planar sliding motions of what are effectively 2D objects. There is little literature addressing the more complex problem of push manipulations on real 3D bodies, which are free to tip or roll. It is possible to use physics simulators to predict the motion of interacting rigid bodies. However, this approach is reliant on explicit knowledge of the objects, the environment, and key physical parameters which can be difficult to tune. Even then, such predictions may not be possible due to inherent limitations of the physical model employed, for example when modelling friction.

Krotkov [9] noted that comparatively little robotics research focuses on the problems of making a robot discover the properties of objects, other than their shapes and positions. Some machine learning approaches have been developed to classify or provide predictions for objects or object classes, e.g. rolling versus non-rolling objects [10], [11], or liftable versus non-liftable objects [12]. These kinds of approach are limited, in that predictions learned may not be generalisable to a new object, pose or push direction, and explicit 6-DOF rigid body motions are not predicted. In contrast, our approach learns to make predictions of explicit 3D rigid body transformations. The probabilistic nature of the learning enables generalisation to novel push directions, object poses, and objects with novel shapes.

One of the issues explored in this paper is how a combination of appropriately trained factored probability density can facilitate a degree of generalisation with respect to making predictions about objects with different shapes or subjected to different manipulative actions than those encountered during training. Other researchers have also looked at various ways of combining the output of multiple predictors, in the context of machine learning approaches to motor control. Work from the computational neuroscience literature, [13], suggests a continuously re-learnable model for modular motor control, where complex control signals can be created from a weighted combination of the outputs of a set of learned forward models, each of which is itself comparatively simple. The forward models may be thought of as corresponding to predictors for different contexts. The work is demonstrated through a simple simulation experiment, in which the learned controllers are applied to a 1D mass-spring-damper system where the three key system parameters can adopt different values. A degree of interpolative generalisation is shown, e.g. two modules that are trained for large mass and small mass can combine their outputs to effectively control a medium-sized mass.

Stoytchev [14] described a robotic system that can learn affordances of sticks and hook-like tools by using them to apply pushing actions to observed objects. A relationship can be learned between the action of a robot wielding a particular tool, and the resulting motion of the pushed object. These experiments featured 2D scenes, in which the motions of very simple objects (puck-like discs) were restricted to sliding on a 2D table-top, under only four discrete possible actions (pushing away from or towards the robot, and in left and right directions). Importantly, while the outcomes of actions could be learned for tools of various different shapes, the system was not able to apply knowledge learned from one tool to make predictions about another tool of similar shape. For example, the behaviours of two different tools that both share a hook-shaped section must be learned separately for each tool – the system cannot apply knowledge about the influence of hook-shape, learned while exploring one tool, to make predictions about a new tool that shares the same shape. In contrast, in this paper we present a method by which knowledge about the behaviour of one pushed object can be transferred to another object which has a different shape. This is because small component parts of each object may share commonalities which may influence the objects' behaviours in common ways – by decomposing objects into local parts, and assigning independent predictive factors to each, we show how a learning system with a degree of shape generalisation is possible.

III. Encoding rigid body kinematics

In this section we set up the basic notation to represent rigid body transformations, which are required to pose the prediction problem in later sections of this paper.

From classical mechanics we know that in order to predict the state of a rigid body, it is sufficient to know its mass, velocity and a net force applied to the body. We do not assume any knowledge of the mass and applied forces, however the motion of a body over time encodes its acceleration – an effect of the applied net force.

Consider three reference frames A, B and O in a 3-dimensional Cartesian space. Frame A can represent a robotic finger which pushes an object with frame B, which in turn is placed on a table top with frame O as in Figure 1. While frame O is fixed, A and B change in time and are observed at discrete time steps $..., t-1, t, t+1, ...$ every non-zero Δt. Frame X at time step t is denoted by X_t, the rigid body transformation between frame X and frame Y is denoted by $T^{X,Y}$.

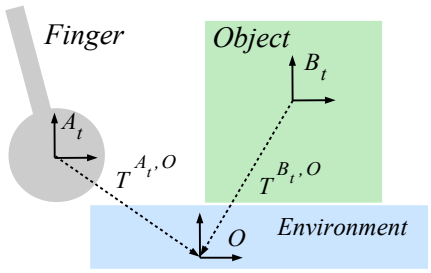

Fig. 1. 2D projection at time t of a robotic finger with frame A_t, an object with frame B_t, and a ground plane with constant frame O.

If the net force and the body mass are constant, transformations T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provide a complete description of the state of a body at time step t in the absence of other bodies. A triple of transformations $T^{B_t,O}$, T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provides a complete description of the state of a body in some fixed frame of reference O which accounts for a stationary environment. Similarly, a second triple of transformations $T^{A_t,O}$, T^{A_{t-1},A_t} and $T^{A_t,A_{t+1}}$ provide such a description for some other body with frame A. The state of a system consisting of two interacting bodies with frames A and B with a ground plane O can be also described by six transformations as shown in Figure 2. Transformation $T^{A_t,O}$ has been replaced by a relative transformation T^{A_t,B_t} which plays a critical role in the estimation of contacts between the two objects.

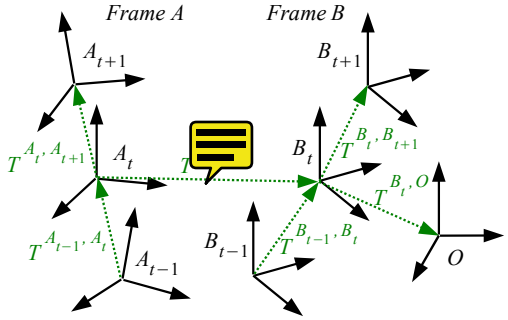

Fig. 2. A system consisting of two interacting bodies with frames A and B in some constant environment with frame O can be described by six rigid body transformations T^{A_t,B_t}, $T^{B_t,O}$, T^{A_{t-1},A_t}, $T^{A_t,A_{t+1}}$, T^{B_{t-1},B_t}, and $T^{B_t,B_{t+1}}$.

The behaviours of interacting bodies described by frames from Figure 2 are independent of any inertial frame [2]. Unfortunately, the representation of transformation $T^{A_t,A_{t+1}}$ as $A_{t+1}(A_t)^{-1}$, or explicitly given inertial frame I,

$$T_{in}^{A_t,A_{t+1}} = T^{I,A_{t+1}}(T^{I,A_t})^{-1} \tag{1}$$

makes transformation (1) dependent on the currently used inertial frame I (see Figure 3). Alternatively, instead of an inertial frame-dependent transformation $T_{in}^{A_t,A_{t+1}}$, one can represent all the transformations in the body frame.

The body frame transformation $T_{body}^{A_t,A_{t+1}}$ is obtained by moving instantaneous frame A, so that at time t it coincides with inertial frame I. Given some instantaneous object frame

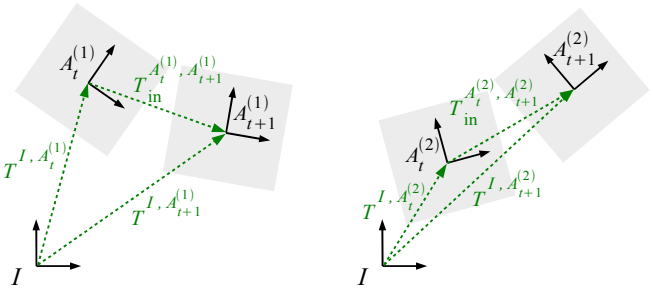

Fig. 3. In the above two scenes a pose change between time step t and $t+1$ as observed in instantaneous object body frame $A^{(1)}$ and the same object in another instantaneous body frame $A^{(2)}$ given inertial frame I are both the same. However because transformations $T^{I,A^{(1)}}$ and $T^{I,A^{(2)}}$ are different, the corresponding transformations in the inertial frame are also different, i.e. $T_{in}^{A_t^{(1)},A_{t+1}^{(1)}} \neq T_{in}^{A_t^{(2)},A_{t+1}^{(2)}}$.

A_t at time t, and the transformation $T_{in}^{A_t,A_{t+1}}$, one can obtain transformation $T_{body}^{A_t,A_{t+1}}$ in the body frame (via a similarity transform):

$$T_{body}^{A_t,A_{t+1}} = (T^{I,A_t})^{-1} T_{in}^{A_t,A_{t+1}} T^{I,A_t} \tag{2}$$

where $T^{I,A_{t+1}} = T_{in}^{A_t,A_{t+1}} T^{I,A_t} = T^{I,A_t} T_{body}^{A_t,A_{t+1}}$.

Given a transformation in the body frame, instantaneous object frame A_t at t and using Equation (2), transformation $T_{in}^{A_t,A_{t+1}}$ in the inertial frame is given by:

$$T_{in}^{A_t,A_{t+1}} = T^{I,A_t} T_{body}^{A_t,A_{t+1}} (T^{I,A_t})^{-1} \tag{3}$$

In further discussion we will retain subscripts in, but suppress subscripts $body$, and assume that all transformations $T^{X,Y}$ are transformations in the body frame X obtained using a similarity transform

$$T^{X,Y} \equiv T_{body}^{X,Y} = (T^{I,X})^{-1} T_{in}^{X,Y} T^{I,X} \tag{4}$$

IV. The Prediction Problem

The prediction problem can now be stated as follows: given that we know or observe the starting state, and the motion of the robotic finger, $T^{A_t,A_{t+1}}$, predict the resulting motion of the object, $T^{B_t,B_{t+1}}$. This is a problem of finding a function f:

$$f : T^{A_t,B_t}, T^{B_t,O}, T^{A_{t-1},A_t}, T^{B_{t-1},B_t}, T^{A_t,A_{t+1}} \longrightarrow T^{B_t,B_{t+1}} \tag{5}$$

The function f is capable of describing all possible effects of interactions between rigid bodies A and B, providing their physical properties and net forces are constant in time,[1] in the limit of infinitesimally small time steps. Furthermore, it can be approximately learned from observations for some small fixed time interval Δt between time steps.

In this work, we will focus on the kinematic regime, where robotic manipulations are performed relatively slowly. Hence we can assume quasi-static conditions, and ignore all frames

[1] A dynamic formulation could explicitly incorporate net forces into the domain and codomain of (5).

at time $t-1$. This conveniently reduces the dimensionality of the problem, giving a simplified function f_{qs}:

$$f_{qs} : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}} \longrightarrow T^{B_t,B_{t+1}} \quad (6)$$

Thus, starting with some initial state of the finger $T^{A_0,O}$ and object $T^{B_0,O}$, and knowing the trajectory of the finger $A_1, \ldots A_T$ over T time steps, one can predict a whole trajectory of the object $B_1, \ldots B_T$, by iterating the prediction obtained from f_{qs}. That is, the output of the prediction at time t is used as input to the prediction for the next time step.

V. Extra information useful for prediction

A learning-based prediction scheme governed by Equation (6) might be expected to generalise reasonably well to those novel pushing motions and object shapes that are spanned by the training set, assuming sufficiently many training examples. However, such a scheme cannot be expected to extrapolate effectively beyond the training set, since Equation (6) contains almost no knowledge of physics.

While a variety of extensions could be explored to improve extrapolative prediction performance, we propose to utilise proximity information about possible surface relations between objects involved in the physical interactions.

Consider a 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and the constant global frame O (Figure 4). We can identify local frames A^l_t and B^l_t, rigidly attached to small local planar surface patches at the contact point, or the points of closest proximity on the object and finger. We define the *agent contact* to be the information about changes in the frames A^l_t and B^l_t between the finger (agent) and object, that is $T^{A^l_t,A^l_{t+1}}$ and $T^{A^l_t,B^l_t}$. This agent information constitutes one type of extra input. By contrast, the term *global* information is used to denote data that specifies changes in the pose of the whole object, and is defined in terms of absolute coordinate frames.

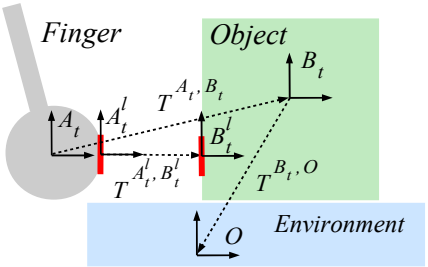

Fig. 4. 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and a ground plane with constant global frame O. Local frames A^l_t and B^l_t describe the local shape of the finger and an object at their point of closest proximity.

As demonstrated in Figure 5, the predictor must take some aspect of the object shape into account. A predictor based on Equation (6), trained on the smaller object B shown in the left scene, will fail to generalise to the larger object (right scene) – it will allow object A (the agent) to pass into object B. Adding agent information can assist in generalisation to novel object shapes, since it implicitly encodes some relevant information about object shape.

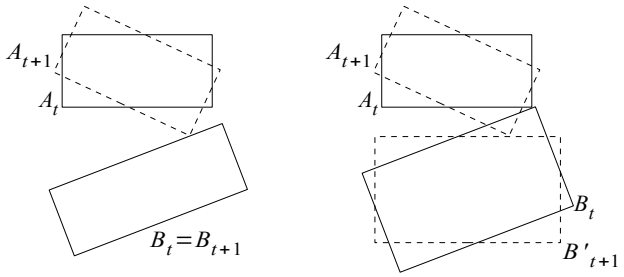

Fig. 5. Two scenes (left and right), each with two objects on a tabletop, viewed from above. Only the shape of object B is different between the scenes. Yet when A rotates to the position shown by the dashed outline at time $t+1$, the resulting displacement of B (represented by the transformation $T^{B_t,B_{t+1}}$) will be quite different.

In addition to representing how an object moves in response to a push by a robotic finger, it is desirable to incorporate learned information about the inherent tendencies of parts of an object to move in various directions with respect to the environment or other objects, regardless of whether the object is being pushed or not. This additional information may help when predicting the motion of a previously unseen object, or the response to a novel push direction, because it provides some prior knowledge about which kinds of motions are possible and which are not.

We can specify this additional information by attaching an arbitrary number N of additional coordinate frames $B^{S^k}_t$ to various parts of the object, $(k = 1 \ldots N)$. Each one of these frames is paired with a corresponding frame attached to the environment $E^{S^k}_t$, as shown in Figure 6.

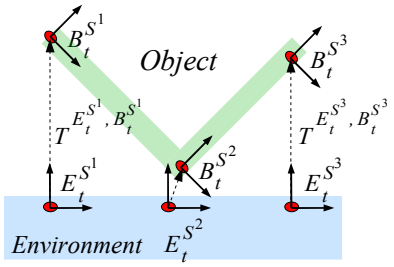

Fig. 6. Co-ordinate frames can be attached to an arbitrary number of localised shapes. So-called *environment contacts* can be learned for each of these shape frames, predicting a distribution of how the frame may move next, given where it is at the present time step.

To obtain the results presented in this paper, the number and location of these *environment contacts* on each of the different objects were determined by hand. The associated transformations $T^{E^{S^k}_t, B^{S^k}_t}$ for $k = 1 \ldots N$ provide the second type of input.

The utility of agent and environment contacts can be seen in Figure 7, which shows that the configuration of finger and object during a backward push is very different to any configuration present in a training set consisting only of forward pushes. A predictor using just global information will fail to generalise to a new push direction that differs markedly from

any observed in the training set. However, by incorporating information from the agent contact, the predictor can learn that the finger does not penetrate the object after contact. Nevertheless, there are other constraints on the object motion, such as the ground plane, which are not encoded by the agent contact. To model these other facts about possible object motion requires the use of additional environment contacts (bottom right of Figure 7).

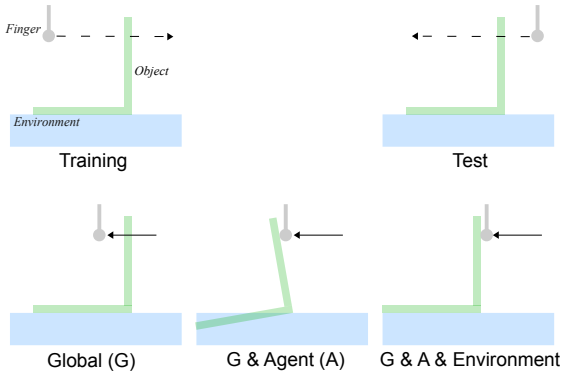

Fig. 7. Schematic diagram (2D projection of 3D scene) in which an object (of L-shaped cross-section) on a supporting surface is pushed by a robotic finger. Various predictors are trained solely on forward pushes (top left), but tested on backwards pushes (top right). The top panels show the push trajectory for the training and test phases, whereas the bottom panels show the outputs from three types of predictor in the test phase. A predictor using just global information will fail to generalise, and will predict that the object does not move as the finger passes through it (bottom left). Adding an agent contact will stop the finger penetrating the object, but does not guarantee that the predicted object motion will respect other impenetrability constraints (bottom middle). Finally, using an additional environment contact attached to the base of the object, a physically plausible motion is obtained (bottom right).

VI. Formulation using Regression or Density Estimation

One way to incorporate the extra information A_t^l,B_t^l and $E_t^{S^k}$, $B_t^{S^k}$, provided by the agent and environment contacts, is simply to enlarge the domain of function f in Equation (6), that is:

$$f'_{qs} : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}, T^{A_t^l,B_t^l}\{, T^{E_t^{S^k},B_t^{S^k}}\}_{k=1\ldots N} \longrightarrow T^{B_t,B_{t+1}} \quad (7)$$

Unfortunately, because the dimensionality of the domain of f'_{qs} grows with the number of environment contacts N, the difficulty of learning the mapping f'_{qs} rapidly increases as more environment contacts are added.

In principle it is straightforward to solve the prediction problem with a nonparametric regression analysis by taking $T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}$ etc. as independent variables, and $T^{B_t,B_{t+1}}$ as the dependent variable. Nevertheless, a powerful regression technique is required since the domain of f_{qs} has at least 18 dimensions, assuming an Euler angle parameterisation is used in the rigid body transformations (see Section VIII-A).

As an alternative to learning mapping (7) by regression, we can also recast f'_{qs} as a conditional probability density (CPD) over possible object motions $T^{B_t,B_{t+1}}$ [1]:

$$p_{qs}(T^{B_t,B_{t+1}}|T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}, T^{A_t^l,B_t^l} \{, T^{E_t^{S^k},B_t^{S^k}}\}_{k=1\ldots N}) \quad (8)$$

This density carries multiple hypotheses about possible object motions, which can be utilised in e.g. Bayesian filters. In [4] we have presented a framework that can successfully track an object's pose, despite ambiguous visual feedback due to occlusions.

VII. Factorisation of the Probability Density

The CPD formulation allows us to combine the extra information represented by frames A_t^l,B_t^l and $E_t^{S^k}$, $B_t^{S^k}$, as a product of densities to approximate p_{qs}. The component densities factorise the conditioning variables of p_{qs} – their domains' dimensionalities are smaller, and so potentially they can manage the complexity of incorporating more information into the predictor. Furthermore, a product of up to N densities can be formed in $2^N - 1$ ways, where the particular form can depend on some current context (for example a variable set of contacts), in which each component corresponds to an appropriately simplified f'_{qs}.

Schematically, for some normalisation constant C we propose:

$$p_{qs} \approx C\ p_{global}\ p_{agent} \prod_{k=1\ldots N} p_{env,k} \quad (9)$$

where

$$p_{global} \equiv p_{global}(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}}, T^{A_t,B_t}, T^{B_t,O}) \quad (10a)$$

$$p_{agent} \equiv p_{agent}(T^{B_t^l,B_{t+1}^l}|T^{A_t^l,A_{t+1}^l}, T^{A_t^l,B_t^l}) \quad (10b)$$

$$p_{env,k} \equiv p_{env}(T^{B_t^{S^k},B_{t+1}^{S^k}}|T^{E_t^{S^k},B_t^{S^k}}) \quad (10c)$$

denote the *global*, *agent* and *environment* density factors repectively [1][2]. Note that the agent factor p_{agent} is conditioned on the motion represented by the transformation $T^{A_t^l,A_{t+1}^l}$, which allows p_{agent} to be used independently of the global factor p_{global}. This is not possible for the environment factors $p_{env,k}$, because the corresponding "action" represented by $T^{B_t^{S^k},B_{t+1}^{S^k}}$ is not known at time t, since it depends on the estimated object movement $T^{B_t,B_{t+1}}$.

The global, agent and environment factors encode information about which candidate rigid body transformations are more or less feasible for each frame of reference respectively. However, once we form the product of these densities, only transformations which are feasible in all factors' frames will have high probability in the resulting combined distribution.

We now consider the relations between transformations expressed in the body frame of the local patches and corresponding transformations in the inertial frames. For coordinate frames as shown in Figure 4, from object rigidity and using Equation (2) we have:

$$T^{A_t^l,A_{t+1}^l} = (T^{I,A_t^l})^{-1} T_{in}^{A_t,A_{t+1}} T^{I,A_t^l} \tag{11a}$$

$$T^{B_t^l,B_{t+1}^l} = (T^{I,B_t^l})^{-1} T_{in}^{B_t,B_{t+1}} T^{I,B_t^l} \tag{11b}$$

$$T^{B_t^{S^k},B_{t+1}^{S^k}} = (T^{I,B_t^{S^k}})^{-1} T_{in}^{B_t,B_{t+1}} T^{I,B_t^{S^k}} \tag{11c}$$

for some inertial frame I. $T^{A_t^l,B_t^l}$ and $T^{E_t^{S^k},B_t^{S^k}}$ can be determined directly using Transform (4), for example:

$$T^{A_t^l,B_t^l} = (T^{I,A_t^l})^{-1} T_{in}^{A_t^l,B_t^l} T^{I,A_t^l} \tag{12}$$

A prediction problem can then be defined as finding that transformation $\widetilde{T}_{in}^{B_t,B_{t+1}}$ in the inertial frame which maximises the product of densities (9):

$$\widetilde{T}_{in}^{B_t,B_{t+1}} = \underset{T_{in}^{B_t,B_{t+1}}}{\operatorname{argmax}} \left\{ p_{global}\ p_{local} \prod_{k=1\ldots N} p_{env,k} \right\} \tag{13}$$

where the similarity transforms (2) (in frame B_t) and (11) must be used to evaluate p_{global}, p_{agent} and the N environment factors $p_{env,k}$ for a given $T_{in}^{B_t,B_{t+1}}$.

VIII. Implementation

We have now presented two formulations of the prediction learning problem: 1) as function approximation, and 2) as density estimation. We have suggested that there may be an advantage to solving the density problem by factorising.

A. Parameterisation

Both formulations of the prediction learning problem critically depend on parameterisations of rigid body transformations, i.e. positions and orientations of a rigid body in a 3D Cartesian space. 3-DOF Positions can be uniquely parameterised by vectors $p \in \mathbb{R}^3$, and 3-DOF orientations by 3×3 rotation matrices $R \in SO(3)$ which comprise a special orthogonal group $SO(3)$ with respect to matrix multiplication (see e.g. [15]). There exist other parameterisations of $SO(3)$ which unlike rotation matrices may not be unique, however they provide a lower-dimensional description of 3-DOF orientations. In this paper we have tested two such parameterisations: Euler angles and quaternions.

1) Euler angles: XYZ Euler angles (α, β, γ) describe rotation $R(\alpha, \beta, \gamma) \in SO(3)$ by combining elementary rotations about axes X, Y and Z, by angle α, β and γ respectively:

$$R(\alpha, \beta, \gamma) = R_Z(\gamma) R_Y(\beta) R_X(\alpha) \tag{14}$$

XYZ Euler angles (α, β, γ) can be computed directly from coefficients of rotation matrix (14) as it is shown e.g. in [15].

Euler angles, as a local parameterisation of $SO(3)$, suffer from singularities - there is no global invertible smooth mapping $(\alpha, \beta, \gamma) \to SO(3)$. Thus there can be two transformations that are arbitrarily close, but have a large difference in one of their Euler angles. This discontinuity places an extra burden on learning methods, which have to recover the circular nature of the Euler angle representation. However, if differences between angles are required (as e.g. for kernel density methods – see Section VIII-C) then the problem can be finessed by computing a circular difference.

2) Quaternions: Unit quaternions provide a global parameterisation of $SO(3)$, but at the cost of using four parameters instead of three as in the case of Euler angles. The unit quaternion q is a normalised 4-dimensional vector associated with a rotation about unit-length axis $\omega \in \mathbb{R}^3$ by angle θ:

$$q = (\cos(\theta/2), \omega \sin(\theta/2)) \tag{15}$$

The set of all unit quaternions comprise a 3-dimensional sphere $S^3 \in \mathbb{R}^4$ and form a double cover of $SO(3)$, since a rotation about axis ω by angle θ is equivalent to a rotation about axis $-\omega$ by angle $-\theta$. Consequently any rotation $R \in SO(3)$ corresponds to exactly two quaternions q and $-q$. Therefore any distance metric defined for quaternions q and q' must also test the case with q and $-q'$.

B. Regression method

We used Locally Weighted Projection Regression (LWPR) [16], a powerful method applied widely in robotics, to estimate the mapping described by Equation (6). The regression scheme was implemented using the LWPR software library [17]. LWPR was chosen because it employs an incremental learning algorithm that can handle a large number of input dimensions.

Several versions of the regression method were implemented, with varying amounts of information, corresponding to the global (referred to as LWPR-G), global and agent (LWPR-GA) and global, agent and environment (LWPR-GAE) information. The Euler angle parameterisation was used for all the LWPR versions. In the case of LWPR-G, there were three rigid-body transformations provided as inputs to LWPR, which formed an 18-dimensional input space. The output was encoded as a 6-dimensional vector, that described the motion of the object $T^{B_t,B_{t+1}}$.

For the case of the global plus agent information (LWPR-GA), the input space was augmented by 6 dimensions to accommodate the transformation $T^{A_t^l,B_t^l}$. The contextual information provided by the agent contact can be characterised as discrete, since the agent contact only acts when the finger and object are touching, i.e. for a discrete set of values in the input space. This is challenging for LWPR, which is better suited to operating with continuous contexts.

With regard to LWPR-GA (and LWPR-GAE), the extension of the output space (to 12 dimensions) with $T^{B_t^l,B_{t+1}^l}$ did not change the prediction of $T^{B_t,B_{t+1}}$, since LWPR treats each output dimension independently as stated in [17]. Thus, it was not possible to take advantage of the extra information provided by $T^{B_t^l,B_{t+1}^l}$ in the same manner as the density estimation approach (Equation (10b)).

Finally, when using (possibly several) environment contacts, the input space was again enlarged, to form domains of up to 42 dimensions. In this formulation of regression, the inputs from the global, agent and environment frames were treated uniformly, whereas the density estimation scheme of Equation (13) made a clear distinction between these three types of input. Furthermore, as there is no special structure, such as a factored density, in our regression framework, we may expect poor performance when extrapolating to novel actions and object shapes.

Predictor	input space	output space
LWPR-G euler	18	6
LWPR-GA euler	24	6
LWPR-GAE euler	24 + N*6	6

TABLE I
DIMENSIONS OF INPUT AND OUTPUT SPACES OF LWPR PREDICTORS, WHERE N IS THE NUMBER OF "ENVIRONMENT CONTACTS".

The dimensions of the input and output spaces of each LWPR predictor are summarised in Table I.

C. Kernel density method

A variant of Kernel Density Estimation (KDE) [18] is used to approximate the conditional densities employed in the product in (13). KDE is an example of a non-parametric method, where an underlying distribution is modelled by a mixture of N identical kernels centred on *training samples*. We shall use "KDEF" to denote the KDE algorithm applied to the factored density (13), and use "KDE" to label the unfactored case (8). As for LWPR, there are three variants of KDEF: KDEF-G, KDEF-GA and KDEF-GAE, where the suffix denotes whether information from agent (A) and environment (E) contacts is utilised, in addition to global (G) information.

Training samples are composed of $D = D^x + D^y$ rigid body transformations, where D, D^x and D^y are numbers of transformations of the joint, marginal and conditional densities respectively. For example, D^y equals one for all conditional distributions in product (13), however D^x equals three for global joint distribution (10a), two for local joint distribution (10b) and one for local contact joint distributions (10c).

The stacked parameters of conditional distributions are denoted by (italic) T^x and T^y:

$$T^x = \{T_1, \ldots, T_{D^x}\} \tag{16a}$$
$$T^y = \{T_1, \ldots, T_{D^y}\} \tag{16b}$$

Transformations T^x and T^y are then mapped onto parameter vectors using one of the parameterisations introduced in Subsection VIII-A:

$$x(T^x) = \begin{bmatrix} x_1(T_1^x) & \ldots & x_{D^x}(T_{D^x}^x) \end{bmatrix}^{\mathrm{T}} \tag{17a}$$
$$y(T^y) = \begin{bmatrix} y_1(T_1^y) & \ldots & y_{D^y}(T_{D^y}^y) \end{bmatrix}^{\mathrm{T}} \tag{17b}$$

where (non-italic) T denotes the vector transpose operation and where each i-th rigid body transformation T_i^x and T_i^y is parameterised by vectors x_i and y_i respectively:

$$x_i(T_i^x) = \begin{bmatrix} x_i^l & x_i^o \end{bmatrix}^{\mathrm{T}} \tag{18a}$$
$$y_i(T_i^y) = \begin{bmatrix} y_i^l & y_i^o \end{bmatrix}^{\mathrm{T}} \tag{18b}$$

with location vectors $x_i^l, y_i^l \in \mathbb{R}^3$ and orientation vectors $x_i^o, y_i^o \in \mathbb{R}^3$ for Euler angles or $x_i^o, y_i^o \in \mathbb{R}^4$ for quaternions.

A product of N factors (13) for transformations T^y given T^x can be written as:

$$p(T^y|T^x) = C \prod_{i=1\ldots N} p_i(y(G_i^{-1}T^yG_i)|x(G_i^{-1}T^yG_i)) \tag{19}$$

where C is a normalisation constant, p_i are single factor conditional densities, and $G_i^{-1}T^xG_i$ and $G_i^{-1}T^yG_i$ are "block" similarity transformations such that $(T_{body}^x)_j = G_i^{-1}(T_{in}^x)_jG_i$ and $(T_{body}^y)_j = G_i^{-1}(T_{in}^y)_jG_i$ where G_i is an instantaneous frame of the i-th factor (also see (2)).

Each i-th conditional density is estimated as a mixture of products of simple kernels:

$$p_i(y|x) = \sum_{j=1\ldots M} K(x, \hat{x}_{i,j}, H_ih_i^x)K(y, \hat{y}_{i,j}, H_ih_i^y) \tag{20}$$

where all training sample vectors $\hat{x}_{i,j}$ and $\hat{y}_{i,j}$ are constructed during learning using relation (17) and remain fixed during prediction. Vectors h_i^x and h_i^y, known as *bandwidths*, are then estimated from training samples using the "multivariate rule-of-thumb" [18]. Bandwidth parameters h_i^x and h_i^y are additionally multiplied by *meta-parameter* $H_i \in \mathbb{R}$ which is estimated during *model selection* (see Section IX).

Each kernel function $K()$ in (20) (and similarly for y) can be rewritten as:

$$K(x, \hat{x}, h) = \prod_{k=1\ldots D^x} \exp\left[-\frac{1}{2}d(x_k, \hat{x}_k, h_k)\right] \tag{21}$$

where $d()$ is a *distance function* which determines a kernel type:

1) For a *Gaussian kernel*:

$$d_{\mathrm{Gauss}}(x, \hat{x}, h) = \sum_{i=1\ldots \mathrm{DIM}(x)} \left(\frac{x_i - \hat{x}_i}{h_i}\right)^2 \tag{22}$$

where $x, \hat{x}, h \in \mathbb{R}^6$ for Euler angles and $x, \hat{x}, h \in \mathbb{R}^7$ for quaternions. (22) corresponds to Mahalanobis distance defined as:

$$d_{\mathrm{Mahalanobis}}(x, \hat{x}, \mathbf{C}) = (x - \hat{x})^{\mathrm{T}}\mathbf{C}^{-1}(x - \hat{x}) \tag{23}$$

with diagonal covariance $\mathbf{C}$ such that $\mathbf{C}_{ii} = h_i$.

2) For a product of a Gaussian kernel and *von Mises–Fisher kernel*:

$$d_{\mathrm{GaussVMF}}(x, \hat{x}, h) = d_{\mathrm{Gauss}}(x^l, \hat{x}^l, h^l) + d_{\mathrm{VMF}}(x^o, \hat{x}^o, h^o) \tag{24}$$

where $h = \begin{bmatrix} h^l & h^o \end{bmatrix}^{\mathrm{T}}$, $h^o \in \mathbb{R}$, and (see [19]):

$$d_{\mathrm{VMF}}(x, \hat{x}, h) = 2h^o\left(1 - \left|(x^o)^{\mathrm{T}}\hat{x}^o\right|\right) \tag{25}$$

where $(x^o)^{\mathrm{T}}\hat{x}^o$ is a quaternion dot product and where absolute value $|\cdot|$ accounts for a $SO(3)$ quaternion double cover problem described in VIII-A.
The Von Mises–Fisher kernel (21) with (25) is an approximation (up to a multiplicative constant [19][20]) of the von Mises–Fisher distribution [21]:

$$f_4(x, \mu, \kappa) = C_4(\kappa) \exp(\kappa x^{\mathrm{T}} \mu) \tag{26}$$

where $C_4(\kappa)$ is a normalisation constant and where μ and κ are unit quaternions. In directional statistics $f_4(x, \mu, \kappa)$ is a probability distribution on $S^3 \in \mathbb{R}^4$ with mean μ and dispersion parameter κ.

Following (13), the prediction problem at each step can be defined as finding that transformation $\widetilde{T}^y$ which maximises the product of conditional densities (19) given fixed T^x, i.e.:

$$\widetilde{T}^y = \underset{T^y}{\operatorname{argmax}}\, p(T^y | T^x) \tag{27}$$

Prior to the maximisation procedure, for each factor i and each training sample j values of $K(x, \hat{x}_{i,j}, H_i h_i^x)$ in (20) are computed and assigned to $w_{i,j}$, referred to as *weights*. The normalised weight $w_{i,j}$ can be interpreted as a probability of generating x from a multivariate Gaussian centred at $\hat{x}_{i,j}$ with covariance $\mathbf{C}_{ii} = H_i h_i^x$. In order to increase the computational efficiency of the optimisation procedure we only consider the $M_i^{max} < M_i$ most relevant training samples, (i.e. those with the highest weights $w_{i,j}$ at the query point x). These are identified by calculating, for each training sample x the value of the distance function d (from (21)). The training samples are then sorted according to the distance using a partial sort algorithm (e.g. [22]) to identify the M_i^{max} most relevant training samples.

Equation (27) is computed using the differential evolution optimisation algorithm[2] (DE) [24]. DE is particularly simple to tune, since it is controlled by only two meta parameters: a *crossover probability* and a *size of population* of candidate solutions. If the number of generations is fixed, the total run time scales linearly with the number of factors N, their dimensionality D_i and the number samples M_i^{max}. Optionally, the entire maximisation procedure is stopped when no further significant improvement is observed.

The optimisation algorithm requires an ability to sample from product (19). This can be achieved by performing the following two step procedure:

1) Randomly choose an factor i and then sample j from a set of samples $\{w_{i,j}\}$ using the importance sampling algorithm with importance weights $w_{i,j}$ (e.g. [25]).
2) Sample from a multivariate Gaussian centred at $\hat{y}_{i,j}$ with covariance $\mathbf{C}_{ii} = H_i h_i^y$, using e.g. Marsaglia polar method [26].

Importantly, after generating a new solution candidate, all sub-vectors y^o have to be normalised when a quaternion parameterisation is used.

All KDEF factors are assumed to be independent, therefore their domains do not sum up. The dimensions of the input and output spaces for KDEF predictors are summarised in Table II.

Predictor	input space			output space
	global	agent	env	
KDEF euler	18	12	6	6
KDEF quat	21	14	7	7
KDEF vmf	21	14	7	7

TABLE II
INPUT AND OUTPUT SPACE DIMENSIONS OF KDEF FACTORS.

IX. EXPERIMENTAL STUDY

A. Overview of experiments

We have explored the ways in which various combinations of input information and learning techniques can make predictions, through three sets of experiments. The experiments use a combination of real training and test data collected from our robot setup (described in Section IX-B) and synthetic data from a simulation environment which has the benefit of perfectly known ground-truth.

Experiments L1, L2 and L3 (see Section X-A) show how our various approaches can (L)earn to make accurate predictions about the motions of previously observed objects, including a polyflap, a box and a cylinder. Thus they test Hypothesis 1 and the interpolative aspect of Hypothesis 2. Experiment A (see Section X-B) compares the capabilities of the different approaches to predict the consequences of novel pushing (A)ctions which are completely different from those encountered during training. Experiment A therefore tests the extrapolative part of Hypothesis 2. Experiments S1 and S2 (see Section X-C) compare the performance of the approaches in very challenging (S)hape generalization problems, where the the systems are tasked with making predictions about novel shapes which are very different to those encountered during training. Experiment S3 tests the ability to interpolatively generalise when small variations in shape induce larger variations in behaviour. The experiments S therefore test Hypothesis 3.

Additionally, Experiments L1-3 (Section X-A) investigate various parameters of the learning and prediction techniques. Each technique was trained on small, medium and large data sets to demonstrate that the methods converge. KDE techniques were attempted with a single global factor, a combination of two factors (global and agent), and multiple factors. We also compare results for different KDE kernel types and $SO(3)$ parameterisations (see Section VIII-C): Gaussian kernels with 1) quaternion and 2) Euler angle parameterisations, as well as 3) von Mises-Fisher kernels. We assess the performance of the LWPR regression method by using three versions – one that incorporates information equivalent to the KDE single "global" factor, and two others that include as inputs the additional information that is used to train the 2 factor and multiple factor KDEF predictors.

To quantify the effect of factorising p_{qs} (equation (8)) as a product of densities we also include results from two sets of variants of the single factor case, denoted by KDE-GA and KDE-GAE. Both of these variants use a single factor, but with an augmented input space compared to the KDEF cases. For KDE-GA, information equivalent to that used by the agent factor was appended. For KDE-GAE, as well as agent information, the transformations used as input by the environment factors were also included, so that KDE-GAE estimated p_{qs} directly in the form given by equation (8). Again these two variants were tested with the quaternion, Euler angle, and von Mises-Fisher parameterisations. Note that KDEF with

[2]Note that one cannot use here mean-shift algorithm [23] mostly due to a product involved in (27).

a single global factor (i.e. KDEF-G) and KDE-G are identical, hence KDE-G is not shown in the results.

The results of Experiments L (Section X-A) show that the KDE methods perform best with Gaussian kernels and with a quaternion parametrisation of $SO(3)$. Hence, in Section X-B and Section X-C only results for Gaussian and quaternion versions of KDE methods are presented. Likewise, we only show results of LWPR techniques using Euler angle $SO(3)$ parameterisations, since these result in a lower dimensional input space under which LWPR performs better.

B. Experimental setup

Multiple experimental trials were performed, in which a robotic arm equipped with a finger performs a random pushing movement towards an object (Figure 8). In each experiment data samples are stored over a series of such random trials. During each trial the arm moves for exactly 10 seconds, with a 1 second buffer at the beginning and the end. The frame rate is 30Hz, and for learning purposes the position of the object is estimated and stored every other frame (or every $1/15$th of a second). *Note that the results do not include any kind of recursive feedback from the vision system - in all results, the system has been tasked with predicting a full 150 time-steps into the future (i.e. an entire ten second push is predicted before the robotic finger begins moving).*

For real experiments, we use a 5-axis Katana robotic manipulator [27] equipped with a single rigid finger, and the motion of pushed objects is captured using a single camera and a visual tracking algorithm [28]. Simulation experiments are carried out using the NVIDIA PhysX physics engine [29].

The simulation environment usefully provides us with perfect ground-truth data against which to evaluate predictions, and also enables a very large number of experiments with many different values of key parameters (e.g. shape of pushed objects). Experiments on real robotic equipment, test the robustness of the method to the vagaries of the real world and the significant errors inherent in our robotic hardware and visual target tracking systems.

The virtual environment (using NVIDIA PhysX) does not replicate the physical properties of the real world perfectly, as is well illustrated in Figure 8. We automatically tuned the parameters of the physics engine to best fit the world. However, we have found that even when optimised (using variety of numerical optimisation techniques, the parameters neither correspond to their true values, nor do they generalise. Partly this is because a physics engine is a global approximator - when you tune its parameters to match one scene, it is unlikely to model a different scene well. However, regardless of how well the simulations correspond to the real world, the simulations still provide a *self-consistent experimental environment* within which to compare the accuracy of predictors that have been trained on data generated within that environment, while providing perfect ground-truth data for training and test motions that are at least physically plausible.

Local frames for environment contacts in the -GAE variants were fixed by hand to the edges of objects. In test cases with new objects the frames were again fixed by hand. An item for future work is to perform this process automatically. The bandwidth of all distributions used in the KDE methods, as well as parameters of the LWPR regression method, were tuned once by hand and kept constant throughout all the experiments.

Both KDE and LWPR approaches require many parameters to be determined before learning from a training set. A model selection procedure was performed during an initial set of experiments (Experiments L1–3, see Section X-A), to establish reasonable values for these parameters. Note that it was not possible to perform fully systematic optimisations due to the size of the parameter spaces. Rather, subsets of parameter space were selected by inspection and then explored using a grid search. Models were evaluated on a hold-out set, distinct from the training set.

In addition to selecting appropriate parameters for the learned predictors, we also attempted to tune the parameters of the PhysX physics simulator, so that a fair comparison could be made between the quality of its predictions and those of our learned predictors. The parameters of the physics engine which we tuned, included static friction, dynamic friction and the coefficient of restitution. Model selection on these parameters was performed as described above for the learned prediction systems, however we actually undertook a more extensive grid search for the PhysX parameters.

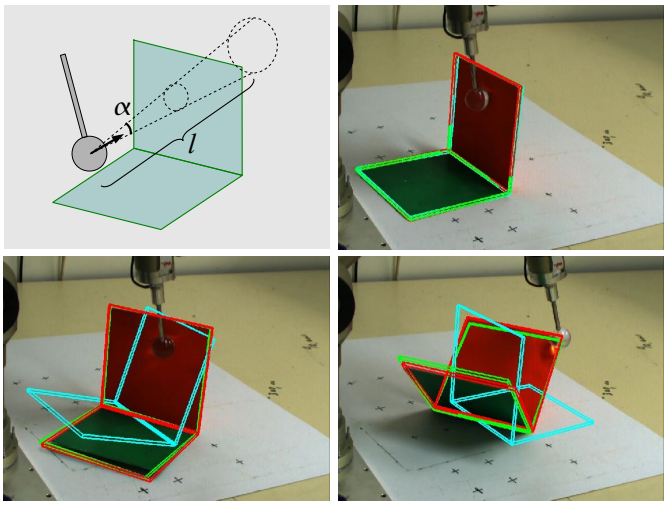

Fig. 8. A 5-DOF robotic arm equipped with a finger performs a random straight-line pushing movement of a variable length l=25±5 cm within a cone with angle α=20 deg towards an object (top left). The movement begins at a random location so that every small region on the upper part of an object is equally likely to be pushed. The object behaviour can be complex and varies depending on the finger trajectory and its pose relative to the object. In the image sequence shown above, the object begins to rotate anti-clockwise (top right - bottom left) before tilting (bottom right). The red wire-frame shows the output from the vision tracking system. The green wire-frame indicates the object pose predicted by the KDEF learning method, while the blue wire-frame is generated by the PhysX simulator. Although the PhysX predictions are qualitatively plausible, it was virtually impossible to tune the simulator so that its predictions match reality for all training data. Note that the entire motion sequence is predicted before the physical push is initiated, without any correction from visual feedback during the push execution.

C. Performance measure

In all experiments, we take the output of the tracked 6D pose of a real object to be ground-truth, and compare it against predictions which were previously forecast by the learned prediction system. The vision system does not provide perfect ground-truth, yielding typical errors of around $\pm$2mm during successful tracking, or arbitrarily large errors when the track is occasionally lost. However, comparing predictions to the outputs of the tracker still provides some useful information about discrepancies in the predictor, although clearly the performance of the predictors is limited by the accuracy of the data on which they are trained. Prediction performance is evaluated as follows.

At any particular time step, t, a large number, N, of randomly chosen points $p_n^{1,t}$, where $n = 1 \dots N$, are rigidly attached to an object at the ground-truth pose, and the corresponding points $p_n^{2,t}$ to an object at the predicted pose. At time step t, an average error E_t can now be defined as the mean of displacements between points on the object at the predicted pose and points on the object at the ground-truth pose:

$$E_t = \frac{1}{N} \sum_{n=1\dots N} |p_n^{2,t} - p_n^{1,t}| \tag{28}$$

Note that for each robotic push action, we predict approximately 150 consecutive steps into the future, with no recursive filtering or corrector steps, hence it is expected that errors will grow with range from the initial object pose. We therefore find it more meaningful to normalise all errors with respect to an "average range", R_t, of the object from its starting position, defined as:

$$R_t = \frac{1}{N} \sum_{n=1\dots N} |p_n^{1,t} - p_n^{1,0}| \tag{29}$$

For a test data set, consisting of K robotic pushes, each of which breaks down into many consecutive predictions over T time steps, we can now define average error and normalised average error:

$$E_{av} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{T} \sum_{t=1}^{T} E_t, \quad E_{av}^{norm} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{T} \sum_{t=1}^{T} \frac{E_t}{R_t} \tag{30}$$

For each set of test data, we also compute final error and normalised final error, which represent the typical discrepancy between prediction and ground truth that has accumulated by the end of each full robotic push:

$$E_f = \frac{1}{K} \sum_{k=1}^{K} |p_n^{2,T} - p_n^{1,T}|, \tag{31a}$$

$$E_f^{norm} = \frac{1}{K} \sum_{k=1}^{K} \frac{|p_n^{2,T} - p_n^{1,T}|}{R_T} \tag{31b}$$

Note that both normalised errors have no units.

We performed 10-fold cross-validation where at the beginning of each experiment all the trials are randomly partitioned into 10 subsets. Prediction was then subsequently performed (10 times) on each single subset, while learning (only for learned approaches) was always performed on the remaining 9 subsets of these trials. All the results were then averaged to produce a single estimation.

X. Results

A. Learning to predict the motions of a rigid body

In Experiments L1, L2 and L3, the robot applies a set of random pushes to objects including a polyflap, a box and a cylinder respectively. In each experiment, we tested both KDE and LWPR approaches. For KDE, we looked at factored (KDEF) and unfactored (KDE) cases with varying amounts of information (-G, -GA, -GAE), and tested the use of Gaussian kernels with 1) quaternion and 2) Euler angle $SO(3)$ parameterisations as well as 3) von Mises-Fisher kernels. Results are presented as graphs (Figures 9-12), image sequences (Figures 10-14), and in Tables III to VIII.

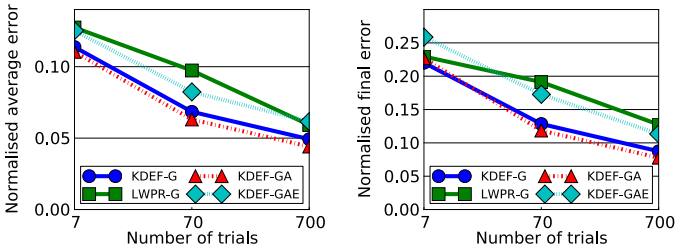

Fig. 9. Experiment L1: forward push on a polyflap, trained and tested on real data. Decrease in average (left) and final (right) prediction errors with increasing number of learning trials. LWPR-G performance versus KDEF with varying input information.

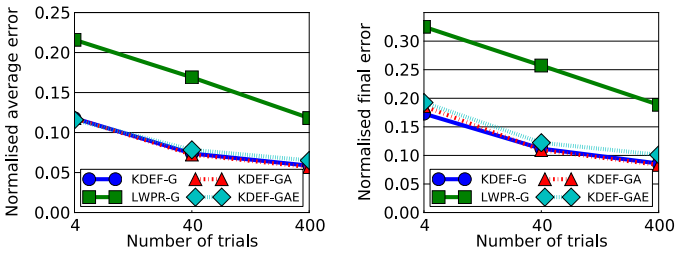

Fig. 11. Experiment L2: forward push on a box, trained and tested on real data. Decrease in average (left) and final (right) prediction errors with increasing number of learning trials. LWPR-G performance versus KDEF with varying input information.

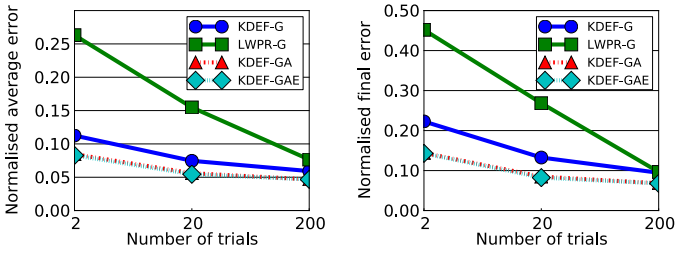

Fig. 12. Experiment L3: forward push on a cylinder, trained and tested on real data. Decrease in average (left) and final (right) prediction errors with increasing number of learning trials. LWPR-G performance versus KDEF with varying input information.

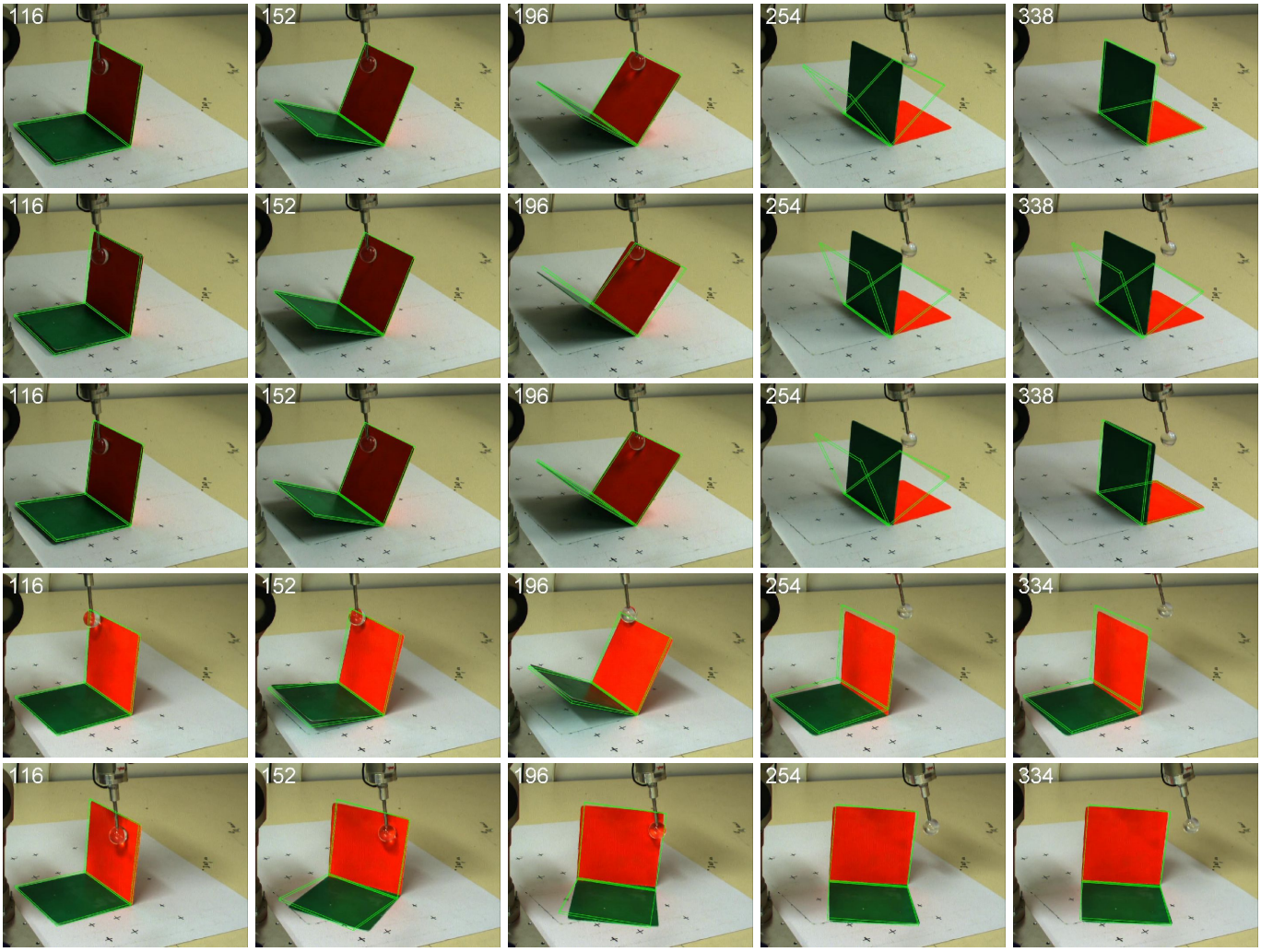

Fig. 10. Experiment L1: learning from real data – polyflap. Green outline shows predictions (from top row to bottom row) by KDEF-GA/quat, LWPR-G, and PhysX for one trial, and by KDEF-GA/quat on two other trials exhibiting different motions of the polyflap. (The frame number is shown in the top left of each image.)

It is observed that Gaussians with quaternions performs best. Also, note that we see cases where a -GA predictor outperforms a -GAE predictor which is unexpected. These results are based on real data using significantly erroneous visual tracking data. In later sections we will show results of tests in our simulation environment, in which the -GAE predictor performs best. This suggests to us that, the -GAE predictor would perform best in real experiments also, were we to upgrade to a more accurate visual tracking apparatus; or were the learning method to be extended to take explicit account of observation noise.

For LWPR we used an Euler angle parameterisation throughout, as this provides a lower dimension input space on which the regression method tends to work best.

1) Experiment L1: learning from real data with a polyflap object: Figure 10 shows example frames for predictions made by a KDEF-GA predictor (top) and an LWPR-G predictor (bottom). Tables III and IV provide detailed numerical results of many such tests of each method. Figure 9 graphs the results of the best of the LWPR methods against the KDEF methods with one, two and many factors.

2) Experiment L2: learning from real data with a box object: Figure 13 shows example frames for predictions made by a KDEF-GA predictor (top) and an LWPR-G predictor (bottom). Tables V and VI provide detailed numerical results of many such tests of each method. Figure 11 graphs the results of the best of the LWPR methods against the KDEF methods with one, two and many factors.

3) Experiment L3: learning from real data with a box object: Figure 14 shows example frames for predictions made by a KDEF-GA predictor (top) and an LWPR-G predictor (bottom). Tables V and VI provide detailed numerical results of many such tests of each method. Figure 12 graphs the results of the best of the LWPR methods against the KDEF methods with one, two and many factors.

B. Generalising to make predictions about previously unencountered manipulative actions

Experiment A tests the extent to which our predictive techniques can generalise to cope with novel kinds of manipulative actions, which have not been previously encountered during

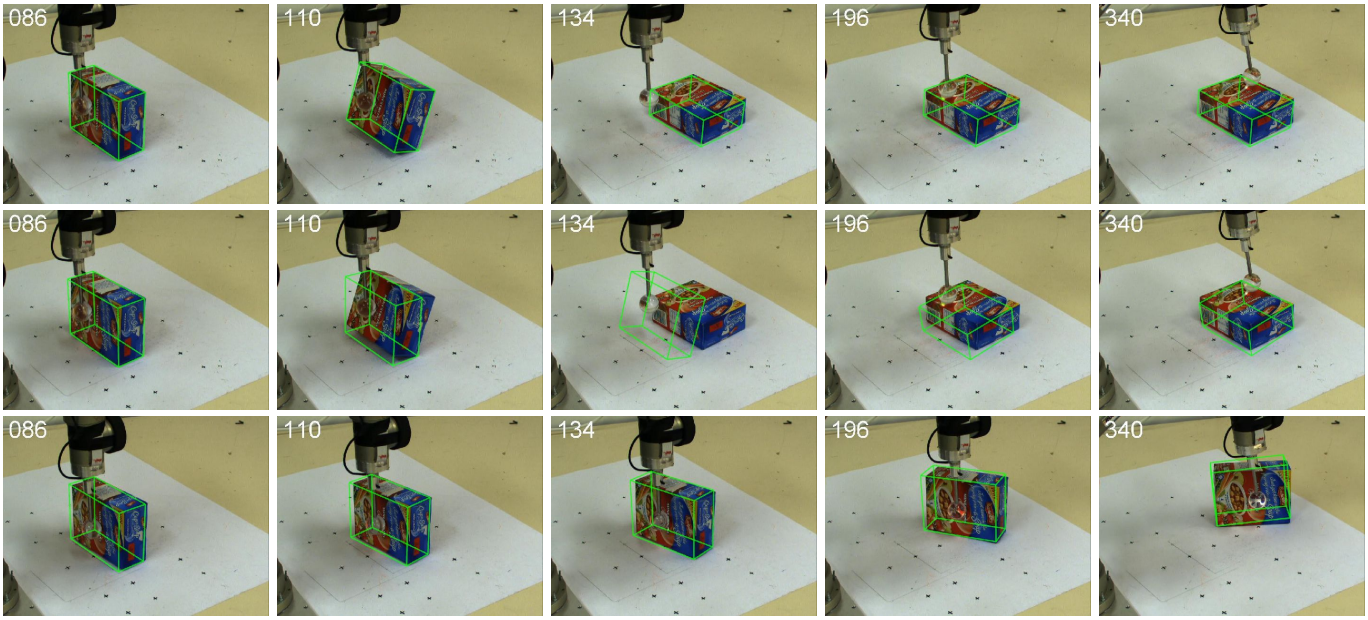

Fig. 13. Experiment L2: learning from real data – box. Green outline shows predictions (from top row to bottom row) by KDEF-GA/quat and LWPR-G for one trial in which the box topples over, and by KDEF-GA/quat on another trial in which the box slides. (The frame number is shown in the top left of each image.)

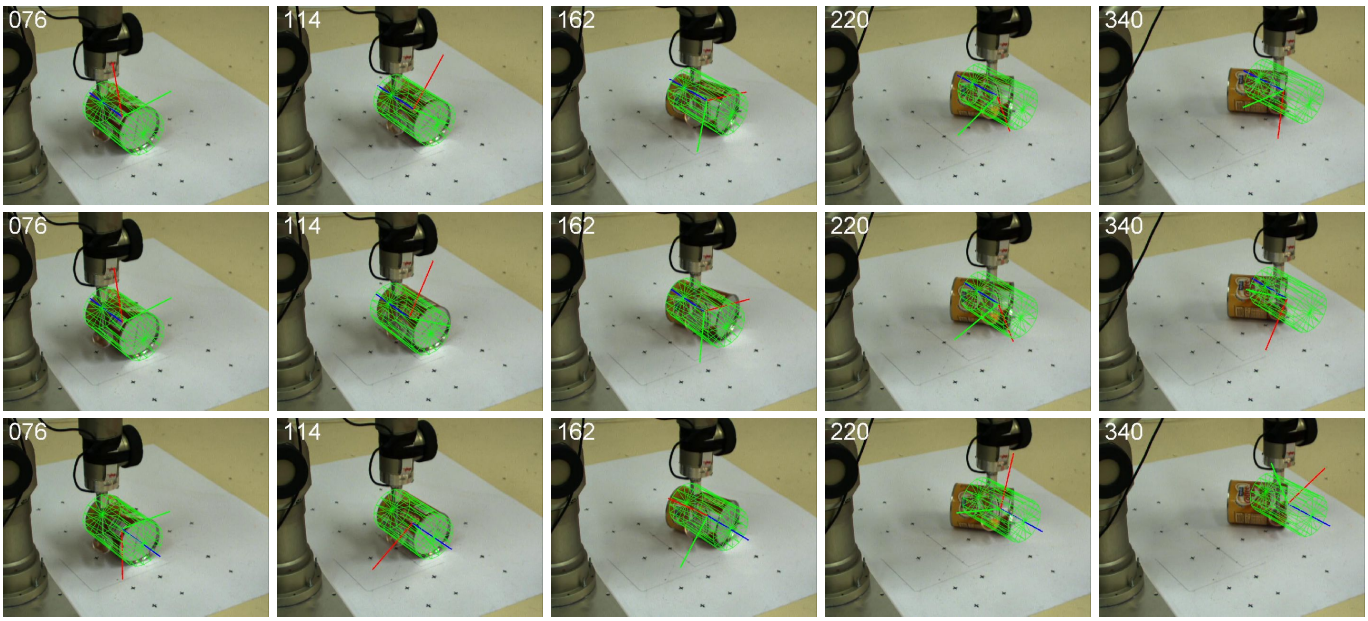

Fig. 14. Experiment L3: learning from real data – cylinder. Green outline shows predictions (from top row to bottom row) by KDEF-GA/quat, LWPR-G, and PhysX for one trial. A set of axes is attached to the cylinder to show the predicted rotation of the cylinder about its axis. (The frame number is shown in the top left of each image.)

training. The predictors were trained with 900 examples of pushes applied to a polyflap in a forwards direction. We then tested the predictors on 100 examples of pushes applied in a backwards pushing direction (see Section V and Figure 7 for an explanation and discussion of forwards and backwards push directions in this context). Figures 19 and 20 show example frames from running the predictors on synthetic and real data.

Figure 15 shows the normalised average error E_{av}^{norm} for tests carried out in a simulation environment and with real data from our laboratory robot setup. More detailed results are provided in Tables IX and X. As before, we use Gaussian kernels with quaternion parameterisation for KDE, and an Euler angle representation with LWPR, for the reasons explained earlier. In each case, we test performance according to the information utilised: global only, global and agent, or global, agent and environment, which for KDEF correspond to the -G, -GA and -GAE cases respectively. Also shown are the results from KDE-GA and KDE-GAE.

For both simulated and real data, it is clear that LWPR fails to generalise in this way – it predicts that the polyflap

will never move at all, because it was never exposed to these kinds of pushes during training. The single-factor KDE variant exhibits a similar level of normalised average error.

In contrast, the KDEF method is able to make physically plausible predictions, which are accurate enough to be useful for planning manipulative pushes. It is interesting to note that, with simulated data, the -GAE method does better than the -GA method, whereas, with real data, the converse is true. We suggest that this discrepancy is again due to inaccuracies in the visual tracking system.

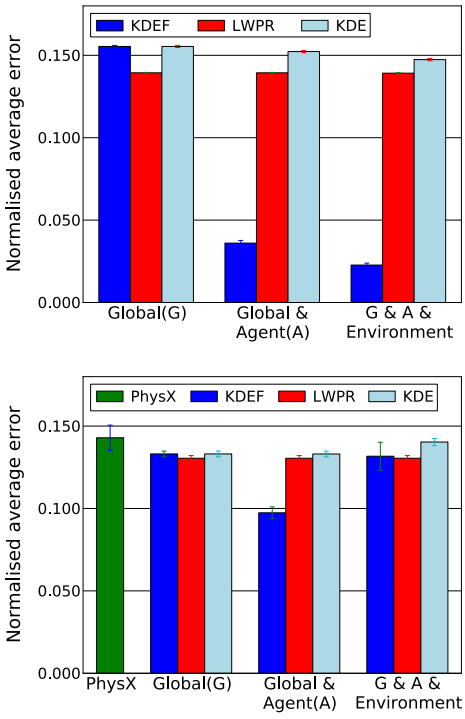

Fig. 15. Experiment A: Generalisation to novel action. Trained on forward push on polyflap, tested on backward push, for simulated (top) and real data (bottom). Comparative performance of predictors vs. information utilised (global/agent/environment), as measured by the normalised average error E_{av}^{norm}.

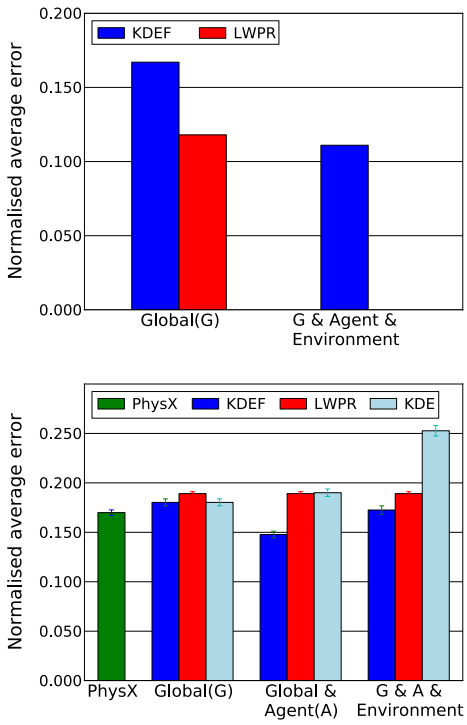

Fig. 16. Experiment S1: Generalisation to novel shape. Trained on polyflap, tested on box, for simulated (top) and real data (bottom). Comparative performance of predictors vs. information utilised, as measured by the normalised average error E_{av}^{norm}. (The simulated results are taken from [3].)

C. Generalising to make predictions about previously unencountered object shapes

Experiments S1 and S2 (S stands for "shape") are designed to test the extent to which learned predictors can generalise to make sensible predictions about novel objects which have very different shapes from those encountered during training.

1) Training on a polyflap and testing on a box: In Experiment S1, the predictors are trained on a polyflap object (a training set of 900 forward pushes), and then tested on a box object (a test set of 200 forward pushes). The experiment has been carried out in simulation (for a subset of methods taking the results from [3]), and with real robot data. Example frames from the experiments are shown in Figure 21. As before, we use Gaussian kernels, with quaternion and Euler angle parameterisations for KDE and LWPR methods respectively, and test performance for KDEF (i.e. factored density), LWPR and the single-factor KDE (KDE-GA and KDE-GAE). The normalised average error E_{av}^{norm} is shown in Figure 16; more detailed results are presented in Tables XI and XII.

As with Experiment A, for the simulated data (assuming perfect visual tracking) the KDEF-GAE performs best, suggesting that this technique will dominate with a high-precision vision system. For the tests with real data, all techniques suffer from poor accuracy. However, the KDEF methods (KDEF-GA and KDEF-GAE) give physically plausible predictions, while the LWPR technique predicts that the box will remain stationary throughout the pushing sequence.

2) Training on a box and a cylinder, and testing on two rigidly connected cylinders: In Experiment S2, we make use of a "double cylinder" object, which consists of two cylinders which have been rigidly joined together This poses a considerable challenge for many learned predictors. For example, a predictor that uses only global information when trained on a single cylinder will predict that cylinders tend to rotate – such a predictor may predict that the double cylinder object should also rotate, when in practice it is constrained to slide without rolling.

In this experiment, we have used a mixed training set, containing 500 examples of forward pushes on a box (which tends to either slide or topple over) and 100 examples of forward pushes on a cylinder (which tends to roll by rotating about its axis). The predictors are then tasked with making predictions about the motion of the novel double cylinder shape.

Example frames are shown in Figures 22-23, and test results

are presented in Figure 17 and Tables XIII and XIV. We see that KDEF performs best, according to the normalised average error measure.

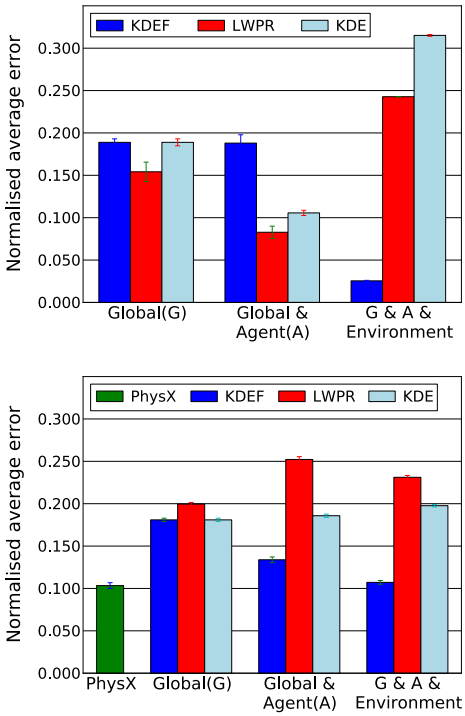

Fig. 17. Experiment S2: Generalisation to novel shape. Trained on cylinder and box, tested on double-cylinder, for simulated (top) and real data (bottom). Comparative performance of predictors vs. information utilised, as measured by the normalised average error E_{av}^{norm}.

3) Training and testing with downward pushes on angled-polyflaps: For Experiment S3 (performed only in simulation) all training and test data involve polyflaps constructed from two square plates. A set of polyflaps is generated by randomly varying the angle at which the two square plates are connected along a common edge. This shape variation is very significant, dramatically changing the finger-object contact relations. As Figure 24 demonstrates, a downwards push from above might cause the entire object to move either leftwards or rightwards, depending on small changes in the angle of the plates. Figure 18 shows the normalised average error E_{av}^{norm} for methods KDEF, LWPR and KDE-GA, using either global only, or global plus agent contact information, with further detail in Tables XV and XVI.

The experiment reveals that adding agent contact information significantly reduces the test error, for the KDEF-GA and KDE-GA methods. However, in this instance LWPR was not able to improve performance when given the extra information. as shown in Figure 25. If the method does not encode information about the contact variability, it may not generalise well in situations where small changes in shape can cause significant and qualitative changes in the resulting motion, even when the robotic push is the same. In contrast, the KDEF technique can accommodate this kind of shape generalisation. We consider this a form of "interpolative" generalisation task, in that the test and training shapes are qualitatively similar and the range of test shapes can be considered to be spanned by the range of training examples.

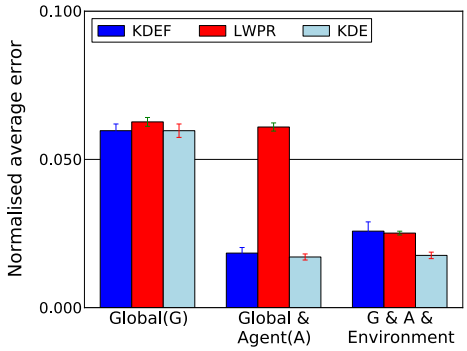

Fig. 18. Experiment S3: Interpolative generalisation to novel shape. Trained with downward pushes on angled polyflaps, tested on similar polyflaps, using simulated data. Comparative performance of predictors vs. information utilised, as measured by the normalised average error E_{av}^{norm}.

XI. Discussion

The results presented are roughly grouped according to the three hypotheses. In the experiments L1, L2 and L3 the ability to learn from real data on a variety of objects was examined. Hypothesis 1 was that learning can outperform physics engine based prediction and is supported by Figure 8. In addition Tables III-VIII and Figures 9-12 show that the quaternion encoding for the KDE approach produces a lower error in the predicted trajectories than either Euler angles or the encoding based on the von Mises-Fisher distribution; and that LWPR worsens in performance as it has more input dimensions. This is important because it shows that the regression approach, while it can learn good predictions, is not able to take advantage of additional information in the way that the product of densities approach does. Note that LWPR does not make use of the factorisation employed in the KDE case, but that instead it is given the input information that the 2 and 3 factor KDE learners have in order to make sure that all fair comparisons are made. All these experiments involved interpolative generalisation with respect to actions, as the actions in the test set are generated randomly from a continuous space, and thus differ from those in the training set while being drawn from the same distribution.

Experiment A tested Hypothesis 2 – that learning can generalise extrapolatively to novel actions. Tables IX and X show that this is the case for 2 and 3 factor versions of the KDE approach, but not for a single global factor for either KDE or for any of the treatments using LWPR. This shows that the hypothesis is correct, and supports our intuition as to why having a product of factors will help (Figure 7). This is shown particularly clearly by Figures 19 and 20.

Experiments S1, S2 and S3 tested Hypothesis 3 – that learning can generalise interpolatively and extrapolatively to novel shapes. Tables XI to XVI show that errors are larger for extrapolative generalisation than interpolative generalisation. The Tables suggest, for both simulated and real data, that KDE

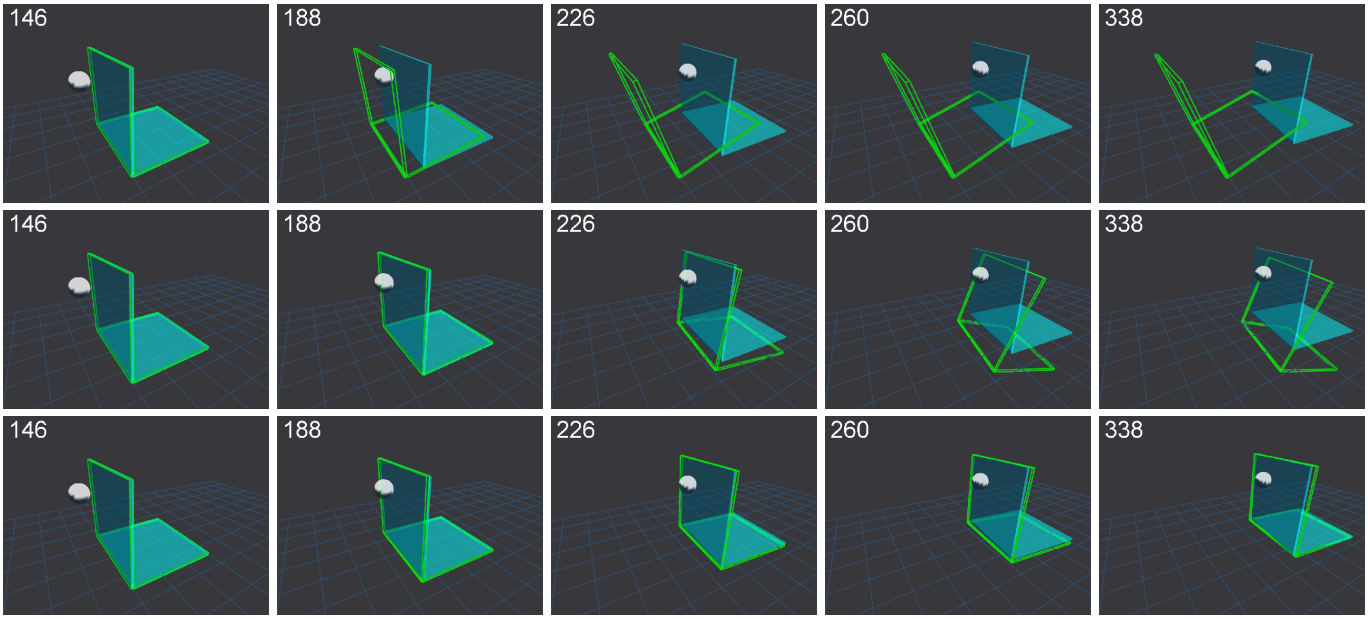

Fig. 19. Experiment A (simulation): Green outline shows predictions (from top row to bottom row) by KDEF-G/quat, KDEF-GA/quat, and KDEF-GAE/quat, compared to simulated 'ground truth' (in cyan). These predictions illustrate the rationale for extra contact information presented in Figure 7. (The frame number is shown in the top left of each image.)

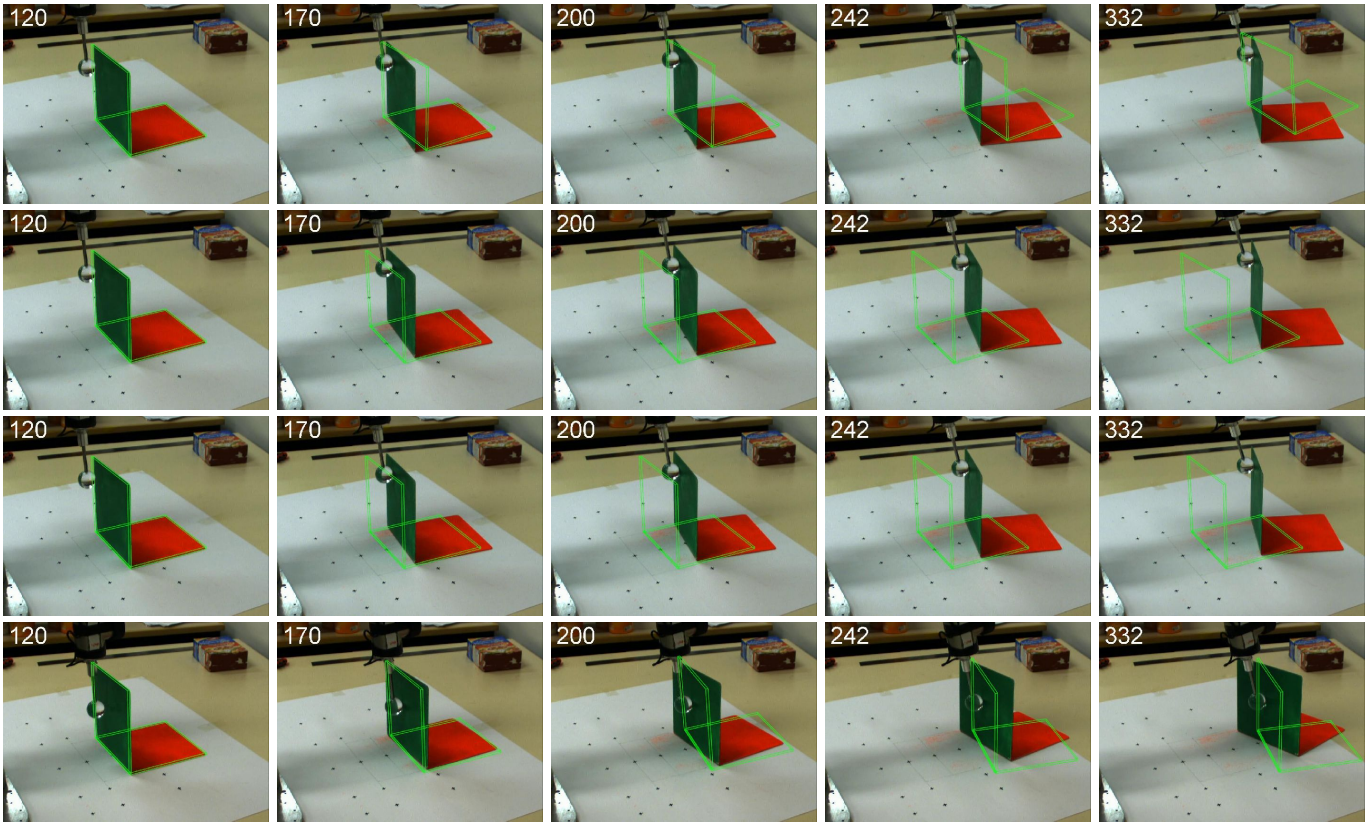

Fig. 20. Experiment A (real data): Green outline shows predictions (from top row to bottom row) by KDEF-GA/quat, KDEF-G/quat, and LWPR-G for one trial and by KDEF-GA/quat on another trial in which the polyflap rotates in a different direction as it slides. Note that the KDEF-G/quat and LWPR-G methods predict that the robot finger passes through the polyflap. (The frame number is shown in the top left of each image.)

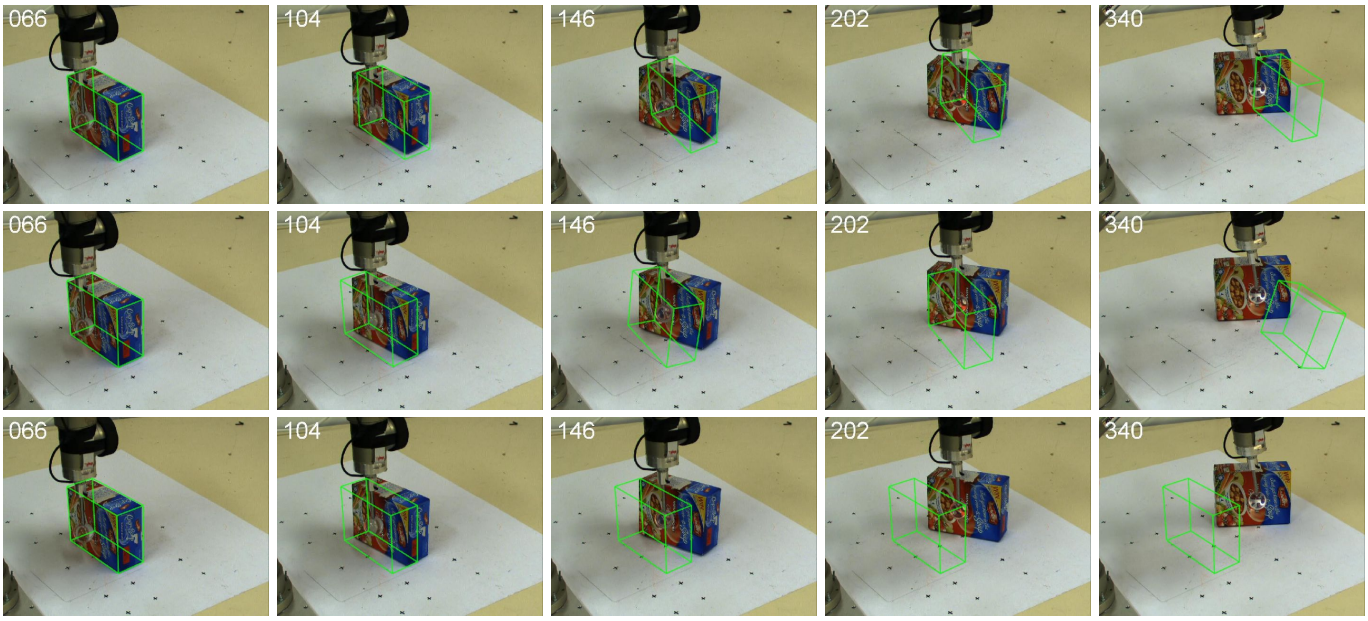

Fig. 21. Experiment S1: extrapolative generalisation to novel shape (real data): Green outline shows predictions (from top row to bottom row) by KDEF-GA/quat, KDEF-G/quat, and LWPR-G for one trial. Note that the KDEF-G/quat and LWPR-G methods predict that the robot finger moves into the box. (The frame number is shown in the top left of each image.)

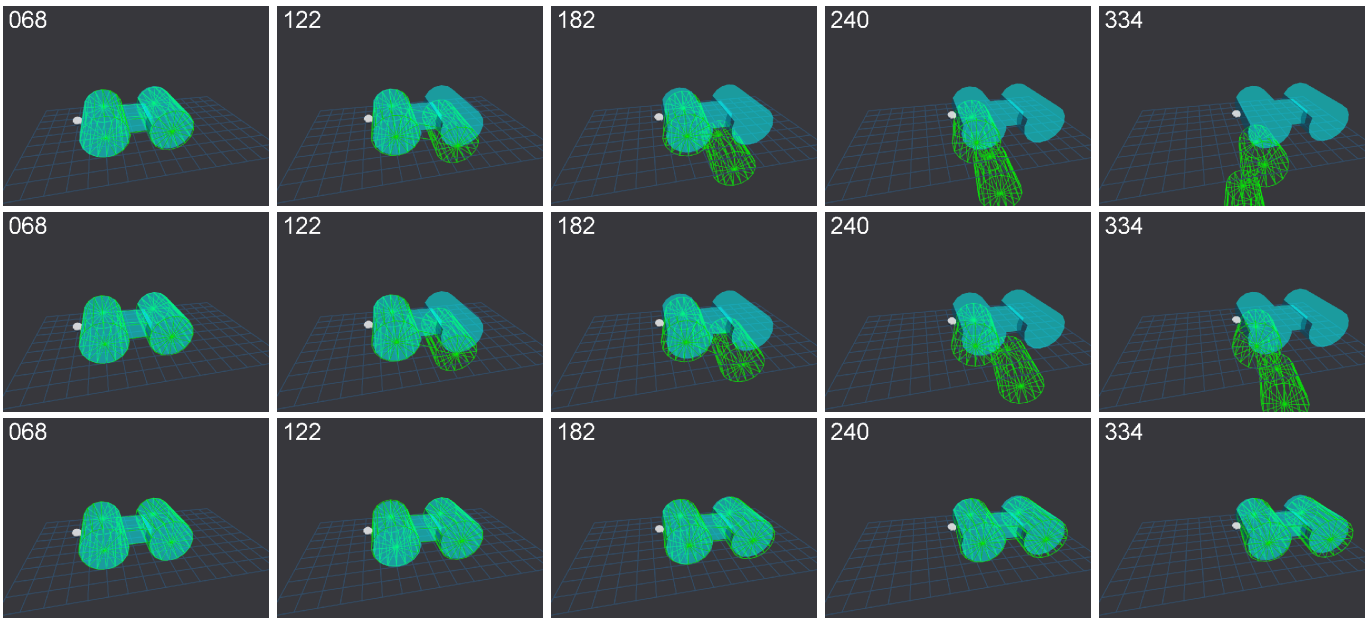

Fig. 22. Experiment S2: extrapolative generalisation to novel shape (simulation): Green outline shows predictions (from top row to bottom row) by KDEF-G/quat, KDEF-GA/quat, and KDEF-GAE/quat, compared to simulated 'ground truth' (in cyan). Note that the -G and -GA methods predict that the object moves into and through the ground plane. (The frame number is shown in the top left of each image.)

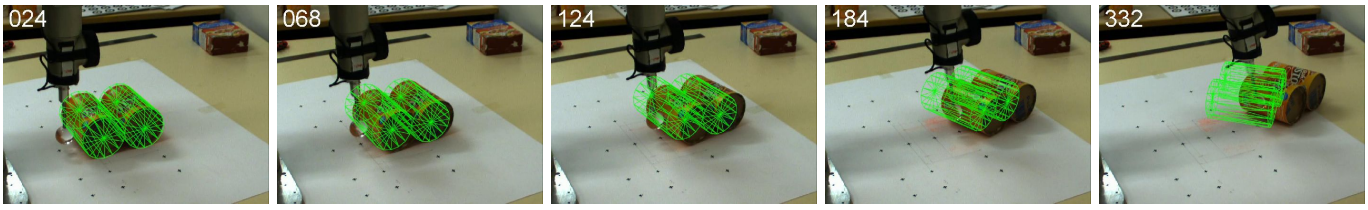

Fig. 23. Experiment S2: extrapolative generalisation to novel shape (real data): Green outline shows prediction by KDEF-GAE/quat. (The frame number is shown in the top left of each image.)

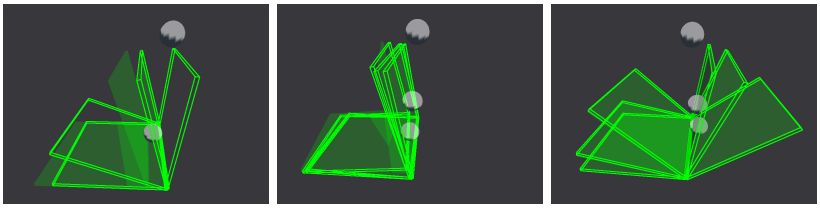

Fig. 24. Experiment S3. These images show simulations in which a fingertip (grey ball) descends onto a polyflap from above. Small variations in the fold angle of the polyflap lead to radically altered object motions, even given identical pushes from the fingertip. When the polyflap is folded acutely, the fingertip will push it to the left, whereas with a fold angle greater than ninety degrees, the same robotic push will move the polyflap to the right. This effect can be extremely difficult for conventional learning approaches to capture.

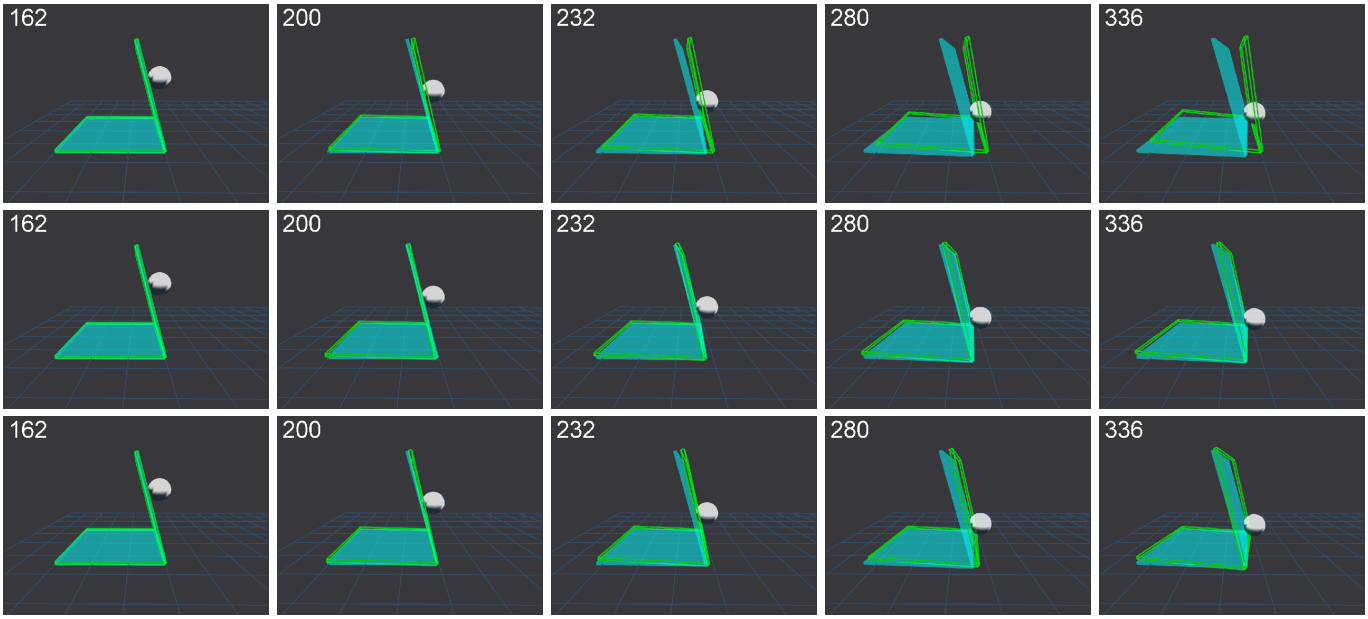

Fig. 25. Experiment S3 (simulation) reveals limitations of the methods using only global information, which fail to predict the motion of an angled polyflap when subjected to a downward push. The augmented input methods can cope well with this kind of shape variation. Green outline shows predictions (from top row to bottom row) by KDEF-G/quat, KDEF-GA/quat, and LWPR-GA, compared to simulated 'ground truth' (in cyan). (The frame number is shown in the top left of each image.)

outperforms LWPR (although note the one exception to this in Table XII). Illustrative cases are shown in Figures 21 to 25.

Finally it is important that the predictions are what a viewer would judge as qualitatively good, ie. that the direction of travel, and the qualitative motions of the objects are correct. This is supported for Experiments L1-3 by Figures 10 to 14, where it can be seen that the paths followed by the predictions and the objects match, even when the precise timing of the trajectories sometimes differs. In Figures 19 to 23 it can be seen that predictions on simulated data are more accurate that those on real data, but that although divergence occurs between the predictions and the real data the predictions produced still show the correct direction of travel, and are physically plausible except for a degree of object-finger interpenetration in Figure 23. It is worth noting that only the KDE learners produced plausible predictions for shape generalisation and action generalisation. In every case for experiments A and S1, the regression prediction was that the object failed to move, with the finger passing straight through the object.

In conclusion we have established in this paper that predictions of object behaviour can be learned using a variety of machine learning techniques; that these are better than the results of physics engine simulation; that they can generalise to new actions and new shapes, and that they produce plausible predictions on real data. We have argued in the paper that extrapolative generalisation, while being an extremely challenging problem, is possible if the correct representation is used. We have further argued that a product of factors approach, using density estimation techniques for each factor, is able to encode compromise predictions that are physically plausible and qualitatively correct. The experiments in the paper have supported this conclusion, and shown that for both extrapolative generalisation and interpolative generalisation with respect to shape, a regression framework is inadequate.

APPENDIX
TABLES OF RESULTS

Predictor	$N = 7$	$N = 70$	$N = 700$
KDEF-G euler	0.112±0.002	0.069±0.002	0.055±0.002
KDEF-G quat	0.114±0.003	0.068±0.002	**0.049**±0.002
KDEF-G vmf	0.143±0.004	0.082±0.003	0.057±0.002
LWPR-G euler	0.127±0.002	0.097±0.002	0.059±0.002
KDEF-GA euler	0.115±0.003	0.069±0.002	0.054±0.002
KDEF-GA quat	0.110±0.003	0.063±0.002	**0.044**±0.002
KDEF-GA vmf	0.123±0.003	0.079±0.002	0.064±0.002
LWPR-GA euler	0.125±0.002	0.124±0.002	0.068±0.002
KDEF-GAE euler	0.129±0.003	0.093±0.003	0.083±0.003
KDEF-GAE quat	0.125±0.003	0.082±0.003	**0.062**±0.002
KDEF-GAE vmf	0.122±0.002	0.094±0.002	0.081±0.002
LWPR-GAE euler	0.125±0.002	0.125±0.002	0.069±0.002
KDE-GA euler	0.108±0.002	0.068±0.002	0.053±0.002
KDE-GA quat	0.109±0.002	0.064±0.002	**0.049**±0.002
KDE-GA vmf	0.121±0.003	0.078±0.003	0.062±0.002
KDE-GAE euler	0.120±0.002	0.100±0.002	0.090±0.002
KDE-GAE quat	0.119±0.002	0.099±0.002	0.087±0.002
KDE-GAE vmf	0.122±0.002	0.095±0.002	**0.087**±0.002
PhysX		0.144±0.003	

TABLE III
EXPERIMENT L1: FORWARD PUSH ON A POLYFLAP, TRAINED ON REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. NUMBER OF TRAINING SAMPLES N. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR $E_{av}^{norm} \pm$ STANDARD ERROR.

Predictor	$N = 7$	$N = 70$	$N = 700$
KDEF-G euler	0.211±0.005	0.126±0.005	0.099±0.004
KDEF-G quat	0.220±0.006	0.128±0.006	**0.087**±0.004
KDEF-G vmf	0.311±0.013	0.175±0.008	0.113±0.005
LWPR-G euler	0.229±0.004	0.191±0.004	0.127±0.003
KDEF-GA euler	0.231±0.007	0.131±0.005	0.099±0.005
KDEF-GA quat	0.227±0.008	0.118±0.006	**0.078**±0.004
KDEF-GA vmf	0.262±0.007	0.163±0.005	0.129±0.005
LWPR-GA euler	0.226±0.004	0.226±0.004	0.128±0.004
KDEF-GAE euler	0.264±0.008	0.183±0.007	0.166±0.007
KDEF-GAE quat	0.259±0.008	0.173±0.009	**0.113**±0.005
KDEF-GAE vmf	0.266±0.005	0.213±0.005	0.174±0.004
LWPR-GAE euler	0.226±0.004	0.227±0.004	0.127±0.003
KDE-GA euler	0.206±0.005	0.122±0.004	0.092±0.004
KDE-GA quat	0.209±0.005	0.114±0.004	**0.089**±0.004
KDE-GA vmf	0.244±0.007	0.152±0.006	0.125±0.006
KDE-GAE euler	0.226±0.004	0.192±0.004	0.174±0.005
KDE-GAE quat	0.222±0.004	0.187±0.004	0.173±0.005
KDE-GAE vmf	0.234±0.004	0.180±0.004	**0.166**±0.004
PhysX		0.293±0.007	

TABLE IV
EXPERIMENT L1: FORWARD PUSH ON A POLYFLAP, TRAINED ON REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. NUMBER OF TRAINING SAMPLES N. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED FINAL ERROR $E_{f}^{norm} \pm$ STANDARD ERROR.

Predictor	$N = 4$	$N = 40$	$N = 400$
KDEF-G euler	0.127±0.004	0.079±0.003	0.061±0.003
KDEF-G quat	0.118±0.003	0.074±0.003	**0.059**±0.002
KDEF-G vmf	0.128±0.003	0.083±0.003	0.066±0.003
LWPR-G euler	0.216±0.005	0.169±0.003	0.118±0.003
KDEF-GA euler	0.131±0.004	0.078±0.003	0.060±0.003
KDEF-GA quat	0.118±0.003	0.073±0.003	**0.057**±0.002
KDEF-GA vmf	0.159±0.006	0.108±0.003	0.097±0.002
LWPR-GA euler	0.190±0.002	0.182±0.003	0.127±0.003
KDEF-GAE euler	0.131±0.003	0.085±0.003	0.065±0.003
KDEF-GAE quat	0.116±0.003	0.078±0.003	**0.065**±0.003
KDEF-GAE vmf	0.118±0.002	0.097±0.002	0.086±0.002
LWPR-GAE euler	0.190±0.002	0.181±0.002	0.136±0.003
KDE-GA euler	0.108±0.003	0.070±0.003	0.057±0.002
KDE-GA quat	0.107±0.003	0.069±0.003	**0.057**±0.002
KDE-GA vmf	0.113±0.003	0.069±0.003	0.058±0.002
KDE-GAE euler	0.152±0.003	0.143±0.003	0.161±0.003
KDE-GAE quat	0.160±0.002	0.205±0.003	0.253±0.002
KDE-GAE vmf	0.136±0.003	**0.111**±0.003	0.127±0.003
PhysX		0.171±0.003	

TABLE V
EXPERIMENT L2: FORWARD PUSH ON A BOX, TRAINED ON REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. NUMBER OF TRAINING SAMPLES N. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR $E_{av}^{norm} \pm$ STANDARD ERROR.

Predictor	$N = 4$	$N = 40$	$N = 400$
KDEF-G euler	0.193±0.007	0.121±0.005	0.090±0.005
KDEF-G quat	0.173±0.005	0.112±0.005	**0.086**±0.004
KDEF-G vmf	0.178±0.005	0.125±0.005	0.100±0.005
LWPR-G euler	0.325±0.005	0.257±0.005	0.189±0.004
KDEF-GA euler	0.228±0.011	0.123±0.005	0.089±0.004
KDEF-GA quat	0.186±0.006	0.109±0.004	**0.083**±0.004
KDEF-GA vmf	0.294±0.016	0.173±0.005	0.152±0.004
LWPR-GA euler	0.299±0.003	0.285±0.003	0.201±0.005
KDEF-GAE euler	0.221±0.007	0.139±0.006	0.101±0.005
KDEF-GAE quat	0.193±0.009	0.122±0.005	**0.101**±0.005
KDEF-GAE vmf	0.182±0.005	0.149±0.004	0.129±0.003
LWPR-GAE euler	0.302±0.003	0.288±0.003	0.215±0.004
KDE-GA euler	0.159±0.005	0.105±0.004	**0.082**±0.004
KDE-GA quat	0.161±0.005	0.104±0.004	0.083±0.004
KDE-GA vmf	0.165±0.005	0.104±0.004	0.084±0.004
KDE-GAE euler	0.218±0.007	0.170±0.004	0.181±0.005
KDE-GAE quat	0.271±0.004	0.377±0.006	0.509±0.006
KDE-GAE vmf	0.196±0.005	**0.149**±0.004	0.158±0.004
PhysX		0.206±0.005	

TABLE VI
EXPERIMENT L2: FORWARD PUSH ON A BOX, TRAINED ON REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. NUMBER OF TRAINING SAMPLES N. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED FINAL ERROR $E_{f}^{norm} \pm$ STANDARD ERROR.

Predictor	$N = 2$	$N = 20$	$N = 200$
KDEF-G euler	0.111±0.004	0.075±0.003	0.063±0.003
KDEF-G quat	0.113±0.003	0.075±0.003	**0.059**±0.003
KDEF-G vmf	0.151±0.004	0.088±0.004	0.071±0.003
LWPR-G euler	0.263±0.003	0.155±0.003	0.077±0.003
KDEF-GA euler	0.084±0.002	0.058±0.002	0.052±0.002
KDEF-GA quat	0.085±0.002	0.056±0.002	**0.047**±0.002
KDEF-GA vmf	0.136±0.002	0.112±0.002	0.109±0.003
LWPR-GA euler	0.251±0.001	0.206±0.002	0.105±0.002
KDEF-GAE euler	0.085±0.003	0.053±0.002	0.050±0.002
KDEF-GAE quat	0.083±0.003	0.055±0.002	**0.047**±0.002
KDEF-GAE vmf	0.110±0.003	0.080±0.002	0.065±0.002
LWPR-GAE euler	0.251±0.001	0.218±0.002	0.116±0.003
KDE-GA euler	0.105±0.003	0.077±0.003	0.068±0.003
KDE-GA quat	0.107±0.004	0.079±0.004	**0.065**±0.003
KDE-GA vmf	0.128±0.004	0.085±0.004	0.092±0.004
KDE-GAE euler	0.107±0.003	0.075±0.003	0.071±0.003
KDE-GAE quat	0.105±0.003	0.080±0.004	**0.068**±0.003
KDE-GAE vmf	0.131±0.004	0.086±0.004	0.091±0.004
PhysX	0.271±0.001		

TABLE VII
EXPERIMENT L3: FORWARD PUSH ON A CYLINDER, TRAINED ON REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. NUMBER OF TRAINING SAMPLES N. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR $E_{av}^{norm}\pm$ STANDARD ERROR.

Predictor	$N = 2$	$N = 20$	$N = 200$
KDEF-G euler	0.209±0.009	0.133±0.008	0.103±0.006
KDEF-G quat	0.223±0.008	0.133±0.007	**0.095**±0.006
KDEF-G vmf	0.281±0.010	0.170±0.008	0.123±0.006
LWPR-G euler	0.452±0.007	0.268±0.004	0.097±0.006
KDEF-GA euler	0.139±0.005	0.085±0.004	0.077±0.005
KDEF-GA quat	0.144±0.005	0.085±0.004	**0.068**±0.004
KDEF-GA vmf	0.215±0.004	0.181±0.004	0.187±0.004
LWPR-GA euler	0.421±0.001	0.352±0.003	0.173±0.005
KDEF-GAE euler	0.142±0.007	0.081±0.005	0.073±0.005
KDEF-GAE quat	0.143±0.007	0.082±0.005	**0.068**±0.005
KDEF-GAE vmf	0.185±0.007	0.138±0.004	0.110±0.004
LWPR-GAE euler	0.421±0.001	0.373±0.003	0.198±0.006
KDE-GA euler	0.199±0.008	0.134±0.007	0.112±0.006
KDE-GA quat	0.201±0.009	0.137±0.008	**0.105**±0.006
KDE-GA vmf	0.236±0.009	0.159±0.008	0.154±0.007
KDE-GAE euler	0.206±0.009	0.131±0.007	0.118±0.006
KDE-GAE quat	0.201±0.008	0.141±0.008	**0.112**±0.006
KDE-GAE vmf	0.247±0.009	0.161±0.009	0.152±0.007
PhysX	0.217±0.001		

TABLE VIII
EXPERIMENT L3: FORWARD PUSH ON A CYLINDER, TRAINED ON REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. NUMBER OF TRAINING SAMPLES N. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED FINAL ERROR $E_{f}^{norm}\pm$ STANDARD ERROR.

		Information Utilised		
Predictor	data	Global (G)	G & Agent (A)	G & A & Env
KDEF	sim	0.155±0.001	0.036±0.002	**0.023**±0.001
LWPR	sim	0.139±0.001	0.139±0.001	0.139±0.001
KDE	sim	n/a	0.152±0.001	0.147±0.001
KDEF	real	0.133±0.002	**0.097**±0.004	0.132±0.008
LWPR	real	0.130±0.002	0.130±0.002	0.130±0.002
KDE	real	n/a	0.133±0.002	0.140±0.002
PhysX	real	0.143±0.008		

TABLE IX
EXPERIMENT A: GENERALISATION TO NOVEL ACTION. TRAINED ON FORWARD PUSH ON POLYFLAP, TESTED ON BACKWARD PUSH, FOR SIMULATED AND REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR $E_{av}^{norm}\pm$ STANDARD ERROR.

		Information Utilised		
Predictor	data	Global (G)	G & Agent (A)	G & A & Env
KDEF	sim	0.397±0.002	0.117±0.009	**0.055**±0.003
LWPR	sim	0.360±0.001	0.360±0.001	0.359±0.001
KDE	sim	n/a	0.392±0.002	0.381±0.002
KDEF	real	0.287±0.003	**0.191**±0.01	0.297±0.022
LWPR	real	0.279±0.003	0.279±0.003	0.279±0.003
KDE	real	n/a	0.287±0.003	0.321±0.006
PhysX	real	0.284±0.016		

TABLE X
EXPERIMENT A: GENERALISATION TO NOVEL ACTION. TRAINED ON FORWARD PUSH ON POLYFLAP, TESTED ON BACKWARD PUSH, FOR SIMULATED AND REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED FINAL ERROR $E_{f}^{norm}\pm$ STANDARD ERROR.

		Information Utilised		
Predictor	data	Global (G)	G & Agent (A)	G & A & Env
KDEF	sim	0.167	n/a	**0.111**
LWPR	sim	0.118	n/a	n/a
KDEF	real	0.180±0.004	**0.148**±0.003	0.173±0.004
LWPR	real	0.189±0.002	0.189±0.002	0.189±0.002
KDE	real	n/a	0.191 ± 0.004	0.254 ± 0.006
PhysX	real	0.170 ± 0.003		

TABLE XI
EXPERIMENT S1: GENERALISATION TO NOVEL SHAPE. TRAINED ON POLYFLAP, TESTED ON BOX, FOR SIMULATED AND REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR $E_{av}^{norm}\pm$ STANDARD ERROR.

		Information Utilised		
Predictor	data	Global (G)	G & Agent (A)	G & A & Env
KDEF	sim	0.429	n/a	0.272
LWPR	sim	**0.233**	n/a	n/a
KDEF	real	0.302±0.007	**0.282**±0.009	0.321±0.014
LWPR	real	0.299±0.003	0.299±0.003	0.299±0.003
KDE	real	n/a	0.294±0.007	0.477±0.011
PhysX	real	0.195±0.005		

TABLE XII
EXPERIMENT S1: GENERALISATION TO NOVEL SHAPE. TRAINED ON POLYFLAP, TESTED ON BOX, FOR SIMULATED AND REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED FINAL ERROR $E_{f}^{norm}\pm$ STANDARD ERROR.

		Information Utilised		
Predictor	data	Global (G)	G & Agent (A)	G & A & Env
KDEF	sim	0.189±0.004	0.188±0.010	**0.025**±0.001
LWPR	sim	0.154±0.011	0.083±0.007	0.243±0.001
KDE	sim	n/a	0.105±0.003	0.315±0.001
KDEF	real	0.181±0.002	0.134±0.003	**0.107**±0.002
LWPR	real	0.200±0.002	0.252±0.003	0.231±0.002
KDE	real	n/a	0.186±0.002	0.198±0.001
PhysX	real	0.103±0.003		

TABLE XIII
EXPERIMENT S2: GENERALISATION TO NOVEL SHAPE. TRAINED ON CYLINDER AND BOX, TESTED ON DOUBLE-CYLINDER, FOR SIMULATED AND REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR $E_{av}^{norm}\pm$ STANDARD ERROR.

		Information Utilised		
Predictor	data	Global (G)	G & Agent (A)	G & A & Env
KDEF	sim	0.500±0.008	0.324±0.017	**0.056**±0.001
LWPR	sim	0.265±0.018	0.153±0.012	0.441±0.001
KDE	sim	n/a	0.302±0.008	0.691±0.004
KDEF	real	0.280±0.003	0.198±0.006	**0.161**±0.004
LWPR	real	0.303±0.005	0.381±0.004	0.361±0.003
KDE	real	n/a	0.287±0.003	0.311±0.002
PhysX	real	0.185±0.006		

TABLE XIV
EXPERIMENT S2: GENERALISATION TO NOVEL SHAPE. TRAINED ON CYLINDER AND BOX, TESTED ON DOUBLE-CYLINDER, FOR SIMULATED AND REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED FINAL ERROR $E_f^{norm} \pm$ STANDARD ERROR.

	Information Utilised		
Predictor	Global (G)	G & Agent (A)	G & A & Env
KDEF	0.060±0.002	0.018±0.002	0.026±0.003
LWPR	0.063±0.001	0.061±0.001	0.025±0.001
KDE	n/a	**0.017**±0.001	0.018±0.001

TABLE XV
EXPERIMENT S3: INTERPOLATIVE GENERALISATION TO NOVEL SHAPE. TRAINED WITH DOWN PUSHES ON ANGLED POLYFLAPS, TESTED ON SIMILAR POLYFLAPS, USING SIMULATED DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR E_{av}^{norm}.

	Information Utilised		
Predictor	Global (G)	G & Agent (A)	G & A & Env
KDEF	0.163±0.005	0.052±0.008	0.106±0.014
LWPR	0.162±0.004	0.158±0.004	0.061±0.002
KDE	n/a	**0.034**±0.003	0.036±0.003

TABLE XVI
EXPERIMENT S3: INTERPOLATIVE GENERALISATION TO NOVEL SHAPE. TRAINED WITH DOWN PUSHES ON ANGLED POLYFLAPS, TESTED ON SIMILAR POLYFLAPS, USING SIMULATED DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED FINAL ERROR E_f^{norm}.

We gratefully acknowledge the support of EU-FP7-IST grants Nos 248273 (GeRT) and 215181 (CogX).

REFERENCES

[1] M. Kopicki, J. Wyatt, and R. Stolkin, "Prediction learning in robotic pushing manipulation," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, 2009, pp. 1–6.
[2] M. Kopicki, "Prediction learning in robotic manipulation," Ph.D. dissertation, University of Birmingham, 2010.
[3] M. Kopicki, S. Zurek, R. Stolkin, T. Mörwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA11)*, May 2011.
[4] 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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA11)*, May 2011.
[5] M. T. Mason, "Manipulator grasping and pushing operations," PhD Thesis, MIT, 1982.
[6] K. Lynch, "The mechanics of fine manipulation by pushing," in *IEEE Int. Conf. on Robotics and Automation*, 1992, pp. 2269–2276.
[7] M. Peshkin and A. Sanderson, "The motion of a pushed, sliding workpiece," *IEEE Journal on Robotics and Automation*, vol. 4, pp. 569–598, 1988.
[8] 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*, 2006, pp. 637–642.
[9] E. Krotkov, "Perception of material properties by robotic probing: Preliminary investigations," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Montreal*, 1995, pp. 88–94.
[10] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA'03*, vol. 3, 2003.
[11] 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.
[12] 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*, 2007, pp. 110–115.
[13] K. M. Haruno M., Wolpert DM., "Mosaic model for sensorimotor learning and control," *Neural Computation*, vol. 13, pp. 2201–2220, 2001.
[14] A. Stoytchev, "Learning the affordances of tools using a behavior-grounded approach," *Lecture Notes in Artificial Intelligence (LNAI)*, vol. 4760, pp. 140–158, 2008.
[15] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.
[16] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.
[17] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *The Journal of Machine Learning Research*, vol. 9, pp. 623–626, 2008.
[18] D. W. Scott and S. R. Sain, *"Multi-Dimensional Density Estimation"*. Elsevier, 2004, pp. 229–263.
[19] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. New York: Dover, 1965.
[20] R. Detry, "Learning of multi-dimensional, multi-modal features for robotic grasping," Ph.D. dissertation, University of Liege, 2010.
[21] R. Fisher, "Dispersion on a sphere," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 217, no. 1130, p. 295, 1953.
[22] D. Knuth, *Searching and Sorting, The Art of Computer Programming, vol. 3*. Addison-Wesley, 1998.
[23] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.
[24] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[25] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.
[26] G. Marsaglia and T. A. Bray, "A convenient method for generating normal variables," *Siam Review*, vol. 6, no. 3, pp. 260–264, 1964.
[27] Neuronics AG, "Katana user manual and technical description," 2004. [Online]. Available: http://www.neuronics.ch
[28] 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.
[29] NVIDIA PhysX, "Physics simulation for developers," 2009. [Online]. Available: http://http://developer.nvidia.com/physx

Marek Kopicki received an MSc in Theoretical Physics at Adam Mickiewicz University in Poznan, Poland. He then obtained an MSc in Advanced Computer Science (2004) and his PhD in Computer Science (2010) from the University of Birmingham where he has worked as a Research Fellow on the FP7 projects CogX and GeRT. His interests include machine learning, and robotic manipulation.

Sebastian Zurek is at the Department of Computer Science, University of Birmingham, UK. He has a B.A. degree in Theoretical Physics from the University of Cambridge, and was a researcher in the Theory of Condensed Matter Group at the Cavendish Laboratory, where he obtained his Ph.D. in 1991. He also has an M.Res. in Brain Imaging from the University of Birmingham. His interests include machine learning and models of human cognition.

Rustam Stolkin is currently Senior Research Fellow in robotics, in the School of Computer Science, University of Birmingham, UK. Prior to coming to Birmingham, he was a Research Assistant Professor at Stevens Institute of Technology, USA. Rustam holds undergraduate and masters degrees in Engineering Science from University of Oxford, and a PhD in Robot Vision, undertaken between University College London and the UK imaging industry. His research interests include robotics, computer vision, sensor systems, and science and engineering education.

Thomas Mörwald received an MSc in Mechatronics at Johannes Kepler University in Linz, Austria. He is a Ph.D. candidate at the Automation and Control Institute of Vienna University of Technology where he has worked as Research Fellow on the FP7 project CogX. His research interests include computer vision, computer graphics and robotics.

Jeremy L. Wyatt obtained a BA in Theology from the University of Bristol, an MSc in Knowledge Based Systems from the University of Sussex, and his Ph.D. in Artificial Intelligence from the University of Edinburgh in 1997. Since then he has worked at the University of Birmingham, where he is Reader in Robotics and Artificial Intelligence. He has published more than 70 refereed articles. His interests include machine learning and cognitive robotics.