# Online Recognition of Navigation Goals through Goal Mirroring

Paper # 698

## ABSTRACT

Goal recognition is the problem of inferring the (unobserved) goal of an agent, based on a sequence of its observed actions and their effects. 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. A recent approach—plan recognition by planning (PRP)—uses a planner to dynamically generate plans for given goals, eliminating the need for the traditional plan library. For online recognition, PRP based recognizers inefficiently make $2|O||G|$ calls to the planner, where $|O|$ is the number of observations, and $|G|$ the number of goals. In this paper, we advocate *goal mirroring*, a PRP approach that directly tackles the efficiency of online goal recognition, and makes $(|O|+1)|G|$ calls in the worst case. We identify two independent decision points within the goal mirroring algorithm where heuristics may be used to further improve run-time, up to $2|G|$ in the best case. We specify heuristics for the domain of navigational goal recognition, and empirically evaluate goal-mirroring in hundreds of experiments in a 3D environment. We also evaluate the use of different planners, and varying recognition difficulty.

## CCS Concepts

**•Computing methodologies → Intelligent agents;**

## Keywords

plan-recognition, goal-recognition, online, mirroring

## 1. INTRODUCTION

Goal recognition is the problem of inferring the (unobserved) goal of an agent, based on a sequence of its observed actions [10, 5, 4, 13]. It is a fundamental research problem in artificial intelligence, closely related to plan, activity, and intent recognition [25].

In *offline recognition* the entire sequence of observations is provided to the agent ahead of time. In contrast, in *online recognition* the sequence of observations is revealed incrementally instead of being known in advance, thus exacerbating an already hard problem.

The prevalent approach to goal recognition, both offline and online, relies on a dedicated *plan library*, a set of plans which represents all known ways to achieve known goals [25].

**Appears in:** *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

These recognition methods vary in the expressiveness of the representation and efficiency of the inference algorithms used. While powerful when the plans are known, these methods fail when the observations come from an unknown plan to achieve a known goal.

A recent, promising approach is plan recognition by planning (PRP), where a planner is used as a black box, to dynamically generate plans that are matched against the observations, eliminating the need for a plan library [19, 22]. This approach targets discrete domains only, and is inefficient for online recognition where it would produce $2|O||G|$ calls to the planner, where $|O|$ is the number of observations, and $|G|$ the number of goals.

This paper advocates *goal mirroring*. Like [19], goal mirroring uses a planner, to generate recognition hypotheses. However, it is designed for efficient, online recognition in continuous environments by using a motion planner, with a baseline number of calls to the planner of $(|O|+1)|G|$.

We identify two key decision points where, by inserting heuristics for navigational goal recognition, we can further influence the number of calls to the planner and the overall run-time. In the first decision point, a heuristic may reduce the number of overall calls to the planner from $|G|(|O|+1)$ to $|G|(1+1)$ at best. In the second decision point, a heuristic prunes unlikely goal candidates, incrementally reducing $|G|$ as observations are added, making fewer calls to the planner for each observation.

We describe the goal mirroring algorithms in detail, and instantiate the two heuristics for recognition of navigation goals in continuous 3D environments. We report on extensive experiments examining the recognition success and efficiency in the mirroring algorithms, over hundreds of navigational goal recognition problems, varying the difficulty of the recognition and the motion planner used.

## 2. RELATED WORK

Goal recognition has many applications, including for example human-robot and human-computer interaction [28], command prediction [13, 5] intelligent learning environments [26, 1], recognizing navigation goals [14, 29], etc. Sukthankar et al. provide a survey of recent work in goal and plan recognition, most of it assuming a library of plans for recognition of known goals [25]. Such methods include a variety of probablistic inference techniques [7, 2], grammar-based approaches [18, 9, 16], and many more. The use of a library limits recognition capabilities to known plans. If the observations are of an unknown plan, even leading to a known goal, library-based methods fail.

Some prior investigations have begun to explore alternative methods. Hong uses a specialized representation and online algorithm to generate possible goals without a plan library [10]. Baker et.al use policies to calculate goal likelihoods by marginalizing over possible actions and generating state transitions, using only limited replanning [3]. Geib [8], and Sadeghipour et al. [21] offer methods that utilize the same library for both planning and plan-recognition. Keren et al. [11] investigates changing the domain to facilitate the goal-recognition process.

We build on *plan recognition by planning* (PRP) [19] which avoids storing a plan-library, but instead utilzes a *planner* to generate plans 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. Sohrabi et. al have further improved this approach to better address unreliable observations and recognize plans as well as goals [22]. 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. Martin et. al refrain from using a planner in the PRP. Instead, for each goal they compute cost estimates using a plan graph [15]. This approach is complementary to ours. Ramirez and Geffner [20] 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.

Vered and Kaminka present an online goal mirroring recognizer [27], similar to the one we present here as the baseline. However, in this paper we go a significant step beyond by introducing new mirroring algorithms including heuristics that aim to improve efficiency. We instantiate such heuristics for recognizing navigation goals and provide a careful analysis.

## 3. GOAL MIRRORING

We begin by giving a general definition of the navigational goal recognition problem. We present an *offline* PRP goal recognition algorithm, (Section 3.1), from which we deduce a naive online recognizer (Section 3.2). We proceed to develop an efficient, baseline *online* PRP recognizer, and highlight the key decision points in the recognition process, where inserted heuristics can improve the efficiency of the online algorithm (Section 3.3). We discuss such heuristics for the task of recognizing navigation goals (Section 3.4).

### 3.1 The Goal Recognition Problem

We define $R$, the *navigational goal recognition* problem as a quintuple $R = \langle W, I, G, O, M\rangle$. $W \subseteq \mathbb{R}^n$ is the world in which the navigation takes place, as is defined in standard motion planning [12]. $I \in W$ represents the initial pose of the agent. $G$ is a set of goals; each goal $g \in W$, i.e., a point. $O$ is a set of observations, where for all $o \in O$, $o \subset W$, i.e., each observation is of a specific subset of the work area i.e., a point or trajectory. $M$ is a set of *plans*, each defined as a trajectory that begins in $I$, and ends in one of the goal positions $g \in G$. For each goal $g$, there exists at least one plan $m_g$ that has it as its end point. Given the problem $R$, a solution to the *goal recognition* problem is a specific goal $g \in G$ that *best matches* the observations $O$. This choice of $g$ is done by finding a plan $m_g \in M$ that *best matches* the observations, and has $g$ as its ending goal position. We explain below.

Solution candidates $m_g \in M$ must minimize the error in matching the observations, i.e., minimize the accumulating distance between all observations $O$, and the trajectory defined by $m_g$ (see [27] for a related formal definition). This is a *necessary*, but *insufficient* condition; in general, any number of potential plans may perfectly match the observations but differ in the unobserved part. Thus we build on a second requirement proposed in [19], which is that $m_g$ be *rational*. $m_g$ must also closely match the ideal plan $i_g \in M$ which a perfectly rational agent would have taken from the initial state $I$ to the goal state $g$. If $m_g$ does not closely match $i_g$ then the agent is presumed to be pursuing an alternative goal.

PRP recognizers avoid representing the plans explicitly, as a library of plans to be used for recognition. Instead, the set of plans $M$ is only implicitly represented, by using a planner to generate solution candidates on the fly. The planner is used at least twice, *for each goal* $g \in G$:

1. It is called to generate the ideal plan $i_g \in M$, which is a sequence of states (a path in continuous spaces) from the initial state $I$ to the goal $g$, ignoring any knowledge of observations.
2. It is called as part of the generation of a candidate $m_g$, so that it can be matched against the observations (necessity condition) and against $i_g$ (sufficiency condition).

In goal mirroring, this is done as follows. First, we observe that in both principle and practice, there are infinite plans that do not match the observations. Thus instead of generating candidates $m_g$ which will often be disqualified, mirroring algorithms use the planner to generate candidate plans $m_g$ that *always* match the observations, by folding the observations into the generated plan. In discrete domains, Ramirez and Geffner offer an elegant way for doing this by modifying the domain input of PDDL-type planners. In continuous domains, we propose folding the observations into the candidate solution by breaking $m_g$ into two pieces:

- A plan prefix, (denoted $m_g^-$) is built by concatenating all observations in $O$ into a single (possibly discontinuous) trajectory. This is a valid plan in $W$, as plans are defined by their effects, which are trajectories, just as are observations (Sohrabi et al. take a similar stance [22]).
- A plan suffix (denoted $m_g^+$) is generated by calling the planner, to generate a trajectory from the last state (point) in the prefix $m_g^-$ (the ending point of the last observation in $O$) to the goal $g$.

Using $\oplus$ to denote trajectory concatenation, a plan $m_g \equiv m_g^- \oplus m_g^+$ is a trajectory from the first observed point in $O$, to $g$. Notice that $m_g$ necessarily *perfectly* matches the observations $O$, since it incorporates them.

The only remaining task is then to check that $m_g$ maximally matches $i_g$, the ideal plan. This is done by evoking a scoring procedure (denoted $match(m_g, i_g)$) which accepts $m_g$ and $i_g$ and provides a measure of their matching. Ramirez and Geffner [19] proposed looking at the *difference* in the cost (often, the length) of $m_g$ and $i_g$ (i.e.,

$cost(m_g) - cost(i_g)$, as a parameter to a Boltzmann distribution which they assumed to model the probability of $g$. Vered and Kaminka [27] advocated looking at the *ratio* ($cost(i_g)/cost(m_g)$) instead. In the experiments described later, we use the ratio.

Algorithm 1 is an *offline* algorithm for a PRP, continuous-domain goal recognizer. In offline goal recognition we assume that the set of observations $O$ is known ahead of time rather than being revealed incrementally. The algorithm follows the process described above. For all goals $g \in G$, the algorithm computes the ideal plan $i_g$ by calling on the planner (line 2). It then creates the prefix plan $m_g^-$ by concatenating together the observed state trajectories and points $o \in O$ (line 3; we use $\bigoplus$ to denote the application of the concatenation operator $\oplus$ to all observations $o \in O$, analogously to $\Sigma$ and $+$, respectively). In line 4 the planner is called a second time, to create the plan suffix $m_g^+$, which is a path from the ending point of $m_g^-$, denoted $end(m_g^-)$, to the goal. Then the candidate solution, $m_g$, is composed in line 5 and in line 6 is measured against $i_g$ using the scoring procedure described above. Greater score means closer matching of $m_g$ to $i_g$, and thus by implication, a better hypothesized matching between the observations and the associated goal $g$. Finally, these rankings are transformed into probabilities $P(G|O)$ via the normalizing factor $\eta = 1/\sum_{g \in G} score(g)$.

**Algorithm 1** PRP Offline Goal Recognizer ($R, planner$)

```
1: for all $g \in G$ do
2:     $i_g \leftarrow planner(I, g)$
3:     $m_g^- \leftarrow \bigoplus_{o \in O} o$
4:     $m_g^+ \leftarrow planner(end(m_g^-), g)$
5:     $m_g \leftarrow m_g^- \oplus m_g^+$
6:     $score(g) \leftarrow match(m_g, i_g)$
7: for all $g \in G$ do
8:     $P(g|O) \leftarrow \eta \cdot score(g)$
```

## 3.2 Online Recognition

In *online* goal recognition, the set of observations $O$ is revealed incrementally. As a naive solution, it is possible to re-run the offline algorithm for each new observation and for every goal. Algorithm 2 presents this process, which iterates for all observations and goals: the computation of $i_g$ (line 4) and the composition and scoring of $m_g$ (lines 7–8) are as before. The generation of the prefix $m_g^-$ is now done incrementally (line 5), and the generation of the suffix $m_g^+$ is carried out from the ending point of the most recent observation. Note that the evaluation of each $g$ in lines (9–10) is now done for each observation, as the algorithm provides online results.

This will result in an inefficient $2|O||G|$ number of calls to the planner, $2|G|$ calls for each incoming observation. In the rest of this section we improve on this upper bound.

**Baseline method: No recomputation of $i_g$.** The first immediate step to address in online conditions is the redundant re-computation of the ideal plan, $i_g$ in Alg. 2, line 4. This plan is recalculated for every new observation, even though the result does not change, as the ideal plan does not take any observations into account and is calculated from the initial state $I$ to every goal.

In Algorithm 3, the generation of $i_g$ for all $g \in G$ is done

**Algorithm 2** Naive Online Goal Recognition ($R, planner$)

```
1: $\forall g, m_g^- \leftarrow \emptyset$
2: for all $o \in O$ do
3:     for all $g \in G$ do
4:         $i_g \leftarrow planner(I, g)$
5:         $m_g^- \leftarrow m_g^- \oplus o$
6:         $m_g^+ \leftarrow planner(end(o), g)$
7:         $m_g \leftarrow m_g^- \oplus m_g^+$
8:         $score(g) \leftarrow match(m_g, i_g)$
9:     for all $g \in G$ do
10:        $P(g|O) \leftarrow \eta \cdot score(g)$
```

only once (lines 2–3), as it was moved out of the main loop. While straightforward, this alteration is highly effective and will result in a reduction of the overall number of calls to the planner by half, from $2|O||G|$ to $(|O|+1)|G|$ as a baseline.

**Algorithm 3** Online Goal Mirroring No Recomputation Of Ideal Plan ($R, planner$)

```
1: $\forall g, m_g^- \leftarrow \emptyset$
2: for all $g \in G$ do
3:     $i_g \leftarrow planner(I, g)$
4: for all $o \in O$ do
5:     for all $g \in G$ do
6:         $m_g^- \leftarrow m_g^- \oplus o$
7:         $m_g^+ \leftarrow planner(end(o), g)$
8:         $m_g \leftarrow m_g^- \oplus m_g^+$
9:         $score(g) \leftarrow match(m_g, i_g)$
10:    for all $g \in G$ do
11:        $P(g|O) \leftarrow \eta \cdot score(g)$
```

**Second method: No re-computation of $m_g$.** Another possibility for further improvement is to make use of previously calculated paths not only for the calculation of the ideal plan, but for all incrementally revealed observations. Taking this approach to the extreme we will result in only calculating $2|G|$ plans. We calculate all $i_g$ ($|G|$ calls to the planner). We also calculate a complete set of $m_g$ plans, each a path from the first point of the first observation, to a different $g$. Thus we now have $i_g$ and $m_g$ which can be used for the initial ranking. These are saved and remain unchanged even as more observations come in. Whichever $m_g$ is closest to the observation is picked as the top ranking goal.

In a sense, this reverses the approach to generating $i_g$ and $m_g$. Instead of generating $m_g$ which perfectly match the observations, and then matching $m_g$ against $i_g$ to assess its rationality, this approach generates $m_g$ which are perfectly rational (they will be almost identical to $i_g$), and tries to match them against the observations.

In the best case, when the observations closely match the originally calculated paths, 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, if the observed agent is not perfectly rational, its observed trajectory will deviate from the saved $m_g$ 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 t 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 $i_g$ which differs from an equivalent—but different—ideal plan $m_g$ carried out by the observed agent.

## 3.3 Heuristic Online Goal Recognition

We identify two key decision points in the recognition process which can be used in order to further improve online recognition by relying on heuristics. These are: (i) recompute plans only if necessary, i.e., if the new observation may change the top hypothesized goal; and (ii) prune (eliminate) goals which are impossible or extremely unlikely (as they deviate too much from the ideal plan $i_g$). 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.

Algorithm 4 presents the general, heuristic, online goal mirroring procedure. The general flow is the same as Algorithm 3, with the following changes. First, we note that all observations are collected into a temporary observation buffer $U$, which keeps observations that have been made available, but not yet incorporated into the prefix plan $m_g^-$. Second, the plans $m_g$, $m_g^-$ and $m_g^+$ are saved between iterations and only recomputed if necessary (explained below). We also maintain the current top goal hypothesis, denoted $v$. This is used, together with the plan $m_v$ associated with this goal, to assess whether a new observation $o$ will cause a change in its ranking as the top hypothesis. This assessment is carried out heuristically, by the heuristic function $RECOMPUTE$ in line 7. If no recompuation is carried out, the hypotheses remain at their current ranking, and we loop back to line 6 to await a new observation. Otherwise, there is a need to recompute the hypotheses by regenerating $m_g$ based on new observations (lines 9–19). For every goal $g \in G$, a second heuristic decision is made (line 10) as to the maintained relevance of $g$ given the latest observation, and the projected plan suffix $m_g^+$. A goal $g$ whose associated plan $m_g$ is deemed extremely unlikely, or even impossible, is removed from the list of goals to consider (line 11). Otherwise, a recomputation of $m_g$ is carried out, using not just the last observation, but all observations since the last iteration through this inner block of code (lines 13–17). Finally, after the ranking of the goals, the new top-ranked goal is saved in $v$.

**Algorithm 4** HEURISTIC ONLINE GOAL MIRRORING $(R, planner)$

```
 1: $\forall g : m_g, m_g^+, m_g^- \leftarrow \emptyset$
 2: $U \leftarrow \emptyset$      ▷ holds observations between updates to $m_g^-$
 3: $v \leftarrow \emptyset$                                ▷ the top-ranked goal
 4: for all $g \in G$ do
 5:    $i_g \leftarrow planner(I, g)$
 6: for all $o \in O$ do
 7:    $U \leftarrow U \oplus o$
 8:    if $RECOMPUTE(m_v, o)$ then
 9:       for all $g \in G$ do
10:          if $PRUNE(m_g^+, o, g)$ then
11:             $G \leftarrow G - \{g\}$
12:          else
13:             $m_g^- \leftarrow m_g^- \oplus U$
14:             $U \leftarrow \emptyset$                        ▷ reset $U$
15:             $m_g^+ \leftarrow planner(end(m_g^-), g)$
16:             $m_g \leftarrow m_g^- \oplus m_g^+$
17:             $score(g) \leftarrow match(m_g, i_g)$
18:       for all $g \in G$ do
19:          $P(g|o) \leftarrow \eta \cdot score(g)$
20:       $v \leftarrow \text{argmax}_{g \in G} P(g|o)$
```

## 3.4 Heuristic Recognition of Navigation Goals

We introduce two heuristics applicable to the navigation goal recognition domain to be inserted in the key decision points in the process. These are inspired by studies of human estimates of intentionality and intended action [6]. 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.

### *3.4.1 The Recomputation Heuristic*

For every new observation we are called to recompute the suffix plan, $m_g^+$ for every goal $g \in G$. This could result in an additional $|G|$ calls to the planner for every added observation. However, for every new observation $o_i$ we have necessarily calculated the plans for the previous observation, $o_{i-1}$. 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 need not re-call the planner.

As previously discussed, we cannot realistically expect the observations to perfectly match the predictions. We need a heuristic that evaluates the difference between the newly seen observation $o$ and the predicted path ($m_g$). A suggestion for such a heuristic, in the domain of 3D navigation, is to measure the distance between the previously calculated plans and the new observation $o$, i.e., by measuring a vertical lines from the new observation point to the nearest point along each of the various $m_g$. If the distance between the new observation and the plan associated with the currently *highest* ranked goal ($m_v$) is smaller than all of the other distances (to other goals) we can assume that the planner is still on the same path and do not need to re-run the planner at all, keeping the previous goal rankings.

Formally, let $v \in G$ represent the highest ranked goal so far and $m_v$ be the calculated (top-ranked) plan associated with it. We will proceed to calculate $d_v = dist(o, m_v)$ as the distance between the observation $o$ and the plan $m_v$. If $d_v \leq min_{g \in G} dist(o, m_g)$ then goal $v$ remains the top-ranked hypothesis and we do not recompute the paths $m_g$, thus avoiding $|G|$ 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 plans $m_g$, the planner will only be utilized twice; once in the initial path generation ($i_g$) and once again for the first observation generating $m_g$ generating only $2|G|$ calls to the planner in total. However, the heuristic allows for non-best cases, where the paths $m_g$ have to be recomputed to account for observations that deviate from the predictions.

### *3.4.2 The Pruning Heuristic*

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, assuming that the observations and plans generated approximate the shortest possible paths between two coordinates.

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 $\alpha_g$, the angle created between the previous observation, the newly received observation and the previously calculated plan for every $g \in G$. $\alpha_g$ 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.

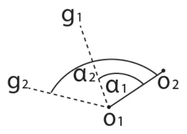


**Figure 1: Illustration of goal angles.**

Figure 1 presents an illustration of the heuristic approach in 2D. For each new observation, $o_i$, we measure the angle $\alpha_i$ created by the new observation, the previous observation $o_{i-1}$ and the previous plans generated for each goal $m_g$. In the figure, two different such plans are shown as the dashed lines between $o_{i-1}$ and each of the two possible goals $g_1$ and $g_2$. If the angle 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.

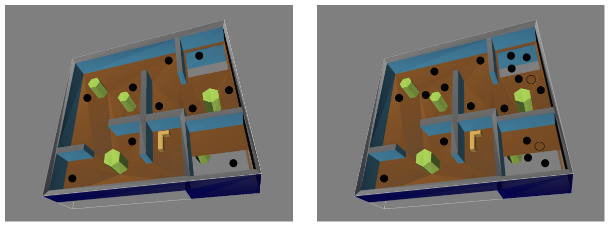
## 4. EVALUATION

We empirically evaluated the different approaches of online goal mirroring over hundreds of goal recognition problems while measuring both the efficiency of the approach in terms of run-time and overall number of calls to the planner, and the performance of the approach in terms of convergence time and how many times the correct goal was ranked first. We additionally wanted to evaluate the effects of planner choice on overall performance.

### 4.1 Experiment Setup

We implemented the different approaches of online goal mirroring to recognize the goals of navigation in 3D worlds. We used TRRT (Transition-based Rapidly-exploring Random Trees), an off-the-shelf planner that guarantees asymptotic near-optimality by preferring shorter solutions, available as part of the Open Motion Planning Library (OMPL [24]) along with the OMPL *cubicles* environment and default robot (Figure 2(a)). Each call to the planner was given a time limit of 1 sec. The cost measure being the length of the path. And for the *Pruning* heuristic we used a threshold angle of 120°.

We generated two observed paths from each point to all others, for *a total of* $110 \times 2$ *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 optimized planner that guarantees asymptotic near-optimality. The longer the run-time the more optimal the path. We used the OMPL *interpolate* method to generate between between 20 and 76 observed points for each path problem.



(a) Original Scenario (b) Clustered Scenario

**Figure 2: Visualization of both original and clustered goals environement**

*Measuring Recognition Results.*

Figure 3 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 run.

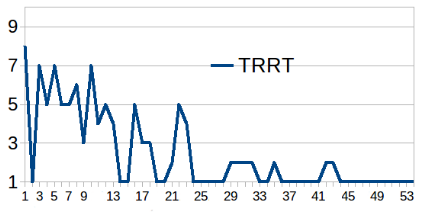


**Figure 3: Recognition result on specific problem.**

For instance, in Figure 3, after the 9th observation, the correct goal was ranked 3 (out of 10) by the recognizer. As more observations come in, it is only natural that the recognition problem becomes easier and indeed towards the end of the observation sequence the recognizer converges to ranking the correct goal at the top (rank 1). Such graphs can be drawn for any online recognition problem to compare the performance of different recognizers. When measuring recognition results we wanted to evaluate both the recognition performance as well as the efficiency of each online approach.

*Recognition Performance Measures.*

We use two measures of recognition performance : (1) the time (measured by number of observations from the end) in which the recognizer converged to the correct hypothesis (including 0 if it failed). Higher values indicate earlier convergence and are therefore better; and (2) the number of times they ranked the correct hypothesis at the top (i.e., rank 1), which indicates their general accuracy. The more frequently the recognizer ranked the correct hypothesis at the top, the more reliable it is, hence a larger value is better.

For example in Figure 3, the recognizer converged to the correct results at observation 44 out of 54. When normalizing for the observation sequence length, to allow comparison across different recognition problems, we measure the normalized convergence of the TRRT recognizer at 18.5%. Of course the earlier the convergence, the better. With regards to the second measure, 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.

*Efficiency Measures.*

In order to evaluate the overall *efficiency* of each approach we also used two separate measures: (1) the number of times the planner was called within the recognition process; and (2) the overall time (in sec.) spent planning. Though these two parameters are closely linked, they are not wholly dependant. While a reduction in overall number of calls to the planner will also necessarily result 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.

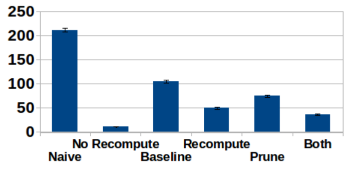
## 4.2 Effects of the different heuristic approaches

We ran the TRRT based recognizer on the above-mentioned 220 problems, the results are displayed in Figures 5 and 4. In all graphs the X axis denotes the different approaches; the first approach, *Baseline*, refers to the 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. 3). The second approach, *No Recomp*, refers to the method of no recomputation at all, meaning 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 $2|G|$ calls. The third approach, *Recompute*, measures the effect of the *Recompute Heuristic* which aims to reduce overall number of calls to the planner by measuring the difference between the previous path (Section 3.4.1). The fourth approach, *Prune*, measures the effect of the *Pruning Heuristic* which aims to reduce the overall number of goals by eliminating unlikely goal candidates (Alg. 4, Section 3.4.2). And the last approach, *Both*, measures the effects of utilizing both the *Pruning* and *Recompute Heuristics*.

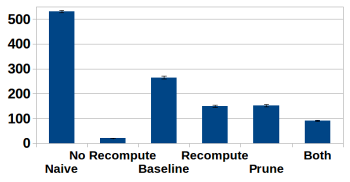
*Efficiency .*

In order to contrast the improvement in run-time and number of calls to the planner made by each of our separate approaches, we also compare against the naive online approach of re-running the offline PRP process for each incremental observation, marked *Naive* (Alg. 2, Section 3.2). By this comparison we can measure just how necessary and efficient the adjustments for online recognition are.

Figure 4(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 212 sec. for each problem. When only calling the planner once in the recognition process the *No Recompute* approach takes an average of only 6.1 sec. When not recomputing the ideal plans, the *Baseline* cuts the *Naive* time in half to an average of 105 sec. The *Pruning* heuristic reduces the average time further to only 74.9 sec. And the *Recomputation* heuristic further reduces the average time to only 49.9 sec. When utilizing both heuristics we achieved a reduction to 36.5 sec. an improvement



(a) Mean Run-Time (Sec.)



(b) Mean Number Of Calls To Planner

**Figure 4: Top: Mean run-time comparison. Bottom: Mean number of calls to planner comparison. In both lower values are better.**

of a substantial 82.8% from the *Naive* approach and 65.2% from the *No Recompute* approach.

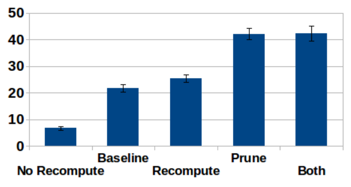
Figure 4(b) displays the average of the results in terms of number of calls made by the recognizer to the planner. Here the y axis denotes the overall number of calls. The *Naive* approach had an average of 530 calls to the planner while with no recomputation at all the *No Recompute* approach had an average of an extreemly efficient 20 calls, i.e. the number of goals. *Baseline* again reduced the number of calls by half compared to *Naive*, while the *Recomputation* and *Pruning* heuristics had similar success with a reduction to 148.9 and 151.4 calls each. Using both heuristics the number of calls was reduced to an average of only 90.6 calls, a reduction of 63.3% from the *Baseline* approach and of 82.9% 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 *recomputation* heuristic outperformed the *pruning* 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 80% from the naive approach. The most efficient method proved to be the *No Recompute* 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 algorithms developed in this paper.

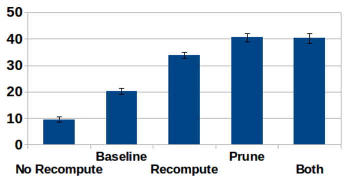
*Performance .*

Figure 5(a) measures the average convergence to the correct result percent, hence higher values mean earlier convergence and are thus better. As we can see with no reuse of the planner at all, for the *No Recompute* column we only get a 6.7% 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.

By converting to the online *Baseline* algorithm, we were able to more than double the convergence percent to 21.8%. Each incremental observation was now taken into account



(a) Mean Convergence Percent



(b) Mean Ranked First Percent

**Figure 5: Top: Comparison of convergence percent (Higher is better). Bottom: Comparison of correct ranking percent (Higher is better).**

during the reuse of the planner and therefore had greater weight on the ranking of the goals. Applying both the *Pruning* and *Recomputation* 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 *Recomputation* heuristic increases it to 25.4% and the *Pruning* to 42.2%, an improvement of 20.4% from the *Baseline* approach. Furthermore, we see that when utilizing both heuristics the high convergence level obtained by the *Pruning* heuristic is maintained.

Figure 5(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, *No Recompute*, the recognizer performs poorly with a low 9.5%. Refraining from recomputation of the ideal plan more than doubles the success here as well, to 20.2%. The *Recomputation* heuristic achieves 33.9% and the *Pruning* 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 *Pruning* method is obtained.

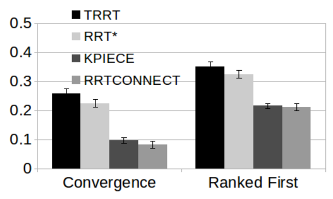
As seen employing the heuristics has made a big impact on overall performance successfully increasing convergence and overall correct rankings. The *Pruning* heuristic clearly outperformed the *Recomputation* heuristic in both measures however a combination of both heuristics maintains the high success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

## 4.3 The effects of planner choice

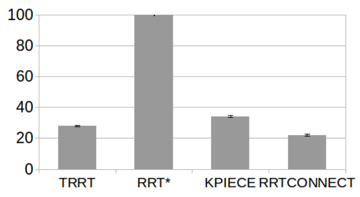
In planner based recognition approaches the recognizer continually utilizes an existing planner within the recognition process. Therefore, in the implementation level another key influential decision is planner choice. We experimented with four off-the-shelf OMPL planners to evaluate their effects on recognition performance while implementing *online goal recognition with no recomputation of the ideal plan* (Alg. 3, Section 3.2). KPIECE1 is a tree-based geometric planner that uses multiple levels to guide a frontier-based exploration of the continuous state space [23]. The other three planners are from the RRT (Rapidly-exploring Random Trees) family [17]. The planners differ in their optimality guarantees: RRT* is an optimized planner that 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. RRTCONNECT provides no optimality guarantees.

Figure 6(a), contrasts the results of the four planner based navigation goal recognizers in terms of the performance criteria discussed. Each of the columns shows the mean recognition results over the same set of 110 recognition problems with higher columns denoting improved results. We see that TRRT and RRT* are clearly and significantly better for *online goal recognition* of paths generated by RRT* than RRT-Connect 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. 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. Figure 6(b) shows the mean running times of the planners over the entire set of 110 recognition problems (i.e., a higher value is worse). The results are shown as percentage of the total time made available to the planners. Thus RRT* takes (by design) 100% of the time allotted, but others do not. Indeed, we see that TRRT is the second quickest, and is beat only by RRTConnect.



(a) Recognition performance



(b) Runtime (sec).

**Figure 6: Comparison of planner performance.**

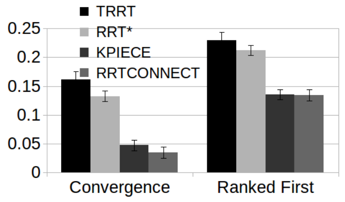
We conclude that the recognition results greatly improve with the optimality of the planner utilized in the goal mirroring process. This is due to the fact that we used an optimally converging planner (RRT*) to generate the observations providing tight dependence between the planner used to generate 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.

## 4.4 Sensitivity to recognition difficulty

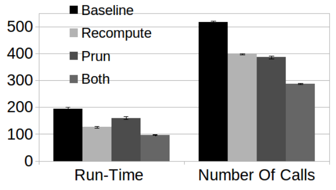
Finally, we also wanted to evaluate the sensitivity of the results shown above to the hardness of the recognition problems. We therefore added another 9 goal points to the recognition problems in the navigation domain (e.g., 19 potential goals in each recognition problem, Figure 2(b)). This means a total of 380 recognition problems. 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.

Figure 7(a) compares the performance (%) of all planners 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 7(b) compares the efficiency of the different online approaches over the *harder* clustered goals problems. These results are consistent with the results from the original scenario in that the *Baseline* approach is still the least efficient, having a higher run-time and larger number of calls to the planner, than the rest. We also see that the most efficient approach is still the approach of utilizing both the *Pruning* heuristic and the *Recompute* heuristic together. In run-time the *Recompute* heuristic is still more efficient than the *Pruning* however for the measure of number of calls made to the planner we see that, for more clustered goals scenarios, the *Pruning* heuristic slightly outperforms the *Recompute* heuristic.



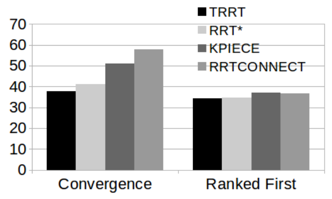
(a) Clustered goals planner performance (Higher values better)



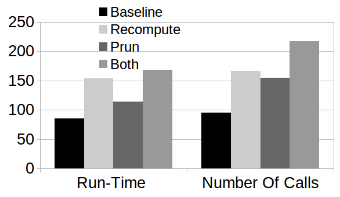
(b) Clustered goals heuristic efficiency (Lower values better)

**Figure 7: Clustered goals performance and efficiency.**

Figure 8(a) compares the deterioration (%) of performance while examining the effect of planner choice. We see the deterioration (%) in each of the performance criteria, when running the recognizers using the different planners on the 380 harder problems. A lower result here is better, indicating less deterioration in performance. For the *ranked-first* measures all planners deteriorated equally. However, TRRT proved more robust than others in the *convergence* measure.



(a) Performance deterioration



(b) Efficiency deterioration

**Figure 8: Deterioration due to hardness of problem (lower values are better).**

Figure 8(b) compares the efficiency deterioration (%) of the improvements of the different heuristics both on run-time and overall number of calls to the planner. Again we used the TRRT planner on the above mentioned 380 hard problems. The results are measured in terms of percent of deterioration, hence a 100% deterioration in run-time means the planner took twice as long on average, on the harder problem. We can clearly see that the least deterioration, both in run-time and number of calls to the planner occurred for the *Baseline* approach of not recomputing the ideal plans. The worst deterioration in terms of run-time occurred for the *Recomputation* heuristic, with a deterioration of 153.5%. This also considerably affected the deterioration of the *Both* approach which combines both heuristics. The *Pruning* heuristic deteriorated considerable less in terms of run-time with only 114% deterioration.

In terms of number of calls made to the planner the worst deterioration occurred for the *Both* approach, with a deterioration of 216.9% while the deterioration for each of the heuristics was considerably less; 155.2% for the *Pruning* heuristic and 153.6% for the *Recomputation* heuristic.

## 5. SUMMARY

We have presented *navigational online goal mirroring*, an efficient navigational online goal recognition approach that does not rely on a plan library, instead uses a planner to generate recognition hypotheses that are continually matched against incremental observations. We identified key decision points which effect run-time and the number of calls to the planner and introduced a generic online 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. 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 the proposed baseline approach and 80% compared to the naive online approach. In terms of convergence and overall first ranking, we see an increase of over 20% in comparison with the baseline approach.

## REFERENCES

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

[2] D. Avrahami-Zilberbrand and G. A. Kaminka. Incorporating observer biases in keyhole plan recognition (efficiently!). In *Proceedings of Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*, Vancouver, British Columbia, 2007.

[3] C. Baker, R. Saxe, and J. B. Tenenbaum. Bayesian models of human action understanding. In *Advances in neural information processing systems*, pages 99–106, 2005.

[4] 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.

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

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

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

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

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

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

[11] S. Keren, A. Gal, and E. Karpas. Goal recognition design for non-optimal agents. pages 3298–3304, 2015.

[12] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

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

[14] L. Liao, D. Fox, and H. Kautz. Hierarchical conditional random fields for gps-based activity recognition. In *Robotics Research: The 11th International Symposium (ISRR)*, Springer Tracts in Advanced Robotics (STAR). Springer Verlag, 2007.

[15] Y. E. Martin, M. D. R. Moreno, D. E. Smith, et al. A fast goal recognition technique based on interaction estimates. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 761–768, 2015.

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

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

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

[19] M. Ramırez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *International Joint Conference on Artificial Intelligence*, 2010.

[20] M. Ramırez and H. Geffner. Goal recognition over POMDPs: Inferring the intention of a pomdp agent. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2009–2014, 2011.

[21] 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.

[22] 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.

[23] 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.

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

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

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

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

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

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