Heuristic Online Goal Recognition in Continuous Domains

Mor Vered
Bar Ilan University, Ramat-Gan, Israel
veredm@cs.biu.ac.il

Gal A. Kaminka
Bar Ilan University, Ramat-Gan, Israel
galk@cs.biu.ac.il

Abstract

Goal recognition is the problem of inferring the goal of an agent, based on its observed actions. An inspiring approach—plan recognition by planning (PRP)—uses off-the-shelf planners to dynamically generate plans for given goals, eliminating the need for the traditional plan library. However, existing PRP formulation is inherently inefficient in online recognition, and cannot be used with motion planners for continuous spaces. In this paper, we utilize a different PRP formulation which allows for online goal recognition, and for application in continuous spaces. We present an online recognition algorithm, where two heuristic decision points may be used to improve run-time significantly over existing work. We specify heuristics for continuous domains, prove guarantees on their use, and empirically evaluate the algorithm over n hundreds of experiments in both a 3D navigational environment and a cooperative robotic team task.

1 Introduction

Goal recognition is the problem of inferring the (unobserved) goal of an agent, based on a sequence of its observed actions [Hong, 2001; Blaylock and Allen, 2006; Baker *et al.*, 2007; Lesh and Etzioni, 1995]. It is a fundamental research problem in artificial intelligence, closely related to plan, activity, and intent recognition [Sukthankar *et al.*, 2014]. The traditional approach to plan recognition is through the use of a plan-library, a pre-calculated library of known plans to achieve known goals [Sukthankar *et al.*, 2014]. Ramirez and Geffner [2010] introduced a seminal recognition approach which avoids the use of a plan library completely. Given a set of goals G, the *Plan Recognition by Planning (PRP)* approach uses off-the-shelf planners in a blackbox fashion, to dynamically generate recognition hypotheses as needed.

The use of PRP in continuous domains, and in an online fashion (i.e., when observations are made incrementally) raises new challenges. The original [2010] formulation, relies on synthesizing two optimal plans for every goal $g \in G$: (i) a plan to reach goal g in a manner compatible with the observations O; and (ii) a plan to reach goal g while (*at least partially*) deviating from O, i.e. complying with $\overline{O}$. The likelihood of each goal is computed from the difference in *costs* of optimal solutions to the two plans. Overall, $2|G|$ planning problems are solved, two for each goal.

However, in *online* recognition the set O is incrementally revealed, and $\overline{O}$ changes with it. Thus two new planning problems are solved with *every new observation*, for a total of $2|G||O|$ calls to the planner instead of $2|G|$. In addition, using an off-the-shelf continuous-space planner to generate a plan that may *partially* go through previous observations, but must not go through all of them, is currently impossible given the state of the art.

We present a general heuristic algorithm for online recognition in continuous domains that solves *at most* $|G|(|O|+1)$ planning problems, and *at best*, $|G|$. The algorithm relies on an alternative formulation, that does not use $\overline{O}$. It has two key decision points where appropriate heuristics reduce the number of calls to the planner: for each new observation, the first decision is whether to generate and solve a new planning problem for each g, or remain with the former calculated plans. In the best case, this may reduce the number of overall calls to the planner to only $|G|$ calls. A second decision is whether to prune unlikely goal candidates, incrementally reducing $|G|$, thus making fewer calls to the planner.

We describe the algorithm in detail, and examine several heuristic variants. Utilizing off-the-shelf continuous-space planners, without any modification, we evaluate the different variants in hundreds of recognition problems, in two continuous-environment tasks: a standard motion planning benchmark, and simulated ROS-enabled robots utilizing goal-recognition for coordination.

2 Related Work

Sukthankar et al. [2014] provide a survey of recent work in goal and plan recognition, most of it assuming a library of plans for recognition of goals. Though successful in many applications, library-based methods are limited to recognizing known plans. Alternative methods have been sought.

Geib [2015], and Sadeghipour et al. [2011] offer methods that utilize the same library for both planning and plan-recognition. Hong [2001] presents an online method, with no use of a library, but lacking the ranking of the recognized goals. Baker et.al [2005] present a Bayesian framework to calculate goal likelihoods, marginalizing over possible actions. Keren et al. [Keren *et al.*, 2015] investigate ways to ease goal recognition by modifying the domain.

Ramirez and Geffner [2010] proposed the PRP formulation (which plans for g twice: with O and with $\overline{O}$), for offline recognition. We build on their earlier formulation [2009], in which they did not probabilistically rank the hypotheses, as we do here. This allows us to more efficiently compute the likelihood of different goals, given incrementally revealed observations. We embed this formulation in a definition of plan recognition for continuous spaces, which also varies from the original in that the recognizer observes effects, rather than actions.

Other investigations of PRP exist. Sohrabi et. al [2016] also observe effects, though in discrete environments, and have also sought to eliminate planner calls, by using a k-best planner in an offline manner to sample the plans explaining the observations. Ramirez and Geffner [2011] extend the model to include POMDP settings with partially observable states. Martin et. al [2015] and Pereira et al. [2017] refrain from using a planner at all, instead using pre-computed information (cost estimates and landmarks, resp.) to significantly speed up the recognition. These approaches complement ours. Vered and Kaminka [2016] present an online recognizer, which we prove is a special case of the algorithm we present here. We go a significant step beyond by introducing heuristics to significantly improve both run-time and accuracy.

3 Goal Recognition in Continuous Spaces

We begin by giving a general definition of the goal recognition problem in continuous spaces (Section 3.1). We proceed to develop an efficient *online* recognizer, which can utilize heuristics to further improve the efficiency (Section 3.2). We then discuss such heuristics in detail (Section 3.3).

3.1 Problem Formulation

We define R, the *online goal recognition problem in continuous spaces* as a quintuple $R = \langle W, I, G, O, M \rangle$. $W \subseteq \mathbb{R}^n$ is the world in which the observed motion takes place, as defined in standard motion planning [LaValle, 2006]. $I \in W$, the initial pose of the agent. G, a set of goals; each goal $g \in W$, i.e., a point. O, a discrete set of observations, where for all $o \in O$, $o \subset W$, i.e., each observation a specific subset of the work area i.e., a point or trajectory. M, a (potentially infinite) set of *plan trajectories*, each beginning in I, and ending in one of the goal positions $g \in G$. For each goal g, there exists at least one plan $m_g \in M$ that has it as its end point.

Intuitively, given the problem R, a solution to the *goal recognition* problem is a specific goal $v \in G$ that *best matches* the observations O. For each goal g, trajectories m_g (ending with g) are matched against the observations O.

Formally, we seek to determine $v \equiv argmax_{g \in G} Pr(g|O)$. Ramirez and Geffner [2009, Thm. 7] have shown that necessarily, a goal g is a solution to the goal recognition problem *iff* the cost of an *optimal* plan to achieve g (denoted here i_g, for *ideal* plan) is equal to the cost of an *optimal* plan that achieves g, while including *all* the observations (a plan we refer to as m_g). Vered and Kaminka [2016] build on this to establish a ranking over the goals. They define the ratio $score(g) \equiv \frac{cost(i_g)}{cost(m_g)}$, and rank goals higher as $score(g)$ gets closer to 1. They show experimentally that the ratio works well in continuous domains, and thus we use it here (ignoring priors on $Pr(g)$ for simplicity): $Pr(g|O) \equiv \eta score(g)$, where the normalizing constant η is $1/\sum_{g \in G} score(g)$.

The next step is to compute the plans i_g (ideal plan, from initial pose I to goal g) and m_g (an optimal plan that includes the observations). As described in [Vered *et al.*, 2016], computing i_g is a straightforward application of a planner. The synthesis of m_g is a bit more complex, as m_g candidates must minimize the error in matching the observations.

To do this, we take advantage of the opportunity afforded by the equal footing of observations O and plans in M in continuous environments. Each observation is a trajectory or point in continuous space. Each plan is likewise a trajectory in the same space, as plans are modeled by their effects. Thus generating a plan m_g that perfectly matches the observations is done by composing it from two parts:

- A plan prefix, (denoted m_g^-) is built by concatenating all observations in O into a single trajectory.
- A plan suffix (denoted m_g^+) is generated by calling the planner, to generate a trajectory from the last observed 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.

Given a goal g and a sequence of observations O, the planner is called twice: to generate i_g and to generate m_g^+, used to construct m_g. The cost of i_g and m_g is contrasted using a scoring procedure, denoted $match(m_g, i_g)$, which uses the ratio as described above. As i_g does not depend on O, it can be generated once for every goal, while m_g needs to be resynthesized from its component parts as O is incrementally revealed. This establishes the baseline of $(1 + |O|)|G|$ calls to the planner [Vered *et al.*, 2016]. We now generalize this procedure to further improve on this baseline.

3.2 Heuristic Online Recognition Algorithm

We identify two key decision points in the baseline recognition process described above, that can be used to improve its efficiency:

- Recompute plans only if necessary, i.e., if the new observation may change the ranking (captured by a $RECOMPUTE$ function);
- Prune (eliminate) goals which are impossible or extremely unlikely (as they deviate too much from the ideal plan i_g), (captured by the $PRUNE$ function).

A good $RECOMPUTE$ heuristic reduces calls to the planner by avoiding unnecessary computation of m_g^+ for new observations. A good $PRUNE$ heuristic reduces calls to the planner by eliminating goals from being considered for future observations. Using appropriate heuristics in these functions, we can reduce the number of calls made to the planner and consequently overall recognition run-time. This section presents the algorithm. The next section will examine candidate heuristics.

Algorithm 1 begins (lines 3–5) by computing the ideal plan i_g for all goals, once. It also sets i_g as a default plan suffix m_g^+. This suffix guarantees that valid (though not necessarily optimal plans) m_g can be created from m_g^+, even in the extreme case where no computation of m_g^+ is ever done. Then, the main loop begins (line 6), iterating over observations as they are made available. We then reach the first decision: should we recompute the suffix m_g^+ (line 7).

We will begin by giving a general outline. The $RECOMPUTE$ function takes the current winning trajectory m_υ (υ is the current top-ranked goal) and the latest observation o. It matches the observation to m_υ and heuristically determines (see next section) whether o may cause a change in the ranking of the top goal υ. If so, then the suffixes m_g^+ of all goals (lines 8–12) will be recomputed (lines 11–12), unless pruned (lines 9–10) . Otherwise (lines 13–15) the current suffix m_g^+ of all goals will be modified based on o, but without calling the planner.

Algorithm 1 ONLINE GOAL RECOGNITION $(R, planner)$

```
1: ∀g : m_g, m_g^- ← ∅
2: υ ← ∅                                   ▷ the top-ranked goal
3: for all g ∈ G do
4:     i_g ← planner(I, g)
5:     m_g^+ ← i_g                         ▷ default value for plan suffix
6: while new o ∈ O available do
7:     if RECOMPUTE(m_υ, o) then
8:         for all g ∈ G do
9:             if PRUNE(m_g^+, o, g) then
10:                G ← G − {g}
11:            else
12:                m_g^+ ← planner(o, g)
13:    else
14:        for all g ∈ G do
15:            m_g^+ ← m_g^+ ⊖ prefix(o, m_g^+)
16:    for all g ∈ G do
17:        m_g^- ← m_g^- ⊕ o
18:        m_g ← m_g^- ⊕ m_g^+
19:    for all g ∈ G do
20:        Pr(g|O) ← η · score(g)
21:    υ ← argmax_{g∈G} P(g|O)
```

Recomputing $\mathbf{m_g^+}$. Here a straightforward call to the planner is made per the discussion in section 3.1, to generate an optimal trajectory. From the initial point (the *last* point in o, as o might contain more than a single point), to the goal g.

Modifying $\mathbf{m_g^+}$. When no recomputation of the suffix is deemed necessary, o will be added to the prefix m_g^-, and the existing m_g^+ must be updated so that it continues m_g^- and leads towards g. The baseline algorithm calls a planner to do this, but the point of this step is to approximate the planner call so as to avoid its run-time cost. This is done by removing (denoted by $\ominus$) any parts that are *inconsistent* with respect to the observation from the beginning of the old suffix m_g^+ .

The old m_g^+ begins where the *old* m_g^- (without o) ended. The new m_g^+ should ideally begin with the last point of the *new* m_g^- (which is the new observation o), and continue as much as possible with the old m_g^+. Thus a prefix of the old m_g^+, denoted ($prefix(o, m_g^+)$, line 15) is made redundant by o and needs to be removed. If o is directly on m_g^+, then $prefix(o, m_g^+)$ is exactly the trajectory from the beginning point of m_g^+ to o. But in general, we cannot expect o to be directly on m_g^+. We thus define the ending point for the prefix to be $\widetilde{o}$, the geometrically closest point to o on m_g^+.

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

Finally, when the algorithm reaches line 16, a valid suffix m_g^+ is available for all goals in G. For all of them, it then concatenates the latest observation to the prefix m_g^- (line 17), and creates a new plan m_g by concatenating the prefix and suffix (line 18). This means that a new $score(g)$ can be used to estimate $Pr(g|O)$ (lines 19–20), and a (potentially new) top-ranked goal υ to be selected (line 21).

3.3 Recognition Heuristics

Algorithm 1 is a generalization of the algorithm described in [Vered *et al.*, 2016]. By varying the heuristic functions used, we can specialize its behavior to be exactly the same (Thm. 1), or change its behavior in different ways.

Theorem 1. *If $RECOMPUTE = \top$ and $PRUNE = \bot$ then Algorithm 1 will generate exactly the same number of planner calls as the algorithm in [Vered* et al.*, 2016].*

Proof. (*Sketch*) By setting $RECOMPUTE$ to be *always true*, and $PRUNE$ to be always false, initially a single call to the planner will be made to calculate i_g, and then a new call to generate m_g^+ will be made for all goals and every observation (since no calls will be skipped and no goals would be pruned). This is in accordance with the behavior of the algorithm reported in [Vered *et al.*, 2016]. □

Let us now turn to examine how to avoid unnecessary planner calls. An ideal scenario would be for Alg. 1 to never compute a new suffix m_g^+, for any goal. In line 5 of Alg. 1 the initial suffixes m_g^+ are set to the ideal plans, i_g. If $RECOMPUTE$ is *always false*, then no new planner calls will be made and m_g^+ will be incrementally modified (Alg. 1, line 15) to accommodate the observations.

This approach offers significant savings (Thm. 2), and in the best case, when the observations closely match the originally calculated paths, can produce good recognition results. However, realistically, the observations may contain a certain

amount of noise or the observed agent may not be perfectly rational. Moreover, it could be that the observed agent is perfectly rational and there is no noise in the observations and yet the approach will fail. This is due to cases where there are multiple perfectly-rational (optimal) plans, which differ from each other but have the exact same optimal cost. In such cases, it is possible that the planner used by the recognizer will generate an ideal plan i_g which differs from an equivalent—but different—ideal plan m_g carried out by the observed agent.

Theorem 2. *If $RECOMPUTE = \perp$ there will be exactly $|G|$ calls to the planner.*

Proof. Straightforward, omitted for space. □

RECOMPUTE. As we cannot realistically expect the observations to perfectly match the predictions, we need a heuristic that evaluates to *false* when the new observation o does not alter the top ranked goal v (saving a redundant $|G|$ calls to the planner), and evaluates to *true* otherwise.

A suggestion for such a heuristic in continuous domains is to measure the shortest distance $dist(o, m_g)$ between o and all plans m_g. If $dist(o, m_v)$ is shortest we can assume the observed agent is still heading towards the same goal and do not need to re-call the planner, keeping the current rankings.

PRUNE. Finally, we introduce a pruning heuristic for observing rational agents in continuous domains. It is inspired by studies of human estimates of intentionality and intended action [Bonchek-Dokow and Kaminka, 2014]. Such studies have shown a strong bias on part of humans to prefer hypotheses that interpret motions as continuing in straight lines, i.e., without deviations from, or corrections to, the heading of movements. Once a rational agent is moving away or past a goal point g, it is considered an unlikely target.

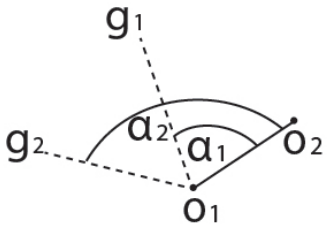

Figure 1: Illustration of goal angles used in pruning heuristic.

To capture this, the $PRUNE$ heuristic takes a geometric approach. We calculate α_g, the angle created between the end-point of the old m_g^-, the newly received observation o and the previously calculated plan m_g. α_g is calculated using the *cosine* formula, $cos(\alpha) = (\vec{u} \cdot \vec{v})/(||\vec{u}||||\vec{v}||)$, where $\vec{u}$ is the vector created by the previous and new observation and $\vec{v}$, the vector created by the previously calculated plan and the new observation.

Figure 1 presents an illustration of the heuristic approach in 2D. For the new observation, o_2, we measure the angle α_i created by the new observation o_2, the previous observation o_1 (which ends m_g^-) and the previous plans $m_{1,2}$ (shown as the dashed lines). If the angle is bigger than a given threshold we deduce that the previous path is heading in the wrong direction and prune the goal. By defining different sized threshold angles we can relax or strengthen the pruning process as needed.

4 Evaluation

We empirically evaluated our online recognition approach and the suggested heuristics 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 and correct ranking of the chosen goal. We additionally implemented our approach on simulated ROS-enabled robots while measuring the efficiency of the algorithm as compared to two separate approaches, one containing full knowledge of the observed agents' intentions and the other containing no knowledge and no reasoning mechanism.

4.1 Online Goal Recognition In A 3D Navigation Domain

We implemented our approach and the proposed heuristics 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 , available as part of the Open Motion Planning Library (OMPL [Şucan *et al.*, 2012]) along with the OMPL *cubicles* environment and default robot. Each call to the planner was given a time limit of 1 sec. The cost measure being the length of the path. For the *Pruning* heuristic we used a threshold angle of $120°$.

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* 110×2 *goal recognition problems*. The observations were obtained by running an 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 optimality. The longer the run-time the more optimal the path. Each problem contained between 20-76 observed points.

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.

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, comparing the different approaches. The results are displayed in Table 1, columns 1–4. *Baseline*, refers to the algorithm of Vered et al. [2016], making $(O+1)|G|$ planner calls. 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, to calculate i_g, the ideal path, for all of the goals. The third approach, *Recompute*, measures the effect of *RECOMPUTE* which aims to reduce overall number of calls to the planner (Section 3.2). The fourth approach, *Prune*, measures the effect of *PRUNE* which aims to reduce the overall number of goals by eliminating unlikely goal candidates (Section 3.2). And the last approach, *Both*, measures the effects of utilizing a combination of both the *Pruning* and *Recompute Heuristics*.

Efficiency Table 1, column 1, displays the average of the results of each approach as the mean of total planner run-time measured in seconds. When only calling the planner once in the recognition process the *No Recompute* approach takes an average of only 7.1 sec. *Baseline* has a time average of 105 sec. The *Pruning* heuristic reduces the average time to only 74.9 sec. And the *Recompute* heuristic further reduces the average time to 49.9 sec. When utilizing both heuristics we achieved a reduction to 36.5 sec. an improvement of a substantial 65.25% from the *Baseline* approach.

The second column displays the average of the results in terms of number of calls made by the recognizer to the planner. The *No Recompute* approach had an average of an extremely efficient 9 calls, i.e. the number of goals. *Baseline* had an average of 265.05 calls while the *Recomputation* and *Pruning* heuristics had similar success with a further 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.

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 60% from the baseline approach. The most efficient method proved to be the *No Recompute* approach, only calculating $|G|$ plans. We will now show that this improvement in efficiency costs considerably in performance.

Performance Table 1, column 3, measures the average convergence to the correct result percent, higher values are better. As we can see with no reuse of the planner at all, *No Recompute* only produces 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 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. When utilizing both heuristics we see that the high convergence level obtained by the *Pruning* heuristic is maintained.

Column 4, measures the percent of times the correct goal was ranked first. 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*, performs poorly with a low 9.5%. *Baseline* 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.

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

4.3 Sensitivity to recognition difficulty

In online, continuous domains, the hardness of the recognition problem could possibly effect recognizer performance and efficiency. We wanted to evaluate the sensitivity of the results shown above to the hardness of the recognition problems. We therefore added another 9 goal points (e.g., 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 10 points, such that navigating towards any one of them appears (to human eyes) to be just as possible as any other.

Table 1, columns 5–6, examines the *efficiency* of the different online recognition approaches over the *harder* clustered goals problems. We omitted the *No Recompute* heuristic in these instances as the behavior of this heuristic is very straightforward. The results are consistent with the results from the original scenario. The *Baseline* approach is the least efficient, having a higher run-time and larger number of calls to the planner, than the rest. The most efficient approach is still the approach of utilizing both the *Pruning* heuristic and the *Recompute* heuristic together. In run-time the *Recompute* heuristic is still more efficient than the *Pruning* however for

	10 goals				19 goals			
	Efficiency		Performance		Efficiency		Performance	
	Run-Time	PlannerCalls	Conv.	Rank.	Run-Time	PlannerCalls	Conv.	Rank.
Baseline	105.02	265.05	21.82	20.24	194.65	516.57	16.37	19.54
Recompute	49.98	148.94	25.44	33.91	126.75	397.85	18.7	22.76
Prune	74.90	151.46	42.16	40.50	160.29	386.53	23.18	24.03
Both	36.49	90.68	42.41	40.21	97.63	287.36	20.98	25.82
No Recompute	7.10	9.00	6.77	9.54	-	-	-	-

Table 1: Comparison of all approaches across scattered and clustered goal scenarios.

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.

Table 1, columns 7–8, examines the *performance* of the different online recognition approaches over the *harder* clustered goals problems. For the harder problems the best performance achieved, in terms of convergence, was by the *Pruning* heuristic with a convergence of 23.18% from the end. In terms of the amount of times the correct goal was ranked first the *Both* approach, combining both *Pruning* and *Recompute* heuristics, only slightly outperformed the *Pruning* approach. The worst performance was achieved by the *Baseline* approach, in terms of both criteria measured; convergence and ranked first, in congruence with the performance results of the scattered goal scenario.

Table 2 measures the deterioration in efficiency and performance with comparison to the scattered goal scenario. The deterioration is measured in terms of deterioration percent, hence a 100% deterioration in run-time means the planner took twice as long on average, on the harder problems. Therefore lower values are better. In terms of efficiency, we can clearly see that the least deterioration, both in run-time and number of calls to the planner, occurred for the *Baseline* approach proving this approach to be very reliable with a deterioration of 85.35% and 94.90% respectively. The biggest deterioration in terms of run-time occurred for the combination of both heuristics with a deterioration of 167.58%. This was considerably caused by the substantial deterioration of the *Recompute* approach which deteriorated by 153.58%. 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, again, 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.

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

4.4 Online Goal Recognition on Robots

As a final set of experiments, and to show the applicability of our approach, we implemented Alg. 1 in a cooperative robotic team task. We used ROS [Quigley *et al.*, 2009] to control simulated robots in Gazebo, using the default ROS motion planner, *move_base*, in the recognition process. We simulated a soccer field, with two robots operating as members of the same team (Figure 2). The observed robot was given an initial goal to travel to, proceeding to execute the plan in a straightforward manner, and the observing robot had to strategically place itself in a pre-chosen position to assist the other robot team member. If the observed robot navigated to *goal 4* the strategic place to assist it on the offense would be to navigate to *goal 3* and vice versa. Likewise also with goals 1 and 2.

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

We compared our online goal recognizer (*OGR*) in its baseline form, to two different approaches: (a) giving full knowledge (*FK*) of the intended goal to the observing robot, ahead of time, allowing the observing robot to navigate directly towards it; and (b) giving no (zero) knowledge (*ZK*) of intended goal, thus forcing the observing robot to wait for its team member to reach its desired goal, before it can navigate towards the complementary location.

To evaluate the different approaches we measured the overall time (in seconds) the simulated robot ran until reaching its target goal. The lower the time the more efficient the robot.The results are displayed in Table 3.

Figure 2: Experiment setup (via RVIZ)

The results show that the goal recognition approach substantially improves on the *zero knowledge* approach, while

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

Table 2: Deterioration of performance and efficiency between scattered and clustered goal scenarios.

		FK	OGR	ZK
I1	**G1**	10.00	17.35	21.50
	G2	5.80	12.31	17.18
	G3	9.10	16.19	20.43
	G4	5.80	17.10	26.03
I2	**G1**	5.80	14.09	17.67
	G2	9.10	19.56	24.45
	G3	10.00	15.89	25.45
	G4	12.35	15.32	32.62
I3	**G1**	12.41	17.36	20.88
	G2	10.10	18.65	24.26
	G3	9.24	10.62	20.66
	G4	5.72	13.40	20.40

Table 3: Online goal recognizer vs. full and zero knowledge

requiring no precalculations; all needed plans are generated on-the-fly via the planner. Understandably our approach falls short of the *full knowledge* approach as it generates hypotheses on the fly following observations, which leads to some deviations from the optimal, direct route.

5 Summary

We presented an efficient, heuristic, online goal recognition approach which utilizes a planner in the recognition process to generate recognition hypotheses. We identified key decision points which effect both overall run-time and the number of calls made to the planner and introduced a generic online goal recognition algorithm along with two heuristics to improve planner performance and efficiency in navigation goal recognition. We evaluated the approach in a challenging navigational goals domain over hundreds of experiments and varying levels of problem complexity. The results demonstrate the power of our proposed heuristics and show that, while powerful by themselves, a combination of them leads to a reduction of a substantial 63% of the calls the recognizer makes to the planner and planner run-time in comparison with previous work. This, while showing an increase of over 20% in recognition measures. We further demonstrated the algorithm in a realistic simulation of a simple robotic team task, and showed that it is capable of recognizing goals using standard robotics motion planners.

References

[Baker *et al.*, 2005] Chris Baker, Rebecca Saxe, and Joshua B Tenenbaum. Bayesian models of human action understanding. In *Advances in neural information processing systems*, pages 99–106, 2005.

[Baker *et al.*, 2007] Chris L Baker, Joshua B Tenenbaum, and Rebecca R Saxe. Goal inference as inverse planning. In *Proceedings of the 29th Annual Meeting of the Cognitive Science Society*, 2007.

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

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

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

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

[Keren *et al.*, 2015] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design for non-optimal agents. pages 3298–3304, 2015.

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

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

[Martin *et al.*, 2015] Yolanda E. Martin, Maria D. R. Moreno, David 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.

[Pereira *et al.*, 2017] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. Landmark-based heuristics for goal recognition. 2017.

[Quigley *et al.*, 2009] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[Ramırez and Geffner, 2009] Miquel Ramırez and Hector Geffner. Plan recognition as planning. In *International Joint Conference on Artifical Intelligence*, pages 1778–1783, 2009.

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

[Ramırez and Geffner, 2011] Miquel Ramırez and Hector 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.

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

[Sohrabi *et al.*, 2016] Shirin Sohrabi, Anton V. Riabov, and Octavian Udrea. Plan recognition as planning revisited. *The Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3258–3264, 2016.

[Şucan *et al.*, 2012] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012.

[Sukthankar *et al.*, 2014] Gita Sukthankar, Robert P. Goldman, Christopher Geib, David V. Pynadath, and Hung Bui, editors. *Plan, Activity, and Intent Recognition*. Morgan Kaufmann, 2014.

[Vered *et al.*, 2016] Mor Vered, Gal A. Kaminka, and Sivan Biham. Online goal recognition through mirroring: Humans and agents. In *Proceedings of the Annual Conference on Advances in Cognitive Systems*, 2016. A slightly modified version appears in Proceedings of the IJCAI 2016 workshop on Human-Agent Interaction Design and Models (HAIDM).