SWIRL: A Sequential Windowed Inverse Reinforcement Learning Algorithm for Robot Tasks With Delayed Rewards

Sanjay Krishnan, Animesh Garg, Richard Liaw, Brijen Thananjeyan,
Lauren Miller, Florian T. Pokorny*, Ken Goldberg

The AUTOLAB at UC Berkeley (automation.berkeley.edu)
*CAS/CVAP, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract. Inverse Reinforcement Learning (IRL) allows a robot to generalize from demonstrations to previously unseen scenarios by learning the demonstrator's reward function. However, in multi-step tasks, the learned rewards might be delayed and hard to directly optimize. We present Sequential Windowed Inverse Reinforcement Learning (SWIRL), a three-phase algorithm that partitions a complex task into shorter-horizon subtasks based on Switched Linear Dynamical transitions that occur consistently across demonstrations. SWIRL then learns a sequence of local reward functions that describe the motion between transitions. Once these reward functions are learned, SWIRL applies Q-learning to compute a policy that maximizes the rewards. We compare SWIRL (demonstrations to segments to rewards) with Supervised Policy Learning (SPL - demonstrations to policies) and Maximum Entropy IRL (MaxEnt-IRL demonstrations to rewards) on standard Reinforcement Learning benchmarks: Parallel Parking with noisy dynamics, Two-Link acrobot, and a 2D GridWorld. We find that SWIRL converges to a policy with similar success rates (60%) in 3x fewer time-steps than MaxEnt-IRL, and requires 5x fewer demonstrations than SPL. In physical experiments using the da Vinci surgical robot, we evaluate the extent to which SWIRL generalizes from linear cutting demonstrations to cutting sequences of curved paths.

1 Introduction

One of the goals of learning from demonstrations (LfD) is to learn policies that generalize beyond the provided examples and are robust to perturbations in initial conditions, the environment, and sensing noise [1]. Inverse Reinforcement Learning (IRL) is a popular framework, where the goal is to infer an unknown reward function from a set of demonstrations [2, 3, 4]. Once a reward is learned, given novel instances of a task, a policy can be computed by optimizing for this reward function using an approach like Reinforcement Learning (RL) [5, 4].

In IRL, a task is modeled as an MDP with a single unknown function that maps states and actions to scalar values. This model is limited in the way that it can represent *sequential tasks*, tasks where a robot must reach a sequence of intermediate state-space goals in a particular order. The sequential structure can facilitate faster learning because the inferred reward may be *delayed* and reflect a quantity observed after all of the goals are reached, and thus, making it very difficult to optimize directly. Furthermore, there may not exist a single stationary policy for a given state-space (a time-invariant map

between states and actions) that achieves all of the goals in sequence, e.g., a figure-8 trajectory in the x,y plane.

To address this problem, one approach is to divide the task into segments with local reward functions. In existing work on multi-step IRL, this sequential structure is defined manually [2]. We propose an approach that automatically learns sequential structure and assigns local reward functions to segments. The combined problem is nontrivial because solving k independent problems neglects the shared structure in the value function during the policy learning phase (e.g., a common failure state). However, jointly optimizing over the segmented problem inherently introduces a dependence on history, namely, any policy must complete step i before step $i+1$. This potentially leads to an exponential overhead of additional states.

Sequential Windowed Inverse Reinforcement Learning (SWIRL) is based on a model for sequential tasks that represents them as a sequence of reward functions $\mathbf{R}_{seq} = [R_1, ..., R_k]$ and transition regions (subsets of the state-space) $G = [\rho_1, ..., \rho_k]$ such that R_1 is the reward function until ρ_1 is reached, after which R_2 becomes the reward and so on. SWIRL assumes that demonstrations have locally linear dynamics w.r.t a provided feature space, are locally optimal (as in IRL), and all demonstrations reach each $\rho \in G$ in the same sequence. In the first phase of the algorithm, SWIRL infers the transition regions using a kernelized variant of an algorithm proposed in our prior work [6, 7]. In the second phase, SWIRL uses the inferred transition regions to segment the set of demonstrations and applies IRL locally to each segment to construct the sequence of reward functions $\mathbf{R}_{seq}$. Once these rewards are learned, SWIRL computes a policy using an RL algorithm (Q-Learning) over an augmented state-space that indicates the sequence of previously reached reward transition regions. We show that this augmentation has an additional space complexity independent of the state-space and linear in the number of rewards.

Our contributions are:

1. A three-phase algorithm, SWIRL, to learn policies for sequential robot tasks.
2. An extension of the Transition State Clustering algorithm that relaxes the local linearity assumption using kernelization.
3. A novel state-space augmentation to enforce sequential dependencies using binary indicators of the previously completed segments, which can be efficiently stored and computed based on the first phase of SWIRL.
4. Simulation and physical experiments comparing SWIRL with Supervised Learning and MaxEnt-IRL.

2 Related Work

The seminal work of Abbeel and Ng [4] explored learning from demonstrations using Inverse Reinforcement Learning. In [4], the authors used an IRL algorithm to infer the demonstrator's reward function and then an RL algorithm to optimize that reward. Our work re-visits this two-phase algorithm in the context of sequential tasks. It is well-established that RL problems often converge slowly in complex tasks when rewards are sparse and not "shaped" appropriately [8, 9]. These issues are exacerbated in sequential tasks where a sequence of goals must be reached. Related to this problem, Kolter et al. studied *Hierarchical Apprenticeship Learning* to learn bipedal locomotion [2], where the algorithm is provided with a hierarchy sub-tasks. These sub-tasks are not learned

from data and assumed as given, but the algorithm infers a reward function from demonstrations. SWIRL applies to a restricted class of tasks defined by a sequence of reward functions and state-space goals.

There are have been some proposals in robotics to learn motion primitives from data. The approaches assume that reward functions are given (or the problem can be solved with planning-based methods). Motion primitives are example trajectories (or sub-trajectories) that bias search in planning towards paths constructed with these primitives [10, 11, 12]. Much of the initial work in motion primitives considered manually identified segments, but recently, Niekum et al. [13] proposed learning the set of primitives from demonstrations using the Beta-Process Autoregressive Hidden Markov Model (BP-AR-HMM). Calinon et al. [14] proposed the task-parametrized movement model with GMMs for action segmentation. Both Niekum and Calinon consider the motion planning setting in which analytical planning methods are used to solve a task and not RL. Konidaris et al. studied the primitives in the RL setting [15]. However, this approach assumed that the reward function was given and not learned from demonstrations as in SWIRL. Another relevant result is from Ranchod et al. [16], who use an IRL model to define the primitives, in contrast to the problem of learning a policy after IRL.

3 Problem Statement and Model

3.1 Notation

Consider a finite-horizon Markov Decision Process (MDP):

$$\mathcal{M} = \langle S, A, P(\cdot,\cdot), \mathbf{R}, T \rangle,$$

where S is the set of states (continuous or discrete), A is the set of actions (finite and discrete), $P : S \times A \mapsto Pr(S)$ is the dynamics model that maps states and actions to a probability density over subsequent states, T is the time-horizon, and $\mathbf{R}$ is a reward function that maps trajectories of length T to scalar values.

Sequential tasks are tasks composed of sequences of sub-tasks. There is a sequence $\mathbf{R}_{seq} = [R_1, ..., R_k]$, where each $R_i : S \times A \mapsto \mathbb{R}$. Associated with each R_i is a transition region $\rho_i \subseteq S$. Each trajectory accumulates a reward R_i until it reaches the transition ρ_i (defined as the first time at which the robot's state is in ρ_k). This process continues until ρ_k is reached. A robot is deemed *successful* when all of the $\rho_i \in G$ are reached in sequence. Further, a robot is *optimal* when it maximizes the expected cumulative reward and is successful. Given observations of an optimally acting robot through a set of demonstration trajectories $D = \{d_1, ..., d_k\}$, can we infer $\mathbf{R}_{seq}$ and G?

Assumptions: We make the following assumptions: (1) the robot's dynamics are linearly parametrized (i.e., linear in some feature space), (2) every demonstration is generated from k distinct stationary, locally optimal policies (maximized w.r.t R_i on the infinite horizon), (3) every demonstration visits each ρ_i in the same sequence, and (4) each R_i is a quadratic of the form.

Remarks: The key challenge in this problem is determining when a transition occurs– identifying the points in time in each trajectory at which the robot reaches a ρ_i and transitions the reward function. The natural first question is whether this is identifiable, that is, whether it is even theoretically possible to determine whether a transition

$\rho_i \rightarrow \rho_{i+1}$ has occurred after obtaining an infinite number of observations. Trivially, this is not guaranteed when $R_{i+1} = R_i$, where it would be impossible to identify a transition purely from the robot's behavior (i.e., no change in reward, implies no change in behavior). Perhaps surprisingly, this is still not guaranteed even if $R_{i+1} \neq R_i$ due to policy invariance classes [8]. Consider a reward function $R_{i+1} = 2R_i$, which functionally induce the same optimal behavior. Therefore, we consider a setting where all of the rewards in $\mathbf{R}_{seq}$ are distinct and are not equivalent w.r.t optimal policies. This formalism is a special case of the Hierarchical Reinforcement Learning [17], where each of the local rewards is a sub-goal and arrival at a ρ is a termination condition.

3.2 Linear-Gaussian Model

Leveraging the four earlier assumptions, we can define a class of sequential tasks for which inference is tractable. The key insight is that under these three assumptions demonstration trajectories take the form of Switched Linear-Gaussian systems. Parameter inference for such systems is well-studied [18]. A demonstration d is a trajectory of state and action tuples $[(s_0, a_0), ..., (s_T, a_T)]$. We first require some regularity assumptions on the demonstrations. Let D be a set of demonstrations $\{d_1, ..., d_N\}$ of a sequential task.

Consider the following system:

$$x_{t+1} = Ax_t + Bu_t + w_t \quad w_t \sim N(0, \Sigma) \quad i.i.d$$

For quadratic rewards in the infinite horizon, the optimal policy is a linear state feedback controller $u_t = -Cx_t$ Given this model, suppose we wanted to control the robot to a final state μ_i with a linear state-feedback controller C_i, the dynamical system that would follow is:

$$\hat{x}_{t+1} = (A - BC_i)\hat{x}_t + w_t,$$

where $\hat{x}_t = x_t - \mu_i$. If this system is stable, it will converge to $x_t = \mu_i$ as $t \rightarrow \infty$. Now, suppose that the system has the following switching behavior: when $\|x_t - \mu_i\| \leq \varepsilon$, change the target state μ_i to μ_{i+1}. The resulting closed loop dynamics are:

$$A_i = (A - BC_i)$$

$$x_{t+1} = A_i \mathbf{x}_t + w_t : A_i \in \{A_1, ..., A_k\}.$$

The equation above defines an SLDS. This model maps back to the general case where the sequence $[\mu_1, ..., \mu_k]$ and their tolerances $[\varepsilon_1, ..., \varepsilon_k]$ define the regions $[\rho_1, ..., \rho_k]$. Each ρ_i corresponds to regions where transitions occur $A_i \neq A_j$. Intuitively, a change in the reward function results in a change of policy (C_i) for a locally optimal agent.

4 Sequential Windowed Inverse Reinforcement Learning

This section describes an algorithm to infer the parameters for the proposed model.

Algorithm Description Let D be a set of demonstration trajectories $\{d_1, ..., d_N\}$ of a task with a delayed reward. SWIRL can be described in terms of three sub-algorithms:

Inputs: Demonstrations D, Dynamics (Optional) P

1. **Sequence Learning:** Given D, SWIRL segments the task into k sub-tasks whose start and end are defined by arrival at a set of transitions $G = [\rho_1, ..., \rho_k]$.
2. **Reward Learning:** Given G and D, SWIRL associates a local reward function with the segment resulting in a sequence of rewards $\mathbf{R}_{seq}$.
3. **Policy Learning:** Given $\mathbf{R}_{seq}$ and G, SWIRL applies reinforcement learning for I iterations to learn a policy for the task π.

Outputs: Policy π

Phase I. Sequence Learning

SWIRL first infers $[\rho_1, ..., \rho_k]$ using an extension of our prior work on robust task segmentation [6, 7]. The overall procedure is summarized in Phase I.

Identifying Transitions Based on the assumptions described earlier, we formulate an SLDS inference problem. Suppose there are k sub-tasks, there will be k dynamical regimes. Consider a single demonstration trajectory $\mathbf{x}_t$ as a noisy observation from a dynamical system:

$$x_{t+1} = A_i \mathbf{x}_t + w_t : A_i \in \{A_1, ..., A_k\},$$

where w_t describes an i.i.d noise process. We want to identify the set of times t across all demonstrations where at t the dynamics matrix is A_i and at $t+1$ it is $A_j, j \neq i$. We model the transitions as instantaneous, and thus, in each demonstration, we have a discrete set of transition time points $\{0, ..., T\}$.

This can be inferred with expectation-maximization (EM) (refer to [19, 20, 21, 6, 7] for details). In typical EM formulations, one must specify the number of mixture components k before hand. However, we apply results from Bayesian non-parametric statistics and jointly solve for the component parameters and the number of components using a Dirichlet Process [22]. Using a DP, the number of components grows with the complexity of the observed data.

Relaxing Local Linearity In [6, 7], we assumed that each segment has locally linear dynamics. We relax the linear dynamics assumption with a kernel embedding of the trajectories. SWIRL does not require learning the exact regimes A_i, it only needs to detect changes in dynamics regime. The basic idea is to apply Kernelized PCA to the features before learning the transitions–a technique used in Computer Vision [23]. By changing the kernel function (i.e., the similarity metric between states), we can essentially change the definition of local linearity.

Let $\kappa(x_i, x_j)$ define a kernel function over the states. For example, if κ is the radial basis function (RBF), then: $\kappa(x_i, x_j) = e^{\frac{-\|x_i - x_j\|_2^2}{2\sigma}}$. κ naturally defines a matrix M where: $M_{ij} = \kappa(x_i, x_j)$. The top p' eigenvalues define a new embedded feature vector for each ω in $\mathbb{R}^{p'}$. We can now apply the algorithm above in this embedded feature space.

Transition Times To Regions We would like to be able to aggregate the transition times into state-space conditions for reward transitions $[\rho_1, ..., \rho_k]$. In the proposed model, these regions corresponded to a Euclidean ball around a target state μ_i, $\|x - \mu_i\| \leq \varepsilon$. Transition times correspond to the event when the trajectory arrives at the

Algorithm 1: Sequence Learning

Data: Demonstration $\mathcal{D}$

1 Fit a DP-SLDS model to $\mathcal{D}$ and identify the set of transitions Θ, defined as all (x_t, t) where $(x_{t+1}, t+1)$ has a different most likely dynamical regime.

2 Fit a DP-GMM to the states in Θ.

3 Prune clusters that do not have one transition from all demonstrations.

4 The result of is $G = [\rho_1, \rho_2, ..., \rho_m]$ where each ρ is a disjoint ellipsoidal region of the state-space and time interval.

Result: G

boundary of the ball. To each of these transition times, there is a corresponding *transition state*–the last state before the dynamics switched.

The distribution of the transition states can be used to approximate the location of μ_i and the radius ε. In general, this will be a complex distribution, and we approximate this with a Gaussian Mixture Model with k mixture components $\{m_1, ..., m_k\}$. As before, we use a DP to non-parametrically set k. Our prior work [6], describes several techniques to improve the robustness of this model such as pruning clusters that do not have at least one transition from each demonstration. Thus, the result is the set of transition regions: $G = [\rho_1, \rho_2, ..., \rho_k]$.

Remarks: In principle, the two-step algorithm (transition learning and region learning) could be described as a single graphical model. However, the coupled problem is quite challenging from an inference perspective and would require techniques such as MCMC. The proposed approach is related to variational approximations where the two parts are decoupled and treated as independent.

Phase II. Reward Learning

After Phase I, each demonstration is segmented into k sub-sequences. Phase II uses the learned $[\rho_1, ..., \rho_k]$ to construct the local rewards $[R_1, ..., R_k]$ for the task. As per our model, each R_i is a quadratic parametrized by a positive semi-definite matrix Q. SWIRL has two variants: when a dynamics model P is given, SWIRL applies Maximum Entropy IRL, and when the dynamics model is null (not-provided), SWIRL estimates the reward using the feature covariance. The Algorithm is summarized in Phase 2.

Model-based For the model-based approach, we use Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) [24]. The idea is to model every demonstration d_i as a noisy sample from an optimal policy. In other words, each d_i that is observed is a noisy observation of some hypothetical d^*.

Since each d_i is a path through a possibly discrete state and action space, we cannot simply average them to find d^*. Instead, we have to model trajectories that the system is likely to visit. This can be formalized with the following probability distribution:

$$P(d_i|R) \propto \exp\{\sum_{t=0}^{T} R(s_t)\}.$$

Paths with a higher cumulative reward are more likely.

Algorithm 2: Reward Learning

Data: Demonstration $\mathcal{D}$ and sub-goals $[\rho_1, ..., \rho_k]$

1 Based on the transition states, segment each demonstration d_i into k sub-sequences where the j^{th} is denoted by $d_i[j]$.

2 If dynamics model is available, apply MaxEnt-IRL to each set of sub-sequences $1...k$.

3 If the dynamics model is not available compute Equation 1 for each set of subsequences.

Result: $\mathbf{R}_{seq}$

MaxEnt-IRL uses the following linear parametrized representation:

$$R(s,a) = x^T \theta,$$

where x is the state vector. The resulting form is:

$$P(d_i|R) \propto \exp\{\sum_{t=0}^{T} x^T \theta\},$$

and MaxEnt-IRL proposes an algorithm to infer the θ that maximizes the posterior likelihood. This posterior inference procedure requires a dynamics model, which we can use a previously known model or the learned A_i from the previous section.

SWIRL applies MaxEnt-IRL to each segment of the task but with a small modification to learn quadratic rewards instead of linear ones. Let μ_i be the centroid of the next transition region. We want to learn a reward function of the form:

$$R_i(x) = -(x-\mu_i)^T Q(x-\mu_i).$$

for a positive semi-definite Q (negated since this is a negative quadratic cost). With some re-parametrization, this reward function can be written as:

$$R_i(x) = -\sum_{j=1}^{d}\sum_{l=1}^{d} q_{ij} x[j]x[l].$$

which is linear in the feature-space $y = x[j]x[l]$:

$$R_i(x) = \theta^T y.$$

In this form, the problem can be analytically solved with techniques proposed in [25]. SWIRL applies MaxEnt-IRL to the sub-sequences of demonstrations between 0 and ρ_1, and then from ρ_1 to ρ_2 and so on. The result is an estimated local reward function R_i modeled as a linear function of states that is associated with each ρ_i.

Model-free: Local Quadratic Rewards When a dynamics model is not available, SWIRL uses a model-free approach for reward construction similarly motivated by the quadratic rewards. The role of the reward function is to guide the robot to the next transition region ρ_i. A straight forward thing approach is for each segment i, we can define a reward function as follows:

$$R_i(x) = -\|x-\mu_i\|_2^2,$$

which is just the Euclidean distance to the centroid.

A problem with using Euclidean distance directly is that it uniformly penalizes disagreement with μ in all dimensions. During different stages of a task, some features will likely naturally vary more than others–this is learned through IRL. To account for this, we derive a reasonable Q that is independent of the dynamics:

$$Q[j,l] = \Sigma_x^{-1},$$

which is the inverse of the covariance matrix of all of the state vectors in the segment:

$$Q[j,l] = (\sum_{t=start}^{end} xx^T)^{-1}, \tag{1}$$

which is a $p \times p$ matrix defined as the covariance of all of the states in the segment $i-1$ to i. Intuitively, if a feature has low variance during this segment, deviation in that feature from the desired target it gets penalized. This is exactly the Mahalonabis distance to the next transition.

For example, suppose one of the features j measures the distance to a reference trajectory u_t. Further, suppose in step one of the task the demonstrator's actions are perfectly correlated with the trajectory ($Q_i[j,j]$ is low where variance is in the distance) and in step two the actions are uncorrelated with the reference trajectory ($Q_i[j,j]$ is high). Thus, Q will respectively penalize deviation from $\mu_i[j]$ more in step one than in step two.

Phase III. Policy Learning

In Phase III, SWIRL uses the learned transitions $[\rho_1,...,\rho_k]$ and $\mathbf{R}_{seq}$ as rewards for a Reinforcement Learning algorithm. In this section, we describe learning a policy π given rewards $\mathbf{R}_{seq}$ and an ordered sequence of transitions G.

However, this problem is not trivial since solving k independent problems neglects potential shared value structure between the local problems (e.g., a common failure state). Furthermore, simply taking the aggregate of the rewards can lead to inconsistencies since there is nothing enforcing the order of operations. The key insight is that a single policy can be learned jointly over all segments over a modified problem where the state-space with additional variables that keep track of the previously achieved segments. To do so, we require an MDP model that also captures the history of the process.

MDPs with Memory RL algorithms apply to problems that are specified as MDPs. The challenge is that some sequential tasks may not be MDPs. For example, attaining a reward at ρ_i depends on knowing that the reward at goal ρ_{i-1} was attained. In general, to model this dependence on the past requires MDPs whose state-space also includes history.

Given a finite-horizon MDP $\mathcal{M}$ as defined in Section 3, we can define an MDP $\mathcal{M}_H$ as follows. Let $\mathcal{H}$ denote set of all dynamically feasible sequences of length smaller than T comprised of the elements of S. Therefore, for an agent at any time t, there is a sequence of previously visited states $H_t \in \mathcal{H}$. The MDP $\mathcal{M}_H$ is defined as:

$$\mathcal{M}_H = \langle S \times \mathcal{H}, A, P'(\cdot,\cdot), R(\cdot,\cdot), T \rangle.$$

Algorithm 3: Policy Learning

Data: Transition States G, Reward Sequence $\mathbf{R}_{seq}$, exploration parameter ε

Initialize $Q(\binom{s}{v}, a)$ randomly
foreach $iter \in 0, ..., I$ **do**
    Draw s_0 from initial conditions
    Initialize v to be $[0, ..., 0]$
    Initialize j to be 1
    foreach $t \in 0, ..., T$ **do**
        Choose best action a based on Q or random action w.p ε.
        Observe Reward R_j
        Update state to s' and Q via Q-Learning update
        If s' is $\in \rho_j$ update $v[j] = 1$ and $j = j + 1$

Result: Policy π

For this MDP, P' not only defines the transitions from the current state $s \mapsto s'$, but also increments the history sequence $H_{t+1} = H_t \sqcup s$. Accordingly, the parametrized reward function R is defined over S, A, and H_{t+1}.

$\mathcal{M}_H$ allows us to address the sequentiality problem since the reward is a function of the state and the history sequence. However, without some parametrization of H_t, directly solving this MDPs with RL is impractical since it adds an overhead of $\mathcal{O}(e^T)$ states.

Policy Learning Using our sequential task definition, we know that the reward transitions (R_i to R_{i+1}) only depend on an arrival at the transition state ρ_i and not any other aspect of the history. Therefore, we can store a vector v, a k dimensional binary vector ($v \in \{0,1\}^k$) that indicates whether a transition state $i \in 0, ..., k$ has been reached. This vector can be efficiently incremented when the current state $s \in \rho_{i+1}$. Then, the additional complexity of representing the reward with history over $S \times \{0,1\}^k$ is only $\mathcal{O}(k)$ instead of exponential in the time horizon.

The result is an augmented state-space $\binom{s}{v}$ to account for previous progress. Over this state-space, we can apply Reinforcement Learning algorithms to iteratively converge to a successful policy for a new task instance. SWIRL applies Q-Learning with a Radial Basis Function value function representation to learn a policy π over this state-space and the reward sequence $\mathbf{R}_{seq}$. This is summarized in Algorithm 3.

5 Experiments

We evaluate SWIRL with a series of standard RL benchmarks and in a physical experiment on the da Vinci surgical robot.

5.1 Methodology

All of the experimental scenarios followed a similar pattern: (1) start with an RL problem with a delayed reward, (2) generate N demonstration trajectories with motion planning in simulated scenarios and kinethestic demonstration in the physical experiments, (3) apply SWIRL, (4) evaluate the performance of the policy as a function of the I iterations. For all convergence curves presented, we show the probability of task success as

a function of the number of RL iterations. For convergence rate, we measure the Area Under Curve of the learning curve (i.e., cumulative expected reward over the learning epoch).
The algorithms considered in the experiments are:

1. **Q-Learning:** This applies a Q-Learning algorithm with the same hyper-parameter setting as SWIRL.
2. **Pure MaxEnt-IRL:** Given N demonstrations this learns a reward using MaxEnt-IRL and no hierarchical structure. Then, it applies Q-Learning with the same hyper-parameter setting as SWIRL until convergence. (Only Phase II and III)
3. **SVM:** Given N demonstrations this learns a policy using a multi-class SVM classifier. There is no further learning after training. (Directly to Policy)
4. **SWIRL (Model-Based) and SWIRL (Model-Free)**

5.2 Parallel Parking

We constructed a parallel parking scenario for a robot with non-holonomic dynamics and two obstacles. The robot can control its speed ($\|\dot{x}\| + \|\dot{y}\|$) and heading ($\theta$), and observe its x position, y position, orientation, and speed in a global coordinate frame. If the robot parks between the obstacles, i.e., 0 velocity within a 15° tolerance, the task is a success and the robot receives a reward of 1. The robot's dynamics are noisy and with probability 0.1 will randomly add or subtract 5° degrees from the steering angle. If the robot collides with one of the obstacle or does not park in 200 timesteps the episode ends. The baseline approach is modeling the entire problem as an MDP with a quadratic reward function at the target state (where the robot parks). For comparison, we use this reward function for Q-Learning and infer a quadratic reward function using MaxEnt-IRL. We call this domain Parallel Parking with Full Observation (PP-FO) (see Figure 1).

Next, we made the Parallel Parking domain a little harder. We hid the velocity state from the robot, so the robot only sees (x, y, θ). As before, if the robot collides with one of the obstacle or does not park in 200 timesteps the episode ends. We call this domain Parallel Parking with Partial Observation (PP-PO).

We generated 5 demonstrations using an RRT motion planner (assuming deterministic dynamics) and applied SWIRL to learn the segments. Figure 1 illustrates the demonstrations and the learned segments. There are two intermediate goals corresponding to positioning the car and orienting the car correctly before reversing.

Performance In the fully observed problem, compared to MaxEnt-IRL, the model-based SWIRL converges to a policy with a 60% success rate with about 3x fewer time-steps. The gains for the model-free version are more modest with a 50% reduction. The supervised policy learning approach achieves a success rate of 47% and the baseline RL approach achieves a success rate of 36% after 250000 time-steps.

The baseline Q-Learning approach directly tries to learn a sequence of actions to minimize the quadratic cost around the target state. This leads to a lot of exploration since the robot must first make "negative" progress (pulling forward). SWIRL improves convergence since it structures the exploration through the segmentation. The local reward functions are better shaped to guide the car towards its short term goal. This focuses the exploration on solving the short term problem first. MaxEnt-IRL mitigates some of the problems since it rewards states based on their estimated cost-to-go,

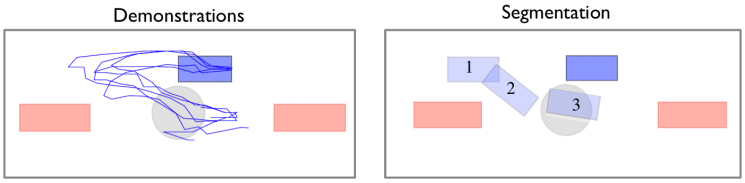

Fig. 1. This plot illustrates (left) the 5 demonstration trajectories for the parallel parking task, and (right) the sub-goals learned by SWIRL.

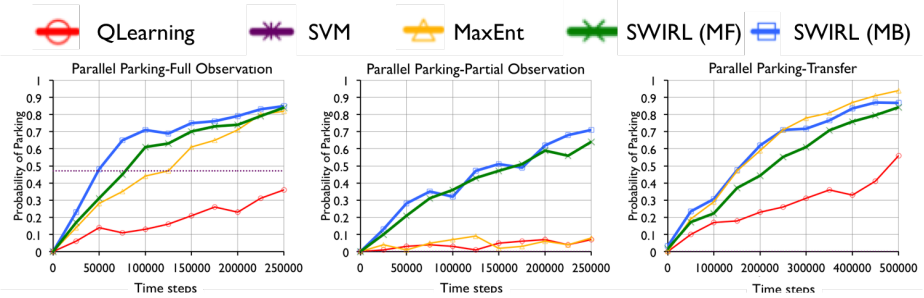

Fig. 2. Performance on a parallel parking task with noisy dynamics with full state observations (position, orientation, and velocity), partial observation (only position and orientation), and transfer (randomly permuting the action space). Success is measured in terms of the probability that the car successfully parked, and (M) denotes whether the approach used the dynamics model. In the fully observed case, both the model-based and model-free SWIRL algorithms converge faster than MaxEnt-IRL and quickly outperforms the SVM. In the partially observed case, MaxEnt-IRL, Q-Learning, and the SVM fail–while SWIRL succeeds. Both techniques also demonstrate comparable transferability to MaxEnt-IRL when the domain's dynamics are perturbed.

but as the time-horizon increases the estimates of this become nosier–leading to worse performance (see technical report for a characterization [26]).

In the partial observation problem (PP-PO), there is no longer a stationary policy that can achieve the reward. The techniques that model this problem with a single MDP all fail to converge. The learned segments in SWIRL help disambiguate dependence on history. After 250000 time-steps, the policy learned with model-based SWIRL has a 70% success rate in comparison to a <10% success rate for the baseline RL, MaxEnt-IRL, and 0% for the SVM.

Finally, we explore how well the constructed rewards transfer if the dynamics are perturbed in the fully observed setting. We expect MaxEnt-IRL to transfer well because it learns a delayed reward, which tends to encode success conditions and not task-specific details. After constructing the rewards, we randomly perturbed the system dynamics by introducing a bias towards turning left. We find that the model-based SWIRL technique transfers to this domain comparably to MaxEnt-IRL until the task is so different that the sub-goals learned with SWIRL are no longer informative. The model-free SWIRL algorithm converges more slowly; requiring 20% more time-steps to converge to the same success rate.

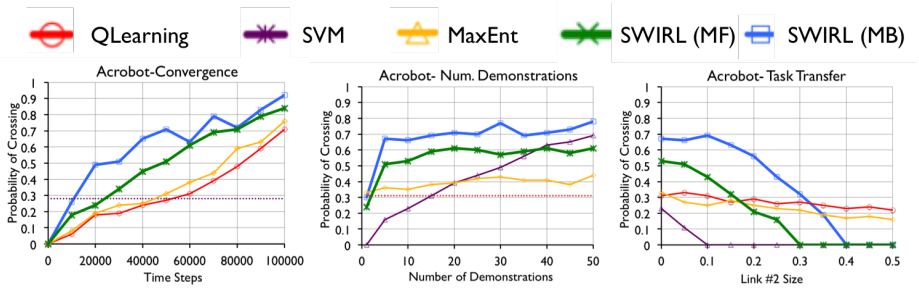

Fig. 3. Acrobot: We measured the performance of rewards constructed with SWIRL and the alternatives. We find that SWIRL (model-based and model-free) converges faster than MaxEnt-IRL, Q-Learning, and the SVM. Furthermore, SWIRL requires less demonstrations, which we measure by comparing the performance of the alternatives after a fixed 50000 time-steps and with varied input demonstrations. We also vary the task parameters by changing the size of the second link of the pendulum and find that the learned rewards are robust to this variation (as before comparing the performance of the alternatives after a fixed 50000 time steps). MaxEnt-IRL shows improved transfer performance since once the task has changed enough the segments learned during the demonstrations may not be informative and may even hurt performance if they are misleading.

5.3 Acrobot

This domain consists of a two-link pendulum with gravity and with torque controls on the joint. The dynamics are noisy and there are limits on the applied torque. The robot has 1000 timesteps to raise the arm above horizontal ($y = 1$ in the images). If the task is successful and the robot receives a reward of 1. Thus, the expected reward is equivalent to the probability that the current policy will successfully raise the arm above horizontal. We generated $N = 5$ demonstrations for the Acrobot task and applied segmentation. These demonstrations were generated by training the Q-Learning baseline to convergence and then sampling from the learned policy. In Figure 3, we plot the performance of the all of the approaches. We include a comparison between a Linear Multiclass SVM and a Kernelized Multiclass SVM for the policy learning alternative. In this example, we find that applying MaxEnt-IRL does not improve the convergence rate. For this state-space, MaxEnt-IRL merely recovers the reward used in the original RL problem. On the other hand, added segments using SWIRL improve convergence rates.

We also vary the number of input demonstrations to SWIRL and find that it requires fewer demonstrations than policy learning and MaxEnt-IRL to converge to a more reliable policy. It takes about 10x more demonstrations for the supervised learning approach to reach comparable reliability. Finally, we find that SWIRL does not sacrifice much transferability. We learn the rewards on the standard pendulum, and then during learning we vary the size of the second link in the pendulum. We plot the success rate (after a fixed 50000 steps) as a function of the increase link size. SWIRL is significantly more robust than supervised policy learning to the increase in link size and has a significantly higher success rate than IRL for small perturbations in link size.

5.4 Summary of Simulated Experiments

Table 1 summarizes the results of our experiments in terms of convergence rate and maximum attained reward on the Parallel Parking domain (with and without partial observation), Acrobot domain, and additional experiments using variants of GridWorld. GridWorld is a two-room map where the robot has to reach two target states in sequence to get the full reward. GridWorld-2 is a substantially harder map with "pits" (i.e., instant failure if reached). The Two-Bridges domain is another GridWorld based environment in which there is a short "unsafe" path between start and goal and a longer "safe" path (which is actually the optimal solution). Please refer to the arXiv report [26] for more details.

Table 1. This table summarizes the convergence rate (**AUC**) and max reward (**MAX**) attained by a Q-learning robot using the alternatives after a fixed number of iterations.

	GridWorld		GridWorld-2		Two-Bridges		PP(FO)		PP(PO)		Acrobot	
	Max	AUC	Max	AUC	Max	AUC	Max	AUC	Max	AUC	Max	AUC
Q-Learning	0.984	10.976	0.861	15.440	1.090	16.270	0.911	109.76	0.311	27.419	0.944	3.447
MaxEnt-IRL	0.987	299.556	0.861	16.956	0.759	16.270	0.950	299.556	0.444	33.128	0.920	44.111
SWIRL (MF)	1.830	322.125	1.764	14.070	**1.751**	**18.953**	**0.991**	164.127	0.934	123.115	0.906	20.935
SWIRL (MB)	**1.835**	**514.113**	**1.827**	**28.632**	1.577	17.141	0.965	**514.113**	**0.958**	**333.897**	**0.987**	**65.512**

5.5 Physical Experiments with the da Vinci Surgical Robot

In physical experiments, we apply SWIRL to learn to cut along a marked line in gauze similar to Murali et al. [27]. This is a multi-step problem where the robot starts from a random initial state, has to move to a position that allows it to start the cut, and then cut along the marked line. We provide the robot 5 kinesthetic demonstrations by positioning the end-effector and then following various marked straight lines. The state-space of the robot included the end-effector position (x, y) as well as a visual feature indicating its pixel distance to the marked line (pix). This visual feature is constructed using OpenCV thresholding for the black line. Since the gauze is planar, the robot's actions are unit steps in the $\pm x, \pm y$ axes. Figure 4 illustrates the training and test scenarios.

As expected, the algorithm identifies two consistent changes in local linearity corresponding to the positioning step and the termination. The learned reward function for the position step minimizes x, y, pix distance to the starting point and for the cutting step the reward function is more heavily weighted to minimize the pix distance. We defined task success as positioning within 1 cm of the starting position of the line and during the following stage, missing the line by no more than 1 cm (estimated from pixel distance). Since we did not have a dynamics model, we evaluated the model-free version of SWIRL, Q-Learning, and the SVM. SWIRL was the only technique able to achieve the combined task. This is because the policy for this task is non-stationary, and SWIRL is the only approach of the alternatives that can learn such a policy.

We evaluated the learned tracking policy to cut gauze. We ran trials on different sequences of curves and straight lines. Out of the 15 trials, 11 were successful. 2 failed due to SWIRL errors (tracking or position was imprecise) and 2 failed due to cutting errors (gauze deformed causing the task to fail). 1 of the failures was on the 4.5 cm curvature line and 3 were one the 3.5 cm curvature line.

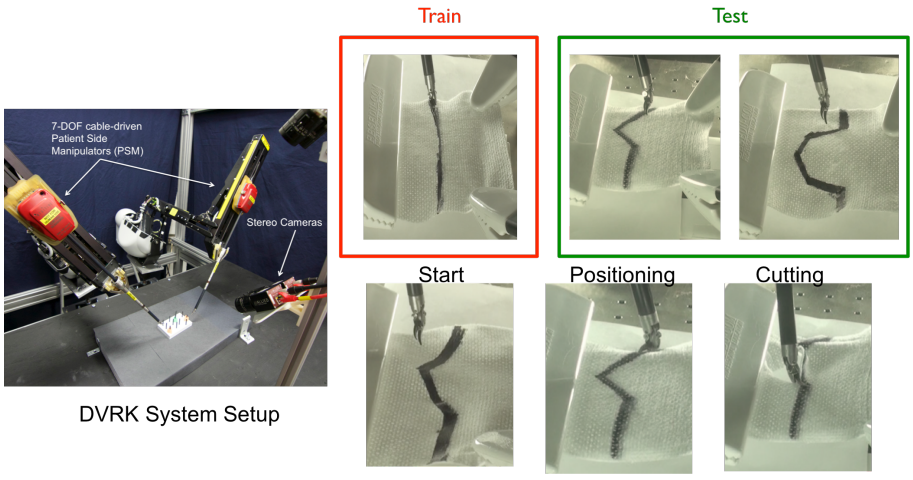

Fig. 4. We collected demonstrations on the da Vinci surgical robot kinesthetically. The task was to cut a marked line on gauze. We demonstrated the location of the line without actually cutting it. The goal is to infer that the demonstrator's reward function has two steps: position at a start position before the line, and then following the line. We applied this same reward to curved lines that started in different positions.

Table 2. With 5 kinesthetic demonstrations of following marked straight lines on gauze, we applied SWIRL to learn to follow lines of various curvature. After 25 episodes of exploration, we evaluated the policies on the ability to position in the correct cutting location and track the line. We compare to SVM on each individual segment. SVM is comparably accurate on the straight line (training set) but does not generalize well to the curved lines.

Curvature Radius (cm)	SVM Pos. Error (cm)	SVM Tracking Error (cm)	SWIRL Pos. Error (cm)	SWIRL Tracking Error (cm)
straight	0.46	0.23	0.42	0.21
4.0	0.43	0.59	0.45	0.33
3.5	0.51	**1.21**	0.56	0.38
3.0	0.86	**3.03**	0.66	0.57
2.5	**1.43**	-	0.74	0.87
2.0	-	-	0.87	**1.45**
1.5	-	-	**1.12**	**2.44**

Next, we characterized the repeatability of the learned policy. We applied SWIRL to lines of various curvature spanning from straight lines to a curvature radius of 1.5 cm. Table 2 summarizes the results on lines of various curvature. While the SVM approach did not work on the combined task, we evaluated its accuracy on each individual step to illustrate the benefits of SWIRL. On following straight lines, SVM was comparable to SWIRL in terms of accuracy. However, as the lines become increasingly curved, SWIRL generalizes more robustly than the SVM.

6 Discussion and Future Work

SWIRL is a three-phase algorithm that first segments a task, learns local rewards, and then learns a policy. Experimental results suggest that sequential segmentation can indeed improve convergence in RL problems with delayed rewards. Results suggest that SWIRL is robust to perturbations in initial conditions, the environment, and sensing

noise. This paper formalizes the interaction and composability of the three phases (sequence, reward, and policy learning). In future work, we will explore extensions to each of the phases and quantify the degree of generalization. We will explore how the Q-Learning step could be replaced with Guided Policy Search, Policy Gradients, and optimal control. We will modify the segmentation algorithm to incorporate more complex transition conditions and allow for sub-optimal demonstrations. We will explore more robotic tasks including suturing, surgical knot tying, and assembly. Another avenue for future work is modeling complex tasks as hierarchies of MDPs.

Acknowledgements: This research was performed at the AUTOLAB at UC Berkeley in affiliation with the AMP Lab, BAIR, and the CITRIS "People and Robots" (CPAR) Initiative in affiliation with UC Berkeley's Center for Automation and Learning for Medical Robotics (CalMR). The authors were supported in part by the U.S. National Science Foundation under NRI Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems, and by Google, UC Berkeley's Algorithms, Machines, and People Lab, Knut & Alice Wallenberg Foundation, and by a major equipment grant from Intuitive Surgical and by generous donations from Andy Chou and Susan and Deepak Lim. We thank our colleagues and the anonymous WAFR reviewers who provided valuable feedback and suggestions, in particular, Pieter Abbeel, Anca Dragan, and Roy Fox.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A Survey of Robot Learning from Demonstration. Robotics and Autonomous Systems **57**(5) (2009) 469–483
2. Kolter, J.Z., Abbeel, P., Ng, A.Y.: Hierarchical apprenticeship learning with application to quadruped locomotion. In: NIPS. (2007) 769–776
3. Coates, A., Abbeel, P., Ng, A.Y.: Learning for control from multiple demonstrations. In: ICML, ACM (2008)
4. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: ICML, ACM (2004) 1
5. Ng, A.Y., Russell, S.J., et al.: Algorithms for inverse reinforcement learning. In: Icml. (2000) 663–670
6. Krishnan*, S., Garg*, A., Patil, S., Lea, C., Hager, G., Abbeel, P., Goldberg, K., (*denotes equal contribution): Transition State Clustering: Unsupervised Surgical Trajectory Segmentation For Robot Learning. In: International Symposium of Robotics Research, Springer STAR (2015)
7. Murali*, A., Garg*, A., Krishnan*, S., Pokorny, F.T., Abbeel, P., Darrell, T., Goldberg, K., (*denotes equal contribution): TSC-DL: Unsupervised Trajectory Segmentation of Multi-Modal Surgical Demonstrations with Deep Learning. In: IEEE Int. Conf. on Robotics and Automation (ICRA). (2016)
8. Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: Theory and application to reward shaping. In: ICML. (1999) 278–287
9. Judah, K., Fern, A.P., Tadepalli, P., Goetschalckx, R.: Imitation learning with demonstrations and shaping rewards. In: AAAI. (2014) 1890–1896
10. Ijspeert, A., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Neural Information Processing Systems (NIPS). (2002) 1523–1530
11. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: IEEE ICRA. (2009)
12. Manschitz, S., Kober, J., Gienger, M., Peters, J.: Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. Robotics and Autonomous Systems (2015)

13. Niekum, S., Osentoski, S., Konidaris, G., Barto, A.: Learning and generalization of complex tasks from unstructured demonstrations. In: Int. Conf. on Intelligent Robots and Systems (IROS), IEEE (2012)
14. Calinon, S.: Skills learning in robots by interaction with users and environment. In: IEEE Int. Conf. on Ubiquitous Robots and Ambient Intelligence (URAI). (2014)
15. Konidaris, G., Kuindersma, S., Grupen, R., Barto, A.: Robot Learning from Demonstration by Constructing Skill Trees. Int. Journal of Robotics Research **31**(3) (2011) 360–375
16. Ranchod, P., Rosman, B., Konidaris, G.: Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), IEEE (2015)
17. Dietterich, T.G.: Hierarchical reinforcement learning with the maxq value function decomposition. J. Artif. Intell. Res.(JAIR) **13** (2000) 227–303
18. Fox, E.B., Sudderth, E.B., Jordan, M.I., Willsky, A.S.: Nonparametric bayesian learning of switching linear dynamical systems. In: NIPS. (2009) 457–464
19. Moldovan, T., Levine, S., Jordan, M., Abbeel, P.: Optimism-driven exploration for nonlinear systems. In: Int. Conf. on Robotics and Automation (ICRA). (2015)
20. Khansari-Zadeh, S.M., Billard, A.: Learning stable nonlinear dynamical systems with gaussian mixture models. Robotics, IEEE Transactions on **27**(5) (2011) 943–957
21. Kruger, V., Herzog, D., Baby, S., Ude, A., Kragic, D.: Learning actions from observations. Robotics & Automation Magazine, IEEE **17**(2) (2010) 30–43
22. Kulis, B., Jordan, M.I.: Revisiting k-means: New algorithms via bayesian nonparametrics. arXiv preprint arXiv:1111.0352 (2011)
23. Mika, S., Schölkopf, B., Smola, A.J., Müller, K., Scholz, M., Rätsch, G.: Kernel PCA and de-noising in feature spaces. In: NIPS. (1998) 536–542
24. Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: AAAI. (2008)
25. Ziebart, B., Dey, A., Bagnell, J.A.: Probabilistic pointing target prediction via inverse optimal control. In: UIST, ACM (2012) 1–10
26. Krishnan, S., Garg, A., Liaw, R., Miller, L., Pokorny, F.T., Goldberg, K.: Hirl: Hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards. arXiv preprint arXiv:1604.06508 (2016)
27. Murali*, A., Sen*, S., Kehoe, B., Garg, A., McFarland, S., Patil, S., Boyd, W., Lim, S., Abbeel, P., Goldberg, K., (*denotes equal contribution): Learning by Observation for Surgical Subtasks: Multilateral Cutting of 3D Viscoelastic and 2D Orthotropic Tissue Phantoms. In: IEEE Int. Conf. on Robotics and Automation (ICRA). (2015)