IEEE Transactions on Cybernetics

# Learning Transferable Kinematic Models of the Motions of Manipulated Objects



| | |
|---|---|
| Journal: | *IEEE Transactions on Cybernetics* |
| Manuscript ID: | Draft |
| Manuscript Type: | Regular Paper |
| Date Submitted by the Author: | n/a |
| Complete List of Authors: | Kopicki, Marek; University of Birmingham, Computer Science<br>Zurek, Sebastian; University of Birmingham, Computer Science<br>Stolkin, Rustam; University of Birmingham, Mechanical Engineering<br>Wyatt, Jeremy; University of Birmingham, School of Computer Science;<br>Morwald, Thomas; TU Wien, ACIN |
| Keywords: | robotics, learning, prediction, manipulation |
| | |

# Learning transferable kinematic models of the motions 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 during manipulation. This is necessary to enable many manipulations to be planned. Physics simulators are commonly used for this purpose, but in practice model many kinds of object interaction poorly. An alternative explored here is for the robot to learn a motion model from data. The robot learns to predict the kinematic behaviour of a rigid body being pushed by a robot finger. To achieve this we first pose the problem in two machine learning frameworks: regression and density estimation. We then show empirically that both frameworks exhibit good interpolative generalisation performance with respect to shape and action. In addition learned predictors for specific objects outperform physics simulation in terms of accuracy of the trajectory. We also examine how to make these learned models capable of transfer learning, where learners are trained on one object or action and transfer well to different objects and actions. To achieve this it is necessary to add information about contacts involving the object. We also show how factorisation of the learning problem improves transfer performance. We present experiments using a physics simulator, and with a robot arm equipped with a rigid finger. We test on hundreds of pushes of four different categories of objects.**

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

## I. Introduction

For a robot to plan object manipulations it must be able to predict the effects of its actions. This paper shows how predictors can be learned from data, and how they can be transfered from one type of object or action, to another. This paper shows that such transferable predictive models (forward models) can be learned for a variety of objects, in the case when the behaviour can be adequately described by a kinematic model. In particular we show how to pose the learning problem faced by the robot in two machine learning frameworks (density estimation and regression); why information about contacts needs to be represented to enable learning transfer; and how to take advantage of this additional information by structuring the learning problem.

The problems tackled here differ from those 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 critical to how it moves. One of the advantages of a robot learning approach over using a physics engine is that the learners can capture the effect on object motion of unobserved variables such as mass distribution and frictional coefficients. In addition the real robot can explore different objects in a particular environment to improve its predictive models over time. The experimental domain is robot push manipulation, in which there is a single contact between the robot and object. This makes action effects particularly hard to predict since the object is free to move independently of the manipulator by tipping or rolling. The prediction problem defined is to predict the precise trajectory of the object multiple steps (here 100-150 steps) into the future.

This paper extends our previous work [1], [2], [3], [4] in several ways. First, our experimental results are extended with additional objects. Second, the transfer learning experiments are conducted on a real robot and in simulation. Third we provide extensive characterisation of the parameter space for the three main algorithms presented. Fourth we show that good predictors can be learned using both regression and density estimation for a variety of real objects; that these object-specific models are significantly better at prediction than a physics engine; and that they allow interpolative generalisation with respect to action and shape. Fifth we show on real data that the approaches enable various degrees of learning transfer to novel actions, and to objects with novel shape. In summary, this paper supports three hypotheses. Hypothesis 1 is that a learning approach can outperform physics engines for prediction of rigid body motion for a variety of objects. Hypothesis 2 is that the learned predictions a) generalise interpolatively and b) show learning transfer with respect to the actions performed on the object. Hypothesis 3 is that the learned predictions a) generalise interpolatively and b) show learning transfer with respect to the shape of the objects.

The paper is laid out as follows. First Section II introduces representations of the motion of pushed objects. This enables a formal statement of the prediction problem in Section III and formulation in both regression and density estimation frameworks. Given this basic formulation Section IV describes why it is necessary to represent contacts between the manipulated object and other surfaces, and how this information can be incorporated into the problem formulation. Section V describes why factoring the representation for density estimation can be beneficial. Section VI then explains implementation details, and Section VII provides details of our experimental studies. Section VIII presents the results of our experimental studies. Section IX reviews related work. We finish with a discussion in Section X.

## II. Encoding rigid body kinematics

In this section we set up the basic notation required to pose the prediction problem in the next section. Without loss of

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.

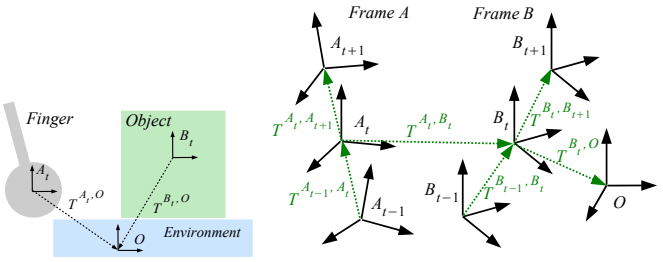


Fig. 1. (Left) 2D projection at time $t$ of a robotic finger with frame $A_t$, an object with frame $B_t$, and a ground plane with constant frame $O$. (Right) A system consisting of two interacting bodies with frames $A$ and $B$ in some constant environment with frame $O$ can be described by six rigid 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 $T^{B_t,B_{t+1}}$

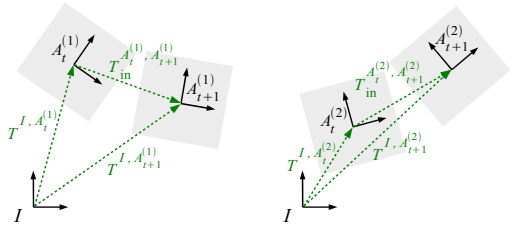

Fig. 2. In the above two scenes a pose change between time step $t$ and $t+1$ as observed in instantaneous object body frame $A^{(1)}$ and the same object in another instantaneous body frame $A^{(2)}$ given inertial frame $I$ are both the same. However because transformations $T^{I,A^{(1)}}$ and $T^{I,A^{(2)}}$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

generality we explain the notation using an example from our application domain (Figure 1). Three reference frames $A$, $B$ and $O$ sit in a 3-dimensional Cartesian space. Frame $A$ is attached to a robot finger which pushes an object with frame $B$, which in turn is placed on a table top with frame $O$.[1] While frame $O$ is fixed, $A$ and $B$ change in time and are observed at discrete time steps $..., t-1, t, t+1, ....$ Frame $X$ at time step $t$ is denoted $X_t$, and the rigid body transformation between a frame $X$ and a frame $Y$ is denoted by $T^{X,Y}$.

From classical mechanics we know that in order to predict the change in state of a rigid body, it is sufficient to know its mass, velocity and a net force applied to the body. Since our method will rely on learning from object trajectories we cannot assume any knowledge of the mass and applied forces. We can however, use the motion of a body over time to encode acceleration – an effect of the applied net force. We therefore use rigid body transformations of the interacting bodies through time (Figure 1). Given the additional assumption that the net force and the body mass are constant, two subsequent rigid body transformations $T^{B_{t-1},B_t}$ and $T^{B_t,B_{t+1}}$ give a complete description of the state of some body B (here the object) at time step $t$ in the absence of the other bodies. Adding the transformation $T^{B_t,O}$ to give a triple of transformations thus provides a complete description of the state of body B in the fixed frame $O$ (the stationary elements of the environment). Similarly, a second triple of transformations $T^{A_t,O}$, $T^{A_{t-1},A_t}$ and $T^{A_t,A_{t+1}}$ provides such a description for some other body (here the finger) with frame $A$.

The state of the overall system consisting of these two interacting bodies with frames $A$ and $B$ and the fixed environment $O$ can thus be described by six transformations as shown in Figure 1. Transformation $T^{A_t,O}$ can be replaced by a relative transformation $T^{A_t,B_t}$, explictly capturing the spatial relationship and thus any contacts between $A$ (finger) and $B$ (object).

We need to express these transformations in a way that supports generalised predictions. In general the behaviours of interacting bodies described by frames from Figure 1 are independent of any inertial frame [2]. Unfortunately, a naïve 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 2). 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 and convert to and from an inertial frame dependent transformation using similarity transforms.

Intuitively these align the inertial and body frames, perform the required transformation and then invert the transformation required to align them. In this manner, given the instantaneous object frame $A_t$ at time $t$, and the inertial frame dependent transformation $T_{in}^{A_t,A_{t+1}}$, one can obtain the body frame dependent transformation $T_{body}^{A_t,A_{t+1}}$:

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

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

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

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

We use this to store learned transformations in the body frame and then produce predictions in some inertial frame. In the rest of the paper 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:

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

## III. The Prediction Problem

We now describe the one-step and then the multi-step prediction problem. The one step prediction problem is formulated as follows: given that we know or observe the starting positions of the finger and object, and know the planned motion of the finger, $T^{A_t,A_{t+1}}$, predict the resulting immediate

[1] Although it is an abuse of notation for brevity we will use $A$, $B$ and $O$ to denote either the frame or the body to which it is attached. So we will talk both of the frame $B$ and the object $B$.

motion of the object, $T^{B_t,B_{t+1}}$. This is a problem of finding a function $f$:

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

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,[2] in the limit of infinitesimally small time steps. Furthermore, it can be approximately learned from observations for some small fixed time interval $\Delta t$ between time steps.

We focus on the kinematic regime, where robotic manipulations are performed relatively slowly. Hence we can assume quasi-static conditions, and ignore all frames at time $t-1$. This conveniently reduces the dimensionality of the problem, giving a simplified function $f_{qs}$:

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

Having solved the one-step prediction problem it is possible to solve the multi-step prediction problem. Given a predictor (either $f$ or $f_{qs}$), the initial state of the finger $T^{A_0,O}$ and object $T^{B_0,O}$, and knowing the trajectory of the finger $A_1, \dots A_T$ over $T$ time steps, one can predict the complete trajectory of the object $B_1, \dots B_T$, by simply iterating the predictions obtained from $f_{qs}$. That is, the output of the predictor at time $t$ is used as the input to the predictor for the next time step.

In principle it is straightforward to acquire a predictor $f$ or $f_{qs}$ by learning it from data. Given sufficient experience of object and finger trajectories we can perform 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 18 dimensions or more depending on the parameterisation of rotations (see Section VI-A).

As an alternative to learning the mapping (6) by regression, we can recast $f_{qs}$ as a conditional probability density (CPD) $p_{qs}$ over possible object motions $T^{B_t,B_{t+1}}$ [1]:

$$p_{qs}(T^{B_t,B_{t+1}} | T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}) \quad (7)$$

The learning problem is then posed as one of density estimation. This framework permits the modelling of differing outcomes caused by disturbances leading to bifurcations in the space of possible trajectories.

## IV. Representing Contacts

The functions proposed can adequately represent the behaviour of a specific pair of rigid bodies, but cannot generalise to other bodies. This is because the prediction problem as posed above does not encode information sufficient to determine the various contacts the bodies make with one another. This is problematic because the kinematic behaviour of an object can change radically as new contacts occur or change position. This is because each contact provides a kinematic constraint on the motion of the contacting bodies. This in turn means that even small changes in the relative position or shape of any of the bodies (finger, object, environment) can create large changes in the future motion of the object with frame $B$. To achieve generalisation, explicit information about the contacts must therefore be captured by the input variables, and the learner must be able to capture the resultant mapping from input to output.

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

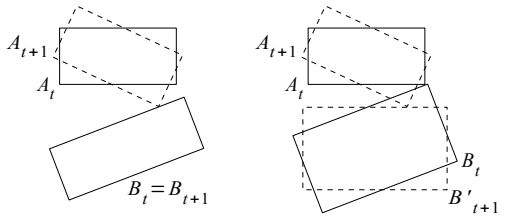


Fig. 3. Two scenes (left and right), each with two objects on a tabletop, viewed from above. Only the shape of object B differs between the scenes. Yet when A rotates to the position shown by the dashed outline at time $t+1$, the resulting displacement of B (represented by the transformation $T^{B_t,B_{t+1}}$) will be quite different.

Figure 3 gives an example of this with respect to shape changes. A predictor based on Equation (6), and trained on the smaller object $B$ (Figure 3 left scene), cannot generalise to the larger object (right scene). This is because the three input reference frames ($A_t$, $B_t$, $A_{t+1}$) in the left and right panels have identical poses. Thus the predictions will be identical even though the shapes are different. For the right panel the prediction will be that the finger $A$ will pass into object $B$. So, as posed there is insufficient information in the domain of the function to allow generalisation. To enable such generalisation information on the contact between $A$ and $B$ is required. If object $B$ has multiple contacts then information on all of them should be included. Physics simulators employ just such contact information to inform their predictions.

Our approach is to use pairs of local frames to encode this contact information. Each pair encodes one transformation between part of the object $B$ and another body. To distinguish these local frame pairs from what has gone before we henceforth refer to the main frame attached to each body (as defined in Section II) as the global frame for that body.

We can formally define these local frame pairs as follows. Reconsider the 2D projection of a robot finger with global frame $A_t$, an object with global frame $B_t$, and the environment global frame $O$ (Figure 4). We define a pair of local frames capturing the finger-object contact as $A^l_t$ and $B^l_t$. These are spatially dynamic, i.e. at any time $t$ they are located at the points of closest proximity on the finger and object respectively. We define the *agent-object contact* information as the transformations $T^{A^l_t,A^l_{t+1}}$ and $T^{A^l_t,B^l_t}$.

We additionally define $N$ pairs of local frames $B^{S^k}_t$ and $E^{S^k}_t$ to capture the object-environment contacts, where ($k = 1 \dots N$) (see Figure 4). We attach the $N$ frames $B^{S^k}_t$ to various parts of the object. Each has a corresponding frame in the environment $E^{S^k}_t$. Because they are spatially dynamic the frames $E^{S^k}_t$ move over surfaces in the environment, as the object moves. We define the *object-environment contact* information

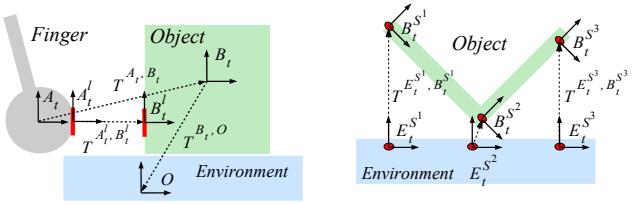


Fig. 4. The left panel depicts a 2D projection at time $t$ of a robotic finger with global frame $A_t$, an object with global frame $B_t$, and a ground plane with constant global frame $O$. Local frames $A_t^l$ and $B_t^l$ are situated on the finger and object at their point of closest proximity. The right panel shows that co-ordinate frames can be attached to an abitrary number of parts of the object, together with paired frames on the environment. The *object-environment contact* behaviour can be learned for each of these frames.

as the set of transformations $T^{E_t^{S^k},B_t^{S^k}}$ for $k = 1 \ldots N$.

To obtain the results presented in this paper, the number and the locations of the frames $B_t^{S^k}$ on each different object were determined by hand. In principle this procedure could be automated, but falls beyond the scope of the current work.

We have already motivated the need for these additional input variables in terms of generalisation in the face of changes in object shape. Their utility can also be seen with respect to changes in action. The top row of Figure 5 shows a training and a test case with different actions. The prediction of the test push trajectory requires some encoding of the kinematic constraint imposed by the contact between the base of the L-shaped flap and the table. This constraint exists in the training push, but is not significant since the flap can rotate around its corner, unconstrained by the base.

We can now consider the effects that different amounts of information might have on generalised predictions. A predictor endowed only with information based on the global frames for each body we refer to as having global information (G). We can add agent-object contact information to this (G+A), and in turn we can add object-environment contact information (G+A+E). Now consider the possible predictions for the test case (Figure 5 bottom row). A predictor using G will predict that the object will not move, because it has no nearby experience to suggest otherwise. A predictor using G+A has information from the training case that the object surface will

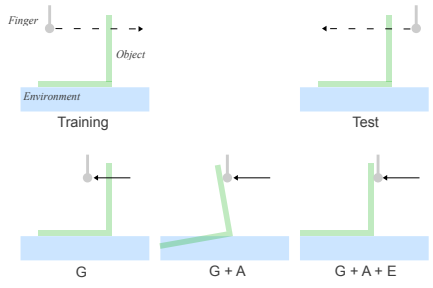


Fig. 5. Schematic diagram (2D projection of 3D scene) in which an object (of L-shaped cross-section) on a supporting surface is pushed by a robotic finger. Various predictors are trained solely on forward pushes (top left), but tested on backwards pushes (top right). The top panels show the push trajectory for the training and test phases, whereas the bottom panels show the outputs from three types of predictor (G, G+A and G+A+E) in the test phase.

move with the finger, so that the finger will not pass through it, but is also capable of predicting that the object rotates about the corner and into the table since it doesn't model the object-environment contact. A predictor using G+A+E will have information about the effect of the contact between the base of the flap and the table and so should avoid predicting a rotation into the table.

One point is critical here: the above analysis only concerns what the information allows, it depends on the ability of the learner to utilise it. To test this we must incorporate the information into each framework. We can simply extend the regression and density estimation frameworks to achieve this. For regression 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:

$$f'_{qs} : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}, T^{A_t^l,B_t^l}\{, T^{E_t^{S^k},B_t^{S^k}}\}_{k=1\ldots N} \longrightarrow T^{B_t,B_{t+1}} \quad (8)$$

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.

The conditional probability density (CPD) $p_{qs}$ over possible object motions $T^{B_t,B_{t+1}}$ [1] is augmented as follows:

$$p_{qs}(T^{B_t,B_{t+1}}|T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}, T^{A_t^l,B_t^l} \{, T^{E_t^{S^k},B_t^{S^k}}\}_{k=1\ldots N}) \quad (9)$$

Again the dimensionality of the conditioning variables makes density estimation hard as the number of contacts grows. One way round this in the density estimation case is to factorize the density in a way that reflects the contact structure. We consider this in the next section.

## V. Factorised Density Estimation

As formulated both learning problems rise in difficulty as further contacts are added. One question is whether the problem formulation for either can be structured so as to take advantage of the natural problem structure. In this section we present one scheme for the density estimation (or CPD) formulation, based on a product of experts.

Specifically the CPD formulation allows us to factorise the density and approximate $p_{qs}$ by making a conditional independence assumption. The unfactored CPD formulation gives a density over possible one step rigid body transformations of the object. We can factorise this by breaking up the conditioning variables into groups according to the contacts. This reflects the notion that the behaviour of the object is independent of contact $j$, conditional on contact $k$. Another way of explaining this is that each component of the product is an expert encoding the likely object motions given a particular kinematic constraint. The product will be maximised by a motion that best satisfies all the constraints simultaneously.

The computational advantage is that since the component densities factorise the conditioning variables of $p_{qs}$ their domains' dimensionalities are smaller, and so potentially they

can better 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 the following factorisation:

$$p_{qs} \approx C\ p_{global}\ p_{agent} \prod_{k=1\ldots N} p_{env,k} \tag{10}$$

where

$$p_{global} \equiv p_{global}(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}}, T^{A_t,B_t}, T^{B_t,O}) \tag{11a}$$

$$p_{agent} \equiv p_{agent}(T^{B_t^l,B_{t+1}^l}|T^{A_t^l,A_{t+1}^l}, T^{A_t^l,B_t^l}) \tag{11b}$$

$$p_{env,k} \equiv p_{env}(T^{B_t^{S^k},B_{t+1}^{S^k}}|T^{E_t^{S^k},B_t^{S^k}}) \tag{11c}$$

denote the *global*, *agent-object* and $k^{th}$ *object-environment* density factors respectively [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. In addition we make this product dynamic in the number of object-environment factors. Once the object surface is beyond some threshold distance from the environment surface its predictor switches off, and when it is close enough it switches on again. This enables us to keep only relevant predictors in the product at any one time – improving prediction quality and improving efficiency.

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:

$$T^{A_t^l,A_{t+1}^l} = (T^{I,A_t^l})^{-1} T_{in}^{A_t,A_{t+1}} T^{I,A_t^l} \tag{12a}$$

$$T^{B_t^l,B_{t+1}^l} = (T^{I,B_t^l})^{-1} T_{in}^{B_t,B_{t+1}} T^{I,B_t^l} \tag{12b}$$

$$T^{B_t^{S^k},B_{t+1}^{S^k}} = (T^{I,B_t^{S^k}})^{-1} T_{in}^{B_t,B_{t+1}} T^{I,B_t^{S^k}} \tag{12c}$$

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:

$$T^{A_t^l,B_t^l} = (T^{I,A_t^l})^{-1} T_{in}^{A_t^l,B_t^l} T^{I,A_t^l} \tag{13}$$

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 (10):

$$\widetilde{T}_{in}^{B_t,B_{t+1}} = \underset{T_{in}^{B_t,B_{t+1}}}{\operatorname{argmax}} \left\{ p_{global}\ p_{agent} \prod_{k=1\ldots N} p_{env,k} \right\} \tag{14}$$

where the similarity transforms (2) (in frame $B_t$) and (12) must be used to evaluate $p_{global}$, $p_{agent}$ and the $N$ environment factors $p_{env,k}$ for a given $T_{in}^{B_t,B_{t+1}}$.

Note that the transformation of the global factor is defined with respect to the inertial frame, whereas the agent and environment transformations are expressed as relative changes.

In summary we have now described two main formulations (regression and density estimation) able to incorporate varying amounts of information (G, G+A, G+A+E). We have also presented a reformulation of density estimation that factorises the prediction problem given information G+A or G+A+E into a product of experts. Which problem formulation and which information combine to provide the best prediction framework? Is the factorised problem better able to exploit the additional information than the unfactorised version. These questions can only be answered using specific regression and density estimation algorithms. Having completed our problem formulation we therefore now turn to the details of the implementations we used for each framework.

## VI. 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. We now detail the variety of parameterisations of rigid body transformations that may be employed, before describing the implementation details for our chosen regression and density estimation algorithms.

### A. Parameterisation

Both formulations of the prediction learning problem critically depend on the parameterisation of rigid-body transformations, i.e. the position and orientation of a rigid body in a 3D Cartesian space. Positions can be parameterised uniquely by vectors $\mathbf{p} \in \mathbb{R}^3$, and 3-DOF orientations by $3 \times 3$ rotation matrices $R \in SO(3)$ which comprise the special orthogonal group $SO(3)$ with respect to matrix multiplication. There exist other parameterisations of $SO(3)$ which unlike rotation matrices may not be unique, but provide a lower-dimensional description of 3-DOF orientations. In this paper we have tested two such parameterisations: Euler angles and quaternions (see e.g. [5]).

Using Euler angles $(\alpha, \beta, \gamma) \equiv \boldsymbol{\phi}$, the parameterisation of a rigid-body transformation becomes $\mathbf{x} = [\mathbf{p}; \boldsymbol{\phi}] \in \mathbb{R}^6$. Euler angles, as a local parameterisation of $SO(3)$, suffer from singularities – there is no global invertible smooth mapping. 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 Subsection VI-C) then the problem can be finessed by computing a circular difference.

If using quaternions, the parameterisation of a rigid-body transformation becomes $\mathbf{x} = [\mathbf{p}; \mathbf{q}] \in \mathbb{R}^7$, where $\mathbf{q}$ is a

unit quaternion. 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 $\omega$ by angle $\theta$ 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) [6], 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 [7]. 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 LWPR-GA, the input space was augmented by 6 dimensions to accommodate the transformation $T^{A_t^l,B_t^l}$. For 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 [7]. 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 (11b)).

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 (14) 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. 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) [8] is used to approximate the conditional densities employed in the product in Equation (14). We shall use "KDEF" to denote the KDE algorithm applied to the factored density (14), and use "KDE" to label the unfactored case (9). 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.

| 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 conditional probability density (CPD) $p_{qs}$ in Equation (9) and each CPD or factor in Equations (11a) to (11c) can be represented (up to a normalisation constant) as mixture of products of simple kernels:

$$P(\mathbf{Y}|\mathbf{X}) \propto \sum_{j=1\ldots M} K(\mathbf{X}, \hat{\mathbf{X}}_j, H\mathbf{h}^{(X)}) K(\mathbf{Y}, \hat{\mathbf{Y}}_j, H\mathbf{h}^{(Y)}) \tag{15}$$

where the vector $\mathbf{X}$ corresponds to the parameters of the transformations associated with the conditioning variables of the CPD, and the components of vector $\mathbf{Y}$ parameterise the transformation representing the predicted object motion. During learning, for each of the $M$ training samples, a pair of vectors $\hat{\mathbf{X}}_j$ and $\hat{\mathbf{Y}}_j$ is constructed from the parameter values of the associated transformations. These vectors then remain fixed during prediction. Vectors $\mathbf{h}^{(X)}$ and $\mathbf{h}^{(Y)}$, known as *bandwidths*, are then estimated from training samples using the "multivariate rule-of-thumb" [8]. These bandwidth parameters $\mathbf{h}^{(X)}$ and $\mathbf{h}^{(Y)}$ are additionally multiplied by a meta-parameter $H \in \mathbb{R}$ which can be estimated by a model selection process (see Subsection VII-B). Note that $H$, $\mathbf{h}^{(X)}$, $\mathbf{h}^{(Y)}$, $\hat{\mathbf{X}}_j$ and $\hat{\mathbf{Y}}_j$ will depend on the particular CPD to be estimated. (If needed, this dependency will be made explicit in the sequel, by using an extra index $k$.)

Each kernel function $K()$ in Equation (15) can be written in the form:

$$K(\mathbf{X}, \hat{\mathbf{X}}, \mathbf{h}) = \prod_{l=1\ldots L} \exp\left[-\frac{1}{2} d(\mathbf{x}_l, \hat{\mathbf{x}}_l, \mathbf{h}_l)\right] \tag{16}$$

where the $\mathbf{x}_l$ are the parameters of the $L$ transformations that are the conditioning variables in the CPD. A similar form exists for $\mathbf{Y}$, with $L = 1$. Furthermore $d()$ is a *distance function* that determines the kernel type:

1) For a *Gaussian kernel*:

$$d_{\text{Gauss}}(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{h}) = (\mathbf{x} - \hat{\mathbf{x}})^{\mathrm{T}} \mathbf{C}^{-1} (\mathbf{x} - \hat{\mathbf{x}}) \tag{17}$$

corresponding to a Mahalanobis distance with diagonal covariance $\mathbf{C}_{ii} = [\mathbf{h}]_i$. Here $\mathbf{x}, \hat{\mathbf{x}}, \mathbf{h}$ are in $\mathbb{R}^6$ for Euler angles or $\mathbb{R}^7$ for the quaternion parameterisation.

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

$$d_{\text{GaussVMF}}(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{h}) = d_{\text{Gauss}}(\mathbf{p}, \hat{\mathbf{p}}, \mathbf{h}^{(p)}) + d_{\text{VMF}}(\mathbf{q}, \hat{\mathbf{q}}, h^{(q)}) \tag{18}$$

where $\mathbf{h} = \left[\mathbf{h}^{(p)}; h^{(q)}\right]$, $\mathbf{h}^{(p)} \in \mathbb{R}^3$, $h^{(q)} \in \mathbb{R}$, and (see e.g. [9]):

$$d_{\text{VMF}}(\mathbf{q}, \hat{\mathbf{q}}, h^{(q)}) = 2h^{(q)} (1 - |\mathbf{q}.\hat{\mathbf{q}}|) \tag{19}$$

where $\mathbf{q}.\hat{\mathbf{q}}$ is the quaternion dot product and taking the absolute value fixes the double cover problem described in Subsection VI-A.

The Von Mises–Fisher kernel (16) with (19) is an approximation (up to a multiplicative constant [10]) of the von Mises–Fisher distribution:

$$f_4(x, \mu, \kappa) = C_4(\kappa) \exp(\kappa x^{\mathrm{T}} \mu) \tag{20}$$

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

The dimensions of the input $\mathbf{X}$ and output $\mathbf{Y}$ spaces for KDEF predictors are summarised in Table II.

Following Equation (14), the prediction problem at each step can be defined as finding that transformation $\widetilde{T}^y$ which maximises the product of conditional densities (10) given fixed $T^x$, i.e.:

$$\widetilde{T}^y = \underset{T^y}{\operatorname{argmax}}\, p(T^y | T^x) \tag{21}$$

Prior to the maximisation procedure, for each factor $k$ and each training sample $j$ values of $K(\mathbf{X}, \hat{\mathbf{X}}_j, H\mathbf{h}^{(X)})$ in (15) are computed and assigned to $w_{k,j}$, referred to as *weights*. The normalised weight $w_{k,j}$ can be interpreted as a probability of generating $\mathbf{X}$ from a multivariate Gaussian centred at $\hat{\mathbf{X}}_{j,k}$ with covariance $\mathbf{C}_{ii} = [H_k \mathbf{h}_k^{(X)}]_i$. In order to increase the computational efficiency of the optimisation procedure we only consider the $M_k^{max} < M_k$ most relevant training samples, (i.e. those with the highest weights $w_{k,j}$ at the query point $\mathbf{X}$). These are identified by calculating, for each training sample, the value of the distance function $d()$ (from Equation (16)). The training samples are then sorted according to the distance using a partial sort algorithm (e.g. [11]) to identify the $M_k^{max}$ most relevant training samples.

Equation (21) is computed using the differential evolution (DE) optimisation algorithm [12]. Note that one cannot use the mean-shift algorithm [13] mostly due to the product involved in (21). DE is particularly simple to tune, since it is controlled by only two meta-parameters: a *crossover probability* and a *population size* of candidate solutions. If the number of generations is fixed, the total run time scales linearly with the number of factors, their dimensionality and the number samples. Optionally, the entire maximisation procedure is stopped when no further significant improvement is observed. The optimisation algorithm requires an ability to sample from the product (10). This can be achieved by performing the following two step procedure:

1) Randomly choose an factor $k$ and then sample $j$ from a set of samples $\{w_{k,j}\}$ using the importance sampling algorithm with importance weights $w_{k,j}$ (e.g. [14]).
2) Sample from a multivariate Gaussian centred at $\hat{\mathbf{Y}}_{j,k}$ with covariance $\mathbf{C}_{ii} = [H_k \mathbf{h}_k^{(Y)}]_i$, using e.g. the Marsaglia polar method [15].

Importantly, after generating a new solution candidate, all sub-vectors $\mathbf{q}$ have to be normalised when a quaternion parameterisation is used.

To improve the efficiency of the implementation, no learning or prediction was performed in a given trial until the initial contact was made between robotic finger and object. This procedure was employed by the KDE, KDEF and LWPR predictors. This means that the learners and predictors are switched on by an initial finger-object contact. In addition, for the KDEF predictor, at each time step, agent and environment factors were only used if the corresponding agent-object or object-environment distance was within a certain threshold.

| 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.

## VII. EXPERIMENTAL STUDY

### A. Overview of experiments

We conducted three experiments: L (learning), A (action) and S (shape). Each experiment test several combinations of information (G,A,E) and learning algorithm (LWPR, KDE, KDEF). The structure of these experiments reflects the structure of our hypotheses: H1-H3.

**Experiment L (learning):** tests two hypotheses. **H1***: a learning approach can outperform physics engines for prediction of rigid body motion for a variety of objects.* **H2 a)***: learned predictions generalise interpolatively with respect to actions.* In the experiment each learning method was used to learn a specialised predictor for each object and the results were compared to the predictions of a physics simulator that had also been tuned to each object. In these experiments we also compared, where relevant, the effects of different parameterisations of rigid body transformations.

**Experiment A (action):** tests the hypothesis **H2 b)***: learned predictions can be transferred to novel actions.* The learners were trained with a reduced action set on a non-symmetric object and then tested on that object with actions which fall well outside the convex hull of the training set. We performed this in simulation and with a real object.

**Experiment S (shape):** tests two hypotheses. **H3 a):** *learned predictions generalise interpolatively with respect to object shape.* **H3 b)***: learned predictions can be transferred to object shape.* Learners were trained on one or more objects, and tested on an object of different shape. For interpolation the new shape fell within the range of shape variation of the training objects, for learning transfer the test object's shape fell outside that set. Interpolation was performed in simulation, and transfer both in simulation and with real objects.

### B. Experimental setup

In each learning trial a robot finger moved in a straight line to push an object for 10 seconds, making contact at a randomly selected point on its surface (Figure 6). Video and finger position were captured at 30Hz, including a 1 second buffer at either end of the finger trajectory. Learning used the pose of the object in every other frame estimated using

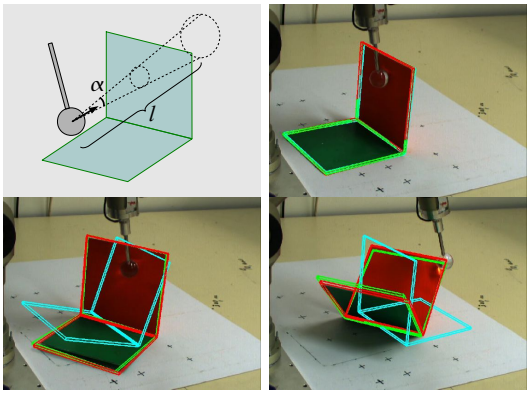


Fig. 6. A 5-DOF robotic arm equipped with a finger performs a random straight-line pushing movement of a variable length $l$=25±5 cm within a cone with angle $\alpha$=20 deg towards an object (top left). The movement begins at a random location so that every small region on the upper part of an object is equally likely to be pushed. In the image sequence shown above, 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.

a particle filter tracker [16]. For test trials the trajectory of the finger was known a priori and the object pose estimated only for the initial frame. Using these a trajectory was predicted for the object over 150 steps (10 seconds). The predicted trajectory was created purely using the forward model and did not use any observations during the push to update its prediction.

For real experiments, we used a 5-axis Katana robotic manipulator equipped with a single rigid finger. Simulation experiments were carried out using the NVIDIA PhysX physics engine. These simulation experiments provided us with perfect ground-truth data against which to evaluate predictions, and also enabled a large number of experiments with different values of key parameters. Experiments on real objects tested the robustness of the methods to disturbances and tracking errors. The PhysX engine was also separately compared to the learning methods when both were trained or tuned on real data.

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 require many parameters to be determined before learning from a training set. A model selection procedure was performed during experiment L, to establish reasonable values for these parameters, which were then used in experiments A and S. It was not possible to perform fully systematic optimisations due to the size of the parameter spaces. Rather, subsets of the 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 and the test set. This model selection process was also performed for the following parameters of the PhysX physics simulator: static friction, dynamic friction and the coefficient of restitution. Because of the reduced parameter space we performed grid search for these PhysX parameters. In addition to model selection the three different parameterisations (Gauss-Euler, Gauss-Quaternion, Von-Mises-Fisher-Quat) of rotations for the density estimation method were studied in experiment L, and subsequently the best solution was used in experiments A and S. For LWPR we used the Euler parameterisation throughout experiments L, A and S.

For clarity a complete set of acronymns for the algorithm-information combinations is given in Table III. The PhysX algorithm has access to full mesh and contact information when making its predictions.

### C. Performance measure

In all experiments, we took the output of the tracked 6D pose of a real object to be ground-truth, and compared 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 ±2mm during successful tracking, or arbitrarily large errors during tracking loss. 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 $n = 1 \ldots N$, 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:

$$E_t = \frac{1}{N} \sum_{n=1 \ldots N} |p_n^{2,t} - p_n^{1,t}| \tag{22}$$

Note that for each 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:

$$R_t = \frac{1}{N} \sum_{n=1 \ldots N} |p_n^{1,t} - p_n^{1,0}| \tag{23}$$

For a test data set, consisting of $K$ robotic pushes, each of which breaks down into many consecutive predictions over $T$

| | Information | | |
|---|---|---|---|
| Predictor | G | G+A | G+A+E |
| LWPR | LWPR-G | LWPR-GA | LWPR-GAE |
| KDE | KDE-G | KDE-GA | KDE-GAE |
| KDEF | KDEF-G | KDEF-GA | KDEF-GAE |
| PhysX | - | - | - |

TABLE III
ALGORITHM-INFORMATION VARIANTS.

time steps, we can now define average error and normalised average error:

$$E_{av} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{T}\sum_{t=1}^{T}E_t, \quad E_{av}^{norm} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{T}\sum_{t=1}^{T}\frac{E_t}{R_t} \quad (24)$$

Note that this normalised error measure necessarily has no units.

For experiment L we performed 10-fold cross-validation. In each case we give the size of the training and test sets. The model selection set was the same size as the test set. For transfer learning experiments A and S fixed and disjoint training and test sets were used.

## VIII. Results

### A. Experiment L: Learning

In Experiment L the robot applied a set of random pushes to a polyflap, a box and a cylinder respectively. All the algorithm variants in Table III were trained and tested. Model selection was performed for all algorithm-information combinations as described previously, including for PhysX. Ten fold cross-validation was performed for all algorithms. The density estimation techniques were studied with three parameterisations: Gaussian kernels with 1) quaternion and 2) Euler angle $SO(3)$ parameterisations as well as 3) von Mises-Fisher kernels. Training (and testing) sets were 200 (25) pushes (cylinder), 400 (50) pushes (box) and 700 (90) pushes (polyflap). Convergence of the best performing learning algorithms with increasing amounts of data is shown in Figure 7 (left column). Table IV shows the generalisation performance after training for each object, and how this varies with the parameterisation method. Figure 7 (right column) shows how performance varies with input information about contacts for the best versions of all the algorithms. Image sequences of predicted vs actual trajectories are shown in (Figures 8-9).

| Predictor | Polyflap | Box | Cylinder |
|---|---|---|---|
| KDEF-G euler | 0.055±0.002 | 0.061±0.003 | 0.063±0.003 |
| KDEF-G quat | **0.049**±0.002 | **0.059**±0.002 | **0.059**±0.003 |
| KDEF-G vmf | 0.057±0.002 | 0.066±0.003 | 0.071±0.003 |
| LWPR-G euler | 0.059±0.002 | 0.118±0.003 | 0.077±0.003 |
| KDEF-GA euler | 0.054±0.002 | 0.060±0.003 | 0.052±0.002 |
| KDEF-GA quat | **0.044**±0.002 | **0.057**±0.002 | **0.047**±0.002 |
| KDEF-GA vmf | 0.064±0.002 | 0.097±0.002 | 0.109±0.003 |
| LWPR-GA euler | 0.068±0.002 | 0.127±0.003 | 0.105±0.002 |
| KDEF-GAE euler | 0.083±0.003 | 0.065±0.003 | 0.050±0.002 |
| KDEF-GAE quat | **0.062**±0.002 | **0.065**±0.003 | **0.047**±0.002 |
| KDEF-GAE vmf | 0.081±0.002 | 0.086±0.002 | 0.065±0.002 |
| LWPR-GAE euler | 0.069±0.002 | 0.136±0.003 | 0.116±0.003 |
| KDE-GA euler | 0.053±0.002 | 0.057±0.002 | 0.068±0.003 |
| KDE-GA quat | **0.049**±0.002 | **0.057**±0.002 | **0.065**±0.003 |
| KDE-GA vmf | 0.062±0.002 | 0.058±0.002 | 0.092±0.004 |
| KDE-GAE euler | 0.090±0.002 | 0.161±0.003 | 0.071±0.003 |
| KDE-GAE quat | **0.087**±0.002 | 0.253±0.002 | **0.068**±0.003 |
| KDE-GAE vmf | **0.087**±0.002 | **0.127**±0.003 | 0.091±0.004 |
| PhysX | 0.144±0.003 | 0.171±0.003 | 0.271±0.001 |

TABLE IV
Experiment L: Forward push on a polyflap/box/cylinder, trained on real data. Shown is the dimensionless measure normalised average error $E_{av}^{norm}\pm$ standard error.

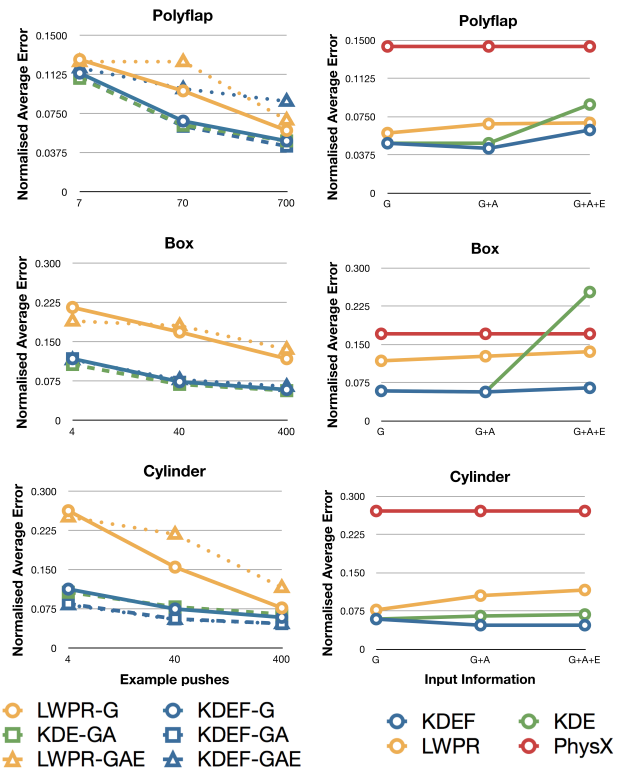


Fig. 7. Experiment L: Convergence of a selection of learning algorithm-information combinations (left column). Change in normalised average prediction errors with varying input information (right column). G - global information; A - agent-object information; E - object-environment information.

**Experiment L discussion:** We may draw several conclusions. First from Table IV and Figure 7 it can be seen that the learned models almost always outperformed the physics simulator on the test set, giving an error approximately one third the size. Thus we find strong support for hypothesis H1. Second the test set contained actions different from those in the training set but which fell within the convex hull of experienced actions, thus requiring interpolative generalisation. There is thus support for hypothesis H2 a). Finally with respect to parameterisations of density estimation Gaussians with quaternions gave the best performance in 14 of the 15 cases (Table IV bold entries). Thus in experiments A and S this parameterisation was used for the density estimation algorithms.

In the image sequences (Figures 8- 9) it can be seen that predictions were accurate and physically plausible for a variety of learning methods even over 150 steps. Note the physics simulator predicts incorrect turning of the cylinder when pushed (Figure 9 column 8). Finally Figure 7 also shows the interaction between different amounts of information in the input space and the learning algorithm. This shows that when learning about a single object that regression performance declined slightly as the dimensionality was increased. It also shows that when learning about a single object additional information about contacts provides no clear advantage for any of the learning methods. This is because for a model of a single object no learning transfer is required. These were the two

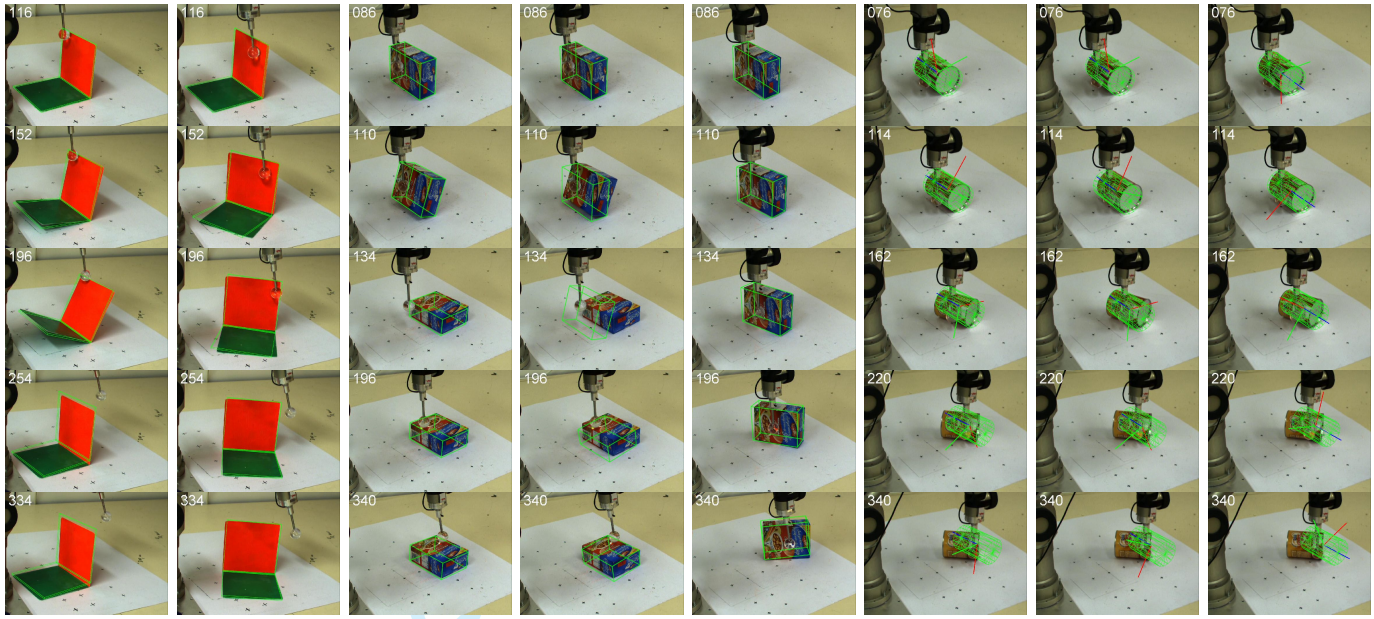

Fig. 9. Experiment L: polyflap, box and cylinder. Green outline shows predictions. Columns 1-2: KDEF-GA/quat on two trials exhibiting different motions of the polyflap. Col 3: KDEF-GA/quat. Col 4: LWPR-G for one trial in which the box topples over. Col 5: KDEF-GA/quat on another trial in which the box slides. Columns 6-8 show the same push of the cylinder. Col 6: KDEF-GA/quat. Col 7: LWPR-G. Col 8: PhysX. Note tha for columns 6-7 the orientation of the cylinder is show by the rotating frame. Only the learned models predict the rotation correctly.(The frame number is shown in the top left of each image.)

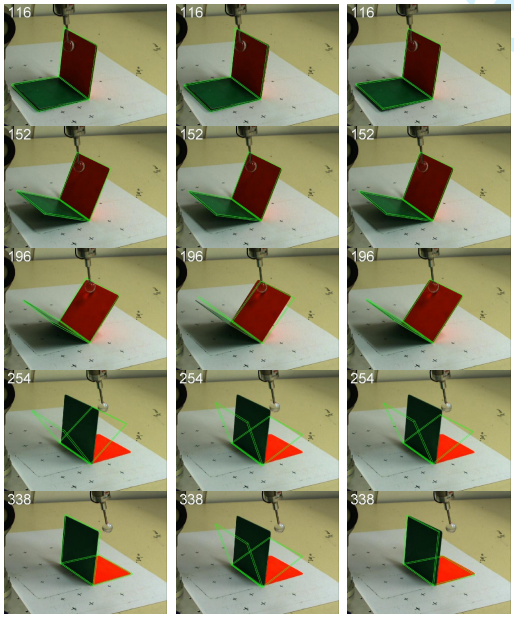

Fig. 8. Experiment L: polyflap. Green outline shows the predictions (from left column to right column) by KDEF-GA/quat, LWPR-G, and PhysX for the same trial. Note that LWPR-G doesn't predict the final fall of the polyflap (The frame number is shown in the top left of each image.)

cases where it was hypothesized that additional information about contacts could improve performance. We now turn to these cases in experiments A and S.

### B. Experiment A: transfer to novel actions

Experiment A tests hypothesis H2 b): whether predictions can be transfered to novel actions, i.e. to those that fall outside the convex hull of actions in the training set. The predictors

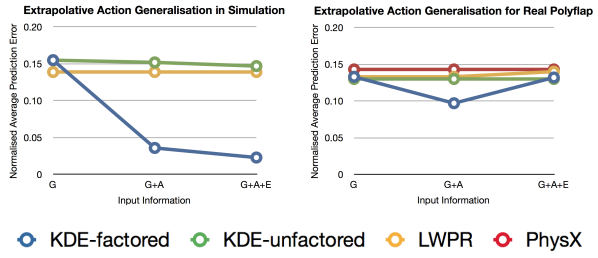


Fig. 10. Experiment A: Transfer of learning to novel actions. Trained on forward push on polyflap, tested on backward push, for simulated (top) and real data (bottom). Comparative performance of predictors vs. information utilised (global/agent/environment), as measured by the normalised average error $E_{av}^{norm}$.

were trained with 900 examples of pushes applied to a polyflap in one direction relative to the polyflap (Figure 5 top left). Predictors were then tested on 100 examples of pushes applied in the opposite direction (Figure 5 top right). Gaussian kernels with quaternion parameterisation were used for the density estimation methods, and an Euler angle representation with LWPR. The same method was followed in simulation and with the real object. In the real case we tested all the algorithm-information variants in Table III. In simulation we tested all the variants except the PhysX simulator, which provided the simulation environment. In each case the error measured is the transfer prediction error, i.e. the prediction error for the novel test actions.

Figure 10 (also see Table V) shows the normalised average error $E_{av}^{norm}$ for tests carried out in simulation (left panel) and with real objects (right panel). Figure 11 shows example predicted trajectories on synthetic and real test cases.

**Experiment A discussion:** The following conclusions may

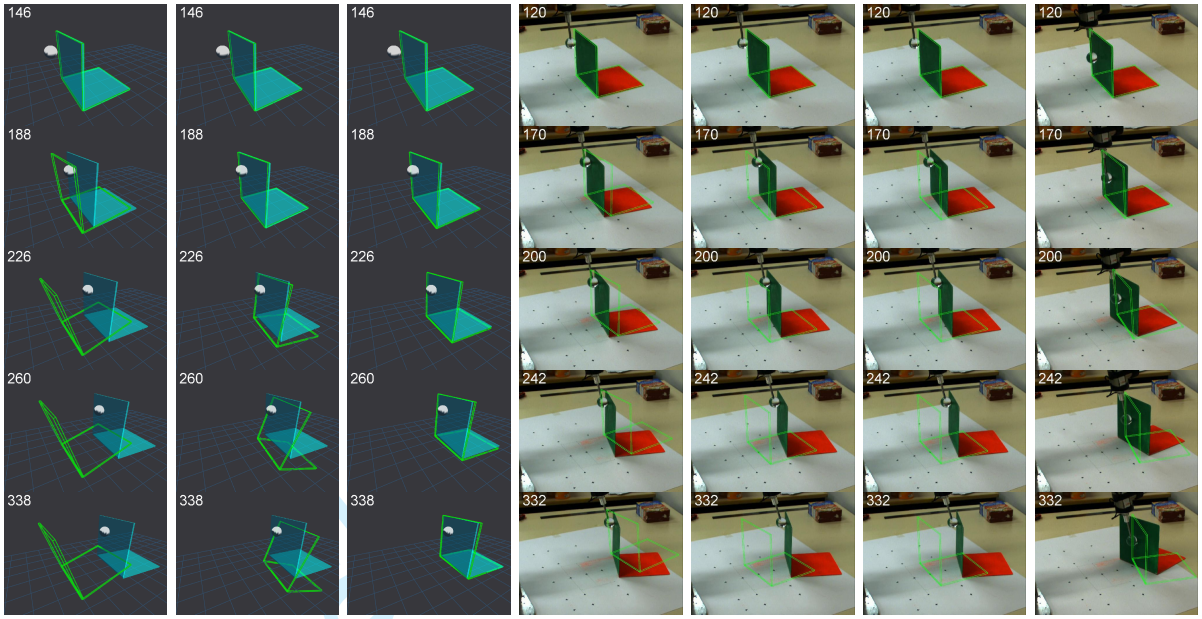


Fig. 11. Experiment A: Green outline shows predictions. Column 1: KDEF-G/quat. Col 2: KDEF-GA/quat. Col 3: KDEF-GAE/quat. Col 4: KDEF-GA/quat. Col 5: KDEF-G/quat. Col 6: LWPR-G. Col 7: KDEF-GA/quat. Note that the KDEF-G/quat and LWPR-G methods predict that the robot finger passes through the polyflap. (The frame number is shown in the top left of each image.)

be drawn. Figure 10 (left panel) shows that in simulation the unfactored KDE and LWPR algorithms had roughly constant transfer prediction error irrespective of increasing information. In contrast the transfer error of KDEF reduced significantly as information about finger-object and object-environment relations was added. The predictions (Figure 11) matched the hypothesised behaviour from Figure 5 (bottom row). With only global information the finger was predicted to pass through the object (Figure 11 column 1). By adding the agent-object information and modelling its effect as a separate factor, the prediction was that the object would move with the finger, but that it penetrated the table (Figure 11 column 2). By also adding object-environment information the prediction prevented this penetration so that the object will move along the table with the finger (Figure 11 column 3).

On real data (Figure 10 right panel) the transfer performance of the learned predictors was slightly better than the prediction error of the physics engine. All the methods had roughly constant performance except for the factored version. Its advantage no longer held when information G+A+E was used, but it retained some advantage with information G+A. Figure 11 (columns 5 and 6) shows that learners (KDEF, LWPR) with information G predict, as in simulation, that the finger passes through the object, and that the object doesn't move at all. In Figure 11 (columns 4 and 7) the KDEF predictor with information G+A correctly predicts the sliding motion of the object, although it fails to capture the slight rotation. Collectively these results provide support for hypothesis H2 b) that learning transfer can occur, but information GAE is required for best performance provided the observations are reliable. In addition only the factored KDE formulation was able to exploit this additional information, either in simulation or for real objects.

| | | 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 V
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 $E_{av}^{norm}$ ± STANDARD ERROR.

### C. Experiment S: shape generalisation and transfer

Experiment S consists of two experiments S-interpolation and S-transfer. The first tests hypothesis H3 a), by measuring generalisation error when a range of shapes are presented in training, and the test shapes fall within that variation, requiring interpolative generalisation. The second tests hypothesis H3 b), by measuring whether predictors learned from one or two objects can transfer their predictions to an object of novel shape from those encountered during training.

*1) S-interpolation:* This experiment was performed only in simulation. All training and test data involved polyflaps constructed from two square plates. A set of polyflaps was generated by randomising the angle at which the plates connect. Even a small angular variation can alter the behaviour of the object significantly. As Figure 12 shows, a downwards push from above will cause the entire object to move either left or right, depending on small changes in the angle of the plates. The training was carried out on 900 downward pushes

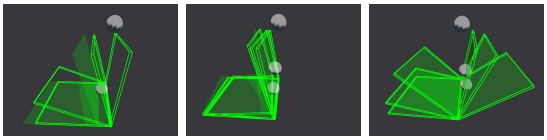

Fig. 12. Experiment S-interpolative. These images show training data for this experiment in which a fingertip (grey ball) descended onto a polyflap from above. Small variations in the fold angle of the polyflap can lead to quite different object motions, given identical pushes from the fingertip. When the polyflap is folded acutely, the fingertip will push it to the left, whereas with a fold angle greater than ninety degrees, the same push will move the polyflap to the right.

of polyflaps with angles varying uniformly around 90° by ±70°, and tested on 100 new polyflaps drawn from the same distribution. Figure 14 (bottom right) shows the normalised average error $E_{av}^{norm}$ for methods KDEF, LWPR and KDE-GA, as the information available varies, with further detail in Table VI. Specific predictions are shown in Figure 13.

**S-interpolation discussion:** The experiment reveals that adding additional information (GA or GAE) significantly reduces generalisation error. Figure 13 (column 1) shows that only using global information G gives inadequate predictions because shape changes are not accounted for, whereas columns 2 and 3 show qualitatively correct predictions when they are. All three learning frameworks are able to exploit this information.

*2) S-transfer:* In Experiment S-transfer, we test in simulation and with real objects whether learning can be transferred to objects of novel shape. We perform the same experiment testing learning transfer from i) a polyflap to a box and ii) a box and a cylinder to a double cylinder. There were 900 training pushes on the polyflap and 200 test pushes on the box for i), and 200 training pushes (100 box, 100 cylinder) and 100 test pushes (double cylinder) for ii). This experiment ran on real objects for i) and ii) and in simulation only for i), giving three train-test conditions in total. We have previously published results from simulation for i) [3]. A Gaussian-quaternion parameterisation was used for the density estimation methods and an Euler angle parameterisation for LWPR. All algorithm-information combinations from Table III were tried. When learning from two objects the same number of factors (experts) were used for each object, and they were matched across the two objects by hand. This means each expert received a mix of data from each object, learning to encode both rolling and sliding or tipping motions. The normalised average error for all three conditions $E_{av}^{norm}$ is shown in Figure 14 and in Tables VII-VIII. Example frames from the experiments are shown in Figure 15.

**S-transfer discussion:** The easier transfer case is from polyflap to box, here the range of motions for the polyflap from one direction are similar to those for a box. Even so the actual transferred learner predictions are physically plausible only when additional information is present, such as for the KDEF-GA method (Figure 15 column 1). In this experiment KDEF-G and LWPR-G again wrongly predict finger motion through the box in columns 2 and 3. The harder transfer case is to learn from two different objects (box and cylinder) and transfer to one quite different object (double cylinder). In this case

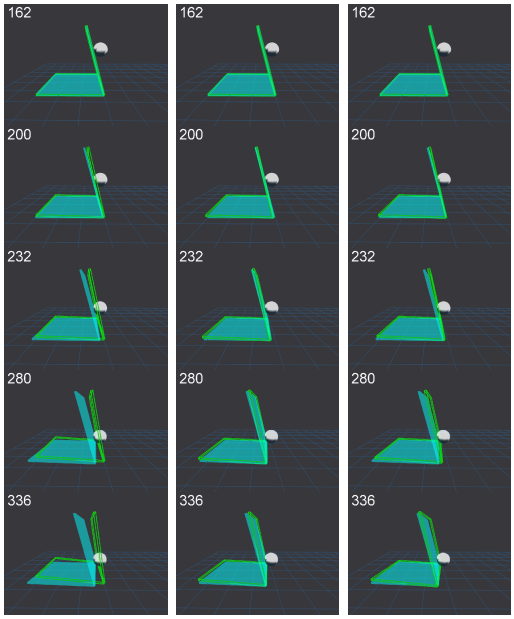


Fig. 13. Experiment S-interpolation (simulation) reveals limitations of the methods using only global information, which fail to predict the motion of an angled polyflap when subjected to a downward push. The augmented input methods can cope well with this kind of shape variation. Green outline shows predictions (left to right column) 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.)

only the KDEF-GAE method was able to produce physically plausible predictions. In Figure 15 (column 6) the prediction is of the double-cylinder sliding along the table, whereas KDEF with information G or G+A makes physically implausible predictions of interpenetration with the table (Figure 15 columns 4 and 5). That the predictions are plausible with real data is shown in Figure 15. These results collectively provide support that only using information G+A+E is sufficient to achieve learning transfer to novel shapes, with that entailing both low prediction errors and physically plausible predictions. In addition only the factored formulation of KDE was able to take advantage of this information. Thus this experiment provides support for hypothesis H3b) for the case of KDE-GAE only.

## IX. Related Work

Most work on push manipulation in robots [17], [18], [19], [20] it is restricted to planar sliding motions of what are effectively 2D objects. Little work addresses push manipulation on real 3D bodies, which are free to tip or roll. Rigid

| | 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 VI
EXPERIMENT S-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 $E_{av}^{norm}$.

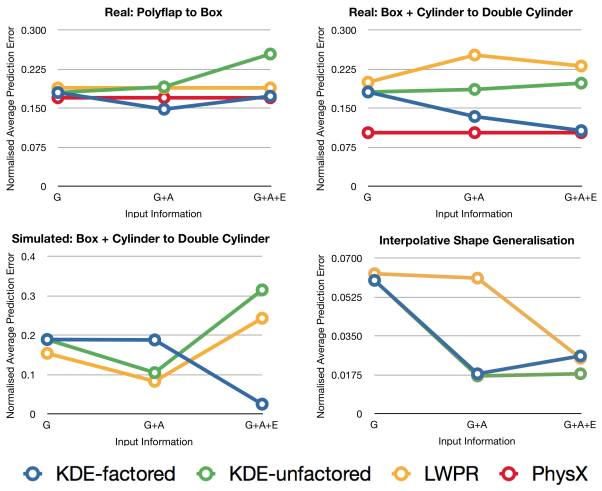


Fig. 14. Results from S-transfer (top row and bottom left) and S-interpolative (bottom right). Comparative performance of predictors vs. information utilised, as measured by the normalised average error $E_{av}^{norm}$.

body simulators are used for prediction, but rely on explicit knowledge of parameters which are difficult to ascertain. Even then, such predictions may not be possible due to inherent limitations of the physical model employed, for example when modelling friction.

Some machine learning approaches have been developed to classify or provide predictions for objects or object classes, e.g. rolling versus non-rolling objects [21], [22], or liftable versus non-liftable objects [23]. These works 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 predicts explicit rigid body transformations, and generalisation to novel push directions, object poses, and shapes.

Stoytchev [24] described a robotic system that learns affordances of sticks and hook-like tools by using them to push observed objects. A relationship is learned between an action with a particular tool, and the resulting motion of the pushed object. This work feaured 2D scenes, in which the motions of puck-like discs were restricted to a plane, under four discrete possible actions. Importantly, while the outcomes of actions could be learned for tools of various shapes, the system was not able to apply knowledge learned from one tool to make predictions about another tool of similar shape.

| | | Information Utilised | | |
|---|---|---|---|---|
| Predictor | data | Global (G) | G & Agent (A) | G & A & Env |
| 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 VII
EXPERIMENT S-TRANSFER: GENERALISATION TO NOVEL SHAPE. TRAINED ON POLYFLAP, TESTED ON BOX, FOR REAL DATA. COMPARATIVE PERFORMANCE OF PREDICTORS VS. INFORMATION USED. SHOWN IS THE DIMENSIONLESS MEASURE NORMALISED AVERAGE ERROR $E_{av}^{norm}\pm$ STANDARD ERROR.

## X. CONCLUSIONS

This paper has made the following contributions to learning transferable models of object behaviour:

**Forward models of objects can be learned.** This is supported by experiment L, where the behaviour of three real objects was learned by 8 of the 9 algorithm-information combinations. Experiment L showed action generalisation, and experiment S interpolative shape generalisation. Thus hypotheses H2 a) and H3 a) have support.

**Learning transfer can be achieved.** The simulated and real runs from experiments A and S both support the hypotheses H2 b) and H3 b) that learning transfer can be achieved with respect to shape and action. The analysis of simulated runs explains why some information-algorithm combinations succeed and some do not.

**Contact information assists transfer.** We hypothesised that modelling agent-object and object-environment contacts is important for learning transferable models. This hypothesis is supported by experiments A and S. In simulation adding information G+A+E can give substantial performance improvements, for both transfer to novel actions (Figure 10) (left panel) and to novel shapes (Figure 14) (bottom left). This result held for shape transfer for one of the two real object cases.

**Factorisation assists transfer.** Modelling contacts is hard. The right representation is required to exploit the available information. Experiments A and S in simulation show that factoring the density estimation problem by contacts aids learning (see Figure 10 (left panel), and Figure 14 (bottom left panel and top right panel). This supports the hypothesis that factorisation of information by contact assists transfer.

**Learning can match or exceed physics engine performance.** In experiment L, in 23 of 24 cases the learners significantly outperformed a tuned physics engine. The learned predictions were usually physically plausible (see Figures 8- 9). This supports hypothesis H1. In experiments A and S learning transfer using factored KDE was able to match or improve on the prediction error of a physics engine (see Figures 10 right panel and Figure 14 top row). Its predictions were also usually physically plausible (see Figure 11 columns 4 and 7, and Figure 15 columns 1,6 and 7).

**Limitations and extensions:** This work is a first attempt to perform transfer learning for motion models of objects.

| | | 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 VIII
EXPERIMENT S-TRANSFER: 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.

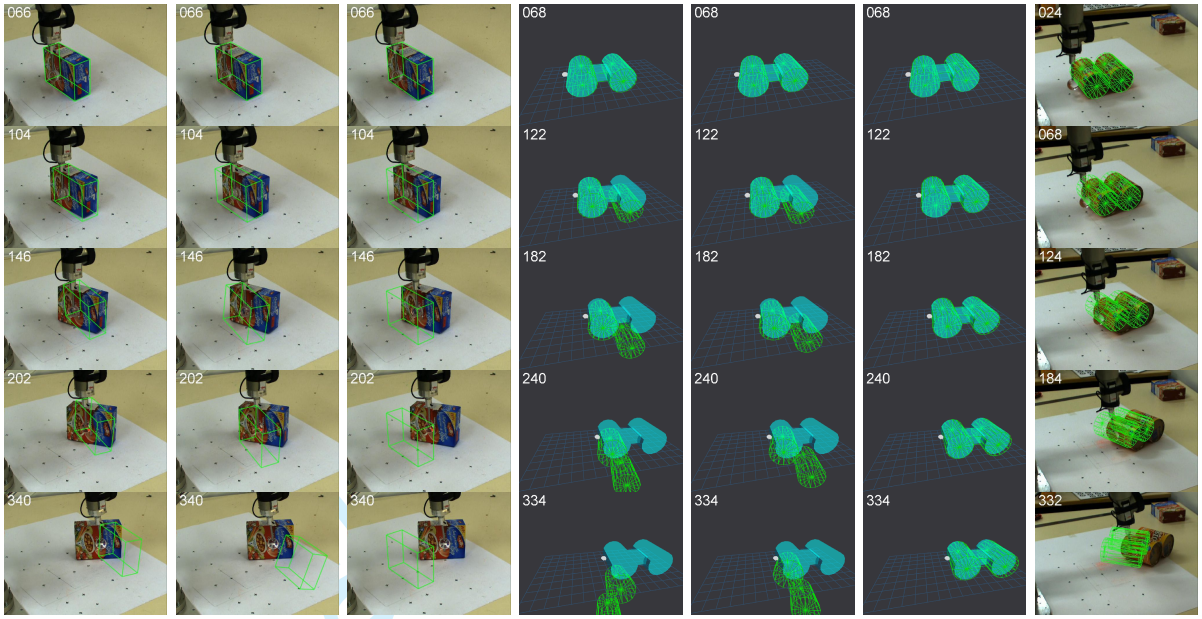


Fig. 15. Experiment S-transfer: extrapolative generalisation to novel shape. Green outline shows predictions. Column 1: KDEF-GA/quat. Col 2: KDEF-G/quat. Col 3: LWPR-G for one trial. Note that the KDEF-G/quat and LWPR-G methods predict that the robot finger moves into the box. Col 4: KDEF-G/quat. Col 5: KDEF-GA/quat. Col 6: KDEF-GAE/quat. Col 7: KDEF-GAE/quat. The frame number is shown in the top left of each image.

There are two limitations to the methods presented. The most important is seen in the degradation of transfer performance in the face of observation noise. We are therefore extending our learning algorithm to take account of this noise. The second is that at the moment learning and transfer both require selection of the attachment points of frames to each body. This process needs to be automated.

This paper establishes that predictions of object behaviour can be learned using a variety of techniques; that they can outperform physics engine predictions; that they can generalise interpolatively to similar actions and shapes; that they can be transfered to novel actions and shapes, and that they produce physically plausible predictions. Transfer learning of such models, while challenging, is possible with the right representation. In particular a factored problem formulation encodes transferable predictions that are physically plausible and qualitatively correct.

## REFERENCES

[1] M. Kopicki, J. Wyatt, and R. Stolkin, "Prediction learning in robotic pushing manipulation," in *ICAR*, 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 *IEEE ICRA 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 *IEEE ICRA 2011*.
[5] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.
[6] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *N. Comp.*, vol. 17, no. 12, pp. 2602–2634, 2005.
[7] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *J. Mach. Learn. Res.*, vol. 9, pp. 623–626, 2008.
[8] D. W. Scott and S. R. Sain, *"Multi-Dimensional Density Estimation"*. Elsevier, 2004, pp. 229–263.
[9] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. NY: Dover, 1965.
[10] R. Detry, "Learning of multi-dimensional, multi-modal features for robotic grasping," Ph.D. dissertation, University of Liege, 2010.
[11] D. Knuth, *Searching and Sorting, The Art of Computer Programming, vol. 3*. Addison-Wesley, 1998.
[12] 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.
[13] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE PAMI*, vol. 17, no. 8, pp. 790–799, 1995.
[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.
[15] G. Marsaglia and T. A. Bray, "A convenient method for generating normal variables," *Siam Review*, vol. 6, no. 3, pp. 260–264, 1964.
[16] T. Mörwald, M. Zillich, and M. Vincze, "Edge tracking of textured objects with a recursive particle filter," in *Proceedings of Graphicon*, 2009.
[17] M. T. Mason, "Manipulator grasping and pushing operations," PhD Thesis, MIT, 1982.
[18] K. Lynch, "The mechanics of fine manipulation by pushing," in *IEEE ICRA 1992*, pp. 2269–2276.
[19] M. Peshkin and A. Sanderson, "The motion of a pushed, sliding workpiece," vol. 4, pp. 569–598.
[20] D. J. Cappelleri, J. Fink, B. Mukundakrishnan, V. Kumar, and J. C. Trinkle, "Designing open-loop plans for planar micro-manipulation," in *IEEE ICRA 2006*, pp. 637–642.
[21] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *IEEE ICRA 2003*, vol. 3.
[22] 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.
[23] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner, "Learning to perceive affordances in a framework of developmental embodied cognition," in *IEEE ICDL*, 2007, pp. 110–115.
[24] A. Stoytchev, "Learning the affordances of tools using a behavior-grounded approach," *Lecture Notes in Artificial Intelligence (LNAI)*, vol. 4760, pp. 140–158, 2008.