Towards Online Goal Recognition Combining Goal Mirroring and Landmarks

Mor Vered†, Ramon Fraga Pereira‡, Maurício Cecílio Magnaguagno‡,
Gal A. Kaminka†
Felipe Meneguzzi‡
†Bar-Ilan University, Israel
‡Pontifical Catholic University of Rio Grande do Sul, Brazil

ABSTRACT

Online goal recognition is the problem of recognizing the goal of an agent based on an incomplete sequence of incrementally revealed observations as early along the recognition process as possible. Recognizing goals with minimal domain knowledge as an agent executes its plan requires efficient algorithms to sift through a large space of hypotheses. We develop an *online* approach to goal recognition which operates in both continuous and discrete domains using a combination of Goal Mirroring and a generalized notion of landmarks adapted from the planning literature. Extensive experiments demonstrate the approach is more efficient and substantially more accurate than the state-of-the-art.

KEYWORDS

Goal Recognition, Plan Recognition, Goal Mirroring, Landmarks

1 INTRODUCTION

Goal recognition is the problem of recognizing the goal of an agent based on an incomplete sequence of observations. This problem may be further categorized into two subsets; in *offline* goal recognition the set of observations, while in itself may be noisy and/or incomplete, is revealed ahead of time. Conversely, in *online* goal recognition the set of observations is revealed incrementally and an hypothesis must be made after each additional observation with no knowledge about which observation may be the final one.

Goal recognition, be it *offline* or *online*, is a widely researched problem which includes many real-world applications such as human-robot interaction [37], intelligent user interfaces [5, 13], and recognizing navigation goals [16]. Most approaches to goal recognition rely on a plan library describing the plans assumed known by the agent being observed to achieve its goals [34]. While these approaches can be computationally efficient [2, 15], they require substantial domain knowledge, and make strong assumptions about the preferences of observed agents.

A different approach is *plan recognition as planning* (PRP) [18, 28, 32], whereby a planner is used in the recognition process to generate recognition hypotheses as needed, eliminating the need for a plan library. This approach has shown that it is possible to carry out effective goal recognition using only a domain-theory describing actions in the environment as domain knowledge. Another approach that refrains from using plan-libraries is that of Pereira et al. [24] where they additionally refrain from using a planner and only rely on pre-computed information, such as planning landmarks.

However, all of these approaches only apply to *offline* goal recognition. Indeed, Vered and Kaminka [35, 36] have shown that the previously mentioned PRP approaches are not applicable to *online* recognition and must therefore be adapted to include multiple executions of a planning algorithm in order to compute alternative ways in which the observed agent can achieve a goal. Therefore, a straightforward implementation of these PRP methods adapted to *online* recognition may prove very computationally expensive.

In this paper, we develop an *efficient* approach for online goal recognition as planning that generalizes over both discrete (STRIPS style) and continuous (navigation) domains. Our approach achieves substantial runtime efficiency by reducing the complexity of the problems sent to an underlying planning algorithm using an online Goal Mirroring technique [35, 36] and minimizing the number of goal hypotheses to be computed using landmarks computed once during run-time, and a landmark-based heuristic [21, 24]. To generalize over discrete and continuous domains, we adapt the notion of planning landmarks [12] to comprise its original planning semantics as well as continuous spatial domains and develop a new and efficient algorithm to generate spatial landmarks. Since our approach can use any type of PDDL [19] planning algorithm or path planner [33], we can leverage current and future advances in efficiency in such algorithms.

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

2 BACKGROUND AND RELATED WORK

Library-based goal recognition assumes the existence of a library of plans leading to known goals. Such methods include probabilistic inference [3, 6], grammar-based approaches [9, 10, 20, 26, 30], and others (see Sukthankar et al. [34] for a larger enumeration). While often efficient, these methods are limited to recognizing goals for which plans are a-priori known, and encoded in the plan library.

ACM Conference, July 2017, Washington, DC, USA

…$ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Mor Vered†, Ramon Fraga Pereira‡, Maurício Cecílio Magnaguagno‡,
Gal A. Kaminka† and **Felipe Meneguzzi‡**
†Bar-Ilan University, Israel
‡Pontifical Catholic University of Rio Grande do Sul, Brazil

Nonetheless, most of these methods may be used for both online and offline recognition, whereby observations may be incrementally revealed or given a-priori to the recognition process.

Plan recognition based on domain-theories removes the reliance on a plan-library instead using the domain description in the recognition process, assuming that any valid sequence of actions is a possible plan. For example, *plan recognition as planning* (PRP) [18, 27, 28, 32] uses a planner to generate plan hypotheses dynamically, based on the domain-theory and the observations. More efficient offline approaches [21, 24] avoid planning altogether, instead generating planning landmarks from the domain theory prior to recognition. Such landmarks are facts (or actions) that *must* be included in plans that achieve specific goals [12], and thus provide strong evidence for recognizing these goals, and indeed, we use *fact* landmarks here. We note that current implementations of these methods work *offline*.

Some domain-theory methods address online recognition. Early seminal work by Hong [13] uses a *goal graph* representation for online goal recognition, constructed from a domain theory and incoming observations; recognized goals are not probabilistically ranked. In contrast, Baker et al. [4] use a Bayesian framework to calculate goal likelihoods by marginalizing over possible actions and generating state transitions. More recently, Martin et al. [17] heuristically estimated the relevant plan costs needed to compute the likelihood of goals using a plan graph and avoiding calls to a planner. Our work complements these methods.

Vered et al. [35], which also address online recognition, compute exact costs utilizing heuristics in the decision of whether to compute additional values. They present an online algorithm useful in continuous spaces, using off-the-shelf motion planners to estimate goal likelihoods. We generalize on their algorithm to also work in discrete domains, and show how to use landmarks, computed only once at run-time, to heuristically reduce the number of planner calls and improve accuracy. An additional novelty we introduce is the generalization of the notion of landmarks to motion planning in continuous spaces, and the use of such landmarks for online recognition.

3 COMBINING LANDMARKS AND PLANNING

We introduce an online goal recognition approach that combines online *Goal Mirroring* [35] in Section 3.1, and *recognition using landmarks* [21, 24], which we extend to work online and in continuous spaces in Section 3.2. We develop a procedure for extracting continuous-space landmarks in Section 3.3. Finally, we combine the approaches into a single recognition algorithm in Section 3.4.

3.1 Online Goal Recognition Using Planning

We define the goal recognition problem R as a quintuple $\langle W, s_0, G, O, M \rangle$ [35]. The first element W in the quintuple is the set of possible states of the world. In continuous spaces, it is the standard motion planning work area [14], $W \in \mathbb{R}^n$. In discrete domains, W is implicitly defined by specifying a classical-planning domain theory; for brevity, we refer the reader to [21, 23, 28, 32] for formal details of domain theories in plan recognition. $s_0 \in W$ is the observed agent's initial state. G is a set of $k \geq 1$ goals $g_1, \ldots, g_k$, each goal $g_j \subseteq W$, i.e., it is a partial state description; a given goal $g_i \in G$ represents all states in W in which it is considered to be achieved. O is a (possibly infinite) *ordered sequence* of observations $[o_1, \ldots]$, where for each observation $o_x \in O$, $o_x \subset W$. In continuous spaces, each observation consists of one or more points in $\mathbb{R}^n$, e.g., it could also be a trajectory. In discrete domains, each observation is of one or more states, described by a partial state description (e.g., the set of all states where block A is on block B). In online recognition, the sequence is revealed incrementally, and the recognizer is not given the sequence's length—thus it never knows if the latest observation received is the last one to become available.

The task of the recognizer is to provide a probability distribution over G, which denotes the likelihood of goals in G, given the observations O. To do this, the recognizer uses the set M of plan trajectories—a set of plans for pairs $\langle s, g \rangle$, where $s \in W, g \in G$. M can be defined by a plan library (as in most earlier approaches, see [34]), or it could be defined implicitly by using a planner to generate such plans dynamically, e.g., using the domain theory used to define W, or using a continuous-trajectory planner (e.g., a robotics motion planner), which operates in W. In all cases, the available observations O are matched (see below) against the plans in M—plans that achieve a goal g as their final state, and that better match O are considered an indication that g is more likely.

Note that this way of looking at goal recognition abstracts away (intentionally) from the solution method. It does not say anything about the matching process, nor how the candidate plans within M are obtained. It does, however, make one important commitment, to observations being states (single or multiple), rather than actions (state transitions). In this, we follow [32, 35]. Observations may be (and often are) *partial* (i.e., some states that the observed agent has gone through are not a part of O), incomplete (i.e., the final goal $g \in G$ is not observed). While some earlier work have also addressed noisy observations (observed states that are different than what should have been observed with no noise), we do not explicitly deal with noisy observation sequences in this paper (though the experiments include many noisy observations sequences).

We now turn to the issue of determining which plan candidate in M best matches the observations. *Online goal mirroring* [35] uses the following procedure: For each goal $g_k \in G$ in a problem R, the procedure compares the costs of two plans: an *ideal plan* denoted i_k, and an *observation-matching plan*, denoted m_k. Ramirez and Geffner [27, Theorem 7] show that necessarily, a goal g_k for which the two plans, m_k and i_k , have equal costs is a solution to the goal recognition problem. We use this to rank the goals. The closer two costs for i_k and m_k are, the higher the likelihood of g_k.

The ideal plan i_k is an optimal plan[1], computed once from the initial state s_0 to each goal $g_k \in G$. The observation-matching plan m_k is constructed for each new observation such that it always visits the states included in all the observations thus far, and then optimally reaches the goal. For online operation, instead of calling the planner to re-generate the plan m_k with each new observation, we construct m_k for each new observation by concatenating two parts. First, the algorithm maintains a plan prefix m^- which is a concatenation of all observations received to date. This is very efficiently done by simply adding the latest observation to the

[1]We assume some cost measure is given, such as the length of the plan trajectory in continuous spaces, or the number of plan steps in discrete environments, etc.

current prefix (which is initially $\emptyset$). As shown in [18] this may be generated only once for all possible goal trajectories. Second, a plan suffix m_k^+ generated by a motion planner, from the last state of the prefix (after incorporating the observations), to the goal state, g_k. Most computation takes place here by calling the planner.

3.2 Online Goal Recognition Using Landmarks

In the planning literature, *fact landmarks* are facts that must be true at some point in every valid plan to achieve a particular goal from an initial state [12]. In other words, fact landmarks (or simply *landmarks*, in this paper) are subsets of W, the set of all possible states of the world. Landmarks are often partially ordered based on the sequence in which they must be achieved. Formally, a fact landmark is a formula (either a conjunctive or disjunctive formula) over a set of facts that must be satisfied (or achieved) at some point along all valid plan executions. Figure 1 shows an example of fact landmarks and their ordering for a Blocks-World example. The root node represents the goal state, whereas leafs are facts of the initial state. Connected boxes represent facts that must be true together, i.e., conjunctive facts. For example, in order to achieve `(on A B)`, the facts immediately preceding it, `(and (holding A) (clear B))` must be true, and so on.

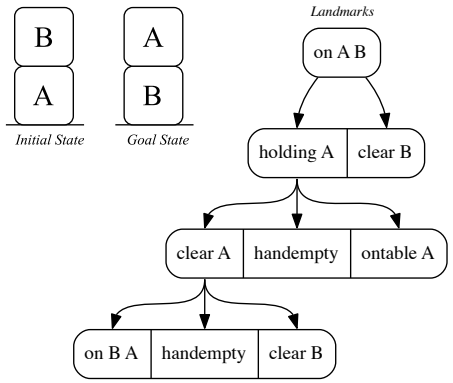

Figure 1: Blocks-World landmarks example.

Given their usefulness for planners and planning heuristics [29], research has yielded multiple notions of landmarks [25], including that of disjunctive landmarks. Briefly, disjunctive landmarks represent an exclusive disjunction over possible instances of variables associated to predicates in the state representation. For example, consider that the agent in the Blocks-World domain can pick-up and hold blocks with both arms. In the two-armed blocks world then, moving a block `A` induces a disjunctive landmark `(or (holding A LeftArm) (holding A RightArm))`. This disjunctive formula defines that the agent can hold the block either with the `LeftArm` or `RightArm`: either one of these facts (but not both simultaneously) must be achieved before moving block `A` to any position, constituting a disjunctive landmark.

Pereira et al. [21, 24] show that it is possible to carry out offline plan recognition by reasoning heuristically about landmarks. The key idea is to maintain a list of ordered landmarks associated with each goal, though partial overlaps are allowed. The *goal completion* heuristic from Pereira et al. [24] matches the observations against this list. This heuristic marks a landmark as *achieved* when facts in the observation match a landmark. The heuristic then uses the ratio of the number of landmarks achieved to the total number of landmarks associated with the goal, inducing a ranking of the goals, used as a proxy for estimating $P(g|O)$.

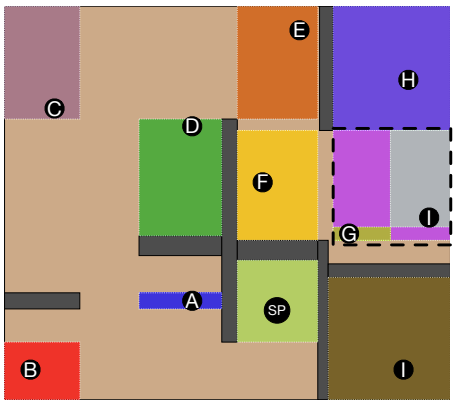

Figure 2: Landmarks for Cubicles environment.

In principle, we can translate the same idea into recognition in continuous domains. In such domains, landmarks can be defined as areas surrounding goals, as illustrated in Figure 2 where black dots represent goals and the surrounding rectangles represent the continuous landmark areas. In this case, to achieve a goal, the observed motion must intersect (go through) the corresponding landmark area. Naturally, we would prefer such areas to be maximal, but must maintain the restriction that landmarks cover only obstacle-free space, and do not intersect completely with other landmarks.

Algorithm 1 Online Goal Recognition With Landmarks.

Require: $L \leftarrow$ ExtractLandmarks(s_0, W, M, G)
Require: $R = \langle s_0, W, G, O, M \rangle$
1: **function** ContOnlineLandmarks(R, L)
2: $achievedFL \leftarrow \emptyset$
3: $activeFl \leftarrow \emptyset$
4: $PruneG \leftarrow \emptyset$
5: **while** new observation $o_i \in O$ is available **do**
6: $activeFL, achievedFL \leftarrow$ AchieveLandmark$(o_i, L, activeFL, achievedFL)$
7: **for all** $g_k \in G$ **do**
8: **if** $l_k \in achievedFL$ **then**
9: $PruneG \leftarrow PruneG + g_k$
10: **else**
11: $PruneG \leftarrow PruneG - g_k$
12: **for all** $g_k \in G \cap PruneG$ **do**
13: $P(g_k|O) \leftarrow$ rank(g_k)

The two characteristic operations between observations and landmarks in continuous environments are: testing whether a landmark has been observed, and differentiating between what constitutes an *active* landmark, and, how to identify a landmark that

has recently been active but no longer fits the last observation (an *achieved* landmark).

We assume Algorithm 1 runs continually to update $P(g_k|O)$, which can then be queried as necessary. Algorithm 1 includes a continuous loop updating the conditional goal probabilities using landmarks and the notions above for online recognition as follows. We begin by pre-computing the landmarks for the problem and providing these (cached landmarks) to the algorithm using the ExtractLandmarks(s_0, W, M, G) function from Algorithm 3.

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

We also need to maintain the currently *active* landmark against which we will compare every additional observation, *activeFl*. In continuous domains the active landmark is the area in which the current observation is found. In discrete domains, it is the set of facts that is currently satisfied. Additionally, we need to maintain *PruneG*, the group of goals that has been pruned out during the recognition process. We maintain this group of pruned goals due to the fact that observations may contain a certain amount of noise that would require backtracking, in which case we need to re-introduce a recently pruned goal back into the goal set, G.

For every incrementally revealed observation, o_i, we check if this observation has caused any landmarks to be activated or achieved via the AchieveLandmark (Algorithm 2) function in Line 6.

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

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

Algorithm 2: Achieve landmark in continuous domains

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

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

3.3 Extracting Landmarks in Continuous Space

We can use any one of a number of landmark extraction algorithms to extract landmarks in discrete environments. Here, we adapt the algorithm of Hoffman et al. in [12] since it efficiently approximates landmark sets that are good enough for the domains we use. This algorithm builds a graph in which nodes represent landmarks and edges represent necessary prerequisites between landmarks, thus representing the landmarks and their ordering. A node in this graph represents a conjunction of facts that must be true simultaneously at some point during the execution of a plan, and the root node is a landmark representing the goal state (Figure 1). Hoffman et al. [12] proves that the process of generating all landmarks and deciding their ordering is PSPACE-complete, which is exactly the same complexity as deciding plan existence [7].

Since the interpretation of landmarks we rely on for plan recognition is that of bottlenecks in the state space, we try to partition a continuous space so that such bottlenecks become identifiable areas in the continuous space. Specifically, to extract landmarks in continuous environments we partition the area using the wall corners as references, to eventually identify pathways between individual "rooms" in the space. Though we define a landmark

Algorithm 3 Landmark Extraction for Continuous Domains.

```
1:  function ExtractLandmarks(s_0, W, M, G)
2:    landmarks ← []                          ▷ Initialize an empty dictionary
3:    for all g ∈ G do
4:      rect ← BoundingBox(g, W)
5:      for all wall ∈ W do
6:        if VisibleFromCentroid(g, wall, W) then
7:          rect ← UpdateBoundingBox(rect, wall)
8:          if rect ∉ landmarks then landmarks[rect] ← ∅
9:          landmarks[rect] ← landmarks[rect] ∪ goal
10:   for all (rect, goals) ∈ landmarks do
11:     if |goals| > 1 then
12:       for all g ∈ goals do
13:         landmarks[MidpointBox(g, goals)] ← g
14:       Remove(landmarks[rect])             ▷ Remove rect index from dictionary
15:     return landmarks
```

Algorithm 4 Goal Mirroring With Landmarks.

Require: $R = \langle s_0, W, G, O, M \rangle$
Require: $L \leftarrow$ ExtractLandmarks(s_0, W, M, G)

```
1:  function OnlineGMwLandmarks(R, L)
2:    achievedFL ← ∅
3:    activeFL ← ∅
4:    for all g_k ∈ G do i_k ← planner(s_0, M, g_k)
5:    while New o_j ∈ O is available do
6:      m⁻ ← m⁻ ∪ o_j
7:      activeFL, achievedFL ← AchieveLandmark(o_j, L, activeFL, achievedFL)
8:      for all g_k ∈ G do
9:        if l_k ∈ achievedFL then
10:         G ← G − g_k
11:       else
12:         m_k^+ ← project(planner(o_j, M, g_k))
13:         m_k ← m⁻ ⊕ m_k+
14:         rank_k ← cost(i_k)/cost(m_k)
15:     for all g_k ∈ G do
16:       P(g_k|O) ← η · rank_k
```

generation algorithm for continuous path planning domains, our approach should work with any notion of numeric landmarks, e.g., recent work on landmarks for hybrid domains [31].

Algorithm 3 extracts continuous landmarks using a world configuration W and the set of goals G, and maps each $g \in G$ to a rectangular area that represents a landmark position[2]. The landmark area for each goal starts as the outermost bounding box in the environment in Line 4. The algorithm iteratively scales it down by generating a horizontal or vertical line limit using the closest visible walls in Line 6. We define visibility as there being no obstacles between the goal and the wall in question and assume that walls correspond to axis-aligned rectangles, though at greater computational cost we could use more sophisticated notions of visibility for any polygonal obstacle through a visibility graph [8, Chapter 5]. If a single landmark area contains more than one goal, we partition this area again based on the midpoint between an arbitrary goal and the remaining ones to obtain new non-overlapping areas for each goal in Line 13, discarding the original area. This partition separates rectangles with multiple goals into a partition analogous to a Voronoi diagram with rectangular areas [1].

Figure 2 illustrates such landmark partition: the black lines represent walls; black dots represent goal candidates; and the different colored rectangles represent landmark areas. We can see the leftmost wall limiting the width of the landmark areas B and C of the two leftmost goals while the center wall limits their height. The dashed area including goals G and I exemplify a partition using the midpoints between these two goals from a square that initially contained both goals. Now that we can compute landmarks for both discrete and continuous domains, we proceed to employ them to perform online goal recognition.

3.4 Goal Mirroring with Landmarks

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

We extensively modify the original *Goal Mirroring* algorithm to use landmark information as a pruning mechanism in Algorithm 4. For simplicity of the algorithm we assume agents do not backtrack and therefore eliminate the need to monitor the last achieved landmark and to maintain a separate set of pruned out goals (as was implemented in Algorithm 1). Like Algorithm 1, we assume a single cached computation of domain specific landmarks for all monitored goals, and the initialization of the previously introduced *achievedFL* and *activeFL* in Lines 2– 3. Additionally, as part of the *Goal Mirroring* algorithm we now calculate the ideal plan, i_k (minimum cost) from the initial position to each possible goal with an additional $|G|$ planner calls (Line 4).

For every newly available observation $o_j \in O$, we update the plan prefix m^- in Line 6 and then proceed to ascertain whether this observation has caused any landmarks to be achieved via the previously introduced AchieveLandmark function. If the observation has caused a landmark to be achieved, *achievedFL* will be updated and we may use the existing fact landmarks to prune unlikely goals in Line 10, in which case we only call the planner to compute plans for those goals whose landmarks have been satisfied in the correct order and have not been passed (Lines 12–14). Note that the plan suffix is a sequence of states, and so we generate a projection of a plan into a sequence of states in Line 12. In the continuous case, this project is straightforward as the plan itself is a sequence of positions in the environment, whereas in the discrete case, this

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

involves executing each action of the plan starting in state o_j and returning the resulting sequence of states.

We use the same ranking procedure as Vered and Kaminka [35] in which the goals are ranked according to the ratio between the initially generated *ideal* plan and the newly generated plan hypothesis, which is comprised of a concatenation of the plan prefix and plan suffix (Lines 13– 14). Finally, the algorithm transforms these rankings into probabilities $P(g_k \mid O)$ via the normalizing factor $\eta = 1/\sum_{g_k \in G} rank(g_k)$ and computes these rankings in Lines 15–16. Mathematically, Algorithm 4 approximates $P(g \mid O)$ for all $g \in G$ using landmarks to rank probabilities, so that, when computing candidate goal probabilities in the bayesian framework of Ramírez and Geffner [2010], we compute $P(O \mid g) = cost(i_k)/(cost(m^-) + cost(m_k+))$. Since $P(g_k \mid O) = \eta \sum_{g_k \in G} P(O \mid g_k)$, our pruning step updates $P(g_k) = 0$ for all ruled-out goals thereby limiting the number of times we need to call the planner.

4 EXPERIMENTS AND EVALUATION

We empirically evaluated our *online* goal recognition approach on both discrete and continuous environments, over hundreds of goal recognition problems while measuring both efficiency and performance.

4.1 Evaluation on Continuous and Discrete Domains

For our continuous environment we used the domain of 3D navigation, where the target is to recognize navigational goals as soon as possible while the observations, i.e., observed agents' positions, are incrementally revealed. We used 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 [33]) along with the OMPL *cubicles* environment and default robot displayed in Figure 3. The yellow polygon representing the robot and the green polygons representing obstacles in the environment. Each call to the planner was given a time limit of 1 sec; and the cost measure being the length of the path. We set 11 points spread through the cubicles environments. We then generated two observed paths from each point to all others, for *a total of* 110×2 *goal recognition problems*. The observations were obtained by running the asymptotically optimal planner RRT* on each pair of points, with a time limit of 5 minutes per run.

For our discrete environments, we used the openly available datasets [22] based on the ones developed by Ramírez and Geffner [27, 28].These datasets comprise 15 domain models with thousands of non-trivial and large goal recognition problems with optimal and sub-optimal plans. We evaluated our approaches using optimal and sub-optimal plans. Each goal recognition problem contains a domain description, initial state, set of candidate goals, a hidden goal, and an observation sequence representing a plan that achieves the hidden goal. For our discrete planner, we used the JavaFF[3], an implementation of Fast-Forward [11] in Java. We ran the experiments for discrete environments with an 8GB memory limit on the JavaVM and a 2-minute time limit.

[3]https://github.com/Optimised/JavaFF

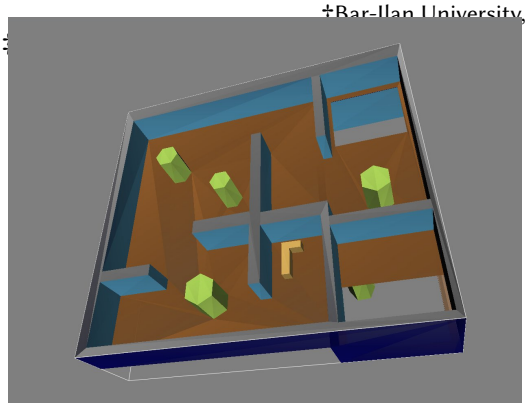

Figure 3: Cubicles environment.

We evaluated our combined approach (Goal Mirroring with Landmarks) both in terms of improvement in efficiency and in terms of overall performance showing that the improvement in efficiency did not come at the expense of performance but rather improved it. We then contrasted the performance with the existing PRP approach (Goal Mirroring) and our newly presented online recognition approach utilizing only the landmarks for ranking and pruning out goals (Online Recognition with Landmarks).

Efficiency Measures. We used two separate measures to evaluate the overall *efficiency* of our approach: the number of planner calls (PC) within the recognition process; and the overall time (Time, in sec.) spent planning. Both these parameters measure the overhead of the *PRP* approach of using the planner and while they are closely linked, they are not wholly dependent. While a reduction in overall number of calls to the planner necessarily results in a reduction in planner run-time, the total amount of time allowed for each planner run may vary according to the difficulty of the planning problem and therefore create considerable differences. Naturally, lower values are better.

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

				Continuous Domains																	
				Goal Mirroring						Goal Mirroring with Landmarks						Online Recognition with Landmarks					
Domain (# problems)	$\|G\|$	$\|O\|$	$\|L\|$	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*	*Time*	*PC*	*TPR*	*FPR*	*RF*	*CV*
Cubicles (220)	11.0	26.5	11.0	104.70	265.0	**100%**	100%	20.2%	21.8%	85.90	184.8	78.2%	61.1%	**24.3%**	**26.2%**	**0.020**	**0**	78.3%	**60.9%**	21.7%	15%

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

Table 1: Experimental results for both continuous and discrete domains.

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

Table 2: Experimental results for discrete domains (large and non-trivial planning problems).

Results. Table 1 shows the experimental results for both continuous and discrete domains across all criteria. For the continuous domain, the combined Goal Mirroring with Landmarks approach achieved the best performance with an improvement both in convergence and the amount of times the recognizer ranked the correct goal hypothesis first. It proved just as reliable as Online Recognition with Landmarks in terms of *TFR* and *FPR*, however not as reliable as Goal Mirroring, which does not prune out goals at all, incurring no risk of overlooking the correct goal.

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

However, there were several instances where the dataset was so complex that both the Goal Mirroring and Goal Mirroring with Landmarks approaches failed. Due to the repeated calls to the planner these approaches timed-out without results. These problems were considerably more complex with a larger number of objects and instantiated actions. The results are summarized in Table 2, where we see the strength of the Online Recognition with Landmarks approach, which does not employ a planner and therefore evades the considerable overhead calculations.

The improvement of run time over the baseline Goal Mirroring approach is presented in Figure 4. For the continuous domain, Goal Mirroring with Landmarks, incorporating landmarks, reduces the run time to 80% while Online Recognition with Landmarks was by far the most efficient with a reduction to only 0.019% of the original Goal Mirroring runtime. For the discrete domain as well we see that Online Recognition with Landmarks more than doubles the reduction in run-time vs. the Online Recognition with Landmarks, which in itself reduces the run-time considerably to between 17%–46%. Figure 5 shows a comparison regarding the amount of times the planner was called within the recognition process for both continuous and discrete domains.

5 CONCLUSIONS

We have developed an efficient online goal recognition approach which works in both continuous and discrete domains using a combination of Goal Mirroring and reasoning over a generalized notion of landmarks. We have shown how to dynamically generate continuous and discrete landmarks and empirically evaluated the efficiency and performance of our approach over hundreds of experiments in both continuous and discrete domains; comparing our results to an existing online goal recognition approach and a

Mor Vered†, Ramon Fraga Pereira‡, Maurício Cecílio Magnaguagno‡,
Gal A. Kaminka† and **Felipe Meneguzzi‡**
†Bar-Ilan University, Israel
‡Pontifical Catholic University of Rio Grande do Sul, Brazil

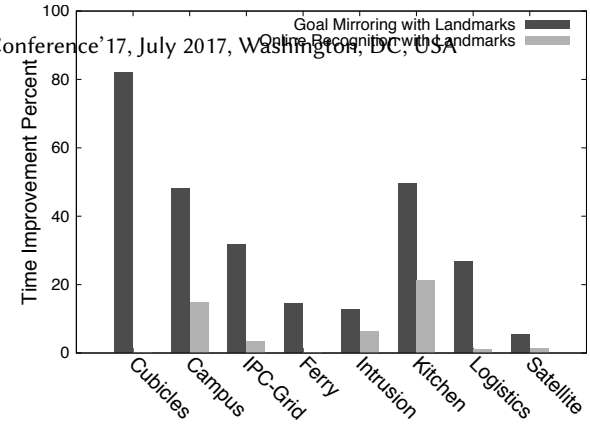

Figure 4: Comparison of runtime improvement.

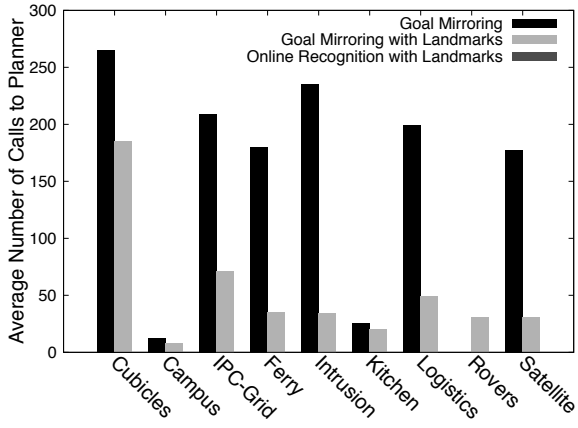

Figure 5: Comparison of average number of calls to planner.

newly defined continuous landmark approach. We have shown that not only is our approach more efficient than the existing online recognizer but also outperforms both other approaches.

However, as our technique continually calls a planner within the recognition process, it might have limitations in recognizing very complex problems, specifically, those for which current planning algorithms are not efficient. Among some of the other limitations is its use of relatively simple landmarks for spatial domains, as well as the assumption that landmarks do not change over the course of the recognition, which would not be realistic for dynamically changing environments. Thus, we believe two important refinements should be the target of future work. First, we aim to refine the notion of spatial landmarks for more informative heuristics, such as the ones developed by Scala et al. [31]. Second, we aim to use techniques to compute landmarks incrementally so as to allow their online recomputation in dynamic domains.

REFERENCES

[1] Franz Aurenhammer. 1991. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.* 23, 3 (1991), 345–405.
[2] Dorit Avrahami-Zilberbrand and Gal A. Kaminka. 2005. Fast and Complete Symbolic Plan Recognition. In *IJCAI-05*.
[3] Dorit Avrahami-Zilberbrand and Gal A. Kaminka. 2007. Incorporating Observer Biases in Keyhole Plan Recognition (Efficiently!). In *AAAI-07*. 944–949.
[4] Chris Baker, Rebecca Saxe, and Joshua B Tenenbaum. 2005. Bayesian models of human action understanding. In *Advances in neural information processing systems*. 99–106.
[5] Nate Blaylock and James F Allen. 2004. Statistical Goal Parameter Recognition.. In *ICAPS*, Vol. 4. 297–304.
[6] Hung Hai Bui. 2003. A general model for online probabilistic plan recognition. In *IJCAI*, Vol. 3. 1309–1315.
[7] Tom Bylander. 1994. The Computational Complexity of Propositional STRIPS Planning. *JAIR* 69 (1994), 165–204.
[8] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. 2005. *Principles of robot motion: theory, algorithms, and implementation.* MIT press.
[9] Christopher Geib. 2015. Lexicalized Reasoning. In *Proceedings of the Third Annual Conference on Advances in Cognitive Systems*.
[10] Christopher W Geib and Robert P Goldman. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence* 173, 11 (2009), 1101–1132.
[11] Jörg Hoffmann and Bernhard Nebel. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *JAIR* 14, 1 (2001), 253–302.
[12] Jörg Hoffmann, Julie Porteous, and Laura Sebastia. 2004. Ordered Landmarks in Planning. *JAIR* 22, 1 (2004), 215–278.
[13] Jun Hong. 2001. Goal Recognition through Goal Graph Analysis. *JAIR* 15 (2001), 1–30.
[14] Steven M. LaValle. 2006. *Planning Algorithms*. Cambridge University Press.
[15] Neal Lesh and Oren Etzioni. 1995. A Sound and Fast Goal Recognizer. In *IJCAI-95*.
[16] Lin Liao, Dieter Fox, and Henry Kautz. 2007. Hierarchical Conditional Random Fields for GPS-based Activity Recognition. In *ISRR*. Springer Verlag.
[17] Yolanda E. Martin, Maria D. R. Moreno, David E Smith, et al. 2015. A Fast Goal Recognition Technique Based on Interaction Estimates. In *IJCAI*. 761–768.
[18] Peta Masters and Sebastian Sardina. 2017. Cost-Based Goal Recognition for Path-Planning. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 750–758.
[19] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. PDDL—The Planning Domain Definition Language. In *AIPS'98*.
[20] Reuth Mirsky and Ya'akov (Kobi) Gal. 2016. SLIM: Semi-Lazy Inference Mechanism for Plan Recognition. In *IJCAI*.
[21] Ramon Fraga Pereira and Felipe Meneguzzi. 2016. Landmark-Based Plan Recognition. In *ECAI*.
[22] Ramon Fraga Pereira and Felipe Meneguzzi. 2017. Goal and Plan Recognition Datasets using Classical Planning Domains. Zenodo. (July 2017). https://doi.org/10.5281/zenodo.825878
[23] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. 2017. Detecting Commitment Abandonment by Monitoring Sub-Optimal Steps during Plan Execution. In *AAMAS*. 1685–1687.
[24] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. 2017. Landmark-Based Heuristics for Goal Recognition. In *AAAI*. AAAI Press.
[25] J. Porteous and S. Cresswell. 2002. Extending Landmarks Analysis to Reason about Resources and Repetition. In *Proceedings of the 21st Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG '02)*.
[26] David V. Pynadath and Michael P. Wellman. 2000. Probabilistic State-Dependent Grammars for Plan Recognition. In *UAI-2000*. 507–514.
[27] Miquel Ramírez and Hector Geffner. 2009. Plan recognition as planning. In *IJCAI*. 1778–1783.
[28] Miquel Ramírez and Hector Geffner. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.
[29] Silvia Richter and Matthias Westphal. 2010. The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks. *JAIR* 39, 1 (2010), 127–177.
[30] Amir Sadeghipour and Stefan Kopp. 2011. Embodied gesture processing: Motor-based integration of perception and action in social artificial agents. *Cognitive Computation* 3, 3 (2011), 419–435.
[31] Enrico Scala, Patrik Haslum, Daniele Magazzeni, and Sylvie Thiébaux. 2017. Landmarks for Numeric Planning Problems.. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 4384–4390.
[32] Shirin Sohrabi, Anton V. Riabov, and Octavian Udrea. 2016. Plan Recognition as Planning Revisited. *IJCAI* (2016), 3258–3264.
[33] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. 2012. The open motion planning library. *IEEE Robotics & Automation Magazine* 19, 4 (2012), 72–82.
[34] Gita Sukthankar, Robert P. Goldman, Christopher Geib, David V. Pynadath, and Hung Bui (Eds.). 2014. *Plan, Activity, and Intent Recognition*. Morgan Kaufmann.
[35] Mor Vered and Gal A Kaminka. 2017. Heuristic Online Goal Recognition in Continuous Domains.. In *IJCAI*.
[36] Mor Vered, Gal A Kaminka, and Sivan Biham. 2016. Online Goal Recognition through Mirroring: Humans and Agents. *The Fourth Annual Conference on*

Advances in Cognitive Systems (2016).

[37] Zhikun Wang, Katharina Mülling, Marc Peter Deisenroth, Heni Ben Amor, David Vogt, Bernhard Schölkopf, and Jan Peters. 2013. Probabilistic movement modeling for intention inference in human–robot interaction. *The International Journal of Robotics Research* 32, 7 (2013), 841–858.