Towards an Autonomous Human Inclusive Robot Learning Framework

A THESIS

submitted by

MANIMARAN S S

for the award of the degree

of

MASTER OF SCIENCE

(by Research)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

AUGUST 2015

THESIS CERTIFICATE

This is to certify that the thesis titled **Towards an Autonomous Human Inclusive Robot Learning Framework**, submitted by **Manimaran S S**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Science (by Research)**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. B. Ravindran
Research Guide
Associate Professor
Dept. of CSE
IIT-Madras, 600 036

Place: Chennai

Date: 27th April 2015

ACKNOWLEDGEMENTS

I express my sincere thanks to my advisor Dr. B. Ravindran, without whose support this work would not have been brought forth. His untiring support and guidance was present all throughout this work. He was always approachable, ready to clear my doubts and taught me a lot regarding research and life. I am grateful for his patience during difficult times and guidance ensuring I was on the right path always. I thank my colleagues from the RISE lab at IIT, Madras who were ideal intellectual companions on my journey. I wish to especially acknowledge Pradyot K.V.N. and Adwiteey Chrungoo for their contributions. I also thank CAIR, DRDO, Bengaluru, for funding and supporting my research work.

ABSTRACT

KEYWORDS: Robot Learning ; Path Planning; Human Activity Recognition; Reinforcement Learning.

Building intelligent robots has been one of the motivating problems in Artificial Intelligence research. Choosing what actions to perform is a fundamental problem every robot must solve. The aptness of the actions depends on the robot's understanding of its environment and the dynamics involved. This understanding is an ever-changing model based on the robot's observations and experiences. In this thesis, we propose a robot-learning architecture and discuss two key modules - a continuous space path planning algorithm and an activity recognition framework.

Path planning in continuous spaces is a challenging problem for which a variety of solutions have been proposed. The performance of sampling based path planning techniques relies on identifying a good approximation to the cost-to-go distance metric, in the case of systems with complex dynamics. We propose a technique that uses reinforcement learning to learn this distance metric on the fly from samples and combine it with existing sampling based planners to produce near optimal solutions. The resulting algorithm - Policy Iteration on Rapdily exploring Random Trees (RRTPI) can solve problems with complex dynamics in a sample-efficient manner while preserving asymptotic guarantees. We provide experimental evaluation of this technique on domains with under-actuated and underpowered dynamics such as the mountain car domain, the acrobot domain and a variant of the octopus arm domain.

Human activity recognition is a key component in enabling social robots to naturally interact with humans. While humans can easily distinguish whether an activity corresponds to a form of communication or an act of daily living, robots are yet not that insightful. This thesis addresses the problem of enabling a social robot to understand an activity as well

as classify it as an action corresponding to a human instruction or an act of daily living. We address this problem by presenting a simple yet effective approach of modeling pose trajectories in terms of directions taken by skeletal joints over the duration of an activity. Specifically, we propose a novel activity descriptor: Histogram of direction vectors (HODV). The descriptor is computationally efficient, scale and speed invariant and provides state of the art classification accuracies using off the shelf classification algorithms on multiple datasets such as the Cornell Activity Dataset (CAD-60), UT-Kinect Dataset as well as on our novel RGBD dataset.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RL	Reinforcement Learning
MDP	Markov Decision Problem
RRT	Rapidly-exploring Random Trees
FQI	Fitted-Q Iteration
HAR	Human Activity Recognition
HODV	Histogram of Direction Vectors
kNN	k-Nearest Neighbors
RRTPI	Policy Iteration on Rapidly-exploring Random Trees
RRST	Rapidly-exploring Random Sample Trees
NN-TD	Nearest Neighbor Temporal Difference
kNN-RRTPI	k Nearest Neighbor Policy Iteration on Rapidly-exploring Random Trees
LL-RRTPI	Locally Linear Policy Iteration on Rapidly-exploring Random Trees
LW-RRTPI	Locally Weighted Policy Iteration on Rapidly-exploring Random Trees
LQR	Linear Quadratic Regulator
HMM	Hidden Markov Models
MEMM	Maximum Entropy Markov Model
SVM	Support Vector Machine
RGBD	Red Green Blue Depth
CAD60	Cornell Activity Dataset
TF-IDF	Term Frequency - Inverse Document Frequency
LOOCV	Leave One sequence Out Cross Validation

NOTATION

S	State space
A	Action Space
T	State Transition Matrix
π	Policy
π^*	Optimal policy
J^π	Value function of policy π
$\hat{J}^\pi$	Estimate of value function of policy π
γ	Discount factor
α	Learning rate
$\mathscr{M}$	Markov Decision Problem
V	Vertex set of RRT
E	Edges of RRT
β	Vector of parameters
P_j	Cartesian coordinates of skeleton joint j
H	Histogram of Direction Vector
j^i_f	Vector of coordinates of joint i at time frame f
d^i_f	Difference vector of coordinates of joint i at time frame f
σ_q	Coordinate of grid index q
τ	Time step
Q^f_q	Vector directions of frame f at index q
h_i	Cumulative direct vector of joint i

CHAPTER 1

INTRODUCTION

Building intelligent robots has been one of the motivating problems for research in AI [Nilsson, 1984]. With advances in computational and physical capabilities, modern day robots can perform a variety of tasks. These tasks maybe situated across several environments, involving interaction with humans such as hospitals, offices and homes. These environments are highly cluttered and change over time. Simply pre-programming a robot is infeasible given the nature of the tasks and environments. Thus, a useful domestic robot must possess the ability to constantly evolve and *learn* to perform new tasks throughout its lifetime. We can apply machine learning techniques for solving various aspects of these problems.

1.1 Robot Learning

Mahadevan [1996] characterizes robot learning as assigning credit to actions in three dimensions- spatial, temporal and task-level, i.e., the robot must learn *when* to perform an action, *where* (as observed by sensors) to perform an action and in *what context* (task currently being solved). In order to do this, the robot must be able to learn and generalize observations across all three dimensions. Based on the nature of input available learning paradigms such as reinforcement learning and inductive learning can be used solve relevant parts of this problem. In this thesis, we use reinforcement learning techniques combined with planning and supervised learning to tackle the robot learning problem.

1.2 Challenges and Motivation

The challenges for a robot learning system can be studied at various levels of the robot's operation. Planning motor control can be challenging as the robot may possess complex

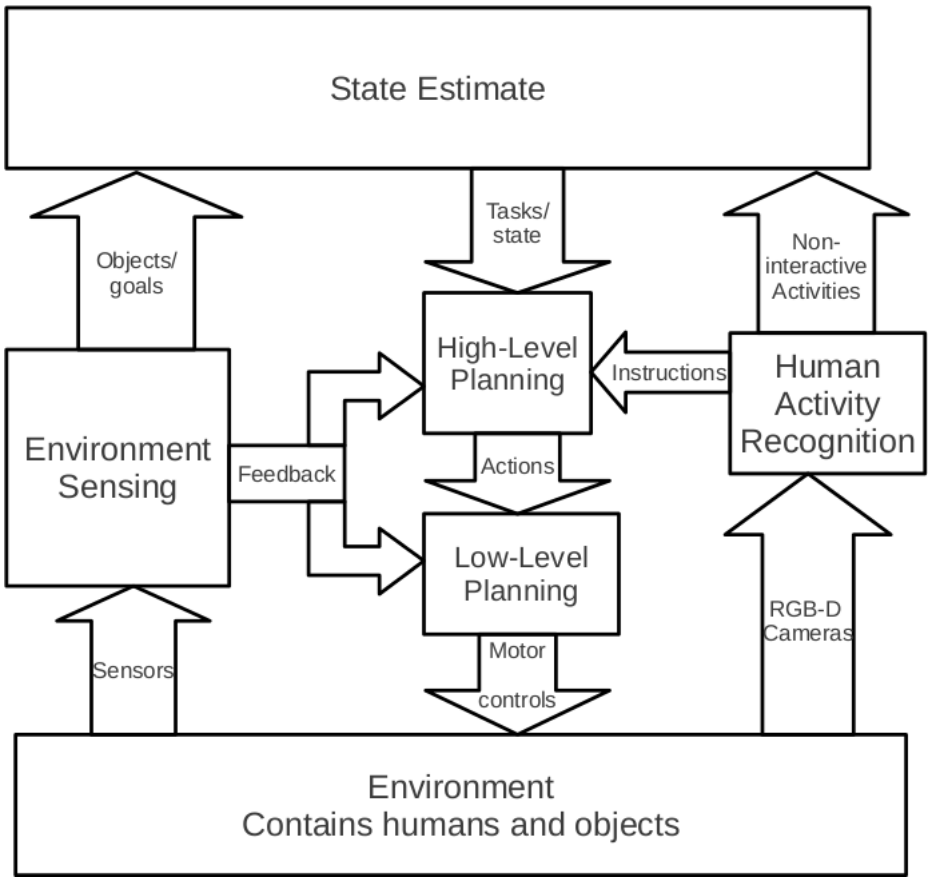

Figure 1.1: Human inclusive robot learning architecture

actuators. It may also pick up new tools to perform certain tasks which result in its dynamics becoming highly non-linear, under-actuated or underpowered. Task level planning for activities like cooking and cleaning, comprising of several basic actions such as *move* and *pickup* is also challenging, as the robot has to adapt its plan according to various environmental factors and human intervention. The robot must also learn to reuse knowledge acquired from solving one task to solve others.

An effective robot learning paradigm must allow the robot to learn autonomously through trial and error. The robot must be able to constantly improve its solutions by interacting with the environment. This active approach is central to Reinforcement Learning (RL). RL is a popular approach for robot learning [Kober *et al.*, 2013; Deisenroth *et al.*, 2013; Nguyen-Tuong and Peters, 2011]. But, a robot left to explore completely on its own will need a long time to learn to perform a new task. Learning would be faster if the robot incorporates knowledge from humans around it. Learning through interactions and observing humans is also an essential part of a domestic robot's learning methodology. In

order to facilitate such interaction, the ability to communicate with humans through natural language and gesturing is needed. There have been several approaches to robot learning with humans in the past [Argall *et al.*, 2009; Knox and Stone, 2009]. However, the robot cannot always rely on humans to give accurate inputs and must continue to improve itself autonomously.

As we can see, an effective robot learning technique should be an amalgamation of multiple machine learning paradigms. We present a robot learning framework or architecture as shown in Figure 1.1.

This framework consists of several modules interacting with each other. There are two levels of planners - a high level planner and a low level planner. The high level planner forms more abstract plans using some basic actions provided by the low level planner. Motor level controls for these basic actions, such as *pickup*, *drop* and *move-to*, are generated by the low level planner. A sensing module collects inputs from several sensors on the robot to provide a variety of information. This includes an estimate the current state of the world, such as the tasks the robot has to perform, the configuration of various objects in the world and feedback for the planners. A Human Activity Recognizer (HAR) gives the robot the ability to understand human activities. This can be used in several ways - for aiding the robot in planning, for providing cues as to when a human needs assistance or to simply improve the robot's understanding of the working of the world. All these modules have to be adaptive must work with an environment that is stochastic and partially observable. For instance, the low-level planner must work in cases where the dynamics become complex and the HAR should work across all humans with variations in performing activities.

1.3 Contribution of thesis

Developing the entire robot learning framework is beyond the scope of this thesis. The contributions of this thesis are two fold and represented by dashed lines in the above figure.

1. We present an algorithm - Policy Iteration on Rapidly-exploring Random Trees (RRTPI) that can solve control problems with complex under-actuated and under-

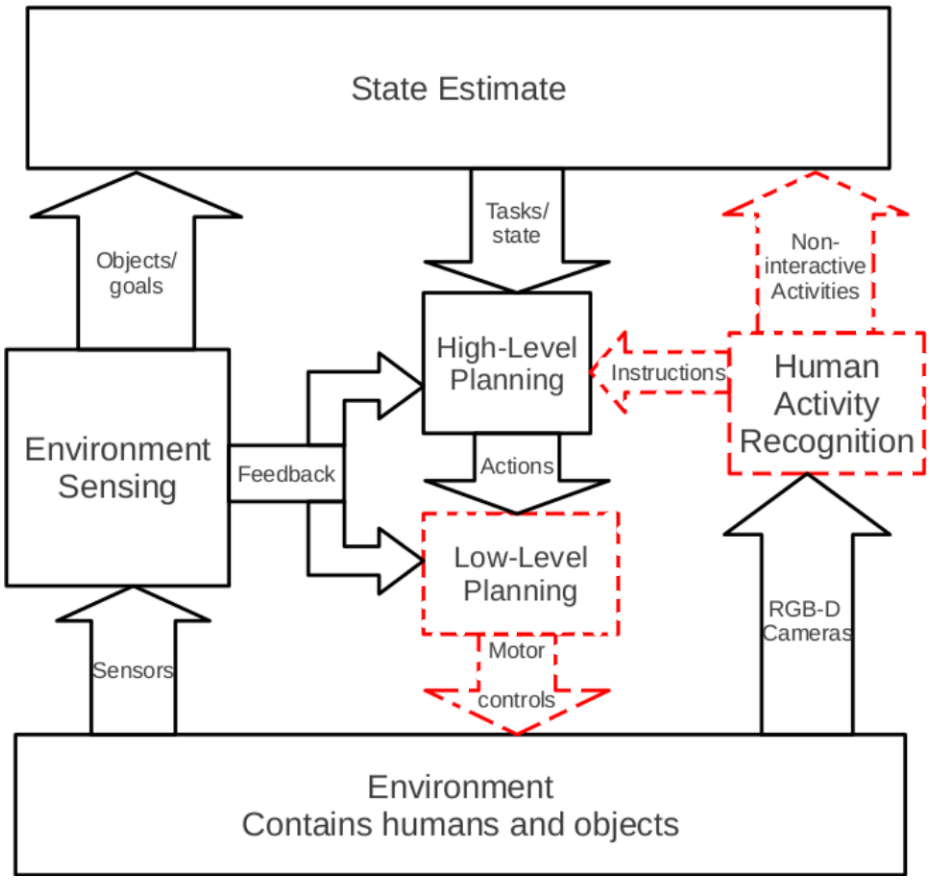

Figure 1.2: Modules presented in thesis.

powered dynamics. This algorithm automatically learns domain dependent metrics which it uses with sampling based planning to produce asymptotically optimal solutions. This is used in the low-level planner to for path planning.

2. We present a computationally efficient technique for human activity recognition - Histogram of Direction Vectors. This algorithm gives high accuracy across different kinds of activities and is person independent. This can be used to recognize human instructions for learning, as well as, observing human activities when providing assistance.

CHAPTER 2

SAMPLING BASED PLANNERS - BACKGROUND

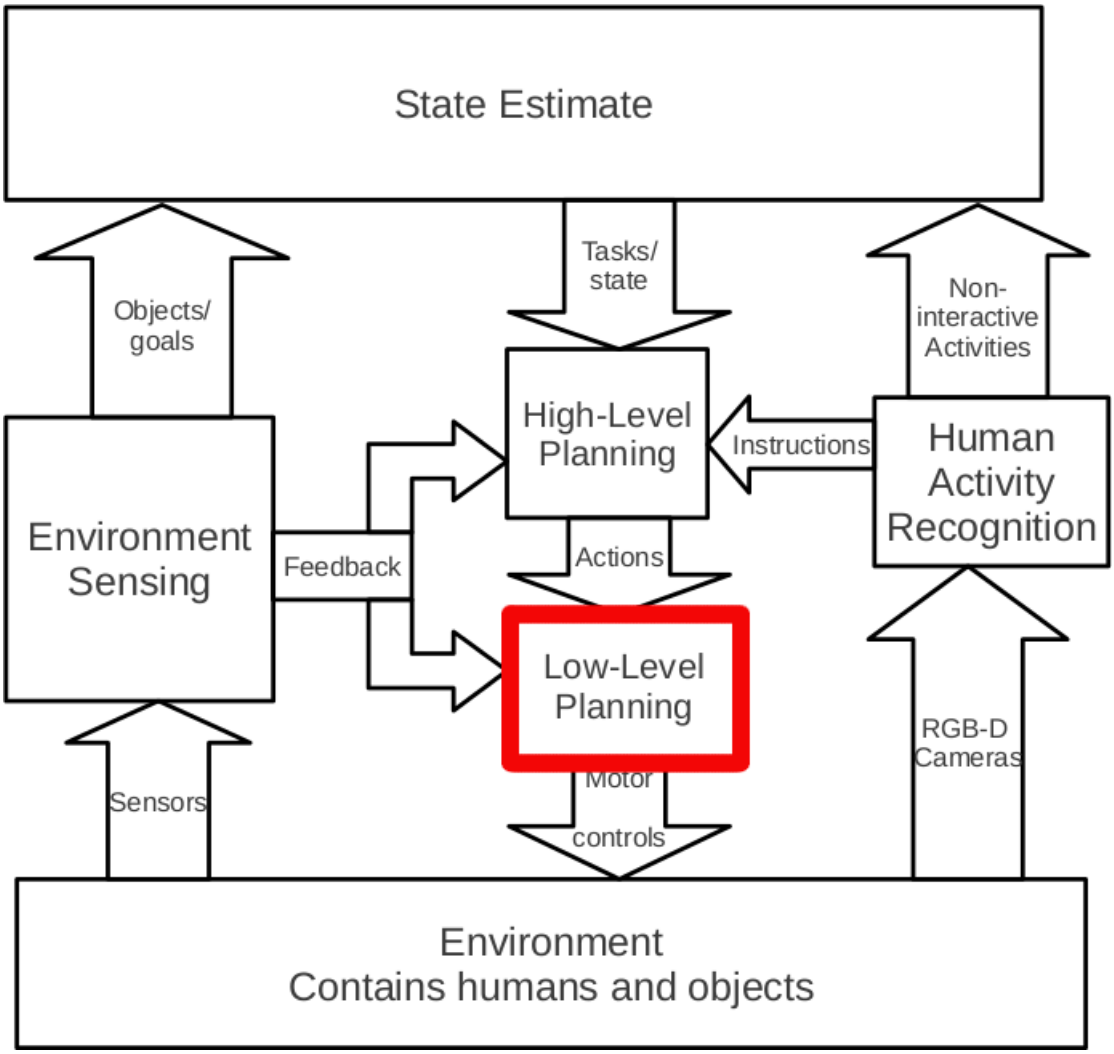

Figure 2.1: Low level planning module in overall architecture.

The problem of finding feasible trajectories from a given starting configuration to a goal in dynamical systems is a central problem in robotics. This problem is known to be at least PSPACE-hard. One approach to solve this problem is to directly use numeric solutions of the Hamilton-Jacobi-Bellman equations. Other approaches formulate the problem as a discrete Markov Decision Problem (MDP). However these methods suffer from the curse of dimensionality.

A popular class of algorithms that are resilient to this issue are sampling based algorithms. These algorithms are reasonably fast and efficient in terms of space [Lavalle, 2006]. Rapidly exploring Random Trees (RRTs) are one such method that are widely used [LaValle and Kuffner, 2001]. RRTs have good space filling properties and possess *asymptotic completeness*, i.e., they eventually find a solution if one exists. However, they do not provide any guarantees regarding optimality. In fact, it was shown that they almost always converge to a sub-optimal solution [Karaman and Frazzoli, 2011].

2.1 Related work

Recently, RRT* an extended version of the RRT method has been developed that also guarantees *asymptotic optimality*, i.e., they eventually converge to an optimal solution as more samples are drawn [Karaman and Frazzoli, 2011]. However, these guarantees are only asymptotic and the performance of RRT based algorithms is highly dependent on the distance measure used. Even solving a simple two state variable control problem with underpowered dynamics can require an inordinate amount of samples due to poor exploration. The reason being, in complex systems that are underactuated or underpowered, the Euclidean distance between two points is not a good estimate of the geodesic distance or the Carnot-Caratheory metric on the sub-Riemannian manifold induced by the system dynamics. It has been shown that RRTs explore space efficiently only when the distance metric reflects the true cost-to-go [Cheng and Lavalle, 2001].

Thus, recent research has concentrated on identifying the correct domain-dependent metric for efficient exploration. Glassman and Tedrake [2010] linearize the system dynamics and use affine quadratic regulators to derive this metric. They show that this results in improved exploration. Perez *et al.* [2012] integrated this technique into the RRT* algorithm to improve exploration while aiming to obtain optimal solutions. They used linear quadratic regulation (LQR) to determine the cost-to-go function as well the tree extension procedure. These methods assume that the dynamics are linearizable and available in closed form. If the robot experiences failure of some joints or picks up new tools, then these dynamics may even change over time. The dynamics can also be discontinuous or

too complex to represent in closed form as in the case of an octopus arm [Engel *et al.*, 2006]. Thus in situations where we cannot define exact closed form dynamics, we need to *learn* the domain dependent metric from experience. This will extend the scope of existing RRT based planners to a wide range of domains that have complex dynamics. Techniques for doing this have been explored in Gammell *et al.*. Here the states are sampled from an ellipsoidal informed subset to direct the RRT towards better quality solutions. Adaptive sampling has also be employed based on a weighting function of the features of the workspace to improve quality of the solutions found by RRTs [Zucker *et al.*, 2008].

One field that has traditionally looked at learning domain dependent metrics from sample data is Reinforcement Learning (RL). Most RL techniques work by estimating the value function, which is an sample based estimate of the cost-to-go, of states from sample trajectories through the state space [Sutton and Barto, 1998]. While these techniques have been well studied for discrete domains, continuous domains are more challenging as generalization and approximation is required to learn the value function. Obtaining a correct estimate depends directly on the quality of the samples. A crucial problem that limits the applicability reinforcement learning methods to continuous domains, is ensuring generation of sufficiently representative samples.

In this work we propose RRTPI a hybrid approach that combines RL with RRT style sampling. RL techniques estimate the cost-to-go of states with minimum domain knowledge, but they require a principled way of generating sufficiently representative sample trajectories in continuous space. On the other hand, RRT based algorithms posses good exploration properties in continuous space, but require an estimate of the cost-to-go in order to work in underactuated and underpowered domains. Combining these approaches enable RRTPI to handle arbitrarily complex domains without making assumptions on the form of the system dynamics and costs. To the best of our knowledge this is the first such hybrid approach combining these two paradigms.

2.2 Reinforcement Learning

The problem of finding an optimal path is modeled as finding optimal solutions to a Markov Decision Process(MDP). A continuous MDP $\mathscr{M}$ is described as $\langle S, A, T, R \rangle$. $S \in \mathbb{R}^n$ is the domain of states. Each state has a corresponding set of allowed actions A_s. This set maybe discrete or continuous. The action set A is defined as $\bigcup_s A_s$. $T(s,a,s')$ is the probability of transitioning from state $s \in S$ on taking action a to a state s'. If the transitions are deterministic we may write it as $T(s,a) = s'$. $R(s,a,s')$ is the expectation of real valued rewards(or cost) associated with taking action, a from state $s \in S$, and reaching state s'. These rewards are generally bounded in $[R_{min}, R_{max}]$. Given some starting state s_0, we pick an action a_0 resulting in a state transition to some s_1 according to T and a reward r_1 sampled from R. After a fixed time step, we pick the next action and so on so forth, resulting in a sequence of states actions and rewards $s_0, a_0, r_1, s_1, a_1, r_2, \ldots$. Our objective is to choose actions $a_0, a_1, \ldots$ such that we obtain maximum cumulative reward or *return*. The return is defined as $\sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward at time step t and $\gamma \in (0,1)$ is the discount factor.

A policy π is a mapping from the state space S to the action space A. $\pi : S \times A_s \rightarrow (0,1)$ that is, it represents the probability of choosing some action at a given state s. A deterministic policy is usually denoted as $\pi(s) = a$ where $a \in A_s$. The state value function of a state $J^\pi(s)$ is the expected return obtained by following policy π starting from the state s. When the set S is discrete, we may express the value function for a given policy using the Bellman equations as,

$$\forall s \in S, \;\; J^\pi(s) = \sum_a \pi(s,a) \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma J^\pi(s')] \tag{2.1}$$

A policy π^* is optimal if $\forall s, \pi \;\; J^{\pi^*}(s) \geq J^\pi(s)$. The optimal value function also denoted as, J^* can be calculated by replacing the expectation over set of actions in equation 2.1 with the max operator. When the state space is continuous, we use parametric or non-paramentric function approximation techniques to represent J.

2.2.1 Policy Evaluation using TD(λ)

Equations 2.1 can be solved using dynamic programming techniques Sutton and Barto [1998]. In most real world scenarios however, the transition probabilities are not available directly or are too complicated to be represented explicitly. A popular class of RL algorithms solve this problem by *sampling*, and estimate the value function. This is known as policy evaluation.

The TD(λ) family of algorithms is a basic method that uses temporal differences to estimate J^{π} [Sutton, 1988]. Consider a set of N trajectories of the form

$$\{s_0, a_0, r_1, s_1, a_1, r_2, \ldots s_{M_i}\}_{i=1}^{N}$$

For $\lambda = 0$ the TD(0) algorithm estimates the state value function $\hat{J}$ using the following update

$$\hat{J}(s_t) \leftarrow (1-\alpha)\hat{J}(s_t) + \alpha(r_{t+1} + \gamma\hat{J}(s_{t+1})) \tag{2.2}$$

$0 \leq i \leq N, 0 \leq t \leq M_i$ where α is the learning rate. The parameter λ is a measure of how much credit is assigned to earlier states in the trajectory. It serves to trade off bias and variance in the estimates with $\lambda = 0$ having least variance and most bias [Sutton, 1988]. TD(λ) converges provably with appropriately decaying values of α.

Methods that handle continuous states usually assume a functional form of J^{π} to estimate the value [Boyan, 2002; Ernst *et al.*, 2005]. We use TD(λ) on the discrete MDP approximation of the continuous problem, as given by a RRT based sampling technique, to get point estimates of J. We then generalize the point estimates to unseen states, by using nearest neighbor methods. Details on the sampling technique and reasoning behind our choice of the policy estimation technique are given in next chapter. We will first describe our sampling technique in the following section.

2.3 Rapidly-exploring Random Trees

In this section we briefly introduce RRTs and describe how we use them to generate a 'sample tree' of trajectories to approximate a given problem. Given a space S, the basic RRT construction is as described in Algorithm 1. The algorithm randomly samples a point in the space and calculates the nearest node in the existing tree from the sampled point. A new point is added to the tree by moving a fixed distance in the direction of the sampled state. If the resulting edge from connecting the nearest point to the new point is collision free then the vertex and edge are added to the tree. This is known as extending the tree. Thus the tree is stretched outwards towards lesser explored areas. This requires a distance measure for calculating the nearest vertex and extending the tree. RRTs possess several attractive properties in terms of exploration. Given a RRT G of size n with set of vertices $V(G)$ and edges $E(G)$ constructed in some space S, then $\forall s \in S \lim_{n\to\infty} Pr(s \in V(G)) \to 1$, Also the probability that a node in the tree will be expanded is proportional to the volume of its Voronoi region. This accounts for the *rapidly-exploring* property of RRTs. We will preserve this property while generating samples.

Algorithm 1: ConstructRRT(N)

Edges $E(G) \leftarrow \emptyset$ and *Vertices* $V(G) \leftarrow \emptyset$;
$n \leftarrow 0$;
while $n < N$ **do**
    Sample a state s_{new} from S;
    $s_{near} \leftarrow \texttt{Nearest}(s_{new}, V(G))$;
    $\{a_{ext}, s_{ext}\} \leftarrow \texttt{Extend}(s_{near}, s_{new})$;
    $V(G) \leftarrow V(G) \cup \{s_{ext}\}$;
    $E(G) \leftarrow E(G) \cup \{(s_{near}, s_{ext})\}$;
    $n \leftarrow n + 1$;
end
return G;

RRTs do not work well in domains with under-actuated and underpowered dynamics. The reason being the Euclidean distance does not accurately measure the true cost to go. Hence we need a technique through which we can estimate this domain dependent metric.

Figure 2.2 shows a regular RRT grown using Euclidean distance in a domain with non-linear dynamics. As seen in the zoomed in section, the red and green states are near

in terms of Euclidean distance but to get to the red state from green, several actions are needed.

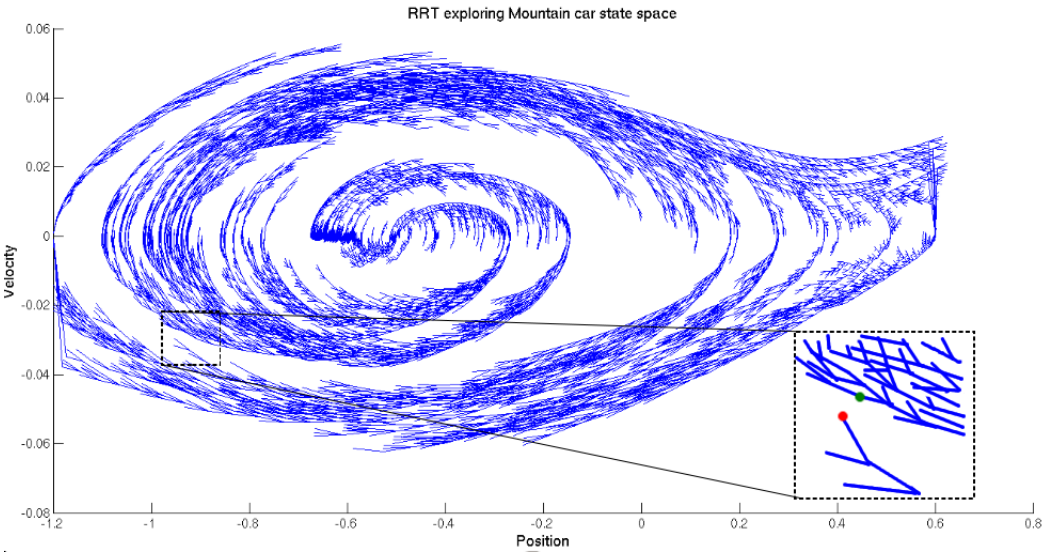

Figure 2.2: Need for domain dependent metrics

Thus as summarized above, traditional RL techniques suffer from lack of exploration while RRT techniques are able to explore well but cannot factor in domain dependent metrics to find optimal solutions. Thus we combine the two almost complementary approaches and present a technique - RRTPI capable of finding optimal paths in non-linear domains.

CHAPTER 3

POLICY ITERATION ON RAPIDLY-EXPLORING RANDOM TREES (RRTPI)

The overall idea in RRTPI is to use estimates of the value function and RRT samples iteratively improving the quality of estimate of the value function as well as the optimality of the trajectories.

3.1 Sampling using RRTs

Given a control problem formulated as an MDP $\mathscr{M}$, we would like to generate sample trajectories using a RRT-like procedure. We assume access to a generative model $\widehat{\mathscr{M}}$ as described by Ng and Jordan [2000]. Given a state s and action a, the model returns a sample from distribution of the next state and a sample reward corresponding to a fixed time step. In our algorithm, we generate samples based on a metric defined by the value function. Thus the `Nearest` and `Extend` functions are defined as shown below based on the metric $\|\cdot\|_J$.

Function 1 Nearest(s,X,$\|\cdot\|_J$)

Data: Set of states X, distance measure given by $\|\cdot\|_J$ and a state s
Result: Return $x_{near} \in X$ such that, $x_{near} = \arg\max_{x \in X} \|x - s\|_J$

Given a real valued function J (the value function in our case) defined on the state space S, we define $\|x-y\|_J = (J(x) - J(y))$ where $x, y \in S$ Note that in the `Extend` function, when the action space is continuous, we simply sample some k actions uniformly and then choose the best amongst them.

We may now define a sampling procedure that returns a tree of samples G from the given problem based on a distance measure $\|\cdot\|_J$. The algorithm closely resembles the

Function 2 Extend(s,s_{target},$\|\cdot\|_J$)

Data: Given state s and a target state s_{target} and a distance metric $\|\cdot\|_J$
Result: A sample $(s_{ext}, a_{ext}, r_{ext})$ such that s_{ext} can be reached from s on performing a_{ext} and is closest to s_{target} as defined by the distance metric

$d_{min} \leftarrow -\infty$;
for *every* $a \in A_s$ **do**
    Sample $(r', s') \leftarrow \widehat{\mathscr{M}}(s, a)$;
    if $\|s' - s_{target}\|_J > d_{min}$ **then**
        $d_{min} \leftarrow \|s' - s_{target}\|_J$;
        $(s_{ext}, a_{ext}, r_{ext}) \leftarrow (s', a, r')$;
    end
end

RRT algorithm in constructing the samples. The ConstructRRST procedure as defined in Algorithm 2, constructs a tree with samples of the form $(s_t, a_t, r_{t+1}, s_{t+1})$.

The tree is grown in a *greedy* manner with respect to the value function J as the action that maximizes reward(or minimizes cost) is chosen in the `Extend` function. Using the value function as a distance measure preserves the efficient exploration property of RRTs discussed above. Given a set of samples, we re-evaluate the value function using TD(0) as described in the next section.

Algorithm 2: ConstructRRST($N, \|\cdot\|_J$)

$E(G) \leftarrow \emptyset, V(G) \leftarrow s_{start}$;
$n \leftarrow 0$;
while $n < N$ **do**
    Sample a state s_{new} from S;
    $s_{near} \leftarrow$ `Nearest`$(s_{new}, V(G), \|\cdot\|_J)$;
    $(a_{ext}, s_{ext}, r_{ext}) \leftarrow$ `Extend`$(s_{near}, s_{new}, \|\cdot\|_J)$;
    $V(G) \leftarrow V(G) \cup \{s_{ext}\}$;
    $E(G) \leftarrow E(G) \cup \{(s_{near}, a_{ext}, r_{ext}, s_{ext})\}$;
    $n \leftarrow n + 1$;
end
return G;

Figure 3.1 shows an unbiased RRST grown in a toy 2D domain. The domain is a simple path planning task in $[0, 100]^2$ where the actions are simple and correspond to moving a fixed distance in any direction. The starting state is $(10, 10)$ and there is an obstacle region(shown in grey) and a goal region(shown in green). Colliding with the obstacle

results in a negative reward and reaching the goal earns a large positive reward.

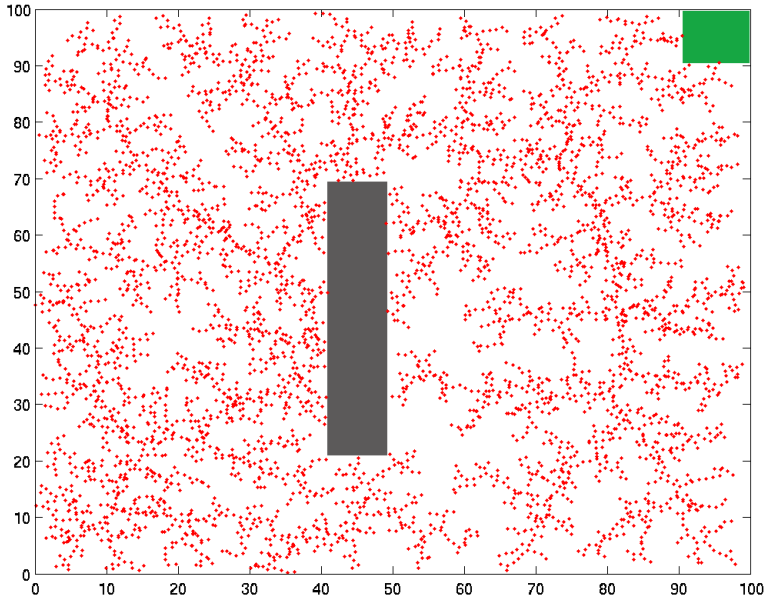

Figure 3.1: Example RRST grown in a simple toy domain. Here the function used J is uniform and unbiased.

3.2 Evaluating value functions

Given a sample tree G as described in the previous section, we formulate a discrete approximation to the original problem as follows. From every leaf node in the tree, a path is traced back to the root. This gives us several sample trajectories. We consider a discrete problem whose states are the nodes of the tree $V(G)$. We evaluate a discrete value function $\widehat{J}: V(G) \mapsto \mathbb{R}$ using the $TD(0)$ update given in Equation 2.2 on the sample trajectories. This can be thought of as backing up values along the trajectories of the tree. $TD(0)$ on discrete domains with a finite set of sample trajectories converges to a fixed value[Sutton, 1988].

This value function is discrete and is defined only on specific points. It has to be generalized across the entire state space of the original problem. This a standard regression task. We use nearest neighbor methods that build local models around a given query point. Depending on the nature of the model and definition of the locality there are several variants.

We use the following techniques and compare them in experiments described later.

3.2.1 Locally constant

Here we simply take the value as the average of the values of k-nearest-neighbors(k-nn) of x. The distance metric used to evaluate the nearest neighbors is Euclidean.

$$J(x) = \sum_{s_i \in \text{Nbr}(x)} \frac{1}{k} \widehat{J}(s_i) \tag{3.1}$$

For $k = 1$, the value of a state is generalized to its Voronoi region. By varying k we can vary the size of the neighborhood over which we generalize. We can reduce the variance in the estimate by learning local models, such as those discussed by Atkeson *et al.* [1997].

3.2.2 Locally linear

We assume the value of the function is linear within a neighborhood. The parameters of the function β are learnt by minimizing the least-squared error using simple linear regression as described below.

$$J(x) = \beta x = \sum_{j=1}^{n} \beta_j x_j + \beta_0 \tag{3.2}$$

$$\beta = \arg\min_{\beta} \sum_{s_i \in \text{Nbr}(x)} (\beta s_i - \widehat{J}(s_i))^2 \tag{3.3}$$

Here the loss function is defined only within the neighborhood, i.e., we use only the k-nearest-neighbors as training for the linear regression model. In higher dimensional state spaces, locally constant estimates tend to perform poorly because of high variance. In such cases linear methods tend to perform better [Atkeson *et al.*, 1997]. This results in a method which has more bias in terms of representation than simple k-nn regression.

3.2.3 Locally linear with Gaussian weights

We employ a Gaussian weighting scheme based on the distance of the points in the neighborhood. Closer points are given more weight [Atkeson *et al.*, 1997]. We define a Gaussian kernel $K(d) = \exp(-d^2)$ where d is a distance measure between the input states in the neighborhood and the target point. Here we use the Euclidean distance. We assign weights to the inputs using this kernel and regress. The modified error function for estimating β becomes

$$\beta = \arg\min_{\beta} \sum_{s_i \in \text{Nbr}(x)} K(|x - s_i|)(\beta s_i - \widehat{J}(s_i))^2 \tag{3.4}$$

We will call these techniques nearest neighbor temporal difference or NN-TD methods. We can define a procedure NN-TD(G) that takes a tree of sample transitions G as input and returns the generalized estimate of the value function. Note that these techniques use a 'lazy' approach to estimate the value at a given point, i.e., they do not perform any calculations until a point is queried and just maintain the set of input points and the corresponding values as such.

Thus in an implementation of this method, the generalization is done only when value function is estimated as $J(s)$ in the `Extend` and `Nearest` functions. Figure 3.2 show an example of the value function approximation obtained using NN-TD. The samples used are from the RRST(Figure 3.1) grown in the toy domain described in the previous section. Such nearest neighbor techniques are preferred as they have low bias in learning and can approximate any arbitrary function given enough data points. Alternatively we can evaluate the value function directly from the set of samples by suitably modifying Fitted Q-Iteration [Ernst *et al.*, 2005]. Here we may use parametric methods such as support vector regression and Gaussian processes regression. However, experimentally they did not perform well. One reason could be the following - these methods operate directly on the vector representation of a state from $\mathbb{R}^n$, whereas $TD(0)$ runs on a tabular representation of states. As the vertices of the set $V(G)$ are actually embedded on a manifold induced by complex dynamics of the system, methods that run on the $\mathbb{R}^n$ representation perform poorly.

Our method based on $TD(0)$ approximates the values better as it operates on the la-

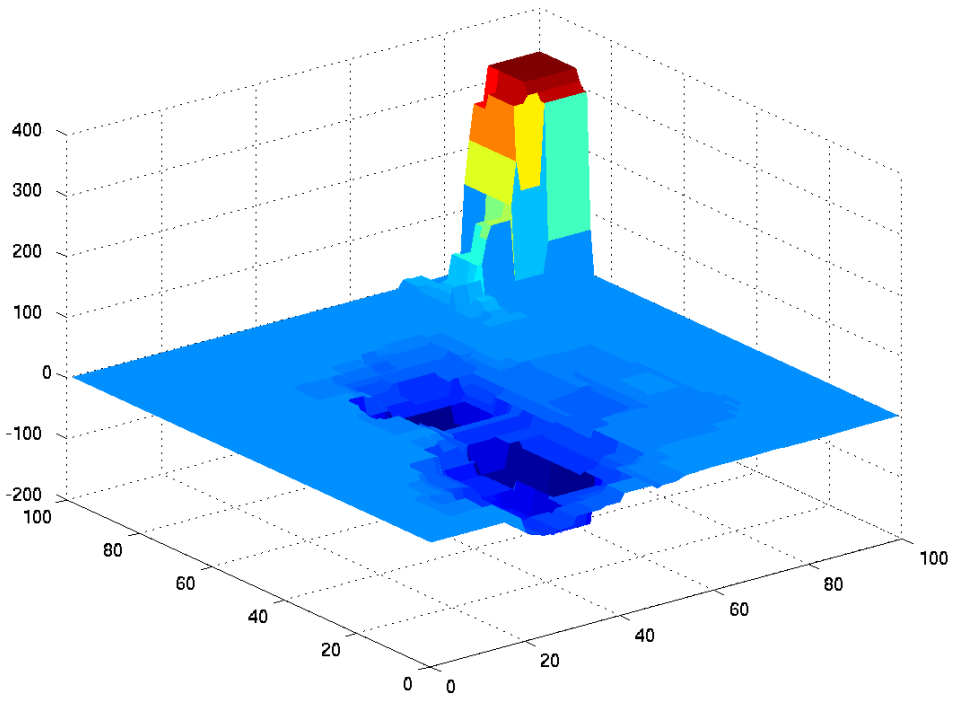

Figure 3.2: The value function is evaluated using the $TD(0)$ + locally linear method described on the 2D-toy domain. The samples used are from the RRST shown in Figure 3.1

tent space of the system. This due to the fact that the geodesic distance along the trajectories approximate the inherent metric of the manifold. Moreover, the accuracy of the approximation improves as the number of points in the trajectories increases. Given that our sampling technique RRST is asymptotically complete, $TD(0)$ combined with nearest neighbor regression is a favorable method to learn the value function, since it allows us to give asymptotic guarantees on the correctness of approximation. Comparisons between various techniques for policy evaluation are presented in the experiments section.

Now that we have described a class of techniques for estimating the value function from a set of samples, and a technique for generating samples by maximizing the value function, we can define an iterative procedure that alternates between RRST and NN-TD to make progressively better discrete approximations and solve the original problem. We describe our algorithm RRTPI in the following section.

3.3 Combing RL and RRT

The RRTPI algorithm is described in Algorithm 3. Given a control problem, we begin with a uniform estimate for the cost-to-go function J_0. We use this to generate a set of sample transitions using the ConstructRRST method described in Section 3.1. We then estimate the value function J_n from these samples using NN-TD. This estimate is used in the subsequent iteration to construct another sampling tree such that it is grown greedily with respect to the previously evaluated value function.

Algorithm 3: RRTPI(N)

```
Initialize uniformly J_0 ← 0;
n ← 1;
while n < N do
    G_n ←ConstructRRST(M_n, ‖·‖_{J_{n−1}});
    J_n ← NN-TD(G_n);
    n ← n + 1;
end
```

As we obtain better samples, the estimate of the optimal value function continues to improve. This proceeds in an iterative manner till we obtain a sufficiently optimal solution. The size of sample set M_n can be changed for different iterations. Typically initial iterations need more samples as the value function might not accurately estimate the optimal cost-to-go. Figures 3.1 and 3.2 show the first step in generating samples and the corresponding value function estimate on the simple toy domain previously described. As shown in the simulations in the following chapter, for sufficiently large M_n and N, the algorithm eventually reaches the optimal cost-to-go function.

3.4 Discussion

This method resembles policy iteration, which is a DP based technique for solving discrete MDPs [Sutton and Barto, 1998]. In policy iteration, an optimal policy π^* is found as follows. First begin with an random policy π. Evaluate this policy, i.e., find the value function J^π using some policy evaluation technique. Then define a new policy that is

greedy w.r.t the estimated value function. This step is called policy improvement. The corresponding new value function is again estimated and the steps are repeated till the policy converges to an optimal one. The similarities with our algorithm are now apparent and hence the name RRTPI.

The NN-TD step in our algorithm corresponds to policy evaluation, and the constructRRST step corresponds to policy improvement. We may think of our algorithm as extending policy iteration to continuous domain using samples to estimate both the policy and the value function. Also, we do not require full knowledge of the system dynamics as in the case of policy iteration. A generative model that allows us to draw samples is sufficient. This is a weaker assumption and allows us to solve a more general class of problems.

The LQR-RRT* technique [Perez *et al.*, 2012] method describes a similar approach of learning the optimal value function from experience, but assumes knowledge of the system dynamics in linearizable form. It is found that the accuracy of LQR methods falls rapidly as the dimensionality of the domain increases [Glassman and Tedrake, 2010]. Thus the assumptions made by LQR-RRT* hinder its ability to model more complex problems effectively. Systems such as the octopus arm, are easy to generate samples from but hard to fully specify in closed form. Our method can handle such domains as it only requires samples. It can also handle aribitrary cost functions. Supporting results are shown in the next chapter.

CHAPTER 4

EXPERIMENTS ON AUTONOMOUS PATH PLANNING

In this chapter we compare our algorithm with the fixed discretization, LQR based policy iteration and other versions of RRTPI. We compare their performance vs. the number of samples seen. The performance is measured by the discounted return obtained by the best trajectory in the sample tree. All results are averaged across 100 runs.

Fixed Discretization We discretize the space into uniform grids and run Dynamic Programming. The number of discretizations is taken as the number of samples for comparison.

LQR based Policy Evaluation The policy evaluation technique NN-TD is replaced with an LQR based evaluation technique. The method described by Perez *et al.* [2012] was followed.

RRTPI variants We use several variants of RRTPI got by choosing various nearest neighbor techniques as discussed in chapter . kNN-RRTPI with different values of k corresponds to the locally constant method. LL-RRTPI uses the locally linear method and LW-RRTPI uses the Gaussian weighting scheme. The neighborhoods for LL-RRTPI and LW-RRTPI are defined using 5-7 nearest neighbors.

All simulations were run on 3.4GHz 4 core system with 16GB of RAM.

4.1 Mountain Car Domain

In this domain, the goal is to drive an underpowered car in a valley up a steep hill. The state is a 2 dimensional continuous space consisting of the position and velocity of the car along the hill (Figure 4.1). The actions correspond to acceleration in the positive or negative direction. The agent keeps incurring a small negative reward till it reaches the goal. Detailed descriptions of the dynamics can be found in Singh and Sutton [1996]. This is an example of a underpowered domain. Comparison of performance is shown in Figure 4.2. The RRTPI algorithms obtain alternating set of samples and value estimates. The performance is represented as the value of the discounted return of the path of the best solution in a given sample set.

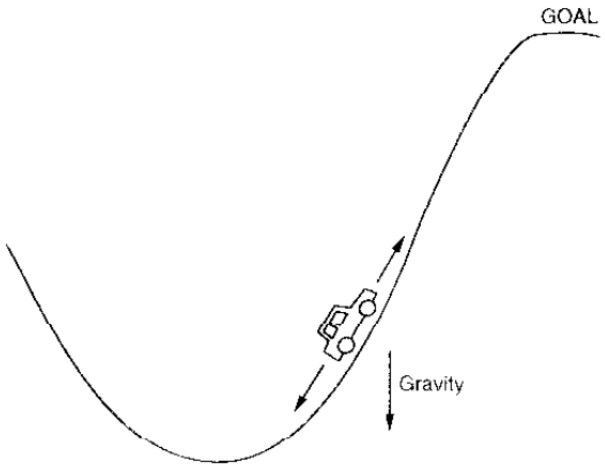

Figure 4.1: Mountain Car Domain

All methods were successfully able to solve the task. Although the discrete algorithm reaches the optimal performance roughly around the same time as the RRTPI algorithms, the space requirements are much higher. This is because at any given time, the RRTPI algorithms need to store only one set of state - values and a nodes and edges of the tree. In this experiment M_n was 2000. Whereas the discrete case needed to store 100×100 states. This problem would compound as the dimensionality of the problem increases. Also all algorithms display more efficient use of samples than plain discretization.

Using LQR gives good estimates initially but eventually both LW-RRTPI and 1nn-RRTPI perform better with the same number of samples. This can be attributed to LQR as

an estimator having lesser variance compared to 1nn. Performance of LL-RRTPI is almost similar to LQR since the respective value function estimates are similar in state spaces with smaller dimensions. Although as the complexity of the domain increases LL-RRTPI scales better as evident from the Acrobot experiments. LW-RRTPI performs the best among the algorithms even in this small dimension setting.

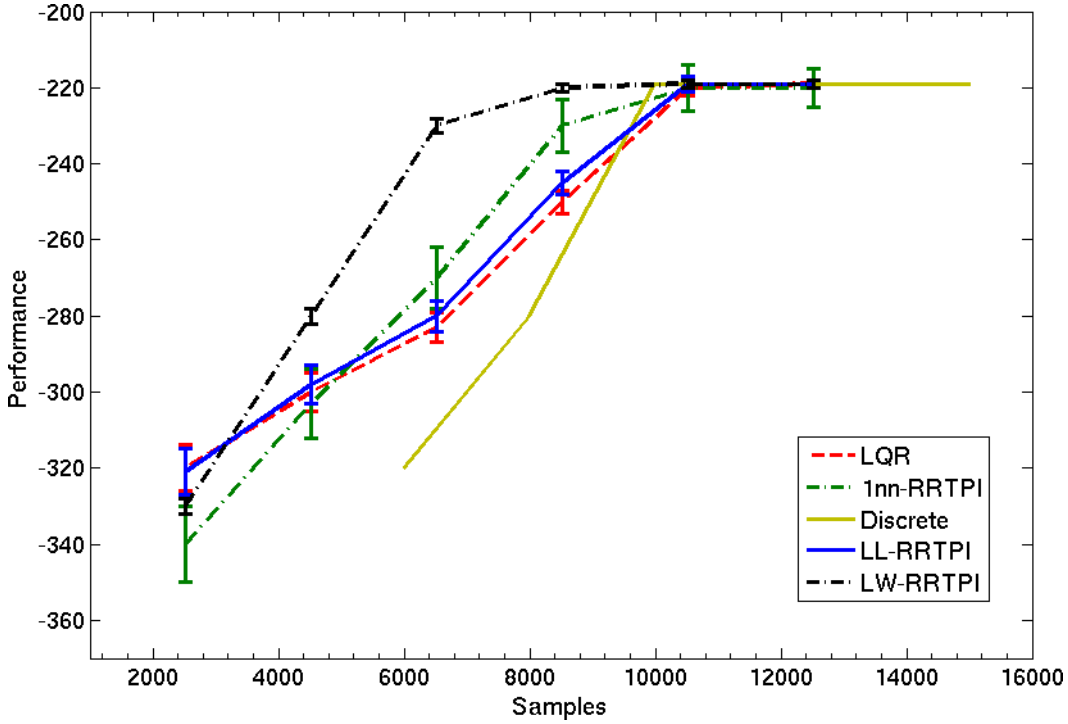

Figure 4.2: Comparison of various algorithms on the mountain car domain

4.2 Acrobot Domain

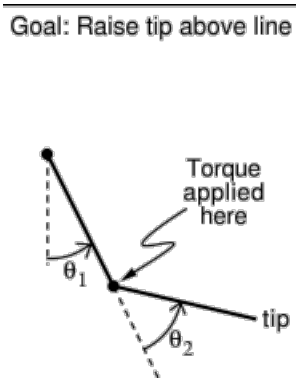

Figure 4.3: Acrobot Domain.

In this task, an acrobot must be brought to a vertically upright position. The acrobot is a two link robot with one fixed unpowered joint and a free joint powered joint (Figure 4.3). The system has four states consisting of the angular position and speed of the two joints. This is an underactuated system as only the free joint can be controlled and the robot has to learn to swing up by building momentum. The exact dynamics can be found in Murray *et al.* [1990]. The results on this domain are shown in Figure 4.4.

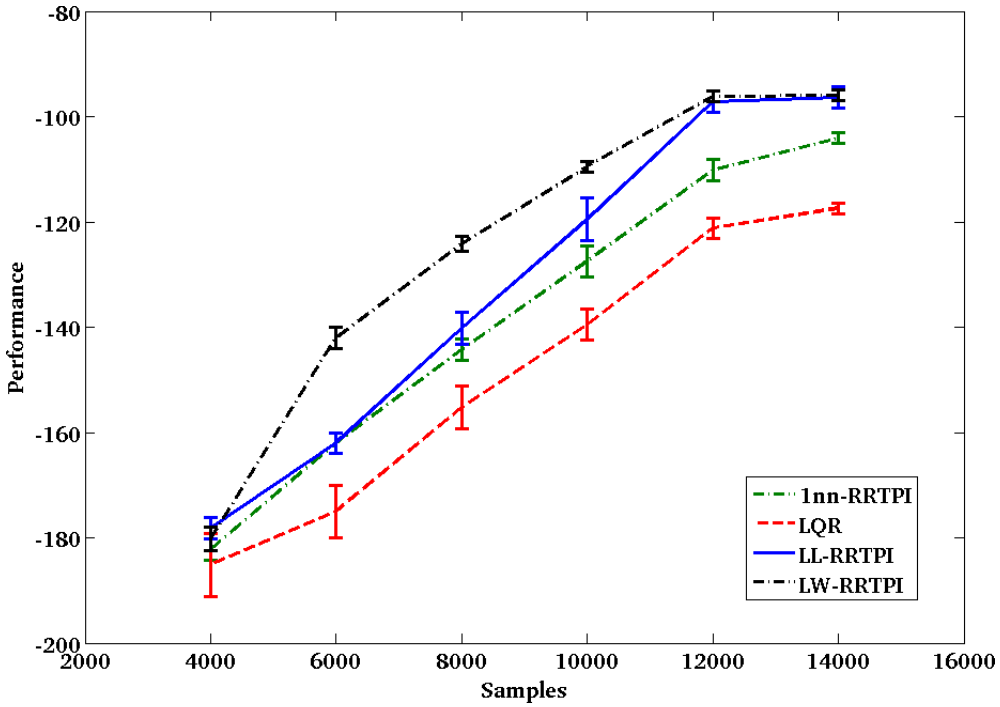

Figure 4.4: Comparison of various algorithms Acrobot domain.

In this task, LQR performs poorly compared to the nearest neighbor methods. Results from discretizations are not reported as they ran out of memory before a solution was found. LL-RRTPI and LW-RRTPI perform better performs better than 1nn due to lower variance in higher dimensions.

4.3 Octopus Domain

The octopus arm is a highly hyper-redundant manipulator. The arm is made up of compartments and it is controlled through activations of the muscles on walls of these compartments. We follow the model of the arm in 2-D space with 10 segments introduced

by Engel *et al.* [2006]. The aim of the arm is to reach a food particle located at some space around the arm. The state space corresponds to the position and velocity of the point masses on each segment. Thus the dimensionality of the state space is $22 \times 4 = 88$. We use a discrete subset of the possible actions as shown in the Figure 4.5. The goal of the arm is reach a target location with its end. The cost associated with every motion is uniform and reaching the goal results in a positive reward. Each segment has its own dynamics leading

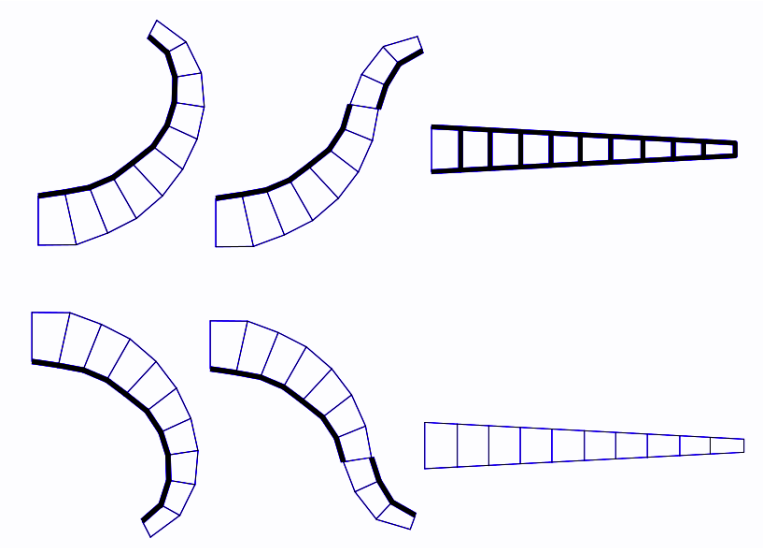

Figure 4.5: The 6 actions used in experiments. The thick lines indicate activation of muscles.

to complicated dynamics for the entire arm. Whereas it is much easier to generate samples for the next state using a one step model of the arm. Thus RRTPI is able to solve such complex high dimensional problems with a relatively small number of samples. We show comparison of the performance of 1nn-RRTPI and LW-RRTPI of the octopus domain in Figure 4.6.

4.4 Summary

We presented RRTPI the first algorithm that combines RRT style sampling with Reinforcement Learning to solve continuous space control problems with complex dynamics. By estimating the domain dependant distance measure from samples, our algorithm is able to work with complex, underactuated and underpowered systems. The algorithm is able to

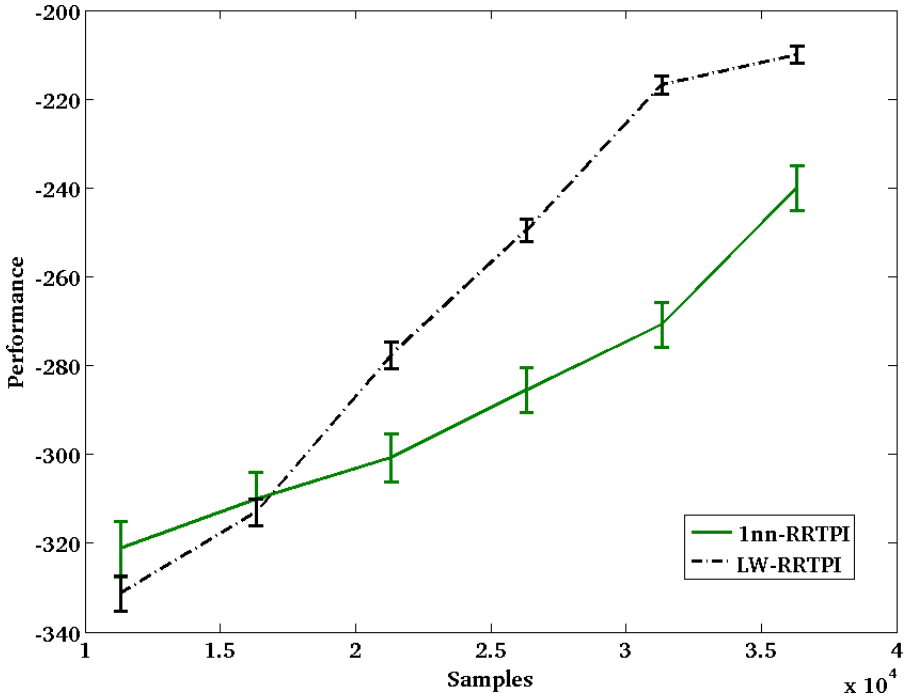

Figure 4.6: Comparison of RRTPI algorithms on the octpus arm domain.

solve a wider class of problems as compared to previous techniques as it works using only samples and does not make any assumptions on the form of the system dynamics.

The iterative nature of the RRTPI algorithm can be used to interleave planning and actual execution in a real robot. For instance, we may plan for a few iterations and once a satisfactory policy and value function are obtained, the robot can execute the resulting trajectory in real-time. This can be simply done by selecting actions greedily according to the value function. We may then use the resulting trajectory as samples for further iterations. If the same task is repeated several times this allows us to constantly improve performance. It can also be used to improve the accuracy of the one step model.

Transfer learning allows us to use knowledge from solving one particular task in solving a new but related task [Taylor and Stone, 2009]. The value function estimate can serve as a good representation for transfer learning [Taylor and Stone, 2005]. From the experiments we can see that RRTPI is sample efficient as compared to discretization and showing sample complexity bounds on these algorithms would be an interesting direction for future research.

CHAPTER 5

RECOGNIZING HUMAN ACTIVITIES

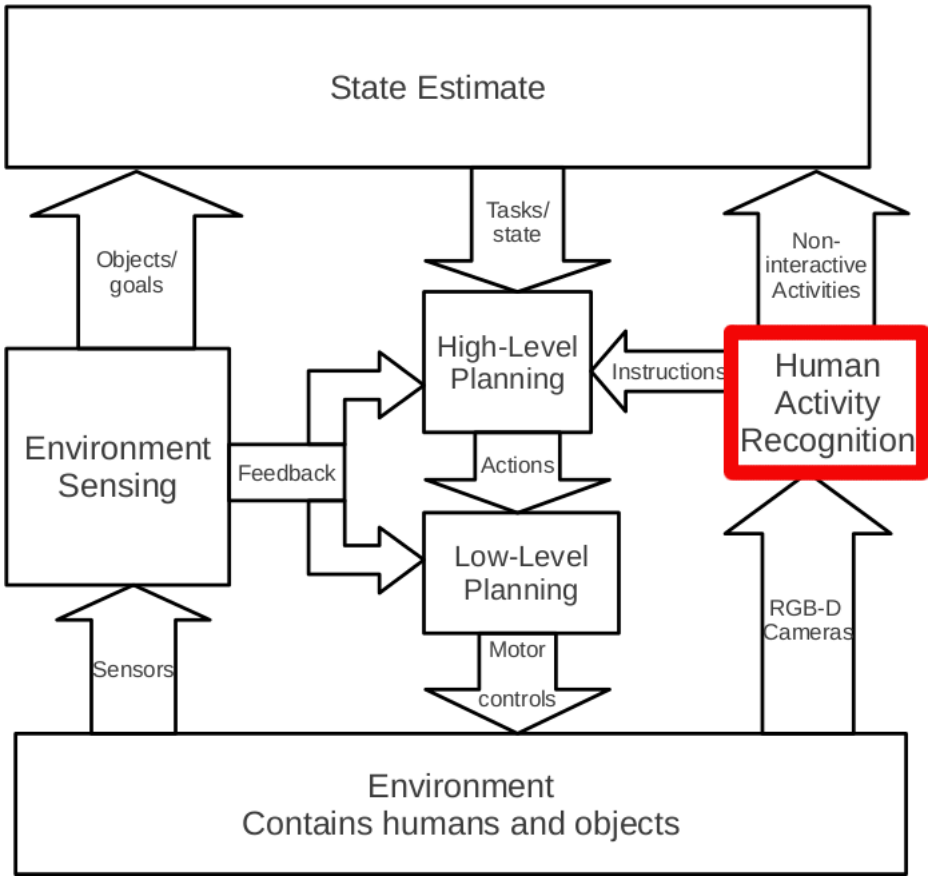

Figure 5.1: Current module discussed in overall framework highlighted in thick red lines.

Modern day robots work in environments wherein humans can act as rich source of knowledge. Robots can incorporate information gained through this unique source in decision making as well as task execution [Salah *et al.*, 2012]. For example, personal and service robots work in highly cluttered households or offices where humans perform activities ranging from simply moving around [Xia *et al.*, 2012] to complex actions involving object interaction [Koppula *et al.*, 2013] . In addition, humans may be involved with human-robot communication through gestures, say to pass some instructions [Korupolu V N *et al.*, 2012]. In such environments, monitoring human activities could give robots insights into whether assistance is being requested (through human instructions) or whether

an action pertains to an act of daily living and offering assistance may be helpful; even if not directly requested. This is similar in principle to 'how humans assist others', i.e., either they assist if assistance is sought or they foresee the need for assistance based on perception and knowledge.

As a result, we are particularly interested in developing a concise representation for a wide variety of actions; both communicative and conventional activities of daily living. We hypothesize that such communicative actions have distinct motion intrinsics as compared to conventional activities of daily living and can hence be used effectively to communicate with robots in the absence of verbal means. In this chapter we present a technique for identifying human activity as a part of the overall robot learning framework as shown in Figure 5.1.

5.1 Related Work

Though human activity recognition has been widely studied by computer vision researchers for over two decades [Aggarwal and Ryoo, 2011; Poppe, 2010], the field, owing to its applications in robotics, has recently started receiving a lot of attention by the robotics community as well. Most works have focused on classifying actions in indoor environments with emphasis on applications relevant to assistive and service robots. Zhu and Sheng [2009] proposed an activity recognition model based on multi sensor fusion from wearable inertial measurement units. They first classified activities broadly into three groups, namely: Zero displacement, Transitional and Strong displacement activity, followed by a finer classification using HMMs. Their approach was however restricted to very few activity classes and was computationally expensive. Mansur *et al.* [2013, 2011] proposed a physics based model to classify actions using HMMs as well. They described activities using joint torques and claimed them to be more discriminative features as compared to kinematic features. Zhang and Parker [2011] on the other hand followed a vision based approach. They proposed a 4D spatio-temporal feature that combined both intensity and Depth information by concatenating the depth as well as intensity gradients within a 4D hyper cuboid. Their approach was however dependent on the size of the hyper cuboid and

could not deal with scale variations. Sung *et al.* [2012] combined human pose and motion, as well image and point-cloud information in their model. They designed a hierarchical maximum entropy Markov model (MEMM), which considered activities as a set of sub-activities. The focus of their work was however on activity detection rather than activity recognition.

While most of these works focused primarily on generating different features, other improvements to robot perception came by the incorporation of domain knowledge [Teo *et al.*, 2012] and objects involved during an activity. This involves recognizing objects [Collet *et al.*, 2011; Sun *et al.*, 2013], tracking objects [Choi and Christensen, 2012] and 3D modeling of indoor environments [Henry *et al.*, 2012].

Gehrig *et al.* [2011] proposed a joint framework for activity recognition combining human intention, activity and motion within a single framework using low level visual features including Histogram of Sparse Flow and Histogram of Oriented Gradients. Koppula *et al.* [2013]; Koppula and Saxena [2013*a*,*b*]; Saponaro *et al.* [2013] incorporated the idea of object affordances to anticipate human activities and plan ahead for reactive responses. Pieropan *et al.* [2013] on the other hand introduced the idea of learning from human demonstration for activity recognition. They modeled interaction of objects with human hands, such that robots observing the humans could learn the role of an object in an activity and hence classify it accordingly.

5.2 Instructive and Non-instructive Activities

While past research excluded the possibility of interaction with the agent, this work aims to understand activities when interaction between robots and humans is possible, especially, in terms of the human passing possible instructions to a robot while also performing conventional activities of daily living. The focus of our work is to utilize distinctions in motion to differentiate between communicative/instructive actions and conventional activities of daily living.

We are interested in recognizing activities that a person may perform to instruct a robot,

in addition to conventional Non Interactive activities like *drinking water, chopping food, resting on a recliner, looking at time, cleaning face* etc. and obtain a descriptive labelling of the activities. Differentiating between Instructive and Non Instructive activities can prove to be beneficial for an assistant robot and complements past research on robot perception. This type of understanding can especially benefit robots involved in symbiotic interaction with humans [Rosenthal *et al.*, 2010]. To the best of our knowledge, such a direction of research has not been explored yet.

We propose the use of human-like *stylized gestures* as communicative actions and contrast them from conventional activities of daily living. *Stylized gestures* are symbolic representations of activities and are widely used by humans across cultures to communicate with each other when verbal communication is not possible. For example, to seek attention, one may raise his arm high and wave, similarly, to ask a robot to cut an object, one may symbolically perform the cutting action using his/her hands without using objects. Incorporating such *stylized gestures* into human robot interaction, not only makes communication between robots and humans more natural and aligned with how humans communicate with each other but can also benefit robots involved in symbiotic interaction with humans.

Before we can begin to develop a system for activity recognition, we need an efficient representation mechanism for human motion. We represent skeletal joint movements over time in a compact and efficient way. Specifically, we model pose trajectories in terms of directions taken by human joints over the duration of an activity. We introduce a novel activity descriptor: Histogram of Direction vectors (HODV) which transforms 3D spatio-temporal joint movements into unique directions; an approach which proves to be highly discriminative for activity recognition. We show that our proposed approach is efficient in distinguishing Instructive and Non Instructive activities in our novel RGBD dataset and also performs equally well on two public datasets: Cornell Activity Dataset (CAD -60) and UT-Kinect Dataset using off the shelf classification algorithms. Figure 5.2 shows the final framework used. The skeletal joints are represented in a simple yet powerful representation as described in section 5.3. The feature vectors are then used with a standard Support vector machine (SVM) classifier to obtain the final classification.

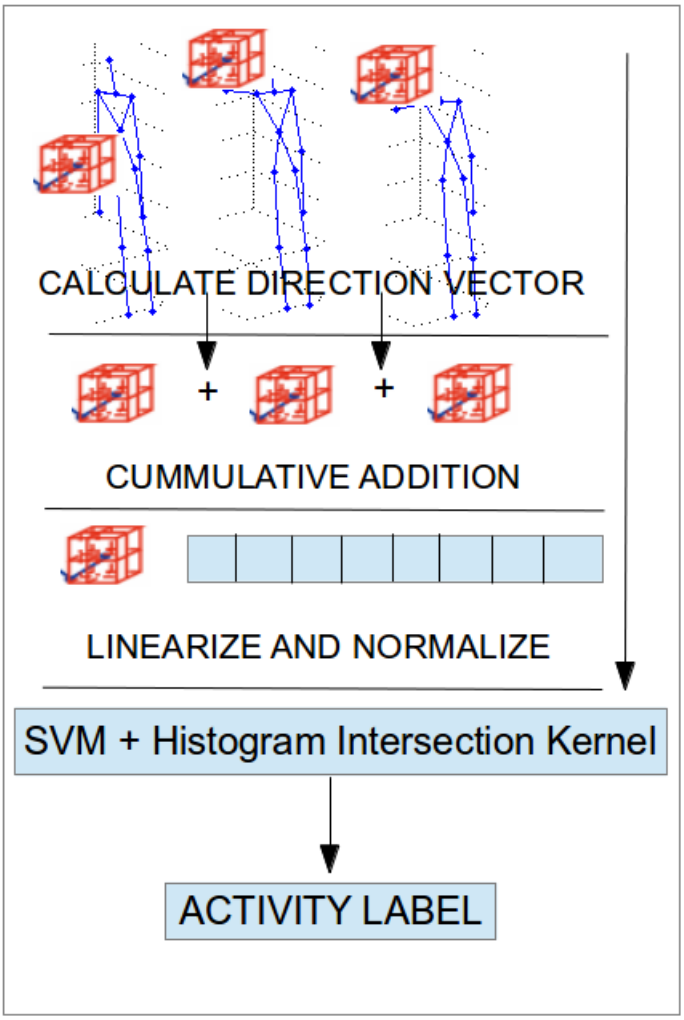

Figure 5.2: Formation of HODV combined with a prediction framework.

5.3 Histogram of Direction Vectors (HODV)

Activities usually consist of sequences of sub-activities and can be fundamentally described using two aspects: a) Motor trajectory and b) Activity context. For example, in a drinking activity, a subject picks a glass or a cup, brings it closer to his/her mouth and returns it. While there are numerous possibilities behind the context of the activity, as a glass could contain juice while a cup could contain coffee, thereby giving more meaning to the activity 'drinking' and answering a question: *What is probably being drunk?* The motor trajectory followed by most people for a generic drinking activity would predominantly be similar. We aim to exploit this similarity and introduce a *local motion* based action representation called *Histogram of Direction Vectors*, defined as the distribution of directions taken by each skeleton joint during all skeleton pose transitions during an activity.

The intuition behind the descriptor is that directions have a clear physical significance and capturing motion intrinsic as a function of direction should be discriminative across classes. We describe the 3D trajectory of each joint separately and construct the final descriptor by concatenating the direction vector histogram of each joint.

The algorithm as described in Figure 5.2, takes RGBD images as input and uses the Primesense skeleton tracker to extract skeleton joints at each frame. Each joint j_f^i is represented as shown:

$$j_f^i = [P_1^i, P_2^i, \ldots, P_{f_{max}}^i] \tag{5.1}$$

where P_f^i is the 3D Cartesian position of the joint i at frame f. The joint locations are then normalized by transforming the origin to the human torso, thereby making them invariant to human translation.

$$\forall f, i,\ P_f^i = P_f^i - P_f^{torso} \tag{5.2}$$

Direction vectors are then calculated for each joint j_f^i by computing the difference between joint coordinates of frame f and frame $f+\tau$, where τ is a fixed time duration (e.g., 0.1 seconds) in terms of frame counts. Mathematically, direction vectors are estimated for each joint at every frame as:

$$\vec{d_f^i} = \left[P_f^i - P_{f+\tau}^i\right], \forall f \in [1, 2, \ldots, f_{max} - \tau] \tag{5.3}$$

At each frame f, the local region around a joint i is partitioned into a 3D spatial grid. We chose 27 primary directions in the 3D space and represented the direction taken by a joint by the nearest primary direction in that grid. The grid entries represent real world directions such as, up, down, up-left, down-right and so on; resulting in a total of 27 directions. The direction vector corresponding to a joint i is mapped onto the index of one of 27 directions, by estimating the 3D euclidean distance between grid coordinates σ_q and the direction vector $\vec{d_f^i}$; with a vector being allotted a particular direction index q corresponding to the minimum distance. The goal is to find the specific direction index q^* that represents the direction which is at minimum euclidean distance from the direction

vector.

$$q^* = \text{argmin}\|\text{d}_\text{f}^\text{i} - \sigma_\text{q}\| \quad \forall \text{q} \in [1,2,\ldots,27] \tag{5.4}$$

where σ_q is the coordinate of grid index q. Let Q^f denote the vector of directions, with Q_q^f denoting the entries of vector Q^f at index q. The grid index q^* is then used to update vector Q^f:

$$Q_q^f = \begin{cases} 1 & \text{if } q = q^* \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

To attain the total number of times a particular direction was taken during an activity, we perform cumulative addition of vector Q^f at each frame.

$$h^* = \sum_f Q^f \tag{5.6}$$

where h^* is a vector revealing the number of times each direction was taken by a joint during the course of an activity. The vector h^* is then normalized to compute the feature vector h_i for joint i.

$$h_i = \frac{h^*}{\|h^*\|_1} \tag{5.7}$$

Normalizing the vector h^* gives us a histogram h_i, representing the probability of occurrence of each direction for a particular joint i, during the course of an activity. Further, each histogram h_i is concatenated to generate the final feature vector $H = [h_1, h_2, \ldots, h_i]$; namely the Histogram of direction Vectors.

We use HODV on our training data obtain a set of feature vectors. We then use an off the shelf classifier such as a Support Vector Machine with an appropriate histogram based distance function to classify testing instances. Forming a HODV requires two parameters - the dimension of the grid σ_q and time step τ. These are evaluated using a separate validation set. To further improve the feature vector, we assign a weight to each bin of the histogram based on how frequently it is observed in particular activities (classes) versus the entire dataset. This is similar to the term frequencyâĂŞinverse document frequency (TF-

IDF) weighting scheme in text corpora. In our predictions we rounded of these weights to either 1 or 0 values to feature ‘mask’. We find using this technique improves the accuracy across several datasets with different kinds of activities.

To summarize we first calculate the direction vector of each joint, sum over the period of this action, linearize and normalize to obtain a final vector that represents the action. This is then used a standard feature vector with a classifier.

CHAPTER 6

EXPERIMENTS ON ACTIVITY RECOGNITION USING HODV

We use publicly available RGBD datasets to evaluate our method. The CAD60 dataset has 12 actions performed by 4 different subjects [Sung *et al.*, 2012; Koppula *et al.*, 2013]. These actions are performed across different locations. They are longer and useful in the context of providing assistance. The UTKinect-Activity dataset has 10 subjects performing 10 actions with each subject performing every action twice [Xia *et al.*, 2012]. These are shorter actions and can be used as gestures in communicating instructions to the robot. We also introduce our own dataset - IITMInstructions3D which has 10 subjects performing 18 actions, half of which are instructive and half are non-instructive. Our method based on HODV achieves better classification than the state of the art in these datasets as shown below.

6.1 IITM Instruction 3D dataset

The activities were captured using a RGBD camera mounted on a customized pioneer P3Dx mobile robot platform. The robot was placed in an environment wherein appearance changed from time to time, i.e., the background and objects in the scene varied. In addition, the activities were captured at various times of the day leading to varied lighting conditions. A total of 5 participants were asked to perform 18 different activities, including 10 Communicative/Interactive activities and 8 Non-Interactive activities, each performed a total of three times with slight changes in viewpoint from the other instances. *'Catching the Robots attention', 'Pointing in a direction', 'Asking to stop', 'Expressing dissent', 'Chopping', 'Cleaning', 'Repeating', 'Beckoning', 'Asking to get phone' and 'facepalm'* were the 10 Robot-Interactive activities. In Robot-Interactive activities like *'Catching the*

Robots Attention', the human raises his arm and waves it. The action is symbolically similar to how a person would call another human if he or she is far away or not looking directly at them. *'Facepalm'* involved the human bringing his/her hand up to his head, similarly, the activity *'chopping'* involved a human repeatedly hitting one of his hands with the other hand, creating a stylized chopping action and so on. The non interactive activities were more conventional activities of daily living like *'Drinking something', 'Wearing a backpack', 'Relaxing', 'Cutting', 'Feeling hot', 'Washing face' 'Looking at time'* and *'Talking on cellphone'*.

We stress that our dataset is different from publicly available datasets as we represent a new mix of activities, more aligned with how humans would perform these in real life. In addition, the dataset involves wide variability in how the activities were performed by different people as subjects used both left and right hands along with variable time durations. For e.g., in the *'Drinking something'* activity, some subjects took longer to drink water and brought the glass to their mouth couple of times, while others took the glass to their mouth just once. The wide variety and variability makes recognition challenging. Of further interest, we can use some of the activities such as *'Facepalm'* and *'Expressing dissent'* to capture qualitative feedback (rewards) that can be given by a human.

6.2 Results

On our dataset, we ran experiments using three different settings. In the first, we classified actions into their respective categories using the entire dataset. In the second setting, we manually separated the activities into Communicative/Interactive activities and Non-Interactive activities and ran our classification algorithm on the two groups independently. In the third setting, we trained a binary classifier and labelled the activities as belonging to either of the two groups. Using the feature masking technique described earlier, we improve the accuracies as shown in Figure 6.2

All experiments were performed using 5 fold cross subject cross validation, such that, at a time, all instances of one subject were used for testing and the instances from the other subjects were used for training. None of the instances used for training were ever present

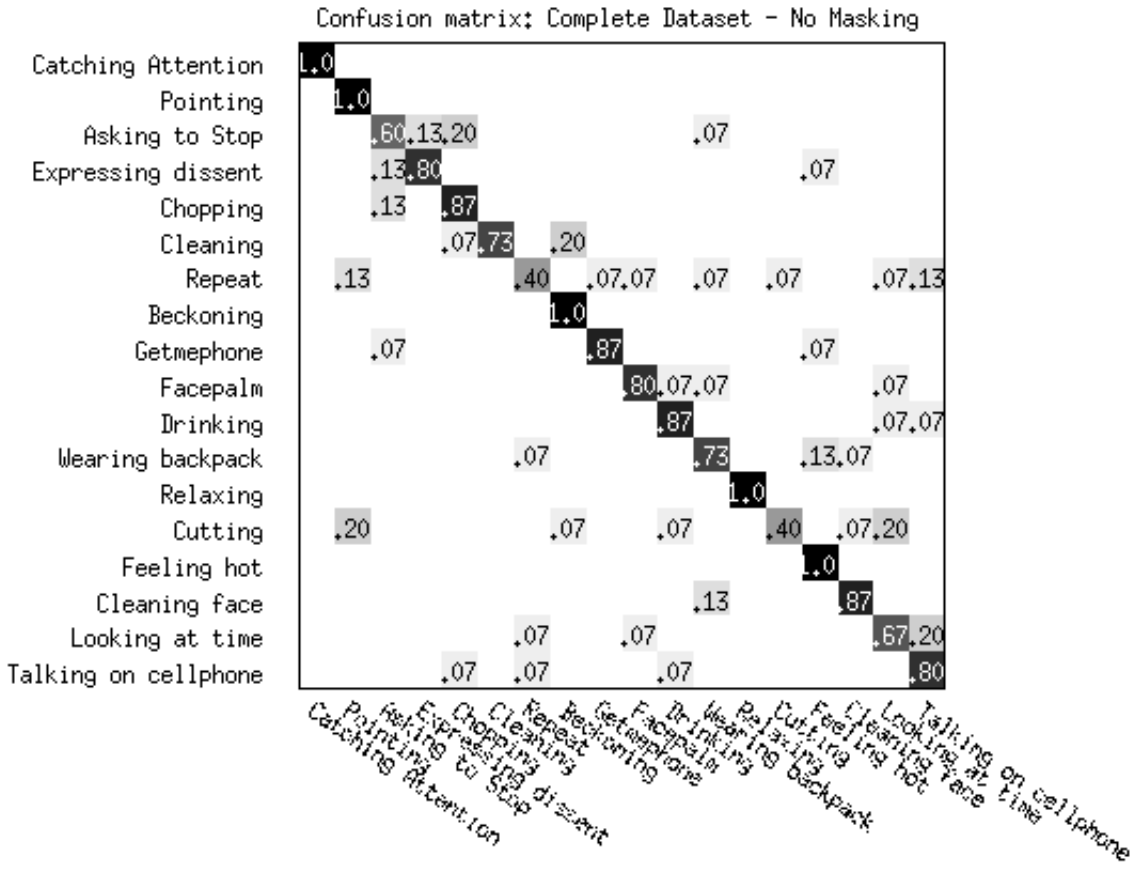

Figure 6.1: IITM Dataset - Prediction using HODV.

in the test set at the same time. Figure 6.1 presents the results of our algorithm on the IITM Instructions 3D dataset.

Most of the activities are classified with good accuracy apart from *Repeat* and *Facepalm*, mostly because of the same motion trajectory in both the actions. Also, as visible in Figure 6.2 many activities such as *Asking to stop, Repeat, Drinking, Wearing backpack* and *Cleaning face* were better classified after feature masking. The average accuracy attained without feature masking was 80%, while with feature masking the average accuracy improved to 82.59%.

Figures 6.3 and 6.4 show the results of our second experimental setting. The average classification accuracy for Interactive actions was 84%, while for Non Interactive actions, the average accuracy was 86.67%. Like in the previous setup, the algorithm was able to accurately classify actions which had distinct motion trajectories but got confused with actions with very similar motion like *Repeat* and *Facepalm*.

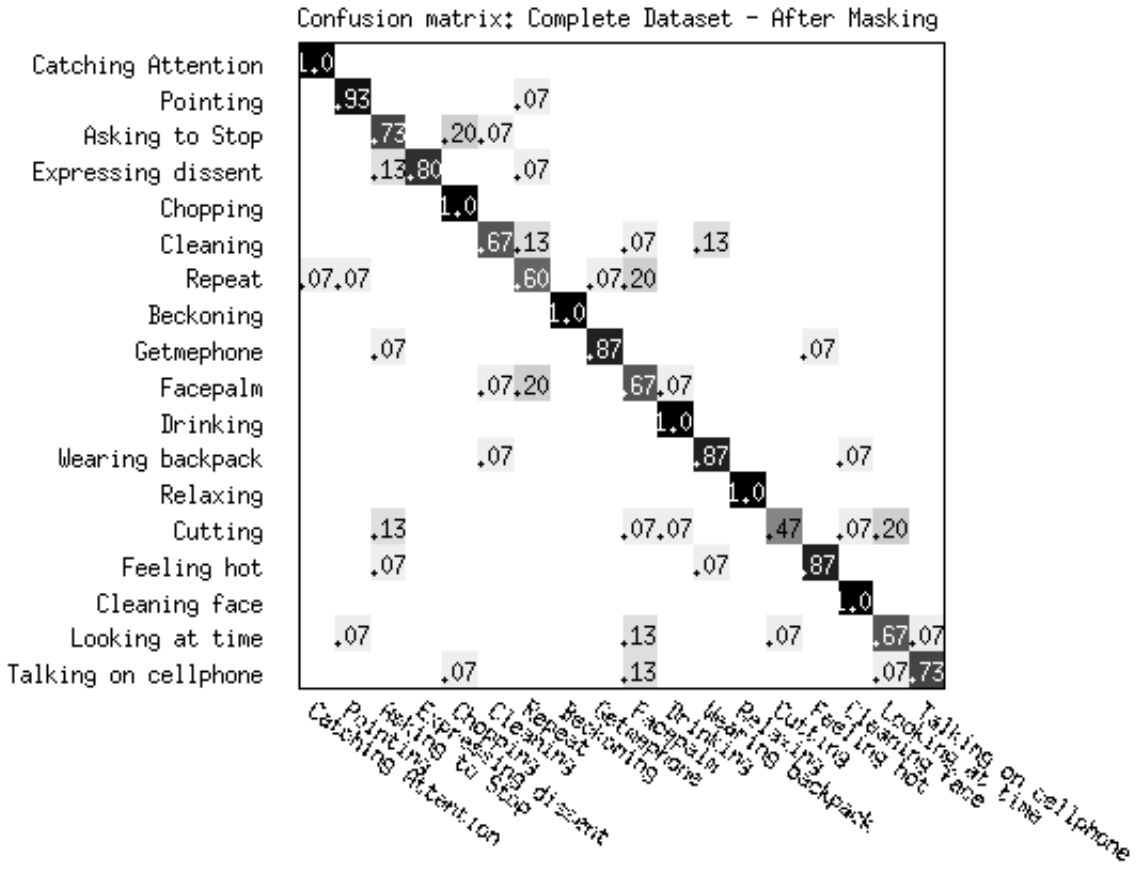

Figure 6.2: IITM Dataset - Prediction using HODV + Feature Masking.

In the third experimental setup, we classified an activity into either of the two groups. The algorithm performed very well and returned an average classification accuracy of 89.26%. Interactive actions were classified with an accuracy of 92.67% while Non Interactive activities were classified correctly with an accuracy of 85%. This classification paradigm could be essential for the development of hierarchical models where the first level could be an Interactive Vs Non-Interactive classification, followed by a finer categorization into an exact activity.

6.2.1 Cornell Activity Dataset (CAD-60)

The dataset comprises of 60 RGBD video sequences of humans performing 12 unique activities of daily living. The activities have been recorded in five different environments: Office, Kitchen, Bedroom, Bathroom, and Living room; generating a total of 12 unique

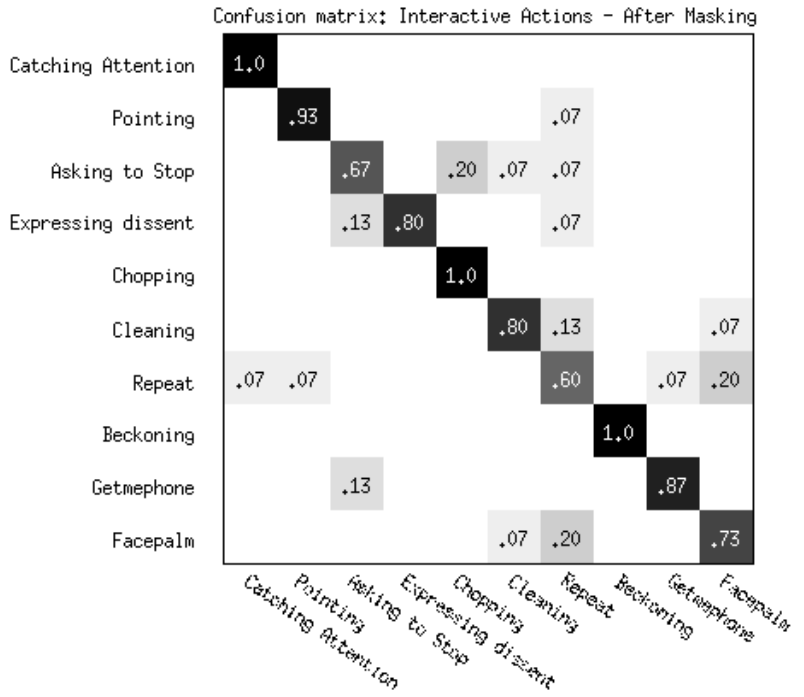

Figure 6.3: Confusion matrix of Interactive/Communicative actions after Feature Masking.

Location	**Sung *et al***		**Koppula *et al***		**HODV**	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Bathroom	72.7	65.0	88.9	61.1	**96.29**	**97.00**
Bedroom	76.1	59.2	73.0	66.7	**77.19**	**76.66**
Kitchen	64.4	47.9	**96.4**	**85.4**	88.59	84.79
Living Room	52.6	45.7	69.2	68.7	**75.04**	**71.87**
Office	73.8	59.8	76.7	75.0	**81.77**	**80.00**
Overall Average	67.9	55.5	80.8	71.4	**83.77**	**82.06**

Table 6.1: Results comparing our algorithm with the state of the art for the CAD60 dataset

activities performed by four different people: two males and two females. The activities are *'Rinsing mouth', 'brushing teeth', 'wearing contact lens', 'talking on the phone', 'drinking water', 'opening pill container', 'cooking (chopping)', 'cooking (stirring)', 'talking on couch', 'relaxing on couch', 'writing on whiteboard'* and *'working on computer'*. We used the same experimental setup (4 fold cross-subject cross validation) and compare precision-recall values for the 'New Person' setting as described in Sung *et al.* [2012]. Table 6.1 shows a comparison of our algorithm with other state of the art approaches. The algorithms mentioned in table 1 use visual features in addition to skeleton data. This work is largely restricted to the use of skeleton data for classification. In spite of this, our approach is able

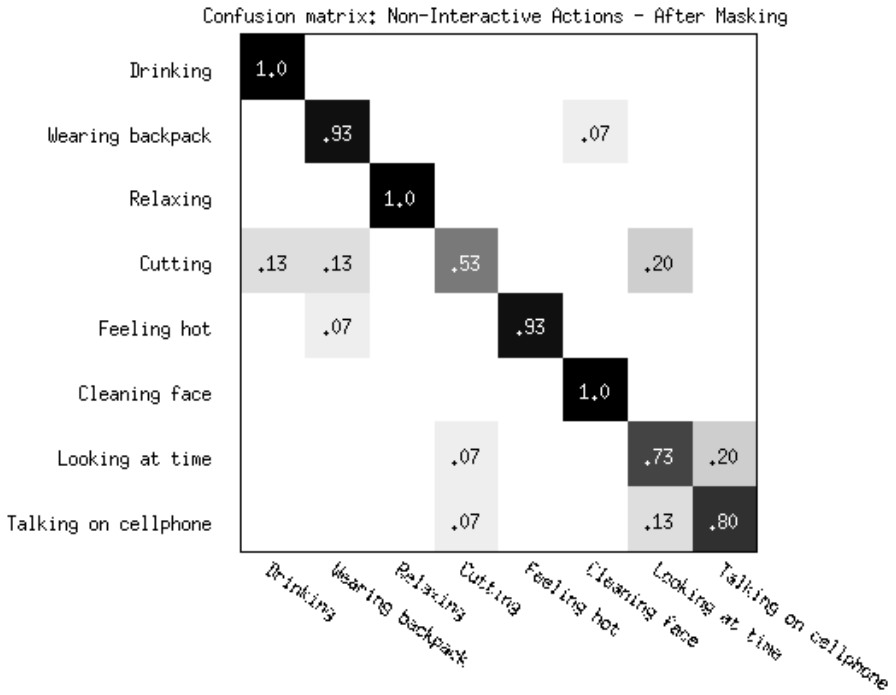

Figure 6.4: Confusion matrix of Non-Interactive actions after Feature Masking

to still out perform the other algorithms.

6.2.2 UTKinect Action Dataset

The UTKinect Action Dataset [Xia *et al.*, 2012] presents RGBD video sequences and skeleton information of humans performing various activities from different views. 10 subjects perform 10 different activities namely: *walk, sit down, stand up, pick up, carry, throw, push, pull, wave hands* and *clap hands*. Each subject performs an activity twice.

There are a total of 200 instances of different activities in this dataset. Since each skeleton is described by 20 joints, our feature vector is of dimensions 20×27, i.e., a total of 540 features were used for classification in this dataset. For this dataset, we compare our approach with the state of the art methodology called histogram of 3D skeleton joint positions (HOJ3D) [Xia *et al.*, 2012] using Leave one Sequence out Cross validation (LOOCV) and cross subject validation as defined previously in this paper. This dataset has activities which look very similar e.g., Sit down and Stand Up. Our high accuracies reveal the superiority of our algorithm in distinguishing such actions, which despite looking similar,

Method	**Accuracy (%)**
HOJ3D (LOOCV)	90.92
HODV (Cross Subject)	84.42
HODV (LOOCV)	87.44
HODV + Masking (Cross Subject)	89.45
HODV + Masking (LOOCV)	91.96

Table 6.2: Comparison of our algorithm with HOJ3D on the UT-Kinect dataset

have distinct trajectory directions, aptly captured by our approach. The overall accuracies attained on the dataset are shown in Table 6.2. Clearly, our approach generates better accuracy as compared to the Histogram of 3D joints algorithm under the LOOCV setting. The performs drops a bit under the cross subject cross validation scheme.

6.3 Summary

- We provide an annotated RGBD Human Robot Interaction dataset consisting of 18 unique activities including 10 'stylized gestures' posed as Interactive activities as well as 8 Non Interactive conventional activities of daily living within the same dataset.
- We propose a novel and computationally efficient activity descriptor based on pose trajectories.
- We provide analysis of our algorithm on two public datasets and demonstrate how our algorithm can be used for both Interactive and Non Interactive activities.

Our algorithm is able to distinguish between Interactive and Non Interactive activities with good accuracy. It works well even when subjects take different time duration to complete an activity. Further, since we follow a histogram based representation, classification is invariant to the number of times an action is performed within an activity. For. e.g., a circle could be made once or five times. As long as the feature vector is normalized and if an action is symmetric (activities involving mirror directions eg: waving), the number of times the action is performed or the starting point of the activity would not hamper classification.

The descriptor also benefits from being computationally efficient as the only calculations involved for each joints are -

- Calculation of direction vectors, which can be performed in constant time.
- Updating appropriate Histogram bins which is linear in the number of frames and can be performed real-time as and when new frames are captured.

This makes HODV an efficient, yet effective feature vector for classifying human activities.

CHAPTER 7

CONCLUSIONS

In this thesis, we work towards developing a learning framework for a domestic robot. This framework supports autonomous learning and incorporates human inputs whenever available.

1. We present a path planning technique that produces optimal paths in complex domains with underpowered and under-actuated dynamics. This technique uses reinforcement learning to automatically derive the domain dependent sub-Reimmanian metric.
2. We also present a method for classifying human activities. This method uses a feature vector based on Histogram of Direction Vectors and gives high accuracy across a variety of instructive and non-instructive activities.

As described in Chapter 1, these modules are a part of a broad architecture for robot learning.

7.1 Future Direction

While each of the modules described offer several opportunities for improvement, they also give us ways to integrate with the various other modules of a robot learning architecture. The RRTPI algorithm we presented, estimates a model of the dynamics of the environment. This can be potentially used as way of learning about the robot's environment. We can also interleave real-time executions with the iterative planning to incorporate feedback from the environment into the model. The HODV feature vector, as mentioned earlier, is based purely on motion of the humans. It might be combined with several other visual features, to give a more holistic approach towards recognizing human intent. Since the feature vector can easily be built in real-time and in a cumulative manner, we can use it to anticipate or predict the likelihood of what the activity, the human is performing at any given instant.

This is useful in planning out high-level tasks in conjunction with a framework such as instructions [Korupolu V N *et al.*, 2012]. And finally, in order to complete the robot learning architecture, we have to develop an 'environment sensing' module.

REFERENCES

Aggarwal, J. and **M. Ryoo** (2011). Human activity analysis: A review. *ACM Comput. Surv.*, **43**(3), 16:1–16:43.

Argall, B., **S. Chernova**, **M. Veloso**, and **B. Browning** (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, (57), 469–483.

Atkeson, C. G., **A. W. Moore**, and **S. Schaal** (1997). Locally weighted learning. *Artificial Intelligence*, **11**, 11–73.

Boyan, J. A., Technical update: Least-squares temporal difference learning. *In Machine Learning*. 2002.

Cheng, P. and **S. M. Lavalle**, Reducing metric sensitivity in randomized trajectory design. *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems*. 2001.

Choi, C. and **H. I. Christensen** (2012). Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *The International Journal of Robotics Research*, **31**(4), 498–519.

Collet, A., **M. Martinez**, and **S. S. Srinivasa** (2011). The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, **30**(10), 1284–1306.

Deisenroth, M., **G. Neumann**, and **J. Peters** (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, **21**, 388–403.

Engel, Y., **P. Szabo**, and **D. Volkinshtein**, Learning to control an octopus arm with gaussian process temporal difference methods. *In Advances in Neural Information Processing Systems*. 2006.

Ernst, D., **P. Geurts**, **L. Wehenkel**, and **L. Littman** (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, **6**, 503–556.

Gammell, J. D., **S. S. Srinivasa**, and **T. D. Barfoot**, Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2014*.

Gehrig, D., **P. Krauthausen**, **L. Rybok**, **H. Kuehne**, **U. D. Hanebeck**, **T. Schultz**, and **R. Stiefelhagen**, Combined intention, activity, and motion recognition for a humanoid household robot. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011.

Glassman, E. and **R. Tedrake**, A quadratic regulator-based heuristic for rapidly exploring state space. *In Proceedings of the IEEE International Conference on Robotics and Automation*. 2010.

Henry, P., **M. Krainin**, **E. Herbst**, **X. Ren**, and **D. Fox** (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, **31**(5), 647–663.

Karaman, S. and **E. Frazzoli** (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, **30**(7), 846–894.

Knox, W. B. and **P. Stone**, Interactively shaping agents via human reinforcement: The tamer framework. *In The Fifth International Conference on Knowledge Capture*. 2009.

Kober, J., **D. Bagnell**, and **J. Peters** (2013). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, **32**(11), 1236–1272.

Koppula, H. S., **R. Gupta**, and **A. Saxena** (2013). Learning human activities and object affordances from rgb-d videos. *International Journal of Robotics Research*.

Koppula, H. S. and **A. Saxena**, Anticipating human activities for reactive robotic response. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013*a*.

Koppula, H. S. and **A. Saxena** (2013*b*). Anticipating human activities using object affordances for reactive robotic response. *Robotics: Science and Systems, Berlin*.

Korupolu V N, P., **S. S. Manimaran**, and **B. Ravindran**, Instructing a reinforcement learner. *In Proceedings of the 25th FLAIRS Conference*. 2012.

Lavalle, S., *Planning Algorithms*. Cambridge University Press, 2006.

LaValle, S. M. and **J. J. Kuffner** (2001). Randomized kinodynamic planning. *International Journal of Robotics Research*, **20**(4), 378–400.

Mahadevan, S., Machine learning for robots: A comparison of different paradigms. *In Workshop on Towards Real Autonomy , IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1996.

Mansur, A., **Y. Makihara**, and **Y. Yagi**, Action recognition using dynamics features. *In IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011.

Mansur, A., **Y. Makihara**, and **Y. Yagi** (2013). Inverse dynamics for action recognition. *IEEE Transactions on Cybernetics*, **43**(4), 1226–1236.

Murray, R., **R. M. Murray**, **J. Hauser**, and **J. Hauser** (1990). A case study in approximate linearization: The acrobot example.

Ng, A. Y. and **M. Jordan**, Pegasus: A policy search method for large mdps and pomdps. *In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. 2000.

Nguyen-Tuong, D. and **J. Peters** (2011). Model learning in robotics: a survey, cognitive processing. *Foundations and Trends in Robotics*, **12**(4).

Nilsson, N. J. (1984). Shakey the robot. *Technical Note 323. AI Center, SRI International*.

Perez, A., **R. Platt**, **G. Konidaris**, **L. Kaelbling**, and **T. Lozano-Perez**, Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. *In Proceedings of the IEEE International Conference on Robotics and Automation*. 2012.

Pieropan, A., **C. H. Ek**, and **H. Kjellstrom**, Functional object descriptors for human activity modeling. *In IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013.

Poppe, R. (2010). A survey on vision-based human action recognition. *Image Vision Comput.*, **28**(6), 976–990.

Rosenthal, S., **J. Biswas**, and **M. Veloso**, An effective personal mobile robot agent through symbiotic human-robot interaction. *In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

Salah, A. A., **J. R. del Solar**, **Ã. MeriÃğli**, and **P.-Y. Oudeyer**, Human behavior understanding for robotics. *In HBU*, volume 7559 of *Lecture Notes in Computer Science*. Springer, 2012.

Saponaro, G., **G. Salvi**, and **A. Bernardino**, Robot anticipation of human intentions through continuous gesture recognition. *In International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2013.

Singh, S. and **R. S. Sutton**, Reinforcement learning with replacing eligibility traces. *In MACHINE LEARNING*. 1996.

Sun, Y., **L. Bo**, and **D. Fox**, Attribute based object identification. *In IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013.

Sung, J., **C. Ponce**, **B. Selman**, and **A. Saxena**, Unstructured human activity detection from rgbd images. *In International Conference on Robotics and Automation*. 2012.

Sutton, R. and **A. Barto**, *Reinforcement learning: An introduction*. MIT press, 1998.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, **3**.

Taylor, M. E. and **P. Stone**, Behavior transfer for value-function-based reinforcement learning. *In In The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. 2005.

Taylor, M. E. and **P. Stone** (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, **10**(1).

Teo, C. L., **Y. Yang**, **H. Daume**, **C. Fermuller**, and **Y. Aloimonos**, Towards a watson that sees: Language-guided action recognition for robots. *In IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012.

Xia, L., **C. Chen**, and **J. Aggarwal**, View invariant human action recognition using histograms of 3d joints. *In Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. 2012.

Zhang, H. and **L. E. Parker**, 4-dimensional local spatio-temporal features for human activity recognition. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011.

Zhu, C. and **W. Sheng**, Multi-sensor fusion for human daily activity recognition in robot-assisted living. *In Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, HRI '09. 2009.

Zucker, M., **J. Kuffner**, and **J. Bagnell**, Adaptive workspace biasing for sampling-based planners. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2008.

LIST OF PAPERS BASED ON THESIS

1. **S. S. Manimaran and B. Ravindran**, “RRTPI: Policy Iteration on Continuous Domains using Rapidly-exploring Random Trees", *In the Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014), pp. 4362-4367. Hong Kong, China. IEEE Press.*

2. **Adwitteey Chrungoo, S. S. Manimaran and B. Ravindran**, “Activity Recognition for Natural Human Robot Interaction", *In the Proceedings of the Fifth International Conference on Social Robotics (ICSR 2014), pp. 84-94. AAAI Press.*