# Path Planning for a UAV by Considering Motion Model Uncertainty

Hossein Sheikhi Darani, Ali Noormohammadi-Asl and Hamid D. Taghirad
Advanced Robotics and Automated System (ARAS), Industrial Control Center of Excellence (ICEE)
Faculty of Electrical Engineering, K. N. Toosi University of Technology, Tehran, Iran
Email: h.sheikhi, a.noormohammadi@ieee.org, taghirad@kntu.ac.ir

***Abstract*—The primary purpose of path planning for unmanned aerial vehicles (UAVs), which is a necessary prerequisite toward an autonomous UAV, is to guide the robot to the predefined target while the chosen path is optimized. This paper addresses the problem of path planning for an unmanned aerial vehicle in a 2D indoor environment, considering motion uncertainty. To cope with this challenge, the problem of motion planning is formulated in three parts. A vision-based extended Kalman Filter (EKF) is used to localize the UAV in the unstructured environment. To overcome motion uncertainty, the problem is modeled as a Markov decision problem (MDP). Finally, a novel dynamic feedback linearization based switching controller is proposed for point-to-point motion. Simulation and experimental results are given to show the effectiveness of the proposed path planning method in practice.**

## I. Introduction

Autonomous Unmanned Aerial Vehicles (UAV) have been utilized extensively in a variety of applications ranging from search and rescue operations, aerial surveillance, and mapping. On the other hand, without any doubt, path planning is a fundamental part of an autonomous robot. In a general sense, path planning means finding a reachable and optimized path from the robot's initial configuration to the target configuration [1]. What is meant by a reachable path is generally a collision-free path, while optimized means a pathway that robot can traverse it in a time-optimal manner.

Configuration space discretization is the starting point in the path planning problem [2]. Cell decomposition and artificial potential field are representative of two conventional approaches developed in the literature [3], [4]. Recently some sampling-based methods like probabilistic roadmap (PRM) [5] and rapidly-exploring random tree (RRT) [6] have been successfully employed in high dimensional configuration spaces. In this paper, due to the probabilistically completeness nature of the sampling-based methods, the PRM is used to initially build a road map.

In the absence of uncertainties in the environment, many methods are developed to give the fastest path based on graph-based algorithms [7]–[9]. In reference [10], some prevalent and modern algorithms like $A^*$, $BasicTheta^*$, $Phi^*$ and $JPS(A^*)$ are subtly compared. However, in real implementations, finding the fastest path is quite sensitive to the uncertainties. In other words, due to the uncertain nature of the robot's motion, the robot may diverge from its planned path, which will lead to a collision, especially in narrow passages. In this paper, the main focus is to find the fastest path while reducing the traversing risk. Therefore, by considering uncertainty in robot motion as the most significant uncertainty in this problem, the higher probability of success for arriving at the desired goal is achieved with the expense of a more sophisticated algorithm.

Markov Decision Process (MDP) is a widely used approach to deal with uncertainty based path planning. Alterovitz et al., in [11] used an MDP based path planning for a medical robot while the robot states were accurately given. Reference [12] have dealt with uncertainty in the robot's motion and also robot sensing by using FIRM (Sampling-based Feedback Motion Planning Under Motion Uncertainty and Imperfect Measurements) and a partially observable Markov decision process (POMDP) approach. While in this paper, it is assumed that some information about the environment's map is given, an Extended Kalman Filter (EKF) is adapted to localize the robot in the environment, whose observation input is based on the robot's onboard camera and the markers mounted on the walls [13]. While results show state estimation of EKF is not precise, it is shown in practice that relying on EKF localization is accurate enough for state estimation in our path planning problem.

Designing a proper controller is another essential part of the implementation of robot path planning. The motion control of the mobile robot is concerned with two major challenges, path tracking control, and posture stabilization. Reference [14] shows that the posture stabilization of a wheeled mobile robot is required at least a discontinuous or time-varying controller, and the linear controller would not perform well. In order to overcome the posture stabilization problem, many controllers have been proposed. References [15]–[17] use switching and variable structure controllers. In [14], [18], [19] nonlinear based controllers such as backstepping and feedback linearized controller are adapted. However, some of these controllers suffer from poor transient performance and slow convergence rate near the goal position. References [14], [20] use a dynamic feedback linearized approach for both trajectory following and posture stabilization problem. While the aforementioned problems are suitably tackled, However, it shows some drawbacks in stabilizing the robot in the desired heading angle, which will be elaborated in more depth in the next sections. The heading angle stabilization problem is more significant in the performance of our implementation, where a vision-based

localization approach is used to estimate the robot's position and orientation. The uncertainty in the localization of the robot, especially in robot's orientation, causes the controller poor performance or failure in convergence to the destination point. Therefore, we proposed a switching based controller to cope with this problem.

The paper is organized as follows. First, some preliminaries about MDP and the switching-controller are given in the next section. Then, the formulation of the proposed algorithm is presented, and finally, the results of simulation and a real implementation of the proposed algorithm is given and compared to $A^*$ implementation in section four, Finally the concluding remarks are presented in the final part of the paper.

## II. Preliminaries

### A. short review on MDP Problem

In robotics literature, Markov Decision Process (MDP) is widely used to handle the uncertainties. An MDP is fully defined by a 4-tuple $< S, A, T, R >$ with the following description.

- $S$ is a finite set of the robot states thanks to the PRM
- $A$ is a finite set of the robot actions at each state that robot is able to take for state transition
- $T$ is the state transition function $S \times A \times S \rightarrow [0, 1]$. $T(s, a, s')$ defines the success probability when the robots performs action $a$ for going from state $s$ to state $s'$
- $R$ is the reward function $S \times A \times S \rightarrow \mathbb{R}$. $R(s, a, s')$ defines the robot reward when action $a$ is taken for transition form state $s$ to state $s'$.

Finding the *optimal policy* $\pi^*$ is the MDP desired objective. In this paper *Bellman Equation* is used to find the $\pi^*$ as follows:

$$V^{\pi^*}(s) = \underset{a \in A}{max} \sum_{s' \in s} T(s,a,s')\big(R(s,a,s') + \gamma V^{\pi^*}(s')\big) \quad (1)$$

$$\pi^*(s) = argmax V^{\pi^*}(s) \quad (2)$$

These equations are solved by policy iteration method, which is based on a dynamic programming approach [21].

### B. short review on EKF localization

Precise localization is another part of a path planning problem. In this paper, we assume that the robot motion in vertical and the horizontal plane are decoupled. As mentioned before, this paper addresses the path planning of the UAV solely in the horizontal plane. To overcome this 2D localization problem, we consider a Gaussian noise for the system and implement an EKF localization algorithm.

**Motion model:** We consider the robot's motion model by unicycle kinematics:

$$X_{k+1} = f\big(X_k, u_k, w_k\big) = \begin{bmatrix} x_k + \big(V_k \delta t + n_v \sqrt{\delta_t}\big) \cos\theta_k \\ y_k + \big(V_k \delta t + n_v \sqrt{\delta_t}\big) \sin\theta_k \\ \theta_k + w_k \delta_t + n_w \sqrt{\delta_t} \end{bmatrix} \quad (3)$$

where $X_k = \big(x_k, y_k, \theta_k\big)$ are state variables. $\big(x_k, y_k\big)$ are the coordinates of the UAV in the horizontal plane, and $\theta_k$ is the robot heading angle in k-th step. The vector $u_k = \big(V_k, w_k\big)^T$ is the system's control input where $V_k, w_k$ are robots linear and angular velocity, respectively.

Vector $w_k = \big(n_v, n_w\big)^T \sim \mathcal{N}\big(0, Q_k\big)$ is the robot motion noise caused by factors such as error in time intervals, error in robot's model and external disturbances. In addition to the the uniform noise in the motion, there is a noise depending on the motion command signal strength. Therefore, we can write the motion noise covariance as follows:

$$Q_k = \begin{bmatrix} \big(\eta V_k + \sigma_b^v\big)^2 & 0 \\ 0 & \big(\eta w_k + \sigma_b^w\big)^2 \end{bmatrix} \quad (4)$$

in which, $\eta, \sigma_b^v, \sigma_b^w$ are constant values.

**Sensor model:** In this paper, we used ArUco library provided in OpenCV [22], [23], to generate black-white patterns, which each of them has a unique id. By detecting these markers, we can obtain the relative range and bearing of UAV to each marker. If we consider the marker $j$ as ${}^jL$ in robot's environment coordinates, the robot's observation model will be as follows:

$${}^j z_k = \big[||{}^j d_k||, atan2\big({}^j d_{2_k}, {}^j d_{1_k}\big) - \theta\big]^T + j_v, j_v \sim \mathcal{N}\big(0, {}^j R\big) \quad (5)$$

where ${}^j d_k = \big[{}^j d_{1_k}, {}^j d_{2_k}\big]^T := \big[x_k, y_k\big]^T - L_j$.

Observation noise for j-th marker denoted by ${}^j v$ is proportional to the distance of the landmark and the incident angle between the camera and landmark:

$${}^j R_k = diag\Bigg(\big(\eta_{r_d} \|{}^j d_k\| + \eta_{r_\phi} |\phi_k| + \sigma_b^r\big)^2, \big(\eta_{\theta_d} \|{}^j d_k\| + \eta_{\theta_\phi} |\phi_k| + \sigma_b^\theta\big)^2\Bigg) \quad (6)$$

in which, all of the parameters $\eta_{r_d}, \eta_{\theta_d}, \eta_{r_\phi}, \eta_{\theta_\phi}, \sigma_b^r, \sigma_b^\theta$ are constant values. We assume that the robot observes $r$ marker $\{L_{i_1}, L_{i_2}, ..., L_{i_r}\}$ in one time interval, thus combination of all system observations will be $z = \big[{}^{i_1} z^T, {}^{i_2} z^T, ..., {}^{i_r} z^T\big]$.

According to independence of observations, observation model may be written as $z = h\big(x\big) + v$ where, $v = \big[{}^{i_1} v^T, ..., {}^{i_r} v^T\big]^T$ is observations noise vector $v \sim \mathcal{N}\big(0, R\big)$ which is obtained from $R = diag\big({}^{i_1} R, ..., {}^{i_r} R\big)$.

### C. Switching Controller

Switching controller we proposed in this paper is based on Dynamic Feedback Linearization (DFL) proposed [14] and a Lyapunov-based controller. DFL controller is able to stabilize a unicycle robot in an ideal position, but in experiments, we faced with a problem in stabilizing the robot's heading angle at the end of the path and when the robot is near the destination coordinates. Heading angle stabilization is more significant when we use this controller in the path planning problem where the robot should traverse through several positions to reach the goal point, and the heading angle error in one of these locations may cause failure in the total path planning.

In order to cope with this problem, the DFL based switching controller is adopted as follows to steer the initial position, $\hat{X}_k^+ = \left(\hat{x}_k^+, \hat{y}_k^+, \hat{\theta}_k^+\right)$, to the target position $\mathbf{v} = (v_x, v_y, v_\theta)$. Namely the proposed variable structure controller is as follows:

$$\begin{cases} \text{DFL Controller} & \text{if} \quad e_p \geq \varepsilon \\ \text{Heading-Angle Controller} & \text{if} \quad e_p < \varepsilon \end{cases} \tag{7}$$

where $e_p$ is defined as $e_p = \sqrt{\left(\hat{x}_k^+ - v_x\right)^2 + \left(\hat{y}_k^+ - v_y\right)^2}$ and $\varepsilon$ is an ideal positive value depending on the robot and the stopping region radius.

The DFL controller stabilizes the system at the origin. In order to stabilize the system in an arbitrary state, we have used the following transformation.

$$\begin{aligned} \begin{pmatrix} \breve{x}_k^+ \\ \breve{y}_k^+ \end{pmatrix} &= \begin{pmatrix} \cos v_\theta & -\sin v_\theta \\ \sin v_\theta & \cos v_\theta \end{pmatrix}^{-1} \begin{pmatrix} \hat{x}_k^+ - v_x \\ \hat{y}_k^+ - v_y \end{pmatrix} \\ \breve{\theta}_k^+ &= \hat{\theta}_k^+ - v_\theta \end{aligned} \tag{8}$$

More details of the DFL controller may be found in [14]. In the heading angle controller, using the $\frac{\breve{\theta}^2}{2}$ as Lyapunov candidate, the following control inputs are proposed.

$$V = 0, while w = K_\theta \tanh k_\theta \breve{\theta}_k^+ \tag{9}$$

in which, $K_\theta$ and $k_\theta$ are positive constants.

Consequently, the first DFL controller is executed as long as the distance between robot pose and its target value is less than $\epsilon$. Then, the controller switches to the heading angle controller and sets the robot heading angle. In the heading angle controller, the linear velocity of the robot is zero; thus, the robot position $\left(x, y\right)$ is fixed.

## III. Path Planning Under Motion Uncertainty

### A. MDP Implementation

In the simulation environment, algorithm 1 is used to calculate the state transition function based on the Monte Carlo approach. Also, in this paper, two different reward functions are considered as follows:

$$R_1(s, a, s') = 1 \tag{10}$$

$$R_2(s, a, s') = MaxT - T(s, s') \tag{11}$$

By using a constant reward for all transitions ($R_1$), MDP will just focus on finding the path by the most successful probability toward the desired goal and finding the fastest path is not considdered at all. While by considering traverse time between states in the second reward function ($R_2$) MDP finds a trade-off solution with a highly probable path while it is fast. In the second equation, $MaxT$ means the maximum traverse time between all transitions and $T(s, s')$ is the time for traversing from state $s$ to state $s'$.

**Algorithm 1** : State Transition Function Psudo-Code

**Require:**
$V, E$
$m$ : number of sample points to generate for each transition
$r$ : error margine
**for all** $v \in V$ **do**
  **for all** $v' \in neighbour(v)$ **do**
    $timeStep :=$ full time takes the robots traverse from $v$ to $v'$ without considering uncertainty
    $fault := 0$
    **for** $i = 1$ to $m$ **do**
      $tempPose := v$
      $n_v, n_w$ = random noise for linear and angular velocities
      **for** $j = 1$ to $timeStep$ **do**
        $tempPose$ := robot position while trajectory controller is running by considering uncertainty with noises $n_v, n_w$
        **if** $Collision(tempPose)$ **then**
          $fault := fault + 1$ & go to line 5
        **end if**
      **end for**
      **if** $distanse(tempPose, v') \geq r$ **then**
        $fault := fault + 1$
      **end if**
    **end for**
    $P(v, v') = 1 - (fault/m)$
  **end for**
**end for**

### B. EKF implementation

The linearized system model, is considered as $\gamma = \left(A, B, G, Q, H, M, R\right)$ which is represented by the following state space:

$$\begin{aligned} x_{k+1} &= Ax_x + Bu_k + Gw_k, \quad w_k \sim \mathcal{N}\left(0, Q\right) \\ z_k &= Hx_k + Mv_k, \quad v_k \sim \mathcal{N}\left(0, R\right) \end{aligned} \tag{12}$$

where matrices $A, B, G, H$, and $M$ are given as follows.

$$A = \begin{pmatrix} 1 & 0 & V_k \delta t \sin\theta_k \\ 0 & 1 & V_k \delta t \cos\theta_k \\ 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} \delta t \cos\theta_k & \\ \delta t \sin\theta_k & \\ 0 & \delta t \end{pmatrix}$$

$$G = \begin{pmatrix} \sqrt{\delta t}\cos\theta_k & 0 & \sqrt{\delta t} & 0 & 0 \\ \sqrt{\delta t}\sin\theta_k & 0 & 0 & \sqrt{\delta t} & 0 \\ 0 & \sqrt{\delta t} & 0 & 0 & \sqrt{\delta t} \end{pmatrix}$$

$${}^jH = \begin{pmatrix} \cos{}^j\gamma & \sin{}^j\gamma & 0 \\ -\frac{\sin{}^j\gamma}{{}^jr} & \frac{\cos{}^j\gamma}{{}^jr} & -1 \end{pmatrix}, M = \mathbf{I}$$

and $\gamma = atan2\left({}^jd_2, {}^jd_1\right)$ and ${}^jr = \sqrt{{}^jd_1^2 + {}^jd_2^2}$

EKF is initialized by an initial input and an initial robot pose estimation through its mean value $\mu_0$ and its covariance $\Sigma_0$. Then it gets a control input $u_1$, a map $m$, and a set of features $z_1 = \{z_1^1, z_1^2, ...\}$ measured at the initial time step, along with correspondence variables $c_1 = \{c_1^1, c_1^2, ...\}$. Its output is an updated estimate of $\mu_1, \Sigma_1$. This procedure is repeated in every

time step with updated inputs. EKF has the two following stages. Firstly a prediction phase, in which the robot's state is predicted using the robot's previous state estimate, input control and motion model as follows:

$$\bar{\mu}_k = A\mu_{k-1} + Bu_{k-1} \tag{13}$$

$$\bar{\Sigma}_k = A\Sigma_{t-1}A^T + GQG^T \tag{14}$$

And then a correction phase, in which the prediction is corrected using by using the measurements:

$$\mu_k = \bar{\mu}_k + K_k\big(z_t - \hat{z}_t\big) \tag{15}$$

$$\Sigma_k = \big(I - K_k H_k\big)\bar{\Sigma}_k \tag{16}$$

in which, the Kalman Gain $K_k$ is obtained from:

$$K_k = \bar{\Sigma}_k H_k^T \big(H_k \bar{\Sigma}_k H^T + R\big)^{-1} \tag{17}$$

### C. Switching Controller Implementation

In the DFL controller, the robot's linear and angular velocity are set as:

$$\begin{aligned} V_k &= V_{k-1} + \left(u_1 \cos \breve{\theta}^+_{k-1} + u_2 \sin \breve{\theta}^+_{k-1}\right)\delta t \\ w_k &= \left(u_2 \cos \breve{\theta}^+_{k-1} - u_1 \sin \breve{\theta}^+_{k-1}\right) V^{-1}_{k-1} \end{aligned} \tag{18}$$

where $u_1$ and $u_2$ are defined as follows:

$$\begin{aligned} u_1 &= -k_{p1}\breve{x}_{k-1} - k_{d1}\dot{\breve{x}}_{k-1} \\ u_2 &= -k_{p2}\breve{y}_{k-1} - k_{d2}\dot{\breve{y}}_{k-1} \end{aligned} \tag{19}$$

where, $k_{p1}$, $k_{p2}$, $k_{d1}$ and $k_{d2}$ should satisfy $k_{d1}^2 - 4k_{p1} = k_{d2}^2 - 4k_{p2}$ and $k_{d2}^2 - k_{d1}^2 > 2\sqrt{k_{d2}^2 - 4k_{p2}}$ conditions.

Experimentally, we imposed conservative bounds on our controller outputs:

$$|V| \leq V_{max} = 0.4m/s \quad |w| \leq w_{max} = 0.5rad/s$$

Also, we set the switching controller parameters as follows: $\epsilon = 0.1 \qquad k_{p1} = k_{p2} = 0.3 \qquad k_{d1} = k_{d2} = 0.4$

## IV. Simulation and Real Implementation Results

The results of simulation and real implementation are presented in this section.

### A. System Overview

In this paper, the Parrot Bebop2 drone is used for both simulation and real implementation. The Parrot Bebop2 (Fig. 1Parrot Bebop2 used in our implementationfigure.1) is a robust and maneuverable UAV which contains two forward-facing and downward-facing cameras, a sonar height sensor and a Linux based real-time operating system. The front camera is streamed in 30 frames per second with $640 \times 480$ resolution [24]. The robot velocity is obtained using the onboard optical flow algorithm and transmitted to the ground station with 4HZ frequency rate, which will significantly influence the uncertainty in the robot's motion model. Commands and images are transmitted via a WiFi connection between the host machine and the Bebop 2 drone. We run our algorithms on the host machine (NVIDIA GeForce GTX 960M, IntelCore i7, 12GB memory), running Ubuntu 16.04 and ROS Kinetic Kame with c++ language.

Fig. 1. Parrot Bebop2 used in our implementation

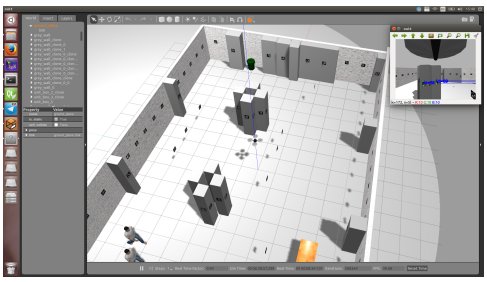

Fig. 2. Simulation environment

Furthermore, Parrot-Sphinx [25] as a simulation package developed in Gazebo by using ROS, is used for the simulation part.

### B. Switching controller comparison by the DFL controller

In simulation environment (Fig. 2Simulation environment-figure.2 ), we can consider any arbitrary start and target positions where in this experiment we set the robot's initial and target positions as:

$$q_{initial} = \big(0, 0, 1.5\big) \quad q_{target} = \big(1, 4, 0\big)$$

In Fig. 3Real and belief trajectories of the robot in switching and DFL controllersfigure.3, the real trajectories of the robot and their belief map in position space are depicted with both switching and DFL controllers. These two trajectories are similar in most of the simulation time except at the and of the flight, since the switching and the DFL controllers are the same and generate the same control input in this time period. However, when the robot gets close to the desired state, the switching controller outperforms the DFL controller. Fig. 4Real and belief heading angle of the robot in switching and DFL controllersfigure.4 shows the heading angle of the robot using these controllers and the robot's belief space. It is clear that the switching controller has better performance in adjusting the heading angle near the goal point.

### C. Path Planning Results

The first task of the robot, which is done in both simulation and the real environment is to find a path from the initial state ($S = 9$) to the desired state ($S = 17$). MDP output for accomplishment of this mission is to travel the following

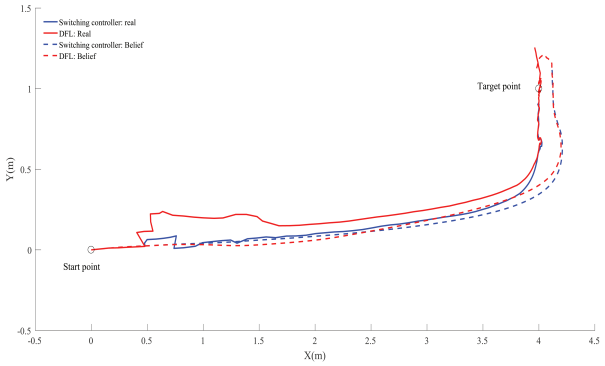


Fig. 3. Real and belief trajectories of the robot in switching and DFL controllers

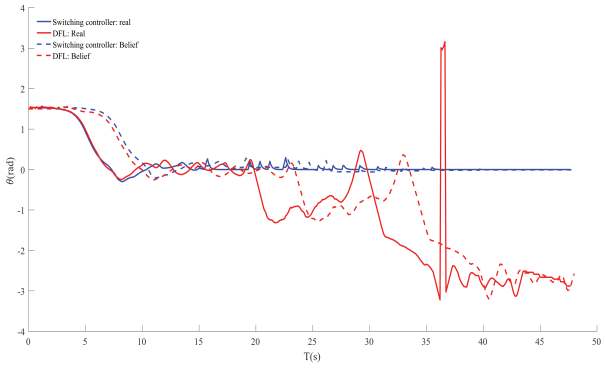


Fig. 4. Real and belief heading angle of the robot in switching and DFL controllers

consecutive states: [9, 8, 10, 7, 15, 18, 17]. Fig. 5Real and belief trajectories results in the simulation environmentfigure.5 shows the traversed path in the simulation environment. In this figure, the blue line shows the belief trajectory while the green one shows the real trajectory of the robot. Furthermore, Fig. 6Real trajectory result in the real environmentfigure.6 shows the aforementioned task in the real environment.

As we mentioned before, two different reward functions

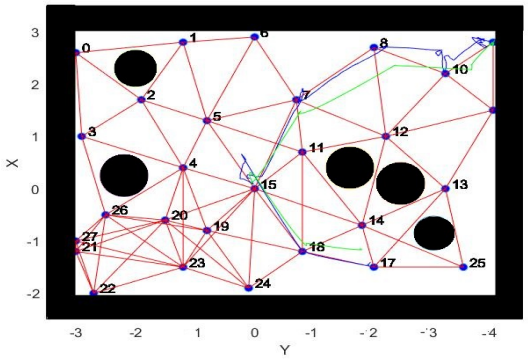


Fig. 5. Real and belief trajectories results in the simulation environment

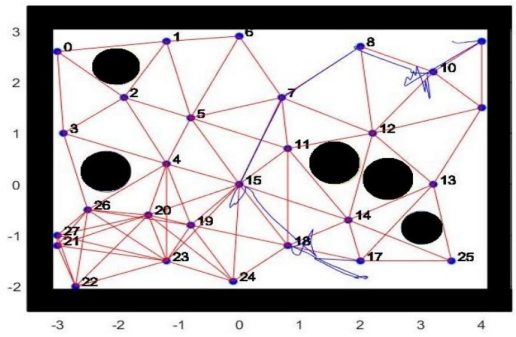

Fig. 6. Real trajectory result in the real environment

TABLE I
PATH PLANNING COMPARISON RESULTS

| Experiment | Algorithm | Reward/Cost | Generated Path |
|---|---|---|---|
| Exp 1 | $A^*$ | time | [3, 4, 5, 7, 12, 10, 9] |
| | MDP | $R_1$ | [3, 2, 5, 7, 8, 10, 9] |
| | | $R_2$ | [3, 2, 5, 7, 12, 10, 9] |
| Exp 2 | $A^*$ | time | [0, 2, 5, 7, 12, 13] |
| | MDP | $R_1$ | [0, 3, 2, 5, 7, 12, 16, 13] |
| | | $R_2$ | [0, 3, 2, 5, 7, 12, 16, 13] |
| Exp 3 | $A^*$ | time | [26, 3, 2, 1] |
| | MDP | $R_1$ | [26, 19, 4, 5, 1] |
| | | $R_2$ | [26, 20, 4, 5, 1] |

are considered in this paper. Table IPath Planning comparison resultstable.1 compares the results of the path planning problem by considering uncertainties using MDP by two reward functions and also without considering uncertainty by using the $A^*$ algorithm. In this table, experiments are defined as follows.

- Exp 1. finding a path from state $s = 3$ to state $s = 9$
- Exp 2. finding a path from state $s = 0$ to state $s = 13$
- Exp 3. finding a path from state $s = 26$ to state $s = 1$

As it is seen in the Exp 1 generated paths the $A^*$ algorithm chooses the fastest path regardless of the probability to collide with the obstacles. While MDP with least possible risk (using reward function $R_1$) avoids traverse from state $s = 3$ to state $s = 4$ directly because it pass near an obstacle. It prefers to chose the lower risk path by passing from state $s = 2$ to traverse from state $s = 3$ to state $s = 4$. The generated path by the second reward function is a tradeoff between $A^*$ and MDP with the first reward function.

## V. CONCLUSIONS

In this paper, a path planning problem for a UAV under motion uncertainty is considered as an MDP problem with two different reward functions, and the results are compared to that of the conventional $A^*$ algorithm. Furthermore, the EKF vision-based localization is used to localize the UAV in an uncertain environment. The implementation results confirm that although the EKF estimates are not completely precise, however, they are accurate enough to be used in the proposed

path planning approach. Moreover, a switching based controller is proposed in this paper for the posture stabilization problem of a mobile robot. The switching control consists of a dynamic feedback linearization and a Lyapunov-based controller. It is verified in the simulation and experimental results that the proposed controller is able to navigate the robot to the desired state, including position and heading orientation, in the presence of robot motion uncertainty. The promising result verifies the further implementation of the proposed algorithm for the path planning and control of autonomous robots.

## VI. Acknowledgment

We are sincerely grateful to the staff of the KN2C robotics team for helping us in carrying out the experiments and collecting data.

## References

[1] Jrme Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research (IJRR)*, 10(6):628–649, 1991.

[2] Howie Choset, Kevin Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation – Errata*. 2010.

[3] Chenghui Cai and Silvia Ferrari. Information-driven sensor path planning by approximate cell decomposition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3):672–689, 2009.

[4] Charles W Warren. Global path planning using artificial potential fields. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 316–321. Ieee, 1989.

[5] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996.

[6] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[7] Esther M Arkin, Refael Hassin, and Asaf Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.

[8] Maxim A Batalin and Gaurav S Sukhatme. The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. *IEEE Transactions on Robotics*, 23(4):661–675, 2007.

[9] Hiroshi Nagamochi and Kohei Okada. A faster 2-approximation algorithm for the minmax p-traveling salesmen problem on a tree. *Discrete Applied Mathematics*, 140(1-3):103–114, 2004.

[10] Frantiek Ducho, Andrej Babinec, Martin Kajan, Peter Beo, Martin Florek, Tom Fico, and Ladislav Juriica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 2014.

[11] Ron Alterovitz, Thierry Siméon, and Ken Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. 2007.

[12] HD Taghirad A Noormohammadi-Asl. Multi-goal motion planning using traveling salesman problem in belief space. *Information Sciences*, 502:164–184, 2019.

[13] S. Thrun, W. Burgard, D. Fox, and R.C. Arkin. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005.

[14] Giuseppe Oriolo, Alessandro De Luca, and Marilena Vendittelli. Wmr control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Transactions on control systems technology*, 10(6):835–852, 2002.

[15] Juan Marcos Toibero, Flavio Roberti, Ricardo Carelli, and Paolo Fiorini. Switching control approach for stable navigation of mobile robots in unknown environments. *Robotics and Computer-Integrated Manufacturing*, 27(3):558–568, 2011.

[16] V Sankaranarayanan and Arun D Mahindrakar. Switched control of a nonholonomic mobile robot. *Communications in Nonlinear Science and Numerical Simulation*, 14(5):2319–2327, 2009.

[17] Yu Tian and Nilanjan Sarkar. Control of a mobile robot subject to wheel slip. *Journal of Intelligent & Robotic Systems*, 74(3-4):915, 2014.

[18] Farzad Pourboghrat. Exponential stabilization of nonholonomic mobile robots. *Computers & Electrical Engineering*, 28(5):349–359, 2002.

[19] Amin Zeiaee, Rana Soltani-Zarrin, Suhada Jayasuriya, and Reza Langari. A uniform control for tracking and point stabilization of differential drive robots subject to hard input constraints. In *ASME 2015 Dynamic Systems and Control Conference*, pages V001T04A005–V001T04A005. American Society of Mechanical Engineers, 2015.

[20] Alessandro De Luca, Giuseppe Oriolo, and Marilena Vendittelli. Control of wheeled mobile robots: An experimental overview. In *Ramsete*, pages 181–226. Springer, 2001.

[21] Bruce L Miller. Finite state continuous time markov decision processes with a finite planning horizon. Technical report, RAND CORP SANTA MONICA CALIF, 1967.

[22] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[23] Sergio Garrido-Jurado, Rafael Munoz-Salinas, Francisco José Madrid-Cuevas, and Rafael Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481–491, 2016.

[24] Parrot bebop 2. https://www.parrot.com/us/drones/parrot-bebop-2.

[25] Parrot-sphinx. https://developer.parrot.com/docs/sphinx/whatissphinx.html.