Plan Recognition in Continuous Domains

Gal A. Kaminka, Mor Vered, Noa Agmon
Computer Science Department
Bar Ilan University, Israel
{galk,veredm,agmon}@cs.biu.ac.il

Abstract

Plan recognition is the task of inferring the plan of an agent, based on an incomplete sequence of its observed actions. Previous formulations of plan recognition commit early to discretizations of the environment and the observed agent's actions. This leads to reduced recognition accuracy. To address this, we first provide a formalization of recognition problems which admits continuous environments, as well as discrete domains. We then show that through *mirroring*—generalizing plan-recognition by planning—we can apply continuous-world motion planners in plan recognition. We provide formal arguments for the usefulness of mirroring, and empirically evaluate mirroring in more than a thousand recognition problems in three continuous domains and six classical planning domains.

Introduction

Plan, activity, and intent recognition (PAIR) (Schmidt, Sridhan, and Goodson 1978; Sukthankar et al. 2014) 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).

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 (Kautz and Allen 1986; Avrahami-Zilberbrand and Kaminka 2005), deterministic and probabilistic grammars (Pynadath and Wellman 2000; Geib and Goldman 2011; Sadeghipour and Kopp 2011; Mirsky and Gal 2016), and other probabilistic models (Charniak and Goldman 1993; Bui 2003; Pynadath and Marsella 2005; Ramırez and Geffner 2011).

Unfortunately, early commitment to a fixed discretization within the plan library leads to inherent information loss. We show below that once discretization is fixed, there are always cases where the information loss will degrade performance.

One approach to avoid such early commitment is to apply PAIR methods based on domain theories, rather than plan libraries. 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) (Ramírez and Geffner 2010; Sohrabi, Riabov, and Udrea 2016; Freedman and Zilberstein 2017; Vered, Kaminka, and Biham 2016), uses off-the-shelf (OTS) planners in recognition. Thus potentially, using a motion planner such as RRT* (Nieto et al. 2010) in PRP could enable direct recognition in continuous spaces. However, current PRP methods are limited to discrete classical-planning domains, and cannot use OTS motion planners.

We present *Mirroring*, an alternative formulation of *plan recognition as planning* suitable for continuous domains, as well as discrete domains. The *Mirroring* formulation relies on an extension of classical planning to continuous domains, which includes discrete settings as a special case. We empirically evaluate the recognition formulation in over a thousand plan recognition problems, spread across three continuous domains and six classical planning domains, often used for benchmarking plan recognition methods.

Motivation

Most past approaches to plan recognition utilize a given *plan library* representing 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. 2014) for a thorough recent survey). A common theme is that they address continuous domains only through fixed, a priori discretization.

However, as shown also in related tasks (such as path planning) the loss of information inherent in discretization can be detrimental (Nash and Koenig 2013). As an example, consider the 2D recognition problem in Figure 1. The initial agent position is represented by the letter *I* and the problem is to recognize towards which one of two goals A, B the agent is heading. The space has been discretized using a grid, and thus ob-

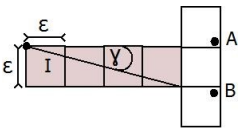

Figure 1: Discretized goal recognition.

servations are of grid cells of size $\epsilon \times \epsilon$. The actual trajectory of the agent (shown in the dark line, starting at top-left corner) clearly favors goal B (under assumption of rationality). The discretized observation sequence of grid cells (gray) do not convey this information; the goal can be recognized only when the agent moves into a grid cell containing either A or B. 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. Of course, it is also the case that given any recognition problem in a continuous domain, we can always find a discretization that works (Thm. 2[1]). Thus the problem is not that a recognizer must not commit to a discretization level; it is that *it should not commit to it early, before it is given the recognition problem*.

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 distinguished in the continuous domain.*

Proof. (Sketch.) The recognition problem example described above, using Figure 1, can be constructed for any given ϵ. In other words, a recognition problem exists for any ϵ such that the goals are distinguishable in the continuous case (e.g., by choosing ϵ after receiving the observations), and are not distinguishable in the discrete case. □

Theorem 2. *For every goal recognition problem R where goals in R are distinguishable in the continuous domain, there is a grid size ϵ, such that the goals can be distinguished also in the discrete domain.*

Proof. (Sketch.) The reverse case is also easily shown. Suppose we know the continuous observations and the goals they distinguish. This means that there exists two points on the line in Figure 1 which allow us to distinguish between the goals based on some recognition procedure. We can then choose a sufficiently small ϵ such that two observations are separated by a distance greater than ϵ, which would allow us to distinguish the points, and hence their use as evidence for the goals. □

Recognition (PAIR) approaches which rely on a domain-theory, rather than a plan library, may offer an opportunity to avoid the early discretization commitment, as they generate recognition hypotheses dynamically, ad-hoc. Some compute specialized invariants and heuristics from the domain theory and incoming observations (Hong 2001; Martin, Moreno, and Smith 2015). Others use the domain theory to compute planning landmarks, and then use them to rank goal hypotheses and speed up the recognition process (Pereira, Oren, and Meneguzzi 2016). All of these work with discrete domain theories, and do not directly translate to continuous domains.

Other domain-theory methods are very relevant to our approach. *Plan recognition as planning* (PRP) (Ramírez and Geffner 2009; 2010; Freedman and Zilberstein 2017; Masters and Sardina 2017) uses domain theories with planners to dynamically generate hypotheses for plan recognition. The (2009) formulation relied on modified planners and could not probabilistically rank the hypotheses. The (2010) 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.

Most closely related to our work here is Sohrabi et. al (2016) which modified the PRP formulation—still in discrete domains—to allow observation of effects, and without computing the plan $p_{\overline{O}}$. Instead, they use a single call to a k-best planner to sample the plans explaining the observations. While this allows them to address erroneous observations which we do not address, it requires a specialized planner, and still assumes actions with instantaneous, discrete effects. Vered et al. (2016; 2017) developed an *online navigation goal recognition* method for continuous domains which uses OTS motion planners. It shares certain elements with the formulation we present, but does not generalize to discrete domains.

The approach we take 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 (Şucan, Moll, and Kavraki 2012). 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 (Fox and Long 2003) 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., (Hoffmann 2003; Li and Williams 2008; Coles et al. 2012; Fernández-González, Karpas, and Williams 2015; Scala et al. 2016; Illanes and McIlraith 2016). Of these, our model is closest to the latter, and it also borrows some of the constraints of Transitional Normal Form for planning (Pommerening and Helmert 2015). We leave recognition in mixed domains, and non-classical planning extensions (e.g., non-deterministic effects, sensory actions) to future work.

Plan Recognition in Continuous Domains

We present "Mirroring" a new extended formulation for plan recognition. We first extend a classical planning model to continuous domains. Then, we define plan recognition problems using this model. Finally we present the Mirroring solution method.

Plans in Continuous Domains

We define *domain theories* as collections of states. *Discrete actions* transition between and through them. We then define

[1] We thank an anonymous reviewer for pointing this out.

plans in these domains as sequences of actions that take the agent from a specified initial state to a specified *goal state*.

Domain Theories: States, Actions, and Transitions. In the rich tradition of factored representations in planning, a *domain theory* 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 (Illanes and McIlraith 2016). 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 $s = \{x(r) = 50.34, y(r) = 24.0, \theta(r) = 90°\}$ 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 $s_2 := s \setminus \{\theta(r) = 90°\}$ 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, continuous domains raise two challenges to this. An example for those is 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 $(s_0, \ldots, s_m)$, $m \in I \subset \mathbb{N}^+$, 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 s_i along the path from s_0 to s_m. By definition, s_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 s_i is the i'th state in p, given the indexes I, generated by the mapping f_a. We notate $s_i \leq s_j$ when $i \leq j$, i.e., s_i is earlier in the sequence defining p. $\texttt{BEG}(p)$ is s_0, and $\texttt{END}(p)$ is the final state s_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 $\text{TRACE}(\pi)$. 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 $cost(\pi) = cost(\text{TRACE}(\pi))$.

Plan Recognition Problems

We begin by defining 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 1 (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 2 (Plan Recognition)**.** *Let R be an offline recognition problem. The plan recognition task is to determine π_R,*

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

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 3 (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:

- $m_\pi^O(0) := (\texttt{BEG}(p), \texttt{BEG}(\sigma))$
- $m_\pi^O(1) := (\texttt{END}(p), \sigma_\Omega))$*, where* $\sigma_\Omega \leq \texttt{END}(\sigma)$*, i.e., may not be the last state in σ*
- $\forall r \in (0,1), \exists i,j$ *s.t.* $m_\pi^O(r) = (p_i, \sigma_j)$ *and* $j < \Omega$*.*
- *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* $r \leq l$*.*
- *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 (Ramírez and Geffner 2009) 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 $P(\pi|g)$ (Ramírez and Geffner 2009, 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 3). For any $r \in [0,1]$, we have $m(r) = (p_i, \sigma_j)$. Then: $E(m(r)) := E(p_i, \sigma_j)$, where E is a distance metric (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. In the navigation domains used in the experiments, we used the Euclidean distance.

Definition 4 (Matching Error and Best Matching)**.** *Given a plan π, an observation sequence O, and a matching m_π^O between them (Definition 3), the matching error is given by*

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

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

$$\widehat{m_\pi^O} = \operatorname*{argmin}_{m_\pi^O} error(m_\pi^O)$$

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, $P(O|\pi) := \frac{1}{1+error(\widehat{m_\pi^O})}$.

A Solution Method and Shortcut

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.

We offer a shortcut to generating hypotheses which are guaranteed to minimize matching errors. Instead of generating hypothesized plans and then test 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 π_g^O for all g is straightforward in most OTS discrete-domain planners, and indeed used in (Ramírez and Geffner 2010). 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., (Mirabel et al. 2016).

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 plan $\hat{\pi}$—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. (2016) for a potential treatment in discrete domains; the continuous domain case is open.

Evaluation

We empirically evaluated the performance of the plan recognition procedure 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. In all cases, we used unmodified OTS planners. In some cases, observations were revealed incrementally, rather than given all at once. For these, we utilized multiple calls to the planner, to re-evaluate hypotheses as more observations were available, essentially solving multiple offline recognition problems. Vered and Kaminka (2017) propose an efficient method for doing so; we followed their baseline, non-heuristic approach, to saving planners calls.

The effects of planner choice

We first evaluate our 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 Open Motion Planning Library (OMPL (Şucan, Moll, and Kavraki 2012)) *cubicles* environment along with the default robot, and experimented with four off-the-shelf OMPL planners.

Each call to the planner was given a time limit of 1 sec. For cost, we measured the length of the path. We set 11 points spread through the cubicles environments. We then generated two observed paths from each point to all others, for *a total of* $220 = 110 \times 2$ *recognition problems*. Each plan consisting of between 20-75 points. The observations were obtained by running the asymptotically optimal planner RRT* on each pair of points (time limit of 5 minutes).

In general, finding the optimal plan is not trivial. We experimented with several planners that differ in their optimality guarantees. Two from the RRT (Rapidly-exploring Random Trees) family (Nieto et al. 2010) offer some guarantees: RRT* guarantees asymptotic optimality; TRRT only guarantees asymptotic *near-* optimality preferring shorter solutions. The two other planners used offer no optimality guarantees: RRTConnect, and KPIECE1 (Şucan and Kavraki 2010).

We use the following criteria: (1) *Conv.*, the convergence measured as percentage of the number of observations from the end where the recognizer converged to the correct hypothesis (including 0 if it failed). Higher values indicate ear-

lier convergence and are therefore better;(2) *Rank*, the number of times they ranked the correct hypothesis at the top (i.e., rank 1), which indicates their general accuracy, hence a larger value is better;and (3) *Time*, the average run-time percent. This refers to a percentage of the total time made available to the planners. Smaller values indicate a more efficient process.

Table 1, columns 1–3, contrasts the results of the four planners, when used in our PRP 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.

However, the two top planners differ from each other very much in run-time (Table 1, 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 1: 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.

Sensitivity to recognition difficulty

In online, continuous domains, the difficulty of the recognition problem—which stems from ambiguity in matching hypotheses—can affect recognition performance (e.g., accuracy) as well as efficiency (e.g., run-time). We wanted to evaluate the sensitivity of the results shown above to the difficulty of the recognition problems. We therefore added 9 goal points to the recognition problems in the navigation domain (i.e., 19 potential goals in each recognition problem) 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. This increased ambiguity leads to greater recognition difficulty, as more hypotheses agree with the observations, over a longer observation sequence.

Table 1, 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.

From the results we can also compare 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%.

Discrete vs. Continuous

We further evaluate the use of the formulation in both discrete and continuous domains.

Testing in Six Discrete Domains. To demonstrate the generality of the approach, we re-ran the *entire* set of benchmark plan-recognition problems used in (Ramírez and Geffner 2010) and then in (Sohrabi, Riabov, and Udrea 2016; Pereira, Oren, and Meneguzzi 2016). 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_f* planner (Haslum 2006). We used mirroring to rank the goal hypotheses (cost of ideal plan vs cost of optimal plan that goes through the observations), and compared the results to the ranking generated by the PRP formulation in (Ramírez and Geffner 2010) (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. Our method ranked it as a winner, along with two others. In 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.

We note in passing that in general, run-times using mirroring were slightly better than using PRP. We are investigating potential reasons for this, but also clarify that the differences may not be significant.

Robot Cooperation Task in ROS. We used ROS (Quigley et al. 2009) to utilize our 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 (Cashmore et al. 2015). We used the ROS standard Gazebo simulator

		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 2: Continuous vs. discrete results (193 Problems). Rows marked *I* denote initial locations. *C,D* (columns) denote continuous and discrete recognizers.

	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 3: Shapes continuous vs. discrete results

to simulate an environment of a soccer field, free of any obstacles, 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. Another robot was positioned elsewhere, and observed the motions of the other robot. Its task was to head towards and position itself in one of several pre-selected locations, to support its teammate. 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 in parallel sides (init point 1 and two Figure 2) and one point past the observed robot in the middle (init point 3). We ran 10–20 runs from each initial position to each of the goals for a total of 193 problems.

Figure 2: Experiment setup (via RVIZ)

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 the same two measures of recognition performance as before (convergence and number of times the true goal was ranked first), though instead of reporting on percentages, Table 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 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. 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. We additionally evaluated our continuous goal recognizer on the shape sketch recognition domain introduced in (Vered, Kaminka, and Biham 2016). 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.

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 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.

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 3 using the previously mentioned performance evaluation criteria. 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 correct rankings, the results differ, indicating that a 1°angle discretization is insufficient.

Summary

In this paper we presented *mirroring* a novel formulation of plan recognition in continuous domains (and generalizing to discrete domains). The key insight is that by using continuous-domain or 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. By using a form of plan recognition by planning, mirroring allows the use of off-the-shelf unmodified planners. It gives rise to a recognition procedure that uses two calls to a planner for each goal, and accounts for missing observations. The formulation has been compared in discrete domains to prior formulations (which cannot address continuous domains), and has been evaluated with four standard motion planners, one robotics planner, and one domain-dependent planner—all without modifying the planners in any way. Overall, we report on results from 9 domains. Future work involves extending towards, and working with, mixed discrete-continuous planners (Coles et al. 2012; Fernández-González, Karpas, and Williams 2015; Scala et al. 2016; Illanes and McIlraith 2016), and investigating a principled approach to addressing noisy observations, which the recognizer should ignore. Additionally, we would like to tackle multi-agent plan recognition, so as to extend the impact of this work in robotics.

Acknowledgments

We thank Kobi Gal, Miguel Ramirez, and Felipe Meneguzzi for very valuable advice. This research was supported in part by ISF grant # 1865/16. Thanks to K. Ushi.

References

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

Bui, H. H. 2003. A general model for online probabilistic plan recognition. In *IJCAI*, volume 3, 1309–1315.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtós, N.; and Carreras, M. 2015. ROSPlan: Planning in the robot operating system. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 333–341.

Charniak, E., and Goldman, R. P. 1993. A bayesian model of plan recognition. *Artificial Intelligence* 64(1):53–79.

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

Fernández-González, E.; Karpas, E.; and Williams, B. C. 2015. Mixed discrete-continuous heuristic generative planning based on flow tubes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1565–1572.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Freedman, R., and Zilberstein, S. 2017. Integration of planning with recognition for responsive interaction using classical planners. In *AAAI*, 4581–4588.

Geib, C., and Goldman, R. 2011. Recognizing plans with loops represented in a lexicalized grammar. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 958–963.

Haslum, P. 2006. Improving heuristics through relaxed search - an analysis of TP4 and HSP*a in the 2004 planning competition. *Journal of Artificial Intelligence Research* 233–267.

Hoffmann, J. 2003. The Metric-FF planning system: Translating"ignoring delete lists"to numeric state variables. *Journal of Artificial Intelligence Research* 20:291–341.

Hong, J. 2001. Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research* 15:1–30.

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

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

Li, H. X., and Williams, B. C. 2008. Generative planning for hybrid systems based on flow tubes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 206–213.

Martin, Y. E.; Moreno, M. D. R.; and Smith, D. E. 2015. A fast goal recognition technique based on interaction estimates. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 761–768.

Masters, P., and Sardina, S. 2017. Cost-based goal recognition for path-planning. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 750–758.

Mirabel, J.; Tonneau, S.; Fernbach, P.; Seppälä, A.-K.; Campana, M.; Mansard, N.; and Lamiraux, F. 2016. HPP: a new software for constrained motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Mirsky, R., and Gal, Y. K. 2016. SLIM: Semi-lazy inference mechanism for plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

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

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

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2016. Landmark-based heuristics for goal recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press.

Pommerening, F., and Helmert, M. 2015. A normal form for classical planning tasks. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

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

Pynadath, D. V., and Wellman, M. P. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, 507–514.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. Ros: an open-source robot operating system. In *ICRA workshop on open source software*. Kobe, Japan.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1778–1783.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ramırez, M., and Geffner, H. 2011. Goal recognition over pomdps: Inferring the intention of a pomdp agent. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009–2014. IJCAI/AAAI.

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

Scala, E.; Haslum, P.; Thiebaux, S.; and Ramirez, M. 2016. Interval-based relaxation for general numeric planning. In *Proceedings of the European Conference on Artificial Intelligence*, 655–663. IOS Press.

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

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 3258–3264.

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

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

Sukthankar, G.; Goldman, R. P.; Geib, C.; Pynadath, D. V.; and Bui, H., eds. 2014. *Plan, Activity, and Intent Recognition*. Morgan Kaufmann.

Vered, M., and Kaminka, G. A. 2017. Heuristic online goal recognition in continuous domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Vered, M.; Kaminka, G. A.; and Biham, S. 2016. Online goal recognition through mirroring: Humans and agents. *The Fourth Annual Conference on Advances in Cognitive Systems*.