Attack Resilient Cloud-based Task Assignment and Trajectory Planning for Multi-Robot System

Sheng Liu, Ziqiao Zhou
University of North Carolina at Chapel Hill

I. Introduction

CLOUD Robotics[1], [2] was proposed in 2010 to overcome the computation bottlenceck in onboard resource, especially when it comes to the collaborative robotic system, where cloud will act as the"remote brain" to provision task allocation and other critical planning for multiple robots. Planning problem for a large group of aerial robots, for example, is a typical application in cloud robotics system, which navigates N identical robots to M goal states. However, behind such system is the risk of misleading decision caused by abnormal robots[1], in which case, large team of aerial robots bear high risk of incorrect flying trajectory or even worse, collisions. Motivated by this problem, our project will incorporate security components into the target assignment and trajectory planning algorithm proposed in [3] for collaborative multi-robot system.

More specifically, the attackers considered in this paper can be categorized into two types: attacks on communication channels and attacks on sensors. By hijacking the communication channels (e.g. man-in-the-middle attack, faking identity) or compromising the robot sensors (e.g. GPS spoofing), the attacker can subvert the goal assignment to leave tasks unaccomplished, or make the cloud server to give the misleading instruction to lead the robot to the wrong path or collision with obstacles. Collisions will make damage to both the robots and the public property, which would cause a great number of economic loss. Moreover, if some critical tasks (like saving people or nuclear platform repair) are not achieved or delayed, the loss is countless.

To address the security problems described above, we propose several security mechanisms. Particularly, to deal with attacks on communication channels, we leverage the public key authentication scheme and symmetric encryption to guarantee the integrity and confidentiality of the message. Furthermore, a trust model [4] based on the previous history record is used to detect the misbehavior and mitigate the negative effect.

To demonstrate the effectiveness of the potential attack and the corresponding defense scheme, we first implemented the task assignment and trajectory planning algorithm proposed by [3] and showed the negative effect caused by the malicious robots. Then, we incorporated the trust model [4] into the algorithm and showed that this scheme can detect the attacker and mitigate the effect.

[1]In our project, both compromised and mis-functioning robots are defined to be abnormal robots

The contribution of this paper are as following:

- We described the potential threat model, and showed the destructive effect caused by abnormal robots in a system without security mechanism, by comparing the results between the optimal and the misleading trajectory;
- We implemented the job assignment and trajectory planning algorithm based on [3];
- We demonstrate the effectiveness of the trust model [4] using simulations through MATLAB.
- We simulated the MAV's local motion planner, sensors, and communication using gazebo and ROS C++;

The rest of this paper will be organized as such: Sec. II introduces some related works; Sec. III defines the notations and some models we used in our methods; Sec. IV describe the kernel functions and detailed simulation method for different components in our system; Sec. V provides our current results from simulation, which mainly focus on the threat model; Sec. VI makes a conclusion we found from our current progress and our following planning.

II. Related Work

Our paper is mainly based on two papers [3], [4]. The first paper proposed a task assignment and trajectory planning algorithm for multi-aerial-robot system and the second one presented a trust model based on the history records to deal with inaccurate information among software agents in large scale open systems.

Also, several papers[5] have studied the security problem of robotics. Bezzo [5] proposed a kalman filter to reduce the influence from the spoofing attack on robot sensors[6], although this method cannot be applied to a highly dynamic environment and it aimed at single robot. The security problem would be amplified in the multi-robot system. For example, the negative effect of one robot may spread to the whole network if no effective security mechanism exists. If there is no authentication or encryption scheme on the communication channels, the attacker can easily tamper or forge messages, sending misleading instructions. A trust model is illustrated in[7] to defend against the attacker in multi-robot patrolling. Currently, most related work in muti-robot system only focus on job assignment and path planning problem. Therefore, we want to raise concern of security issues in multi-robot system.

In the muti-robot system, motion planner is usually structured into two layers[8]: global motion planner and local motion planner. The global motion planner is always centralized, which is responsible to generate the trajectories and assign jobs for robots with the inputs of all robots' self-reported states, meanwhile local motion planner usually focusing on the shape maintain and collision avoidance. In our project, we simplify this system as a one-layer centralized controller performing task assignment and trajectory planning.

III. Problem Definition

A. Task Assignment and Trajectory Planning

Like the paper [3], we consider the problem of task assignment and trajectory planning for multi robots system with static obstacles. The cloud server will act as a centralized controller to make plans to accomplish the goal, while the (identical) robots follow the instructions and deliver reports (e.g. sensing data) as feedback. Specifically, the goal of the cloud server is to find the optimal solution to assign M tasks to M of N robots (assume each robot can only finish one task) and make motion planning for each robot. Here, the optimal plan means to minimize the maximum cost of all robots (e.g. overall traveling distance), meanwhile keep each trajectory collision-free.

The basic idea of our motion planning is to concurrently search optimal trajectory for each robot, meanwhile using different offset times to deal with the collision problem between each robot. The algorithm can be divided into two steps. At the first phase, we leverage PRM (Probabilistic Roadmap Method) and Hungarian algorithm [9] to find the optimal task assignment and trajectory for each robot. Then, in order to avoid collision between two robots sharing the same path, we set different offset times for each robot to start running. We will elaborate our algorithm below.

Let $\mathcal{Z_M}$ represents the set of integers from 1 to M, i.e. $\mathcal{Z_M} = \{1, 2, ..., M\}$. The position of robot i at time point t is denoted as $x_i(t)$, where $i \in \mathcal{Z_N}$ is the robot ID. Moreover, we define $x_i(0) = s_i$ as the initial state of robot i and g_j as the position of goal j, where $j \in \mathcal{Z_M}$. The goal of our system is to find a mapping function $\phi : \mathcal{Z_N} \mapsto \mathcal{Z_M} \bigcup 0$ to indicate the assignment of each task to one of N robots. Specifically,

$$\phi_i = \begin{cases} j & \text{, if robot i is assigned to goal j} \\ 0 & \text{, otherwise} \end{cases}$$

Also, we construct a direct graph $G = (V, E)$ to represent the probabilistic roadmap, where the node set V represents the group of milestones and each $e_{ij} \in E$ represents a collision-free path segment. Then, we can represent each path in graph G as an ordered list of edges, i.e. $P(v_i, v_j) = \{e_{ik}, e_{kl}, ..., e_{nj}\}$. The cost of this path can be defined as:

$$||P(v_i, v_j)|| = \sum_{e_{kl} \in P(v_i, v_j)} ||e_{kl}|| \tag{1}$$

where, the cost function can be the distance of each path. The optimal path is the one with the minimal overall costs:

$$P^*(v_i, v_j) = arg \min_{P(v_i, v_j)} ||P(v_i, v_j)||$$

The whole trajectory $\gamma_{ij}(t)$ can be constructed by concatenation of segments:

$$\gamma_{ij}(t) = \begin{cases} \tau_{ik}(t), & t \in [t_i, t_k], \\ \tau_{kl}(t), & t \in [t_k, t_l], \\ ..., & ... \\ \tau_{nj}(t), & t \in [t_n, t_j], \end{cases}$$

We can use PRM to search the optimal collision-free path for each robot independently. Then, after constructing a cost matrix for each robot-goal pair, Hungarian algorithm [9] is leveraged to find the task assignments with the minimal overall costs. However, the solution found is not guaranteed to satisfy the collision-free requirements, as some robots may share the same path segment. To deal with this issue, we first induce an ordering among all the robots as follows:

$$s_i \in P^*(s_j, g_{\phi_j^*}) \implies \Psi_i \prec \Psi_j \quad \forall i \neq j \tag{2}$$

$$g_i \in P^*(s_j, g_{\phi_j^*}) \implies \Psi_i \succ \Psi_j \quad \forall i \neq j \tag{3}$$

These equations can be interpreted as, when the start point of robot i is in the path of robot j, robot i has lower priority than robot j (to let robot j arrive first), while, when the goal point of robot i is in the path of robot j, robot i has higher priority than robot j (to let robot i arrive first). By using these relations, we can build a partial ordering among all the robots, which can be represented by a mapping function Ψ: $\mathcal{Z_N} \mapsto \mathcal{Z_N}$ where Ψ_i indicates the priority of robot i.

With ordering of each robot, we can set different offset times for each robot to initiate. Particularly, we first set time offset $t_{\Psi_{max}}$ of the robot with the highest priority as $t_{\Psi_{max}} = 0$. Then, we iteratively compute the offset time t_{Ψ_i} of the next robot i such that its trajectory will not collide with robots j with higher priorities, i.e. the minimal value t_{Ψ_i} satisfying the condition $\forall t, j, x_i(t - t_{\Psi_i}) \neq x_j(t - t_{\Psi_j})$. This method can help us find collision-free paths by separating each robot in timeline.

B. Security Problem

The task assignment and trajectory planning algorithm can be directly applied to the cloud system, by viewing the cloud server as a centralized planner. However, this optimal algorithm only works well under the assumption that all the robots (tenants) are honest and behave correctly. This assumption may not hold if the robots are compromised or the communication channels are hijacked by the adversary. Specifically, the attacker would upload some fake sensing data or give misleading instruction to the robots. In our scenario, this would lead to tasks not accomplished or even worse, the collision of robots. In this paper, we mainly consider two kinds of attackers:

Attack on Communication Channel:. this kind of adversary can be further categorized into three types: Man-in-the-middle attack (the attacker can eavesdrop, tamper or replay messages between cloud server and robots), Sybil attack where the attacker pretends to be multiple legal robots in the network, and fraud where the adversary feigns to be an existing legal robot. Many research works have studied these kinds of attacks (though may not in the robotics field) and demonstrated that the strong authentication mechanism and encrypted communication channel can mitigate or thwart this attack.

To build a strong authentication scheme, we will use public key authentication. We let pk denote the public key which is published to other users and sk denote the private key which is kept secret. Each client has a unique public-private key pair. The private key sk can be used to generate signatures which cannot be forged by others who do not have a knowledge of sk. However, anyone who has the corresponding public key pk can verify the signature. Specifically, we leverage OpenSSL(TLS) to build the public key infrastructure and the signature generated can be represented as $E_{TLS}(sk, msg||nonce)$. Here, $nonce$ is a random number concatenated to the end of the message to prevent the replay attack. The verification process can be proceeded as $msg||nonce = D_{TLS}(pk, E_{rsa}(sk, msg||nonce))$

Next, to encrypt the communication channel, we use symmetric-key encryption (the key used to encrypt the message is the same as the one used to decrypt the message), since the symmetric encryption is much faster than asymmetric one. With the communication channels encrypted, the attacker can no longer forge the message of legal robots. Particularly, we use Advanced Encryption Standard (AES) scheme and the decrytion process can be performed as $msg = D_{AES}(key, E_{AES}(key, msg))$. Furthermore, for message integrity, we use keyed-hash message authentication code (HMAC) which is an irreversible function.

Attack on Sensors:. This mainly includes the typical cyber physical attacks on sensors, such as GPS spoofing̃citebezzo2014attack. Instead of hijacking the communication channel, this attack only compromise the sensors of the robots and mislead the robots to send incorrect sensing data. For example, the attacker can send forged robot position data to the cloud server to impose a negative effect on the motion planning. To deal with this issue, we leverage a trust model based on the history of each robot to differentiate the malicious robot and the honest ones. We will propose the scheme in the next section.

C. Trust Model

To detect the unreliable robot, we use the trust and confidence model based on history records proposed by [4]. Let $\mathcal{F}_i$ be the probability that robot i can fulfills its task, and $\mathcal{O}_i^{1:t}$ be the history records for robot i collected during time period $[0, t]$ in cloud side. Then we can use the expectation for $\mathcal{F}_i$ based on $\mathcal{O}_i^{1:t}$ to represent the level of trust $\mathcal{T}_i$ for robot i:

$$\mathcal{T}_i = E[\mathcal{F}|\mathcal{O}_i^{1:t}] \tag{4}$$

To be more specific, the outcome $\mathcal{O}_i^{1:t}$ can be whether the robot accomplished the task or running on the correct path. Since a binary result can be used to represent whether the robot fulfills the task, we can leverage the Beta Distribution to model the probability of this variable. The probability density function f of Beta Distribution can be formulated as:

$$f(\mathcal{F}_i|\alpha, \beta) = \frac{(\mathcal{F}_i)^{\alpha-1}(1-\mathcal{F}_i)^{\beta-1}}{\int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\mathrm{d}x}, \text{where } \alpha, \beta > 0 \tag{5}$$

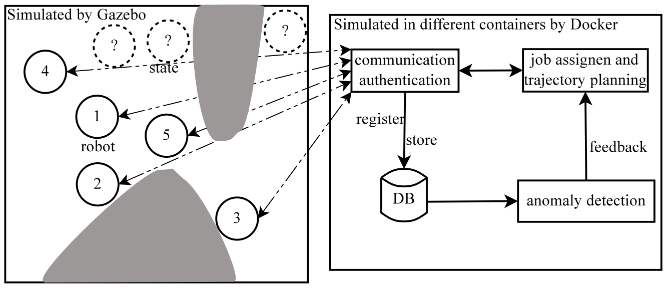

Simulated by Gazebo
?
?
?
4
state
1
robot
5
2
3
Simulated in different containers by Docker
communication authentication
job assignen and trajectory planning
register
store
feedback
DB
anomaly detection

Fig. 1. System Overview

where, α and β indicate the number of successful outcomes and unsuccessful outcomes, respectively. In the initial case where $\alpha = \beta = 1$, the distribution is uniform, making success a fifty-percent chance. By receiving new report, the parameters α and β can be updated according to whether it is a successful result. Then, we can rewrite the Eqn.(4) to:

$$\mathcal{T}_i = E[\mathcal{F}|\alpha, \beta] = \frac{\alpha}{\alpha+\beta} \tag{6}$$

However, as the history records accumulated, the prediction tends to be more accurate. Therefore, we need another parameter $\mathcal{C}_i$ for the robot's $\mathcal{T}_i$ to measure the confidence in the value of trust. Specifically, we define $\mathcal{C}_i$ as the proportion of the probability distribution that approximate $\mathcal{T}_i$ (or bounded by $[\mathcal{T}_i - \epsilon, \mathcal{T}_i + \epsilon]$).

$$\mathcal{C}_i = \frac{\int_{\mathcal{T}_i-\epsilon}^{\mathcal{T}_i+\epsilon} y^{\alpha-1}(1-y)^{\beta-1}\mathrm{d}y}{\int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\mathrm{d}x} \tag{7}$$

Then, with a high $\mathcal{C}_i$ (exceeding a threshold) and low $\mathcal{T}_i$, we can determine the robot i to be untrusted.

IV. Method

As it is shown in Fig. 1, our proposed system consists four components: message encryption module, motion planning solver, history record database and anomaly detection module. Message encryption module is deployed in the session layer to encrypt and decrypt the message, verify signatures and check the packet integrity. Motion planning solver executes the goal assignment and trajectory planning algorithm to select the appropriate robots and its corresponding path to accomplish the tasks. History record database is used to store reports from robots, which can be the position of each robot, the figure taken from its camera, or other kinds of data from sensors. Through this data, the cloud server can infer the correctness of each robot behavior (by comparing the reported position with its planning path, or analyze the image from its camera). This information can be further leveraged by the anomaly detection module to update the trust parameter of each robot. Once the detector has a high confidence that one robot may be malicious, an alert would be sent to the motion planner solver. The task solver will redistribute the task assigned to the suspicious robots and maintain the robot ID in the blacklist. Note that, this is the system architecture on cloud server. The remote robots only require message encryption module.

TABLE I
ROBOT TABLE

Field	Type	Comments
rid	int	PRI
name	text	UNIQUE
certpath	text	file path to store certificate
move	int	references the ts in Trajectory table
new	int	references the ts in State table
reputation	int	represents the reputation for each robots

TABLE II
ROBOT STATE TABLE

Field	Type	Comments
sid	int	PRI, auto_increment
tid	int	task ID
rid	int	robot ID
ts	int	timestamp
x	double	x coordinate
y	double	y coordinate
z	double	z coordinate
neighbor	text	$(rid_1, rid_2, .. rid_k)$
time	datetime	default now()

A. Database Design

To detect anomaly, we need a database to store the history record of each robot. The schema is shown in Table I, Table II, Table III, Table IV. Table I stores the information of each robot (e.g. robot ID, name) and Table II represents the current state of each robot (e.g. position coordinate, neighbor robot IDs). In Table III and Table IV, we save the result of trajectory planning and task assignment, respectively. We use mysql++ to build our database.

B. Secure Communication

This component is designed to prevent network attacks defined in Sec. III-B. As described in the previous section, in order to provide authentication and avoid tampering, both cloud server and robots will generate the message signature using other side's public key (i.e. $E(pk_{rev}, msg||nounce)$. Then, each message should be encrypted using the sender's key (i.e. $E(key, msg)$) to guarantee the confidentiality and thwart eavesdropping. Since the attacker does not have access to the private key, he/she cannot forge messages or decrypt the packets. Also, we need to pad a timestamp in each packet to avoid replay attack (the receiver will drop the outdated packets). In fact, we can use timestamp as $nounce$ for convenience.

The key exchange procedure should be booted at the initial phase, to prevent the attacker from joining the tasks. In our system, we have a specific robots pool which includes all `known` robots predefined by our administrator. This pool stores the certificate of each `known` robot. However, not all `known` robots will join a specific task (some robots may be online, while others offline). Thus, we define a set of `legal` robots to be the group of robots who will join the task. Before the task starts, our system requires the administrator to register all `legal` robots first.

On the client side, the robots should send its certificate, task ID to the cloud to register the task. After receiving the requests, the communication module needs to verify the certificate, and if correct, assign a unique robotID to the robot which will be used as identity in the subsequent communication.

TABLE III
TRAJECTORY TABLE

Field	Type	Comments
rid	int	PRI
time	datetime	PRI
ts	int	timestamp
waypoints_str	text	path= $(x_1, y_1), ..., (x_k, y_k)$

TABLE IV
TASK TABLE

Field	Type	Comments
tid	int	PRI task id
priority	int	priority of each task
x	double	x coordinate
y	double	y coordinate
z	double	z coordinate
rid	int	0 or robot rid who finished this task
waypoints_str	text	path= $(x_1, y_1), ..., (x_k, y_k)$

C. Job Assignment and Trajectory Planning

This module is designed to find the optimal solution of task assignment and trajectory planning. As elaborated above, the problem for job assignment is to find the optimal ϕ:

$$\phi^* = arg \min_{\phi} (\sum_{i \in \mathcal{Z_N}} ||\gamma(s_i, g_{\phi_i})||^p)^{\frac{1}{p}} \tag{8}$$

which can be solved by the Hungarian algorithm [9]. To avoid the multi-agent collision, [3] proposed to simply assign different time offsets to robots based on their priority. We describe the algorithm as follows: Note that, our security modules (message encryption module and anomaly detection module) are independent from the cloud application (motion planning). Here, we use motion planning to give a typical example about how our security mechanism works. Our scheme can be extended to other applications.

Algorithm 1: Goal Assignment and Trajectory Planning

```
Data: G, $s$, $g$
Result: trajectories $\gamma_{\phi^*}(t - \hat{t}_\Psi)$
for $i \in \mathcal{Z_N}$ do
    for $j \in \mathcal{Z_M}$ do
        Compute the optimal trajectory $\gamma_{ij}(t)$ from $s_i$ to $g_j$;
    end
    Compute the optimal assignment $\phi^*$ based on (8);
    Assign priority $\Psi$ to different robots based on (3);
    for $i \in \mathcal{Z_N}$ do
        Compute time offset based on priority;
    end
end
```

D. Anomaly Detection

In this paper, we build an anomaly detector based on the trust model proposed by [4]. Since a binary result is enough

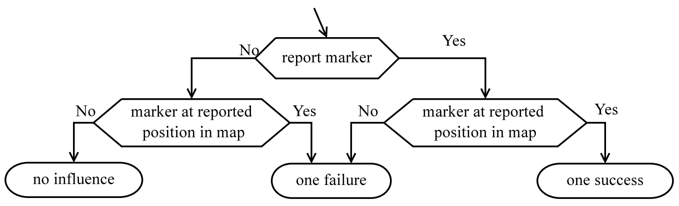

report marker
No
Yes
marker at reported position in map
No
Yes
marker at reported position in map
No
Yes
no influence
one failure
one success

Fig. 2. Logic of processing self reports

to indicate whether the attacker would perform the correct behavior, we leverage the Beta Distribution to model the probability of honest robot behavior. The probabilistic density function can be formulated using Eqn.(5) and α represents the number of honest behaviors, while β indicates the malicious one. We update the value of α and β according to the reports. These reports can be sensing data from robots (e.g. camera), or extra information from other sources (e.g. from surveillance cameras on site). Specifically, the robots can periodically upload its camera image to the cloud server, and the server can process the image to determine whether it is running on the correct path.

Some of the uploaded reports may be forged or incorrect. But, since it is intractable for the attacker to consume so many resources to compromise a large number of robots in the network, we think the server can obtain relatively accurate results by aggregating all the reports. As we mentioned in Sec. III-C, we will build a trust and confidence parameters for each robot, according to the value of α and β. If the confidence value is very high and the trust is very low (by setting a threshold), we can determine the robot is suspicious. Below, we give a concrete example of how the server make an analysis of the position data uploaded by the robots.

Self Updating. To check whether the position reported from robot itself is correct or not, we should first judge if the reported position is consistent with the trajectory assigned by cloud. However, this is not enough for the sensor-compromised robot, because the attacker can send fake information to the cloud. Therefore, we need another observation to confirm the judgment. Other than GPS sensors, we can also use the robot camera to detect the context where robot is located. By comparing the image with the real context given the reported position, we can judge the correctness of the robot position. Specifically, our experiments randomly place several markers on the ground which can be captured by the camera, and let the robots report whether the camera detected that marker or not together with the location information to the cloud.

Neighbor Updating. Besides the report from the evaluated robot itself, the report from other robots near it can also be used to generate $\mathcal{O}$. The interesting case is that when there are several honest neighbor robots surrounding the malicious one. In that case, the attacker can be detected by voting from its neighbors (denoted by $\mathcal{N}$) within a visible or sensible distance (denoted by $\sqsubseteq\mathcal{D}$). To evaluate robot i, we can utilize the neighborhood information sensored by robot j, where `distance` $(l_i,l_j) < \sqsubseteq\mathcal{D}$, for all $j \in \mathcal{N}$. As it is shown in Fig. 3, only two neighbor robots 1 and 2 reported that they can

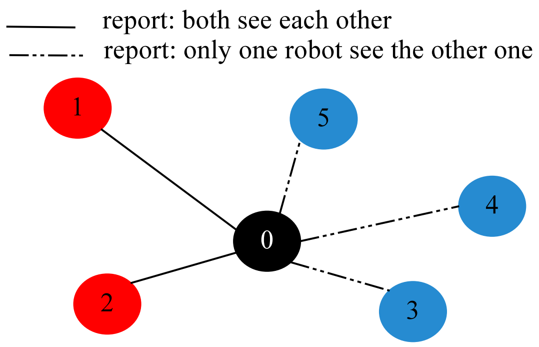

report: both see each other
report: only one robot see the other one
1
5
4
0
2
3

Fig. 3. Example of neighbors' reporting

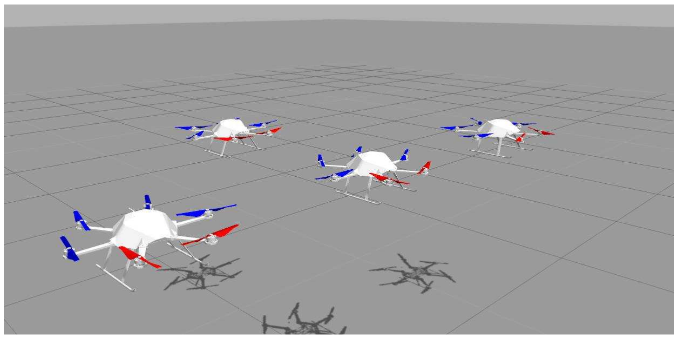


Fig. 4. Four hexacopter hovering in an empty world

see the robot 0, while other three didn't agree. As long as the honest robots outnumber the malicious ones, the aggregation should be good.

Note that, we do set the marker on the map in our simulation. However, with the time limitation, we have not accomplished the report aggregation part, which will be left to the future work.

E. Simulation

To investivate our design, we used the framework of RotorS[10], which provides hexacopter models in Gazebo integrated with ROS, as shown in Fig. 4. For different components, we used different programming language. In our simulation, specifically, all functions within robots and the communication module are coded through C++, while the centralized motion planning and the detector are implemented using MATLAB. We set a communication node as a agent for each robot, which is responsible for sending and receiving message. After the key initialization procedure including the registration, this communication node will periodically send GPS, velocity, and nearest distance from other robots to cloud. Currently, the robot will send registration request to cloud once it is started. Then the connection will be maintained until the cloud explicitly sends a task ending signal. On the cloud side, a background C++ process will listen to port 8888, receive information, pass the information to the database and the MATLAB process which is responsible for the job assignment and trajectories planning as well as the anomaly detection, and send results to robots. A problem to be noted in practice is the synchronization problem for different robots. Although

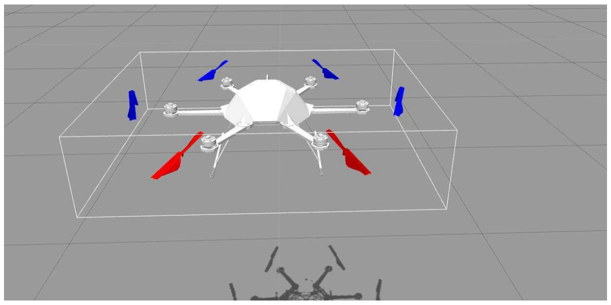


Fig. 5. Hexacopters Model from Rotors simulator

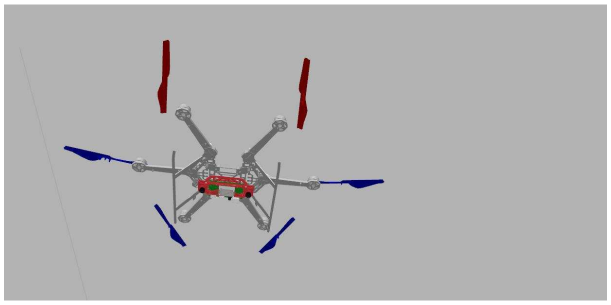


Fig. 6. Robot Model with camera and odometry sensors

the robots is assumed to send information with the same rate, it is not guaranteed that the cloud will receive all packets concurrently. Thus, we will only update the planning for the robots who send messages within a predefined interval.

V. Result

We used the gazebo to simulate the workspace and ROS to simulate the driver for the aerial robot. Each aerial robot is a hexacopter equipped with a camera and a GPS sensor as shown in figreffig:robot. The camera in our project is only used to check the marker we placed in our map, and the GPS is used to report the position.

A. Goal Assignment and Planning

Our experiments will be conducted using the simulation. The scenario we used for the Goal Assignment and Planning is shown in Fig. 7. Four blue asterisks represent the hexacopters, two red pentagrams represents task goals. The green lines in Fig. 7(b) shows the optimal trajectories with accurate data input for global planner. The green linesFig. 7(c) shows the optimal trajectories according to the inputs where the reported position of Robot 4 is (40,9)(red asterisks) while the real is (45,4), where the blue line is the path from real position to the next waypoint in the view of Robot 4. Comparing these two figures, we found that the misleading trajectories will assign the task from Robot 3 to Robot 4, which increase the real costs, and that the global plan will give a potential collision-existed path for Robot 4.

B. Anomaly Detection

As it is mentioned in Sec. III-C, we used the Beta distribution to build our trust and confidence model. To evaluate

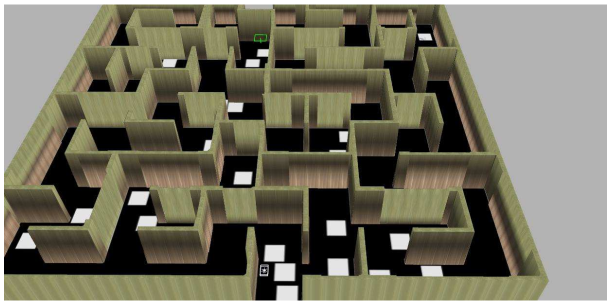


Fig. 8. 3D map with road marker in Gazebo

the influence of amount noise to our detection model, we choose three different noise sources. The first setting is to differentiate an aggressive attacker (introduce failure check with the probability of 0.7) and the normal robot (normal sensor may has some Gaussian noise up to 10% noise), in which case we can detect not much high $\mathcal{T}$ (for example 0.4) with a relatively stable $\mathcal{C}$ (0.7). The second one is when attacker try to pretend to be a good robot by decrease its frequency of malicious behaviors, for example do a malicious behavior with probability of 0.4(with $\mathcal{T}$ bound 0.6, and confidence bound 0.8). The detection may also get benefit if the information (like GPS or valocity) from normal robots are accurate enough (0.025 noise), where we can use a higher $\mathcal{T}$(0.8) bound and a moderate $\mathcal{C}$ bound even when the attacker is conservative. The results are shown in Fig. V-B, which include the average trust, confidence, and detection accuracy. From those figures, confidence will be accumulated with the increasement of history.

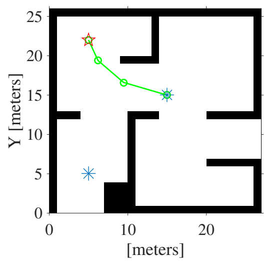
VI. Conlusions and Future Work

In this paper, We described the potential threat model and showed the negative effect caused by malicious robots. Also, we implemented the task assignment and trajectory planning algorithm using MATLAB, and the communication encryption scheme using OpenSSL. Meanwhile, we build a detector based on trust model using MATLAB, and try to use Gazebo to simulate the multi-robot motion planning scenarios.

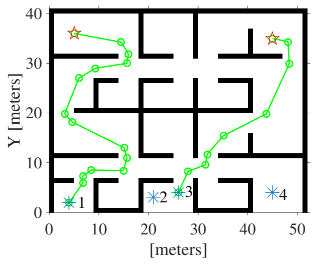
The future work includes several aspects. The first one is to extend the 2D algorithm to 3D world, as currently our motion planning algorithm is implemented only in 2D world. Next, we want to evaluate the effectiveness of our trust model in a dynamic environment. We will finish the program for report aggregation (our map maker process) and also figure out how to efficiently and reliably obtain an honest historic record. We also would like to incorporate the security mechanism into other applications other than motion planning.

VII. Acknowledgement

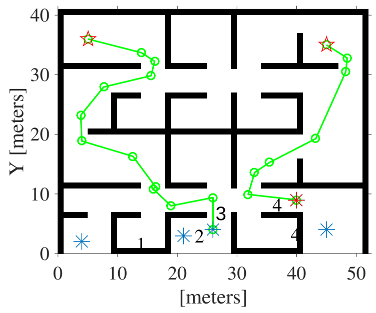
The libraries used in our project includes the openssl, mysql++, apache thrift, boost, some of libraries (cv_bridge, image_transport, sensor_msg, geometry_msg, mav_msg, nav_msg, hector_gazebo_plugins) from ros-indigo, and the framework and some resources from rotors_simulator for the robot simulation. We also used the Robotics, Image



(a) A simple Trajectories

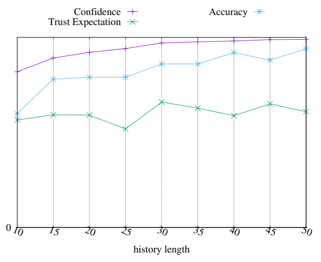


(b) Correct Optimal Trajectories

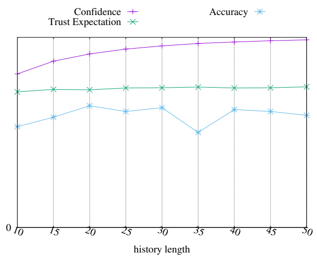


(c) Misleading Trajectories

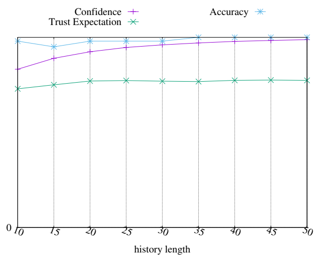
Fig. 7. Goal Assignment and Planning



(a) normal noise=0.1 and attacker noise=0.7



(b) normal noise=0.1 and attacker noise=0.4



(c) normal noise=0.05 and attacker noise=0.4

TABLE V
TASKS OVERVIEW

Task	Leader	Helper
Develop the topic of our prject	Sheng and Ziqiao	–
Motion planning and task assignment (MATLAB)	Sheng	Ziqiao
Implement secured channel (openssl, C++)	Ziqiao	Sheng
Robot's control nodes (Gazebo and ROS C++)	Ziqiao	Sheng
Design the Trust and confidence model	Sheng	Ziqiao
Implement database (mysql) and assemble modules	Ziqiao	Sheng
Implement Detection Module (MATLAB)	Sheng	Ziqiao
Paper Writing	Sheng and Ziqiao	–
Presentation	Sheng and Ziqiao	–

Processing and Database tool in MATLAB, as well as a Munkres algorithm code library.

The effort from Sheng and Ziqiao is listed in Table V, where the helper actively participates into the tasks directed by the leader.

REFERENCES

[1] J. Kuffner, "Cloud enabled humanoid robots," *Humanoids: Whats next*, 2010.

[2] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 398–409, 2015.

[3] M. Turpin, M. Kartik, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[4] W. L. Teacy, J. Patel, N. R. Jennings, and M. Luck, "Travos: Trust and reputation in the context of inaccurate information sources," *Autonomous Agents and Multi-Agent Systems*, vol. 12, no. 2, pp. 183–198, 2006.

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

[6] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via gps spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[7] C. Pippin and H. Christensen, "Trust modeling in multi-robot patrolling," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 59–66.

[8] J. Alonso-Mora, R. Knepper, R. Siegwart, and D. Rus, "Local motion planning for collaborative multi-robot manipulation of deformable objects," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5495–5502.

[9] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[10] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotorsa modular gazebo mav simulator framework," in *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.