Title:

- Robust Path Planning in GPS-Denied Environments Using the Gaussian Augmented Markov Decision Process

Authors:

- Peter Lommel (Corresponding author)
  3660 Technology Drive
  Minneapolis, MN 55418
  phlommel@alum.mit.edu

- Marc W. McConley
  Draper Laboratory MS 77
  555 Technology Square
  Cambridge, MA 02139-3563
  mmcconley@draper.com

- Nicholas Roy
  MIT CSAIL
  77 Massachusetts Ave, 33-315
  Cambridge, MA
  02139
  nickroy@mit.edu

Abstract:
As the field of autonomous robotics continues to mature, the ability for robots to operate reliably in the presence of noise and incomplete information is becoming increasingly critical. State estimation techniques such as the Kalman filter explicitly model sensor noise and incomplete information by computing a probability distribution over the possible states. The uncertainty of the posterior probability distribution is a measure of the expected error of the state estimate, but the challenge is how to incorporate knowledge of uncertainty into the decision making process. Ignoring the uncertainty that accompanies probabilistic state estimates during planning can lead to plan failures where the estimate is highly uncertain and likely to be wrong.

In this paper we describe a path planning algorithm that chooses actions based on both the mean and covariance of the state estimate, efficiently planning in the space of Kalman filter beliefs. We demonstrate that the planner is robust to potential sensor noise and uncertainty in the robot's pose, achieving significantly better performance in GPS-denied environments compared to shortest-path planners.

Keywords:

- planning, uncertainty, Kalman filtering.

Paper type:

- Regular paper

Robust Path Planning in GPS-Denied Environments Using the Gaussian Augmented Markov Decision Process

Peter Lommel, Marc W. McConley, and Nicholas Roy *Member, IEEE*
phlommel@alum.mit.edu, mmcconley@draper.com, nickroy@mit.edu
MIT CSAIL, 32 Vassar St., Cambridge, MA 02139

***Abstract*— As the field of autonomous robotics continues to mature, the ability for robots to operate reliably in the presence of noise and incomplete information is becoming increasingly critical. State estimation techniques such as the Kalman filter explicitly model sensor noise and incomplete information by computing a probability distribution over the possible states. The uncertainty of the posterior probability distribution is a measure of the expected error of the state estimate, but the challenge is how to incorporate knowledge of uncertainty into the decision making process. Ignoring the uncertainty that accompanies probabilistic state estimates during planning can lead to plan failures where the estimate is highly uncertain and likely to be wrong.**

In this paper we describe a path planning algorithm that chooses actions based on both the mean and covariance of the state estimate, efficiently planning in the space of Kalman filter beliefs. We demonstrate that the planner is robust to potential sensor noise and uncertainty in the robot's pose, achieving significantly better performance in GPS-denied environments compared to shortest-path planners.

***Index Terms*— planning, uncertainty, Kalman filtering.**

I. Introduction

As autonomous robotics continues to mature, the reliability with which robots make decisions is becoming increasingly critical. For example, there exist numerous motion planning algorithms that allow a robot to compute an optimal or approximate plan to reach some goal position but in almost all cases, these algorithms assume a prior map of the world and perfect knowledge of the robot's position within the map [1]. In reality, a mobile agent moving about the world will naturally encounter environments in which a highly accurate position estimate is not readily available. For example, GPS can become unreliable in dense urban environments and is often unavailable indoors. In these environments, the agent's position can usually still be tracked by registering perceptual data (such as range scans or vision data) against a map of the environment, in combination with a model of the agent dynamics. However, an agent's sensing capabilities may be limited by cost, space and weight restrictions, and the map and agent sensors may contain noise and errors; probabilistic state estimation techniques such as the Kalman filter [2] or Monte Carlo techniques [3] can compensate by integrating multiple measurements over time into a probability distribution over positions.

Even when the position estimate is probabilistic, the simplest approach to planning is to ignore the full distribution and assume that the maximum likelihood estimate of the posterior distribution is correct; any resulting errors can be modelled as disturbances in the control loop. Under a set of well-known assumptions [2], [4], the most likely state in the distribution will generally converge to the true position of the agent, allowing the agent to use classical planning techniques as if the position estimate were known perfectly.

This approach is intuitively successful in situations where the uncertainty remains small, but as the quality of the sensor worsens, the probability distribution over possible positions becomes increasingly uncertain, and the probability of an error in the state estimate will grow. The planner will become correspondingly likely to choose an incorrect action, leading to a failure such as a collision or failure to arrive at the goal position. One such example is shown in figure 1. For a robot to act reliably in the real world, robotic planners must be aware of the uncertainty of the state estimate, and incorporate this uncertainty into how it chooses actions.

In this paper, we will address the problem of motion planning for an agent with an uncertain location within a known map, such as from a prior mapping process. In section II we first describe the motion planning problem and show how it can be posed as a Markov decision process (MDP) [5], a planning methodology where actions have noisy or uncertain outcomes. The MDP assumes that the state itself is fully known at all times. In section IV we describe an extension to the MDP known as the augmented Markov decision process (AMDP) [6] and we couple the AMDP with Extended Kalman Filter state estimation to form a motion planner called the Gaussian AMDP (GAMDP). We show how the GAMDP can be used to efficiently compute motion plans that are robust to positional uncertainty.

II. Path Planning

The traditional path planning problem is described by the following: 1) a set of possible states describing the configuration of the agent and its environment (sometimes called configuration space), 2) the locations of any obstacles within the environment (a map), and 3) an initial state and a goal state [1]. We assume that the robot dynamics are sufficiently described by the set of possible collision-free motions between states of the world. The solution to the traditional path planning problem is a sequence of states from the initial state to the goal state which does not pass through any obstacles.

A. The Markov Decision Process

One problem with pure shortest-path motion planners is that the robot assumes that it has perfect control of its position,

This research was supported by C. S. Draper Laboratory and their support is gratefully acknowledged.

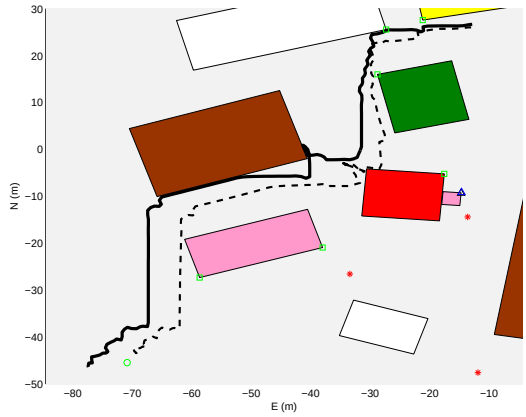


Figure 1. An example failure of a shortest-path MDP policy. The path goes through a region with few or no range measurements, causing the positional uncertainty to grow. Although the mean of the state estimate is approximately at the goal, the state estimate covariance is large, the mean estimate has diverged substantially from the true position and the agent is in fact far from the goal.

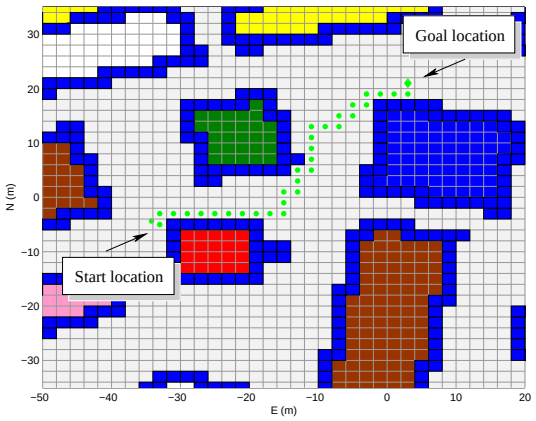


Figure 2. The path shown in this figure is an example of a path chosen by an MDP planner. Here the states are the cells of a grid overlayed on the map, and the available actions are motions between unoccupied adjecent cells. The MDP allows for these actions to be non-deterministic; the policy specifies the optimal action from any cell in the grid. The dotted cells correspond do the most likely path through the grid to the goal according to the policy.

allowing it to move arbitrarily close to obstacles in achieving the shortest path. Such plans can lead to collisions if there is any error in the robot's motion. In order to address this problem, many robots use the Markov decision process (MDP) approach to planning. The MDP planner captures the agent's dynamics using a set of actions. Each action is modeled as a stochastic process, with some randomness in the outcome. The objective of the planner is to find an action sequence with minimum expected cost (or maximum expected reward), where the probability distribution of the cost of a plan is a function of the distribution of outcomes of the actions. For example, if collisions are costly, and a plan contains actions that have high likelihood of collision, then the plan has a high expected cost.

The generic MDP [7] is described by the tuple $\langle \mathbb{S}, \mathbb{A}, T, R \rangle$, where:

- $\mathbb{S}$ is the set of possible states. For path planning, this is typically a discrete set; in this paper the MDP states are the cells of the 2-D grid map.
- $\mathbb{A}$ is the set of possible actions. For path planning in the grid map, this set is typically the actions to move to the 4- or 8-connected neighbors.
- $T : (\mathbb{S} \times \mathbb{A}) \mapsto \Pi(\mathbb{S})$ is the transition function which maps states and actions to distributions over posterior states ($\Pi : \mathbb{S} \mapsto [0, 1]$, and $\sum_{s \in \mathbb{S}} \Pi(s) = 1$). $T(s, a, s')$ is the probability of ending up in state s', given that action a was executed from state s.
- $R : (\mathbb{S} \times \mathbb{A}) \mapsto \mathbb{R}$ is a reward function which maps states and actions to real numbers. The reward state describes the agent's goal (i.e., goal states have a large positive reward), the cost of motions (some small negative reward or cost corresponding to the time taken to move across a grid square), and constraints (i.e., collisions have large negative rewards).

The initial state, s_0, and a discount factor, γ, are sometimes included in the MDP tuple as well[1]. The MDP typically requires a discrete state space, such as a discrete grid imposed over the map. Any motion plan computed by the MDP is therefore restricted to actions that move the agent between neighbouring grid squares. This discretization does introduce an approximation, but the approximation can be made arbitrarily close with appropriate discretization of the grid.

The solution of the MDP is a policy, a mapping from states to actions, and the optimal policy is defined as the policy that maximizes the expected reward of the agent from each state over a horizon, or expected lifetime, of the agent. This expected reward is usally called the value function, $V_\pi(s)$, and is defined over states s for a given policy π, as:

$$V_\pi(s) = E\left[\sum_{i=0}^{n-1} \gamma^i R(s_i, \pi(s_i))\right] \quad (1)$$

where $\gamma \in [0, 1]$ is the discount factor and n is the horizon length. If we assume that once the robot reaches the goal, it stays there permanently, the planning problem can be modelled with an infinite horizon, allowing us to find a single optimal policy and value function $V_{\pi_n}(s) = V_{\pi_{n-1}}(s) = V_{\pi^*}(s)$. We can evaluate the expectation in equation 1 over the transition probabilities, to arrive at a recursive form of the value function known as the Bellman equation [5] which gives a set of $|\mathbb{S}|$ linear equations in $|\mathbb{S}|$ unknowns which can be solved for $V_\pi(s)$ for each s. These equations can be solved in a number of ways, including policy iteration [8] and dynamic programming [5] (value iteration).

An example planning problem is shown in figure 2. Note the discretization of the map into a grid, where obstacles (cells of high cost) are represented as coloured blocks and the white grid cells are free space, with a relatively low cost of motion. The complete solution to the MDP results in a policy: a value $V(s)$ for each grid square, and a corresponding action that achieves that expected value. The dotted squares represent the maximum-likelihood path from the start to the goal according to the policy. If each action resulted in the most likely next state, then the robot would expect to follow that path.

B. The Partially Observable Markov Decision Process

While the MDP allows the planner to represent uncertainty in the outcome of each robot action, it does not in any way

[1]The discount factor is used to reduce the impact of distant rewards and costs compared to immediate costs and rewards, operating on the principle that a dollar today is worth more than a dollar tomorrow.

represent uncertainty in the robot position. In reality, a robot typically only has partial information about its position in the form of noisy or incomplete observations. Just as the MDP models actions probabilistically in order to generate robust motion plans, observations can be modelled probabilistically to maintain an accurate position estimate over time. After each observation is received, a probability distribution or "belief" is updated over the possible states (e.g., agent positions). However, the MDP can still only be applied using a single state estimate calculated from the belief, such as the maximum likelihood estimate or posterior mean estimate. The MDP policy will then choose the action corresponding to only that state, regardless of the relative likelihood of other states.

The partially observable Markov decision process (POMDP) [9] is a model for choosing actions based on the complete state probability distribution, or belief. The POMDP policy is a mapping from beliefs to actions, and similar to the MDP, the optimal policy chooses actions that maximize the expected agent reward over time. Computing the expected reward is substantially harder, since the expectation is not only over the distribution of possible outcomes of each action, but also over the distribution of possible states the agent might have been in. But, by planning in the space of beliefs, the POMDP policy is robust not just to action failures but also to state estimation errors.

The POMDP is described formally by the tuple $\langle \mathbb{S}, \mathbb{A}, T, R, \Omega, O \rangle$, where:

- $\langle \mathbb{S}, \mathbb{A}, T, R \rangle$ are as before
- Ω = the set of possible observations
- $O : (\mathbb{S} \times \mathbb{A} \times \Omega) \mapsto [0, 1]$ is an observation function which maps states, actions and observations to probabilities. $O(s, a, z)$ is the probability of receiving observation $z \in \Omega$ after transitioning, via action $a \in \mathbb{A}$, into state $s \in \mathbb{S}$. $O(s, a, z)$ is also referred to as the emission probability.

Analogous to the MDP, the initial state distribution, $\Pi_0(s)$ and the discounting factor, γ, may be included in the tuple as well. Although the set of actions $\mathbb{A}$ may be identical to the actions in the MDP, moving the robot from state to state, the POMDP for motion planning may include a set of actions that are specific to sensing, such as stopping to take measurements. These actions would have no utility in a motion planning MDP, as the state is always known, but would generate informative observations when the state is not known.

We have added a new set, Ω, and a new function, $O(s', a, z)$, to the tuple. Ω is the set of actual observation values that the agent might receive from its sensors. The observation function, $O(s', a, z)$, is a conditional probability density function (PDF) (or probability mass function (PMF) in the discrete case) over Ω, conditioned on arriving in state s' via action a. We can track the posterior belief b_t at time t recursively using the Bayes' filter equation [3]:

$$b_t(s_t) = p(s_t | z_t, a_t, z_{t-1}, a_{t-1} \ldots) \quad (2)$$

$$= \alpha p(z_t | s_t, a_t) \sum_{s_{t-1}} p(s_t | s_{t-1}, a_t) p(s_{t-1} | z_{t-1}, a_{t-1} \ldots) \quad (3)$$

$$= \alpha O(s_t, a_t, z_t) \sum_{s_{t-1}} T(s_{t-1}, a_t, s_t) b(s_{t-1}). \quad (4)$$

The majority of algorithms for solving POMDPs [8]–[10] involve computing the optimal value function, $V^*(b)$, the expected future (discounted) reward of the policy. The value function is defined over the space of beliefs which is both continuous and high-dimensional. Computing any high-dimensional function is not trivial, and the POMDP value function is generally computationally tractable only for problems with very small action and observation spaces and/or very short horizon length, and the asymptotic complexity is $O(|\mathbb{S}|^2 |\Omega| |\mathbb{A}|^{|\Omega|^{(n-1)}})$ (for horizon length n)[2].

In order to incorporate the probabilistic nature of the state estimate into the path planning algorithm, we need to extend the path planning problem description to the POMDP framework, defining a set of observations and the corresponding emission probabilities from each state and action. However, simply applying the general POMDP solution techniques to motion planning is likely to fail. Let us assume that the robot has a range sensor that is being used to track its position; the range sensor may have a large number of possible values, leading to a large set of possible observations. Additionally, most planning problems require a reasonably long sequence of motions to achieve the goal, which would require a reasonably long horizon length. Since the complexity of solving a POMDP is strongly dependent on both the horizon length and size of the observation space, solving the motion planning problem as a POMDP quickly grows intractable.

For motion planning problems, however, computing a complete value function may be unnecessary. The value function is defined over the complete belief space, that is, we compute a value for every possible probability distribution that can be represented in the POMDP. The space of probability distributions that the POMDP can represent is much larger than the space of probability distributions that the robot will encounter.

A number of approximation methods have been employed to alleviate the complexity of the POMDP. These include planning over a finite set of sampled beliefs [11], applying heuristics to the MDP solution [8], [12], [13] and hierarchical methods (grouping states) [14]. The key to these approximations, as well as ours, lies in the reduction of the $(n-1)$-dimensional multinomial POMDP belief space representation (where n is the number of possible world states). One of the difficulties in reducing the belief space representation is that the set of beliefs must be closed with respect to all possible observation updates. That is, the posterior belief resulting from any observation and any prior belief must also be contained in the set of beliefs. This is commonly refered to as the curse of history [11]. The motivation for our algorithm comes from the existence of such a representation which has been successfully applied to the problem of robot navigation [6]. The Kalman Filter (KF) [2] efficiently maintains an estimate of the robot state in the form of a Gaussian belief.

The KF assumes that the system dynamics and observations can be expressed in the linear form[3]:

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t \quad (5)$$

$$z_t = Hx_t + \nu_t \quad (6)$$

[2]For some special classes of POMDPs, better complexity results can be obtained [10].

[3]The extended Kalman filter (EKF) extends the Kalman filter to the case of non-linear system and observation equations [15]

where x_t is the state at time t, u_t is the action at time t, the observation is z_t and the matrices A, B and H express the linear dynamic and observation models. ϵ and ν are Gaussian white noise processes applied to the linear dynamics and observation models. The initial belief is Gaussian with mean $\hat{x}_0$ and covariance P_0. The belief update after an action u_t is given by

$$\hat{x}_t = A\hat{x}_{t-1} + Bu_t \quad (7)$$
$$P_t = AP_{t-1}A^T + W, \quad (8)$$

and the belief update after an observation z_t is given by

$$K = P_{t^-}H^T\left(HP_{t^-}H^T + V\right)^{-1} \quad (9)$$
$$\hat{x}_{t^+} = \hat{x}_{t^-} + K[z_t - H\hat{x}_{t^-}] \quad (10)$$
$$P_{t^+} = (I - KH)P_{t^-}(I - KH)^T + KVK^T \quad (11)$$

where W and V are the input and measurement noise variances, respectively. The (Gaussian) belief at any time t is fully described by the mean, $\hat{x}_t$, and the covariance matrix, P_t.

If the state of the system can be accurately tracked using a Kalman filter, then the belief space of our POMDP approximation should be restricted to the set of Gaussian beliefs. In fact, if the robot will never experience a non-Gaussian belief, such a planner will actually give an exact solution to the corresponding POMDP. We will only approximate the complete set of Gaussian beliefs by planning over a discretization of the mean and covariance parameters, but this approximation can be made arbitrarily fine. As a result, the wide range of applicability of the Kalman filter implies a wide range of applicability of our algorithm.

III. The Gaussian Augmented Markov Decision Process

Having chosen to work in the space of Gaussian beliefs, we need a planner which maps the Gaussian means and covariances to actions. This is very similar to an existing algorithm called the augmented Markov decision process (AMDP) [6]. The AMDP operates by planning in the space of beliefs that have been "compressed" to the mode of the distribution and its entropy—the uncertainty of the distribution. The AMDP plans over a discretized version of the approximate, "compressed" belief space, effectively computing a policy for a discrete, belief-state MDP. Once the POMDP problem is converted into the AMDP model (computing transition probabilities and rewards in the space of modes and entropies), generating plans for new goals is relatively efficient. The AMDP representation is the simplest that allows the planner to distinguish between state estimates that are certain, and those where the state estimator does not have high confidence. The planner is then able to choose trajectories which not only move the robot towards the goal, but do so with maximum expected value.

One disadvantage to the original AMDP formulation, however, is that converting a POMDP model to an AMDP model is still computationally demanding. A full probability distribution over the entire underlying state space must be constructed for each "compressed" belief to capture the effect and expected reward of each action.

Although POMDP models as described in section II-B are rarely described in terms of linear Gaussian models, the dynamics and observation functions of a Kalman filter describe exactly the transition and observation functions of a very specific, but relatively common, continuous state POMDP. If we can use the Kalman filter process for efficient state estimation during plan execution, then we can use the same process for efficient model construction during the planning process.

We therefore extend the AMDP to use the analytical KF equations for computing the parameters of the AMDP model. The advantage of the KF equations over the Bayesian updates of equation 2 are firstly, that the transitions in belief space resulting from the Bayesian updates are specific to the discrete state space, while the KF equations directly describe transitions of the GAMDP states Secondly, as we will see in section III-.4, under a pair of reasonable assumptions the KF equations describe a deterministic transition in the covariance of the GAMDP state, leading to an overall sparse transition function and fast planning. The disadvantage is a loss of generality; the KF equations only apply to Gaussian distributions.

1) The GAMDP State Space: Each state of the GAMDP is a Gaussian distribution, parameterized by the Kalman filter mean and covariance. Therefore, the complete set of possible states (in a two dimensional world) is:

$$\mathbb{S}_c = \{[x, y, \sigma_x, \sigma_y, \rho_{xy}] | x, y \in \mathbb{R}, \sigma_x \in (0, \infty), \sigma_y \in (0, \infty), \rho_{xy} \in (-1, 1)\} \quad (12)$$

In order to apply MDP algorithms such as value iteration though, we approximate the mean and covariance parameters with discrete sets of values:

$$\Sigma = \{0, \Delta_\sigma, 2\Delta_\sigma, \ldots, \sigma_{max}\} \quad (13)$$

$$\Theta = \{-1, -1 + \Delta_\rho, -1 + 2\Delta_\rho, \ldots, 1\} \quad (14)$$

where σ_{max} characterizes the largest uncertainty that the agent expects to see and Δ_σ and Δ_ρ are small discrete increments. The discrete set of GAMDP states is then $\mathbb{S}' = \mathbb{S} \times \Sigma \times \Sigma \times \Theta$. Although $\mathbb{S}'$ consists of a set of discrete beliefs, each $b \in \mathbb{S}'$ is exactly a well-formed Gaussian distribution.

2) The GAMDP Action Space: The action space for the GAMDP is identical to the action space for the POMDP model described in section II-B. The utility of any purely sensing actions in the POMDP will be preserved, in that these actions affect the entropy of the posterior distribution and therefore still cause the GAMDP state to change.

3) The GAMDP Reward Function: The reward function must be generated to predict the immediate reward for a belief mode and variance. The natural GAMDP reward function for a belief, $b \in \mathbb{S}'$, is the expected immediate reward:

$$R'(b, a) = \sum_{s \in \mathbb{S}} b(s)R(s, a) \quad (15)$$

where $b(s)$ is the probability of the agent being in state $s \in \mathbb{S}$, obtained by numerically integrating the Gaussian probability density function, defined by the state $b \in \mathbb{S}'$, over the grid cell corresponding to s.

4) The GAMDP State Transition Function: The GAMDP state transition function must map states and actions to posterior states (or posterior distributions): $T' : (\mathbb{S}' \times \mathbb{A}') \mapsto \Pi(\mathbb{S}')$. Since the MDP formulation does not include observations, the effect of observations must be captured by the transition function as well. That is, $T'(b, a)$ describes the evolution of the Gaussian distribution b, as a result of temporal dynamic effects conditional on the action a, as well as the expected effect of the observation updates. The KF state and covariance propagation and measurement update equations (5-11), accomplish precisely this. In order to compute the posterior state, $b' = T(s, a)$, using equations (7) through (11), we need to specify three things: the initial conditions, $\hat{x}(0)$ and $P(0)$, the total duration of action a, Δt_a (this determines how far into the future we need to propagate the state and covariance; this is not the same as Δt in equations 5-11), and the input vector $u(\tau), \tau \in [0, \Delta t_a)$. The initial mean and covariance, $\hat{x}(0)$ and $P(0)$ are taken directly from the initial state s. The total time, Δt_a, and the input vector, $u_a(\tau)$ are specified by the action, a.

Since the actual values of the observations which will be received during operation are not available during planning, the states transition non-deterministically according to the likelihood of the observations. Under a pair of assumptions about the uncertainty in the agent dynamics and observations described in the following section, only the posterior mean is non-deterministic; the posterior covariance is independent of the actual observation and is therefore deterministic. By examining each part of the state update independently, we will see that the transition function is very sparse and can be computed efficiently.

IV. Computing Sparse Transitions Efficiently

Firstly, let us assume that the process Jacobians (A and B from equations (7) and (8)) and the measurement Jacobian (H from equations (9) through (11)) are deterministic. That is, A, B and H do not depend on any random variables; in our case this means that they do not depend on the state, x, or the actual observation, z. Under these assumptions, there are no random variables on the RHS of equations (8), (9) or (11), and as a result, these equations describe a deterministic transition of the covariance matrix P.

Unfortunately, the posterior mean, $\hat{x}_t$ does depend on the observation z_t, otherwise we would have an entirely deterministic GAMDP. However, we gain an additional simplification by noticing that the probability of the posterior mean $\hat{x}_{t^+}$ can be computed the prior:

$$p(\hat{x}_{t^+}) = p\left(E_{x_{t^+}|z_t}[x_{t^+}]\right) \quad (16)$$

$$= \int_{z_t} p(z_t) dz_t \text{ such that } E_{x_{t^+}|z_t}[x_{t^+}] = \hat{x}_{t^+} \quad (17)$$

$$\propto \int_{z_t}\int_{x_{t^-}} p(z_t|x_{t^-})p(x_{t^-}) dx_{t^-} dz_t \text{ s. t.} \ldots \quad (18)$$

$$= p(x_{t^-}) \text{ s. t. } E_{x_{t^+}|z_t}[x_{t^+}] = \hat{x}_{t^+}. \quad (19)$$

Equation (17) follows from the fact that the observation is the only random variable in the measurement update, so the probability of the posterior distribution is exactly the probability of the received observation. Equation (19) can be shown trivially by the fact that all distributions $p(x_{t^-})$ and $p(z_t|x_{t^-})$ are Gaussian; this equation would not hold if observations of different environmental features had different sensor models. We therefore see that for our purposes the probability of the posterior mean is just the prior probability, and can be computed efficiently from equations (7) and (8). Figure 3 depicts the effect of our sparse transition function; regardless of the received measurements, the posterior covariance is identical. Although the posterior mean state is non-deterministic, the probabilities are solely a function of the action update (which is fast to compute). As a result, for any action we can compute the complete set of posterior belief states and their respective probabilities using a single EKF update step.

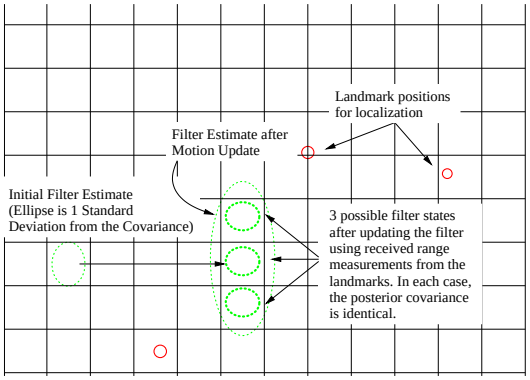


Figure 3. The EKF motion and update process leads to sparse transitions as the posterior covariance is independent of the actual measurements. Note that when uncertainty is low, all posterior mean states may be within the same grid cell leading to an entirely deterministic transition in a discretized grid world.

If we consider the validity of these assumptions, we see that in practice, the process and measurement models for range and bearing localization do not violate the assumption that the Jacobian matrices do not depend on the actual observation values, z. Due to coupled process dynamics and non-linear state dependence of some of the measurements however, the assumption that the Jacobian matrices do not depend on the state may be violated. Notice that this problem is precisely that which can affect the extended Kalman filter (EKF) [15]—the circumstances under which this approximation is problematic is when the dynamics or measurements exhibit very non-linear dependence on the state. For example, if the noise variance is very small compared to the state covariance, then deviation of the measurement from its expected value can cause a large deviation in the posterior state estimate and the approximation of the EKF will be inaccurate. Similarly, if the noise variance is large in comparsion to the state covariance, deviations in the posterior state from its expected value will be small and our approximation will be accurate. Since GAMDP trajectories will generally result in state estimates with small covariances, we can safely assume that the EKF is applicable and our corresponding approximation is reasonable.

A second circumstance under which the assumption of deterministic Jacobian can be violated occurs with line-of-sight measurements (e.g., landmark bearing measurements). In some cases, a small change in the state could change whether or not a measurement is received, and we may think of the difference between receiving and not receiving the measurement as a

Algorithm 1: GAMDP State Transitions
Input: GAMDP state and action space, characteristic velocity uncertainty
Output: GAMDP state transitions
GAMDPTRANSITIONS($\mathbb{S}', \mathbb{A}'$)
(1) **foreach** ($a \in \mathbb{A}'$)
(2) **foreach** ($s \in \mathbb{S}'$)
(3) $u_a(t), \Delta t_a$ = ACTIONSPECS(a)
(4) $\bar{x}_0, P_0$ = INITIALCONDITIONS(s, σ_v)
(5) $\bar{x}, P$ = EKF($u_a(t), \Delta t_a, \bar{x}_0, P_0$)
(6) $T(s, a)$ = ConvertState($\bar{x}, P$)
(7) **return** $T(s, a)$

very non-linear change in the measurement Jacobian matrix. The accuracy of the expected value approximation of equation (10) depends on the amount of information gained from the measurements (i.e. the measurement noise variance, V). If the planner chooses a landmark bearing measurement from a belief state, but, due to discretization error, the landmark is not visible, the expected gain in information may not occur. Figure 4 shows an example of this scenario. The evolution of the covariance matrix is shown for a simulated trajectory in which the position is estimated with a low-grade (automotive [16]) IMU and bearing measurements to landmarks. In this simulation, the agent gets near the goal position and then attempts to reduce the uncertainty in its position, as specified by the GAMDP policy, by taking a landmark bearing measurement. Beginning at approximately 500 seconds, we can see the position uncertainty repeatedly converging as a result of the bearing measurements and then diverging as a result of the velocity and acceleration uncertainty. However, the uncertainty never converges to the point that the planner expects it to (shown in the figure by the solid black line).

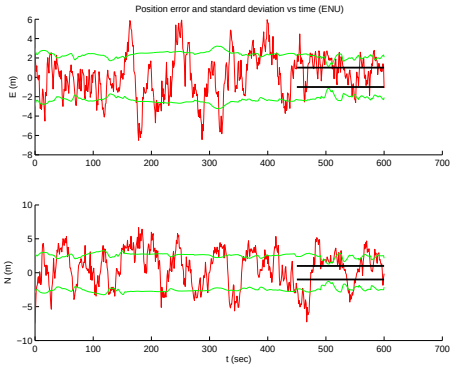


Figure 4. In this figure we can see the effects of the repeated, failed landmark bearing measurements on the position uncertainty—the uncertainty never reaches the point that the planner expects. The standard deviation that the planner expects is shown by the straight solid line beginning at 450 seconds.

This scenario could be avoided by allowing the covariance transitions to be non-deterministic. Fortunately, there appears to be a simpler modification to the algorithm which will prevent this occurrence. If we set a lower bound on the effectiveness of a sensor, by only assuming that a measurement will be received if the environment feature is visible from the entire grid cell for example, we can ensure conservative but reasonable performance from the planner.

An algorithm for computing the GAMDP transition function is given in Algorithm 1.

Algorithm 2: GAMDP Algorithm
Input: a POMDP,γ,ϵ
Output: a GAMDP policy
GAMDP(POMDP)
(1) Compute the GAMDP state space, $\mathbb{S}'$, per section III-.1
(2) Compute the GAMDP action space, $\mathbb{A}'$, per section III-.2
(3) Compute the GAMDP reward function, R', per section III-.3
(4) T'=GAMDPTRANSITIONS($\mathbb{S}', \mathbb{A}'$) (Algorithm 1)
(5) π' = VALUEITERATION($\langle \mathbb{S}', \mathbb{A}', T', R' \rangle$, γ,ϵ)
(6) **return** π'

In Algorithm 1, the function ACTIONSPECS($\cdot$), returns the sequence of control inputs and duration for action a. The function INITIALCONDITIONS($\cdot$) computes the mean and covariance to be used in the EKF equations from the initial GAMDP state and the characteristic velocity uncertainty. The function EKF($\cdot$) returns the mean and covariance that result from application of the EKF equations given the initial conditions and control sequence, and the function ConvertState($\cdot$) converts a mean and a covariance matrix into a GAMDP state.

The complete GAMDP algorithm is given in Algorithm 2.

V. EXPERIMENTAL RESULTS

In this section we analyze the results of a set of motion planning problems. First we will statistically compare the GAMDP and MDP motion planners using a set of experiments in which the map and sensor qualities were varied. Then we will examine specific scenarios which showcase the differences between the MDP and the GAMDP.

Experimental Conditions

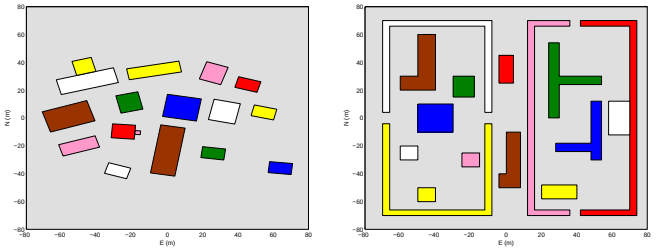
Our agent was simulated using a 6 degree-of-freedom robot motion model, although only 3 degrees of freedom were relevant to the experiments below. The simulator used a second order kinematic motion model ($\ddot{x} = a$) including Coriolis effects due to the non-inertial robot reference frame. The simulator assumed that the agent had a controller such that the closed loop agent response could be modeled by a first order response to a velocity and attitude reference ($a = \frac{1}{\tau_1}(v_r - v)$, $\omega = \frac{1}{\tau_2}(\theta_r - \theta)$), that is, the control signal was a reference velocity and attitude.

The agent in our simulations was equipped with the following sensors:

- A strap-down 3-axis IMU (accelerometer and gyroscope).
- Range measurements from active RF beacons at known locations within the environment.
- Bearing measurements from known landmarks within the field-of-view; perfect data association is assumed. The measured bearing was the angle between body x-axis and the landmark.

The noise for each sensor was assumed to be Gaussian, and we varied the noise for each sensor to test the robustness of the planners to errors in the state (position) estimate. An extended Kalman filter was used to estimate the 6 degrees-of-freedom of the agent using the dynamics and measurement models.

State space: The planner assumed an *a priori* map describing the location of obstacles (e.g., buildings), landmarks (e.g., distinctive building features such as windows, corners, etc.) and hazards. Hazards were states that resulted in large reward penalties, and were modelled in two ways. Some hazards ("point-hazards") were equivalent to point obstacles below the usual sensor field-of-view, for example curbs, potholes, etc., and the penalty was incurred only if the agent is in the same state as the hazard. The agent incurred a reward of -10000 for occupying the same grid cell as a point-hazard. The second kind of hazard ("visibility-hazard") was modeled by a reward of $-1000e^{\frac{r}{25}}$, where r is the range to the hazard, whenever the agent was visible by the hazard. The agent also incurred a cost of -1 for occupying an empty grid cell and a reward of 10000 for occupying the same grid cell as the goal location.



(a) Warehouse Site (b) Two Fortresses

Figure 5. The simulations were performed on each of these maps. (a) Warehouse site map, (b) Two Fortresses map.

Action space: This map was discretized into an occupancy grid [17] with resolution $2m \times 2m$ per grid cell. There were eight actions available to the planners; one each for moving one grid cell North, East, South or West, and one each for taking a landmark bearing measurement with a field-of-view of $\pi/2$ radians ($90°$) facing North, East, South or West. The bearing measurement action did not cause the agent to translate, but may have required the agent to rotate.

After each EKF update, the mean estimate of the (x, y) position given by the EKF was projected into the grid map to identify the current discrete state, and the planner returned its estimate of the optimal action. If the action involved a motion of the agent to a neighbouring grid square (as distinct from a bearing measurement), the control reference velocity and attitude were calculated in order to move the agent from the current mean state estimate to the center of the selected neighbouring grid square. If the action required a rotation to take a bearing measurement, the control reference velocity and attitude were calculated in order to rotate the robot from the current mean orientation estimate to the desired heading. For either action, if the mean state estimate was wrong, the chosen action (and resulting control output) would not actually move the robot in the desired manner; a planner that recognizes and compensates for this possibility is the goal of this research.

A. Quantitative Analysis

A set of simulations was conducted to compare the effect of state uncertainty on the MDP and GAMDP planners. The simulations were broken into six different conditions. These subsets varied using the two different maps shown in figure 5, and using the three different sensor qualities, (navigation, tactical and automotive [16]) described in table I. By varying the sensor noise level, we were able to vary the amount of uncertainty that the agent experienced, allowing us to test the extent to which the GAMDP is more robust to the uncertainty than the MDP.

Table I

THE STANDARD DEVIATION OF THE MEASUREMENT NOISE IS GIVEN FOR EACH SENSOR, FOR EACH QUALITY LEVEL.

Quality Level	σ_{acc} ($\frac{\frac{m}{s}}{rthr}$)	σ_{gyro} ($\frac{deg}{rthr}$)	σ_{sn}(m)	σ_{lm}(deg)
Navigation	.012	.3	1	3
Tactical	.048	1.02	4	3
Automotive	.09	3.42	8	3

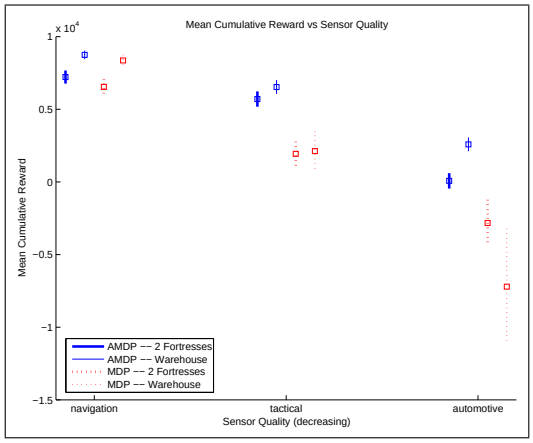


Figure 6. Mean cumulative reward and standard error vs. sensor quality. The GAMDP planner is represented by the solid blue line and the MDP planner is represented by the dashed red line.

The performance of each algorithm for each of the sensor noise levels in table I was estimated from an average over 160 trials, generated from eight different randomly-generated hazard scenarios with twenty random motion planning problems in each hazard scenario[4]. The eight hazard scenarios consisted of two scenarios with ten point-hazards, two scenarios with twenty point-hazards, two scenarios with forty point-hazards, and two scenarios with five visibility-hazards.

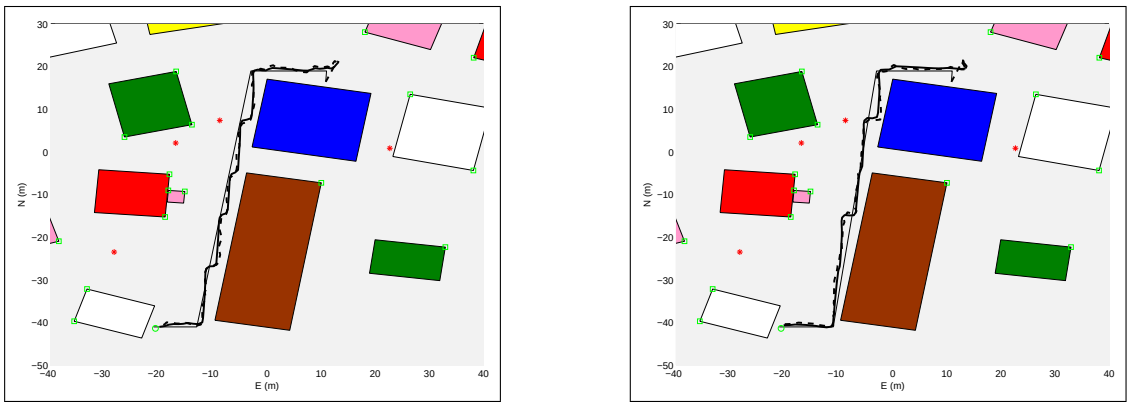
Figure 6 shows how the mean cumulative reward changed with the sensor quality for the MDP and the GAMDP. We see that as the uncertainty increased, the performance of the MDP planner suffered while the GAMDP planner remained significantly better.

In particular, the most interesting change was from the navigation to tactical grade sensors. Firstly, the percentage of simulations in which the MDP planner successfully reached the goal position drops 39.1% between the two sensor grades, while the GAMDP planner dropped only 16.8%. Secondly, increasing the uncertainty also increased the number of simulations in which the MDP planner ran into hazards, while the number of simulations in which the GAMDP planner ran into hazards stayed relatively constant.

B. Specific Scenarios

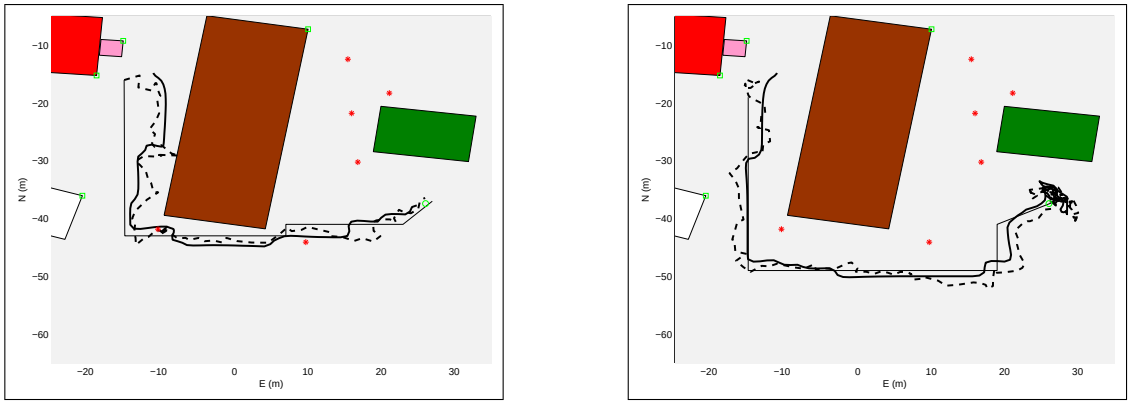
The performance of the GAMDP can be shown best in four specific scenarios. Each of these scenarios highlighted

[4]When generating randomized start, goal and hazard positions, any positions that lay within a building were discarded and the position generated again.



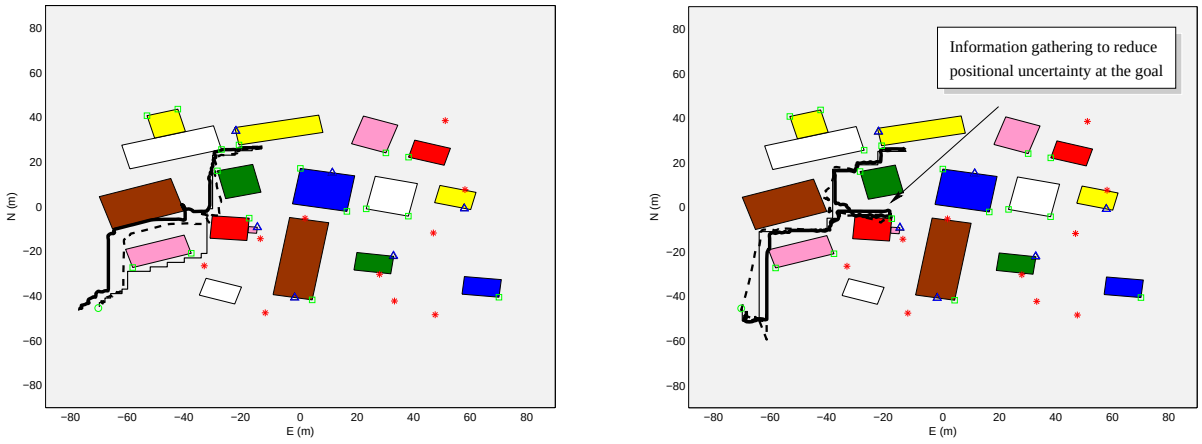
(a) MDP Path (b) GAMDP Path

Figure 7. (a) The MDP path with navigational grade IMU (small uncertainty). (b) The GAMDP chose a very similar path to the MDP when the uncertainty was small. In both figures, the thin solid line is the planned path, the dashed line is the estimated path and the bold solid line is the actual path.



(a) MDP Path (b) GAMDP Path

Figure 8. (a) The MDP with automotive grade IMU (large uncertainty). Since the MDP had no observability into the uncertainty in the agent's position, it chose shorter paths, dangerously close to hazards. (b) The GAMDP chose paths that maintained a safer distance from the obstacles. Even with substantial uncertainty, it was very unlikely that the agent would run into a hazard. In both figures, the thin solid line is the planned path, the dashed line is the estimated path and the bold solid line is the actual path.

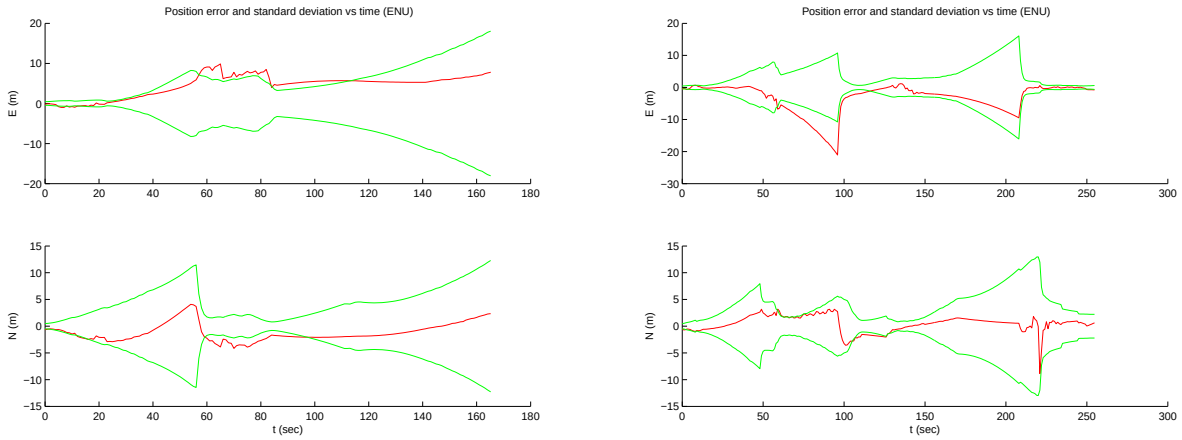


(a) MDP Path (b) GAMDP Path

Figure 9. (a) Since the MDP had no observability of the uncertainty in the agent's position, it was not able to choose to localize before executing the 'stop' action. As a result it did not successfully arrive at the goal position. (b) Since the GAMDP did have observability into the uncertainty in the agent's position, it chose to use the information gathering action (landmark bearing measurement) in order to localize before executing the 'stop' action. As a result it was able to successfully reach the goal position. The solid line is the planned path, the dashed line is the estimated path and the dotted line is the actual path.

a hypothesis that was confirmed or a lesson that was learned from the simulated experiments.

The first scenario confirmed the hypothesis that the MDP planner would perform very well when the uncertainty remained small, and demonstrated that the GAMDP planner behaved in a manner very similar to the MDP planner when the uncertainty was small. Figures 7(a) and 7(b) show the planned, actual and estimated paths for the MDP and GAMDP in a simulation that was run using the navigation-grade IMU and range sensors. As expected, the GAMDP and MDP paths



(a) MDP (b) GAMDP

Figure 10. A comparison of errors and uncertainty. The solid red line is the error, and the green bounds represent the standard deviation of the uncertainty estimate. (a) On the left we can see that when using the MDP planner, the agent's position uncertainty remained large throughout the simulation. (b) When using the GAMDP planner, the position uncertainty was reduced using information gathering actions at the end of the simulation.

were very similar, and the planned, actual and estimated paths were very similar as well.

The next two scenarios demonstrated the two advantages that the GAMDP gained over the MDP as the uncertainty increased. Figures 8(a) and 8(b) show the planned, actual and estimated paths for the MDP and GAMDP in a simulation using the tactical-grade sensors. Although the uncertainty in the stochastic MDP transitions was increased by using a larger characteristic velocity uncertainty, the MDP was not able to capture the fact that the uncertainty continued to grow beyond one transition. It was overly confident that it would not run into the hazards and it chose a path dangerously close to them. In this simulation, as a result of the uncertainty, the agent strayed from the planned path and ran right over a hazard. The GAMDP planner, however, was able to capture the fact that the agent's position uncertainty would become moderately large by the time it reached the obstacles. As a result, it was willing to choose a slightly longer path in order to maintain a very low probability that the agent will run into the hazards.

The second advantage of the GAMDP planner compared to the MDP planner was an increased likelihood that the agent would successfully reach the desired goal location. Figure 9 shows example paths in which the automotive-grade sensors were used. These low-grade sensors resulted in a large positional uncertainty as the agent approached the goal position. The GAMDP planner used an information gathering action (the landmark bearing measurement) to localize before completing its path. Figure 10(a) shows the agent's positional uncertainty, which converges as the agent nears the goal location. In contrast, the MDP had no observability into the agent's position uncertainty and could not choose to take actions to localize before completing the path. Figure 10(b) shows the agent's position uncertainty, which remained large throughout the simulation. In this example, the GAMDP planner successfully reached the goal position and the MDP planner did not.

C. Time Analysis

The increase in robustness of the GAMDP does introduce additional computational cost compared to the MDP planner. The GAMDP state space is much larger than the MDP state space (by a factor of $|\Sigma|^2$ in this case), causing value iteration to take longer for the GAMDP than for the MDP. However, the dominant cost of solving a single motion planning problem using the GAMDP is the cost of generating the state transition function. Computing a GAMDP state transition generally requires a matrix inversion to propagate the EKF state, which means that each GAMDP state transition takes much longer to compute than an MDP state transition. Combined with the increase in the number of states, the GAMDP state transition function takes considerably longer to compute than the MDP state transition function. Even though the transition function can be computed off-line, this increase in computational time is significant. An additional computational penalty results from the need to calculate the reward function by computing the belief corresponding to each GAMDP state and applying equation (15). However, once the transition and reward functions are generated, solving additional planning problems in the same domain allow us to amortize the computational cost of building the GAMDP model, so that solving additional problems is generally fast.

The mean computational times for the GAMDP and MDP, over the set of 960 simulations, are shown in table II. The GAMDP planner contained 2.56 million states, and the MDP contained 6400 states. All computations were performed using a 2.2 GHz Intel Pentium 4 processor, using C++.

Table II

THE MEAN COMPUTATIONAL TIMES FOR THE GAMDP AND MDP.

	GAMDP	MDP
Transitions (off-line)	4×10^4 sec	1 sec
Rewards (off-line)*	4800 sec	76 sec
Value Iteration	100 sec	2 sec

Note that the time to compute the GAMDP solution for any particular problem is partly dependent on the termination condition of value iteration (step 5 of algorithm 2); frequently the optimal policy can be found in many fewer iterations than solving for the optimal value function, especially if the transition function is nearly deterministic and if the order is chosen carefully that state values are computed. The timing results in table II are for a termination condition that led to only a single step of complete value iteration, but resulted

in the policies and results shown in figures 6 to 10. The number of steps of value iteration required to solve for the optimal policy will vary, but experimentally good policies could generally be found in tens of seconds.

VI. Conclusions and Future Work

The intent of this research was to develop an efficient path planner which is robust to the types of uncertainty that an agent might experience in the real world. We noted that a POMDP approach is optimal under uncertainty, but is not computationally tractable for the typically large state and action spaces of the path planning problem. We also noted that the underlying causes of the complexity of the POMDP stem from the fact that the POMDP plans over the set of beliefs, which is an $(n-1)$ dimensional simplex for n discrete states. This insight has allowed developers to create tractable algorithms which approximate the solution to the POMDP by planning over a subset of the belief space. If this subset is chosen to be representative of the set of beliefs that the agent will experience, the approximate solution will be very robust to uncertainty. If we pair our planner with an EKF, a natural approximation of the belief space presents itself, namely, the set of Gaussian distributions.

We demonstrated, via simulation, that if an agent's states can be estimated using an EKF, then the GAMDP provides a computationally tractable planner which is significantly more robust to uncertainty than the basic MDP planner.

We found that one disadvantage of the GAMDP is that, although a policy can be computed relatively efficiently using the value iteration algorithm given a model, pre-computation of the model can be very costly and must be done off-line. This means that our representation of the map cannot change during implementation, which prohibits us from solving problems in which parts of the map are dynamic, or parts of the map are estimated on-line. Another disadvantage of the GAMDP is its heavy reliance on the fidelity of the sensor models and its sensitivity to the resolution of the discrete state approximations.

A. Future Work

Future work on the EKF extension to the GAMDP will focus in three areas. One of these areas is further verification of the algorithm. Although simulations showed that the algorithm provided a significant increase in robustness for a model whose complexity is representative of a real world application, many real world implementation issues cannot be captured by a simulation. In order to verify that these issues are not prohibitive to implementation of the algorithm, future work will involve testing the algorithm on a real system. In addition, before implementing the algorithm on a real system, a study must be done to verify that any violations of the assumption in the covariance transition model can be eliminated with an appropriately conservative sensor model, and to gain insight into how conservative the models need to be.

Another focal area of future work will be improvements to the algorithm. These improvements will include a variable resolution version of the algorithm which will make the solution faster, and make the algorithm applicable to larger maps (larger state spaces). Although it was not considered in this paper, the problem of uncertainty in the map is not to be neglected. It has received much attention in the literature [18]–[20]. This motivates another improvement to the algorithm, the ability to plan in dynamic and uncertain environments, which will allow the agent to do both planning and mapping on-line. Preliminary studies indicate that both of these improvements can be accomplished by learning (or estimating) the GAMDP value function using samples from the model, similar to Q-learning [21].

The third area of future research will be application of the algorithm to more difficult problems. These will include problems with multiple cooperative and/or adversarial agents, and problems with additional tasks such as target tracking.

References

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[2] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME-Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[3] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[4] P. M. Newman and J. J. Leonard, "Consistent convergent constant time SLAM," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI*, Acapulco Mexico, August 2003.

[5] R. Bellman, *Dynamic Programming*. NJ: Princeton University Press, 1957.

[6] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Advances in Neural Processing Systems 12*, vol. 12, 1999, pp. 1043–1049.

[7] M. Puterman, *Markov Decision Processes-Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc., 1994.

[8] A. Cassandra, "Exact and approximate algorithms for partially observable markov decision processes," PhD Thesis, Brown University, Providence, RI, 1998.

[9] E. Sondik, "The optimal control of partially observable markov decision processes," PhD Thesis, Stanford University, Stanford, CA, 1971.

[10] M. Littman, "Algorithms for sequential decision making," PhD Thesis, Brown University, Providence, RI, 1996.

[11] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for pomdps," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, 2003.

[12] W. S. Lovejoy, "Computationally feasible bounds for partially observable Markov decison processes," *Operations Research*, vol. 39, pp. 192–175, 1991.

[13] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "3-d evolutionary algorithm based off-line / on-line path planner for uav navigation," *IEEE Transactions on System, Man and Cybernetics, Part B*, December 2003.

[14] G. Theocharous, "Hierarchical learning and planning in partially observable markov decision processes," Ph.D. dissertation, Michigan State University, Lansing, MI, 2002.

[15] R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*. New York, NY: John Wiley & Sons, Inc., 1997.

[16] W. Travis and D. M. Bevly, "Navigation errors introduced by ground vehicle dynamics," *Online Journal of Space Communication*, vol. 9, 2006.

[17] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *ICRA*, 1985.

[18] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robotic Vehicles*, I. J. Cox and G. T. Wilfong, Eds. Orlando, FL: Springer-Verlag, 1990.

[19] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Ng, "Simultaneous mapping and localization with sparse extended information filters: Theory and initial results," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[20] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, 2004, to appear.

[21] C. J. Watkins, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.