2016 American Control Conference (ACC)
Boston Marriott Copley Place
July 6-8, 2016. Boston, MA, USA

A Stochastic Approach for Attack Resilient UAV Motion Planning

Nicola Bezzo, James Weimer, Yanwei Du, Oleg Sokolsky, Sang H. Son, Insup Lee

Abstract— **In this paper we propose a stochastic strategy for motion planning of unmanned aerial vehicles (UAVs) subject to malicious cyber attacks. By injecting erroneous information and compromising sensor data, an attacker can hijack a system driving it to unsafe states. In this work we bear on the problem of choosing optimal actions while one or more sensors are not reliable. We assume that the system is fully observable and one or more measurements (however unknown) return incorrect estimates of a state. We build an algorithm that leverages the theory of Markov decision processes (MDPs) to determine the optimal policy to plan the motion of a UAV and avoid unsafe regions of a state space. We name *Redundant Observable MDPs* (ROMDPs) this class of markovian processes that deal with redundant attacked measurements. A quadrotor case study is introduced and simulation and experimental results are presented to validate the proposed strategy.**

I. Introduction

In recent years autonomous robotic systems are becoming more and more popular for both civilian and military operations. Thanks to the continue growing of sensing and computation capabilities, this class of cyber-physical systems can perform complex missions in unknown environments, with small or even inexistent user interactions. However this increase in functionality is also introducing security vulnerabilities which can compromise the integrity of the system. In fact not only a robot has to deal with noisy measurements from its sensors and possible disturbances from the environments, but nowadays, also malicious attacks on the sensing and communication infrastructure of the system need to be taken into account to avoid reaching unsafe regions of a state space. In this work we consider a way-point navigation case study for an unmanned aerial vehicle (UAV) that needs to fly between two locations within a certain environment. In normal conditions (not attacked) the robot will periodically estimate its location using GPS, cameras, and inertial sensors and compensate for disturbances (e.g., wind) to reach its desired goal. If the measurements noise and disturbances are high, the controller may not be robust enough to drive the UAV to the final goal. If now we also consider that a malicious attack could hide within the measurements noise and disturbance, the system will be confused about its state which could lead to an action in favor of the attacker - e.g., the attacker hijacks the robot to a third unwanted location. Examples of these attacks have already been exploited and demonstrated with real vehicles like in [1] where a GPS was spoofed hijacking a yacht off route.

Nicola Bezzo is with the Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA 22902, USA nbezzo@virginia.edu
James Weimer, Yanwei Du, Oleg Sokolsky, and Insup Lee are with the PRECISE Center, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA {weimerj, duyanwei}@seas.upenn.edu {sokolsky, lee}@cis.upenn.edu
Sang H. Son is with the Real-Time Cyber-Physical Systems Research Laboratory, DGIST, Korea son@digist.ac.kr

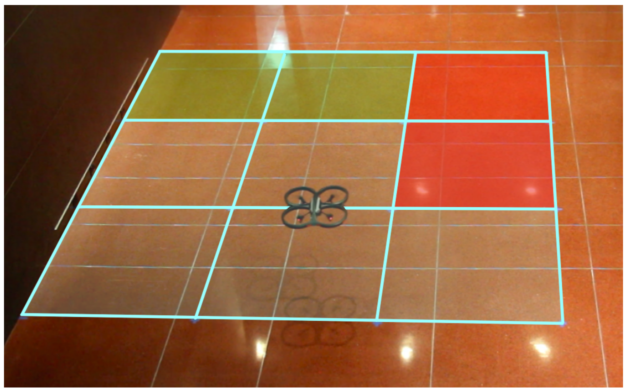


Fig. 1. Example of mission envisioned in this work: the UAV needs to reach goal areas (green) and avoid the undesired areas (red) in the environment while one or more measurements are maliciously compromised by attacks.

Fig. 1 shows an example of a waypoint navigation scenario in which a quadrotor UAV needs to cross a grid, to reach a goal point (green areas) and avoid undesired regions (red areas). If the robot is confused about its current state, an improper action can drive the system inside the undesired regions of the workspace. The example displayed in Fig. 1 will be used as a reference for the experimental validations farther on in this paper.

Thus, within this work we are interested in determining the optimal actions to take to control an autonomous robotic system given uncertain estimations of its state, due to possible attacks on sensor measurements. Problems like the one described above, can be mapped into Markov decision processes (MDPs) and applied not only to robotic systems but to any cyber-physical system that consists of sensor measurements for state estimation, like power plants, medical systems, and transportation systems. The proposed technique in this paper, named *Redundant Observable Markov Decision Process* (ROMDP) considers uncertainties in the actions and leverages redundant sensor measurements with partially compromised observations (partially here is in terms of number of sensors that return different observations) to determine the optimal action to take and minimize the probability of reaching undesired regions of a workspace. Note that, differently from partially observable MPDs (POMDPs) in which a system is not capable of observing its state for a lack of estimation information, within our strategy we consider that at least one of the observations (not known a priori) contains complete knowledge of the state.

The contribution of this paper is threefold: i) we develop an algorithm that leverages MDPs to find the optimal planning policy when malicious attacks on one or more sensors are present, ii) the proposed technique works as a worst case attack-resilient controller dealing with up to $N-1$ sensors under attack, and iii) we validate the proposed technique through extensive simulations and hardware implementations on a quadrotor UAV.

The rest of the paper is organized as follows. In Section II we review some of the recent literature on the topics of attack detection and estimation. In Section III we formally define the problem under investigation followed by details about the ROMDP algorithm in Section IV. A quadrotor case study is then presented in Section V outlining simulation results under sensor attacks with noisy measurements and environment disturbances, followed by an indoor experiment with a self localizing UAV architecture. Conclusions and future work are finally drawn in Section VI.

II. Related Work

The study of high assurance vehicular systems is a recent topic that is attracting several researchers in both the control and computer science communities. Malicious attacks are defined as adversarial actions conducted against a system or part of it and with the intent of compromising the performance, operability, integrity, and safety of the system. The main difference between failure and malicious attack is that the former is usually not coordinated while an attack is usually camouflaged or stealthy and behaves and produces results similar or expectable by the dynamics of the system and environment disturbances.

Even though this area of study is still at an early stage, some preliminary work on vehicular security was performed in [2] in which the authors showed through intensive experiments on common cars, that an attacker could take over a vehicle and compromise its safety. Specifically it was shown that the CAN bus system is unprotected and several functionalities of a car can be controller and accessed by different devices in the vehicle.

Standing from a control perspective, authors in [3] propose a resilient consensus algorithm based on receding-horizon control to deal with replay attacks between an operator and a remotely controlled unmanned ground vehicle. In [4] the authors use plant models for attack detection and monitoring in cyber-physical systems. In [5], motivated by SCADA system security control, the authors incorporate knowledge of the physical system under control to detect cyber attacks that change the behavior of a control system.

For deterministic linear systems this problem of attack detection and estimation has been mapped into an l_0 optimization problem in [6]. In [7] we leverage the work in [6] and present a state estimation method for linear systems in the presence of attacks showing that an attacker cannot destabilize the system by exploiting the difference between the model used for the state estimation and the real physical dynamics of the system. In [8] we propose a recursive implementation based on the linear-quadratic estimator in which together with the update and predict phases a shield procedure is added to remove the malicious effects of attacks on noisy sensor measurements.

In this paper we move one step forward by considering sensor attacks on stochastic systems with input-output probabilistic models. To solve this problem we leverage the theory of MDP [9], [10] to obtain an optimal planning policy.

III. Problem Formulation

Within this work we are interested in finding an optimal strategy to maximize the probability that a robot under malicious attack can reach a desired state without being hijacked. We assume that the robot dynamics can be represented as a discrete-time linear time-invariant (LTI) system of the following form

$$\begin{aligned} \mathbf{x}_{k+1} &= A\mathbf{x}_k + B\mathbf{u}_k + \mathbf{d}_k, \\ \mathbf{z}_k &= C\mathbf{x}_k + \nu_k, \\ \mathbf{y}_k &= \mathbf{z}_k + \lambda_k, \end{aligned} \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$, $\mathbf{u}_k \in \mathbb{R}^m$, and $\mathbf{d}_k \in \mathbb{R}^m$ are the state vector, control inputs, and disturbance at time k respectively. The disturbance is considered bounded, meaning that we assume a maximum disturbance above which value, the robot becomes unstable or uncontrollable. $\mathbf{z}_k$ is the set of measurements without the effects of attack while $\mathbf{y}_k = [y_{k,1}, y_{k,2} \cdots y_{k,N}] \in \mathbb{R}^N$ is the sensor measurements vector with attack where $y_{k,i} \in \mathbb{R}$ is the measurement taken by the i^{th} sensor at time k. λ_k is the attack vector in which each term represents an attack of magnitude $\|\lambda_{k,i}\|$. The attack vector is assumed to be arbitrary, but stealthy i.e., hidden within the noise and disturbance profile of the system. Finally, the sensor measurement noise $\nu_k = \mathcal{N}(0, V, e_\nu)$ is expressed as a truncated Gaussian random variable (i.e., bounded normal distribution) with maximum error e_ν.

Given $\{s^{\text{goal}}, s^{\text{bad}}\} \in \mathcal{S}$ with $\mathcal{S}$ a finite set of states of the world, our problem for this work can be expressed as follows

Problem 1: **Optimal Planning against Attacked Sensors in Stochastic Systems** Given a vehicle with N sensors measuring state $s \in \mathbf{x}$ while one or more sensor measurements $y_i \in \mathbf{y}$ are maliciously compromised by an adversarial attacker, find the optimal action policy π that maximizes the probability that the system can reach a desired state s^{goal} without being hijacked to s^{bad}.

In our previous work, [7], [8], an upper bound on the maximum tolerable number of attacked sensors was imposed equal to $N/2$ to provide attack-resiliency guarantees. In this work we relax this constraint and consider a worst case scenario where up to $N-1$ compromised sensors are allowed. This scenario is considered as the extreme case, since to have all sensors attacked would imply a completely unobservable system from which it is impossible to estimate a state.

IV. Redundant Observable Markov Decision Processes

In this section we outline a novel algorithm for solving Markov decision processes (MDPs) when one or several measurements (up to $N-1$) return incorrect observations of the world. The technique that we derive in this work deals also with sensor faults; however, here we are far more interested in the attack scenario where an attacker could drive the system to undesired/unsafe states (e.g., a bad region of a workspace) while hiding under the expected disturbance and noise model dynamics of the system. POMDP could be used to solve our problem but it would be an unnecessary and unpractical over-complication. The ROMDP approach presented next leverages MDP making the problem more computationally efficient while guaranteeing optimality.

A. ROMDP Framework

The Redundant Observable MDP (ROMDP) framework attempts to solve the problem of computing the best policy to safely navigate a UAV when its measurements are not consistent because under attack.

A ROMDP can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \mathcal{C} \rangle$ where:

- $\mathcal{S}$ is a finite set of states of the world;
- $\mathcal{A}$ is a finite set of actions;
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Gamma(\mathcal{S})$ is the transition probability: a function mapping elements of $\mathcal{S} \times \mathcal{A}$ into discrete probabilities distributions over $\mathcal{S}$. Specifically we will represent $\mathcal{P}_a(s, s') = Pr(s_{k+1} = s' | s_k = s, a_k = a)$ as the probability that action a in state s at time k will result into state s' at time $k+1$. This mapping is crucial to take into account the possible effects that any disturbance can have on the system;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function that specifies the instantaneous reward received by taking each action at each state. $\mathcal{R}_a(s)$ is the reward received by taking action a from state s;
- $\mathcal{O}$ is a finite set of weighted observations that follows a certain action a. For instance $\mathcal{O}_k = \{s_{i,N_i}, s_{j,N_j}\}$ indicates that two states, s_i and s_j were observed after taking N measurements at time k. s_i was recorded N_i times and s_j, N_j times with $N = N_i + N_j$;
- $\mathcal{C} : \mathcal{O} \rightarrow \Gamma(\mathcal{S})$ is a confidence function mapping elements of $\mathcal{O}$ into discrete probabilities distributions in $\mathcal{S}$. We will use $\mathcal{C}(s)$ to represent the probability that the robot is in state s given the observation $\mathcal{O}$.

Specifically here $\mathcal{O}$ and $\mathcal{C}$ are the two elements added from the conventional definition of MDP.

A naive approach for calculating $\mathcal{C}$ is to classify $\mathcal{O}$ based on the number of different observations for each state and evenly divide the probability among the observed states. This approach however does not consider the difficulty of attacking multiple sensors and can be limiting in scenarios where a system has several sensors. Thus, a better model is to consider that attacking multiple sensors is more complicated and less probable, which increases the probability associated with the state that receives more observations and decreases the confidence for the state that is observed the least. The transition probability for these scenarios, which we call *transition confidence* will have the following form

Definition 1: **Transition Confidence** If starting from a state s and choosing action a, a system observes different states $s_i \in \mathcal{O}$, each with a confidence probability $\mathcal{C}(s_i)$ s.t. $\sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) = 1$, then the transition probability to state s' which we call *transition confidence* becomes

$$\mathcal{P}_a(\mathcal{O}, \mathcal{C}, s') = \sum_{s_i \in \mathcal{O}} Pr(s' | s_i, \mathcal{C}(s_i), a) = \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_a(s_i, s') \tag{2}$$

Given these premises, our ROMDP framework follows the MDP process in which a measure of the cumulative future received reward is maximized over a finite or infinite horizon. Within a finite-horizon approach the system has a finite lifetime K which in the majority of cases is known and can be approximated while, if an infinite-horizon approach is chosen, $K = \infty$. More specifically we have

$$E\left[\sum_{k=0}^{K} \gamma^k r_k\right] \tag{3}$$

in which the rewards are summed over the lifetime of the system, but discounted geometrically using the discount factor $0 < \gamma < 1$. Finally the robot should act as to maximize (3). The larger the discount factor, the more future rewards affect the current decision making. Since we are considering a Markov process, the current state and action are the only mean to predict the next state. In order to find the best action at each state, a *policy* π, mapping $\mathcal{S} \rightarrow \mathcal{A}$, is necessary to describe the behavior of the robot. Therefore π_k will be the policy to be used to choose the action a_k on state s_k, from the k^{th} time to K. Let $V_{\pi,k}(s)$ represent the expected sum of rewards gained by executing policy π_k from state s, then the k^{th} value associated with policy π_k from state s is given by

$$V_{\pi,k}(s) = R_{\pi_k(s)}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{\pi_k(s)}(s, s') V_{\pi,k-1}(s') \tag{4}$$

which means that to evaluate the future, we need to consider all resulting states s', the probability of their occurrence $\mathcal{P}_{\pi_k(s)}(s, s')$ and their value $V_{\pi,k-1}(s')$.

If the robot is not able to determine its state with complete reliability, we need to consider all possible outcomes when implementing a certain policy π, to better select the optimal action a to take. Following our ROMDP formulation above, we can modify (4) to consider uncertainties in the observations due to malicious attacks, as follow

$$V_{\pi,k}(\mathcal{O}) = R_{\pi_k,\mathcal{O}} + \gamma \sum_{s' \in \mathcal{S}} \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_{\pi_k}(s_i, s') V_{\pi,k-1}(s') \tag{5}$$

with

$$R_{\pi_k,\mathcal{O}} = R_{\pi_k}(\mathcal{O}) = \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) R_{\pi_k}(s_i) \tag{6}$$

Equation (6) represents the expected reward the system receives using policy π_k given a certain confidence that the robot is in a certain state s. A mapping from $\mathcal{O}$ to $\mathcal{S}$ will give the identity $V_{\pi,k}(s_i \in \mathcal{O}) = V_{\pi,k}(\mathcal{O}) \; \forall s_i \in \mathcal{O}$. This mapping is necessary to project the obtained values back into the original state space $\mathcal{S}$.

So far we have derived a formula to calculate a value function given a policy. We are now interested in deriving a policy based on the value function. A greedy policy with respect to the value function in (5) is defined as

$$\pi_V(\mathcal{O}) = \arg\max_a \left[R_{a,\mathcal{O}} + \gamma \sum_{s' \in \mathcal{S}} \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_a(s_i, s') V_a(s') \right]$$

which represents the policy obtained at every step taking the action that maximizes the expected immediate reward and the discounted value of the next state. The optimal policy at the k^{th} step, $\hat{\pi}_k$ is achieved from the $(k-1)$ step value function $\hat{V}_{k-1} = V_{\hat{\pi}_{k-1},k-1}(s')$.

$$\hat{\pi}_k(\mathcal{O}) = \arg\max_a \left[R_{a,\mathcal{O}} + \gamma \sum_{s' \in \mathcal{S}} \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_a(s_i, s') \hat{V}_{k-1} \right]$$

Now, let's introduce the following result in MDP, which is also applicable herein for our ROMDP formulation.

Proposition 1: **Blackwell Optimality** In any ROMDP with finitely many states $\mathcal{S}$ and finitely many actions $\mathcal{A}$, there is a stationary policy $\hat{\pi}$ that is optimal for every discount factor γ that tends to one.
We omit here the proof for the Blackwell optimality since it follows the same proof applied to MDP, found in [11]. Thus, using this result, since the state space and actions are finite, there exists an optimal stationary policy $\hat{\pi}(\mathcal{O})$ from which the optimal value function $\hat{V}(\mathcal{O})$ is defined as

$$\hat{V}(\mathcal{O}) = \max_a \left[R_{a,\mathcal{O}} + \gamma \sum_{s' \in \mathcal{S}} \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_a(s_i, s') \hat{V}(s') \right] \tag{7}$$

It is important to note that if at any time, all observations are superimposing (i.e., only one state is observed at each iteration), then we are in the classical MDP scenario with $\mathcal{O} = \{s\}$ and $\mathcal{C}(s) = 1$. Solving (7) can be computationally expensive especially for large state and action spaces. Fortunately, there are several methods in the literature for finding optimal MDP policies using finite horizon iterations [12], [9]. The most common way is to use the *value iteration* process that computes iteratively the sequence $\hat{V}_k$ of discounted finite-horizon optimal value functions in (7). The iterative algorithm presented below computes improved estimates of the optimal value function in ROMDPs.

Algorithm 1 ROMDP Value Iteration Algorithm

```
1:  $k \leftarrow 1$
2:  $V_k(s) \leftarrow 0 \; \forall s$
3:  while $k \leq K$ or $|V_k(s) - V_{k-1}(s)| \geq \epsilon \; \forall s \in \mathcal{S}$ do
4:      $k \leftarrow k + 1$
5:      for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ do
6:          $\mathcal{Q}_{a,k}(s) \leftarrow R_a(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_a(s_i, s') V_{k-1}(s')$
7:          $V_k(s) \leftarrow \max_a \mathcal{Q}_{a,k}(s)$
8:      end for
9:      if $\mathcal{O}$ is not a singleton then
10:         $\mathcal{Q}_{a,k}(\mathcal{O}) \leftarrow R_a(\mathcal{O}) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_a(\mathcal{O}, \mathcal{C}, s') V_{k-1}(s')$
11:         $V_k(\mathcal{O}) \leftarrow \max_a \mathcal{Q}_{a,k}(\mathcal{O})$
12:     end if
13: end while
```

Using this framework, a system acts in order to maximize the expected sum of rewards that it gets on the next k steps. Selecting an improper reward function value may lead to different actions. If the objective is to avoid undesired states, the reward for these undesired states has to be large and negative (approaching -∞) as described in the following lemma.

Lemma 1: **Necessary and Sufficient condition for Resiliency in ROMDPs** At every time k a system modeled as a ROMDP is able to avoid an undesired state s^{bad} at $k+1$ if (*necessary condition*) $\exists \; a \in \mathcal{A}$ s.t. $\mathcal{P}_a(\mathcal{O}, s^{\text{bad}}) = 0$ and it is guaranteed to avoid an undesired state s^{bad} at $k+1$ if (*sufficient condition*) one of the following conditions is satisfied:

1) the transition probability associated with s^{bad} is null or more formally $\forall \; a \in \mathcal{A}$, $\mathcal{P}_a(\mathcal{O}, s^{\text{bad}}) = 0$.
2) the necessary condition is satisfied and the reward associated to s^{bad} is a sufficiently large negative value, more specifically if $R(s^{\text{bad}}) < -\frac{R(s^{\text{goal}})}{\min C(s) P_a(s,s')}$

Proof: Necessary Condition: The proof is straightforward and can be derived by using the transition probability definition. ■

Proof: Sufficient Condition: The proof for condition 1) is intuitive. To guarantee the second sufficient condition we need to satisfy the following inequality

$$\begin{aligned} & R_{a_i}(\mathcal{O}) + \gamma \sum_{s' \in \mathcal{S} \setminus s^{\text{bad}}} \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_{a_i}(s_i, s') \hat{V}(s') \\ & \quad + \gamma \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_{a_i}(s_i, s^{\text{bad}}) R(s^{\text{bad}}) \\ & \quad < R_{a_j}(\mathcal{O}) + \gamma \sum_{s' \in \mathcal{S}} \sum_{s_i \in \mathcal{O}} \mathcal{C}(s_i) \mathcal{P}_{a_j}(s_i, s') \hat{V}(s') \end{aligned} \tag{8}$$

Now, assuming worst case scenario, (8) becomes

$$\min_{C, P_a} \left[C(s) P_a(s, s') \right] R(s^{\text{bad}}) + R(s^{\text{goal}}) < 0 \tag{9}$$

in which, we have assumed $R_{a_i} = R_{a_j}$ since normally the workspace is uniform with the same reward everywhere besides at the goal and the areas to avoid. Thus, we have obtained the inequality for the second sufficient condition of the lemma. ■

Although the conditions for Lemma 1 are limit cases, a system will always try to maximize its reward and thus take an action that minimizes the probability of reaching s^{bad}. As it will be shown in the simulation results in Section V-C, an improper selection of the parameters in (7) can lead to unsafe situations.

V. Quadrotor Case Study

The case study investigated in this paper is a motion planning way-point navigation mission in which a quadrotor aerial vehicle needs to cross a workspace from a starting point to a desired goal avoiding undesired locations.

A. Quadrotor Model and Controller

Fig. 2 shows the coordinate system and free body diagram for modeling the quadrotor dynamics. The quadrotor center of mass in the inertial frame is specified by the position vector $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$. The roll, pitch, and yaw Euler angles, ϕ, θ, and ψ, specify the quadrotor attitude and the body angular velocity is then defined as the rate vector $\omega = \begin{bmatrix} p & q & r \end{bmatrix}^T$, [13], [14].

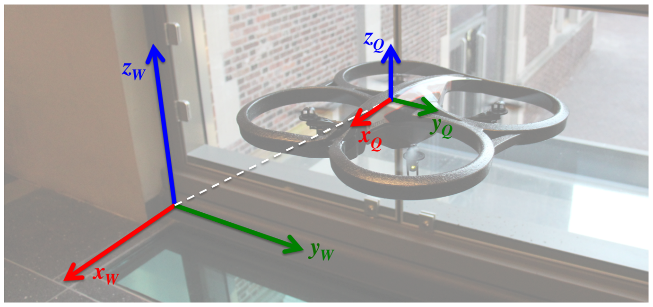


Fig. 2. Quadrotor and World frames.

The controller is derived by linearizing the equations of motion and motor models at an operating point that corresponds to the nominal hover state $\mathbf{x} = \{x, y, z\}$, $\theta = 0$, $\phi = 0$, $\psi = \psi_0$, $\dot{\mathbf{x}} = 0$ and $\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$ with roll and pitch angles small. The nominal values for the inputs at hover are $u_1 = mg$, $u_2 = u_3 = u_4 = 0$.

In order to control the quadrotor to follow a desired trajectory, we use the architecture in Fig. 3.

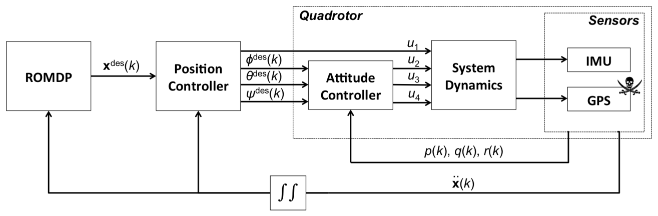


Fig. 3. Diagram of the overall controller used on a quadrotor for ROMDP operations.

The position control is used to track a desired trajectory ξ characterized by $\mathbf{x}_\xi(t)$ and $\psi_\xi(t)$. Using a PID feedback controller on the error $e_k = (\mathbf{x}_{k,\xi} - \mathbf{x}_k)$ we can control the position and velocity of the quadrotor to maintain a desired trajectory. After linearization, we can obtain the relationship between desired roll and pitch angles and desired accelerations as follows

$$\phi^{\text{des}} = \frac{1}{g}\left(\ddot{x}^{\text{des}} \sin(\psi_\xi) - \ddot{y}^{\text{des}} \cos(\psi_\xi)\right) \tag{10}$$

$$\theta^{\text{des}} = \frac{1}{g}\left(\ddot{x}^{\text{des}} \cos(\psi_\xi) + \ddot{y}^{\text{des}} \sin(\psi_\xi)\right) \tag{11}$$

and

$$u_1 = mg + m\ddot{z}^{\text{des}} \tag{12}$$

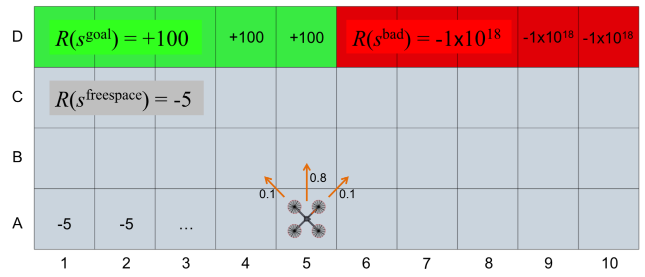
Finally, the attitude control is realized using a PD controller as follows

$$\begin{pmatrix} u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} k_{p,\phi}(\phi^{\text{des}} - \phi) + k_{d,\phi}(p^{\text{des}} - p) \\ k_{p,\theta}(\theta^{\text{des}} - \theta) + k_{d,\theta}(q^{\text{des}} - q) \\ k_{p,\psi}(\psi^{\text{des}} - \psi) + k_{d,\psi}(r^{\text{des}} - r) \end{pmatrix} \tag{13}$$

B. Environment Setup

The environment configuration plays an important role in ROMDPs. The state space is discretely represented as an occupancy grid which maps the environment as an array of cells each representing a state and holding a probability value associated with the action taken by the robot. The minimum size of the cell has to be greater than the maximum displacement we can record due to measurements noise. Choosing small cells has the benefit that the robot could have a finer estimate of its state, however, fewer cells in the environment allows for a faster computation of the ROMDP algorithm. A better strategy which we plan to investigate in future work is to have adaptive cells whose size changes over time based on the observed state. Fig. 4 shows an example of occupancy grid for the missions considered in this work. Green colored cells represent the goals to reach while red colored cells are areas to be avoided.

At each step the vehicle will make some observations and compute Algorithm 1 that will produce the optimal action to perform. We define a finite set of primitive actions as follows



D
C
B
A
$R(s^{\text{goal}}) = +100$
+100
+100
$R(s^{\text{bad}}) = -1\text{x}10^{18}$
$-1\text{x}10^{18}$
$-1\text{x}10^{18}$
$R(s^{\text{freespace}}) = -5$
0.1
0.8
0.1
-5
-5
...
1
2
3
4
5
6
7
8
9
10

Fig. 4. Example of discretized environment. Each cell represents a different position state. Moving forward from state A5, the quadrotor has 0.8 probability of reaching B5 and 0.2 probability of reaching B4 and B6.

- $\mathcal{A}$ = { *move forward (F), move backward (B), move left (L), move right (R)* }.

Each action is mapped into the position and attitude control described in the previous section. Specifically here $\{F, B\}$ is mapped into u_3 and $\{L, R\}$ into u_2 control inputs. Associated with each action there is a transition probability $\mathcal{P}_a$. If there are not disturbances in the environment, each action will make the quadrotor move to the next cell in the direction of the action. Since UAVs often exhibit position drifts due to noise and disturbances, here we build a stochastic model of the disturbance effect on each action. Specifically we assume that there is a probability $\mathcal{P}_a(s, s')$ that the UAV will end up in the desired state s' from s, and $1 - \mathcal{P}_a(s, s')$ probability to reach either adjacent cells of s' ($s'-1$ or $s'+1$), as shown in Fig. 4.

C. Simulation Results

This section presents a series of Matlab simulation results on the ROMDP framework applied to a waypoint navigation case study for a quadrotor vehicle.

In the first simulation in Fig. 5, a 5×5 cells environment with 3 goal states s^{goal} and 2 undesired states s^{bad} is presented. Associated with each goal cell there is a reward $\mathcal{R}_a(s^{\text{goal}}) = 100$ while for the unwanted cells $\mathcal{R}_a(s^{\text{bad}}) = -1 \times 10^{18}$. For all other freespace locations of the environment $\mathcal{R}_a(s) = -5$. $\mathcal{P}_a(s, s^{\text{des}}) = 0.8$ and $P_a(s, s^{\text{des}} - 1) = P_a(s, s^{\text{des}} + 1) = 0.1$, as depicted in Fig. 4. Three position sensor measurements are available. A mismatch in the sensor measurements (i.e., multiple inconsistent observations) is displayed with gray colored cells. The first row of Fig. 5(a-e) displays consecutive snapshots for the motion of the quadrotor from state to state. The second row (f-j) shows the set of actions associated with the first row and calculated via the ROMDP algorithm presented in this paper. In Fig. 5(b,c) two of the three sensors are compromised. The attack is stealthy hidden within the disturbance model and it manifests as two measurements on the adjacent cell from the actual position of the quadrotor. The effect of the ROMDP is clearly visible in Fig. 5(d) and the corresponding action policy (i) in which the quadrotor chooses to move forward instead of going right. In fact, since the robot is undecided about its position, it considers both observations in Fig. 5(d) as reasonable: moving right in that particular case results in $2/3 \times 0.1 = 0.067$ probability of ending inside one of the bad areas. Although, this is a small chance, the controller decides to move forward and then to conclude safely the mission in Fig. 5(e). Fig. 6 shows the velocities of the quadrotor for

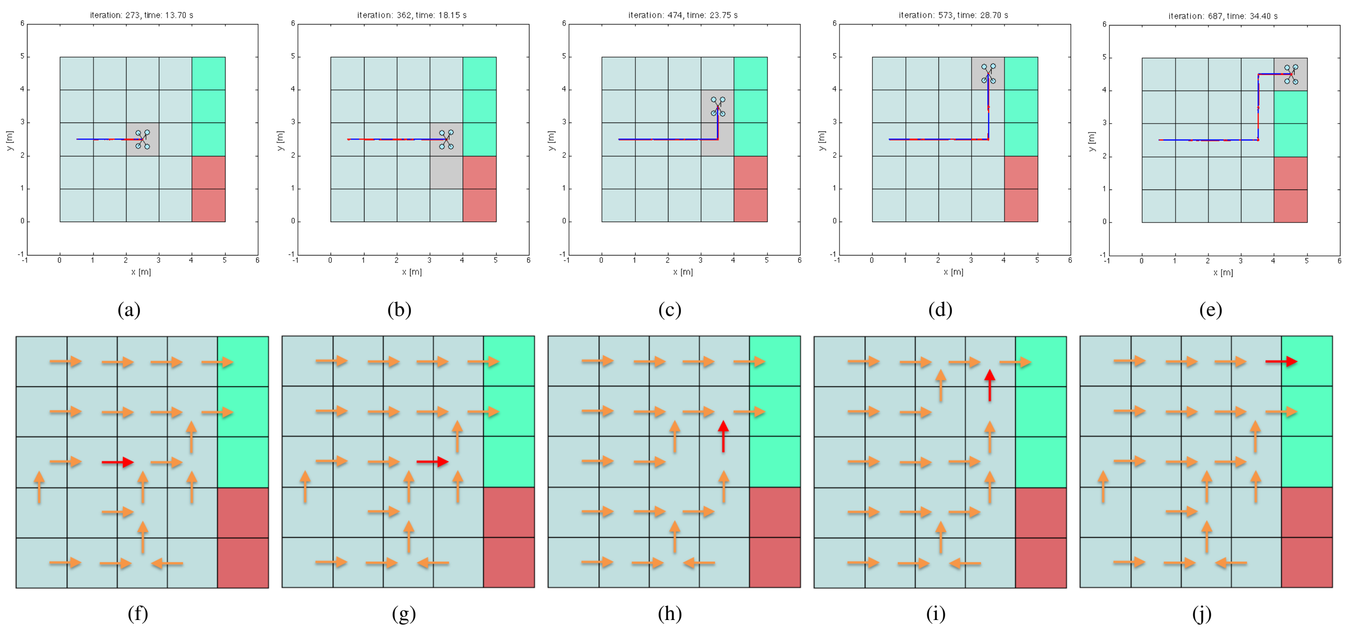


Fig. 5. Simulation results of the ROMDP algorithm implementation on a quadrotor waypoint navigation. The second row shows the computed action policy relative to the steps on the first row.

the mission presented in Fig. 5. Each "bump" in the two plots of the figure represents an action taken by the ROMDP controller. The sequence of actions for the mission in Fig. 5 is $\{R, R, R, F, F, R\}$.

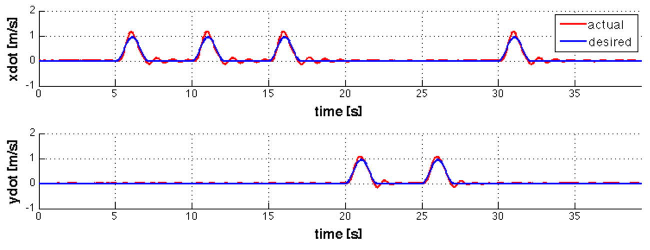


Fig. 6. Quadrotor x and y velocities

Fig. 7 shows a more complex simulation result in which disturbance (e.g., wind) is injected in the negative y direction drifting the quadrotor away from its path. Fig. 7(a) displays the cumulative trace of the attacked measurement (gray colored cells) in comparison with the actual path of the quadrotor. The attacker tries to leverage the disturbance effects, making the quadrotor believe that it is in a safe location, while it is actually in a cell where a lateral motion could send it inside the undesired red cell. The resilient ROMDP algorithm intervenes moving the robot forward and then right (Fig. 7(b)) where, although the disturbance, it reaches the desired goal.

Finally, both Table I and Table II summarize some simulation results run with different parameters and different environment setups. The first row of the table shows the setup for the *goal* and *bad* areas while the remaining cells of the environment have the same uniform reward ($R(s^{\text{freespace}}) = -5$). Last column shows the outcome from the simulations: "goal" means that the quadrotor was able to reach the goal, "bad" means that it run inside one of the bad regions, and "lm" means that it got stuck in a local minimum. As the reder can observe, the results on this tables

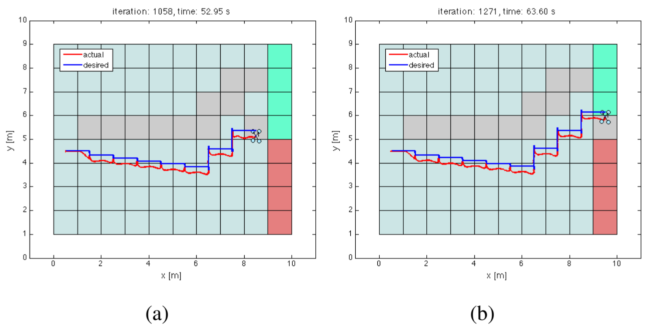


Fig. 7. Simulation results with environmental disturbances in the -y direction.

TABLE I

5×5 AREA, 3 GOOD CELLS, 2 BAD CELLS, ATTACK ON THE LEFT

🟩🟩🟩🟥🟥

Case	γ	$R(s^{\text{goal}})$	$R(s^{\text{bad}})$	$R(s^{\text{freespace}})$	Result
1	0.1	100	-1.0e18	-5	goal
2	0.1	1.0e6	-100	-5	bad
3	0.5	100	-1.0e18	-5	goal
4	0.5	1.0e6	-100	-5	bad

TABLE II

5×5 AREA, 1 GOOD CELL , 4 BAD CELLS, ATTACK ON THE RIGHT

🟥🟥🟩🟥🟥

Case	γ	$R(s^{\text{goal}})$	$R(s^{\text{bad}})$	$R(s^{\text{freespace}})$	Result
1	0.1	100	-100	-5	goal
2	0.5	100	-1.0e18	-5	lm
3	0.5	1.0e6	-100	-5	bad

conforms with Lemma 1. If $R(s^{\text{bad}})$ is sufficiently negative, the robot is always guaranteed to avoid s^{bad}, which means that sometime it can get trapped in local minima to avoid any risk of reaching undesired states.

D. Hardware Implementation

Preliminary indoor experiments using an AR.Drone 2.0 Parrot quadrotor [15] were implemented with different environment setups. Here we show the results for one of these implementations on a 3×3, 3.6 m^2, cells environment, as depicted in Fig. 1. The Parrot is equipped with two cameras (front and underneath), a sonar facing downward, and an IMU. Onboard, the vehicle has limited computational power allowing only low-level attitude control, leaving the position control and ROMDP algorithm presented in this paper on a base station linux-based machine. The base station laptop is equipped with a quad-core i7 CPU running the Robot Operating System (ROS) [16]. The linux box communicates with the quadrotor using standard Wi-Fi protocol at an average rate of 200 Hz. The high level position estimator is implemented with an Extended Kalman Filter (EKF) which fuses together vision, inertial, and sonar measurements, as described in [17]. To implement our ROMDP strategy we need at least two position measurements. Because the Parrot has limited capabilities, we duplicated the measurement as if two measurements were propagated from the quadrotor to the base station. Then one of these measurements was attacked. The environment setup for this experiment follows Fig. 1 with 2 goal cells and 2 undesired cells. Fig. 8 shows the position estimates from both the good and the attacked sensors in comparison with the desired position command, all recorded during the hardware implementation. The quadrotor was under attack starting from its initial state for two consecutive steps in the y direction (middle plot in Fig. 8). The optimal sequence of actions chosen by the ROMDP solver was $\{L, F, F\}$.

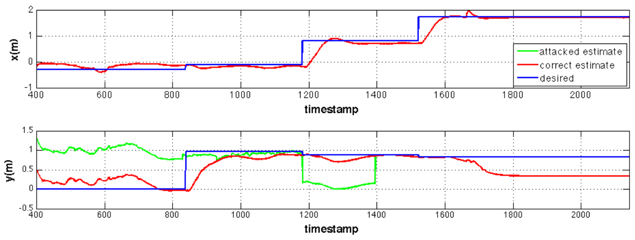


Fig. 8. Position estimate for the ROMDP quadrotor experiments.

VI. Conclusions and Future Work

In this paper we have presented a stochastic approach for optimal planning under malicious attack on sensors. Our ROMDP framework leverages the MDP theory to obtain an optimal policy (i.e., actions) to drive a vehicle that has not consistent observations of the world. We use redundancy in the sensor measurements considering sensor attacks hidden within the disturbance profile of the system. Resiliency against attacks is achieved by properly selecting the reward function thus avoiding actions that could hijack the vehicle to undesired regions of a state space. As demonstrated in the simulation and experimental results, this technique is especially advantageous for systems that are highly sensitive to environmental disturbances like aerial vehicles that are affected by weather perturbations. The main drawback of this approach is that, because it leverages MDP, it is computationally expensive, growing with the number of actions and the square of the number of states.

Future work will be centered on: i) studying the environment topology and performing reachability analysis, ii) using model checkers, like PRISM [18] to analyze ROMDPs and iii) analyzing different disturbance and attack models.

Acknowledgments

This research was supported in part by Global Research Laboratory Program (2013K1A1A2A02078326) through NRF, by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No. B0101-15-0557, Resilient Cyber-Physical Systems Research), and by NSF CNS-1505799 and the Intel-NSF Partnership for Cyber-Physical Systems Security and Privacy. This material is based in part on research sponsored by DARPA under agreement number FA8750-12-2-0247.

References

[1] "Spoofers Use Fake GPS Signals to Knock a Yacht Off Course. http://www.technologyreview.com/news/517686/spoofers-use-fake-gps-signals-to-knock-a-yacht-off-course."

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. of USENIX Security*, 2011.

[3] M. Zhu and S. Martinez, "On resilient consensus against replay attacks in operator-vehicle networks," in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 3553–3558.

[4] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[5] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proc. of the 6th ACM symposium on information, computer and communications security*. ACM, 2011, pp. 355–366.

[6] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[7] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. Pappas, "Robustness of attack-resilient state estimators," in *Proc. of the 5^{th} ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, Apr. 2014, pp. 163–174.

[8] N. Bezzo, J. Weimer, M. Pajic, O. Sokolsky, G. J. Pappas, and I. Lee, "Attack resilient state estimation for autonomous robotic systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*. IEEE, 2014, pp. 3692–3698.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[10] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, "Acting under uncertainty: Discrete bayesian models for mobile-robot navigation," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*, vol. 2. IEEE, 1996, pp. 963–972.

[11] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2009, vol. 414.

[12] M. L. Littman, "The witness algorithm: Solving partially observable markov decision processes," *Brown University, Providence, RI*, 1994.

[13] N. Bezzo, B. Griffin, P. Cruz, J. Donahue, R. Fierro, and J. Wood, "A cooperative heterogeneous mobile wireless mechatronic system," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 1, pp. 20–31, 2014.

[14] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[15] "AR.Drone 2.0 Parrot. http://ardrone2.parrot.com."

[16] "Robotic Operating System. http://www.ros.org."

[17] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, 2014.

[18] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for markov decision processes with co-safe ltl specifications," in *Proc. of IEEE International Conference on Intelligent Robotic Systems (IROS)*. IEEE, 2014.