Mirroring: A General Approach For Plan And Goal Recognition

Mor Vered

Department of Computer Science

Ph.D. Thesis

Submitted to the Senate of Bar-Ilan University

Ramat-Gan, Israel January 2018

This work was carried out under the supervision of Prof. Gal A. Kaminka, Computer Science Department, Bar-Ilan University.

Acknowledgments

First and foremost I would like to thank my advisor, Prof. Gal A. Kaminka. Thank you for setting the bar too high for comfort but not out of reach. Thank you for believing in me when I did not and teaching me that we are all imposters in our way. Thanks for rejoicing in my success and sharing my disappointments. You have taught me much more beyond science and for that I am eternally grateful.

My dad, my protective bodyguard in life. Thanks for all of your love and support, I could not do this without you.

To my shadow in life and my watching angel, mom, this is for you.

Thanks to my wonderful children who have taught me the meaning of love, life, patience and how to best manage my time.

And finally last but by no means least I would like to thank my supportive partner and best friend Sahar. Thanks for being the drive and motivation behind my ambition and enabling me to pursue my dreams. You are a rare gentleman and I am very thankful. I love you. Better together.

Abstract

Plan recognition is the task of inferring the plan of an agent, based on an incomplete sequence of its observed actions. The problem may be further sub categorized into offline and online plan recognition. In *offline* versions of the problem, the entire sequence is given to the recognizer at once. In contrast, in *online* recognition the observations are provided incrementally. The traditional approach to plan recognition has been to compare observations against a dedicated *plan library* representing all known plans to achieve known goals in a manner that facilitates efficient inference. This approach fails when encountering unknown plans or when dealing with continuous domains where the potential plan possibilities may be infinite. A recent approach to plan recognition uses a planner to dynamically generate plans for given goals, thus eliminates the need for the a traditional plan library. While an inspiring approach it poses many problems for online recognition in continuous environments.

A key problem in previous formulations of plan recognition over *continuous* domains is the early commitment to specific discretizations of the environment and the observed agent's actions, often leading to a reduction in recognition accuracy. To address this we introduce *Mirroring*. Inspired by mirroring processes hypothesized to take place in human brains, *Mirroring* is a formalization of recognition problems which admits continuous environments, as well as discrete domains. We further develop *Mirroring* as a complete online goal recognition approach that uses a black-box planner to generate recognition hypotheses, avoiding the prevalent assumption in current approaches, which rely on a dedicated *plan library*. Due to the suitability over continuous domains we can now apply continuous-world motion planners in plan recognition.

We proceed to provide formal arguments for the usefulness of Mirroring, and

empirically evaluate Mirroring over a thousand recognition problems in three continuous domains and six classical planning domains. We also proceed to contrast machine and human recognition in two challenging domains, revealing insights as to human capabilities; and finally we compare Mirroring to library-based methods.

As Mirroring requires multiple calls to a planner within the recognition process it can be inefficient for online recognition. Recognizing goals with minimal domain knowledge as an agent executes its plan requires efficient algorithms to sift through a large space of hypotheses. We therefore identify two independent decision points within the *Mirroring* algorithm where heuristics may be used to improve online run-time. We specify such heuristics for continuous domains, prove guarantees on their use, and empirically evaluate both the performance and efficiency of our algorithm over hundreds of experiments in both a 3D navigational environment and a cooperative robotic team task. We additionally test the durability of our approach by experimenting over scenarios with varying recognition difficulty, with both evenly spread and clustered goals.

As a final optimization method we further develop an online approach to recognize goals in both continuous and discrete domains using a combination of *Mirroring* and a generalized notion of landmarks adapted from the planning literature. Extensive experiments demonstrate the approach is more efficient and substantially more accurate than state-of-the-art.

Table of Contents

List of Figures

List of Algorithms

List of Tables

1 Introduction

The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say

J.R.R. Tolkien, The Fellowship of the Ring

As the human population and life expectancy increases so does the need for integrating robots and virtual agents closely into everyday human life. These agents may provide care and attention where otherwise manpower is lacking. To create better agents that interact seamlessly with humans we need to draw lessons from what we know of human social cognition. Designing an agent inspired by these processes will provide an agent that is more predictable, less threatening and overall welcome to its human benefactor.

One important aspect of human social cognition is the innate ability to perform quick and efficient intention recognition. This ability enables humans to reason about the hidden goals of other agents around them through observations of their actions. In humans this ability is hypothesized to come from the existence of a *mirror neuron system* [85, 84]. Mirror neurons have first been discovered in the

early 90's. These neurons were seen to fire both when a monkey manipulated an object and also when it saw another animal manipulate an object. Recent neuro-imaging data indicates that the adult human brain is also endowed with a *mirror neuron system* for matching the observation and execution of actions within the adult human brain. This system is hypothesized to give humans the ability to infer the intentions leading to an observed action using their own internal mechanism. It is also attributed to other high level cognitive functions such as imitation, action understanding, intention and language evolution. Consequently, the human mirror neuron system may be viewed as a part of the brains' very own plan/goal recognition module and can be used to recognize the actions and goals of other agents from a series of observations of the other agents' actions.

It is therefore no wonder that plan, activity, and intent recognition (PAIR) [99, 91, 18, 40] is a fundamental research area in artificial intelligence, tackling the problem of inferring the hidden mental attitudes of an observed agent. Given a partial sequence of observations of an agent, PAIR algorithms infer one or more of the following: a complete sequence of the agent's actions and their effects, future actions, a classification of the observed activity, and the intended goal(s) [10, 9, 52, 39, 12, 11, 49, 63]. The problems have many applications in continuous environments, e.g., for recognizing intended gestures and sketches whether in air or on paper [86, 92], anticipating user commands [10], suspicious behavior recognition [45] or for recognizing intended navigational goals [113].

Also, as autonomous agents and robots are being incorporated more and more into everyday lives there is a rising need to address the behavior of these agents in team scenarios. These could be heterogeneous teams, constructed of both humans and agents, working together in some joined task such as a search and rescue scenarios [20]. Or these could be homogeneous teams working in unison as in assembly scenarios [50] and object handovers [96]. Another example is a soccer game scenario where the agents may want to cooperate when on the same team but may be on opposing teams as well [59]. In all of these cases there is a need for cooperation without explicit interaction. In order to do so in an efficient way the agent has to perform some kind of plan recognition to try and infer another agents' intention.

In these complex recognition problems the set of observations is not necessar-

ily complete or sequential; observations can be missing in the beginning, middle or end of the process. The manner in which observations are obtained also plays a significant role. The recognition problem may be divided into two variants. In *offline recognition* the entire sequence of observations is provided to the agent before the recognition process begins. In contrast, in *online recognition* the sequence of observations is revealed incrementally instead of being known in advance, with the object of identifying the goal or plan as early in the recognition process as possible.

The prevalent approach to goal recognition relies on a dedicated *plan recognition library*, which represents all known ways to achieve known goals [99]. As the problem of plan recognition is NP complete, recognition methods vary widely in the expressiveness of the representation and efficiency of the inference algorithms used. While powerful when the plans are known, this does not work when the observations come from an unknown plan to achieve a known goal. An additional difficulty is raised when adding goals to the set of recognizable goals, as plans for them need to be inserted into the library in order to be recognized. Moreover, and perhaps more importantly, the use of a dedicated plan recognition library is not compatible with underlying principles of integrated agents: an agent that plans and acts in an environment will need a separate plan recognition library for recognizing other agents' actions, despite having implicit knowledge about what plans in the environment look like.

In addition, current approaches to plan recognition have focused on discrete descriptions of the agent's interactions with its environment. Continuous domains were traditionally addressed by a separate discretization component, translating angles, positions, motions—sometimes entire trajectories—into discrete symbols. Unfortunately, early commitment to a fixed discretization leads to inherent information loss. We show that once discretization is fixed, there are always cases where the information loss will degrade performance.

Inspired by mirroring processes in primates, we have developed *Mirroring* for plan recognition. It has been hypothesized that the human ability to perform quick and efficient plan recognition comes from the newly discovered *mirror neuron system* for matching the observation and execution of actions within the adult human brain [85]. The mirror neuron system gives humans the ability to infer the

intentions leading to an observed action using their own internal mechanism. It may therefore be viewed as a part of the brains' own plan recognition module and can be used to recognize the actions and goals of one or more agents from a series of observations of the other agents' actions.

Mirroring is a plan recognition technique which relies on a model of planning that extends classical planning to model domains with continuous and/or discrete variables in an efficient *online* manner. By utilizing a planner as a black box, to dynamically generate plans that are matched against the observations, eliminating or ranking recognition hypotheses throughout the process, *Mirroring* eliminates the need for storing plans in advance through the traditional plan library. It is designed to efficiently handle incremental, continuous observations and tightly integrates planning and recognition: Whatever plan can be planned, can also be recognized.

Other notable exceptions that also focus on library-free recognition include [80, 81, 57, 94]. However, these inspiring approaches target discrete domains only, and are inefficient for online recognition, where observations are incrementally revealed. They would have additional difficulty in operating in continuous domains which have infinitely parameterized actions (observations).

1.1 Mirroring Components

To carry out this recognition process, in particular re-using an off-the-shelf planner in service of recognition, several components are needed. These are shown in Figure 1.1, and described below in detail.

Figure 1.1 shows the input of the recognition process. The first is of the observations of the agent whose possible goal the system is attempting to recognize, and the second is the set of possible goals (normally given once, but can dynamically change). The nature of these observations may differ between varying recognition problems and we may assume, without loss of generality, that these observations include the initial position of the agent as it may easily be obtained from them. When attempting to recognize a destination, the observations may represent locations in space, when attempting to recognize shapes drawn on paper,

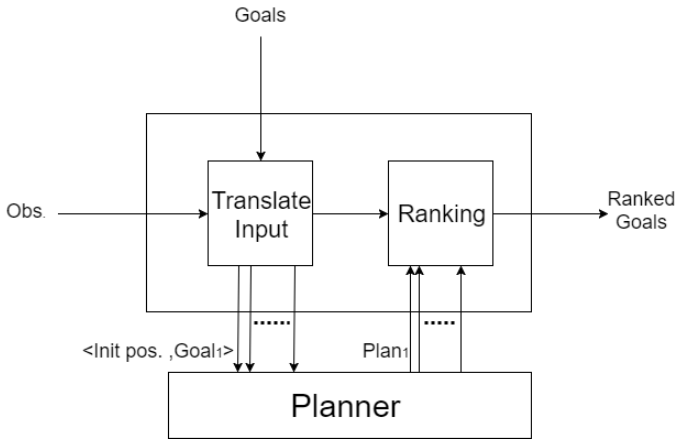


Figure 1.1: Components needed for planner based recognition.

the observations may represent possible sketches already drawn; we will delve deeper into both of these examples in the Chapter 5.

Translate Input The goal and observations are fed as input into the *Translate Input* component, whose task is to prepare the input to be sent to the *Planner*. This is a key step within any recognition approach whereby a planner is being used. Since planners are not designed to accept observational history as input, part of the work of the recognizer, before utilizing the planner, is to incorporate the history of the observations in a coherent manner as part of the planners' input.

With each new observation the *Translate Input* component iterates over all possible goals, folding the known observations into the input fed to the planner by updating either the *Initial Position* or the *Goal*. As there is no standard language for planning, different planners will require slightly different input preparation. Because the planners used are *off-the-shelf* planners, incorporating information from the observations as input to the planners is not a trivial task. *Mirroring* achieves this by utilizing the planner to only calculate part of the plan, excluding the part already achieved as seen in the observations. In contrast, Ramírez and Geffner [81] achieved this by transforming the domain theory to incorporate observations as new fluents that must take place as additional preconditions.

Planner After the observations have been translated, they are sent to the *Planner* as either part of the *initial position* or the *goal*. The output of the planner is a complete plan to achieve the goal starting at the initial position given. If the planner is unable to generate a plan it issues an error which indicates that the plan is not achievable given the input, and the corresponding goal will eventually be ranked at the bottom.

Thus taken together, the *Translate Input* and the *Planner* components work essentially as a generate-and-test process. The *Translate Input* component sets up possible hypotheses, and the *Planner* tests them, returning a plan to indicate that the hypothesis passed, or error (no plan) to indicate that the hypothesis should be discarded. This process may be called once for each goal , repeatedly for each goal ([94]) or just once ([108],[81]) depending on the implementation.

The end result of this process is a set (thus, unordered) of hypotheses that match the observations thus far, generated without relying on a stored set of examples, or plans. This set may be analyzed in various ways, in order of likelihood [81] or relevance [100].

Ranking Recognition Hypotheses There could be potentially endless ways to rank the possibly hypotheses generated by the planner. One way of determining a ranking order over the set of recognition hypotheses is to rank them based on errors when compared to the ideal plans for reaching each goal from the initial position. In fact the *Mirroring Ranking* component compares each plan with the ideal plan. The goal which displayed the smallest difference between these two plans will be ranked highest. We will elaborate more on this ranking procedure during in Chapter 3.

We have extensively evaluated this approach over hundreds of experiments utilizing several different planners, both *off-the-shelf* and domain specific. We examined different instances of each individual component and measured recognition success in terms of several impacting factors. We experimented with multiple domains, both continuous and discrete and have further contrasted the recognition results with those achievable with library-based methods. Finally we com-

pared the performance of *Mirroring* to human subject recognition while drawing lessons with respect to human recognition capabilities and the information needed in order to replicate them.

1.2 Thesis Overview

This dissertation comprises 11 chapters, organized into three main parts (see Figure 1.2). This chapter constitutes the introduction to this dissertation. The next chapter surveys the related work. Chapters 3– 5 constitute Part 1 of the dissertation, which deals with a thorough description and implementation of the baseline *Mirroring* algorithms for online and offline recognition. Chapters 6– 7 constitute Part 2 of the dissertation and address the cognitive inspiration for this work by comparing the performance of Mirroring to that of human recognition. Chapters 8– 10 constitute Part 3 of the dissertation and deal with adapting *Mirroring* to apply to *efficient* online recognition. And finally, in Chapter 11, we provide our conclusions and discuss future work.

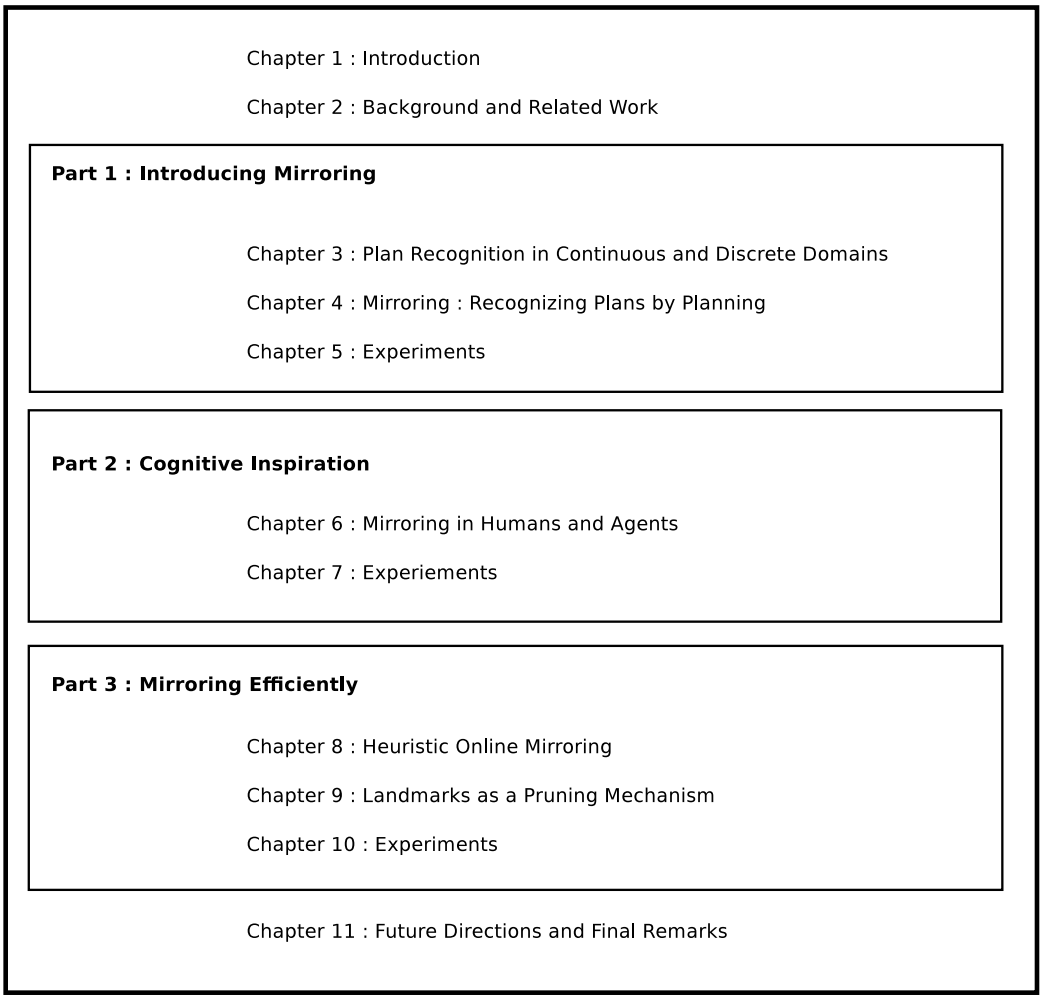


Figure 1.2: Dissertation Structure.

1.3 Publications

Results that appear in this dissertation have been published in the proceedings of the following refereed journals, conferences, books and workshops:

- Kaminka, Gal A., **Vered, M.** and Agmon N. “Plan Recognition in Continuous Domains.” To be published in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

- **Vered, M.**, and Kaminka, Gal A. ”Heuristic Online Goal Recognition in Continuous Domains.” To be published in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017 [107].

- **Vered, M.**, and Kaminka, Gal A. ”Online Goal Recognition Combining Landmarks and Planning.” *Goal Reasoning workshop IJCAI-17*. 2017.

- **Vered, M.**, Kaminka, Gal A. and Biham S. ”Online Goal Recognition through Mirroring: Humans and Agents.” In *Proceedings of the Annual Conference on Advances in Cognitive Systems (ACS-16)*, 2016 [108].

- **Vered, M.**, and Kaminka, Gal A. ”Towards Sketch Recognition By Mirroring : Extended Abstract.” In *Proceedings of the 14th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, 2015 [106].

2 Background and Related Work

> ”Don’t become a mere recorder of facts, but try to penetrate the mystery of their origin.”
>
> *Ivan Pavlov*

The problem of goal recognition; recognizing a goal or intention without complete knowledge, has many applications for everyday real life scenarios, including human-robot and human-computer interaction [110], command prediction [52, 10, 11], smart environments [112], intelligent learning environments [15, 3, 103], monitoring user needs [74, 111], recognizing navigation goals [55, 113], recognizing intended gestures and sketches [86, 92] and many more.

The majority of prevalent approaches rely on a dedicated *plan library* as the basis for the recognition process. The plan library efficiently represents all known plans to achieve known goals. In this manner the observations are matched against existing plans to determine the most likely plan candidate (see Sukthankar et al. for an extensive survey of recent work [99]). Methods vary in the representation and inference algorithms used: action decomposition graphs [48, 6], Bayesian networks [21, 1], hidden Markov model variants [16, 10], conditional random fields [55, 104, 54, 40], grammar-based approaches [78, 35, 33, 62, 63], case-based plan recognition [105] and many more. A common theme is that they address continuous domains only through fixed, a priori discretization. We will show in Chapter 3 that this approach may lead to inferior recognition success.

The use of a library, while often efficient, limits recognition capabilities to recognizing goals for which plans are known a-priori and encoded in the plan

library. Additionally, the requirement to store recognition knowledge separately from plan execution knowledge wastes space. If the observations are of an unknown plan, even leading to a known goal, traditional pure library-based methods fail. And when wishing to add to the set of recognizable goals, we would also have to insert plans for recognizing the new goals into the library (e.g., manually or by learning), in order for the new goals to be recognized. Many such methods include a variety of probabilistic inference techniques [16, 104, 7, 54]. Cox et al. [25], present a library-base technique that attempts to handle novel plans. They perform plan retrieval based on similarities among concrete planning situations. However, the method uses a representation that is inappropriate for continuous environments where actions and ensuing states are not discrete.

Nonetheless, by an inefficient re-running of the offline algorithm, most of these methods can be used for both online recognition, where the observations are incrementally revealed and offline recognition, whereby the observations while incomplete by themselves are given a-priori to the recognition process.

Some methods attempt to modify the plan library itself by unifying the plan-recognition and plan-execution libraries: *Agent tracking* [100, 51] uses an agent's own BDI plan to recognize a BDI plan being executed by another. Similarly, Sadeghipour et al. [87, 88] represent (and store) shape drawing *plans*, that can be used both for recognition and execution by the agent. Most recently, Geib et al. [34] advocated the use of combinatoric categorical grammars as a representation for both generating and recognizing plans.

Various other approaches to plan recognition have also been taken. Instead of a plan library these approaches utilize a *domain theory* in the recognition process. Plan recognition based on domain-theories removes the reliance on a plan-library, assuming that any valid sequence of actions is a possible plan and using the domain description in the recognition process. Some domain-theory methods address online recognition. Early seminal work by Hong [39] uses a *goal graph* representation for online goal recognition, constructed from a domain theory and incoming observations; recognized goals are not probabilistically ranked. In contrast, Baker et al. [8] use a bayesian framework to calculate goal likelihoods by marginalizing over possible actions and generating state transitions using only limited replanning. Martin et al. [56] take an extreme approach, using signifi-

cant offline computation to eliminate all online planner calls by pre-computing cost estimates. Keren et al. [49] investigates changing the domain to facilitate the goal-recognition process. All of these approaches work with discrete domain theories, and do not directly translate to continuous domains.

Mirroring is most closely related to *plan recognition by planning* (PRP) [81, 80]. A plan recognition approach which avoids storing a plan-library, but instead uses domain theories with planners to dynamically generate hypotheses for plan recognition on the fly. Here, an unmodified planner is used as a black box to generate recognition hypotheses that match the observations. A heuristic comparison between the generated plans and an optimal plan that ignores the observations is used to probabilistically rank the hypotheses.

The Ramirez and Geffner formulation[80] relied on modified planners and could not probabilistically rank the hypotheses. The extended formulation [81] uses *off-the-shelf* (OTS) classical planners to probabilistically rank goals. However this formulation is inherently limited to discrete domains, as it requires computing a *optimal* plan that *necessarily deviates* from the observations. This requirement is meaningless in continuous domains, as any small ϵ deviation from an optimal plan that matches the observations would fulfill this requirement, at the expense of the ranking procedure used in this PRP formulation.

The use of PRP in continuous domains, and in an online fashion also raises new challenges. The original formulation [81], relies on synthesizing two optimal plans for every goal: (i) a plan to reach a goal in a manner compatible with the observations ; and (ii) a plan to reach a goal while (*at least partially*) deviating from the observations. The likelihood is then computed for each goal from the difference in *costs* of optimal solutions to the two plans. Overall, $2|G|$ planning problems are solved, two for each goal. In *online* recognition the set of observations is incrementally revealed. Thus two new planning problems are solved with *every new observation*, for a total of $2|G||O|$ calls to the planner instead of $2|G|$. In addition, using an off-the-shelf continuous-space planner to generate a plan that may *partially* go through previous observations, but must not go through all of them, is currently impossible given the state of the art.

Sohrabi et al. have further improved this approach to better address unreliable observations and recognize plans as well as goals [95] and Masters et al. have

improved the efficiency of the existing approach by reducing overall plan computation within the context of path planning [57]. However these approaches are targeted towards *offline* goal recognition and are inefficient in online recognition, where observations are incrementally revealed. Moreover, their formulation is for discrete worlds, while we focus on continuous worlds.

Other investigations have begun to address the inefficiencies of online PRP recognition. Periera et al. have taken a more efficient offline approach [71, 73] which avoids planning altogether, instead generating planning landmarks from the domain theory prior to recognition. Such landmarks are actions (or state properties) that *must* be included in plans that achieve specific goals [38], and thus provide strong evidence for recognizing these goals. Indeed we use them in Chapter 9 where our results show that offline-computed information can be beneficial, but the question of the expense of the offline computation versus online use remains open.. We note that all of these methods work offline and in discrete domains.

Martin et al. also refrain from using a planner in the PRP. Instead, for each goal they compute cost estimates using a plan graph [56]. This approach is complementary to ours. Ramirez and Geffner [82] precompute a policy for recognition, thus trade significant offline computation for faster online responses. We avoid this tradeoff, focusing instead on online run-time improvements.

Inspired by these investigations, *Mirroring* uses a planner (as a black box) to generate plan and goal hypotheses on-the-fly [108, 107]. Unlike previous work *Mirroring* is uniquely addressed towards *online* recognition in both continuous and discrete domains while including heuristics that aim to improve efficiency.

Part I

Introducing Mirroring

3 Plan Recognition in Continuous and Discrete Domains

> "Life can show up no other way than that way in which you perceive it."
>
> *Neale Donald Walsch*

To date, PAIR approaches have focused on discrete descriptions of the agent's interactions with its environment, via plan libraries. Continuous domains were traditionally addressed by a separate discretization component, translating angles, positions, motions—sometimes entire trajectories—into discrete symbols. This facilitates the use of powerful algorithms that use a variety of recognition algorithms that utilize plan libraries: hierarchical graphs [48, 6], deterministic and probabilistic grammars [78, 33, 87, 62, 34, 7, 35], and other probabilistic models [21, 16, 77, 82].

Unfortunately, early commitment to a fixed discretization within the plan library leads to inherent information loss. Indeed, we will later empirically show that once discretization is fixed, there are always cases where the information loss will degrade performance. This has also been shown in related tasks (such as path planning) where there too the loss of information inherent in discretization can be detrimental [64].

One approach to avoid such early commitment is to apply PAIR methods based on domain theories, rather than plan libraries. Domain theories describe states in the world and actions that transition between states and can be used more generally and flexibly, to identify goals and plans that are not specified. Such methods

dynamically generate recognition hypotheses *as needed*, thus in principle avoid the early commitment made in plan libraries. In particular, *Plan Recognition as Planning* (PRP) [81, 106, 108, 95, 107, 57], uses off-the-shelf (OTS) planners for recognition. Thus potentially, using a motion planner such as RRT* [65] in PRP could lead to successful, straightforward recognition in continuous spaces. However, current PRP methods are limited to discrete classical-planning domains, and cannot use OTS *motion* planners.

3.1 The Discretization Problem

One of the common ways of translating a motion navigation problem in a continuous domain to fit a discrete model is by discretizing the environment into a regular grid. This is done by overlaying the grid on top of the existing environment and thereby generating a grid world formed of cells. A graph can then be generated by placing vertices either in the corners or centers of each grid but *only* in those locations within the cell. Any location within these cells will be translated to one of these vertices and any path from one cell to another will have to pass along them. Nash et al. [64] have shown that the shortest grid paths can be substantially longer than the true shortest path possible in square grids and have shown that this expands to triangular and hexagonal grids as well. We would like to demonstrate how this could lead to an impossibility of goal recognition.

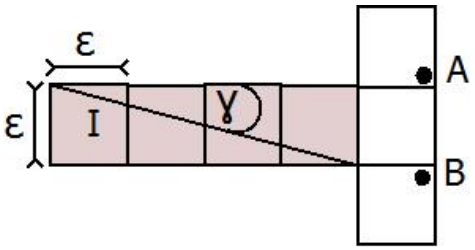


Figure 3.1: Discretized goal recognition.

Here's an intuitive example. Figure 3.1 shows an instance of a 2D recognition problem in a continuous environment that underwent the aforementioned discretization process of dividing the world into a grid. In this instance we used a square grid, which is the most common and simplest type of grid to implement. The initial agent position is represented by the letter *I* while letters *A* and *B* represent positions of possible goal hypotheses. The actual trajectory of the agent is represented by a dark, straight line from the initial position in the top-left corner,

to goal B, while the corresponding observed grid trajectory is highlighted by gray grid blocks. Thus observations are of grid cells of size $\varepsilon \times \varepsilon$, rather than of the actual trajectory.

As can easily be seen the actual trajectory of the agent displays behavior that clearly favors goal B (under the assumption of a rational agent pursuing the shortest path between two points) and could potentially lead to early recognition. However the discretized observation sequence of grid cells, highlighted in gray, does not convey this information; with this representation the goal can be recognized only when the agent moves into a grid cell containing either A or B up to which point the probability for either of the goals would be equal. In this instance the existing discrete grid model representation falls short within the problem of goal recognition.

In a continuous environment the initial state of an agent $I \in S$ where $S \subseteq \mathbb{R}^n$ necessarily includes not only positions but headings and angles in each position. In this way the tendency of the trajectory could be observed as early as after the first action and taken into account in the goal ranking analysis. However, even if the angle and/or heading of the agent was also discretized there would still be some discretization factor ε under which this representation would fail.

Therefore, we would argue that this can hold for any discretization factor (specifically in our example; grid size). If the recognizer could set ε ad-hoc, it could potentially recognize the goal earlier. Note that this is true for any given ε-grid (Thm. 1), and clearly analogous examples can be created for non-grid a priori discretizations.

To show this we first provide several necessary definitions. We define the recognizer as a mathematical mapping of observations to goal distributions, and we define the term *distinguishable* between goals. Two goals will be distinguishable over a certain observation sequence (which excludes the goal state itself) if the probability to achieve each goal is different, ultimately allowing for different goal rankings. More formally :

Definition 1. *We define a* C-Optimal *plan $P_o : I \rightarrow G$, as a plan from an initial state I to goal state G that is optimal with respect to value under criterion C, $C(P_o)$.*

Definition 2. *We define a* recognizer *as a function that maps an observation sequence, $O \subseteq P_o$, to a probability distribution over the goal set G, recognizer* : $O \rightarrow Pr[G]$.

Definition 3. *A* C-Optimal *recognizer is a recognizer that maps $O \subseteq P_o$ to a probability distribution $P[G]$ such that*
if $C(P_i : I \rightarrow g_i) < C(P_j : I \rightarrow g_j)$, $(g_i, g_j) \in G$,without loss of generality and P_i and P_j agree with O
then the recognizer will output probabilities such that $Pr(g_i) < Pr(g_j)$.

A C-Optimal recognizer will necessarily prefer more direct plans with lower costs to achieve possible goals, a phenomenon also found in human judgment of observations [13].

Definition 4. *In recognition problem R, two goals $(A,B) \in G$ are* distinguishable *if for an existing observation sequence $O \subset P \setminus G$, $Pr(A|O) \neq Pr(B|O)$. Similarly, $(A,B) \in G$ are indistinguishable if for every observation sequence $O \subset P \setminus G$, $Pr(A|O) = Pr(B|O)$.*

Note that the definition of *distinguishable* purposefully excludes the final goal state from the observation sequence $O \subset P \setminus G$. In a problem of goal recognition it would be trivial to attempt to identify a goal from an observation sequence that includes the final goal state. We therefore investigate the problem of goal recognition where the final state is yet to be seen and there is merit in early recognition. We can now proceed to prove the power of the continuous representation over the discrete representation.

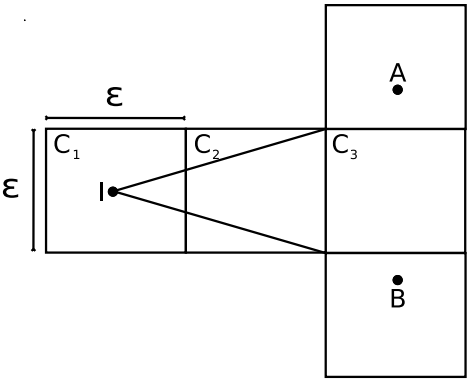


Figure 3.2: Discretization limitation example.

Theorem 1. *For every grid cell size ε there exists a goal recognition problem R such that goals in R are indistinguishable in the discrete domain, yet distinguishable in the continuous domain.*

In order to prove the Theorem let us first look at Figure 3.2 and prove two supplementary lemmas.

Lemma 2. *There exists a goal recognition problem R such that in its discrete representation the goals in R are indistinguishable for every observation sequence* $O \subset P \setminus G$.

Proof. Consider Figure 3.2. For any *fixed* ε, there exists a goal recognition problem R where there are two possible goals—equally likely—in locations A and B and the initial position is I.

For each C-Optimal observation sequence $O_c \subset P \setminus G$ in the continuous world, we can construct a corresponding C-Optimal observation sequence $O_d \subset P \setminus G$ under discrete representation. These sequences are a subset of a C-Optimal plan and therefore will take the optimal, direct path to achieving each goal. Necessarily under discrete representation all of these sequences will prove identical and have to pass through cells C_1 and C_2, $\Rightarrow O_d \subseteq \{C_1, C_2, C_3\}$.

As the recognizer is defined to be a consistent function, given the same input, O_d, we will receive the same distribution over all of the goals for each possible observation sequence , $recognizer(O_d) = Pr[G]$.

Hence, for every separate C-Optimal observation sequence under continuous representation, from I to each $g \in G$, $O_c \subset P \setminus G$, the discrete representation will give us the same rankings over the goals, $Pr[G]$. Within this probability distribution $Pr[G]$ we have two possibilities; either $Pr(A|O) \neq Pr(B|O)$ or $Pr(A|O) = Pr(B|O)$.

If $Pr(A|O) = Pr(B|O)$ the lemma holds and A and B indistinguishable for every $O_d \subset P \setminus G$

Let us assume for contradiction that $Pr(A|O) \neq Pr(B|O)$. This means that for every possible observation sequences, the C-Optimal recognizer will always prefer the same goal over the other implicitly stating that $C(P : I \to A) < C(P : I \to B)$, without loss of generality. This cannot be true because the observation sequences are C-Optimal with regards to each different goal. Therefore the probabilities must be equal $Pr(g_i) = Pr(g_j)$ making the goals A and B indistinguishable for every $O_d \subset P \setminus G$.

In consequence there exists an R in the discrete domain, such that $\forall O \subset P \setminus G$ the goals in G are indistinguishable .

□

Lemma 3. *In the continuous domain, the goals in R are distinguishable.*

Proof. Consider again the example in Figure 3.2. In the continuous domain there are two observation sequences $O_c \subset P \setminus G$ that are subsets of C-Optimal plans, one to goal A and the other to goal B.

We need to show that there exists a C-Optimal recognizer for which $Pr(A|O_1) \neq Pr(B|O_2)$ for each coordinate along these observation sequences excluding I. Consider the function l_2 which represents the Minkowski distance metric corresponding to the Euclidean Distance function [60]. For each coordinate along O_c the function measures the geometric distance to the goal. The closer the goal the higher it will be ranked.

Each observation sequence $O_c \subset P \setminus G$ that excludes I is part of a C-Optimal plan to either A or B. Therefore it takes the most direct route leading to the goal and will necessarily be geometrically closer to one of the goals. As the function l_2 measures geometric distances from each possible observation sequence to the goals, it will rank one goal over the other. By using l_2, $Pr(A|O_1) \neq Pr(B|O_2)$ and A and B are distinguishable in the continuous domain.

□

Lemma 2 has shown that there exists a goal recognition problem R such that in its discrete representation the goals in R are indistinguishable for every observation sequence $O \subset P \setminus G$. Lemma 3 has shown that in that same goal recognition problem the goals are distinguishable in the continuous domain. Therefore for grid cell size ε there exists a goal recognition problem R such that the goals in R are indistinguishable in the discrete domain, yet distinguishable in the continuous domain and Theorem 1 follows straight forward from the lemmas.

Recognition (PAIR) approaches which rely on a domain-theory, rather than a plan library offer an opportunity to avoid the early discretization commitment, as they generate recognition hypotheses dynamically, ad-hoc. Most of these approaches work with discrete domain theories and therefore cannot be directly

translated to continuous domains [39, 56, 70]. However, PRP based approaches which use domain theories with planners to dynamically generate hypotheses for plan recognition may pose a solution potential [80, 81]. The [81] formulation uses *off-the-shelf* (OTS) classical planners to probabilistically rank goals. However this formulation is inherently limited to discrete domains, as it requires computing an *optimal* plan that *necessarily deviates* from the observations. This requirement is meaningless in continuous domains, as any small ε deviation from an optimal plan that matches the observations would fulfill this requirement, at the expense of the ranking procedure used in this PRP formulation.

Mirroring relies on a model of planning that extends classical planning to model domains with continuous and/or discrete variables. This is done in order to admit a broad class of motion planners, e.g., from the OMPL library [98]. As a result, the model does not offer facilities for explicitly representing more advanced planning models and languages, e.g., supporting conditional effects, inequality tests in preconditions of actions, sensing actions, etc. In contrast, some modern task planners utilizing PDDL 2.1 and above [32] admit such advanced features, while also allow planning in mixed continuous-discrete domains. Investigations of planning models allowing mixed domains are on-going (see, e.g., [36, 53, 23, 30, 90, 41]. Of these, our model is closest to the latter, and it also borrows some of the constraints of Transitional Normal Form for planning [75]. We leave recognition in mixed domains, and non-classical planning extensions to future work.

3.2 Domains

We define *domains* which are collections of states. *Discrete actions* transition between and through the states. The sequence of such transitions being measurable by a cost metric. We then define *plans* in these domains, where such plans are sequences of actions that take the plan-executing agent from a (partially-) specified initial state to a (partially-) specified *goal state*.

Domains: States, Actions, and Transitions. In the rich tradition of factored representations in planning, a *domain* W is defined as a tuple $\langle F, V, A, cost \rangle$, where F is a *finite* set of *fluents* (described below), V is a set of sets $\{V_f | f \in F\}$ (each set V_f holds the range of values potentially associated with f), and A is a *discrete,*

possibly infinite set of *actions*, which encode feasible transitions between states, (i.e., transform values of fluents). *cost* is a metric, allowing such transformations to be measured.

We allow describing states via numeric-valued fluents, somewhat similarly to [41]. Actions in A may transition from one state to another via *paths* through the state space, rather than through discrete state as in classical planning. This is because in continuous state spaces, a transition from a state (point) a to a state b may go through countless other states in between.

FLUENTS AND STATES. A fluent $f(e_0,...,e_n) \in F$ is an expression where f is the *fluent name*, $n \in \mathbb{N}$ (inc. 0) the *fluent arity*, and e_i are constant *entities* in a known set. For brevity, we often refer to the fluent by its name. A *fluent literal* (for brevity: a literal)is a pairing of a fluent and a specific value $v \in V_f$, the *fluent range*. We denote a literal as $f = v$. For our purpose here, fluents in F may have a boolean value, or they may have numeric values (e.g., in $\mathbb{R}$).

Fluents in F are used as the basis for describing states. A set of fluent literals is *inconsistent* if it contains at least two literals $f = v_1, f = v_2$, where $v_1 \neq v_2$. Otherwise, the set is *consistent*. A state s induced from F is *a maximal consistent set of fluents literals*. Put differently, a state s is a set of fluents from F, each paired with a value from its associated V_f, such that: $|s| = |F|$, and s is consistent. A non-maximal consistent set of fluent literals is a *partial state*, and may be used to formally collect all states of which it is a subset.

For example, the pose (position and orientation) of a robot r on a 2D floor may be described by the fluent set

$$F \triangleq \{x(r), y(r), \theta(r)\}$$

where r is a constant symbol for the robot. A set of fluent literals $s = \{x(r) = 50.34, y(r) = 24.0, \theta(r) = 90^\circ\}$ is a state: it is consistent, and assigns a value to all fluents in F. However, the set $s_1 := s \cup \{y(r) = 32.45\}$ is not consistent (two different values for $y(r)$), and the set $s_2 := s \setminus \{\theta(r) = 90^\circ\}$ is a *partial* state (describes all states where the robot is in location $(50.34, 24.0)$, regardless of θ's value). The set of all possible literals of $f \in F$ is $L_f := \{f\} \times V_f$. The complete set of states in W is: $S_W := \bigtimes_{f \in F} L_f$.

ACTIONS. An action $a \in A$ transforms fluent literals, changing their values. We define $\texttt{PRE}_a$, the preconditions of a, as a set of fluent literals. a is *applicable* in a state s when $\texttt{PRE}_a \subseteq s$. The results of applying an applicable action a in s, are specified by a function $\delta(s,a)$.

In discrete domains, the function $\delta(s,a)$ yields a single new state s_{new}. To generate this new state, classical planners typically rely on two additional sets of fluent literals associated with every action a: $\texttt{ADD}_a$, a set of literals to be added to s, and $\texttt{DEL}_a$, a set of literals to be deleted. Then $\delta(s,a) := (s \setminus \texttt{DEL}_a) \cup \texttt{ADD}_a$.

However, motion planning in continuous domains raises two challenges to this. Imagine a navigation motion planner in 2D/3D which outputs a path (i.e., a plan) from an initial position I to a goal position g. Often, motion planners represent such plans by an ordered finite sequence of *waypoints* $(I, s_1, \ldots, s_k, g)$. Each transition from one waypoint to the next, itself defining a path between the waypoints, is assumed to be managed by the robot, and would be considered an atomic MOVETO action (we ignore smoothing and other constraints for simplicity). Waypoints discretize the domain by sampling, but the sampling decision is made ad-hoc, e.g., the number of waypoints can vary even given the same pair I, g. This reality of how motion plans are represented—even for this simplified case—raises two challenges to modeling this process in a manner compatible with classical planning formulations.

First, motion actions almost invariably go through other states of the domain as they are applied. A robot moving from waypoint A to waypoint B, regardless of how close they are, necessarily moves through infinite points that lie in between. Thus the effects of an action taken in a continuous domain is not just the ending state, but an $|F|$-dimensional path defined by the ordered (potentially infinite) sequence of states.

Second, as motion planners delay their discretization, they do not accept a finite set of actions A. The position and number of waypoints varies depending on factors such as a required minimal refinement, constraints on the moving body, time available, the sampling algorithm of the planner, etc. (see the OMPL website for a large sample of different planners which explore such decisions). This means that we cannot model them as accepting a finite set A, which commits to a fixed discretization. Instead, we model them as choosing actions from an infinite,

discrete set of actions A. In reality, of course, they do so implicitly, by generating the discretization which implies choosing the actions, e.g., transitioning between waypoints.

To model actions and plans in continuous domains, we first extend the definition of actions $a \in A$ to allow effects as paths, rather than just points. We use the following notation. A path p is a (possible infinite) ordered sequence of states $(p_0, \ldots, p_m)$, $m \in I \subset \mathbb{N}^+$, the set of indexes. I is obtained by a monotonically increasing mapping $f_a : [0,1] \mapsto \mathbb{N}^+$, representing the relative position of the intermediate p_i along the path from p_0 to p_m. By definition, p_0 is in relative position 0 ($f_a(0) := 0$) and similarly $f_a(1) = m$. Otherwise, and $0 < f_a(0 < i < 1) < m$. Thus p_i is the i'th state in p, given the indexes I, generated by the mapping f_a. We notate $p_i \leq p_j$ when $i \leq j$, i.e., p_i is earlier in the sequence defining p. $\texttt{BEG}(p)$ is p_0, and $\texttt{END}(p)$ is the final state p_m. The concatenation of two paths p, q is denoted by the operator $\oplus$, such that $r := p \oplus q$ is a path with $\texttt{BEG}(r) = \texttt{BEG}(p)$, $\texttt{END}(r) = \texttt{END}(q)$. If $\texttt{END}(p) \neq \texttt{BEG}(q)$ then r is called *partial*.

Using this notation, we now redefine $\delta(s,a)$ as returning a path p, with $\texttt{BEG}(p) = s$, $\texttt{END}(p) = s_{new}$. The index-generating mapping is not necessarily given to the planner. We require that it exists, but it is possible for the planner to determine it ad-hoc. This allows generating the intermediate states (between s and s_{new}) in any desired granularity.

Given a specific set of indexes I (i.e., a specific granularity), we generate $\delta(s,a) := \bigoplus_{i \in I}((s_{i-1} \setminus \texttt{DEL}_a(s_i)) \cup \texttt{ADD}_a(s_i))$ where $\texttt{DEL}_a(s_i)$ and $\texttt{ADD}_a(s_i)$ are sets of fluent literals as described above, representing the intermediate delete and add effects (respectively) between state s_{i-1} into state s_i. Note that the discrete representation is a special case. Setting $I = \{0,1\}$, yields $\delta(s,a) = (s_0 \setminus \texttt{DEL}_a(s_{new})) \cup \texttt{ADD}_a(s_{new})$.

Plans are sequences of actions. Specifically, a plan π from initial state s_0 to goal state s_g is a *finite* sequence of actions $(a_1, \ldots, a_k)$, such that: (i) a_1 is applicable in s_0, (ii) each action $a_i, 0 < i \leq k$ is applicable in s_{new}^{i-1}, the final state of its predecessor in the sequence, and (iii) the resulting path

$$p_\pi := \delta(\texttt{END}(\delta(\texttt{END}(\ldots \delta(\texttt{END}(\delta(s_0, a_1)), a_2) \ldots), a_{k-1}), a_k))$$

, has $\texttt{END}(p_\pi) = s_g$. The path p_π is called the *execution trace* of π, denoted TRACE(π). The metric $cost(p)$ associates a non-negative *cost* to a path p, defined such that for any two paths p, q, $cost(p) + cost(q) = cost(p \oplus q)$, even if $p \oplus q$ is partial. We define $cost(\pi) = cost(\text{TRACE}(\pi))$.

3.3 Plan-Recognition Problems

We now define a general recognition problem without reference to a particular objective, nor a solution method (e.g., calling a planner or using a plan library). For brevity, in this and future elaborations, we informally refer to *a state* (or *partial state*) $s \in W$, when we mean $s \in S$ (or $s \subset S$, resp.) where S is the set of all possible states induced by F in $W = \langle F, A, cost \rangle$. Similarly, we informally refer to *actions* or *plans* in W (and may write $a \in W$, $\pi \in W$).

Definition 5 (Recognition Problem)**.** *A recognition problem is a tuple* $R := \langle W, O, I, G \rangle$ *where* W *is a domain theory as defined above,* O *a sequence of observations,* $I \in W$ *an initial state,* G *a set of goals in* W*. Observations and goals are defined below.*

Each goal $g \in G$ is a (possibly partial) state s_g, associated with a prior probability $P(g)$. This definition of a goal essentially defines it as a disjunction of states. s_g defines either a single state (if s_g is not a partial state), or a set of induced states $S_g \subset S$ in W, all of which have the exact same fluent literals as in s_g, and are interpreted as a disjunction: an action resulting in any one of them is considered to have achieved the goal g.

We denote O, the sequence of observations as $[o_0, \ldots, o_n]$, where $n \in \mathbb{N}$, and $o_0 := I$. Every o_i is a state. Thus observations are of effects, not actions. When n is known, R is called an *offline* problem, otherwise R is an *online* problem.

A recognition problem may be used as the basis for different tasks. We define the plan recognition task for *offline* problems:

Definition 6 (Plan Recognition)**.** *Let* R *be an offline recognition problem. The plan recognition task is to determine* π_R*,*

$$\pi_R = \underset{\pi \in W}{\operatorname{argmax}} P(\pi | O)$$

i.e., π_R is the plan hypothesis π with maximal probability, given the observation sequence. Lacking any dependence between the observations and plans in W, deciding on π_R ignores O. The dependence between observations and plans is made explicit by the notion of matching (partial) observations to plans, in particular also taking their goals into account.

Definition 7 (Matching Observations to a Plan)**.** *Let π be a plan, $\sigma = \text{TRACE}(\pi)$. Let O be an observation sequence defined in the recognition problem R. We define the path $p = \bigoplus_{o \in O} o$, i.e., the (partial) path created by concatenating all observations, in order.*

The matching of O to π is a mapping $m_\pi^O : [0,1] \mapsto p \times \sigma$, such that: (i) $m_\pi^O(0) := (\texttt{BEG}(p), \texttt{BEG}(\sigma))$ (ii) $m_\pi^O(1) := (\texttt{END}(p), \sigma_\Omega))$, where $\sigma_\Omega \leq \texttt{END}(\sigma)$, i.e., may not be the last state in σ (iii) $\forall r \in (0,1), \exists i, j$ s.t. $m_\pi^O(r) = (p_i, \sigma_j)$ and $j < \Omega$. (iv) Let $m_\pi^O(r) = (p_i, \sigma_j), m_\pi^O(l) = (p_h, \sigma_k)$, where $r, l \in (0,1)$. If $i < h$ then $j < k$, and $r \leq l$. (v) Let $m_\pi^O(r) = (p_i, \sigma_j), m_\pi^O(l) = (p_h, \sigma_k)$, where $r, l \in (0,1)$. If $j < k$ then $i < h$, and $r \leq l$.

This defines a matching such that the first state in the first observation is matched to the beginning of the plan π, and the end of the final observation to an arbitrary final state σ_Ω. This agrees with our notion that observations are typically partial, at least in that they often do not include π's final goal state. The last two conditions dictate a dual-sided monotonicity of the matching.

In general, potentially infinite matchings exist. The key is to determine a plan π whose matching with the observations maximizes $P(\pi|O)$. To do this, we consider the goal of the plan. Let π_g denote a plan $\pi \in W$ with the goal $g \in G$. Under the assumption that the observed agent is pursuing a single goal $g \in G$ (thus $P(\pi_{q \neq g}|g) = 0$), we use Bayes rules to compute

$$\begin{aligned} P(\pi|O) &= \beta P(O|\pi) P(\pi) \\ &= \beta P(O|\pi) P(\pi|g) P(g) \end{aligned}$$

$P(g)$ is given in R. β is a normalizer depending on $P(O)$ only. Maximizing this expression therefore entails maximizing $P(O|\pi)$ while *also* maximizing $P(\pi|g)$. We utilize two principles in this process.

The first principle is the *principle of rationality*. Following [80] and others, assuming it is pursuing a goal g, the observed agent is assumed to prefer cheaper plans π_g. Thus the closer a plan π is to an *optimal* plan $\hat{\pi}_g$ for a goal g, the more we should increase $P(\pi|g)$ [80, Theorem 7]. Rather than matching the actual plans to test for equality, we use their costs:

$$\forall g \in G, P(\pi|g) := \frac{cost(\hat{\pi}_g)}{cost(\pi)}$$

As $cost(\hat{\pi}_g)$ is minimal ($\hat{\pi}_g$ is optimal), $P(\pi|g)$ is well defined probability function, equal to 1 only when the observed plan is optimal, otherwise between 0 and 1.

The second principle is used to maximize $P(O|\pi)$. This is the heart of the matching between observations and a plan. We want to determine an optimal matching, as one that minimizes some matching error metric. Intuitively, if O completely overlaps with π then the matching error should be minimized, and $P(O|\pi)$ maximized.

To do this, we define a state-distance *metric* E over the matching m (Definition 7). For any $r \in [0,1]$, we have $m(r) = (p_i, \sigma_j)$. Then: $E(m(r)) := E(p_i, \sigma_j)$. We use the Euclidean distance (norm). Intuitively, E measures the error in any single point in the matching. We then use it over the entire matching to determine the matching error between the observations and the plan π.

Definition 8 (Matching Error and Best Matching)**.** *Given a plan π, an observation sequence O, and a matching m_π^O between them (Definition 7), the matching error is given by*

$$error(m_\pi^O) = \int_{r\in[0,1]} E(m_\pi^O(r))$$

The best matching $\widehat{m_\pi^O}$ is given by

$$\widehat{m_\pi^O} = \underset{m_\pi^O}{\operatorname{argmin}}\, error(m_\pi^O)$$

The matching error of the best matching for O, π is therefore $error(\widehat{m_\pi^O})$ We then estimate $P(O|\pi) := 1$ if $error(\widehat{m_\pi^O}) = 0$. Otherwise, $P(O|\pi) := \frac{1}{1+error(\widehat{m_\pi^O})}$.

3.4 Defining Continuous Environments

In order to define the goal recognition problem R we define P, a general solver for the continuous space which will solve the plan recognition problem for every $T_G = \langle P_G, G_G, O_G \rangle$. Whereby P_G is the (discrete) planning domain , which is composed of the triplet $\langle F, I, A \rangle$ (fluents, initial state and actions, respectively), G_G are possible goals, and O_G is a discrete observation sequence (as defined in [81] by Ramirez and Geffner). Indeed, we show :

Lemma 4. *$T_G = \langle P_G, G_G, O_G \rangle \subseteq R = \langle W, O, I, G \rangle$, i.e., a solver for R will solve the problem T_G.*

Proof. We prove the lemma by showing that T_G is a special case of R, i.e., every problem in T_G can be represented as a problem in R. According to [81] $P_G = \langle F, I_G, A_G \rangle$ where F is a set of fluents, $I_G \subseteq F$ and A_G is a set of actions. In our definition $W = \langle F, A, cost \rangle$, thus $R = \langle \langle F, A, cost \rangle, O, I, G \rangle$. $F \subseteq \mathbb{R}^n$ represents all possible states of the agent, which is equivalent in both definitions, thus it follows directly that $I \equiv I$, $G \equiv G_G$, $O_G \subsetneq O$. A_G and A represent the possible sets of actions, where each $a^g \in A_G$, transferring from state s_i to s_j, corresponds to $a \in A$, where the outcome of a includes a (possibly infinite) sequence of states, which in the discrete case (as in A_g) includes only one state - $\{s_j\}$.

□

It seems the inverse direction may be true as well, through a process of discretization. The general argument is that every continuous domain may be discretized, and the discrete formulations of PRP may be applied in it. This is incorrect. Inherently, there will be cases where a direct formulation of continuous PRP will be better.

We have now extended a classical planning model to continuous domains. Next we will define plan recognition problems using this model and present the Mirroring solution method.

4 Mirroring : Recognizing Plans by Planning

”It’s not what you look at that matters, it’s what you see.”

Henry David Thoreau

As previously mentioned existing PRP methods prove inefficient for online goal recognition in which they rely on synthesizing two optimal plans for every goal $g \in G$: (i) a plan to reach goal g in a manner compatible with the observations O; and (ii) a plan to reach goal g while (*at least partially*) deviating from O, i.e. complying with $\overline{O}$. The likelihood $Pr(g|O)$ is then computed for each $g \in G$ from $\Delta(g,O)$, the difference in *costs* of optimal solutions to the two plans. Overall, $2|G|$ planning problems are solved, two for each goal. In *online* recognition the set O is incrementally revealed, and $\overline{O}$ changes with it. Thus two new planning problems are solved with *every new observation*, for a total of $2|G||O|$ calls to the planner instead of $2|G|$.

In this chapter we present a naive approach to goal recognition by planning which utilizes the planner a minimal amount of time, then we delve into a description of a general *Offline* PRP algorithm and conclude with a general algorithm for online recognition in continuous domains that solves the plan-recognition problem with a new baseline of $|G|(|O|+1)$ planner calls. This algorithm relies on the formulation described in the previous chapter and hence does not need to use $\overline{O}$.

4.1 Mirroring as a General Plan Recognition Approach

Determining a plan π that maximizes the estimates defined above can be expensive. In particular, the second principle seems to require an expensive search for hypotheses that minimize matching errors, e.g., by multiple calls to a planner to generate candidates.

Mirroring offers a shortcut to generating hypotheses which are guaranteed to minimize matching errors. Instead of generating hypothesized plans and then testing them for their matching error, we synthesize a plan hypothesis π_g^O for each $g \in G$, such that π_g^O *passes through the observations* and continues to g. The concatenation of all observations forms a skeleton path, which can be used as a constraint on generating an optimal plan passing through it. The final segment of this path will be from the last point of the last observation, to the goal g, thus creating a full plan from I to g. As the resulting plan is generated to match the observations perfectly, it will have a matching error of 0, and therefore a maximal $P(O|\pi_g^O) = 1$. Note that this is true of all plans π_g^O.

The generation of optimal plans $\hat{\pi}_g$ for all g is straightforward in most OTS discrete-domain planners, and indeed used in [81]. In continuous domains, this is done by using OTS motion planners that allow inputting way-points and other path constraints that must be respected in the output, e.g., [61].

Thus now we have a set of solution candidates: a set of $|G|$ plans, each $\pi_g^O, g \in G$ maximizing $P(O|\pi)$. All that remains is to compute the optimal plans $\hat{\pi}_g$—in service of the principle of rationality—so that their costs can be compared as described before. This requires a single call to a planner, on the planning problem defined by I and g, which are given in R. Plans maximizing $P(\pi|g)$ will be selected as π_R.

One caveat with this shortcut method is that while it handles missing observations (i.e., gaps in the observations, including from the last observation to the goal), it may be susceptible to noisy observations, which should have been ignored. We do not address this here; see Sohrabi et al. [95] for a potential treatment in discrete domains; the continuous domain case is open.

4.2 Offline Mirroring

As previously mentioned, in *offline* plan recognition the set of observations, O, while it may be incomplete in itself, is revealed ahead of time rather than being revealed incrementally. Algorithm 1 is an *offline* algorithm for a continuous-domain goal recognizer that works by planning, *sans* a plan library according to the guidelines of PRP. The algorithm follows the process described above.

Initially we calculate the path $p = \bigoplus_{o \in O} o$, i.e., the (partial) path created by concatenating all observations, in order (line 3; we use $\bigoplus$ to denote the concatenation of all observations $o \in O$, analogously to Σ and $+$, respectively). As was shown in [57], the prefix plan need only be calculated once as it will necessarily be identical for all of the goals.

Then, for all goals $g \in G$ (keeping in mind that each goal $g \in G$ is a (possibly partial) state s_g), the algorithm computes the corresponding optimal plan $\hat{\pi}_g$, from initial state $I \in W$ to goal g, by calling on the planner (line 5). In line 6 the planner is called a second time, to create the plan suffix π_g, which is a path from the last seen observation o_n, where $|O| = n$, to goal g. Then the candidate plan hypothesis, π_g^o is composed in line 7 and in line 8 is measured against $\hat{\pi}_g$ using the scoring procedure described above. Greater score means closer matching of π_g^o to $\hat{\pi}_g$, and thus by implication, a better hypothesized matching between the observations and the associated goal g. As differences between them grow, the ratio of the costs decreases, resulting in a lower score. Finally, these rankings are transformed into probabilities $P(G|O)$ via the normalizing factor $\eta = 1/\sum_{g \in G} score_i$ (lines 9–10) and the goal with the highest probability is chosen as the leading goal candidate hypothesis (line 11).

4.3 Online Mirroring

Algorithm 2 is an *online* algorithm for a continuous-domain goal recognizer that works by planning, *sans* a plan library. In online goal recognition we assume that the set of observations O is incrementally revealed and the task is to converge to the correct goal hypothesis as early on in the recognition process as possible.

Algorithm 1 OFFLINE RECOGNIZER $(R := \langle W, O, I, G \rangle, planner)$

1: $p \leftarrow \emptyset$
2: **for all** $o \in O$ **do**
3: $\quad p = \bigoplus_{o \in O} o$ ▷ the (partial) path created by concatenating all obs
4: **for all** $g \in G$ **do**
5: $\quad \hat{\pi}_g \leftarrow planner(I, g)$ ▷ generate optimal plan from initial state to goal g
6: $\quad \pi_g \leftarrow planner(o_n, g)$ ▷ generate suffix plan from last obs. to goal g
7: $\quad \pi_g^O \leftarrow p \bigoplus \pi_g$ ▷ compose the candidate plan hypothesis for goal g
8: **for all** $g \in G$ **do**
9: $\quad P(g|O) = \eta \cdot \frac{cost(\hat{\pi}_g)}{cost(\pi_g^O)}$ ▷ calculate probability for goal g
10: $\pi_R \leftarrow \text{argmax}_{g \in G} P(g|O)$ ▷ the plan that maximizes $P(g|O)$ will be selected as the leading plan hypothesis and the corresponding goal g as the leading goal candidate

A *naive* approach would be to re-run the offline algorithm for each new observation and for every goal. This will results in an inefficient $2|O||G|$ number of calls to the planner, $2|G|$ calls for each incoming observation; one call for calculating the optimal plan $\hat{\pi}_g$ (Algorithm 1 line 5) and another for calculating the plan suffix π_g (Algorithm 1 line 6). From here on we will relate to this approach as *Naive*.

However, as the optimal plan, $\hat{\pi}_g$, does not depend on O, it may be generated only once for every goal, while the candidate plan hypothesis, π_g^O, needs to be re-synthesized each time from its component parts. As O is incrementally revealed p needs to be updated to incorporate the latest observation and π_g must be recalculated by the planner, from the last seen observation point to each g. We therefore come up with a new baseline approach, which we will later refer to as *Baseline*.

In the new *Baseline Online Mirroring* algorithm, Algorithm 2, the generation of the optimal plans, $\hat{\pi}_g$ for all $g \in G$, is done only once (lines 2–3), as it was moved out of the main loop. This immediate alteration is highly effective and will result in a reduction of the overall number of calls to the planner by half, from a previous baseline of $2|O||G|$ to $(|O|+1)|G|$ calls. The composition and scoring of π_g^O (lines 8–9) are as before. The generation of the observations path, p, is now done incrementally (line 6), and the generation of the suffix plan, π_g is carried

out from the ending point of the most recent observation, o (line 7). Note that the evaluation of each g in lines (10–12) needs to be incrementally done for each observation in order for the algorithm to provide online results.

Algorithm 2 BASELINE ONLINE MIRRORING $(R := \langle W, O, I, G \rangle, planner)$

1: **for all** $g \in G$ **do**
2: $p \leftarrow \emptyset$
3: $\hat{\pi}_g \leftarrow planner(I, g)$ ▷ generate optimal plan from I to goal g
4: **while** New $o \in O$ is available **do**
5: **for all** $g \in G$ **do**
6: $p = \bigoplus_{o \in O} o$ ▷ the (partial) path created by concatenating all obs
7: $\pi_g \leftarrow planner(o, g)$ ▷ generate suffix plan from last obs. to goal g
8: $\pi_g^O \leftarrow p \bigoplus \pi_g$ ▷ compose candidate plan hypothesis for goal g
9:
10: $P(g|O) = \frac{cost(\hat{\pi}_g)}{cost(\pi_g^O)}$ ▷ calculate probability for goal g
11: **for all** $g \in G$ **do**
12: $P(g|O) \leftarrow \eta \cdot P(g|O)$
13: $\pi_R \leftarrow \text{argmax}_{g \in G} P(g|O)$ ▷ the plan that maximizes $P(g|O)$ will be selected as the leading plan hypothesis and the corresponding goal g as the leading goal candidate

This online algorithm establishes the baseline of $(1 + |O|)|G|$ calls to the planner [108].

5 Experiments

> "All life is an experiment. The more experiments you make the better."
>
> *Ralph Waldo Emerson*

We empirically evaluated the performance of the *Mirroring* algorithm described above in several continuous and discrete domains. Indeed, we measure its performance in over a thousand recognition problems, spread over three continuous domains, and six discrete domains. We additionally contrasted the performance of the Mirroring algorithm as a PRP method in comparison with a widely accepted library-based approach.To further evaluate the factors impacting recognition success we evaluated recognition success while experimenting with both unmodified OTS planners and an especially built shape planner, in some cases also comparing different planner performance on the same problem. We additionally, evaluated the sensitivity of the recognition approach, by contrasting results over recognition problems with varying levels of difficulty. Finally, we measured the performance of the algorithm as a continuous goal recognizer, contrasting the results with a discrete goal recognizer and comparing against previous work.

5.1 Experimental Domains

We would like to begin with a general overview of the different experimental domains used repeatedly within this dissertation.

5.1.1 Six Discrete Benchmark Domains

In order to demonstrate the generality of *Mirroring* as a PRP approach and to show that it does not fail when comparing to existing PRP approaches we re-ran the *entire* set of benchmark plan-recognition problems used in [81] and then in [95, 70]. All in all, there are 450 problems in six classical planning domains: KITCHEN, BLOCKS WORLD, LOGISTICS, INTRUSION DETECTION, IPC-EASY, and CAMPUS. We used the default *HSP: Heuristic Search Planner* [14] as the black-box planner used within the PRP process.

5.1.2 Navigation Goal Recognition

A very useful domain in order to evaluate plan recognition is the domain of navigational goal recognition. Here, the target is to recognize navigational goals as soon as possible while the observations, i.e. observed agents' positions, are incrementally revealed. As this is a very popular and highly researched domain there are several possibilities of existing OTS motion planners to use. We also use it to fully demonstrate the effects of the discretization process describes in Chapter 3.

Open Motion Planning Library A very useful benchmark 3D environment is the Open Motion Planning Library (OMPL [98]). We implemented online *Mirroring* algorithms to recognize the goals of navigation in 3D worlds.

As the black box planner used in the *Mirroring* process we used TRRT (Transition-based Rapidly-exploring Random Trees), an off-the-shelf planner that guarantees asymptotic near-optimality by relying on the notion of minimal work path preferring shorter solutions [42]. Therefore when we hereafter relate to the *cost measure* of the plan we simply mean the length of the path. This planner is available as part of the Open Motion Planning Library along with the OMPL *cubicles* environment and default robot. Each call to the planner was given a time limit of 1 sec.

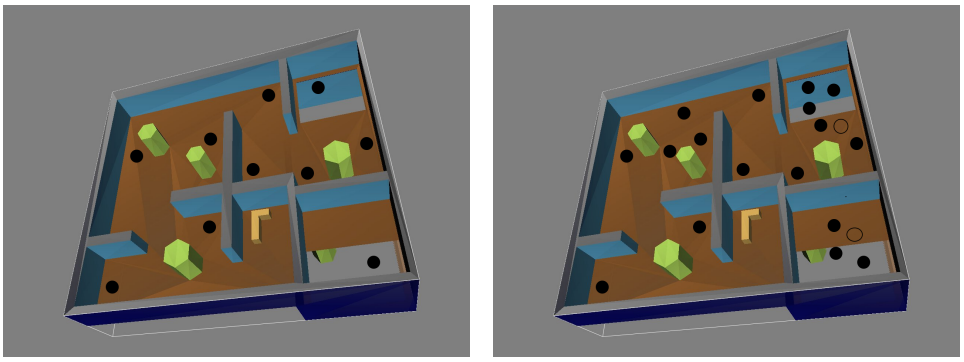
To generate goal recognition problems, we experimented with two challenging scenarios.

- For the first scenario we arbitrarily selected 11 points spread out relatively evenly over the cubicles environment (Figure 5.1(a)). The yellow polygon representing the robot and the green polygons representing obstacles in the environment. We generated observed paths from each point to all others, for *a total of* $110 \times 2 = 220$ *goal recognition problems*. The observations were obtained by running the RRT* planner on each pair of points, with a time limit of 5 minutes per run. RRT* was chosen because it is an asymptotically optimal version of RRT and will converge to the optimal path as a function of time [47]. Five minutes was a substantial amount of time that enabled us to obtain paths that appear near-optimal. We used the OMPL *interpolate* method to generate between *20* and *76* observed points for each path problem. This method inserts a number of states in a path so that the path is made up of exactly *n* states that are inserted uniformly (more states on longer segments). Changes were performed only if a path had less than the specified 20 states.

- In order to evaluate the sensitivity of the Mirroring recognition approach we additionally created another recognition scenario within the same environment. We therefore added 9 goal points to the recognition problems in the navigation domain (i.e., 19 potential goals in each recognition problem Figure 5.1(b)), for a total of 380 recognition problems. These extra points were specifically added in close proximity to some of the preexisting points, such that navigating towards any one of them appears (to human eyes) to be just as possible as any other.

Robot Operating System To show the applicability of the Mirroring approach to robotic applications, we implemented *Mirroring* in a cooperative robotic team task. We used ROS [79] to utilize the recognition algorithm to recognize the goals of navigation in 3D worlds using the ROS *MoveBase* default planner. An alternative interface—to discrete planners—is described in [19]. We used the ROS standard Gazebo simulator to simulate an environment of a soccer field, free of any obstacles, with two robots operating as team members (Figure 5.2).

The observed robot was given an initial goal to travel to, proceeding to execute



(a) Original Scenario (b) Clustered Scenario

Figure 5.1: Visualization of original and clustered goals environment.

the plan in a straightforward manner, and the observing robot had to strategically place itself in a pre-chosen position to assist the other robot team member. If the observed robot navigated to *goal 4* the strategic place to assist it on the offense would be to navigate to *goal 3* and vice versa. Likewise also with goals 1 and 2.

The observed robot always started at the same initial point in the middle of the field, while we experimented with 3 different starting points for the observing robot; two points behind the observed robots position and on parallel sides (Figure 5.2, init points 1 and 2) and one point past the observed robot in the middle of the field (init point 3). We ran 10–20 runs from each initial position to each of the goals for a total of 193 problems.

Figure 5.2: Robotic soccer experiment setup (via RVIZ)

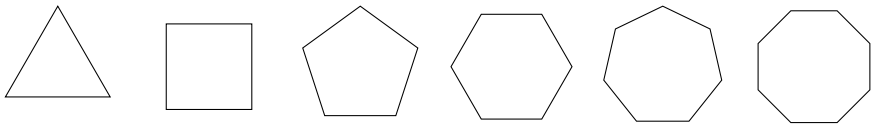


Figure 5.3: Regular polygons tested in the shape recognition experiment.

5.1.3 Shape Sketch Recognition

As a final set of experiments and in order to evaluate Mirroring and contrast its performance with that of human recognition we designed an especially built shape planner to recognize geometric shapes as they were being drawn. Here the task is to recognize 2D hand-drawn regular polygons, as early as possible, as they are revealed one edge at a time.

The shape-drawing planner takes as input a partial drawing (represented as a sequence of angles and edges), and a goal shape type (equilateral triangle,square, etc.) and attempts to complete the drawing to the goal shape or report failure if it cannot be done. To complete the drawing, the planner looks at the angles between edges that are already drawn (if any), and adds edges that complete the regular polygon whose angles best match the angles observed.

The recognition data for the experiment is a data-base of scanned hand-drawn regular polygons. We asked three people (2 females and 1 male, ages 26-29) to create a data base of 18 hand drawn regular polygons : 3 triangles, 3 squares, 3 pentagons, 3 hexagons, 3 septagons and 3 octagons. The participants were instructed to draw the shapes as accurately as they could without the use of any external aids, making them as regular as possible (i.e., equilateral, equiangular) as seen in Figure 5.3. Shapes were drawn in various scales, rotations, and translations with respect to the center of the page. Naturally, hand drawings, even under these ideal conditions, reflect quite a bit of inaccuracy (see Figure 5.4, top).

We wanted to use the recognizer on the same images as humans, allowing observations of one edge at a time. We therefore scanned each image and manually separated the edges into different, consecutive, images so when presenting them sequentially it would appear that they were constructed edge by edge: In the first image only the first edge would appear, in the second image the first and the

second edges would appear and so on.

To obtain the line information from each image we used OpenCV to implement a Hough Transform [27], a feature extraction technique commonly used in computer vision. The performance of the technique on two example drawings is shown at the bottom of Figure 5.4.

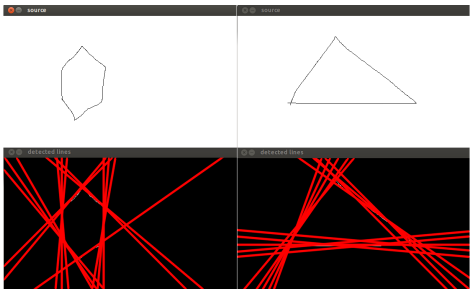


Figure 5.4: Drawn shapes (above) and their Hough transforms (below).

For each detected edge we were able to extract the following information: the initial and final x,y coordinates, the ρ parameter, which is the algebraic distance between the line and the origin, and Θ, the angle of the vector orthogonal to the line and pointing toward the half upper plane. From this it was easy to find the slope and intercept of each line along with the initial and end point coordinates.

Because of noise in perception (e.g., scanning noise) and drawing inaccuracy, the Hough transform often generates several candidate lines for each edge (can be seen in Figure 5.4). To find the common lines we used open-source hierarchical clustering software [26]. We defined each node to have equal weight and used Euclidean distance to measure the distances between each node and gave a threshold of 100 to check for affinity between nodes. Following this we had the number of lines recognized and the slope and intercept of each line.

5.2 Evaluation Criteria

Let us examine the recognizer output on a specific problem. Figure 5.5 is an instance of the recognition result on a given problem. The X-axis marks the incrementally revealed observations. The Y axis measures the rank of the correct goal hypothesis among all the goals ranked by the recognizer, thus lower is better (rank

1 indicates that the correct goal was ranked as the top hypothesis). Naturally, this rank is only known post-hoc, as the recognizer does not have access to the ground truth during the actual run.

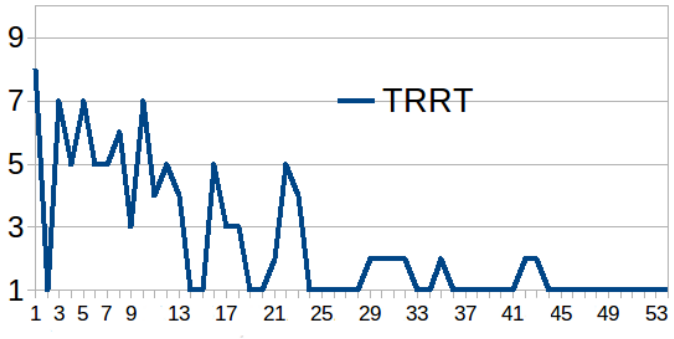


Figure 5.5: Recognition result example. The Y axis denotes the goal ranking and X axis denotes the incoming observations.

Let us look at Figure 5.5. After the 9th observation, the correct goal was ranked 3 (out of 10) by the recognizer. As more observations come in the recognition problem becomes easier and finally converges to ranking the correct goal at the top, i.e. rank 1, after the 44th observation. Such graphs can be drawn for any online recognition problem to compare the performance of different recognizers.

When measuring recognition results we want to evaluate both the recognition performance as well as the efficiency of each online approach.

Recognition Performance Measures We used two separate measures to estimate recognition performance :

- The time in which the recognizer converged to the correct hypothesis (including 0 if it failed). We will further refer to this measure as *Convergence*. This is measured by counting the number of observations from the end, hence higher values indicate earlier convergence and are therefore better. We normalize using the length of the observation sequence.

- The number of times the correct hypothesis was ranked at the top (i.e., rank 1), which indicate general accuracy. We will further refer to this measure

as *Ranked First*. The more frequently the recognizer ranked the correct hypothesis first, the more reliable it is. Again higher values indicate more correct hypotheses during the recognition process. We again normalize using the length of the observation sequence.

For example in Figure 5.5, the recognizer converged to the correct result at observation number 44 out of 54. When normalizing for the observation sequence length, we measure the normalized convergence of the TRRT recognizer at 18.5%. The earlier the convergence, the better. With regards to the amount of times the planner ranked the correct goal as the top hypothesis, the recognizer ranked the correct goal at the top 29 times. Again when normalizing according to observation sequence length we learn that 53.7% of the observations were correct meaning that the recognizer was correct more than half of the time.

Efficiency Measures In order to evaluate the overall *efficiency* of each approach we used several different measures, each relevant to specific experiments:

- The number of times the planner was called within the recognition process. The lower the value the more efficient the process. This measure is only relevant for PRP based approaches which utilize a planner within the recognition process. Recognizers that do not utilize a planner will have no planner calls at all.

- The overall time (in sec.) spent planning. Again lower values are better, indicating less time spent utilizing the planner within PRP approaches.

- The overall run-time the simulated robot ran until reaching its target goal (measured in sec.). This would include time spent planning as well as time spent executing and evaluating the plan. The lower the time the more efficient the robot, hence lower values are better.

- The distance, length of the path, the simulated robot covered until reaching its target. The smaller the distance, the more direct the path from start position to goal, with less detours taken on the way. Again lower values are better indicating a more efficient path to reach the goal and also, indirectly,

earlier recognition success, as the robot did not need to re-plan or deviate much from the original plan.

Though these parameters are closely linked, they are not wholly dependent. While a reduction in overall number of calls to the planner will also necessarily result in a reduction in planner run-time and overall run-time, the total amount of time allowed for each planner run may vary according to the difficulty of the planning problem and therefore create considerable differences.

5.3 Results

5.3.1 Mirroring vs. Library-Based Methods

For the first set of experiments we wanted to evaluate the performance of *Mirroring*, as a PRP approach, with comparison to that of library-based planners. Mirroring has the principled advantage over library-based methods in terms of storage, and in being able to handle any arbitrary initial observed state. However, this may also be a disadvantage, in principle: Mirroring does not utilize prior knowledge even when it can be made available. To evaluate this aspect, we contrasted Mirroring with an especially built HMM (hidden Markov model) [29] using Matlab and MathWorks [28] as HMMs are often used as a standard technique, e.g., [10].

Testing the HMM on plans unknown to it is a valid, but futile exercise, where the superiority of Mirroring, and indeed any PRP based approach, would be obvious. In order to evaluate HMMs vs Mirroring, when the plans are *known* to the HMM, we used the OMPL navigation domain described previously (Section 5.1) with 110 goal recognition problems. In order to do this, we first needed to discretize the navigation problem according to the guidelines presented in Section 3.

We created a robot-size cell grid in the 3D environment, each cell represented by a state in a hidden Markov model. For each such recognition problem we trained one HMM using the same paths generated by the Mirroring implementation. In particular, in order to increase the recognition success of the HMM, we used 20 paths generated by the asymptotically-optimal planner RRT*. In other words, we created a specialized HMM, trained on 20 examples of asymptotically-optimal data, for each recognition problem. HMM training and recognition were

carried out using the standard MATLAB HMM package [28]. Mirroring recognition was measured while utilizing the TRRT planner with a time limit constraint of 10 seconds.

Figure 5.6 contrasts the recognition results of HMMS and Mirroring. We can clearly see that even without any prior knowledge, Mirroring is on-par with the HMM results. In terms of *Convergence* Mirroring achieved 27.5% while the HMM achieved 26.3% recognition success. In terms of the *Ranked First* measure Mirroring achieved 37.2% while the HMM reported at 37.4%. The error bars indicating that the differences are not substantial. Obviously, as more prior knowledge is available, this can change.

In order to more fully understand how much additional knowledge would improve the performance of the HMM. We ran another experiment, this time giving the HMM 95 instances to learn from, instead of 20. As expected the HMM Convergence results improved to 37.4% and the Ranked First results improved to 46.8% surpassing the Mirroring recognizer. However, it is to be taken into account that these instances are not always available and also that any additional knowledge must include time for learning. In the current instance the learning time of the HMM of 95 samples took about 11 hours while for the previous 20 samples it took only 5.5 hours. Our conclusion is that Mirroring should be preferred when relatively less data is available, or when the number of possible plans is very large (or infinite, as is in this specific domain).

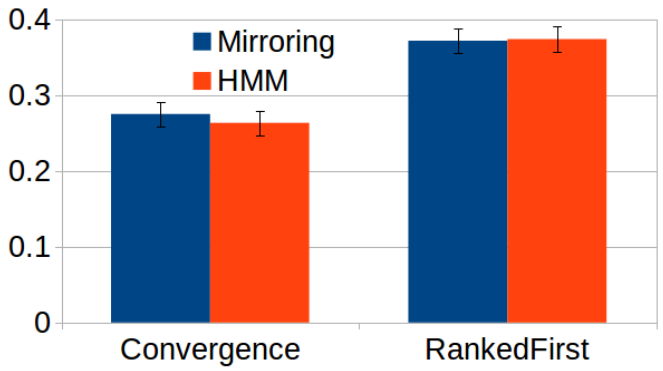


Figure 5.6: Mirroring Vs. HMM Results

5.3.2 Mirroring on Robots

we also implemented the Mirroring approach on simulated ROS-enabled robots while measuring the efficiency of the algorithm as compared to two separate approaches, one containing full knowledge of the observed agents' intentions and the other containing no knowledge and no reasoning mechanism.

We used the ROS [79] domain described in Section 5.1 to utilize the recognition algorithm to recognize the goals of navigation in 3D worlds. We simulated an environment of a soccer field, with two robots operating as team members; One robot (observed) was given an initial goal to travel to, and would head there as soon as an experiment trial began and the second robot observed the motions of the other robot positioning itself in one of several pre-selected locations, to support its teammate. We experimented with 3 different starting points for the observing robot, initpoints 1–3 (Figure 5.2) and ran 10–20 runs from each initial position to each of the goals for a total of 193 problems.

We used the *run-time* and *distance* efficiency measures described in the previous Section and compared between three different approaches :

- The *Mirroring* online plan recognizer in its *Baseline* form, marked as *MIRR*.
- An online goal recognizer which is given full knowledge of the intended goal to the observing robot, ahead of time, allowing the observing robot to navigate directly towards it, marked as *FK*. This approach signifies the state-of-the-art and should be the most efficient both in terms of run-time and distance covered.
- An online goal recognizer which is given no (zero) knowledge of the intended goal, thus forcing the observing robot to wait for its team member to reach its desired goal, before it can navigate towards the complementary location, marked as *ZK*. This approach will be efficient in distance, as the agent takes no detours along the process but should be heavy in time - because the observing agent has to wait for the observed agent to finish its run.

		FK		MIRR		ZK	
		Time (Sec.)	**Dist.**	**Time (Sec.)**	**Dist.**	**Time (Sec.)**	**Dist.**
I1	**G1**	**12.41**	6.10	17.36	6.92	20.88	6.34
	G2	**10.10**	4.60	18.65	6.83	24.26	4.57
	G3	**9.24**	4.04	10.62	4.45	20.67	4.03
	G4	**5.72**	1.97	13.40	4.10	20.40	1.98
I2	**G1**	**10**	4.59	15.89	5.30	25.45	4.57
	G2	**12.35**	6.14	15.32	6.52	32.62	5.83
	G3	**5.80**	1.97	14.10	3.44	17.67	1.84
	G4	**9.10**	4.00	19.56	6.13	24.45	4.02
I3	**G1**	**5.80**	1.97	12.3	4.91	17.18	1.92
	G2	**5.80**	1.97	17.10	6.26	26.03	6.11
	G3	**9.10**	4.00	16.19	4.99	20.43	3.88
	G4	**10.00**	4.59	17.35	5.47	21.49	4.38

Table 5.1: Mirroring vs. full and zero knowledge

The results are displayed in Table 5.1.

As seen in Table 5.1 with regards to performance time, the results show that Mirroring substantially improves on the *zero knowledge* approach, while requiring no precalculations; all needed plans are generated on-the-fly via the planner. Understandably, Mirroring falls short of the *full knowledge* approach as it generates hypotheses on the fly following observations, which leads to some deviations from the optimal, direct route. With regards to the distance of the path, both *ZK* and *FK* outperform *MIRR* since they both take a direct route to each of the goals.

5.3.3 Mirroring vs. Existing PRP Approach

As mentioned in Section 5.1 we re-ran the *entire* set of benchmark plan-recognition problems used in [81] and then in [95, 70] consisting of 450 problems in six classical planning domains. Mirroring was used to rank the goal hypotheses (cost of ideal plan vs cost of optimal plan that goes through the observations), and compared to the ranking generated by the PRP formulation in [81] (cost of optimal plan that deviates from observations vs cost of optimal plan that goes through the observations).

Out of the 450 problems, there were *only two* where the results differed: In two variants of the same problem (with a full set and a partial set of observations), the original formulation ranked the true goal as the winner. Mirroring ranked it as

a winner, along with two others. These are two variants of the same recognition problem: One with 70% of the observations, and one with the full set of observations. For smaller sets of observations of these problems, VK and RG are in agreement. In an additional 4 problems, the calls to the planner failed in generating the plan that goes through the observations. Since both formulations require this plan, both methods failed.

When it comes to run-time results the differences in performance are greater. Figure 5.7 compares the run-time results of both approaches over the successful runs of the benchmark plan-recognition problems. The X-axis displays the different percent of observations revealed for each run. The Y-axis indicates indicate the average time it took each recognizer to converge to a result, in seconds. Lower values are better indicating less run-time and more efficient performance.

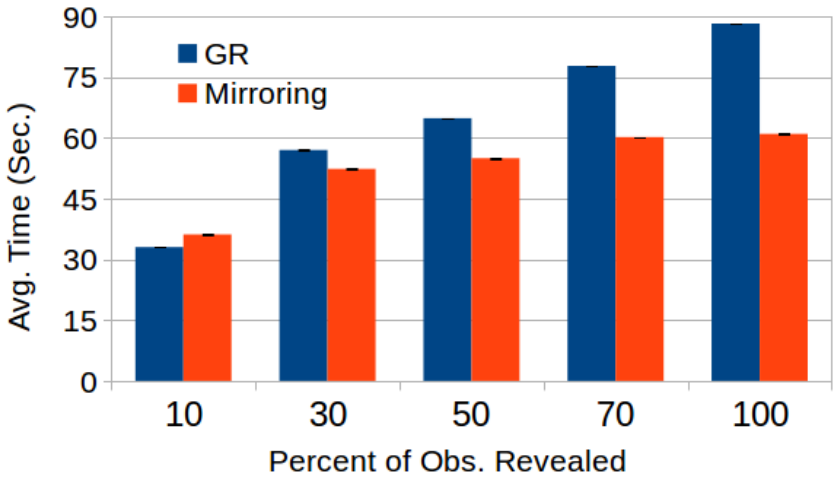


Figure 5.7: Run-time Results

In general, run-times using Mirroring were significantly better than using the previous PRP approach. Indeed this is consistent over all percentages of observations revealed apart from the initial 10 %. We hypothesize that this is because when there are less observations, it is substantially easier to generate a plan deviating (*at least partially*) from the observations and that is the reason that, in the Ramirez and Geffner approach, calculating $\overline{O}$ was extremely fast.

5.3.4 Continuous vs Discrete Plan Recognition

To demonstrate the shortcomings of the discretization process discussed in Chapter 3 we devised a series of experiments evaluating the recognition success of the *Mirroring* approach in both discrete and continuous environments, over two separate domains.

Robot Operating System Experiment We used the ROS [79] domain described in Section 5.1, this time to evaluate the impact of discretization. We compared the recognition performance of the recognizer when using the continuous planner, to that when the recognition is carried out on a discretized grid. In particular, We divided the environment into robot-sized grid cells and converted all consecutive points along the path to the middle of each of the corresponding cells in the grid. In the same manner we also converted the goal locations.

We use two of the previously described recognition performance measures; convergence and number of times the true goal was ranked first. Instead of reporting on percentages, Table 5.2 uses ratios (thus 90% is reported as 0.90). We remind the reader that higher values indicate earlier convergence and are therefore better.

Each column in Table 5.2 marked **C** reports on the results from using the continuous-planner in recognition. Each column marked **D** reports the results from the discretized recognition process. The rows, marked *I*, denote different initial locations.

We see that for all problems the results are higher for the *continuous* recognizer over the *discrete* instance. This arises from the fact that the discrete recognizer may lose information in the discretization process. It is safe to say that with a reduction of the discretization factor these differences will decrease until the performance will be equivalent. However finding just the right amount of granularity could prove wasteful and domain specific. This illustrates further the substantial need for specified recognizers fit to operate in continuous domains.

Shape Recognition Experiment We additionally evaluated the continuous, Mirroring, plan recognizer on the shape sketch recognition domain introduced

		Goal 1		Goal 2		Goal 3		Goal 4	
		C	**D**	**C**	**D**	**C**	**D**	**C**	**D**
I1	**Conv**	**0.90**	0.86	**0.88**	0.65	**0.46**	0.00	**0.21**	0.00
	Rank	**0.93**	0.86	**0.89**	0.66	**0.47**	0.05	**0.30**	0.00
I2	**Conv**	**0.87**	0.82	**0.94**	0.74	**0.32**	0.00	**0.60**	0.15
	Rank	**0.90**	0.83	**0.96**	0.75	**0.51**	0.03	**0.79**	0.57
I3	**Conv**	**0.96**	0.67	**0.53**	0.40	**0.82**	0.35	**0.87**	0.26
	Rank	**0.96**	0.68	**0.54**	0.44	**0.82**	0.31	**0.89**	0.43

Table 5.2: ROS continuous vs. discrete results.

in [108] and described in Section 5.1. Here the task is to recognize 2D hand-drawn regular polygons, as early as possible, as they are revealed one edge at a time. We used the same human drawn, 18 shape data base the same domain-dependent planner. These shapes were hand drawn in various scales and rotations and naturally contained quite a bit of noise in terms of angle and edge sizes.

We contrast the performance of this continuous angle representation with a discretization of the angle size into either a 1°, 10°, 20°or 40°angle. For example, if the size of an observed angle was 78°, in terms of 40°discretization it would be translated to an 80°angle. The results are displayed in Table 5.3 using the performance evaluation criteria of *Convergence* and *Ranked First* which were introduced in the previous Section. Columns 1–5 measure the average convergence percent over of all shapes over all discretization factors, with *cont* denoting the continuous representation. Columns 6–10 measure the amount of time the recognizer ranked the correct goal as first. As we are measuring convergence and the correct ranking of the chosen goal, higher values indicate earlier convergence to the correct result or more correct rankings, hence are better.

Over all criteria the continuous recognizer outperformed or performed just as well as all of the discretized recognition processes. Indeed we can see that over all criteria the results achieved in *Cont* and *1* are fairly similar, this can be understood as 1°being a sufficient amount of discretization granularity to recognize most of the shapes. However, for the *Pentagon* and *Triangle* shapes Convergence and *Hexagon* and *Triangle* shapes Ranked First measure, the results differ, indicating that a 1°angle discretization is insufficient.

	Average Convergence Percent					Average Ranked First Percent				
	1	**10**	**20**	**40**	**Cont**	**1**	**10**	**20**	**40**	**Cont**
Octagon	**0.67**	0.42	0.50	0.13	**0.67**	**0.75**	0.42	0.50	0.13	**0.75**
Septagon	**0.62**	0.38	0.29	0.14	**0.62**	**0.62**	0.38	0.29	0.14	**0.62**
Hexagon	**0.56**	0.33	0.33	0.33	**0.56**	0.56	**0.61**	**0.61**	**0.61**	**0.61**
Pentagon	0.50	**0.60**	0.40	0.40	**0.60**	**0.60**	**0.60**	0.40	0.50	**0.60**
Square	**0.75**	**0.75**	0.50	0.25	**0.75**	**0.75**	**0.75**	0.50	0.25	**0.75**
Triangle	0.33	**0.44**	**0.44**	0.33	**0.44**	0.44	**0.56**	0.44	0.33	**0.56**

Table 5.3: Shapes continuous vs. discrete results

5.3.5 Effects of Planner Choice

One of the key factors that may impact recognition success within any PRP based approach, and *Mirroring* in particular, is the choice of planner being used within the recognition process. In order to measure the effect of this factor we empirically evaluated the approach by comparing the recognition success of several different planners in recognizing the navigation goals in a benchmark 3D environment, where the target is to recognize navigational goals as soon as possible while the observations, i.e. observed agents' positions, are incrementally revealed. We used the aforementioned first scenario constructed with the Open Motion Planning Library (OMPL [98]) *cubicles* environment along with the default robot, (Figure 5.1(a)). We tested over the previously described 220 goal recognition problems and experimented with four off-the-shelf OMPL planners.

The planners differ in their optimality guarantees. Two from the RRT (Rapidly-exploring Random Trees) family [65] offer some guarantees. RRT* guarantees asymptotic optimality; it will utilize the full duration of time allotted to it to generate incrementally improving solutions. TRRT only guarantees asymptotic *near-* optimality by taking into consideration state costs to compute low-cost paths complying with a predefined optimization objective according to minimum path-length, thus preferring shorter solutions. The two other planners used offer no optimality guarantees: RRTCONNECT, and KPIECE1 [97].

Table 5.4, columns 1–3, contrasts the results of the four planners, when used in the *Mirroring* formulation. The first two columns measure recognition performance. Each of the columns shows the mean recognition results over the same set of recognition problems with higher values denoting improved results. We see

that TRRT and RRT* are clearly and significantly better for online goal recognition of paths generated by RRT* than RRTConnect and KPIECE. RRT* and TRRT both tend to produce paths closer to optimal, RRT* being asymptotically optimal and TRRT guaranteeing asymptotic *near*-optimality. Indeed we see that they are nearly indistinguishable from each other in terms of recognition success (though TRRT is a bit better).

However, the two top planners differ from each other very much in run-time (Table 5.4, column 3). Every call to the planners was limited to one second of run-time. But given that a planner is called with each new observation, for each one of the goals, the mean total time can grow very quickly. As we can see RRT* takes (by design) 100% of the time allotted, but others do not. Indeed, we see that TRRT is the second quickest, and is beaten only slightly by RRTConnect.

	10 Goals (220 Problems)			**19 Goals** (380 Problems)		
	Conv	**Rank**	**Time**	**Conv**	**Rank**	**Time**
TRRT	**25.82**	**35.02**	28.10	**16.11**	**22.95**	36.56
RRT*	22.52	32.55	100.40	13.26	21.24	100.46
RRTCONNECT	8.21	21.20	**22.11**	3.45	13.42	**21.33**
KPIECE	9.69	21.59	34.11	4.74	13.56	49.86

Table 5.4: Recognition with various planners.

We conclude that the recognition results greatly improve with the optimality of the planner utilized in the recognition process. This is due to the fact that we used an optimally converging planner (RRT*) to generate the observations providing tight dependence between the planner used to generate the observations and the planner used in the recognition process. Moreover, in the 3D navigation domain, TRRT seems to offer a remarkable choice for this task: It produces good results, while being very fast.

5.3.6 Sensitivity to Recognition Difficulty

In online, continuous domains, the difficulty of the recognition problem could possibly effect recognizer performance and efficiency. By varying levels of difficulty we refer to problems that have more goals, and where the goals are clustered closer together exacerbating the recognition task. We wanted to evaluate the sensitivity of the results shown above to the hardness of the recognition problems.

We therefore added another 9 goal points to the recognition problems in the navigation domain (e.g., a total of 19 potential goals in each recognition problem), Figure 5.1(b). This gave us a total of 380 goal recognition problems. In order to increase the difficulty of the recognition task these extra points were specifically added in close proximity to some of the preexisting 10 points, such that navigating towards any one of them appears (to human eyes) to be just as possible as any other.

Table 5.4, columns 4–6, compares the different planners performance (%) over the now *harder* clustered goals problems. We can see that the relative performance success ordering remains as it was for the original scenario. TRRT and RRT* once again significantly outperform KPIECE and RRTCONNECT, in both *convergence* and *ranked first* measures.

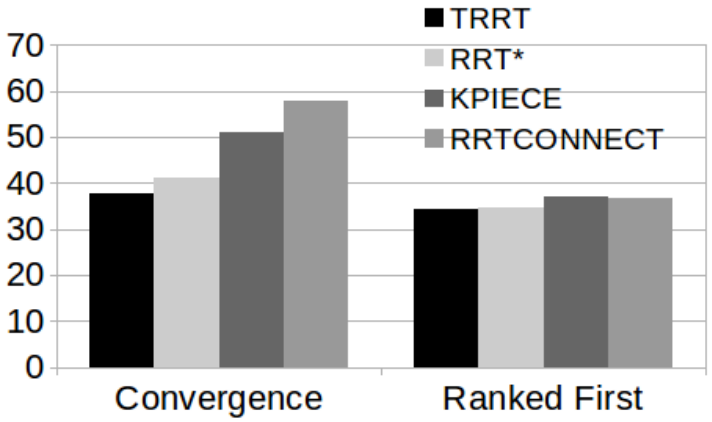


Figure 5.8: Clustered goals planner performance deterioration.

Figure 5.8 compares the deterioration (%) along each of the performance criteria across the different planners over the 380 harder problems. A lower result here is better, indicating less deterioration in performance. We see that for the *ranked-first* measures all planners deteriorated equally with deterioration percents ranking from 34.4%–37.18%. However, for the *convergence* measure TRRT deteriorated by only 37.59%, followed closely by RRT* with 41.13% while KPIECE and RRTCONNECT deteriorated considerably with 51.12% and 57.97%.

5.4 Summary

This part of the dissertation presents *Mirroring* as an approach to plan and goal recognition that does not rely on a plan library, instead using a planner to generate recognition hypotheses that are continually matched against incremental observations. *Mirroring* relies on a new, general formulation of plan recognition in continuous domains which can also generalize to discrete domains. The key insight is that by using motion planners in recognition, we avoid early commitment to a granularity (discretization) level, and thus can choose the best discretization for the recognition problem at hand. The use of *Mirroring* allows the use of OTS, unmodified planners. It gives rise to a recognition procedure that uses two calls to a planner for each goal while accounting for missing observations and efficiently dealing with online recognition.

We evaluated the approach in a two challenging navigational goals domain over hundreds of experiments and varying levels of problem complexity as well as within the context of shape recognition, where by a planner is re-used by a recognition process, allowing drawn-shape recognition by drawn-shape planning.

The approach was additionally compared to prior formulations in discrete domains (which cannot address continuous domains), and has additionally been evaluated with four standard motion planners, and one robotics planner, —all without modifying the planners in any way. The approach has a number of technical advantages specific to recognition (such as fast on-line computation with no preprocessing), but most importantly, is particularly suited to agents, where a complete agent is expected to have a planner for its own goals, and this can be utilized for recognition, without the need for a separate source of recognition knowledge.

Part II

Cognitive Inspiration

6 Mirroring in Humans and Agents

> ”There is nothing either good or bad, but thinking makes it so.”
>
> *William Shakespeare (Hamlet)*

Humans use sketches, drawn on paper, on a computer, or via hand gestures in the air, as part of their communications with agents, robots, and other humans. They may use them in computer graphics applications that require sketch-based modeling [43], in innovative assistive robotic applications [109, 68, 87], or in other sketch-based user interfaces in tablets and other ubiquitous computing devices [66].

To understand how humans perform this recognition we draw from neuroscience, psychology and cognitive science. There has been evidence that humans ability to do online shape recognition comes from the newly discovered *mirror neuron system* for matching the observation and execution of actions within the adult human brain [85]. The mirror neuron system gives humans the ability to infer the intentions leading to an observed action using their own internal mechanism.

Mirror neurons have first been discovered to exist in macaque monkeys in the early 90’s [69]. These neurons for manipulation were seen to fire both when the monkey manipulated an object in a certain way and also when it saw another animal manipulate an object in a similar fashion. Recent neuroimaging data indicates that the adult human brain is also endowed with a mirror neuron system where it is attributed to high level cognitive functions such as imitation, action

understanding, intention attribution and language evolution. The human mirror neuron system may be viewed as a part of the brains' own plan recognition module and can be used to recognize the actions and goals of one or more agents from a series of observations of the other agents' actions.

To recognize shapes in sketches, most existing work focuses on offline (post-drawing) recognition methods, trained on large sets of examples which serve as a plan library for the recognition method [4, 93, 92, 101]. Given the infinite number of ways in which shapes can appear—rotated, scaled, translated—and given inherent inaccuracies in the drawings, these methods do not allow on-line recognition, and require a very large library (or expensive pre-processing) in order to recognize even a small number of shapes that may have been translated, rotated or scaled [2].

Inspired by mirroring processes hypothesized to take place in socially-intelligent brains [69, 46] we present an online shape recognizer that identifies multi-stroke geometric shapes without a plan library using online *Mirroring*. As such, the recognizer uses a shape-drawing planner for drawn-shape recognition, i.e., a form of plan recognition by planning. This method (1) allows recognition of shapes that is immune to geometric translations, rotations, and scale; (2) allows considerable reduction in storage space - eliminates the need for storing a library of shapes to be matched against drawings (instead, only needs a set of possible Goals and a planner that can instantiate them in any manner); and (3) allows fast on-line recognition.

However, the key advantage rises from the point of view of the complete agent that uses the recognition as part of its interactions: an agent that that must not only recognize sketches, but also produce them, and therefore necessarily have a drawing planner already will be able to use the Mirroring method to recognize sketches, without relying on a separate shape recognition library. This is the motivation for our work.

6.1 Related Work

Intelligent systems increasingly rely on sketching, hand-drawing and gestures as input. In its essence sketching and gesturing are one of the fundamental ways

for humans to interact and is therefore often an important part of any intelligent system. The problem of sketch recognition may be also viewed as a very important instance of plan recognition, since part of what makes a system intelligent is the ability to foresee the needs and intentions of its users.

A particular aspect of drawn shapes is that they can be drawn in an infinite number of ways within the drawing area. Furthermore, given that the shapes are drawn by humans, both edges and vertices's are drawn with quite a bit of inaccuracy, in edge curvature (i.e., they are not straight lines), in the accuracy of angles between edges, or even in the drawing of angles themselves (for instance, whether two edges actually intersect in a vertex).

Thus shape recognition—whether offline or on-line—faces the following key challenge: there exist essentially infinite numbers of possible sketches of each goal shape. From the point of view of plan- or goal- recognition, this poses the challenge of recognizing a small set of goals, given an infinitely-large plan library. Naturally, the challenge is exacerbated in on-line shape recognition, as the agent cannot easily tell whether it has seen all observations.

Most approaches to shape recognition use global geometric properties extracted from the drawings, and specialized to the recognition task. For instance, Paulson and Hammond designed a system that works by computing specific tests for all possible shapes, then sorts the matching hypotheses (matching shapes) in order of best fit [67]. Ulgan et al. use a neural network, trained on the relation between the internal angles of a shape and its classification [102]. All of these are offline approaches: they carry out the recognition process only once the drawing is completed.

Online methods in the same spirit (relying on specialized geometric features) include [31, 44]. These implement methods that are invariant to scale and rotation. They use global geometric properties extracted from input shapes ahead of time and associated with certainty degrees using fuzzy logic. These methods required substantial work ahead of time in selecting the best feature to identify a given shape while the initial shape selection process takes into account specific shape related properties. Another method for on-line recognition that also requires considerable training, i.e., an extensive set of examples, is the use of HMMs for sketch recognition [92].

The advantage of the Mirroring approach over these methods is in the utilization of an existing planner in order to perform the recognition process. Thus eliminating the need for training ahead of time and for specific preparatory analysis for each individual plan.

Mirroring relates to the model tracing approach [24] in the sense that the system must possess a computational model capable of solving the problems given to the student. The difference lies in the complexity of implementing a similar mechanism that will be able to work in a continuous, unpredictable domain that has to deal with missing knowledge, noise, and an infinite possibility of solutions.

We follow previous work [88, 87] in treating the problem of on-line recognition of shapes, as they are being drawn, as a problem of on-line goal recognition by Mirroring. However, In [88, 87] Sadeghipour et al. explicitly represent (and store) shape drawing *plans*, that can be used both for recognition and execution by the agent. In contrast, we do not store plans, but instead use a *planner* to generate plans on the fly. Thus in these previous works different rotations of the same shapes have to be stored as separate plans in the plan library, and the plan library must account for all rotations.

A technique similar in spirit to that of Sadeghipour et al. is that of agent tracking [100], which uses a virtual agent's own BDI plan to recognize a BDI plan being executed by another agent. And similarly, this approach stores plans, rather than utilize a planner as we do.

7 Experiments

”If you cannot fail, you cannot learn.”

Eric Ries

We wanted to evaluate *Mirroring* as a recognition technique inspired by mirroring processes hypothesized to take place in human brains. In order to do that we had to contrast the performance of the *Mirroring* algorithm with human recognition performance. We did this in the two vastly different, continuous domains presented in Section 5.1; sketch recognition and navigational goal recognition.

7.1 Shape Recognition

In order to contrast the Mirroring algorithms' recognition performance with that of humans we again utilized the shape planner in order to recognize regular geometric shapes. Here the task is to recognize 2D hand-drawn regular polygons. We had three people draw (by hand) equilateral triangles, squares, pentagons, hexagons, septagons, and octagons, for a total of 18 drawings. Shapes were drawn in various scales and rotations. Naturally, hand drawings, even under ideal conditions, reflect quite a bit of inaccuracy. Each of the 18 drawings was revealed one edge at a time, with the goal of correctly identifying the goal shape, i.e., there are 18 online recognition problems. Observations, of the edges were generated by using machine vision to analyze the drawings.

The key to the experiment is to utilize the same hand-drawn inputs for both human recognition as well as machine recognition. The performance of the recognizer (and humans) on this data can be potentially used to generate two types

of insights. First, by noting failures and successes in specific cases, we can learn about recognition capabilities and weaknesses. Second, by contrasting human and machine recognition, we can make some deductions as to how humans do or do not carry out the recognition process.

The player we utilized within the recognition process was a pre-developed shape-drawing planner, which takes a partial drawing (as an initial state), and a goal shape type, and attempts to complete the drawing to the goal shape (or report failure if cannot be done, e.g., attempting to complete a 4-edge open polygon into a triangle). To rank hypotheses (see Chapter 4, Algorithm 2, line 10), we looked at the ratio between the ideal internal angle size for the goal shape, and the mean observed internal angle.

Since planners are not designed to accept observational history as input, part of the work of the recognizer, before utilizing the planner, is to incorporate the history of the observations into the planners input. In the current instance the recognizer adjusts the current goals given the history of the observations.

We implicitly fold the observation history O_i into g_x by creating a new goal, g'_x, that removes from g_x edges already seen, and is comprised only of the remainder of the polygon,i.e., the part expected to be completed if the observations $O_0 \dots O_i$ are to be a part of g_x.

For regular polygons, computing the polygon remainder involves calculating the expected angles in vertices, and the expected size of each remaining edge. Under ideal conditions, all edges already observed are equally-sized, and all observed angles are identical. In reality, however, inaccuracies in the drawing of shapes leads to edges that are not all the same size, and shapes that similarly are not ideal. Because of this, the recognizer must make some assumptions in its prediction of how the polygon will be completed (i.e., in what actual edge sizes and internal angles will be utilized).

We chose an optimistic heuristic for this assumption. We ignore the length of observed edge, and instead divide up the remaining angles equally among the remaining vertices. As the angles are thus fixed, and the open ends of the polygon are known, the edge sizes become fixed.

Thus, the planner accepts a goal shape, and returns a plan—a set of edges—

that will complete the drawing of the goal shape, from the initial state (or it may return a result that indicates no plan is possible). By iterating over all possible goal shapes, one can systematically check all possible shapes (out of those still not ruled out), for each new observation.

The last step in the Mirroring process is to rank the possible goals. We ranked the goals based on errors, when compared to the ideal goal shapes. The idea is to measures similarity according to the differences in edge relations and in overall angles comprising the shapes. The shape with the minimal amount of difference is ranked highest.

Incidentally, this use of error in angle, specifically, as the ranking criteria, seems to also agree with studies of human estimates of intentionality and intended action [13]. Such studies have shown a strong bias on part of humans to prefer hypotheses that interpret motions as continuing in straight lines, i.e., without deviations from or corrections to, the heading of movements.

Human recognition data collection Using these images we then conducted a human recognition experiment in order to collect data about human recognition performance. 20 human subjects (14 men and 6 women ages 19–52, with a mean age of 29) participated in the experiment. The shapes tested were the following regular polygons : equilateral triangle, square, pentagon, hexagon, septagon, octagon, as displayed in Figure 5.3. The participants were instructed to observe each edge and then to answer the following questions (after watching each edge), all using software built for this purpose (Figure 7.1).

1. Which *one* shape do you think it is ? Only one option must be chosen.

2. Which other shapes could it be ? The participants were asked to rank, in consecutive order, the remaining shapes they thought likely. This field may be left empty.

3. Which shapes it definitely could not be ?

The participants were asked to take into consideration that the shapes may appear in any size and rotation and that, as the shapes were drawn by humans, may

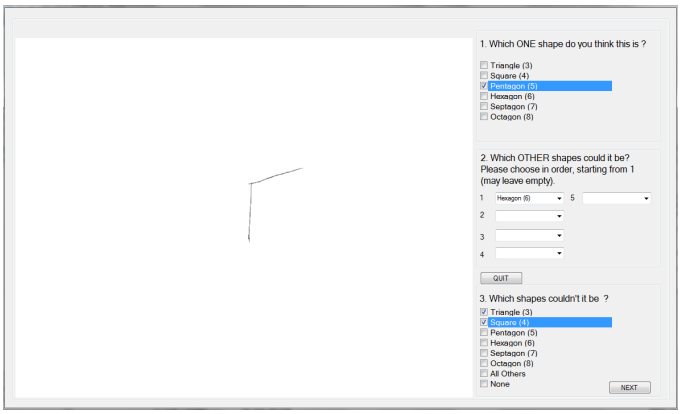


Figure 7.1: Human subjects experiment Interface.

be drawn inaccurately. It was also explained to the participants that questions 2 and 3 were not necessarily complimentary in the sense that in question 2 one might pick one other shape that you seem most likely and in question three you may enter that all shapes were possible or only some of the remaining shapes.

This input, along with the initial and end point coordinates (the anchor points), were fed to the recognizer. To be able to test the significance of each of the major components we divide and contrast the *Mirroring* algorithm into two separate instances :

- *The Ranking Recognizer*. This is the complete *Mirroring* recognizer described above that proceeds to rank the recognition results as explained in Chapter 3.

- *The Non-Ranking Recognizer*. The *Mirroring* recognizer that does not rank hypotheses at all; i.e., no ordering on the results, all possible goals have an equal chance of being chosen.

All the data (human subject results, ranking recognition results, non-ranking recognition results) was initially examined separating each polygon goal (i.e., all the data for triangles was separated from the data for squares). This is because necessarily, the number of observations in the observation stream O differs between these. A triangle has a maximum of three observations; a septagon has

seven. Throughout the experiment humans had immediate access to the goal library; they were shown the possible goals at all time, visually, so they did not have to rely on memory. After each observation was revealed, human subjects were asked to provide a ranking for the goals, and to rule out any goals which they felt were no longer possible. We naturally examine the results for each question separately.

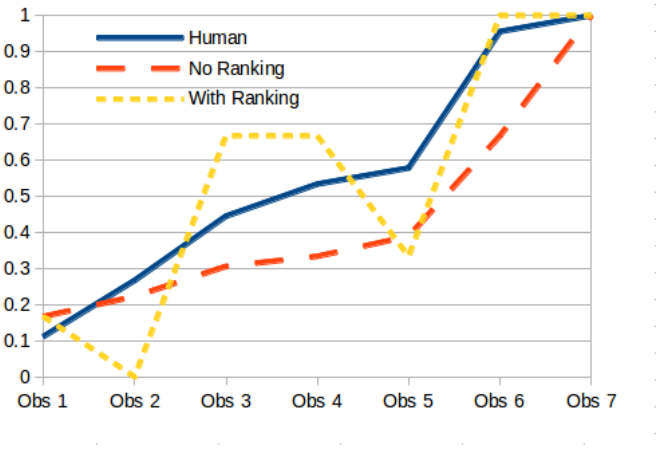
Question 1. Which shape is it? Question 1 allows the recognizer (machine or human) only a single guess as to the goal shape. Thus this is a conservative test of accuracy, as the guess either matches or does not match the ground truth.

To illustrate the progression of a recognition process as observations accumulate, Figure 7.2a shows the results for question 1, for septagons. The figure contrasts the performance of the non-ranking recognizer, with that of the ranking recognizer, with the mean performance of the human subjects. The horizontal axis in Figure 7.2a counts the incrementally accumulating observations (edges). Thus the marking "Obs 3" denotes three edges that are revealed to the recognizer. The vertical axis measures success, as the ratio of correct guesses to the total number of guesses: 0 means the recognizer in question never succeeded in guessing the correct shape at the given point, 1 means it always did. In the case of non-ranking recognition, which cannot choose a top hypothesis, the statistically expected success rate for random selection is used ($1/k$ for k choices).

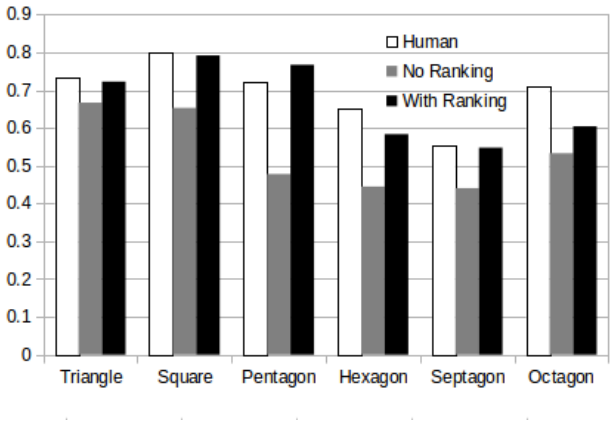
The figure shows clear monotonically increasing success for both human recognition as well as the non-ranking recognizer, with the human recognition consistently better than the non-ranking recognizer. The monotonic behavior of the graph reflects the fact that with each observed edge, less goal shapes are possible, and thus can be ruled out. The ranking recognizer performs inconsistently. For two observed edges, it consistently ranks the ground truth hypothesis (the septagon) below the top, likely because the average angle errors in the drawings, and thus *never* succeeds at this conservative test, with only two edges visible. However, with more edges becoming observable, it can (and does) perform better than human recognition.

Figure 7.2b shows the results for question 1, over all shapes. To "normalize" for the different number of observations, we examine *the convergence rate* by

looking at the area-under-the-curve for each line, for any shape, and divided it by the number of observations. Successful results, early on, result in high convergence values.



(a) Convergence achieved for the septagon shape along all observations.



(b) Mean convergence achieved for each shape, by each recognizer.

Figure 7.2: Question 1 results. Higher values are better.

Figure 7.2b also shows clearly that human and ranking recognition are superior to non-ranking recognition, in all shapes but triangles. On close examination of triangles, it turns out after observing two edges, the ranking procedure was in fact generating better rankings, but was consistently putting triangle (the correct hypothesis) in the second rank. So its score there was 0 for two edges, while

the non-ranking recognizer was expected statistically to be correct at least part of the time, and thus scored better. In all other shapes, the ranking recognizer performed on par with human recognition success, slightly below it (and for pentagons, slightly above it).

To evaluate the relationship between the human results and the Mirroring recognizers' we performed a z-test comparing the human results to both the ranking recognizer results, Figure 7.3a and the non-ranking recognizer results, Figure 7.3b.

	Human vs. Ranking Recognizer		
	Question 1	Question 2	Question 3
Triangles	0.1846305185	0.9598504315	0.1506408531
Squares	0.3193511582	0.9998503387	0.017189658
Pentagons	0.650604497	0.9939619089	1.7729465635E-005
Hexagons	0.0028643917	0.999827136	2.3553546891E-009
Septagons	0.3885795877	0.2922368636	1.6929433364E-006
Octagons	0.000003899	1.092129719993E-010	1.7653697948E-009

(a) Compared to ranking recognizer.

		Human vs. Non-Ranking Recognizer	
	Question 1	Question 2	Question 3
Triangles	0.0083263085	0.9853769938	0.1506408531
Squares	1.20448661E-007	0.9999999979	0.017189658
Pentagons	0	1	1.772946563E-005
Hexagons	1.70419234E-014	0.9999999922	2.355354689E-009
Septagons	2.03179482E-005	0.9998563512	1.692943336E-006
Octagons	4.30211422E-014	0.0051791896	1.765369795E-009

(b) Compared to non-ranking recognizer.

Figure 7.3: Z-Test values recognizers vs human recognition success.

When choosing a significance level of 5% we can see that the values agree with the qualitative analysis of Figure 7.2b. For the first question we will reject H0 for the triangle, hexagon and octagon shapes showing a significant difference between the ranking recognizers' results and the human results. However, for the non-ranking recognizer we will reject H0 for all of the values.

Finally, we also evaluated the results in terms of the percent of the shape that had to be disclosed to the user before a definite identification, shown in Figure 7.4. Here, a lower value is better. The figure shows a clear superior performance of the ranking recognizer over human recognition and over the non-ranking recognizer, in all cases except hexagons, where the human and ranking techniques are essentially equally successful.

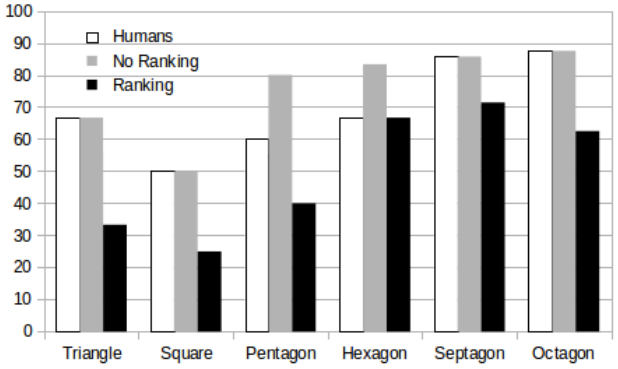


Figure 7.4: Percent of shape uncovered before identification. Lower percentage is better.

Question 2: How likely is the shape? We now relax the test of accuracy a little. Rather than the conservative success/failure test of question 1, we now ask the recognizers to provide a ranked ordering of the hypotheses, and mark the rank of the correct hypothesis in their response. Thus putting the correct hypothesis at the top of the list generates a score of 1, and putting it at the bottom of list generates a score of 6 (i.e., here, a lower score is better).

While the ranking recognizer techniques always provide full rankings, the human subjects did not. Human subjects sometimes did not rank a shape, when they deemed it unlikely (but not necessarily impossible—which is why the answers to questions 2 and 3 are not complementary for humans). We thus differentiate between two cases where the correct hypothesis did not appear in the human subject's answer to question 2:

- The correct shape was not ranked, and explicitly stated in question 3 as a shape that was not a possibility. This was marked as an error. We show these errors separately in Figure 7.6 below.

- The correct shape was not ranked, but also not chosen in question 3. In this case we assumed the human allowed for its possibility, but dismissed it as unlikely (but not impossible). We then use the statistically expected rank of the correct shape in the fully-ranked list, over all combinations where

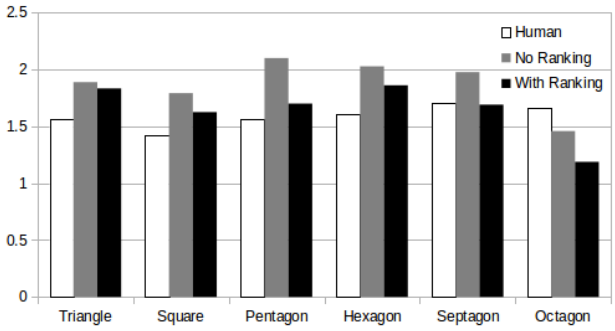


Figure 7.5: Question 2 mean ranking convergence.

it occupied non-ranked slots. For example, if the participant ranked three other shapes as possible but not the correct shape, then the correct shape would be given a rank of $(4+5+6)/3$.

For the non-ranking recognizer, we used the statistically expected rank, given all possible orderings of the hypotheses. For example, if the recognizer ranked two shapes as possible (with the correct shape as one of them), the new ranking of the correct shape would be calculated as $(1+2)/2$. If the correct shape was not ranked it was necessarily chosen as impossible and regarded as an error.

Figure 7.5 presents the average convergence of the ranking score of each shape along all of the observations received. The vertical axis marks the convergence (lower score here is better). The figure shows that human recognition stays fairly consistent and is mostly superior to both ranking and non-ranking recognition methods. For the Octagon shape humans perform less well than both recognizers raising future questions relating to humans' perception of larger shapes in general. We can also see that the ranking recognition was ultimately better than the non-ranking recognition.

From the Z-Test results displayed in Figures 7.3 we can see that with a significance level of 5% we don't reject H0 for either the ranking and the non-ranking algorithms establishing a clear connection between the results.

As mentioned before we excluded the errors from our result analysis and addressed them separately. Figure 7.6 shows the errors made by all three recognizers, i.e., correct hypotheses that have been actively disqualified by the recognizer.

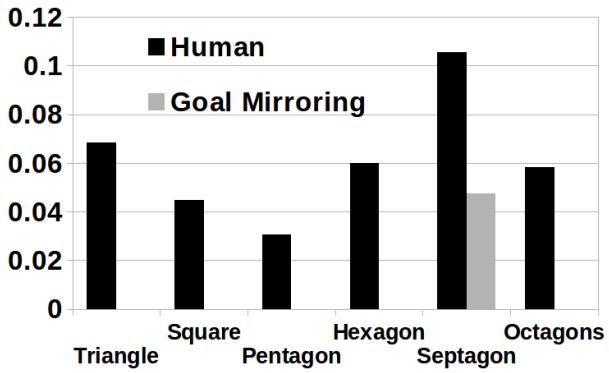


Figure 7.6: Average error ratio [0-1].

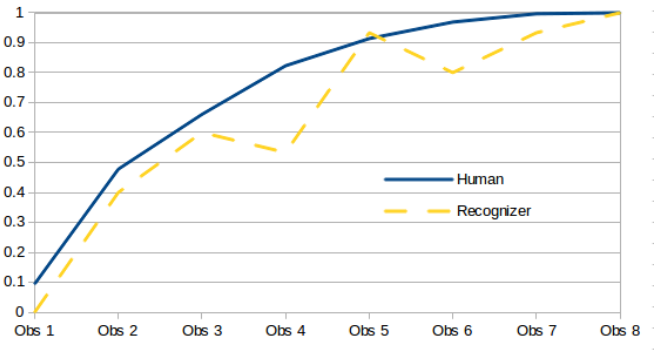
With the exception of septagons, only humans make such errors, as the results demonstrate. This resulted from a great deal of noise in the septagon shapes, expressed as a large difference in expected angle size. Furthermore, even in septagons, the machine shape recognizers make far less errors than humans.

Question 3: What shape couldn't it be? In the final question, we examined a separate factor that can be utilized to improve recognition. While questions 1 and 2 focused on positive recognition—the ability to correctly guess (hypothesize) at the correct goal shape, question 3 addresses negative recognition—the ability to rule out hypotheses from consideration. The two, as explained earlier, are not complementary. For instance, one could rule out 4 of 6 hypotheses, and still completely fail on questions 1 and 2 (ranking the correct hypothesis second).

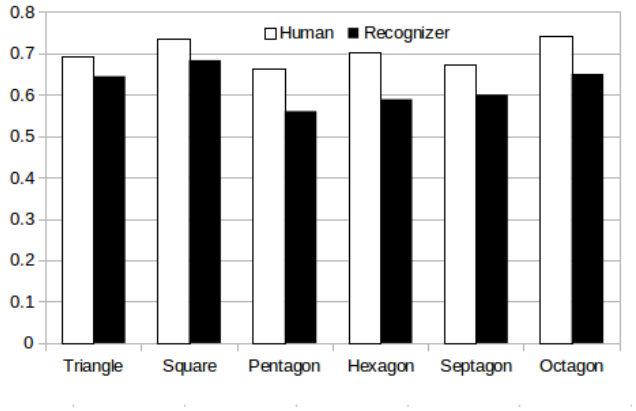
We measure performance in this question as follows. We note how many of the incorrect shapes (necessarily, 5) are ruled out in the response to question 3. This is specified in ratio form, with the worst score being $0 = 0/5$ when no shapes are ruled out, and the best score being $1 = 5/5$. Thus a higher score (1 is the maximum) is better. And as before, with less observations this is better. Thus we again utilize the convergence measure.

For example, Figure 7.7a shows the results for octagons. The horizontal axis measures the number of observed edges. The vertical axis marks performance score (high is better) as described above. Because ranking has no significance in this particular question the calculation for the performance of both the ranking and

non-ranking recognizer is identical. Therefore we utilize only one line (marked arbitrarily as “Recognizer”) to denote their performance in the figure.



(a) Mean convergence of incorrect shapes for each observation in the octagon shape.



(b) Mean convergence achieved in question 3 for each shape.

Figure 7.7: Question 3 results. Higher values are better.

There are several interesting observations based on this figure. First, the machine recognizers surpass human performance at one point (5 observed edges), but otherwise offer inferior (if close) performance to human recognition. Thus there is, apparently, the possibility that humans are not disqualifying hypotheses in these stages, even when they are in fact no longer relevant. Second, humans also make another kind of mistake, where they make an incorrect disqualification of an hypothesis with the first observation. Necessarily, observing a single edge, no shape hypothesis can be disqualified. Yet humans tend to jump to conclusions, eliminating at least one hypothesis from being considered.

Figure 7.7b presents the mean convergence for each shape in respect to Question 3. Clearly, human disqualification is faster in all cases—but this result should be taken with some caution, as we note that such early disqualification of hypotheses results in the errors discussed above (in addressing question 2).

The Z-Test results displayed in Figures 7.3 again agree with the convergence results. And we can see that with a significance level of 5% we reject H0 for septagons, hexagons and octagons.

7.2 Navigation Goal Recognition

Here the recognition task is to identify the goal location of an object observed moving in a 3D continuous world. The observations are made incrementally, as a sequence of way-points reached by the object as it moves. To carry out the experiments we again utilized the Open Motion Planning Library (OMPL; [98]), in particular its *cubicles* 3D environment, the default robot, and the TRRT planner that comes with OMPL (see Section 5.1). The cost measure used for the ranking procedure (Algorithm 2, line 10) is simply the length of the path.

We selected six points spread out over the environment as the goal set, G. Four of the points were chosen arbitrarily over the larger visible surface of the cubicles environment, the same surface where the observed object starts; these points were picked that they might be intuitive or easy goal positions for humans. Two additional points were specifically chosen to be harder, based on some pilot studies. One point requires humans to think about the other side of the environment (a point on the "other floor" of the environment, point F). The other point hangs in mid-air in the opening between the two "floors" of the environments, point E. The points are shown in Figure 7.8 and labeled in alphabetical order.

To generate recognition problems, we generated paths from a fixed starting position to all six goal points. Two paths were generated for each goal, for a total of 12 problems: one path generated by the asymptotically-optimal RRT* algorithm implemented in OMPL [98], and another hand-modified to deviate from this optimal path by taking a longer route. The motivation for generating such paths is to examine recognition performance with observations of non-optimal plans, which test the rationality assumption of Mirroring.

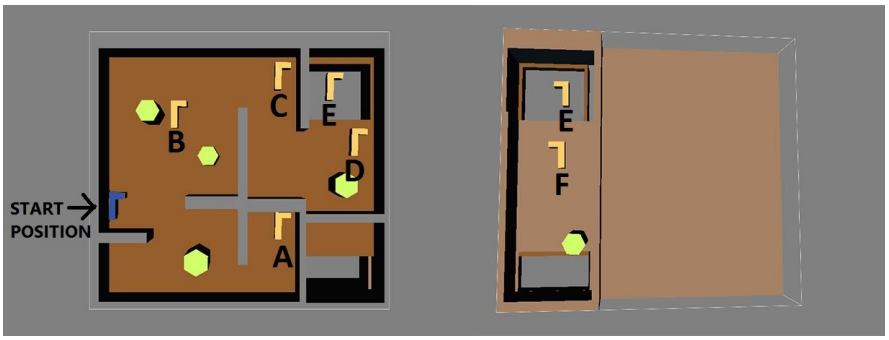


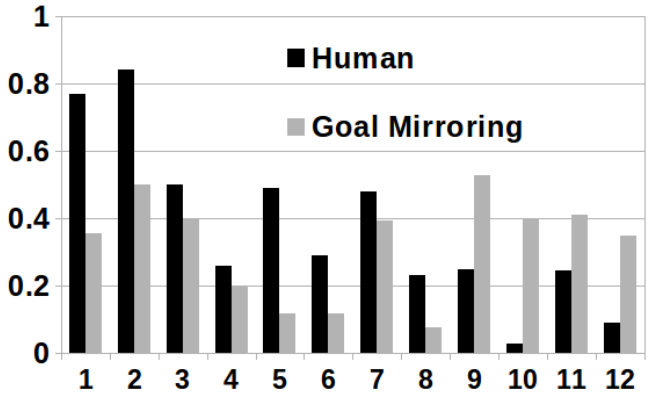
Figure 7.8: ***Cubicles*** **environment.**

we tested 19 human subjects (8 women; ages 17–51, mean 27.5). Results for this domain are shown in Figure 7.9; the X-axis denotes the 12 paths, organized in pairs: 1–2 for goal point A, 3–4 for goal point B, etc. Here again, humans had immediate access to the goal library; they were shown the possible goals at all time, visually, so they did not have to rely on memory. After each observation was revealed, human subjects were asked to provide a ranking for the goals, and to rule out any goals which they felt were no longer possible. the participants were instructed to assess each observation and then rank the leading goal hypothesis.

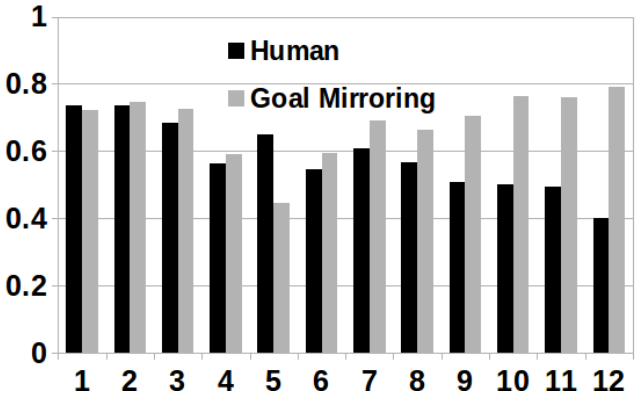
Optimality Assumption As discussed in Chapter 3, generating a candidate hypothesized plan that matches the observations is only a necessary step in plan recognition. Multiple such hypotheses exist and can be generated, and the key is to determine a sufficient condition for selecting between them. In library-based methods, a plan can be a solution if it can be found or inferred from in the library of plans. But in planner-based methods, where candidates are generated from scratch, dynamically, a different approach is needed.

Previous works [81] have proposed to assume that the observed plan is carried out by a (bounded) rational agent, which means that the observed plan should approximate the optimal plan for getting the initial state to the goal state. Recognizers based on this assumption prefer hypothesized plans that better match the optimal plan.

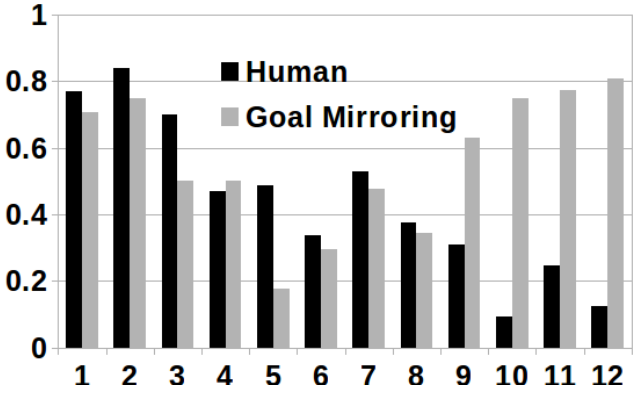
To evaluate this, we purposely generated two different observed paths to each possible goal in the 3D navigation domain, as described above. One path was the



(a) Convergence



(b) AUC

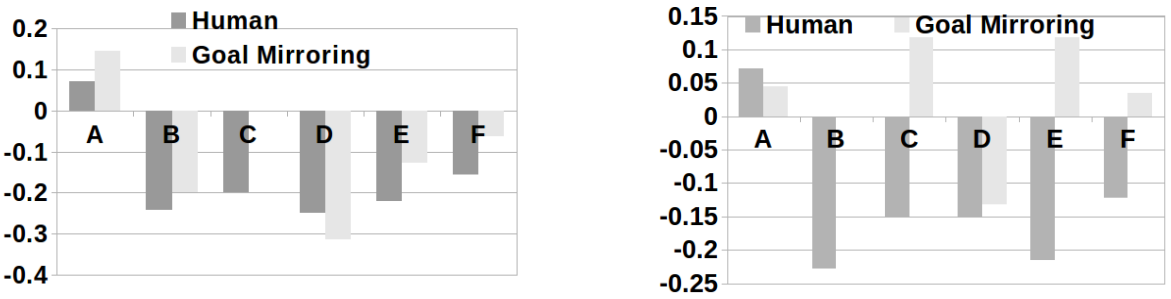


(c) RankedFirst

Figure 7.9: Recognition results in the 3D navigational domain.

optimal. The other was modified from it by introducing a detour which made it longer, though still smooth and executable by the moving object. By comparing the performance of Mirroring and humans on each of these two paths we hope to gain insight as to the importance of the rationality assumption in recognition.

Figure 7.10 presents the deterioration in both human and Mirroring recognition when observing an optimal and a non-optimal plan (path) to the same goal, across the three criteria. The X-axis in each figure shows the goal. The bars mark improvement or degradation: values that are larger than zero indicate an improvement in performance when observing the non-optimal plan. Values smaller than zero indicate degradation in recognition. Higher values are better: Larger positive values indicate more improvement; smaller (closer to 0) negative values show less degradation.



(a) Convergence

(b) RankedFirst

Figure 7.10: Performance degradation between optimal and non optimal paths.

The figure shows that overall, Mirroring deteriorates less, or improves more, than humans. This happens across all goals except D, for the convergence measure (Figure 7.10(a)) and for all goals but one (A) for the ranked-first measure (Figure 7.10(b)). The results are especially striking in the ranked first measure, where sometimes Mirroring results positively improve when observing non-optimal plans, while humans results degrade.

From the perspective of building automated recognizers, these are promising results, demonstrating the robustness of Mirroring. From a perspective of human cognitive modeling, these results hint at a very strong reliance of humans on the

rationality of the observed plan. As a result Mirroring, as presented here, is still not a good enough model of human recognition capabilities. Coupled with the results of the previous section we see that overall, Mirroring is more conservative, risk-averse, while humans, by comparison, are risk-seeking, committing early and assigning more weight to rationality.

7.3 Summary

We presented Mirroring within the context of shape recognition, where by a planner is re-used by a recognition process, allowing drawn-shape recognition by drawn-shape planning. The approach has a number of technical advantages specific to recognition (such as no plan library and fast on-line computation with no pre-processing), but most importantly, is particularly suited to agents, where a complete agent is expected to have a planner for its own goals, and this can be utilized for recognition, without the need for a separate source of recognition knowledge.

We instantiated the shape recognition approach in the recognition of regular polygons, and evaluated the performance of different ranking and non-ranking variants of the recognizer against human subjects' recognition of scanned hand-drawn regular polygons. The evaluation utilized several different evaluation criteria. Across the board, the ranking recognition proved superior to the non-ranking recognition. In some cases, the ranking recognizer surpassed human recognition results (e.g., it required less of the polygon to be observed before settling on the correct response). However, in general the ranking recognizer performed on par, or just below, human levels of recognition. Through one of the evaluation tests (question 3) we show that humans make negative recognition mistakes, both in disqualifying hypotheses too early, or in holding on to them even once it is proven they are incorrect. However, it might be that the tendencies leading to these mistakes might also account for the better performance of humans.

As stated, the planner's input is comprised of the anchor points (initial and ending open ends of the polygon) and line parameters (slope and intercept). Because of this, translating shapes in 2D space, and rotating them, are completely transparent to the recognizer. Indeed, the results presented here are based on

shapes drawn by hands in various scales, rotations, and translated. Thus we expect the Mirroring approach to be particular scalable to realistic scenarios, where these transformations are to be expected.

In future work, we hope to study the differences between human and machine recognition, especially in the ability to disqualify hypotheses early on. We believe that the biases that humans exhibit, when understood properly may be exploited and may prove highly useful for agents working closely with humans.

Part III

Mirroring Efficiently

8 Heuristic Online Mirroring

”Simplicity is the soul of efficiency.”

Austen Freeman

In the following section we proceed to further develop *Mirroring* into a general, *heuristic* algorithm for online recognition in both continuous and discrete domains. Mirroring now solves the plan-recognition problem with a baseline of *at most* $|G|(|O|+1)$ planner calls and a new *best* of only $|G|$ planner calls. We achieve this by identifying two key decision points within the Mirroring process where heuristics may be inserted in order to further reduce the number of calls made to the planner and consequently the overall run-time.

The first decision point questions whether to generate and solve a new planning problem for each goal and for each observation. If not, we may choose to remain with the previously calculated plans. The purpose of this heuristic is to refrain from calling the planner when unnecessary, again reducing the overall run-time of the recognition process. We will show that, in the best case, this heuristic may reduces the number of overall calls to the planner from $|G|(|O|+1)$ to only $|G|(1+1)$.

The second decision point questions whether to prune unlikely goal candidates from the set of all possible goals. It aims to decrease overall run-time by reducing the number of times the planner is called. It achieves this by pruning out goals that seem unlikely or impossible thereby incrementally reducing $|G|$ as observations are added and making fewer calls to the planner for each observation.

We describe the improved Mirroring algorithm in detail, instantiate the two heuristics for recognition of navigation goals in continuous 3D environments and

additionally show a novel method of pruning out goals using a combination of Mirroring and disjunctive landmarks. In order to evaluate the approach we have contrasted *Mirroring* recognition performance and efficiency, with and without the aforementioned heuristics, over hundreds of experiments in continuous and discrete environments and several different challenging domains. We empirically show that these heuristics give place to a considerable improvement in performance.

8.1 Minimum Plan Generation

We would like to begin with presenting one possibility that immediately comes to mind for improvement on the baseline *Mirroring* algorithm introduced in Chapter 3. We are referring to the possibility of re-using the optimal plans. As a reminder, the optimal plans, $\hat{\pi}_g$, describe the ideal plan from the initial state to each of the goals $g \in G$. We have previously used these plans as the standard against which other generated plan hypotheses have been compared, enabling us to rank the hypotheses and come up with a leading goal candidate, r.We are now referring to re-using these previously calculated plans as the general plan hypotheses, π_g^O, against which we will compare all incrementally revealed observations.

In this manner we are taking an extreme approach of no plan recalculation at all. This will result in only $|G|$ calls to the planner to calculate the initial $|G|$ plans, one for each corresponding goal. We first generate all of the optimal plans, $\hat{\pi}_g$, and then match incoming observations against them. These plans are saved and will remain unchanged even as more observations come in. Instead of computing π_g^O for each $g \in G$, each new observation is assessed against the original $\hat{\pi}_g$. Whichever goal, g is closest to the observation is picked as the top ranking goal. Note that we use the term "closest" to indicate that the cost towards achieving it is smaller and not necessarily a geometric evaluation of distance.

In the best case, when the observations closely match the originally calculated plans, this approach will work very efficiently. However, realistic conditions may not favor the best case scenario. For example, the observations may contain a certain amount of noise. Or, in cases where the observed agent is not perfectly rational, its observed trajectory will deviate from the saved $\hat{\pi}_g$ plans.

Even worse, it could be that the observed agent is perfectly rational, there is no noise in the observations, and yet the approach will fail. This is due to cases where there are multiple perfectly-rational (optimal) plans, which differ from each other but have the exact same optimal cost, see Figure 8.1. The figure illustrates a scenario in which we have two optimal plans (illustrated as the red and green lines) from position I to goal G due to a static obstacle in the environment, illustrated as the black rectangle. In such cases, it is possible that the planner used by the recognizer will generate an optimal plan, $\hat{\pi}_g$ (red line), which differs from an equivalent—but different—optimal plan $\hat{\pi'}_g$, (green line), carried out by the observed agent and the observations will not match.

As this approach does not make use of the information available in each observation, while it may prove highly effective it must cost much in performance. In Chapter 10 we will empirically show that for realistic conditions, consisting of the use of a time constrained, asymptotically optimal planner to generate the observations and a time constrained, suboptimal planner in the recognition process, this approach does not work well.

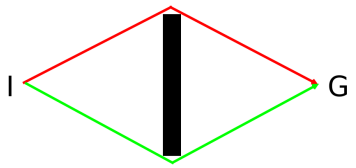


Figure 8.1: Two optimal plans.

8.2 Heuristic Mirroring

We hereby present the *Heuristic Mirroring* algorithm, an efficient online goal recognition algorithm for continuous domains which works according to the guidelines of PRP and is inspired by mirroring processes hypothesized to take place in the human brain. We identify two key decision points in the baseline recognition process described above, that can be used to improve its efficiency. These are :

- *Recompute* : Recompute plans only if necessary, i.e., if the new observation may change the ranking (captured by a $RECOMPUTE$ function); and
- *Prune* : Eliminate goals which are deemed impossible or extremely unlikely (as they deviate too much from the ideal plan $\hat{\pi}_g$), (captured by the $PRUNE$

function).

A good *RECOMPUTE* heuristic reduces calls to the planner by avoiding unnecessary computation of π_g for new observations. A good *PRUNE* heuristic reduces calls to the planner by eliminating goals from being considered for future observations and reducing $|G|$. By inserting domains specific heuristics into these decision points we can reduce the number of calls made to the planner and consequently overall recognition run-time. This section presents the general heuristic algorithm. The next section will examine candidate heuristics for specific domains.

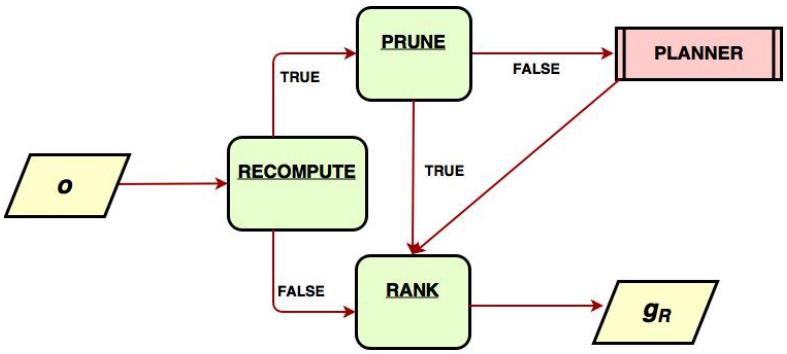


Figure 8.2: Diagram describing *Mirroring* decision cycle.

Figure 8.2 represents the general decision cycle of *Mirroring* while emphasizing the aforementioned decision points. The incoming observation $o \in O$ reaches the first decision point, *RECOMPUTE*. Here we determine if we need to recompute the suffix plan. If there is no recomputation necessary the process will then skip all other steps and rank the possible goal hypotheses according to existing knowledge alone. Otherwise the process enters the second decision point, *PRUNE*. Here we determine for each goal if it may be removed from the set of all possible goals. If the goal is pruned we have no need to calculate further for that goal. Only if both decision points have failed; i.e. we needed to recompute π_g and did not prune out any goals, we will utilize the planner in order to generate more knowledge for the ranking procedure following which we will have the leading goal candidate, r.

Algorithm 3 presents the general, heuristic, online Mirroring procedure. The

Algorithm 3 HEURISTIC MIRRORING $(R := \langle W, O, I, G \rangle, planner)$

1: $\forall g \in G : \pi_g^O, p \leftarrow \emptyset$ ▷ generate initial plan hypotheses
2: $G' \leftarrow G$ ▷ Save goal set in case of backtracking
3: $r \leftarrow \emptyset$ ▷ top-ranked goal
4: **for all** $g \in G$ **do**
5: $\hat{\pi}_g \leftarrow planner(I, g)$ ▷ generate optimal plan from I to g
6: $\pi_g \leftarrow \hat{\pi}_g$ ▷ default value for plan suffix
7: **while** new $o \in O$ available **do**
8: **if** $RECOMPUTE(\pi_g^O, o, r)$ **then** ▷ Accepts the set of all plan hypotheses
9: **for all** $g \in G'$ **do**
10: **if** $PRUNE(\pi_g, o, g)$ **then**
11: $G' \leftarrow G' - \{g\}$
12: **else**
13: $\pi_g \leftarrow planner(o, g)$
14: **else**
15: **for all** $g \in G'$ **do**
16: $\pi_g \leftarrow \pi_g \ominus prefix(o, \pi_g)$ ▷ Updating the suffix plan to exclude the last observation
17: **for all** $g \in G$ **do**
18: $p \leftarrow p \oplus o$ ▷ update obs. path to include last seen obs.
19: $\pi_g^O \leftarrow p \oplus \pi_g$ ▷ compose candidate plan hypothesis for goal g
20:
21: $P(g|O) = \frac{cost(\hat{\pi}_g)}{cost(\pi_g^O)}$ ▷ calculate probability for goal g
22: **for all** $g \in G$ **do**
23: $P(g|O) \leftarrow \eta \cdot P(g|O)$
24: $r \leftarrow \text{argmax}_{g \in G} P(g|O)$ ▷ the plan that maximizes $P(g|O)$ will be selected as the leading plan hypothesis and the corresponding goal r as the leading goal candidate

general flow is the same as the baseline algorithm (Algorithm 2), with the following changes. We need to maintain the current top goal hypothesis, denoted as r (line 3). This will later be used to assess whether a new observation $o \in O$ will cause a change in its ranking as the top hypothesis (line 8).

We begin (lines 4–5) by computing the optimal plan $\hat{\pi}_g$ only once for all possible goals, as was done in Algorithm 2. However we now also set the optimal plan for each goal, $\hat{\pi}_g$, as the default plan suffix π_g (line 6). This suffix guarantees that *valid* (though not necessarily optimal) plan hypotheses, π_g^O can be created from π_g, even in the extreme case where no computation of π_g is ever done (as was described in Section 8.1). By using the term *valid* plans we are referring to plans that agree with the observations already seen (see chapter 3).

The main loop begins on line 7, iterating over observations as they are made available. We then reach the first decision point addressing whether we need to recompute the suffix plan, π_g, or not. This assessment is carried out heuristically by the heuristic function $RECOMPUTE$ in line 8. The $RECOMPUTE$ function takes as input the formerly calculated set of all plan hypotheses (for all $g \in G$) , the latest observation o and the current leading goal, r. It matches the observation to π_r (the currently leading plan hypothesis, part of the set π_g^O) and heuristically determines (see next section) whether the new observation o may cause a change in the ranking of the top goal. If so, then the suffixes π_g of all goals (lines 3–13) will need to be recomputed (line 13) , unless pruned (line 11) . Otherwise (lines 14–16) the current suffix π_g of all goals will be modified based on o, but without calling the planner.

Recomputing the suffix plan, π_g (line 13). Here a straightforward call to the planner is made per the discussion in chapter 3, to generate an optimal trajectory from the initial point, the *last* point in o (as o might contain more than a single point), to the goal g.

Modifying π_g (line 16). When no recomputation of the suffix plan is deemed necessary the last seen observation, o, will be added to the prefix p. Additionally the existing π_g must be updated so that it continues p and leads towards g. The baseline algorithm calls a planner to do this but the point of $RECOMPUTE$

heuristic is to approximate the planner call so as to avoid its run-time cost. This is done by removing (denoted by $\ominus$) any parts that are *inconsistent* with respect to the observation from the beginning of the old suffix π_g.

The previous π_g begins where the *former p* (without o) ends. The new π_g should ideally begin with the last point of the *new p* (which is the new observation o), and continue as much as possible with the former π_g. Thus a prefix of the old π_g, denoted ($prefix(o, \pi_g)$, line 16) is made redundant by o and needs to be removed. If o is a direct part of π_g, then $prefix(o, \pi_g)$ will be the exact trajectory from the beginning point of π_g to o. However, in general we cannot expect o to be directly part of π_g. We thus define the ending point for the prefix to be $\widetilde{o}$, the geometrically closest point to o on π_g. If indeed no recomputation is carried out, the hypotheses remain at their current ranking and we loop back to line 7 to await a new observation.

An ideal scenario would be for Alg. 3 to never compute a new suffix π_g, for any goal. This would correspond directly to the *Minimum Plan Generation* approach presented in Chapter 3, Section 8.1. In line 6 of Alg. 3 the initial suffixes π_g are set to the ideal plans, $\hat{\pi}_g$. If $RECOMPUTE$ is *always false*, then no new planner calls will be made and π_g will be incrementally modified (Alg. 3, line 16) to accommodate the observations.

Theorem 5. *If* $RECOMPUTE(\pi_g^O, o, r) = false \; \forall o \in O \Rightarrow$ *there will be exactly* $|G|$ *calls to the planner.*

Proof. In order to prove Theorem 5 we refer to Alg. 3. Initially a single call to the planner will be made to calculate $\hat{\pi}_g$ for every goal for a total of $|G|$ planner calls (line 5). As we can see for every observation $o \in O$ the $RECOMPUTE$ method is called in line 8. If $RECOMPUTE$ returns $false$ the algorithm skips to line 14 in which the suffix plan, π_g is calculated for every goal $g \in G'$. This calculation is done without calling the planner but by modifying $\pi_{\mathbf{g}}$ as explained above. Following this procedure the goals may now be ranked according to existing knowledge alone (lines 17-24). Since the only planner call occurs in line 13 the algorithm will never call the planner again hence generating exactly $|G|$ planner calls.

□

As Theorem 5 demonstrates this approach offers significant savings and in the best case, when the observations closely match the originally calculated paths, can produce good recognition results.

However, realistically, the observations may contain a certain amount of noise or the observed agent may not be perfectly rational. Moreover, it could be that the observed agent is perfectly rational and there is no noise in the observations and yet the approach will fail. As in the case of the *Minimum Plan Generation* approach (Chapter 3, Section 8.1) this is due to cases where there are multiple perfectly-rational (optimal) plans, which differ from each other but have the exact same optimal cost. In such cases, it is possible that the planner used by the recognizer will generate an ideal plan $\hat{\pi}_g$ which differs from an equivalent—but different—ideal plan π'^{O}_{g} carried out by the observed agent.

In such a case, the incrementally computed π^{O}_{g} may grow to become much more costly from $\hat{\pi}_g$; at any given observation point, because there is no re-planning of π_g carried out, π^{O}_{g} will have a prefix p that matches one optimal plan, and a suffix π_g that jumps from the last point of p to the closet point to it on $\hat{\pi}_g$.

In line 10 we reach the second decision point : for every goal $g \in G$, a second heuristic decision is made as to the maintained relevance of goal g given the latest observation, and the projected plan suffix π_g. This assessment is carried out heuristically, by the heuristic function *PRUNE*. A goal g whose associated plan hypothesis π^{O}_{g} is deemed extremely unlikely, or even impossible, is removed from the list of goals to consider (line 11).

Intuitively, when the newest observation o leads *away* from a goal g (see next section), we may want to eliminate the goal from being considered further, by permanently removing it from G. This is a risky decision, as a mistake will cause the algorithm to become unsound (will not return the correct result, even given all the observations). On the other hand, a series of correct decisions here can incrementally reduce G down to a singleton ($|G| = 1$), which will mean that the number of calls to the planner in the best case will approximate $(|O| + 1)$.

Otherwise, if both heuristics have failed, a recomputation of π_g is carried out by calling the planner, using not just the last observation, but all observations since the last iteration through this inner block of code (lines 13–17). Finally, when the algorithm reaches line 16, a valid suffix π_g is available for all goals in G. For all

of them, it then concatenates the latest observation to the prefix p (line 18), and creates a new plan pi_g^O by concatenating the prefix and suffix (line 19). This means that a new probability may be calculated for each goal (line 21) and a (potentially new) top-ranked play hypothesis π_R may be selected (line 24).

Heuristic Mirroring Theoretical Bound. As previously described Algorithm 3 is an improved version of the *Baseline Mirroring* algorithm (Algorithm 2) introduced in chapter 4. We contend that by varying the heuristic functions used, we can specialize its behavior to be exactly the same as the baseline (Thm. 6), or change its behavior in different ways.

Theorem 6. *If $RECOMPUTE = \top$ and $PRUNE = \bot$ then Algorithm 3 will generate exactly the same number of planner calls as Algorithm 2.*

Proof. In order to prove Theorem 6 we refer to Algorithm 3. Initially a single call to the planner will be made to calculate $\hat{\pi}_g$ for every goal for a total of $|G|$ planner calls (line 5). Then, for every observation $o \in O$ the $RECOMPUTE$ method is called in line 8. Assuming $RECOMPUTE$ to be *always true* we will proceed to line 9 in which we go over all of the possible goals $g \in G'$ and evaluate the $PRUNE$ function. Assuming that $PRUNE$ is always *false* (line 10) we will always execute line 13 in which a new call to the planner will be made to generate π_g for all goals and every observation (since no calls will be skipped and no goals would be pruned). This is in accordance with the behavior of the Baseline Mirroring algorithm (Algorithm 2) first reported in [108] in which $|G|(|O|+1)$ planner calls are made.

□

8.3 Heuristic Recognition of Navigation Goals

We instantiate two novel heuristics applicable to the navigation goal recognition domain to be inserted in the previously described key decision points within the

Mirroring algorithm. These are inspired by studies of human estimates of intentionality and intended action [13]. Such studies have shown a strong bias on part of humans to prefer hypotheses that interpret motions as continuing in straight lines, i.e., without deviations from or corrections to, the heading of movements. Therefore the heuristics are biased toward rational acting agents, at least as this is expressed in 2D or 3D motion plans.

8.3.1 The Recomputation Heuristic

For every new observation we are called to recompute the suffix plan, π_g for every goal $g \in G$. This could result in an additional $|G|$ calls to the planner for every added observation. However we must take into account that for every new observation o we have necessarily calculated the plans for the previous observation, o'. By saving these previously calculated plans we may now consider whether the new observation is in agreement with previously calculated plans. If the observation matches, it means we may continue to rely on former calculated plans and do not need to re-call the planner.

As previously discussed, we cannot realistically expect the observations to perfectly match the predictions. In order to do this the heuristic needs to evaluate to *false* when the new observation o does not alter the top ranked goal g_r (saving a redundant $|G|$ calls to the planner), and evaluates to *true* otherwise. In order to do this the heuristic needs to evaluates the distance between the newly seen observation o and each of the candidate plan hypotheses π_g^O (one for each goal $g \in G$).

A suggestion for such a heuristic is to measure the distances between the new observation o and all previously calculated plans π_g^O. For example, in the domain of 3D navigation this may be achieved by measuring straight lines from the new observation point to the nearest point along each of the various plan hypotheses π_g^O using the *Euclidean Distance* formula. If the distance between the new observation and the plan associated with the currently *highest* ranked goal π_r is smaller than all of the other distances (to other goals) we can assume that the planner is still heading on the same path and do not need to re-run the planner at all, keeping the previous goal rankings.

Formally, let r represent the highest ranked goal so far and π_r be the calculated (top-ranked) plan associated with it. We will proceed to calculate $d_r = dist(o, \pi_r)$ as the distance between the last incrementally revealed observation o and the plan π_r. If $d_r \leq min_{g \in G} dist(o, \pi_g^O)$, meaning the leading goal candidate measures the shortest distance to the new observation as opposed to all the other plan hypotheses, then goal r remains the top-ranked hypothesis and we do not recompute the plans π_g^O, thus avoiding a redundant $|G|$ calls to the planner.

In a best case scenario, where the observed agent generates observations that are perfectly rational and in accordance with one of the hypothesized plans π_g^O, the planner will only be utilized $2|G|$ times; once in the initial path generation ($\hat{\pi}_g$) and once again for the first observation generating π_g^O therefore generating only $2|G|$ calls to the planner in total. However, the heuristic allows for non-best cases, where the paths π_g^O have to be recomputed to account for noisy observations that deviate from the predictions.

8.3.2 The Pruning Heuristic

Finally, we introduce a pruning heuristic for observing rational agents in continuous domains. While calling the planner is wasteful when unnecessary, it is also wasteful to call the planner for goals that are highly improbable—or even impossible—given the observations. Thus, another idea is to prune goals from being considered at all, reducing $|G|$ as observations come in. Here we again rely on the rationality of the observed agent, inspired by studies of human estimates of intentionality and intended action [13]. Such studies have shown a strong bias on part of humans to prefer hypotheses that interpret motions as continuing in straight lines, i.e., without deviations from, or corrections to, the heading of movements. Once a rational agent is moving away or past a goal point g, it is considered an unlikely target.

The *PRUNE* heuristic considers whether to exclude a goal g from further consideration. In the navigation goal recognition domain, we take a geometric approach towards such pruning by examining the angle between the current estimated heading of the observed agent, and the path leading towards g. This is done as follows.

We calculate α_g, the angle created between the previous observation , the newly received observation and the previously calculated plan for every $g \in G$. α_g is calculated using the *cosine* formula, $cos(\alpha) = (\vec{u} \cdot \vec{v})/(||\vec{u}||||\vec{v}||)$, where $\vec{u}$ is the vector created by the previous and new observation and $\vec{v}$, the vector created by the previously calculated plan and the new observation.

Figure 8.3 presents an illustration of the heuristic approach in 2D. For each new observation, o_i, we measure the angle α_i created by the new observation, the previous observation o_{i-1} and the previous plan hypotheses generated for each goal, π_g^O. In Figure 8.3 two such plans are illustrated as the dashed lines between o_{i-1} and each of the two possible goals g_1 and g_2. If the newly calculated angle, α_i, is bigger than a given threshold we deduce that the previous path is heading in the wrong direction and rule out that goal entirely. By defining different sized threshold angles we can relax or strengthen the pruning process as needed.

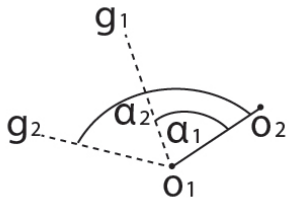


Figure 8.3: Illustration of goal angles.

9 Landmarks as a Pruning Mechanism

> ”The highest type of efficiency is that which can utilize existing material to the best advantage.”
>
> *Jawaharlal Nehru*

In the following section we proceed to instantiate another manner of improving the efficiency of *Mirroring* for online recognition in both discrete and continuous domains. We introduce a novel method of pruning out goals using a combination of Mirroring and disjunctive landmarks. As previously mentioned, in online recognition, observations are provided incrementally and the objective is to recognize the intended goal as soon as possible, without knowledge which observation is the final one. This poses an even harder recognition problem, since the recognition algorithm must cope with a stream of observations while attempting to infer the actual goal with minimal information. Furthermore, most existing approaches assume all observations, even if noisy or incomplete, are received at once (*offline*) at the end of their execution [73].

We now develop an efficient approach for online goal recognition as planning that generalizes over both discrete (STRIPS style) and continuous (navigation) domains. The approach achieves substantial runtime efficiency by reducing the complexity of the problems sent to an underlying planning algorithm using online Mirroring [108, 107] while minimizing the number of goal hypotheses to be computed using landmarks computed once during run-time, and a landmark-based heuristic [71, 73].

To generalize over discrete and continuous domains we adapt the notion of planning landmarks [38] to comprise its original planning semantics as well as continuous spatial domains and develop a new and efficient algorithm to generate spatial landmarks. Since the Mirroring approach can use any type of PDDL [58] planning algorithm or path planner [98], we can leverage current and future advances in efficiency in such algorithms.

We introduce three key contributions: (a) a novel goal recognition approach for both discrete and continuous domains; (b) an online approach to efficiently recognize goals early in the observed agent's plan execution; (c) a novel notion of landmarks encompassing discrete and continuous domains and an algorithm to generate such landmarks. We evaluate the resulting approach empirically over hundreds of recognition problems in classical and motion planning domains. The results show superior efficiency and generally superior recognition quality over the state of the art.

We introduce an online goal recognition approach that combines online *Mirroring* [107] and *recognition using landmarks* [71, 73], which we extend to work online and in continuous spaces. We further develop a procedure for extracting continuous-space landmarks and proceed to combine the approaches into a single recognition algorithm.

9.1 Online Goal Recognition Using Landmarks

In the planning literature, *fact landmarks* are facts that must be true at some point in every valid plan to achieve a particular goal from an initial state [38]. In other words, fact landmarks (or simply *landmarks*) are subsets of the domain W (previously defined in Chapter 3). Landmarks are often partially ordered based on the sequence in which they must be achieved. Formally, a fact landmark is a formula (either a conjunctive or disjunctive formula) over a set of facts that must be satisfied (or achieved) at some point along all valid plan executions. Figure 9.1 shows an example of fact landmarks and their ordering for a Blocks-World example. The root node represents the goal state, whereas leafs are facts of the initial state. Connected boxes represent facts that must be true together, i.e., conjunctive facts. For example, in order to achieve `(on A B)`, the facts immediately preceding it,

`(and (holding A) (clear B))` must be true, and so on.

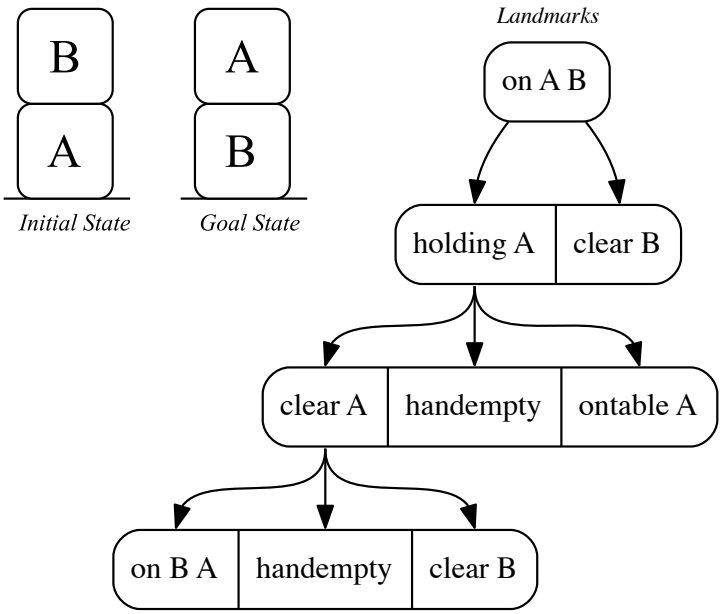


Figure 9.1: Blocks-World landmarks example.

Given their usefulness for planners and planning heuristics [83], research has yielded multiple notions of landmarks [76], including that of disjunctive landmarks. Briefly, disjunctive landmarks represent an exclusive disjunction over possible instances of variables associated to predicates in the state representation. For example, consider that the agent in the Blocks-World domain can pick-up and hold blocks with both arms. In the two-armed blocks world then, moving a block `A` induces a disjunctive landmark `(or (holding A LeftArm) (holding A RightArm))`. This disjunctive formula defines that the agent can hold the block either with the `LeftArm` or `RightArm`: either one of these facts (but not both simultaneously) must be achieved before moving block `A` to any position, constituting a disjunctive landmark.

Pereira et al. [71, 73] show that it is possible to carry out offline plan recognition by reasoning heuristically about landmarks. The key idea is to maintain a list of ordered landmarks associated with each goal, though partial overlaps are allowed. The *goal completion* heuristic from Pereira et al. [73] matches the observations against this list. This heuristic marks a landmark as *achieved* when facts in the observation match a landmark. The heuristic then uses the ratio of the number

of landmarks achieved to the total number of landmarks associated with the goal, inducing a ranking of the goals, used as a proxy for estimating $P(g|O)$.

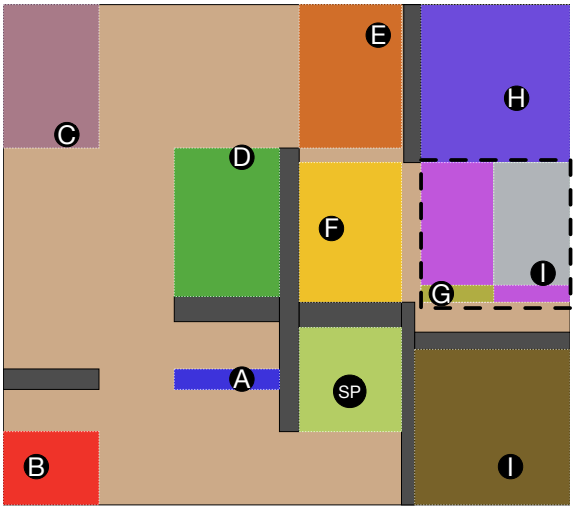


Figure 9.2: Landmarks for Cubicles environment.

In principle, we can translate the same idea into recognition in continuous domains. In such domains, landmarks can be defined as areas surrounding goals, as illustrated in Figure 9.2. This Figure presents the previously introduced 3D OMPL navigation environment (Chapter 5 Section 5.1) where black dots represent goals and the surrounding rectangles represent the continuous landmark areas. In this case, to achieve a goal, the observed motion must intersect (go through) the corresponding landmark area. Naturally, we would prefer such areas to be maximal, but must maintain the restriction that landmarks cover only obstacle-free space, and do not intersect completely with other landmarks.

The two characteristic operations between observations and landmarks in continuous environments are: testing whether a landmark has been observed, and differentiating between what constitutes an *active* landmark, and, how to identify a landmark that has recently been active but no longer fits the last observation (an *achieved* landmark).

We assume Algorithm 4 runs continually to update $P(g_k|O)$, which can then be queried as necessary. Algorithm 4 includes a continuous loop updating the conditional goal probabilities using landmarks and the notions above for online recognition as follows. We begin by pre-computing the landmarks for

Algorithm 4 Online Goal Recognition With Landmarks.

Require: $L \leftarrow$ EXTRACTLANDMARKS(I,W,G) ▷ Initial generation of landmarks
Require: $R = \langle W,O,I,G \rangle$
1: **function** CONTONLINELANDMARKS(R,L)
2: $achievedFL \leftarrow \emptyset$ ▷ Ordered set of *fact* landmarks
3: $activeFl \leftarrow \emptyset$ ▷ Last achieved landmark
4: $PruneG \leftarrow \emptyset$ ▷ Set of pruned goals
5: **while** new observation $o \in O$ is available **do**
6: $activeFL, achievedFL \leftarrow$ ACHIEVELANDMARK$(o,L,activeFL,achievedFL)$
7: **for all** $g_k \in G$ **do**
8: **if** $l_k \in achievedFL$ **then**
9: $PruneG \leftarrow PruneG + g_k$
10: **else**
11: $PruneG \leftarrow PruneG - g_k$
12: **for all** $g_k \in G \cap PruneG$ **do**
13: $P(g_k|O) \leftarrow$ RANK(g_k)

the problem and providing these (cached landmarks) to the algorithm using the EXTRACTLANDMARKS(I,W,G) function from Algorithm 6.

Once the landmarks have been computed, the main procedure CONTONLINELANDMARKS(R,L) continually updates the goal ranking in an *online* manner with every new observation, o. In order to do so, we maintain a number of data structures to keep track of the landmarks and pruned goals in between updates from new observations (Lines 2-4). First, we maintain $achievedFL$, the, initially empty, *ordered* set of fact landmarks that have already been *achieved*. In both continuous and discrete domains these landmarks represent positions in the state space that we have been in before but are there no longer. In continuous domains the achieved landmarks are areas that the agent has passed through but is no longer in. In discrete domains these are all of the facts that were satisfied before, but are possibly no longer satisfied.

We also need to maintain the currently *active* landmark against which we will compare every additional observation, $activeFl$. In continuous domains the active landmark is the area in which the current observation is found. In discrete

Algorithm 5 Achieve landmark in continuous domains.

```
1: function ACHIEVELANDMARK( o, L, activeFL, achievedFL)
2:     if activeFL ≠ ∅ ∧ o ∩ activeFL = ∅ then
3:         achievedFL ← achievedFL ∪ activeFL    ▷ activeFL is passed, i.e., a fact landmark
4:         activeFL ← ∅
5:     else if activeFL = ∅ then
6:         for all l_m ∈ L do
7:             if o ∩ l_m ≠ ∅ then
8:                 activeFL ← l_m                ▷ Found an active landmark
9:     return activeFL, achievedFL
```

domains, it is the set of facts that is currently satisfied. Additionally, we need to maintain *PruneG*, the group of goals that has been pruned out during the recognition process. We maintain this group of pruned goals due to the fact that observations may contain a certain amount of noise that would require backtracking, in which case we need to re-introduce a recently pruned goal back into the goal set, G.

For every incrementally revealed observation, o, we check if this observation has caused any landmarks to be activated or achieved via the ACHIEVELANDMARK (Algorithm 5) function in Line 6.

Because landmarks are necessary conditions to each goal, we can use the last ordered landmark associated with a goal to be the threshold, which provides evidence that an agent is pursuing a certain goal. Now that we can mark landmarks as *achieved*, we use them to infer that a certain goal can be removed from further consideration for recognition, as it has been passed. Thus, each goal g_k has a corresponding landmark $l_k \in L$ as the area that contains (i.e., is necessary for) that goal position in continuous domains and the set of facts that must be satisfied in discrete domains. We then check these landmarks l_k for each goal g_k, and, if it has been passed we prune g_k out or reinstate it if necessary in Lines 9–11. Finally, in Line 13, we iterate over all unpruned goals ranking them in decreasing order according to percentage of achieved landmarks. Consequently, the goals with the highest completion percentage will be ranked first and so on in consecutive order.

Analogously to the discrete case, we match observations to landmarks, by in-

tersecting observations (points) with each landmark. However, unlike the discrete case where, once an observation causes a landmark to be achieved, the next observation will no longer be equal to the landmark (i.e., the next step will go out of the landmark), in the continuous case this may not be so. We may see several consecutive observations, all in the same landmark area. Only once the observations *no longer* match the landmark we can mark it as *achieved*. Thus continuous landmarks, in the form of areas in spaces, define an *inclusive* disjunction: multiple observations within an area cause the landmark to be marked activated. We therefore define $activeFL$ as the currently *active* landmark while $achievedFL$ marks the landmarks that have been active but are now *achieved*.

Algorithm 1 is a general algorithm that evaluates whether landmarks are achieved in both discrete and continuous domains. Variable $activeFL$ either holds the recently active landmark, which means that the observations up till now were within that landmark area, effectively making the goal corresponding to the landmark a leading goal hypothesis, or it could be $\emptyset$. Line 2 determines if a new incoming observation o has just caused an *active* landmark to become *achieved*. If $activeFL$ is not empty **and** o is not currently in the $activeFL$ area, it means the observations have just left the $activeFL$ area and have therefore caused that landmark to be achieved. We can therefore add it to the $achievedFL$ set (Line 3) and re-initialize $activeFL$ (Line 4). However, if $activeFL$ is *empty* we check if this new observation has caused any landmarks to be *activated*. In this case we check whether the observation is part of a landmark area and insert it into $activeFL$ (Lines 5– 8).

9.2 Extracting Landmarks in Continuous Space

We can use any one of a number of landmark extraction algorithms to extract landmarks in discrete environments. Here, we adapt the algorithm of Hoffman et al. in [38] since it efficiently approximates landmark sets that are good enough for the domains we use. This algorithm builds a graph in which nodes represent landmarks and edges represent necessary prerequisites between landmarks, thus representing the landmarks and their ordering. A node in this graph represents a conjunction of facts that must be true simultaneously at some point during the

execution of a plan, and the root node is a landmark representing the goal state (Figure 9.1). Hoffman et al. [38] proves that the process of generating all landmarks and deciding their ordering is PSPACE-complete, which is exactly the same complexity as deciding plan existence [17].

Since the interpretation of landmarks we rely on for plan recognition is that of bottlenecks in the state space, we try to partition a continuous space so that such bottlenecks become identifiable areas in the continuous space. Specifically, to extract landmarks in continuous environments we partition the area using the wall corners as references, to eventually identify pathways between individual "rooms" in the space. Though we define a landmark generation algorithm for continuous path planning domains, the approach should work with any notion of numeric landmarks, e.g., recent work on landmarks for hybrid domains [89].

Algorithm 6 extracts continuous landmarks using a domain configuration W and the set of goals G, and maps each $g \in G$ to a rectangular area that represents a landmark position[1]. The landmark area for each goal starts as the outermost bounding box in the environment in Line 4. The algorithm iteratively scales it down by generating a horizontal or vertical line limit using the closest visible walls in Line 6. We define visibility as there being no obstacles between the goal and the wall in question and assume that walls correspond to axis-aligned rectangles, though at greater computational cost we could use more sophisticated notions of visibility for any polygonal obstacle through a visibility graph [22, Chapter 5]. If a single landmark area contains more than one goal, we partition this area again based on the midpoint between an arbitrary goal and the remaining ones to obtain new non-overlapping areas for each goal in Line 13, discarding the original area. This partition separates rectangles with multiple goals into a partition analogous to a Voronoi diagram with rectangular areas [5].

Figure 9.2 illustrates such landmark partition: the black lines represent walls; black dots represent goal candidates; and the different colored rectangles represent landmark areas. We can see the leftmost wall limiting the width of the landmark areas B and C of the two leftmost goals while the center wall limits their height. The dashed area including goals G and I exemplify a partition using the midpoints

[1] Note that we include additional parameters to match the algorithm for extracting landmarks for discrete domains.

Algorithm 6 Landmark Extraction for Continuous Domains.

```
1: function EXTRACTLANDMARKS(I, W, G)
2:     landmarks ← []                                  ▷ Initialize an empty dictionary
3:     for all g ∈ G do
4:         rect ← BOUNDINGBOX(g, W)
5:         for all wall ∈ W do
6:             if VISIBLEFROMCENTROID(g, wall, W) then
7:                 rect ← UPDATEBOUNDINGBOX(rect, wall)
8:                 if rect ∉ landmarks then landmarks[rect] ← ∅
9:                 landmarks[rect] ← landmarks[rect] ∪ goal
10:    for all (rect, goals) ∈ landmarks do
11:        if |goals| > 1 then
12:            for all g ∈ goals do
13:                landmarks[MIDPOINTBOX(g, goals)] ← g
14:            REMOVE(landmarks[rect])  ▷ Remove rect index from dictionary
15:        return landmarks
```

between these two goals from a square that initially contained both goals. Now that we can compute landmarks for both discrete and continuous domains, we proceed to employ them to perform online goal recognition.

9.3 Mirroring with Landmarks

In general, PRP recognizers repeatedly call a planner during recognition, and this is exacerbated in online recognition, as the goal recognizer previously described calls the planner to compute a new plan suffix, π_g with every observation, and for every goal $g \in G$. By combining Mirroring and the evidence provided by landmarks, we exploit both the flexibility of an online recognition approach that utilizes a planner within the recognition process and the efficiency of reasoning about landmarks. Specifically, we use the information conveyed by the landmarks as a pruning mechanism with which we may rule out hypotheses, reducing $|G|$ and therefore the number of calls to the planner and overall run-time.

We extensively modify the Baseline *Mirroring* algorithm to use landmark information as a pruning mechanism in Algorithm 7. For simplicity of the algorithm we assume agents do not backtrack and therefore eliminate the need to monitor

Algorithm 7 Mirroring With Landmarks.

Require: $R = \langle W, O, I, G \rangle$
Require: $L \leftarrow$ EXTRACTLANDMARKS(I, W, G)
1: **function** ONLINEMIRRORINGWITHLANDMARKS(R, L)
2: $achievedFL \leftarrow \emptyset$
3: $activeFL \leftarrow \emptyset$
4: **for all** $g \in G$ **do** $\hat{\pi}_g \leftarrow$ PLANNER(I, g)
5: **while** New $o \in O$ is available **do**
6: $p = \bigoplus_{o \in O} o$ ▷ the (partial) path created by concatenating all obs
7: $activeFL, achievedFL \leftarrow$ ACHIEVELANDMARK$(o, L, activeFL, achievedFL)$
8: **for all** $g_k \in G$ **do**
9: **if** $l_k \in achievedFL$ **then**
10: $G \leftarrow G - g_k$
11: **else**
12: $\pi_{g_k} \leftarrow$ PROJECT(PLANNER(o, g_k))
13: $\pi^O_{g_k} \leftarrow p \bigoplus \pi_g$
14: $P(g_k|O) = \frac{cost(\hat{\pi}_{g_k})}{cost(\pi^O_{g_k})}$ ▷ calculate probability for goal g_k
15: **for all** $g_k \in G$ **do**
16: $P(g_k|O) \leftarrow \eta \cdot P(g_k|O)$

the last achieved landmark and to maintain a separate set of pruned out goals (as was implemented in Algorithm 4). Like Algorithm 4, we assume a single cached computation of domain specific landmarks for all monitored goals, and the initialization of the previously introduced *achievedFL* and *activeFL* in Lines 2– 3. Additionally, as part of the *Mirroring* algorithm we now calculate the ideal plans, $\hat{\pi}_g$ (minimum cost) from the initial position to each possible goal with an additional $|G|$ planner calls (Line 4).

For every newly available observation $o \in O$, we update the plan prefix p in Line 6 and then proceed to ascertain whether this observation has caused any landmarks to be achieved via the previously introduced ACHIEVELANDMARK function. If the observation has caused a landmark to be achieved, *achievedFL* will be updated and we may - use the existing fact landmarks to prune unlikely goals in Line 10, in which case we only call the planner to compute plans for those goals whose landmarks have been satisfied in the correct order and have not been passed (Lines 12–14). Note that the plan suffix is a sequence of states, and

so we generate a projection of a plan into a sequence of states in Line 12. In the continuous case, this project is straightforward as the plan itself is a sequence of positions in the environment, whereas in the discrete case, this involves executing each action of the plan starting in state o and returning the resulting sequence of states.

We use the same ranking procedure as Vered et al. [107], in which the goals are ranked according to the ratio between the initially generated *ideal* plan and the newly generated plan hypothesis, which is comprised of a concatenation of the plan prefix and plan suffix (Lines 13– 14). Finally, the algorithm transforms these rankings into probabilities $P(g_k \mid O)$ via the normalizing factor $\eta = 1/\sum_{g_k \in G} rank(g_k)$ and computes these rankings in Lines 15–16. Mathematically, Algorithm 7 approximates $P(g \mid O)$ for all $g \in G$ using landmarks to rank probabilities, so that, when computing candidate goal probabilities in the Bayesian framework of Ramírez and Geffner [81], we compute $P(O \mid g) = cost(\hat{\pi}_g)/(cost(p) + cost(\pi_g))$. Since $P(g_k \mid O) = \eta \sum_{g_k \in G} P(O \mid g_k)$, the pruning step updates $P(g_k) = 0$ for all ruled-out goals thereby limiting the number of times we need to call the planner.

10 Experiments

> "I pass with relief from the tossing sea of Cause and Theory to the firm ground of Result and Fact."
>
> *Winston S. Churchill*

We empirically evaluated the performance of *mirroring* along with the different heuristics described above over hundreds of goal recognition problems in the complex continuous 3D OMPL environment described in Chapter 5 Section 5.1. We contrasted the performance of the suggested heuristics with the *baseline online mirroring* algorithm to show the improvements both in terms of performance as well as efficiency.We further evaluated the sensitivity of the recognition approach, by contrasting results in easier and harder goal recognition problems.

We additionally examined the effect of the Landmarks described in Chapter 9 over hundreds of recognition problems in classical and motion planning domains the literature [82] as well as a complex navigation domain. The results show superior efficiency and generally superior recognition quality over the state of the art.

10.1 Heuristic Instantiation

10.1.1 Effects of the Different Heuristic Approaches

In order to evaluate the different heuristic approaches we again utilized the 220 problems generated on the continuous 3D navigation domain of the Open Motion

Planning Library (OMPL [98]) along with the default cubicles environments and TRRT planner at a time constraint of at most 1 sec (Chapter 5, Section 5.1, Figure 5.1(a)). The *cost measure* of the plan referring to the length of the path. We ran separate runs for each approach and then contrasted the results. The results are displayed in Figures 10.2 and 10.1. In all of the graphs the X axis denotes the different approaches;

1. Naive refers to the suggested online recognition approach of re-running the PRP based *offline* recognition algorithm as-is, and calling the planner to re-calculate each of the ideal plans (Algorithm 1, Chapter 4, Section 4.2). By contrasting the improvements in run-time and number of calls to the planner made by each of the separate approaches against this approach we can measure just how necessary and efficient the adjustments for only recognition are.

2. *Minimum* refers to the method of no recomputation at all - *Minimum Plan Generation* (Chapter 8, Section 8.1). In this instance the planner is only utilized once in the beginning of the process for all of the goals. All incrementally received observations will be compared against these initially calculated plans for a total of only $2|G|$ calls to the planner.

3. *Baseline* refers to the online recognition method of refraining from recomputing the ideal path to compare against with every incoming observation; with the improved online recognition baseline of $(O+1)|G|$ calls (Alg.2, Chapter 4, Section 4.3).

4. *Recompute* measures the effect of the *Recompute Heuristic* which aims to reduce overall number of calls to the planner by recomputing plans only when necessary; i.e. refraining from calling the planner when the new observation does not alter the top ranked goal (Alg.3, Chapter 8, Section 8.3.1).

5. *Prune* measures the effect of the *Pruning Heuristic* which aims to reduce the overall number of goals by eliminating unlikely goal candidates (Alg.3, Chapter 8, Section 8.3.2).

6. And finally, *Both*, measures the effects of utilizing both the *Pruning* and *Recompute* Heuristics simultaneously.

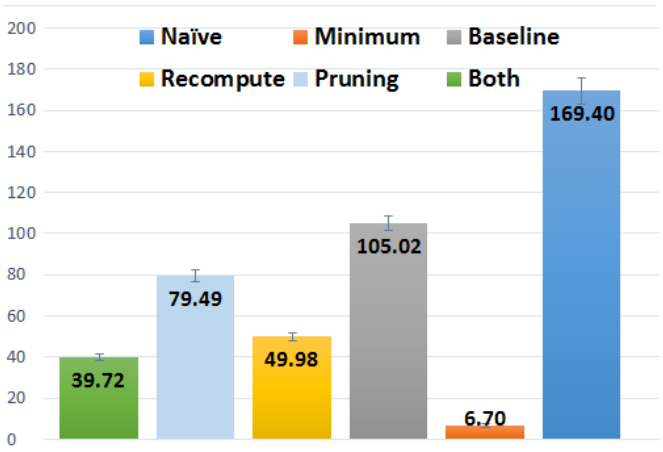
Efficiency In order to evaluate recognition efficiency we used the evaluation criteria first presented in chapter 5, Section 5.2.

- The time in which the recognizer converged to the correct hypothesis (including 0 if it failed). Measured by counting the number of observations from the end, hence higher values indicate earlier convergence and are therefore better.

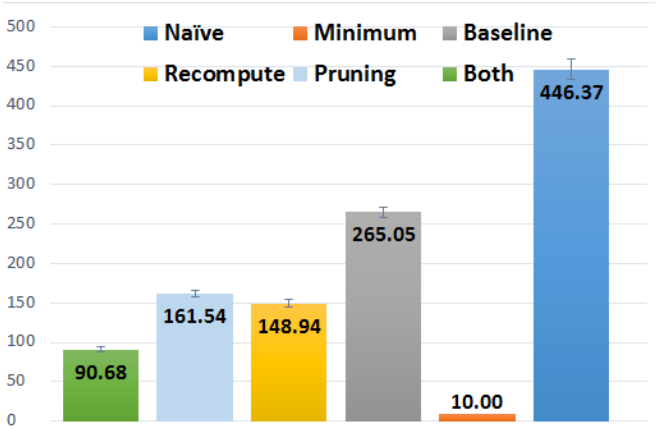
- The number of times the correct hypothesis was ranked at the top (i.e., rank 1). Again higher values indicate more correct hypotheses during the recognition process.

Figure 10.1(a) displays the average of the results of each approach as the mean of total planner run-time where the Y axis denotes the time measured in seconds. The *Naive* approach takes an average of 169.4 sec. for each problem. When only calling the planner once in the recognition process the *Minimum* approach takes an average of only 6.7 sec. When not recomputing the ideal plans, the *Baseline* reduces the *Naive* time to an average of 105.02 sec. The *Prune* heuristic reduces the average time further to only 79.49 sec. And the *Recompute* heuristic further reduces the average time to only 49.98 sec. When utilizing both heuristics we achieved a reduction to 39.72 sec. an improvement of a substantial 76.55% from the *Naive* approach and 62.18% from the *Baseline* approach.

Figure 10.1(b) displays the average of the results in terms of number of calls made by the recognizer to the planner. Here the y axis denotes the overall number of calls. The *Naive* approach had an average of 446.37 calls to the planner while with no recomputation at all the *Minimum* approach had an average of an extremely efficient 10 calls, i.e. the number of goals. *Baseline* substantially reduced the number of calls by almost half in comparison with the *Naive* approach, generating an average of only 265.05 calls. The *Recompute* and *Prune* heuristics had similar success with a reduction to 148.94 and 161.54 calls each. Finally,



(a) Mean Run-Time (Sec.)



(b) Mean Number Of Calls To Planner

Figure 10.1: Efficiency comparison. Lower values are better.

while utilizing both heuristics the number of calls was reduced to an average of only 90.68 calls, a reduction of 65.79% from the *Baseline* approach and of 79.69% from the *Naive* approach.

In conclusion we see that employing the heuristics makes a big impact on run-time and successfully reduces overall number of calls to the planner. While the *Recompute* heuristic outperformed the *Prune* heuristic, both in run-time and overall number of calls, utilizing both heuristics can reduce both run-time and number of calls made to the planner by over 75% from the naive approach. The most efficient method proved to be the *Minimum Plan Generation* approach, only

calculating $|G|$ plans. We will later show that this improvement in efficiency costs considerably in performance. Given these results, in the next figures we omit the original *Naive* method and focus only on the heuristics with comparison to the *Baseline* approach.

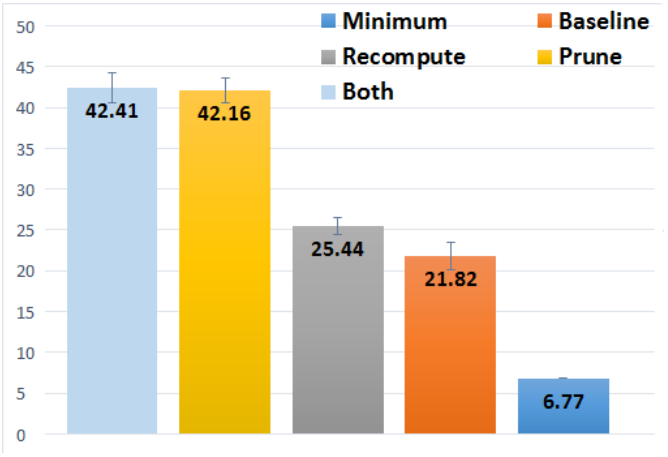
Performance In order to evaluate recognition performance we used the evaluation criteria first presented in chapter 5.

- *Convergence*. The time in which the recognizer converged to the correct hypothesis (including 0 if it failed). This is measured by counting the number of observations from the end, hence higher values are better.

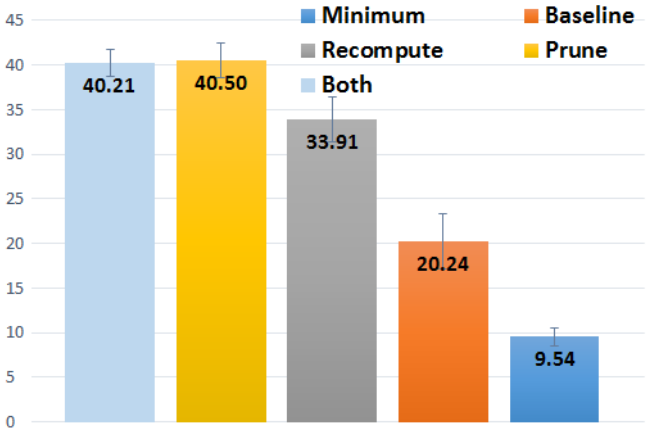
- *Ranked First*. The number of times the correct hypothesis was ranked at the top (i.e., rank 1). Again higher values indicate more correct hypotheses during the recognition process.

Figure 10.2(a) measures the average convergence to the correct result percent, higher is better. With no reuse of the planner at all, for the *Minimum* approach we only get a 6.77% convergence. As this approach does not make use of the incrementally revealed observations within the recognition process, any deviation from the initially calculated path, i_g, will have considerable impact on recognition results. These results also reflect the fact the the planner used to generate these initial plans is in itself, sub-optimal.

By converting to the *Baseline* algorithm, we were able to more than double the convergence percent to 21.82%. Each incremental observation was now taken into account during the reuse of the planner and therefore had greater weight on the ranking of the goals. Applying both the *Prune* and *Recompute* heuristics further improve the overall convergence. By eliminating goals the ranking process now proved to be easier, as there were less goals to compare to. Furthermore, the early elimination of goals in the pruning process was able to also eliminate the further noise these goals might introduce to the ranking process, when their paths deviated from the optimal. The *Recompute* heuristic increases it to 25.44% and the *Prune* to 42.16%, an improvement of 20.4% from the *Baseline* approach. Furthermore,



(a) Mean Convergence Percent



(b) Mean Ranked First Percent

Figure 10.2: Performance comparison. Higher values are better.

we see that when utilizing both heuristics the high convergence level obtained by the *Prune* heuristic is maintained.

Figure 10.2(b) measures the percent of times the correct goal was ranked first out of overall observations. Here too a higher value is better and will reflect on overall reliability of the ranking procedure. The results mostly agree with the convergence results. With no planner reuse at all, *Minimum*, performs poorly with a low 9.54%. Utilizing the *Baseline* approach, which refrains recomputing the ideal plan, more than doubles the success here as well, to 20.24%. The *Recompute* heuristic achieves 33.91% and the *Prune* heuristic increases the results to 40.5%,

an improvement of 20.3% from the *Baseline* approach. Again, when applying both heuristics the success level of the *Prune* method is obtained.

As seen employing the heuristics has made a big impact on overall performance successfully increasing convergence and overall correct rankings. The *Prune* heuristic clearly outperformed the *Recompute* heuristic in both measures however a combination of both heuristics maintains the high success rate leading to an improvement of over 20% in both measures.

10.1.2 Sensitivity to Recognition Difficulty

We now attempt to show that the proposed heuristics improve performance and efficiency even when facing more difficult recognition problems. By varying levels of difficulty we refer to problems that have more goals, and where the goals are clustered closer together exacerbating the recognition task. We used the *harder* 3D navigation goal recognition scenario described in Chapter 5 Section 5.1 and presented in Figure 5.1(b). In this problem another 9 goal points were added to the recognition problems in the navigation domain for a total of 19 potential goals in each recognition problem and 380 recognition problems.

	19 goals			
	Efficiency		**Performance**	
	Run-Time	**PlannerCalls**	**Conv.**	**Rank.**
Baseline	194.65	516.57	16.37	19.54
Recompute	126.75	397.85	18.7	22.76
Prune	160.29	386.53	23.18	24.03
Both	97.63	287.36	20.98	25.82

Table 10.1: Comparison of all approaches across clustered goal scenario.

Table 10.1, columns 1–2, examines the *efficiency* of the different online recognition approaches over the *harder* clustered goals problems. We omitted the *Naive* heuristic in these instances as the behavior of this heuristic is very straightforward. The results are consistent with the results from the original scenario. The *Baseline* approach is the least efficient, having a higher run-time and larger number of calls to the planner, than the rest. The most efficient approach is still the approach of utilizing both the *Prune* heuristic and the *Recompute* heuristic together. In run-time the *Recompute* heuristic is still more efficient than the *Prune* however for the

measure of number of calls made to the planner we see that, for more clustered goals scenarios, the *Prune* heuristic slightly outperforms the *Recompute* heuristic.

Table 10.1, columns 3–4, examines the *performance* of the different online recognition approaches over the *harder* clustered goals problems. For the harder problems the best performance achieved, in terms of convergence, was by the *Prune* heuristic with a convergence of 23.18% from the end. In terms of the amount of times the correct goal was ranked first the *Both* approach, combining both *Prune* and *Recompute* heuristics, only slightly outperformed the *Prune* approach. The worst performance was achieved by the *Baseline* approach, in terms of both criteria measured; convergence and ranked first, in congruence with the performance results of the scattered goal scenario.

	Deterioration			
	Efficiency		**Performance**	
	Run-Time	**PlannerCalls**	**Conv.**	**Rank.**
Baseline	85.35%	94.90%	24.98%	3.46%
Recompute	153.58%	167.11%	26.49%	32.88%
Prune	114.01%	155.20%	45.02%	40.67%
Both	167.58%	216.90%	50.53%	35.79%

Table 10.2: Deterioration of performance and efficiency between scattered and clustered goal scenarios.

Table 10.2 measures the deterioration in efficiency and performance with comparison to the scattered goal scenario. The deterioration is measured in terms of deterioration percent, hence a 100% deterioration in run-time means the planner took twice as long on average, on the harder problems. Therefore lower values are better. In terms of efficiency, we can clearly see that the least deterioration, both in run-time and number of calls to the planner, occurred for the *Baseline* approach proving this approach to be very reliable with a deterioration of 85.35% and 94.90% respectively. The biggest run-time deterioration occurred for the combination of both heuristics with a deterioration of 167.58%. This was mostly caused by the substantial deterioration of the *Recompute* approach which deteriorated by 153.58%. The *Prune* heuristic deteriorated considerably less in terms of run-time with only 114% deterioration.

In terms of number of calls made to the planner, again, the worst deterioration occurred for the approach that combined the two heuristics, *Both*, with a deteriora-

tion of 216.9% while the deterioration for each of the heuristics by themselves was considerably less; 155.2% for the *Prune* heuristic and 153.6% for the *Recompute* heuristic.

In terms of performance deterioration we again see that the most resilient approach in terms of performance, as well as efficiency, proved to be the *Baseline* both in terms of Convergence and Ranked first with a deterioration of 25.98% in convergence and only 3.46% in ranked first. The biggest deterioration in convergence occurred for the *Both* approach, as was in the efficiency results. However, in terms of ranked first the biggest deterioration occurred for the *Prune* heuristic. This was, in part, due to the fact that clustered goals make the pruning process considerably less efficient as the goals are too close to be pruned.

10.2 Combining Landmarks and Mirroring

As a final set of experiments we empirically evaluated the suggested *online* goal recognition approach which combines mirroring with information gained by using disjunctive landmarks, on both discrete and continuous environments over hundreds of goal recognition problems while measuring both efficiency and performance.

As the continuous environment we again used the Open Motion Planning Library (OMPL [98]) domain of 3D navigation, along with default cubicles environment and TRRT planner, where the target is to recognize navigational goals as soon as possible while the observations, i.e., observed agents' positions, are incrementally revealed. We used the *easy* scenario with 11 points spread through the cubicles environments and two observed paths from each point to all others, for *a total of* 110×2 *goal recognition problems* (Chapter 5, Section 5.1, Figure 5.1(a)).

For the discrete environments, we used the openly available datasets [72] based on the ones developed by Ramírez and Geffner [80, 81].These datasets comprise 15 domain models with thousands of non-trivial and large goal recognition problems with optimal and sub-optimal plans. We evaluated the approaches using optimal and sub-optimal plans. Each goal recognition problem contains a domain description, initial state, set of candidate goals, a hidden goal, and an observation

sequence representing a plan that achieves the hidden goal. As the discrete planner, we used the JavaFF[1], an implementation of Fast-Forward [37] in Java. We ran the experiments for discrete environments with an 8GB memory limit on the JavaVM and a 2-minute time limit.

				Continuous Domains																	
				MIRRORING						MIRRORING WITH LANDMARKS						ONLINE RECOGNITION WITH LANDMARKS					
Domain (# problems)	$\|G\|$	$\|O\|$	$\|L\|$	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*
Cubicles (220)	11.0	26.5	11.0	104.70	265.0	**100%**	100%	20.2%	21.8%	85.90	184.8	78.2%	61.1%	**24.3%**	**26.2%**	**0.020**	**0**	78.3%	**60.9%**	21.7%	15%

				Discrete Domains																	
				MIRRORING						MIRRORING WITH LANDMARKS						ONLINE RECOGNITION WITH LANDMARKS					
Domain (# problems)	$\|G\|$	$\|O\|$	$\|L\|$	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*
Campus (15)	2.0	5.4	8.6	0.441	12.8	60.0%	21.3%	57.3%	41.3%	0.212	7.7	**96.4%**	**1.7%**	**96.4%**	**96.4%**	**0.065**	**0**	92.8%	3.5%	92.8%	92.8%
IPC-Grid (61)	8.3	21.8	10.2	10.36	209.1	**87.2%**	19.4%	36.6%	35.6%	3.29	71.2	55.6%	**10.5%**	**45.8%**	**41.5%**	**0.335**	**0**	59.4%	21.8%	32.6%	31.1%
Ferry (28)	7.5	24.2	28.5	55.24	179.5	83.1%	10.2%	59.2%	57.2%	7.98	35.4	**83.3%**	**3.1%**	**82.4%**	**82.1%**	**0.101**	**0**	82.4%	5.4%	72.5%	71.9%
Intrusion (45)	16.6	13.1	16.0	2.02	235.5	**100%**	7.2%	55.3%	55.3%	0.257	34.7	75.5%	**3.6%**	**67.1%**	**67.1%**	**0.127**	**0**	87.6%	3.9%	57.1%	55.1%
Kitchen (15)	3.0	7.4	5.0	0.141	25.4	70.1%	18.4%	44.6%	36.1%	0.07	20.0	77.6%	**17.9%**	**62.6%**	**58.3%**	**0.04**	**0**	**100%**	50%	23.9%	23.9%
Logistics (61)	10.4	24.4	16.1	53.82	199.3	**95.4%**	14.7%	26.9%	25.8%	14.39	49.6	61.7%	**6.7%**	**49.1%**	**48.4%**	**0.594**	**0**	56.1%	9.5%	40.5%	40.5%
Rovers (28)	6.0	24.9	19.8	*Timeout*	-	-	-	-	-	58.87	31.1	**76.8%**	**4.8%**	**76.2%**	**75.1%**	**0.867**	**0**	72.1%	8.5%	62.1%	62.1%
Satellite (28)	6.4	16.9	10.1	93.89	177.2	**100%**	33.8%	36.1%	36.1%	5.18	30.6	81.8%	9.4%	**72.8%**	**71.9%**	**1.09**	**0**	78.2%	**9.3%**	64.4%	64.1%

Table 10.3: Experimental results for both continuous and discrete domains.

We evaluated the combined approach (MIRRORING WITH LANDMARKS) both in terms of improvement in efficiency and in terms of overall performance showing that the improvement in efficiency did not come at the expense of performance but rather improved it. We then contrasted the performance with the Baseline (Online MIRRORING) algorithm and the newly presented online recognition approach utilizing only the landmarks for ranking and pruning out goals (ONLINE RECOGNITION WITH LANDMARKS).

Efficiency Measures We used two separate measures to evaluate the overall *efficiency* of the proposed approach: the number of planner calls (PC) within the recognition process; and the overall time (Time, in sec.) spent planning. Both these parameters measure the overhead of using the planner in the *mirroring* approach and while they are closely linked, they are not wholly dependent. While a

[1] `https://github.com/Optimised/JavaFF`

reduction in overall number of calls to the planner necessarily results in a reduction in planner run-time, the total amount of time allowed for each planner run may vary according to the difficulty of the planning problem and therefore create considerable differences. Naturally, lower values are better.

Performance Measures We used several complementary measures for a thorough evaluation of recognition performance. *True positive rate* (TPR) measures the number of times an approach recognized the correct goal as a possible hypothesis, i.e. didn't prune it out due to landmarks although didn't necessarily rank it as the chosen hypothesis. We measure mean average percent TPR over all recognition steps. Higher TPR values are better, indicating a measure of the reliability of the system. This value corresponds to 1-FNR (False negative rate). *False positive rate* (FPR) refers to how many goals were not pruned out due to landmarks. We measure FPR in percent out of overall goal number. Lower FPR values are better indicating more pruning and therefore a more efficient algorithm. *Ranked first* (RF) is the number of times the correct hypothesis was not only recognized as a possible hypothesis but also ranked first. *Convergence* (CV) to the correct answer indicates the time step in which the recognizer converged to the correct hypothesis from the end of the observation sequence (or 0 if it failed). Higher values indicate earlier convergence and are therefore better.

	Discrete Domains							
	ONLINE RECOGNITION WITH LANDMARKS							
Domain (# problems)	$\|G\|$	$\|O\|$	$\|L\|$	*Time*	*TPR*	*FPR*	*RF*	*CV*
Blocks-World (92)	20.2	20.3	21.0	**0.251**	**39.4%**	**3.9%**	**38.1%**	**37.2%**
Depots (28)	8.8	27.4	33.2	**0.812**	**49.5%**	**9.5%**	**32.1%**	**30.6%**
Driver-Log (28)	7.1	21.7	10.7	**0.574**	**51.8%**	**8.8%**	**43.7%**	**40.1%**
DWR (28)	7.2	51.8	45.0	**0.708**	**45.1%**	**7.9%**	**43.1%**	**33.5%**
Miconic (28)	6.0	35.5	25.5	**0.711**	**82.5%**	**9.7%**	**62.6%**	**61.2%**
Sokoban (28)	7.1	27.7	9.8	**0.772**	**58.1%**	**14.1%**	**36.0%**	**29.5%**
Zeno-Travel (28)	6.8	21.1	8.5	**1.23**	**70.8%**	**6.4%**	**61.3%**	**59.7%**

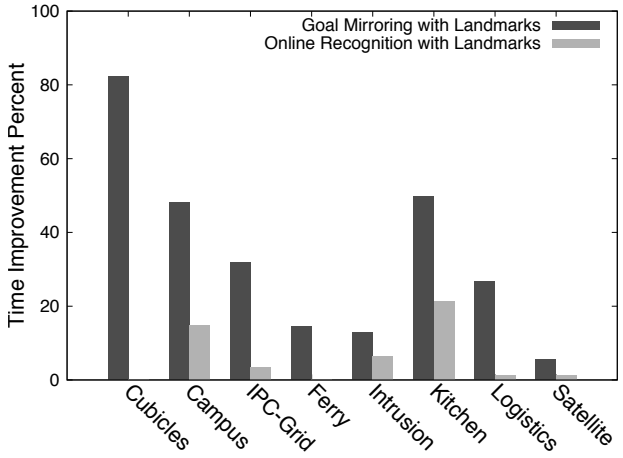
Table 10.4: Experimental results for discrete domains (large and non-trivial planning problems).

Results Table 10.3 shows the experimental results for both continuous and discrete domains across all criteria. For the continuous domain, the combined MIRRORING WITH LANDMARKS approach achieved the best performance with an improvement both in convergence and the amount of times the recognizer ranked the correct goal hypothesis first. It proved just as reliable as ONLINE RECOGNITION WITH LANDMARKS in terms of *TFR* and *FPR*, however not as reliable as MIRRORING, which does not prune out goals at all, incurring no risk of overlooking the correct goal.

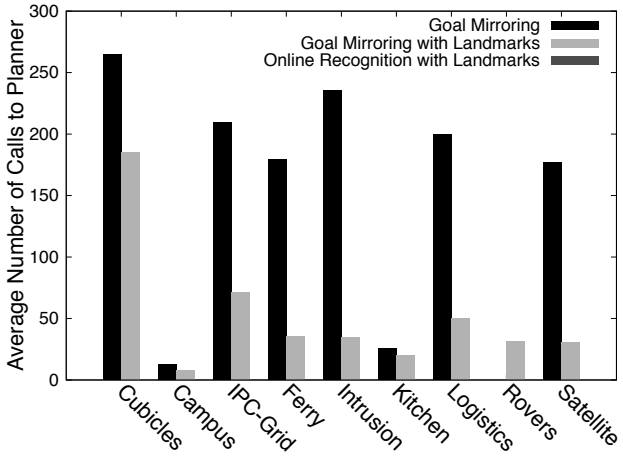
We see that the combined approach of MIRRORING WITH LANDMARKS achieves the overall best results in terms of *convergence* and *ranked first* in the discrete domains. However, unlike in the continuous domain, using the ONLINE RECOGNITION WITH LANDMARKS technique also provided good results, sometimes even better than the MIRRORING approach (Campus, Ferry domain, Logistics, Satellite). In the *Rovers* domain problem we see an instance where MIRRORING was unable to find a solution within the given time limit, however when utilizing landmarks, the MIRRORING WITH LANDMARKS approach was able to finish and provide better results than ONLINE RECOGNITION WITH LANDMARKS. This is due to the complex nature of the dataset and highlights the advantages of using landmarks as a pruning mechanism.

However, there were several instances where the dataset was so complex that both the MIRRORING and MIRRORING WITH LANDMARKS approaches failed. Due to the repeated calls to the planner these approaches timed-out without results. These problems were considerably more complex with a larger number of objects and instantiated actions. The results are summarized in Table 10.4, where we see the strength of the ONLINE RECOGNITION WITH LANDMARKS approach, which does not employ a planner and therefore evades the considerable overhead calculations.

The improvement of run time over the baseline MIRRORING approach is presented in Figure 10.3(a). For the continuous domain, MIRRORING WITH LANDMARKS, incorporating landmarks, reduces the run time to 80% while ONLINE RECOGNITION WITH LANDMARKS was by far the most efficient with a reduction to only 0.019% of the original MIRRORING runtime. For the discrete domain as well we see that ONLINE RECOGNITION WITH LANDMARKS more than



(a) Runtime Improvement



(b) Average Planner Calls

Figure 10.3: Efficiency comparison.

doubles the reduction in run-time vs. the ONLINE RECOGNITION WITH LANDMARKS, which in itself reduces the run-time considerably to between 17%–46%. Figure 10.3(b) shows a comparison regarding the amount of times the planner was called within the recognition process for both continuous and discrete domains.

10.3 Summary

We presented a heuristic approach to Mirroring to improve the overall efficiency of the algorithm. We identified key decision points which effect both overall run-time

and the number of calls made to the planner and introduced a generic online goal recognition algorithm along with two heuristics to improve planner performance and efficiency in navigation goal recognition. We evaluated the approach in a challenging navigational goals domain over hundreds of experiments and varying levels of problem complexity. The results demonstrate the power of our proposed heuristics and show that, while powerful by themselves, a combination of them leads to a reduction of a substantial 63% of the calls the recognizer makes to the planner and planner run-time in comparison with previous work. This, while showing an increase of over 20% in recognition measures.

We then developed a novel heuristic pruning technique by reasoning over a generalized notion of landmarks.We have shown how to dynamically generate continuous and discrete landmarks and empirically evaluated the efficiency and performance of our approach over hundreds of experiments in both continuous and discrete domains; comparing our results to the Baseline *Mirroring* approach and a newly defined continuous landmark approach. We have shown that not only is our approach more efficient than the existing online recognizer but also outperforms both other approaches.

11 Future Directions and Final Remarks

”What would you do if you weren’t afraid?”

Sheryl Sandberg

We summarize the key contribution of this thesis in Section 11.1. We discuss future directions for this research in Section 11.2.

11.1 Summary of Key Contribution

When I first set out on my academic adventure I knew I wanted to model agents that will have a positive impact on everyday human life. To achieve this goal I needed the agents to, in a sense, *understand* the humans they are operating along with. This led me to many questions about human behavior modeling. It occurred to me that in order to achieve my goal of creating efficient, intelligent agents, that would be able to incorporate seamlessly into human lives I will also have to understand people and consequently human social behavior models.

This dissertation compiles the first piece of the puzzle I attempted to uncover, the innate human ability of intention recognition. Intention recognition being a very important part of any social interaction it is naturally something that humans excel at. This led me to learn about the *mirror neuron system*. Recent neuro-imaging data indicates that the *mirror neuron system* within the human brain is in charge of matching the observations and the execution of actions. This system

is hypothesized to give humans the ability to infer the intentions leading to an observed action using their own internal mechanism. The human mirror neuron system may be viewed as a part of the brains' very own plan/goal recognition module and can be used to recognize the actions and goals of other agents from a series of observations of the other agents' actions.

The result of this work are efficient, cognitive inspired Mirroring algorithms that work online and offline in continuous and discrete domains. Mirroring utilizes a planner to generate recognition hypotheses that are continually matched against incremental observations and works in continuous as well as discrete domains. In Chapter 3 we introduced *Mirroring* as a general formulation of plan recognition in continuous domains which can also generalize to discrete domains. By using motion planners in the recognition process, we avoid early commitment to a granularity (discretization) level, and thus can choose the best discretization for the recognition problem at hand. The use of *Mirroring* also allows the use of OTS, unmodified planners and enables us to re-use existing resources to the best advantage. Mirroring gives rise to a recognition procedure that uses two calls to a planner for each goal while accounting for missing observations.

We were then able to contrast *Mirroring* with human recognition in Chapter 6. To evaluate this aspect we investigated Mirroring within the context of shape recognition, where by a planner was re-used by the recognition process, allowing drawn-shape recognition by drawn-shape planning. We instantiated the shape recognition approach in the recognition of regular polygons, and evaluated the performance of different ranking and non-ranking variants of the recognizer against human subjects' recognition of scanned hand-drawn regular polygons. In general, *Mirroring* performed on par, or just below, human levels of recognition exposing a couple of interesting insights into the human recognition process. We learned that humans make negative recognition mistakes, both in disqualifying hypotheses too early and in holding on to them even once it is proven they are incorrect. However, it might be that the tendencies leading to these mistakes might also account for the better performance of humans. With each new insight the importance of each particular in the human brain became more tangible as well as how much we don't yet know.

After delving into exactly *what* Mirroring is we attempted to tackle the prob-

lem of *how* best to go about performing it (Chapter 8). We had to deal with several difficulties arising from the continual re-use of a planner within the recognition process. We therefore identified two key decision points which directly effect overall run-time and the number of calls made to the planner. We then introduced a generic online recognition algorithm along with two heuristics to improve planner performance and efficiency in navigation goal recognition. We proceeded to further improve Mirroring efficiency by utilizing a combination of Mirroring along with previous work; information gained from disjunctive landmarks calculated once in advance.

We evaluated our approach in several challenging domains (Chapter 5 Section 5.1).

- A highly complex, continuous, 3D, navigational goals domain which included hundreds of experiments and varying levels of problem complexity.
- The *entire* set of discrete benchmark plan-recognition problems used in [81] including 450 problems in six classical planning domains.
- An especially built shape planner to recognize geometric shapes as they were being drawn.
- A cooperative robotic team task implemented using ROS [79] in which we utilized our recognition algorithm to recognize the goals of navigation in 3D worlds using the ROS *MoveBase* default planner.

We measured the success of our approach both in terms of efficiency and in terms of performance. The results demonstrate the power of our approach, showing that it is capable of efficiently recognizing goals using several different, OTS, standard motion planners. Additionally, while evaluating our proposed heuristics, we have shown that while powerful by themselves, a combination of them leads to a reduction of a substantial 63% of the calls the recognizer makes to the planner and planner run-time in comparison with the proposed baseline approach and 80% compared to the naive online approach. In terms of convergence and overall first ranking, we saw an increase of over 20% in comparison with the baseline approach.

11.2 Future Directions

In future we hope to study the differences between human and machine recognition, especially in the ability to disqualify hypotheses earlier on. Understanding the biases that humans exhibit may prove useful for improving the recognition process as well as for facilitating any human agent cooperation. We are also interested in finding methods for automatically calibrating the thresholds in the ranking procedure to better handle inaccuracy and noise in perception. Another interesting area we hope to investigate is to incorporate learning into our mechanism. Learning in itself is highly connected to imitation and therefore a combination of Mirroring along with the benefits of learning should be very interesting to explore.

As our technique continually calls a planner within the recognition process, it might have limitations in recognizing very complex problems, specifically, those for which current planning algorithms are not efficient. A limitation of some of the efficiency improvements we have presented is the use of relatively simple landmarks for spatial domains, as well as the assumption that landmarks do not change over the course of the recognition, which would not be realistic for dynamically changing environments. Thus, two additional important refinements should be made in the future; First, to refine the notion of spatial landmarks for more informative heuristics, such as the ones developed by Scala et al. [89]. Second, to use techniques to compute landmarks incrementally so as to allow their online recomputation in dynamic domains.

References

[1] D. Albrecht, I. Zukerman and A. Nicholson. Bayesian models for Keyhole plan recognition in adventure game. *User Modeling and User-Adapted Interaction*, 8(1–2):5–47, 1997.

[2] A. Amanatiadis, V. Kaburlasos, A. Gasteratos, and S. Papadakis. Evaluation of shape descriptors for shape-based image retrieval. *IET Image Processing*, 5(5):493–499, 2011.

[3] O. Amir and Y. Gal. Plan recognition and visualization in exploratory learning environments. In *ACM Transactions on Interactive Intelligent Systems*, 3(3):16:1–16:23, 2013.

[4] D. Anderson, C. Bailey, and M. Skubic. Hidden Markov model symbol recognition for sketch-based interfaces. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-04)*, pages 15–21, 2004.

[5] F. Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[6] D. Avrahami-Zilberbrand and G. A. Kaminka. Fast and complete symbolic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[7] D. Avrahami-Zilberbrand and G. A. Kaminka. Incorporating observer biases in keyhole plan recognition (efficiently!). In *Proceedings of the National Conference on Artificial Intelligence (AAAI-07)*, pages 944–949, 2007.

[8] C. Baker, R. Saxe and J. B. Tenenbaum. Bayesian models of human action understanding. *Advances in Neural Information Processing Systems*, 99–106, 2005.

[9] C. L. Baker, J. B. Tenenbaum, and R. R. Saxe. Goal inference as inverse planning. In *Proceedings of the 29th Annual Meeting of the Cognitive Science Society*, 2007.

[10] N. Blaylock and J. Allen. Statistical goal parameter recognition. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 297–304, 2004.

[11] N. Blaylock and J. Allen. Fast hierarchical goal schema recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-06)*, pages 796–801, 2006.

[12] N. Blaylock and J. F. Allen. Statistical goal parameter recognition. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04)*, volume 4, pages 297–304, 2004.

[13] E. Bonchek-Dokow and G. A. Kaminka. Towards computational models of intention detection and intention prediction. *Cognitive Systems Research*, 28:44–79, 2014.

[14] B. Bonnet and H. Geffner. HSP: Heuristic search planner. *Citeseer*, 1998.

[15] J. S. Brown, R. R. Burton and K. M. Larkin. Representing and using procedural bugs for educational purposes. In*Proceedings of the 1977 annual conference (ACM)*, 247–255, 1977.

[16] H. H. Bui. A general model for online probabilistic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, volume 3, pages 1309–1315, 2003.

[17] T. Bylander. The Computational Complexity of Propositional STRIPS Planning. *Journal of Artificial Intelligence Research (JAIR-94)*, 69:165–204, 1994.

[18] S. Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1):31–48, 2001.

[19] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtós, and M. Carreras. Rosplan: Planning in the robot operating system. In *Proceedings of The International Conference on Automated Planning and Scheduling (ICAPS-15)*, pages 333–341, 2015.

[20] J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3):367–385, 2003.

[21] E. Charniak and R. P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence (AIJ)*, 64(1):53–79, Nov. 1993.

[22] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. Principles of Robot Motion: Theory, Algorithms, and Implementation. *MIT press*, 2005.

[23] A. J. Coles, A. I. Coles, M. Fox, and D. Long. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research*, 44:1–96, 2012.

[24] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1994.

[25] M. T. Cox and B. Kerkez. Case-based plan recognition with novel input. *Control and intelligent systems*, 34(2), 2006.

[26] M. J. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, 2004.

[27] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[28] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis: probabilistic models of proteins and nucleic acids. *Cambridge University Press*, 1998.

[29] S. R. Eddy. Hidden markov models. *Current Opinion in Structural Biology*, 6(3):361–365, 1996.

[30] E. Fernández-González, E. Karpas, and B. C. Williams. Mixed discrete-continuous heuristic generative planning based on flow tubes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-15)*, pages 1565–1572, 2015.

[31] M. J. Fonseca and J. A. Jorge. Using fuzzy logic to recognize geometric shapes interactively. In *Proceeding of The Ninth IEEE International Conference on Fuzzy Systems*, volume 1, pages 291–296. IEEE, 2000.

[32] M. Fox and D. Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR-03)*, 2003.

[33] C. Geib and R. Goldman. Recognizing plans with loops represented in a lexicalized grammar. In *The twenty fifth AAAI Conference on Artificial Intelligence (AAAI-11)*, pages 958–963, 2011.

[34] C. W. Geib. Lexicalized reasoning. In *Proceedings of the Third Annual Conference on Advances in Cognitive Systems*, 2015.

[35] C. W. Geib and R. P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

[36] J. Hoffmann. The metric-ff planning system: Translating“ignoring delete lists”to numeric state variables. *Journal of Artificial Intelligence Research (JAIR-03)*, 20:291–341, 2003.

[37] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR-01)*, 14(1):253–302, 2001.

[38] J. Hoffmann, J. Porteous, and L. Sebastia. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research (JAIR-04)*, 22(1):215–278, 2004.

[39] J. Hong. Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research (JAIR-01)*, 15:1–30, 2001.

[40] D. H. Hu, S. J. Pan, V. W. Zheng, N. N. Liu, and Q. Yang. Real world activity recognition with multiple goals. In *Proceedings of Ubiquitous Computing (UbiComp-08*, 2008.

[41] L. Illanes and S. A. McIlraith. Numeric planning via search space abstraction. In *Ninth Annual Symposium on Combinatorial Search*, 2016.

[42] L. Jaillet, J. Cortés, and T. Siméon. Transition-based rrt for path planning in continuous cost spaces. In *IEEE International Conference on Intelligent Robots and Systems (IROS-08)*, pages 2145–2150. 2008.

[43] J. Jorge and F. Samavati. *Sketch-based interfaces and modeling*. Springer, 2010.

[44] J. A. Jorge and M. J. Fonseca. A simple approach to recognise geometric shapes interactively. In *Graphics Recognition Recent Advances*, pages 266–274. Springer, 2000.

[45] B. Kaluža, G. A. Kaminka, and M. Tambe. Detection of suspicious behavior from a sparse set of multiagent interactions. In *Proceedings of the Eleventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-12)*, 2012.

[46] G. A. Kaminka. Curing robot autism: A challenge. In *Proceedings of the 2013 international conference on Autonomous Agents and Multi-Agent Systems (AAMAS-13)*, pages 801–804. 2013.

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

[48] H. A. Kautz and J. F. Allen. Generalized plan recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, pages 32–37. 1986.

[49] S. Keren, A. Gal and E. Karpas. Goal recognition design for non-optimal agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-15)* ,3298–3304, 2015.

[50] W. Y. Kwon and I. H. Suh. A temporal bayesian network with application to design of a proactive robotic assistant. In IEEE International Conference on *Robotics and Automation (ICRA-12)*, pages 3685–3690. 2012.

[51] J. E. Laird. It knows what you're going to do: adding anticipation to a Quakebot. In *Proceedings of the Fifth International Conference on Autonomous Agents*, 385–392, 2001.

[52] N. Lesh and O. Etzioni. A sound and fast goal recognizer. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[53] H. X. Li and B. C. Williams. Generative planning for hybrid systems based on flow tubes. In *Proceedings of The International Conference on Automated Planning and Scheduling (ICAPS-08)*, pages 206–213, 2008.

[54] B. Liang, C. Chen, Y. H. Guan and X. Y. Huang. Estimating the missing traffic speeds via continuous conditional random fields. *Web Technologies and Applications*, 35–43, 2015.

[55] L. Liao, D. Fox and H. Kautz. Hierarchical conditional random fields for GPS-based activity recognition. *The 11th International Symposium on Robotics Research (ISRR)*, 2007.

[56] Y. E. Martin, M. D. Moreno, and D. E. Smith. A fast goal recognition technique based on interaction estimates. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-15)*, pages 761–768, 2015.

[57] P. Masters and S. Sardina. Cost-based goal recognition for path-planning. In *Proceedings of the Sixteenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-17)*, pages 750–758. 2017.

[58] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL—The Planning Domain Definition Language. In *The AIPS-98 Planning Competition Committee*, 1998.

[59] J. P. Mendoza, J. Biswas, P. Cooksey, R. Wang, S. D. Klee, D. Zhu, and M. M. Veloso. Selectively reactive coordination for a team of robot soccer champions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-16)*, pages 3354–3360, 2016.

[60] H. Minkowski. Space and time. *The Principle of Relativity. Dover Books on Physics.*, pages 73–91, 1952.

[61] J. Mirabel, S. Tonneau, P. Fernbach, A. K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux. Hpp: A new software for constrained motion planning. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 383–389. IEEE, 2016.

[62] R. Mirsky and Y. Gal. SLIM: Semi-lazy inference mechanism for plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-16)*, 2016.

[63] R. Mirsky, Y. Gal and S. M. Shieber. CRADLE: an online plan recognition algorithm for exploratory domains. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(3), 2017.

[64] A. Nash and S. Koenig. Any-angle path planning. *AI Magazine*, 34(4):85–107, 2013.

[65] J. Nieto, E. Slawinski, V. Mut, and B. Wagner. Online path planning based on rapidly-exploring random trees. In *IEEE International Conference on Industrial Technology (ICIT)*, pages 1451–1456. IEEE, 2010.

[66] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[67] B. Paulson and T. Hammond. A system for recognizing and beautifying low-level sketch shapes using NDDE and DCR. In *ACM Symposium on User Interface Software and Technology (UIST-07)*, 2007.

[68] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[69] G. Pellegrino, L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Understanding motor events: A neurophysiological study. *Experimental Brain Research*, 91(1):176–180, 1992.

[70] R. F. Pereira and F. Meneguzzi. Landmark-based plan recognition. *arXiv preprint arXiv:1604.01277*, 2016.

[71] R. F. Pereira and F. Meneguzzi. Landmark-Based Plan Recognition. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-16)*, 2016.

[72] R. F. Pereira and F. Meneguzzi. Goal and Plan Recognition Datasets using Classical Planning Domains. *Zenodo*, July 2017.

[73] R. F. Pereira, N. Oren, and F. Meneguzzi. Landmark-Based Heuristics for Goal Recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-17)*. 2017.

[74] M. E. Pollack, L. Brown, D. Colbry, C. E. McCarth, C. Orosz, B. Peintner and S. Ramakrishnan. Autominder: an intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems*, 44(3):273–282, 2003.

[75] F. Pommerening and M. Helmert. A normal form for classical planning tasks. In *Proceedings of The International Conference on Automated Planning and Scheduling (ICAPS-15)*, 2015.

[76] J. Porteous and S. Cresswell. Extending Landmarks Analysis to Reason about Resources and Repetition. In *Proceedings of the 21st Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG '02)*, 2002.

[77] D. V. Pynadath and S. Marsella. Psychsim: Modeling theory of mind with decision-theoretic agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1181–1186, 2005.

[78] D. V. Pynadath and M. P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *UAI-2000*, pages 507–514, 2000.

[79] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, page 5. Kobe, 2009.

[80] M. Ramírez and H. Geffner. Plan recognition as planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1778–1783, 2009.

[81] M. Ramírez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-10)*, 2010.

[82] M. Ramírez and H. Geffner. Goal recognition over POMDPs: Inferring the intention of a pomdp agent. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 2009–2014, 2011.

[83] S. Richter and M. Westphal. The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR-10)*, 39(1):127–177, 2010.

[84] G. Rizzolatti. The mirror neuron system and its function in humans. *Anatomy and Embryology*, 210(5–6):419–421, 2005.

[85] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi. Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, 3(2):131–141, 1996.

[86] D. Rubine. *Specifying gestures by example*, volume 25. ACM, 1991.

[87] A. Sadeghipour and S. Kopp. Embodied gesture processing: Motor-based integration of perception and action in social artificial agents. *Cognitive Computation*, 3(3):419–435, 2011.

[88] A. Sadeghipour, R. Yaghoubzadeh, A. Rüter, and S. Kopp. Social motorics–towards an embodied basis of social human-robot interaction. In *Human Centered Robot Systems*, pages 193–203. Springer, 2009.

[89] E. Scala, P. Haslum, D. Magazzeni, and S. Thiébaux. Landmarks for Numeric Planning Problems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4384–4390, 2017.

[90] E. Scala, P. Haslum, S. Thiébaux, and M. Ramírez. Interval-based relaxation for general numeric planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-16)*, pages 655–663, 2016.

[91] C. Schmidt, N. Sridhan, and J. Goodson. The plan recognition problem: an intersection of psychology and artificial intelligence. *Artificial Intelligent*, 11:45–83, 1978.

[92] T. M. Sezgin and R. Davis. Hmm-based efficient sketch recognition. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 281–283. ACM, 2005.

[93] D. Sharon and M. Van De Panne. Constellation models for sketch recognition. In *Proceedings of the Third Eurographics Conference on Sketch-Based Interfaces and Modeling*, pages 19–26. Eurographics Association, 2006.

[94] S. Sohrabi, A. V. Riabov, and O. Udrea. Plan recognition as planning revisited. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 3258–3264, 2016.

[95] S. Sohrabi, A. V. Riabov, and O. Udrea. Plan recognition as planning revisited. *The Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3258–3264, 2016.

[96] K. W. Strabala, M. K. Lee, A. D. Dragan, J. L. Forlizzi, S. Srinivasa, M. Cakmak, and V. Micelli. Towards seamless human-robot handovers. *Journal of Human-Robot Interaction*, 2(1):112–132, 2013.

[97] I. A. Şucan and L. E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *Algorithmic Foundation of Robotics VIII*, pages 449–464. Springer, 2010.

[98] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[99] G. Sukthankar, R. P. Goldman, C. Geib, D. V. Pynadath, and H. Bui, editors. *Plan, Activity, and Intent Recognition*. Morgan Kaufmann, 2014.

[100] M. Tambe and P. S. Rosenbloom. Resc: An approach for real-time, dynamic agent tracking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, volume 95, pages 103–110, 1995.

[101] R. S. Tumen, M. E. Acer, and T. M. Sezgin. Feature extraction and classifier combination for image-based sketch recognition. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, pages 63–70. Eurographics Association, 2010.

[102] F. Ulgen, A. C. Flavell, and N. Akamatsu. Geometric shape recognition with fuzzy filtered input to a backpropagation neural network. *IEICE Transactions on Information and Systems*, 78(2):174–183, 1995.

[103] O. Uzan, R. Dekel, O. Seri and Y. Gal. Plan recognition for exploratory learning environments using interleaved temporal search. *AI Magazine*, 36(2):10–21, 2015.

[104] D. L. Vail, M. M. Veloso and J. D. Lafferty Conditional random fields for activity recognition In *Proceedings of the 2007 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-07)*, 1331–1338, 2007.

[105] S. Vattam, D. Aha and M. W. Floyd. Error tolerant plan recognition: an empirical investigation. *The Twenty-Eighth International Flairs Conference*, 2015.

[106] M. Vered and G. A. Kaminka. Towards sketch recognition by mirroring. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, pages 1867–1868. 2015.

[107] M. Vered and G. A. Kaminka. Heuristic online goal recognition in continuous domains. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

[108] M. Vered, G. A. Kaminka, and S. Biham. Online goal recognition through mirroring: Humans and agents. In *Proceedings of the Annual Conference on Advances in Cognitive Systems (ACS-16)*, 2016.

[109] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

[110] Z. Wang,K. Mülling, M. P. Deisenroth, H. Ben Amor, D. Vogt, B. ölkopf, and J. Peters. Probabilistic movement modeling for intention inference in human–robot interaction. *The International Journal of Robotics Research*, 32(7):841–858, 2013.

[111] J. S. Weber and M. E. Pollack. Evaluating user preferences for adaptive reminding. *Extended Abstracts on Human Factors in Computing Systems (ACM)*, 2949–2954, 2008.

[112] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose and J. M. Rehg. A scalable approach to activity recognition based on object use. *IEEE 11th International Conference on Computer Vision (ICCV-07)*, 1–8, 2007.

[113] Q. Zhu. Hidden markov model for dynamic obstacle avoidance of mobile robot navigation. *Robotics and Automation, IEEE Transactions on*, 7(3):390–397, 1991.

לבסוף אנו מפתחים גישה חדשה לזיהוי תכניות בסביבות רציפות ודיסקרטיות על ידי שילוב של שיטת *השיקוף* ושיטה קיימת המנצלת אבני דרך בסביבה. ניסויים מקיפים מראים ששיטה חדשה זו הינה יותר מדויקת ויעילה משיטות קיימות.

תקציר

זיהוי תכניות הינה משימה של הסקת תכניתו של סוכן בהתבסס על רצף חלקי של תצפיות של הפעולות שאותן ביצע הסוכן. הגרסה הלא-מכוונת של הבעיה הינה הגרסה שבה הרצף כולו מתקבל כקלט. לעומת זאת בגרסה המכוונת של הבעיה התצפיות מסופקות אחת אחרי השנייה לפי הסדר.

גישה חדשה ומעוררת השראה - *זיהו תכניות על ידי תיכנון* - משתמשת ביכולת לתכנן תכניות על מנת לייצר בצורה דינאמית תכניות שונות אשר ישמשו לזיהוי התצפיות. בכך בעצם מוחקים את הצורך בשימוש בספריית תכניות קיימת. בסביבות רציפות ועבור זיהוי תכניות מקוונות. שיטה זו נתקלת במספר קשיים.

הקושי הראשון הינו שגרסאות קודמות של פורמליזציה לבעיית זיהוי זו התחייבו מבעוד מועד לרמת דיסקרטיציה מסוימת של הסביבה ושל תצפיות הסוכן. התחייבות זו עלולה להוביל לירידה בדיוק הזיהוי. על מנת להתמודד עם בעיה זו אנו קודם כל מספקים פירמול חדש ומקיף לבעיית זיהוי התכניות. פירמול זה לוקח בחשבון סביבות רציפות וגם סביבות דיסקרטיות. לאחר מכן אנו מציגים את שיטת ה *שיקוף*, שיטה חדשה, העובדת בסביבות מכוונות עבור בעיית זיהוי התכניות. ההשראה עבור גישה זו נבעה ממחקרים אודות קיום נוירוני מראה הנמצאים בתוך המוח האנושי. שיטת *השיקוף* נמנעת מההנחה המקובלת אודות קיום ספרייה מקיפה של תכניות המייצגות את כל הדרכים להשיג כל מטרה. גישות המסתמכות על הנחה זו מוגבלות רק לידע המיוצג בספרייה ולכן אינן יעילות. אנו מראים שדרך שיטת ה *שיקוף* - צורה כללית יותר של זיהוי תכניות על ידי תיכנון - אנו נוכל לתכנן תכניות גם בעולמות רציפים.

אנו מספקים הוכחות פורמליות עבור יעילות גישת *השיקוף* ובוחנים את הגישה בצורה אמפירית על ידי יותר מ 1000 בעיות זיהוי תכניות שונות בשלושה עולמות רציפים ושישה עולמות דיסקרטיים קלאסים. בנוסף אנו משווים את תוצאות הגישה לתוצאות שהושגו על אותן בעיות זיהוי על ידי אנשים וכך מספקים תובנות לגבי תהליך זיהוי התכניות המתבצע במוח האנושי. לבסוף אנחנו משווים את שיטת *השיקוף* לשיטות קיימות העובדות עם ספריות של תכניות.

הקושי השני איתו עלינו להתמודד הוא שבצורתו הנוכחית, זיהוי תכניות על ידי תיכנון אינו עובד ביעילות עבור בעיות מכוונות ולכן אינו יכול להשתמש במתכנני התכניות הקיימים בעולמות הרציפים. על מנת להתמודד עם קושי זה אנו מזהים שתי נקודות מפתח בתהליך השיקוף. באמצעות הכנסת יוריסטיקות מתאימות בנקודות אלו נוכל לשפר את זמן הריצה עבור הגרסאות המכוונות של הבעיה. אנו מביאים דוגמאות ספציפיות ליורסיטיקות העובדות בעולמות רציפים, מוכיחים את יעילותן ועורכים מאות ניסויים הבודקים גם את יעילות הגישה וגם את אחוזי ההצלחה. הניסויים מתבצעים בסביבת תלת מימדית של בעיית ניווט ובנוסף בסימולציית רובוטים הדורשים לבצע שיתוף פעולה. על מנת לבדוק את עמידות השיטה אנו עורכים ניסויים נוספים ברמות שונות של קושי כאשר המטרות הסופיות מפוזרות בצורה אחידה או מקובצות ביחד במרחב.

תוכן עניינים

עבודת דוקטורט זו בוצעה בהנחייתו של פרופסור גל קמינקא, המחלקה למדעי המחשב, אוניברסיטת בר-אילן.

שיקוף - גישה כללית עבור
זיהוי מטרות או תכניות

חיבור לשם תואר "דוקטור לפילוסופיה"

מאת :

מור ורד

המחלקה למדעי המחשב

הוגש לסנט של אוניברסיטת בר-אילן

רמת גן, ישראל טבת תשע"ח