Modular Task and Motion Planning in Belief Space*

Dylan Hadfield-Menell[1], Edward Groshev[2], Rohan Chitnis[2], and Pieter Abbeel[1]

***Abstract*— The execution of long-horizon tasks under uncertainty is a fundamental challenge in robotics. Recent approaches have made headway on these tasks with an integration of task and motion planning. In this paper, we present Interfaced Belief Space Planning (IBSP): a modular approach to task and motion planning in belief space. We use a task-independent interface layer to combine an off-the-shelf classical planner with motion planning and inference. We determinize the problem under the maximum likelihood observation assumption to obtain a deterministic representation where successful plans generate goal-directed observations. We leverage properties of maximum likelihood observation determinizations to obtain a simple representation of (optimistic) belief space dynamics that is well-suited to planning. Our interface is implemented with standard belief state queries, requiring only the ability to sample, compute unnormalized likelihoods, and compute maximum likelihood states. Our contribution is a novel algorithm for task and motion planning in belief space that has minimal dependence on the details of the inference engine used. IBSP can work with a broad class of black box state estimators, with zero changes to the algorithm. We validate our approach in simulated tasks for the PR2 that account for continuous state, different types of initial state distributions, and negative observations.**

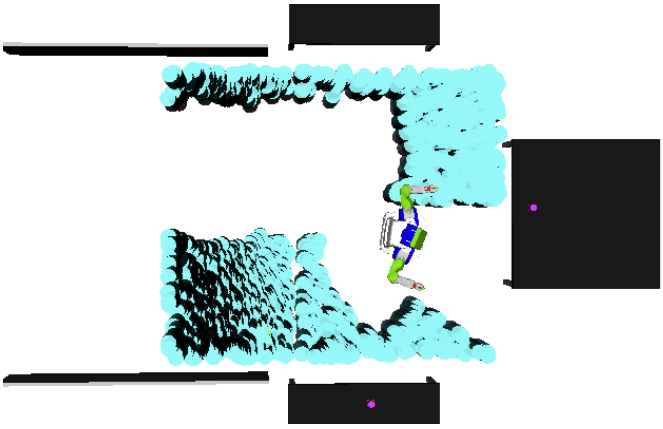

Fig. 1: A screenshot from one of our experiments. The robot is tasked with navigating to the other side of the corridor through a uniform distribution over obstacles. Our low-level refinement algorithm detects likely obstructions and propagates this information to the high level. The objects shown are from the posterior distributions of multiple obstacles after several negative observations.

I. INTRODUCTION

A central goal in robotics is the execution of long-horizon tasks in the face of uncertainty. Readers that have spent frantic mornings searching for lost keys understand the challenges such problems present. Completing such a task corresponds to reasoning about multi-modal belief distributions and obtaining a meaningful observation can require prohibitively long sequences of primitive actions.

A solution to this task is a policy that accounts for uncertainty in locations of objects, uncertainty in robot position, and non-determinism in the dynamics (among other challenges). Solving such problems exactly is far beyond the state of the art in partially observable Markov decision processes (POMDP).

Given the intractability of exact solution, we propose an approximate approach. Our starting point takes inspiration from recent methods for fully observed task and motion planning. This is a challenge in its own right, but careful applications of abstraction, lazy discretization, and motion planning have made inroads [1], [2]. These approaches plan in an *abstract* representation of a problem (without continuous variables) then use sampling and motion planning to *refine* abstract plans into fully grounded plans with continuous parameterizations.

An important development in approximate solutions to POMDPs is the *maximum likelihood observation* (MLO) determinization [3]. This approximation assumes that each belief state produces its maximum likelihood observation. The result is a deterministic problem that encourages goal directed information-gathering behavior. In this work, we extend the domain abstraction techniques from Srivastava et al. [1] to MLO approximations of large continuous POMDPs.

Our primary contribution relies on an abstraction mechanism that compactly represents belief state dynamics for MLO determinizations. As an example, consider a fluent, *BGraspPose*(o, o_pose_bel, r_pose, grasp), whose arguments are, respectively, an object reference, a distribution over object poses, a distribution over robot poses, and a grasp. This fluent is true if the (unobserved) actual poses represent a successful grasp with some preset probability. We show in Section IV-A that directly determinizing this formulation and applying the skolemization approach from [1] leads to unsatisfactory behavior.

Our approach avoids this issue by using properties of MLO determinizations. If *BGraspPose* fails to hold, then either 1) the maximum likelihood state does not represent a successful grasp; or 2) it does, but there is too much uncertainty in the belief to have a high probability of success. The key insight

[1]{dhm, pabbeel}@eecs.berkeley.edu
[2]{eddiegroshev, ronuchit}@berkeley.edu
* This research was funded in part by Intel through its Science and Technology Center on Embedded Computing. Dylan was also supported by a Berkeley Fellowship.

we leverage is that, in a maximum likelihood observation determinization, the correct response to the first scenario is to change the maximum likelihood state, while the correct response to the second is to observe more.

We show how to use this insight to generate useful abstractions in Section IV-B and give a novel algorithm, Interfaced Belief Space Planning (IBSP), that applies a determinization-replan approach to large continuous POMDPs. IBSP extends the interface of [1] and enforces a clear separation among extracting a plan skeleton from a domain description, refining a plan skeleton, and determining success or failure of a plan. Our implementation uses an off-the-shelf classical planner to generate plan skeletons and sampling and trajectory optimization to generate refinements and determine their validity.

An important property of our solution is that it relies on standard belief state queries, such as sampling and maximum likelihood computation. Such structured access to the belief state allows IBSP to work with a broad class of potentially complex state-estimation methods with no algorithmic changes. We validate our approach in a set of simulated partially observable mobile manipulation tasks. A representative task is object retrieval from one of many possible drawers. Long action sequences are required to generate useful observations and correctly accounting for negative observations is necessary for success. Our algorithm reasons about obstructions and occlusions to generate plans to find its goal object, while ensuring safety requirements are met.

II. Related Work

Our work builds on recent results in deterministic task and motion planning. Early work in task and motion planning was embodied in the aSyMov system [4], where a task plan was generated and used to guide motion planning. The focus of this work was on determining a way to solve motion plans in parallel. Dornhege et al. use semantic attachments to do task and motion planning with classical planners [5]. This approach makes use of task planners, but requires symbolic representation of discretized locations. Havur et al. use local search to find an optimal geometric configuration before discretizing the continuous plane into non-uniform cells and using a hybrid planner to find a feasible motion plan [6]. Lagriffoul et al. parameterize symbolic actions with coarse geometric representations of objects and grasp types. During refinement they estimate the feasible region for continuous parameters and use these regions as constraints to reduce geometric backtracking [7].

Maximum likelihood observation determinization was introduced by Platt et al. [3]. They used this approach to frame planning in belief space as an underactuated control problem. They use LQR and transcription methods to find trajectories in belief space and characterize scenarios where a replanning strategy is guaranteed to succeed. Van Den Berg et al. use LQG to solve Gaussian belief space planning problems that incorporate collision avoidance constraints [8]. They are able to plan without the maximum likelihood observation assumption and explore the approximations introduced with maximum likelihood determinizations.

Our determinization-replan approach shares similarity with determinization-replan approaches for probabilistic planning, such as FF-replan [9]. Both leverage determinization in order to obtain major performance gains from classical planners. The maximum likelihood observation assumption is similar to the most likely transition determinization from that literature, while our optimistic belief updates are similar to an all outcomes determinization.

Knowledge space representations for planning are an old idea in artificial intelligence that dates back to McCarthy and Hayes [10]. Bonet and Geffner provide in-depth experimentation and analysis of discrete deterministic partially observed planning [11]. They provide conditions under which discrete formulations of partially observable planning (with binary, factored beliefs) can be compiled to sound and complete representations. Our algorithms are aimed at large continuous problems for robotics while their work focuses on classical planning.

The Belief Space Hierarchical Planning in the Now planning and execution system is a task and motion planning approach to handle uncertainty [12]. They construct task plans in belief space under maximum likelihood observation determinization. Levihn et al. extend this system to construct shorter plans at execution time by smart replanning and reconsideration [13]. They formulate the regression of belief goals under Gaussian distributions and plan with an exact representation of belief state dynamics. In contrast, we use references to the belief so our approach more easily extends to arbitrary belief representations.

Srivastava et al. formulate open world POMDPs with a probabilistic program [14] and develop a generalization of point-based value iteration to that setting. Both methods attempt to solve very large POMDPs. In their setting the challenge is the (unbounded) size of the world and complexity of corresponding beliefs. The challenge in our work stems from uncertainty about continuous quantities.

Gashler et al. use a contingent planner to generate plans that react to feedback from the world [15]. They generate contingent plans and use external function calls during their planning step to generate effects of actions. However, they represent poses and belief updates symbolically for task planning. We use pose and belief references to plan abstractly. We refine and verify plans with the *true* belief.

Nebel et al. use a three valued logic for the TidyUp project that allows fluents to take the value *uncertain* [16]. They assume that uncertain fluents become known once the robot is close enough and so do not explicitly plan sensing actions. In contrast, we represent uncertainty directly and combine reasoning about uncertainty with reasoning about maximum likelihood states.

III. Technical Background

A. Observable planning formulation

We use a STRIPS-style formalism to define our planning problems. We introduce it with a pick-and-place task that

will serve as a running example for the paper.

We define a (fully observed) planning problem, Π as a tuple, $\Pi = \langle \mathcal{E}, \mathcal{F}, I, G, \mathcal{A} \rangle$, with the following definitions:

$\mathcal{E}$: a set of *entities* within our domain; e.g., individual objects or locations.

$\mathcal{F}$: a set of *fluents* that describe relations between entities.

$I \in 2^{\mathcal{F}}$: a conjunction of fluents that are initially true.

$G \in 2^{\mathcal{F}}$: a conjunction of fluents that characterizes the set of goal states.

$\mathcal{A}$: a set of parametrized *operators* that describes ways the agent can alter the world. Each $op \in \mathcal{A}$ is characterized by: *preconditions*, *pre(op)*, a conjunction of fluents that captures the set of states where an operator is applicable; and *effects*, *eff(op)*, which specifies the fluents that become true or false after executing *op*.

A solution to Π is a sequence of operators and states, $\{(op_1, s_1), \ldots, (op_N, s_N)\}$. To be a valid solution, the preconditions for each operator must be satisfied in the state that precedes it. op_1's preconditions must be satisfied in the initial state and the goal fluents must be satisfied in s_N. Subsequent states must add the effects of the intermediate operator to the previous state (i.e., s_{i+1} is obtained by applying op_i's effects to s_i). We illustrate this with a specification of a pick-and-place task:

$\mathcal{E}$: objects (o_i, represented as strings), Null (a nonexistent object), object poses (continuous 6DOF), robot poses (continuous 20DOF), robot trajectories (continuous $20 \cdot T_{max}$ dimensional), and grasps (continuous 6DOF).

$\mathcal{F}$: *Loc*(obj, obj_pose), *RLoc*(r_pose),
Obstructs(obj, traj), *Holding*(obj, grasp),
Connects(r_pose1, r_pose2, traj),
GraspPose(obj, obj_pose, r_pose, grasp).

$\mathcal{A}$
- *MoveTo*(r_pose1, r_pose2, traj)
  pre: *RLoc*(r_pose1) $\wedge$ *Connects*(r_pose1, r_pose2, traj) $\wedge \forall$i $\neg$*Obstructs*(o_i, traj)
  eff: *RLoc*(r_pose2) $\wedge$ $\neg$*Rloc*(r_pose1)
- *Pick*(obj, obj_pose, r_pose, grasp)
  pre: *RLoc*(r_pose) $\wedge$ *Holding*(Null, Null) $\wedge$ *GraspPose*(obj, obj_pose, r_pose, grasp) $\wedge$ *Loc*(obj, obj_pose)
  eff: *Holding*(obj, grasp) $\wedge$ $\neg$*Loc*(obj, obj_pose) $\wedge$ $\neg$*Holding*(Null, Null) $\wedge \forall$t $\neg$*Obstructs*(obj, t)
- *Place*(obj, obj_pose, r_pose, grasp)
  pre: *RLoc*(r_pose) $\wedge$ *Holding*(obj, grasp) $\wedge$ *GraspPose*(obj, obj_pose, r_pose, grasp)
  eff: $\neg$*Holding*(obj, grasp) $\wedge$ *Loc*(obj, obj_pose) $\wedge$ *Holding*(Null, Null) $\wedge \forall$ t s.t. *overlaps*(obj, obj_pose, t): *Obstructs*(obj, t)

B. Abstraction of continuous variables

While this representation unambiguously specifies problems, the presence of continuous parameters and conditional effects that depend on these parameters (e.g., new obstructions from placing an object) prevents direct solution with a classical planner.

To use task planners in this setting, we abstract the continuous aspects of our state in the style of Srivastava et al. [1]. The key step replaces continuous variables by a discrete set of references to useful sets of continuous values. These references are entities in a discrete formulation where the initial state includes fluents that characterize these sets. This process is called *skolemization* and we refer to the introduced entities as *skolem variables*.

For example, in order to derive an abstract representation of the pick operation for object o_i, we need a reference to the pose of the object, the pose of the robot and the grasp it will use to pick it. To abstract the pose of o_i we introduce a new entity, *ObjPose*(o_i), and include the fluent *Loc*(o_i, *ObjPose*(o_i)) to characterize it. Intuitively, the new symbol can be understood as 'the pose that makes *Loc*($o_i, \cdot$) true.' We replace the robot pose and grasp with similar symbols, parameterized by *ObjPose*(o_i), and include a *GraspPose* fluent that links the object identifier to the introduced entities.

To capture the effects of actions that use these parameters, we use *sound* representations: facts which are guaranteed to become true are included, but conditional effects that depend on the particular binding of a skolem variable are not. For example, after executing a place action, *Loc*(o_i, ObjPose(o_i)) will always be true, but the particular obstructions introduced will depend on the exact binding of ObjPose(o_i). We include the former in our description and rely on an interface layer to discover conditional effects (e.g., introduced obstructions) as they become relevant. The process of assigning values to the variables of a high level plan is called *refinement* of the high level plan.

Algorithm 1 shows pseudocode for such an interface layer. Lines 1-5 set up our planning problem. Line 6 constructs a high level plan. Lines 8-10 attempt to find a motion plan for the high level plan. If refinement fails, we identify a conditional effect of the high level plan that may have caused failure, update our description, and replan. This integration is complete for a subclass of planning problems [1].

C. Assumed maximum likelihood observations

In this paper, we consider problems from a more general class than is described in Section III-A—problems with partial observability. To solve partially observable problems with deterministic solvers, we use the *maximum likelihood observation* (MLO) determinization of Platt et al. [3]. Here we give a brief description of their method.

The introduction of partial observability into a planning problem associates each state with a distribution over observations. For example, a state where a robot's depth sensor is pointed at an object might give a noisy measurement of the object's pose. A solution is a policy that maps a *belief state*, a probability distribution over states, to actions.

The central idea behind the MLO is that, if we fix the observation for each belief state, updates to our belief are deterministic. For example, belief dynamics with Gaussian state distributions and Gaussian noise are defined, for each observation, by the Kalman filter update equations. Fixing the observation for each state reduces this to a deterministic

Algorithm 1 An interface for deterministic problems

procedure INTERFACEPLAN(S_0, γ)
  $S \leftarrow S_0$
  $s \leftarrow$ExtractSymbols(S_0)
  // Keep track of the successfully refined plan prefix
  $p_{pre} \leftarrow None$
  $R_{pre} \leftarrow None$
  $p_{post} \leftarrow$ Plan(s, γ)// unrefined plan skeleton
  while not resource limit reached **do**
    $R_{post} \leftarrow$ MotionPlan(s, p_{post})
    $is\_fail, i_{fail}, failure$
      $\leftarrow$ CheckSuccess($S, (p_{post}, R_{post})$)
    if $\neg is\_fail$ **then**
      return $(p_{pre} + p_{post}, R_{pre} + R_{post})$
    end if
    $p_{pre} \leftarrow p_{pre} + p_{post}[: i_{fail}]$
    $R_{pre} \leftarrow R_{pre} + R_{post}[: i_{fail}]$
    $S \leftarrow$ ApplyActions(S_0, p_{pre}, R_{pre})
    $s \leftarrow$ ExtractFailureSymbols($S, failure$)
    $p_{post} \leftarrow$ Plan(s, γ)
    if max attempts reached **then**
      reset pose generators
      reset $S, s, p_{post}, p_{pre}, R_{pre}$
    end if
  end while
end procedure

update. The utility of this approach, compared with simpler approximations like the most-likely-state determinization, is that it can explicitly plan information gathering actions.

IV. INTERFACED BELIEF SPACE PLANNING

In this section, we present our primary contribution: IBSP, a modular approach to task and motion planning in belief space. Section IV-A extends our formulation of the problem to include partial observability. Section IV-B details an approach based on optimistic observations to obtain a useful abstraction of belief space dynamics. Section IV-C describes the IBSP algorithm in detail.

A. Belief space planning formulation

In our full description of the problem, we need to allow for probabilistic representations in belief space. We restrict ourselves to problems where the uncertainty is confined to the continuous state (e.g., we may not know o_1's location, but we know its name and whether we are holding it). The entities for a partially observed version of our pick-and-place domain are distributions over object poses, observations, and the entities described in Section III-A. We replace the fluents from our pick-and-place domain with belief fluents that are true if the corresponding (deterministic) fluent holds with high probability. For example, we have a fluent, *BGraspPose*, which is true if *GraspPose* is true with high enough probability. Our input specifies a distribution over observations that is conditioned on the state.

We divide our operators into *actions*, which alter the true state, and *observations*, which do not alter the true state but garner information about it. Actions are deterministic in belief space (even when the underlying dynamics are not) and follow a similar parametrization to the deterministic case. Observation operators are parameterized by the belief about state variables that determine the observation distribution.

This dichotomy restricts non-determinism to observations, so determinizing observations obtains a representation suitable for task planning. In principle, one can directly determinize the problem in this representation and apply skolemization to get a sound description of abstract belief dynamics. However, such an approach will often fail to generate meaningful high-level plans.

We illustrate this issue with a belief space formulation of the *Pick* operator. To simplify description, we represent objects in the frame of the robot and assume that the robot's pose in its own frame is known.

In the full (intractable) representation, we have the following description:

BPick(o, o_pose_bel, r_pose, grasp)
pre: *BLoc*(o, o_pose_bel) $\wedge$ *RLoc*(r_pose)
  $\wedge$ *BGraspPose*(o, o_pose_bel, r_pose, grasp)
  $\wedge$ *Holding*(Null, Null)
eff: *Holding*(o, grasp) $\wedge$ $\neg$*BLoc*(o, o_pose_bel)
  $\wedge$ $\neg$*Holding*(Null, Null)
  $\wedge$ $\forall$t, h, g $\neg$*BObstructs*(o, t, h, g)
  $\wedge$ $\forall$ o_p, r_p $\neg$*BOccludes*(o, o_p, r_p)

The first precondition captures that the continuous parameter o_pose_bel represents our current belief about the pose of o. The problematic precondition is *BGraspPose*. The issue is that this a conditional effect of observation: it can only become true if properties of the belief state that depend on continuous values are true. The practical implication is that a task plan generated from such a representation would never select an observation action[1]. Formulating a sound logic over general belief states dynamics that enables this reasoning is a challenging task and easily results in cumbersome and intractable representations.

B. Optimism for observations

The insight we use to solve this issue is that belief updates for MLO often exhibit restricted dynamics: they will typically concentrate belief at the maximum likelihood state for a distribution. An action operator, on the other hand, can change the maximum likelihood state for a distribution (and potentially introduce variance). We need a formulation that allows the task planner to effectively select between these options. Thus, our approach reformulates a problem in terms of two types of belief state fluents: those which characterize the maximum likelihood state, and those which characterize the certainty in our belief. The latter of the two shares similarities with the BVLoc(ϵ) fluents from Kaelbling and Lozano-Pérez [12].

[1]In [1] conditional effects only appear as negative preconditions and this allows sound representations to generate meaningful plans.

Returning to the preconditions for *BPick*, there are two reasons *BGraspPose* could fail to hold: either 1) the maximum likelihood state doesn't contain a valid grasp pose or 2) the maximum likelihood state admits at least 1 valid grasp pose, but the distribution has too much variance to allow a successful grasp with high probability. In the first case, we must manipulate the state (e.g., move obstructions), while in the other we must observe it. We replace belief fluents with maximum likelihood fluents and uncertainty fluents:

MLPick(o, o_pose_bel, r_pose, grasp)
pre: *MLLoc*(o, o_pose_bel) ∧ *RLoc*(r_pose)
∧ *MLGraspPose*(o, o_pose_bel, r_pose, grasp)
∧ ¬*UncertainGP*(o_pose_bel, r_pose, grasp)
∧ *Holding*(Null, Null)
eff: *Holding*(o, grasp) ∧ ¬*MLLoc*(o, o_pose_bel)
∧ ¬*Holding*(Null, Null)
∧ ∀t, h, g ¬*MLObstructs*(o, t, h, g)
∧ ∀ o_p, r_p ¬*MLOccludes*(o, o_p, r_p)

In order to achieve ¬*UncertainGP* facts, we include an observation operator:

ObserveGP(r_obs_pose, o, o_pose_bel)
pre: *MLLoc*(o, o_pose_bel) ∧ *RLoc*(r_obs_pose)
∧ ∀o_i¬*MLOccludes*(o_i, o_pose_bel, r_obs_pose)
∧ *MLInView*(r_obs_pose, o_pose_bel)
eff: ∀ r_p, g ¬*UncertainGP*(o_pose_bel, r_p, g)

We call this an optimistic observation because the actual effect is not guaranteed. We rely on the interface layer to discover cases when it does not hold and correct our representation. Note that this assumption of optimism is only made in the high level description. The full algorithm plans with a complete belief representation: if multiple observations are needed to successfully localize an object, the algorithm will plan to take them. In the event that our plan was overly optimistic, BCheckSuccess on line 10 of Algorithm 2 (described in the next section) will detect the overly optimistic behavior and trigger replanning.

C. The IBSP algorithm

In this section, we detail the necessary changes to the interface from Srivastava et al. to handle our belief space formulation. The primary change is that the refinement operation must find an assignment of continuous variables that maximizes the probability of success, instead of simply satisfying constraints. Algorithm 2 shows pseudocode with changes to Algorithm 1 highlighted in red.

We take as input a safety parameter, ϵ. This determines the acceptable probability of failure for each step in the plan. The first change replaces *MotionPlan* with *MLRefine*. Both functions return a dictionary mapping skolem variables to continuous values. The difference is that *MotionPlan* does collision-free motion planning to connect successive states, while *MLRefine* minimizes the probability of collision. *MLRefine* can be defined either by direct trajectory optimization to minimize an unnormalized likelihood of collision, or by sampling objects from the belief state and minimizing the number of samples that are in collision. The implementation for our experiments adopts the second approach.

Algorithm 2 The IBSP planning algorithm

1: **procedure** IBSP(B_0, γ, ϵ)
2: $B \leftarrow B_0$
3: $b \leftarrow$ ExtractSymbols($B_0, None$)
4: $p_{post} \leftarrow$ Plan(b, γ)
5: $p_{pre} \leftarrow None$
6: $R_{pre} \leftarrow None$
7: **while** not resource limit reached **do**
8: $R_{post} \leftarrow$ MLRefine(B, p_{post})
9: $is\_fail, i_{fail}, failure$
10: $\leftarrow$ BCheckSuccess($B, (p_{post}, R_{post}), \epsilon$)
11: **if** $\neg is\_fail$ **then**
12: **return** $(p_{pre} + p_{post}, R_{pre} + R_{post})$
13: **end if**
14: $p_{pre} \leftarrow p_{pre} + p_{post}[: i_{fail}]$
15: $R_{pre} \leftarrow R_{pre} + R[: i_{fail}]$
16: $B \leftarrow$ MLFilter (B, p_{pre}, R_{pre})
17: $b \leftarrow$ ExtractFailureSymbols($B, failure$)
18: $p_{post} \leftarrow$ Plan(b, γ)
19: **if** max attempts reached **then**
20: reset pose generators
21: reset $B, b, p_{post}, p_{pre}, R_{pre}$
22: **end if**
23: **end while**
24: **end procedure**
25: **procedure** IBSP-EXECUTE($B, \gamma, world, \epsilon$)
26: $(p, R) \leftarrow IBSP(B, \gamma, \epsilon)$
27: **for** $i \in len(p)$ **do**
28: $B \leftarrow$ ExecuteAndFilter($p[i], R[i], B, world$)
29: **if** ¬BCheckSuccess($B, (p[i+1:], R[i+1:])$) **then**
30: **return** IBSP-execute($B, \gamma, world, \epsilon$)
31: **end if**
32: **end for**
33: **end procedure**

The second change is the introduction of *BCheckSuccess*, which examines a refinement and checks that preconditions are satisfied. Algorithm 3 shows pseudocode for this method. We define preconditions with a function that maps a state to true or false and indicates whether the given relationship holds. We determine that each of these is satisfied with probability greater than $1 - \epsilon$ by Monte Carlo sampling from the belief state. $N_{samples}$ determines the fidelity of this sampling.

When we detect that a precondition does not hold, we must decide which type of failure information to propagate to the high level. We do this by checking if the predicate holds in the maximum likelihood state (Algorithm 3 Line 9). If it does, then our error is caused by uncertainty in the belief state, and we propagate an uncertainty failure. Otherwise, we return a maximum likelihood fluent as the violated precondition. The exceptions to this rule are predicates about observations (e.g., occlusions). These are deterministic in an MLO approximation, so it does not make sense to propagate

uncertainty failures for them.

Algorithm 3 Determining failure or success of a refinement

```
1:  procedure BCHECKSUCCESS(B, p, R, ε)
2:      // Iterate over the sequence of high level actions
3:      // and their refinements
4:      for p_i, R_i ∈ zip(p, R) do
5:          preds ← ExtractPreconds(p_i, R_i)
6:          W ← Sample(B_i, N_samples)
7:          for pred ∈ preds do
8:              if PercentSucceeds(pred, W) < 1 − ε then
9:                  if ¬pred.eval(B.ML) or IsObs(pred) then
10:                     return (False, i, ¬ML(pred))
11:                 else
12:                     return (False, i, Uncertain(pred))
13:                 end if
14:             end if
15:         end for
16:         // update B with the results of this action
17:         B ← MLFilter(B, p_i, R_i)
18:     end for
19:     return (True, -1, None)
20: end procedure
```

The third change is the addition of *MLFilter*, which updates the belief state based on the maximum likelihood observation that occurred. This is a natural extension of ApplyActions from Algorithm 1 and is implemented with a black-box state estimation method.

An important observation is that these implementations only require us to be able to sample from our belief distribution and compute unnormalized likelihoods. In order to compute the MLO, we need to be able to compute a maximum likelihood observation and maintain a filtered distribution. These are generic operations for belief states and are implemented for many state estimation techniques. Lightweight dependence on the form of the belief distribution allows one to switch state estimators with no change to the algorithm. Thus, IBSP is a belief space planning algorithm that operates with black box access to the belief updates. As a result, our experiments run on several different initial state distributions with no modification to the planning code.

V. EXPERIMENTS

We evaluate IBSP in partially observable pick-and-place and navigation tasks. The robot in our simulation is a full model of the Willow Garage PR2. We provide results for several prior distributions on object locations: uni-modal Gaussian, multi-modal Gaussian, and uniform.

A. Observation Operator Specification and Generators

The maximum likelihood component of the determinized problem essentially follows the dynamics specified in Section III-A. In order to extend this domain to handle partial observability, we add the following fluents and operators:

Fluents:

MLInView(r_pose, o_pose_bel)
MLOccludes(o, o_pose_bel, r_obs_pose)
MLObstructs(o, traj)
UncertainGP(o_pose_bel, r_pose, grasp)
UncertainObstructs(o, traj)

Operators:

- *ObserveGP*(r_obs_pose, o, o_pose_bel)
  pre: *MLLoc*(o, o_pose_bel) ∧ *RLoc*(r_obs_pose)
  ∧ *MLInView*(r_obs_pose, o_pose)
  ∧ $\forall o_i \neg$*MLOccludes*(o_i, o_pose_bel, r_obs_pose)
  eff: ∀ r_p, g ¬*UncertainGP*(o_pose_bel, r_p, g)
- *ObserveTrajectory*(r_obs_pose, traj)
  pre: *RLoc*(r_obs_pose)
  ∧ *MLInView*(r_obs_pose, traj)
  ∧ $\forall o_i \neg$*MLOccludes*(o_i, traj, r_obs_pose)
  eff: $\forall o_i \neg$*UncertainObstructs*(o_i, traj)

The additional skolem functions needed to support this functionality are *ViewPose*(obj) and *ViewPose*(traj) for object and trajectory references. The generator for object view poses draws samples from a 1-meter unit disc around the object, and the head pose is fixed to point at the object's maximum likelihood position. The generator for trajectory view poses returns the first step along the trajectory that has non-negligible probability of being in collision. To observe the uncertain step, a base pose is sampled from regions with negligible collision probability along the same trajectory.

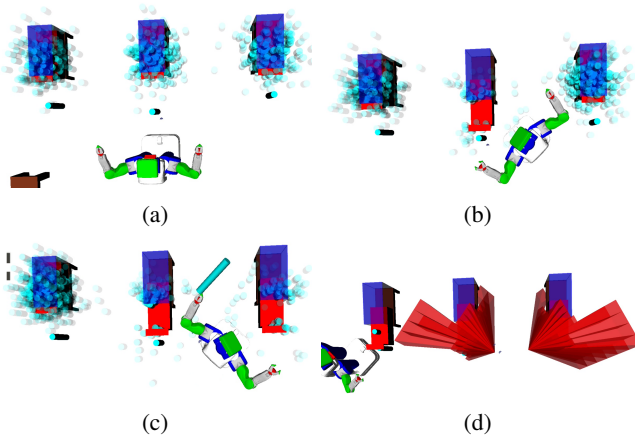

Fig. 2: Intermediate belief states from a sample execution in our drawer search domain. The robot is initialized with a mixture of Gaussians distribution that places the object in one of three drawers. Beliefs are illustrated with samples from the posterior distribution and likelihoods are indicated by the level of transparency. In this run, the object was in the leftmost drawer and was found after searching the other drawers because they had a higher likelihood of containing the object. (a) shows the initial belief. (b) and (c) show intermediate states where IBSP replanned after receiving an observation that indicated its current plan would fail. Notice that we only clear the obstruction in front of the drawer when it prevents opening the drawer far enough to observe the object. (d) shows the view cones from negative observations that were generated during planning.

(a) Initial belief state (b) After 1 observations (c) After 2 observations (d) After 3 observations

Fig. 3: Execution trace of interfaced belief space planning traversing a corridor with uncertain obstacles. The blue cylinders are drawn from a uniform prior distribution. After reaching the other side of the corridor, the robot will search for and pick up two target objects from the three tables.

B. Observation model and state estimation

In our experiments, we use an observation model that accounts for false negatives, negative observations, and occlusion. We model our sensor as a simulated head-mounted Kinect. We use ray-casting with objects fixed to their maximum likelihood locations to build an estimate of the visible region of the MLO for a given pose.

Under our model, we obtain false negatives in relation to the overlap of the object with the visible region. We measure this overlap as the ratio of penetration depth to the object radius. Penetration depth is the minimum translation distance that removes the object from the visible region and object radius is the radius of the smallest sphere that encloses the object. This amounts to observing a false negative with probability $1 - \min\left(1, \frac{\text{penetration depth}}{\text{object radius}}\right)$. If we do not get a false negative, then we observe the object's pose with Gaussian noise.

In our domains, correctly accounting for negative observations is critical: without them it is impossible to guarantee that a trajectory is safe without first localizing all objects it could collide with. To account for this, we use a factored representation of the belief state. For each object we maintain an explicit posterior distribution over its position (e.g., Kalman filter) that summarizes positive observations and a set of view cones where it was not observed that summarizes negative observations. We refer to the first component as the positive observation distribution and the second as the negative regions.

Likelihood queries in this model are relatively straightforward. We implement sampling with probabilistic rejection sampling: we sample from the positive observation distribution and discard the sample with probability proportional to the corresponding observation likelihood. We answer maximum likelihood queries by sampling repeatedly and returning the sample with maximum likelihood.

C. Evaluations

We implement our experiments in Python and use OpenRave [17] to represent the environment. We use trajectory optimization for our motion planning [18] and implement custom collision checking in Flexible Collision Library [19]. We use the Fast-Forward [20] and Fast-Downward [21] task planners to solve our high-level planning problems. Experiments were run in series on Intel Core i7-4770K machines with 16GB RAM. We validate our approach in three distinct scenarios.

To explore collision avoidance in robot navigation tasks under uncertainty, we evaluate performance in a corridor domain. We place the robot at one end of a corridor and present it with the goal of finding and grasping objects on the other end of the corridor. A solution in this domain traverses the corridor observing at regular intervals to ensure collision avoidance. Uniform distributions model the locations of a varying number of obstructing columns throughout the corridor. This task would be essentially insoluble without proper accounting of negative observations.

The second task demonstrates the ability for our system to reason about occlusions. We set a goal of grasping a target object from a cluttered table with other objects on it: some have deterministic locations and others have a Gaussian prior distribution on their location. The target object's location is also drawn from a Gaussian distribution. IBSP plans to remove any occlusions, observe the target object, and then pick it up. In our experiments, 3 of the objects on the table have unobserved locations.

Our final scenario illustrates planning under a multi-modal initial state distribution. The robot has lost its keys and is trying to locate them among three drawers they could be in. We use a mixture of Gaussians as the prior for the target's location. Each drawer that could contain the object is obstructed from being fully opened by an object. IBSP begins by assuming the target is placed at its maximum likelihood location. It discovers that the target is occluded by the drawer containing it and plans to open this drawer. If the drawer obstruction impedes the drawer from opening far enough for the maximum likelihood observation to be useful, this obstruction fact is also discovered and accounted for. During execution, we perform an observation of the interior of the drawer. If the object is observed we continue with the existing plan and achieve our goal. If it is not, our system chooses to replan, as the negative observation indicates the object is in a different drawer.

Table I shows timing results for the three described tasks. Figure 2 and Figure 3 show intermediate steps and belief states for the drawer and corridor tasks. The supplementary video demonstrates plan execution for all three tasks.

Problem Type	% Solved	Avg Time (s)	Avg # Planner Runs
Uni. corridor, 1 obj	100	232	1.88
Uni. corridor, 2 obj	100	445	2.45
Uni. corridor, 3 obj	88	926	2.78
Uni. corridor, 4 obj	58	1168	2
Table, 5 obj	95	89	2.2
Table, 10 obj	95	162	2.21
Table, 15 obj	83	135	2.26
Table, 20 obj	70	229	2.03
Table, 25 obj	68	166	1.84
Table, 30 obj	48	122	1.96
3-drawer search	93	325	2.11

TABLE I: Solution percentages, running times, and number of high-level planner calls for IBSP in occluded search and navigation tasks. 'Uni. corridor N obj' refers to our corridor navigation task with N obstructions. 'Table, N obj' is the cluttered table task with N objects on the table. '3-drawer search' is the multi-modal search problem. Results are averaged across 30 independent runs. Planning runs timeout after 1800s.

VI. Discussion and future work

We presented a novel approach to combining task and motion planning in belief space through the *maximum likelihood observation* determinization principle. Our approach enforces a clear separation among task planning, motion planning, and inference and has the ability to integrate with complex state estimation methods (which are too expensive to include in the inner loop of a planning algorithm). We evaluated our algorithm on several domains, where it completed challenging sequences of perception and geometric reasoning by task planning in an abstract representation of the problem.

The primary limitation we encountered in our experiments stems from computational complexity in rejection sampling. This can involve a large number of intersection queries and in environments with many geometric constraints this can slow planning down considerably.

Another issue arises from belief distributions with many modes. This is a challenge for MLO approximations in general but can be exacerbated by our system. As an extreme example, consider a search task where the prior distribution has 100 modes scattered around a large room. Our system will pick one of these to be the maximum likelihood observation and proceed to investigate it. This decision is made with no regard to the cost of investigating that location and so this setting can result in plans that are quite suboptimal. While this is a challenging problem and minimizing execution cost is not our goal, it is clear that a better solution is needed for IBSP to be practical in similar scenarios.

Future work will include experiments on a real PR2, investigations to the performance of this method with a noisy transition model for the environment, and methods to deal with high variance distributions efficiently.

References

[1] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," *International Conference on Robotics and Automation (ICRA)*, 2014.

[2] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *International Conference on Intelligent Robots and Systems (ICRA)*, 2014.

[3] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Robotics: Science and Systems (RSS)*, 2010.

[4] F. Gravot, S. Cambon, and R. Alami, "aSyMov: a planner that deals with intricate symbolic and geometric problems," in *Robotics Research*. Springer, 2005, pp. 100–110.

[5] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *Towards Service Robots for Everyday Environments*. Springer, 2012, pp. 99–115.

[6] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach," *International Conference on Robotics and Automation (ICRA)*, 2014.

[7] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson, "Efficiently combining task and motion planning using geometric constraints," *International Conference on Robotics and Automation (ICRA)*, 2014.

[8] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[9] S. W. Yoon, A. Fern, and R. Givan, "FF-Replan: A baseline for probabilistic planning." in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 7, 2007, pp. 352–359.

[10] J. McCarthy and P. Hayes, *Some philosophical problems from the standpoint of artificial intelligence*. Stanford University USA, 1968.

[11] B. Bonet and H. Geffner, "Planning under partial observability by classical replanning: theory and experiments," in *Proceedings-International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 22, no. 3, 2011, p. 1936.

[12] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, pp. 1194–1227, 2013.

[13] M. Levihn, L. P. Kaelbling, T. Lozano-Perez, and M. Stilman, "Foresight and reconsideration in hierarchical planning and execution," *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[14] S. Srivastava, S. Russell, P. Ruan, and X. Cheng, "First-order open-universe POMDPs," in *Uncertainty in Artificial Intelligence (UAI)*, 2014.

[15] A. Gaschler, R. Petrick, M. Giuliani, M. Rickert, and A. Knoll, "KVP: A knowledge of volumes approach to robot task planning," in *International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 202–208.

[16] B. Nebel, C. Dornhege, and A. Hertle, "How much does a household robot need to know in order to tidy up your home?" in *AAAI Workshop on Intelligent Robotic Systems*, July 2013.

[17] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, July 2008.

[18] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: Science and Systems (RSS)*, vol. 9, no. 1, 2013, pp. 1–10.

[19] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2012.

[20] Jörg Hoffman, "FF: The fast-forward planning system," *AI Magazine*, vol. 22, pp. 57–62, 2001.

[21] M. Helmert, "The fast downward planning system." *Journal of Artificial Intelligence (JAIR)*, vol. 26, pp. 191–246, 2006.