A Physics-Based Model Prior for Object-Oriented MDPs

Jonathan Scholz JKSCHOLZ@GATECH.EDU
Martin Levihn LEVIHN@GATECH.EDU
Charles L. Isbell ISBELL@GATECH.EDU
Georgia Institute of Technology, 801 Atlantic Dr. Atlanta, GA 30332 USA

David Wingate WINGATED@MIT.EDU
Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139 USA

Abstract

One of the key challenges in using reinforcement learning in robotics is the need for models that capture natural world structure. There are methods that formalize multi-object dynamics using relational representations, but these methods are not sufficiently compact for real-world robotics. We present a physics-based approach that exploits modern simulation tools to efficiently parameterize physical dynamics. Our results show that this representation can result in much faster learning, by virtue of its strong but appropriate inductive bias in physical environments.

1. Introduction

One of the fundamental challenges in deploying robots outside of the laboratory is that robots must interact with previously unknown objects. Imagine designing a robot to rearrange your furniture. The robot must be able to safely move each type of object, while avoiding collisions. Reinforcement learning (RL) offers an attractive solution: rather than specifying a complete world model, we can specify a *model space* for the robot to estimate and use to plan online. For this approach to be efficient, the model must represent *relational dynamics* that capture how object movement depends on the state of other objects.

Object-Oriented Markov Decision Processes (OO-MDPs) represent dynamics as a finite set of object attributes and relationships (Diuk et al., 2008). In this way, OO-MDPs both manage large state-spaces and can generalize to unseen states; however, there are three critical properties that limit the usefulness of OO-MDPs for real-world dynamics:

1. The dynamics model is discrete, and cannot exploit the geometry of physical state spaces (*e.g.* actions cause *displacements* in *coordinate frames*).
2. It is missing the notion of an *integrator*, which models the evolution of state dimensions conditional on other state dimensions (*e.g.* velocity changes position without external force).
3. Relations are represented with first-order predicates, which cannot define relations parametrically (*e.g.* $contact_\theta(obj_1)$).

As we will show, the first two issues can be overcome by using *state-space regression* as the core dynamics model; however, overcoming the limitations of first-order predicates to represent relationships for real-world applications is a more serious challenge.

In this paper we present Physics-Based Reinforcement Learning (PBRL), where an agent uses a full physics engine as a model representation. PBRL leverages several decades of progress at distilling physical principles into useful computational tools that make it possible to simulate a wide array of natural phenomena, including rigid and articulated bodies (Liu, 2013), fabric (Bhat et al., 2003), and even fluids (Stam, 1999). These tools encode the differential dynamics and constraints that govern rigid-body behavior and, like the OO-MDP, yield a model parameterization in terms of a space of object properties. Modern simulators thus offer a large, but structured, hypothesis-space for physical dynamics. By drawing on these simulation tools, PBRL can offer a compact and accurate description of relational dynamics for physical systems.

We compare PBRL to OO-LWR, a generalization of OO-MDP that uses Locally-Weighted Regression as a core dynamics model. Our results show that PBRL is considerably more sample efficient than OO-LWR, potentially leading to qualitatively different behavior on large physical systems.

2. Preliminaries: OO-MDPs and State-Spaces

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

Object Oriented Markov Decision Processes. An MDP is defined by $\langle S, A, T, R, \gamma \rangle$ for a state space S, action space A, transition function $T(s,a) \to P(s)$, reward function $R(s) \to \mathcal{R}$, and discount factor $\gamma \in [0,1)$. The agent tries to find a policy $\pi(s) \to a$ which maximizes long-term expected reward $V_\pi(s) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right]$ for all s.

In model-based RL, the agent is uncertain about some of the components of the underlying MDP and must refine its knowledge from experience. If information about parameters is represented in parametric form and updated in accordance to Bayes' rule with new information, it is referred to as Bayesian RL (BRL) (Vlassis et al., 2012). In this paper we are primarily interested in the transition model T, and will consider several possible model priors $P(T)$. The overall object manipulation problem then is to use observed transition samples to update model parameters, and plan using a learned T.

The primary objective behind OO-MDPs is to exploit the unique pattern of stationarity in the dynamics model of tasks with multiple interacting bodies. Unlike previous methods for factoring MDP dynamics, such as Dynamic Bayesian Networks, the OO approach allows model dependencies to vary throughout the state-action space. For example, the next-state of a chair in a kitchen only depends on the state parameters of other chairs if it is about to collide with them. To capture this, an OO-MDP defines a set of object classes $\mathcal{C} = \{C_1, \ldots, C_c\}$ (*e.g. table*, *chair*, *wall*), attributes $\mathcal{A}(\mathcal{C}) = \{C.a_1, \ldots, C.a_n\}$ (*e.g.* wheels-locked), and relations $r : C_i \times C_j \to$ Boolean (*e.g.* $\mathit{contact\text{-}left}(\mathit{chair}, \mathit{wall})$) enumerating the possible relationships between objects.

These relations allow dependencies to be formalized as a collection of functions that convert a state into a set of boolean literals, the "condition" associated with that state:

$$\mathit{Cond}(s,a) \mapsto \{p_1(s,a), p_2(s,a), \ldots, p_n(s,a)\}$$

In this fashion, the OO-MDP uses a collection of first-order expressions to partition the state-action space into sets with *homogeneous dynamics*. Model learning then amounts to learning the action effects under each condition, where a condition is a particular assignment to the OO-MDP predicates and attributes, rather than for each state.

Physical domains and State-Space Dynamics. This paper is concerned with physical planning, such as object manipulation with mobile robots or sprites in physically-realistic video games. The low-level state representation in these domains is typically the position and velocity of one or more rigid bodies. This representation is referred to as the *state-space* representation in control theory, and comes from Newton-Euler dynamics (Sontag, 1998). Actions correspond to the forces and torques that can be applied to these bodies to move them. Standard notation defines a *state-space* in terms of state x and control u:

$$\dot{x} = f(x,u) \tag{1}$$

where $\dot{x}$ is the first time-derivative of the state; however, in order to remain consistent with the RL literature we will use s to denote the state, a to denote actions, and s' to denote next state. This corresponds to the discrete-time version of Eq. 1, as is common in the controls literature, and is obtained by applying $f(s,a)$ for a finite time-interval. When state vectors include more than one object we use superscripts to indicate the state dimensions, (*e.g.* s^i for object i).

In general we assume that the agent can apply forces directly to one object at a time (though objects may interact via contact). Consequently, an action is uniquely defined by a force and torque vector and a target-object identifier. In two dimensions, we require six parameters to represent the state s^i: two for position in the (x,y) plane, one for orientation θ, and three for their derivatives. An action requires four parameters: two for the force $\langle f_x, f_y \rangle$, one for a torque τ_θ, and one for a target object index i. For k objects this results in the overall model signature of:

$$f(\mathbb{R}^{6k+4}) \to \mathbb{R}^{6k} \tag{2}$$

3. A Physics-Based Approach

The central focus of this paper is to identify and exploit the structure of physical object dynamics. To motivate our approach, we first describe OO-LWR, an implementation of OO-MDP generalized to handle state-space dynamics.

3.1. Object-Oriented Locally-Weighted Regression

3.1.1. Collision Predicates

In the original OO-MDP, contact predicates were tied to the adjacency properties of states arranged in a grid; however, in reality objects can come into contact from any orientation, and react according to where the collision occurred in the coordinate-frame of the object. To handle this reality, the object boundary can be discretized into a set of n_s contact sectors, each of which is assigned a contact predicate:

$$\mathit{contact}_{\theta_1}(\mathit{obj}), \ldots, \mathit{contact}_{\theta_n}(\mathit{obj})$$

Importantly, sector predicates must be computed in the coordinate frame of the object. For example, the reaction of a shopping cart to a collision depends on its direction relative to its wheels, not the grocery store in which it is located.

Fig. 1 illustrates sector-based collision detection applied to objects in an apartment. Lines indicate the level of discretization, and dark (red) dots represent the sectors in collision. The number of sectors n_s is a free parameter of the model.

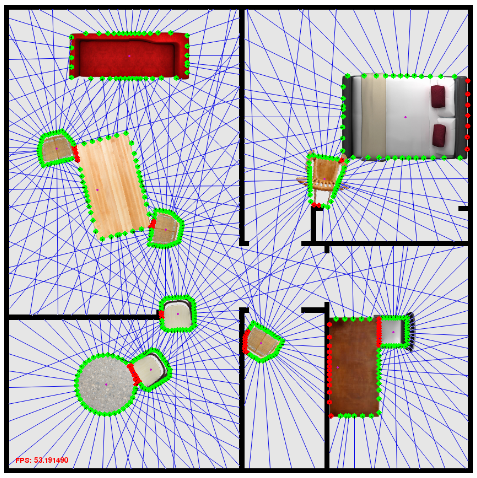


Figure 1. Detecting collision sectors for contact predicates (OO-LWR) in an apartment task.

3.1.2. Local State-Space Models

As mentioned in Section 2, *state-space* dynamics are differential, defining displacements from the current state. Objects can also have differential constraints (*e.g.* wheels) causing transitions to be non-linear in the state and action parameters; therefore, the effect model must be (a) compatible with the *state-space* representation, (b) invariant to object pose, and (c) capable of representing non-linear functions.

The building block for *state-space* regression models matching Eq. 2 are scalar-valued predictors of the form $f(\mathbb{R}^{n+4}) \rightarrow \mathbb{R}^1$ for each dimension of each object. To simplify notation here we use the superscript to denote individual state dimensions of a single object, rather than an object index (*e.g.* s_1^1 denotes the x coordinate of object 1 at time $t = 0$, not the full state of object 1). For a single object the function $f(s_t, a_t) \rightarrow s_t'$ can be written as:

$$\begin{bmatrix} f^1(s_t, a_t) + \epsilon \\ \cdots \\ f^n(s_t, a_t) + \epsilon \end{bmatrix} = \begin{bmatrix} s_t^{1\prime} \\ \cdots \\ s_t^{n\prime} \end{bmatrix} \tag{3}$$

For these individual predictors we use locally-weighted regression (we summarize the main properties of LWR here, but for a more thorough overview see (Nguyen-Tuong & Peters, 2011)). LWR is a kernel-based generalization of linear regression that permits interpolation of arbitrary non-linear functions. In LWR, a kernel function is used to compute a positive distance $w_i = k(X^*, X^i)$ between a query point X^* and each element X^i of the training set, which are collected into a diagonal matrix W. Kernels are typically decreasing functions of distance from the query, such as the Gaussian or "squared-exponential": $k(X^*, X^i) \propto e^{-(X^* - X^i)^2/\lambda}$.

Defining the training data $\tilde{X} := [s, a]_{t=0}^T$ and $y := [s']_{t=0}^T$, and the query $\tilde{X}^* := [s^*, a^*]$, state predictions can be estimated for each output dimension with weighted least-squares:

$$\beta_i^* = ((\tilde{X}^T W \tilde{X})^{-1} \tilde{X}^T W {y_i}' \tag{4}$$

$$s_i' = \tilde{X}^{*T} \beta_i^* \tag{5}$$

In contrast to parametric approaches, the model parameters β^* are re-computed for each query. As a result, the regression coefficients may vary across the input space, allowing LWR to model nonlinear functions with linear machinery.

Pose invariance is achieved by first transforming s_t' and a_t' to the s_t frame, then dropping the position components of s_t. Transforming all observations in this fashion yields a displacement model for individual objects that generalizes across position, at the expense of being able to capture position-dependent effects. However, the only position-dependent effect in the domains we consider is collisions, which are handled by the contact predicates. Furthermore, learning collisions between *dynamic* bodies with LWR would require a single monolithic model over the joint *state-space* of all objects, which would require an infeasible number of observations. Therefore in OO-LWR we use a collection of independent single-body, pose-invariant LWR models.

In two-dimensions the resulting signature for a single-body LWR model is $f(\mathbb{R}^6) \rightarrow \mathbb{R}^6$, which computes a state displacement in the query frame (note that angular velocity is frame-independent in two-dimensions):

$$f(\dot{x}, \dot{y}, \dot{\theta}, f_x, f_y, \tau_\theta) \rightarrow (\delta x, \delta y, \delta\theta, \delta\dot{x}, \delta\dot{y}, \dot{\theta}) \tag{6}$$

The overall transition in the original state space is then obtained by transforming the local-frame position, orientation, and linear velocity back to the world-frame. In this fashion, LWR can exploit the geometric nature of the *state-space* representation, where coordinate transformations are internal to the regression model (challenge 1). By building a regression model in which a given output dimension can depend on multiple state dimensions, this approach can also effectively handle integration effects (challenge 2). We now discuss how these models can be fit.

3.1.3. Fitting OO-LWR Models

Recall that the purpose of predicates in an OO-MDP is to segment the state-action space into sets with distinct object dynamics. The process of training an OO-LWR model on a history of observations $h = [s_t, a_t, s_{t+1}]_{t=0}^T$ is therefore achieved by assigning observations to effect models by *condition*, where a condition is a boolean string containing the output of all relations (e.g. collision predicates) applied to that state. For example, all training instances in which the front of a given chair is in collision should be assigned to the same condition. By using *state-space* regression for the individual effect models, OO-LWR is able to model rigid body motion for multiple objects; however, OO-LWR does not naturally admit a compact representation of the space of possible collisions. As we will see, there are considerable performance implications for larger domains.

3.2. Physics-based Reinforcement Learning

In PBRL the basic idea is to view a *physics engine* as a *hypothesis space* for arbitrary nonlinear rigid-body dynamics. This representation allows us to compactly describe transition uncertainty in terms of the parameters of the underlying physical model of the objects in the world. We capture this uncertainty using distributions over the relevant physical quantities, such as masses and friction coefficients, and obtain transitions by taking the expectation of the simulator's output over those random variables.

3.2.1. Physical Quantities as Latent Variables

At its core, a physics engine uses systems of differential equations to capture the fundamental relationship between force, velocity, and position. During each time step the engine is responsible for integrating the positions and velocities of each body based on extrinsic forces (*e.g.* provided by a robot), and intrinsic forces (*i.e.* differential constraints).

Differential constraints are ubiquitous in natural environments, and arise whenever bodies experience forces that depend on their configuration relative to one another. A wheel rolling along a surface, a door rotating around a hinge, and a train gliding along a track are all examples of differential constraints acting on a body. For RL purposes, these parameters provide attractive learning targets that may prove more efficient than more general functional forms, such as non-parametric regression.

In PBRL we model the *state-space* dynamics f in terms of the agent's beliefs over objects' inertial parameters and the existence and parametrization of physical constraints, such as wheels. Like a standard Bayesian regression model, this model includes uncertainty in the process input parameters (physical parameters) and in output noise. If $f(\cdot;\tilde{\Phi})$ denotes a deterministic physical simulation parameterized by $\tilde{\Phi}$, then the core dynamics function is:

$$s_{t+1} = f(s_t, a_t; \tilde{\Phi}) + \epsilon \tag{7}$$

where $\tilde{\Phi} = (\tilde{\phi})_{i=1}^{n}$ denotes a full assignment to the relevant physical parameters for all n objects in the scene, and ϵ is zero-mean Gaussian noise with variance σ^2.

For any domain, $\tilde{\Phi}$ must contain a core set of inertial parameters for each object, as well as zero or more constraints. Inertial parameters define rigid body behavior in the absence of interactions with other objects, and constraints define the space of possible interactions.

In the general case inertia requires 10 parameters; 1 for the object's mass, 3 for the location of the center of mass, and 6 for the inertia matrix; however, if object geometry is known, we can reduce this to a single parameter m by assuming uniform distribution of mass.[1] This is sufficient for our purposes (for a full parametrization see (Niebergall & Hahn, 1997; Atkeson et al., 1986)).

We focus on three types of constraints that arise frequently in mobile manipulation applications: anisotropic friction, distance, and non-penetration.

Anisotropic friction is a velocity constraint that allows separate friction coefficients in the x and y directions, typically with one significantly larger than the other. An anisotropic friction joint is defined by the 5-vector $J_w = \langle w_x, w_y, w_\theta, \mu_x, \mu_y \rangle$, corresponding to the joint pose in the body frame, and the two orthogonal friction coefficients. Anisotropic friction constraints can be used to model wheels, tracks, and slides.

A distance joint is a position constraint between two bodies, and can be specified with a 6-vector $J_d = \langle i_a, i_b, a_x, a_y, b_x, b_y \rangle$ which indicates the indices of the two target objects a and b as well as a position offset in each body frame. Distance joints can be used to model orbital motion, such as hinges or pendulums.

Non-penetration, or contact constraints, are responsible for ensuring objects react appropriately when they come into contact. Object penetration is detected during state integration based on object geometry, and is resolved by computing two types of collision-forces. The first is normal to each collision surface, and pushes objects apart. The magnitude of this force is controlled by the coefficient of restitution r, which is a rigid-body property that can be interpreted as "bounciness". The second is tangential to each collision surface, which captures contact friction and allows transfer of angular momentum. This force is proportional to a contact-friction coefficient μ_c.

In general this model-space is over-complete: not all bodies will have both hinges and wheels. The model must therefore allow constraint effects to be added and removed. This can be accomplished by including auxiliary variables for represented components, *e.g.* using a Dirichlet Process prior on constraints; however, this issue can be avoided for cases where the effects of interest can be represented with a finite number of constraints, and where individual constraints can be nullified for certain parameter settings.

We satisfy these conditions by including only a single wheel constraint, and bounding the number of distance constraints by the number of unique pairs of objects. One wheel is sufficient for modeling the bodies typically found

[1]Mass is often parameterized in this fashion in modern simulation tools, such as Box2D (Catto, 2013)

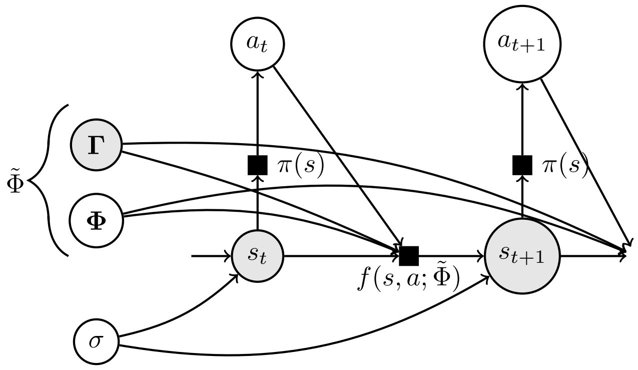


Figure 2. Graphical model depicting the online model learning problem, and the assumptions of PBRL, in terms of states s and actions a. Latent variables Γ (geometric properties) and Φ (dynamics properties) parameterize the full time-series model. $\pi(\cdot)$ denotes the policy and $f(\cdot)$ denotes the dynamics function. We assume Γ to be observable.

indoors, such as shopping carts and wheel chairs, because they have only one constrained axis (multiple coaxial wheels can be expressed by a single constraint). The wheel can be nullified by zeroing the friction coefficients, and the distance constraints can be nullified by setting $i_a = i_b$.

In summary, our dynamics model for a single body is represented by a set ϕ containing the mass m, restitution r, contact-friction μ_c, plus k distance constraints J^d, and a single anisotropic constraint J^w.

$$\phi := \{m, r, \mu_c\} \cup \{J^d\}^k \cup \{J^w\}^1 \tag{8}$$

Fig. 2 illustrates our approach and modeling assumptions. We split object parameters into two sets according to whether they are potentially observable by the agent. The first, Φ, denotes the *un-observable* physical properties that are needed to parameterize object dynamics, such as friction and mass. The second, Γ, describes geometric information such as polygons or meshes, and are needed to compute inertial forces and collision effects. Note that these both describe physical object *properties*, and are distinct from object *state* parameters (position and velocity). We then define $\tilde{\Phi} = \Phi \cup \Gamma$ as the full set of object properties which are sufficient to parametrize the physical dynamics of all objects in the model.

Inferring Φ from s and a is the model learning problem, and is the focus of this work. Deciding a from s and Φ is the planning problem, which we consider in Section 4. Inferring s and Γ from sensor observations is the vision problem, which is outside the scope of this work.

In summary, PBRL provides a model prior for object dynamics in terms of a small set of latent physical parameters. The goal of this approach is expressiveness, and the core technical challenge is estimating Φ from time-series data, considered next.

Property $(*)$	**Distribution**
m, r	Log-Normal(μ_*, σ_*^2)
μ_c, μ_x, μ_y	Truncated-Normal$(\mu_*, \sigma_*^2, 0, 1)$
w_x, w_y, a_x, a_y	Truncated-Normal$(\mu_*, \sigma_*^2, a_{min}^{xy}, a_{max}^{xy})$
b_x, b_y	Truncated-Normal$(\mu_*, \sigma_*^2, b_{min}^{xy}, b_{min}^{xy})$
w_θ	Von-Mises$(\mu_{w_\theta}, \kappa_{w_\theta})$
i_a, i_b	Categorical(p_*)

Table 1. Univariate distributions for each physical parameter, with $*$ used to indicate subscripting for the appropriate property.

3.2.2. A Prior Over Physical models

In order to fully specify a PBRL model we must assign priors over each parameter of each body to restrict support to legal values. Mass m and restitution r can take values in $\mathbb{R}^+$, all friction coefficients $\{\mu_c, \mu_x, \mu_y\}$ can take values in $[0, 1]$, all position-offset parameters $\{w_x, w_y, a_x, a_y, b_x, b_y\}$ can take values within the bounds of the appropriate object, orientation $\{w_\theta\}$ can take values in $[-\pi, \pi]$, and index i_a, i_b can take values in $\{1, \ldots, k\}$ for k objects in the world. To represent the agent's beliefs over these parameters, we assign the distributions denoted in Table 1 for each object. In general this model prior would be initialized with uninformative values, and be updated from posterior statistics as the agent receives data, considered next.

3.2.3. Fitting physical models

Now we consider inferring physical parameters Φ from a history of manipulation data $\{s_t, a_t, s_{t+1}\}_{t=0}^T$. Let h denote a matrix of observed transitions:

$$h = \begin{bmatrix} s_1 & a_1 & s_1' \\ s_2 & a_2 & s_2' \\ \vdots & \vdots & \vdots \\ s_T & a_T & s_T' \end{bmatrix} \tag{9}$$

We should use h to update the the agent's beliefs about Φ and the noise term σ. In a Bayesian approach this is expressed as the model posterior given history h:

$$P(\Phi, \sigma | h) = \frac{P(h|\Phi, \sigma)P(\Phi)P(\sigma)}{\int_{\Phi,\sigma} P(h|\Phi, \sigma)P(\Phi)P(\sigma)} \tag{10}$$

where $\Phi = \{\phi_1, \phi_2, \ldots, \phi_k\}$ is the collection of hidden parameters for the k objects in the domain, and σ is a scalar. This expression is obtained from Bayes' rule, and defines the abstract model inference problem for a PBRL agent.

The prior $P(\Phi)$ can be used to encode any prior knowledge about the parameters, and is not assumed to be of any particular parametric form. For a particular assignment to Φ, Eq. 7 implies a Gaussian likelihood over next states:

$$P(h|\Phi,\sigma) = \prod_{t=1}^{n} P(s'_t|\Phi,\sigma,s_t,a_t)$$
$$= \prod_{t=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(s'_t - f(s_t,a_t;\tilde{\Phi}))^2}{2\sigma^2}\right) \tag{11}$$

Eq. 11 tells us that the likelihood for proposed model parameters are evaluated on a Gaussian centered on the predicted next state for a generative physics world parameterized by $\tilde{\Phi}$ (*i.e.*, with known geometry and proposed dynamics). Due to Gaussian noise, the log-likelihood for Φ is obtained by summing squared distances between the observed value and the predicted state for each state and action:

$$\ln P(h|\Phi,\sigma) \propto -\sum_{t=1}^{n}\left((s'_t - f(s_t,a_t;\tilde{\Phi})\right)^2 \tag{12}$$

Along with the prior defined in Table 1, this provides the necessary components for a Metropolis sampler for Eq. 10. These posterior samples can then be used by a (stochastic) planner, which we consider next.

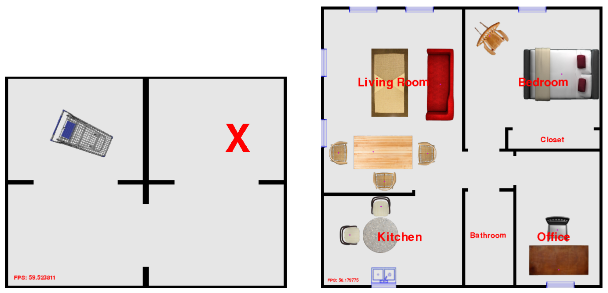
For planning, transition samples from a PBRL model can be obtained by first sampling the physical parameters Φ from the model posterior, stepping the physics world for the appropriate state and action, and (optionally) sampling the output noise. If $P(\Phi,\sigma|h)$ represents the agent's current model beliefs given a history of observations h, the full generative process for sampling transitions in PBRL is:

$$\begin{aligned} \Phi,\sigma \sim &\quad P(\Phi,\sigma|h) \\ \epsilon \sim &\quad N(0,\sigma^2) \\ s_{t+1} = &\quad f(s_t,a_t;\tilde{\Phi}) + \epsilon \end{aligned} \tag{13}$$

4. Multi-Body State-Space Planning

Planning in object-oriented physical domains follows the typical structure of model-based RL algorithms: an agent uses sampled transitions to construct a domain model, and selects actions using this model with a planning algorithm. Optimal planning and control for high-dimensional non-linear physical systems with differential constraints is an open problem in optimal control and robotics (Sontag, 1998).

A popular approach for robotics domains is gradient-based policy-search, such as the PILCO framework (Deisenroth et al., 2013). Despite being policy based, PILCO can handle collisions and multiple objects with an appropriate choice of shaping potentials for pushing objects away from obstacles (Deisenroth et al., 2011); however, these methods require gradients of the cost function, which for the model-based case requires that the dynamics function be differentiable. At present we have not explored methods for obtaining derivatives of physical models parameterized by



(a) Shopping Cart (b) Apartment

Figure 3. Simulated manipulation domains

physical properties although this is an exciting direction of future work which is complementary to the model-learning problem considered here.

For simplicity we turn to forward-search, value-based planning. In principle, sparse-sampling and other Monte-Carlo methods are compatible with our domains and modeling assumptions. However, the domains we are interested were too complex to achieve reasonable results with these methods, even by coarsely discretizing the state space. We therefore obtained the results presented below using A* (LaValle, 2006), which had access to the ML estimates of OO-LWR and the MAP estimates from PBRL. Note that although A* discretized the state space for the sake of planning, transitions were still computed using the full continuous representation (to machine precision).

5. Evaluation

We evaluate OO-LWR and PBRL w.r.t noise sensitivity, scalability, and model mis-specification (PBRL only). We use a realistic two-dimensional physical simulation based on the Box2D engine (Catto, 2013) extended to include anisotropic friction. Fig. 3 shows our two domains: a single-body world containing a shopping cart, and a multi-body world containing several types of household furniture. In all cases reward is proportional to the L_2 distance of the objects from user-defined goal configurations.

5.1. Shopping Cart Task

The first task is to push a shopping-cart to the goal configuration marked by the red cross in Fig. 3(a). The cart was modeled as a single body with a wheel constraint parallel to the handle axis, and behaved similarly to a real shopping cart which can pivot around points along the wheel axis, but can not translate along the same axis. Because the cart can collide with the wall, the model must be able to handle collisions. It must also be capable of modeling the non-linear behavior of the cart with sufficient accuracy to produce a plan over the long horizon of the task.

We present results under two learning conditions for this task: one which is noise-free, and a second in which the training observations were corrupted by Gaussian noise ($\sigma = 0.25$). In addition to our primary comparison of PBRL and OO-LWR, we also include the performance of an agent directly using the LWR model described in Section 3.1.2. This was included to decouple the object-oriented approach from the use of an effect model which can model state integration. An agent given access to the true model is provided as a baseline.

Fig. 4(a) shows the online performance of agents using each of these models in the noise-free condition. Each trace represents an average over 10 episodes. At each step the agent received an observation, updated its model, and selected a new action using an A* planner.

In the absence of noise the PBRL agent was able to recover the true model after two steps, and performed nearly as well as the baseline agent. The OO-LWR agent was slower to learn, but also reached the goal configuration. We note that despite attaining the same final reward, the online behavior of these agents was very different. Because OO-LWR had to (re)learn a separate model for each collision sector, it tended to bump into walls. This behavior was not penalized in the reward function here, but in situations where collisions are undesirable are during learning, the margin between PBRL and OO-LWR could be considerably higher.

The LWR agent failed to reach the goal configuration because it lacked the ability to model collisions, due to pose-invariance. It was therefore greedy with respect to the reward function, and the value at which it plateaus corresponds to the distance of the wall separating the start and goal configurations.

In the presence of training noise we observed the same overall pattern of results, but with more gradual learning as was required to average out the noise. What appears to be a small steady-state error for the OO-LWR agent was in fact due to this trace averaging across runs, some of which had not obtained sufficient accuracy to plan a successful path around the wall.

5.2. Apartment Rearrangement Task

The next task is a multi-object rearrangement problem in a simulated apartment, and demonstrates the behavior of both methods at larger scales. The apartment task contains 11 objects with various shapes and physical properties, including fixed wheels (dining table, office desk), large mass (couch, bed), small mass (chairs), and a revolute constraint (kitchen table).

The prior for PBRL is a categorical distribution defined over a collection of pre-learned modes from individual object trials. This was done because sampling a joint set of object parameters under the continuous prior in Table 1 using MCMC was very slow to mix. Addressing this issue with more sophisticated mixture-based priors and sampling methods will be a topic for future work.

Fig. 4(c) compares the online performance of PBRL and OO-LWR on the apartment task. In this domain, the inefficiency of OO-LWR is apparent: even after 1000 observations the OO-LWR agent was incapable of modeling domain dynamics with sufficient accuracy to produce a valid plan. This result is not surprising, given that OO-LWR requires $2^{|O|n_s}$ separate effect models to fully describe the collision space over $|O|$ objects. However, the physics-based representation of collision dynamics yields qualitatively different behavior. In contrast to the predicate-based approach, the PBRL agent quickly obtained an accurate estimate of full relational dynamics of the task, and produced a viable plan.

6. Related Work

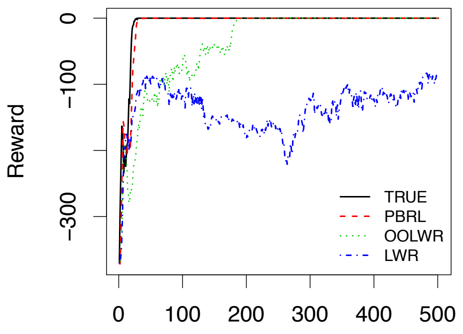
The Relocatable-Action MDP (RAMDP) (Leffler et al., 2007) proposed a clustering method for generalizing action effects across states. This was successfully applied to robot-navigation in a small domain (without velocity). In each dynamics regime (wood, cloth, or collision), robot motion was sufficiently consistent to cluster together, resulting in a more compact model. The core strength of this approach, in contrast to for example Factored-MDPs (Degris et al., 2006), is that statistical dependency between attributes is no longer stationary but rather on a function which is evaluated at each query state. The OO-MDP (Diuk et al., 2008) can be viewed as a successor to this idea, which formalizes the state-clustering process using first-order predicates, and introduces object attributes as arguments to these predicates.

The idea of estimating physical parameters from data has a rich history in the robotics, graphics, and computer vision literature. It arises in vision for model-based tracking (Kakadiaris & Metaxas, 2000; Duff et al., 2010), and in graphics for data-driven tuning of simulation parameters *e.g.* for cloth simulation (Bhat et al., 2003), rigid-body motion (Bhat et al., 2002), and even humanoid motion (Liu et al., 2005).

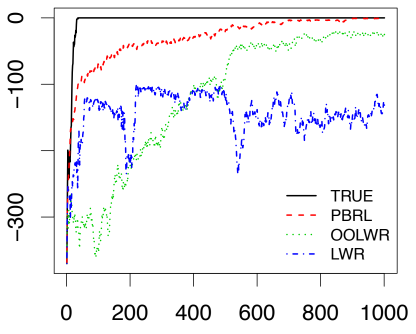
The challenge of controlling an initially unknown system and estimating its relevant parameters online has also been addressed within the controls subfield indirect adaptive control (Landau, 2011). Adaptation is typically done in two stages. In the first stage, the dynamical system parameters are estimated using a Parameter Adaptation Algorithm (PAA). In the next stage, these parameter estimates are used to update the controller. While most PAA methods assume a linear mode (Landau, 2011), PBRL can be seen as an PAA method supporting non-linear model estimation using Bayesian approximate inference. As such,

	k	λ	n_s	n_{tps}	ϵ_c	prior	MCMC
Shopping Cart	1000	1.5	4	10	20	continuous	2e4,1e3,10,30
Apartment	1000	1.5	4	10	20	categorical	5e3,1e3,10,1

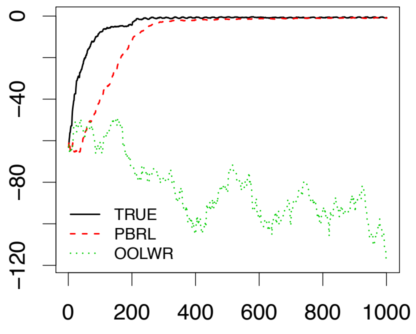
Table 2. Table of the relevant algorithm parameters for each experiment. k: number of nearest neighbors (LWR,OO-LWR), λ: bandwidth (LWR,OO-LWR), n_s: number of sectors (OO-LWR), n_{tps}: number of raycast collision tests per sector (OO-LWR), ϵ_c: collision radius (OO-LWR), prior: type of prior (PBRL), MCMC: sampler parameters (iterations, burn-in, thin, number of chains) (PBRL).



(a) Shopping-Cart task



(b) Shopping-Cart with noisy training data



(c) Apartment task

Figure 4. Online performance of various agents under different domain sizes and training conditions.

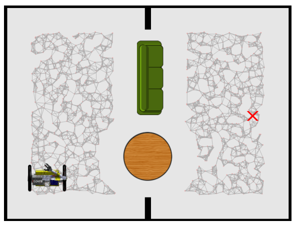
PBRL is complementary to the existing controls literature.

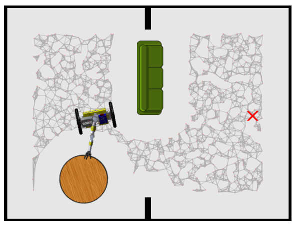
7. Discussion and Conclusions

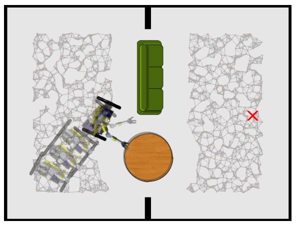
In this paper we presented two physics-inspired approaches to modeling object dynamics for physical domains. The first, OO-LWR, leveraged only the geometric properties of physical dynamics, and the second extended this by exploiting modern physical simulation methods. Our results suggest that PBRL has a learning bias which is well matched to RL tasks in physical domains.

An example of a reasoning pattern enabled by a PBRL representation is illustrated in Fig. 5, which depicts a Navigation Among Movable Obstacles (NAMO) problem (LaValle, 1998). In NAMO the task is to find a minimum-cost path to a goal position which may be obstructed by movable obstacles. If the robot begins with no knowledge of the dynamics of these obstacles, it can benefit from the learning efficiency of the PBRL approach. We demonstrated this in (Levihn et al., 2012; 2013) in which a preliminary version of PBRL enabled a robot to quickly infer that a round table is indeed static, without having to try every action at its disposal.

Extending this work will require broadening the set of physical models supported by a single PBRL prior. However, this greater expressiveness comes at the cost of a larger parameter space. In order to be feasible for online applications, our goal is to find the right balance between over-precise physical models which are brittle and hard to fit, and coarse models that lack expressive power. We feel that the wheel model presented here provides such an ex-


(a)


(b)


(c)

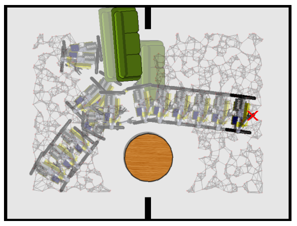

(d)

Figure 5. (a) Initial state (b) expected outcome (c) actual outcome; model updated (d) final solution

ample for furniture-like applications. However, in real-world scenarios, it may be useful to incorporate the flexibility of non-parametric methods into a PBRL approach, in order to guard against model mis-specification.

More generally, PBRL can be viewed as an ontological constraint on the world model: it is governed by the laws of physics. We hope that this approach helps to close the representational gap between the sorts of models used in Reinforcement Learning and the models that robotics engineers use in practice. If successful, this approach may yield opportunities for learning representations that are currently engineered by hand in robotics.

References

Atkeson, C., An, C., and Hollerbach, J. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3), 1986.

Bhat, K., Seitz, S., Popović, J., and Khosla, P. Computing the physical parameters of rigid-body motion from video. *ECCV*, 2002.

Bhat, K.S., Twigg, C.D., Hodgins, J.K., Khosla, P.K., Popović, Z., and Seitz, S.M. Estimating cloth simulation parameters from video. In *SIGGRAPH*. Eurographics Association, 2003.

Catto, Erin. Box2D physics engine, 2013. URL http://www.box2d.org.

Degris, T., Sigaud, O., and Wuillemin, P. Learning the structure of factored markov decision processes in reinforcement learning problems. In *ICML*. ACM, 2006.

Deisenroth, M., Rasmussen, C., and Fox, D. Learning to control a low-cost manipulator using data-efficient reinforcement learning. 2011.

Deisenroth, M., Neumann, G., and Peters, J. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2013.

Diuk, C., Cohen, A., and Littman, M.L. An object-oriented representation for efficient reinforcement learning. In *ICML*. ACM, 2008.

Duff, D.J., Wyatt, J., and Stolkin, R. Motion estimation using physical simulation. In *ICRA*. IEEE, 2010.

Kakadiaris, L. and Metaxas, D. Model-based estimation of 3d human motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12), 2000.

Landau, Ioan Doré. *Adaptive control: algorithms, analysis and applications*. Springer, 2011.

LaValle, S. Rapidly-exploring random trees a new tool for path planning. 1998.

LaValle, S. *Planning algorithms*. Cambridge university press, 2006.

Leffler, Bethany R, Littman, Michael L, and Edmunds, Timothy. Efficient reinforcement learning with relocatable action models. In *Proceedings of AAAI*, volume 22. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

Levihn, M., Scholz, J., and Stilman, M. Hierarchical decision theoretic planning for navigation among movable obstacles. In *WAFR*, 2012.

Levihn, M., Scholz, J., and Stilman, M. Planning with movable obstacles in continuous environments with uncertain dynamics. In *ICRA*, May 2013.

Liu, C.K. Dart: Dynamic animation and robotics toolkit, 2013. URL https://github.com/dartsim/dart/wiki.

Liu, C.K., Hertzmann, A., and Popović, Z. Learning physics-based motion style with nonlinear inverse optimization. In *ACM Transactions on Graphics (TOG)*, volume 24. ACM, 2005.

Nguyen-Tuong, D. and Peters, J. Model learning for robot control: a survey. *Cognitive processing*, 12(4), 2011.

Niebergall, M and Hahn, H. Identification of the ten inertia parameters of a rigid body. *Nonlinear Dynamics*, 13(4), 1997.

Sontag, E. *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer, 1998.

Stam, J. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999.

Vlassis, N., Ghavamzadeh, M., Mannor, S., and Poupart, P. Bayesian reinforcement learning. In *Reinforcement Learning*. Springer, 2012.