Journal of Computer & Robotics 15 (1), 2022 65-79

Computer
& Robotics

Global Path Planning of Quadrotor Using Reinforcement Learning

Mehdi Khakbaz [a,*], Majid Anjidani [b]

[a] *Department of Electrical Engineering, Sabzevar Branch, Islamic Azad University, Sabzevar, Iran*
[b] *Department of Computer Engineering and Information Technology, Payame Noor University, Tehran, Iran*

Received 14 May 2022; Accepted 12 July 2022

Abstract

This paper aims to improve the trajectory by an extended reinforcement learning based method in which a new tracking algorithm is used for mobile robot applications with low-rate control command. There are some trajectories that underactuated robots, like quadrotors, are unable to track; hence a suitable trajectory should be designed with respect to the robot's dynamics. In this paper, the initial trajectory is generated by Rapidly-exploring Random Tree Star algorithm which is not suitable for quadrotor application. Then, the initial trajectory is improved by an extension of Path Integral Policy Improvement with Covariance Matrix Adaption (PI2-CMA) algorithm. The extension includes improving tracking algorithm and controller performance considering low-rate control command. Our proposed algorithm succeeded to reduce the cost of tracking by designing safer and shorter trajectories which are more suitable for real robots. Furthermore, the results show that the proposed tracking algorithm and controller improve the performance of tracking. The hardware requirements for implementing our proposed method are a webcam and a personal computer; therefore with a low-cost implementation of the proposed method, a suitable trajectory is designed. In this paper, the initial trajectory is improved by an extension of PI2-CMA algorithm in which the trajectory tracking is performed such that reciprocating motions are avoided. Also, desired velocity and acceleration are used by controller for better tracking.

Keywords: Quadrotor robot, Path planning, Trajectory tracking and Reinforcement learning

1. Introduction

During recent years, the quadrotors have been used for a variety of applications; from military applications [1] to commercial usages like toys [2]. The use of quadrotors are even evident in other areas like load transportation [3], inspection of buildings [4], mapping [5], identification and exploration of unknown regions [6], delivery of goods [7] and search and rescue [8]. Firstly, the quadrotors should be able to automatically map the environment and localize their positions by capturing data through their sensors and secondly plan a free obstacle trajectory according to a defined mission, and finally, to track the trajectory. Recently many research has been done in the field of path planning and obstacle avoidance.

The purpose of path planning is to find a trajectory in free obstacle space in a way that the quadrotor is able to track the trajectory. The path planning has two categories: local and global [9].

The Local Path Planning(LPP) tries to identify the obstacles of the environment and get the required knowledge through the sensors. The LPP generates a free obstacle trajectory along the movement [10]. In Global Path Planning(GPP), the environment is static and known so before the robot starts to move the algorithm can generate a complete trajectory between the start point and the destination point. The focus of GPP is to produce the smooth trajectories for the robot [11]. Therefore, another algorithm might be needed for trajectory smoothing and eliminating extra waypoints [12].

In some LPP methods, a combination of mathematical functions is used to represent the smooth trajectory. As an example, one of the most popular LPP methods is Dubin’s curves in which

[*] Corresponding Author. Email: mehdi_khakbaz@iaus.ac.ir

discontinuities arise at the junction of line and arc. The discontinuities cause errors while tracking [12]. Two types of curves have been used to resolve the discontinuities: the parametric curves whose curvature is a function of their arc length such as Clothoids, and the curves which coordinates have a closed-form expression such as B´ezier curves and B-splines [13].

There are a number of GPP algorithms which are considered as the most common ones. They are namely: Probability RoadMap(PRM) [14], Visibility Graph(VG), the class of Rapidly-exploring Random Tree(RRT) algorithm [15] [16], Space Skeletonization [17] and cell decomposition [18] [19] methods. These methods are considered global when they are applied to the whole configuration space. They can be made local by restricting their application to a subset of configuration space around the current configuration of the robot [19]. In our research, we applied Rapidly-exploring Random Tree Star (RRT*) algorithm as a global path planner. Also there are some GPP algorithm based on deep reinforcement learning which we explore them in the following:

In [20], a deep reinforcement learning based method is presented for quadrotor autonomous navigation in semi-known environments. Their method uses the dueling double deep recurrent Q-learning to implement global path planning with the obstacle map as input. Also, for effectively conducting real-time autonomous obstacle avoidance with monocular vision, a contrastive learning-based feature extraction is combined with their method.

Another deep Q-network (DQN) based method is proposed in [21] for motion planning of quadrotors. The method generates local motion plans in the form of motion primitives using raw depth images from a front-facing camera. DQN is trained with around 75,000 which takes a raw depth image and relative position information as its input, and yields a motion primitive selection as its output.

In [22], a UAV path planning method based deep reinforcement learning is presented. The double deep Q-networks (DDQNs) in the method are trained by exploiting structured map information of the environment.

Neglecting the robot's dynamic leads to trajectory incompatibility of the trajectory with the robot and in order to improve the trajectory, a complementary method needs to be applied. As evidence, B´ezier curves are used in [23] and [24] and spline curves are applied in [25] and [26] to improve the trajectory. In [27] and [28] a GPP algorithm generates an initial trajectory as input to a closed loop structure including a robot’s dynamic, a controller and an optimization method. The output of the structure is the designed trajectory. A variety of optimization methods such as dynamic programming [29], Quadratic Programming(QP) [30] [31], Genetic Algorithm(GA) [32] [33], artificial network [34], Linear–Quadratic Regulator(LQR) [35], Particle Swarm Optimization(PSO) [33] can be used for trajectory improving. In these methods, the problem must be formulated in a particular form, and the optimal trajectory can be obtained by minimizing the cost function with respect to the desired constraints.

Studies show that various methods have been used by researchers to modify the trajectory. In the following we explore a few of these research:

In [30], an initial trajectory is generated with the RRT algorithm, and a controller based on Model Predictive Control (MPC) is designed to generate the feasible trajectories which satisfy given constraints. It is done by using an inner-simulator to track the trajectories. The MPC problem is transformed into QP form which can be numerically solved.

In [31], an initial trajectory in obstacle-free space is generated by the Informed Optimal Rapidly-exploring Random Trees-star (IRRT*) algorithm and the problem of optimizing the trajectory is stated in a QP form considering dynamic constraints. The constraints, which are not in QP forms cannot be used by this method.

Optimization methods are widely used to adjust the trajectory parameters as mentioned earlier. The environment might slightly change, and a real robot is incapable of adapting to new changes after the optimization. We are going to propose a learning method which can learn a smooth and continuous trajectory considering the robot's dynamics and can be applied to the real robot. An alternative approach for adjusting the trajectory parameters is Reinforcement Learning (RL) methods. As an example, Path Integral Policy Improvement (PI2) [36] [37] [38] is a RL algorithm with interesting features like having an arbitrary state-dependent cost function part, exploration noise as the only open algorithmic tuning parameter, numerically robust performance in high-dimensional learning problems and better performance than gradient-based methods [39]. PI2 has been compared with some of the most efficient methods like REINFORCE [40] [41], GPOMDP [42], PG [43], ENAC [44] and PoWER [45] on multiple applications, and the results show that PI2 surpasses all of them [36]. The Path Integral Policy Improvement with Covariance Matrix Adaptation (PI2-CMA) learning algorithm is a modified version of the PI2 algorithm, which automatically and optimally adjusts the exploration noise. PI2-CMA optimizes the trajectory which is generated by Dynamic Movement Primitives (DMPs). The trajectory involves the position, velocity, and acceleration at any time, and is tracked using a PD controller by default.

In this paper, we propose a modified version of PI2-CMA which is applied for improving the initial trajectory, and we show that our modified version of this algorithm minimizes the cost considering the robot's dynamics. The cost is calculated during tracking the trajectory by a proposed tracking algorithm which is suitable for mobile robots with low-rate control command. A modified controller, which considers the desired velocity and acceleration, is used in the proposed tracking algorithm. Since our proposed method benefits from the PI2-CMA algorithm, there is no need to state the problem in a specific form meaning that there is no limitation in the definition of the cost function. We have implemented a two-dimensional position measurement system based on image processing and a communication system between quadrotor and PC to evaluate the behavior of the proposed algorithm. In our practical experiment, trajectories are generated by DMPs and tracked by using our proposed tracking algorithm.

The rest of the paper is organized into four sections. The proposed path planning and trajectory tracking are discussed in section 2, and the implementation and results are presented in section 3 and 4, respectively. We conclude in section 5.

2. Quadrotor Path Planning And Trajectory Tracking

The quadrotor task is passing through all waypoints of the trajectory, which is placed in obstacle-free space. We already know that adjusting the task and improving the controller considering the robot's dynamic can reduce the cost. Two steps are needed to accomplish the task: the first step is generating a suitable path for moving from the starting point to the ending point and the second is tracking the trajectory by a strategy. We describe the strategy in the rest of this section.

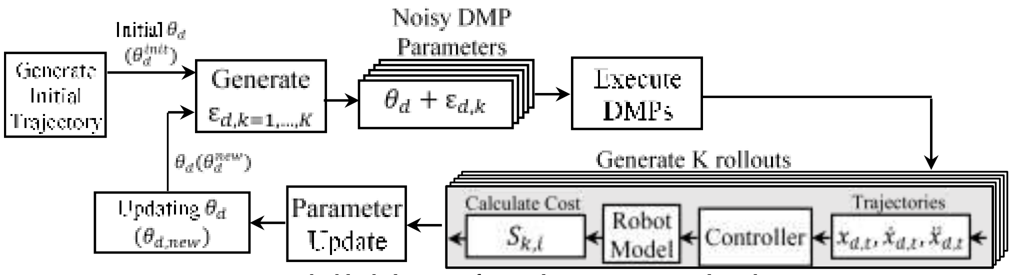
2-1. Proposed path Planning Algorithm

It is preferred that the robot’s dynamic and the measurement errors be considered during path planning. Short, smooth and continuous trajectories with slow variation in velocity and acceleration, are more suitable to track. Also, it is necessary to keep a safe distance from obstacles to avoid a collision. Therefore, the path planning problem can be formulated as a reinforcement learning problem, in which, the initial trajectory has input role. The lowest cost trajectory which is experimented through learning is the output of the learning problem, in other words, the output is the most suitable trajectory that is experimented by the robot. The cost function is defined according to required features, and more rewards are given to the trajectories with lower cost, in this way the required features are obtained.

In our research, PI2-CMA reinforcement learning algorithm is used to optimize the trajectory. The advantages of using PI2-CMA are: having an arbitrary state-dependent cost function part, automatic adjustment of exploration noise, no open algorithmic tuning parameter, numerically robust performance in high-dimensional learning problems and better performance than gradient-based methods [39]. The PI2-CMA algorithm works based on the reward-weighted averaging methods [46]. It minimizes the cost function through an iterative process of exploration and parameter updating. The policy improvement algorithm [47] is extended for mobile robot applications with low-rate control command, and it is shown in Figure (1). The gray block in Figure (1) denotes our special tracking algorithm and controller in which is named “generate K rollouts” as discussed in the next section. In PI2-CMA, DMPs are usually used to produce the path as a special case of parameterized policies which its attractive feature in this research is to provide instantaneous values of velocity and acceleration of the trajectory. DMP equations are expressed as [39]:

$$\begin{bmatrix} \frac{1}{\tau}\dot{\eta}_t \\ \frac{1}{\tau}\ddot{x}_{d,t} \end{bmatrix} = \begin{bmatrix} -\alpha_x\, \eta_t \\ \alpha_z\left(\beta_z\left(goal_d - x_{d,t}\right) - \dot{x}_{d,t}\right) \end{bmatrix} + \begin{bmatrix} 0 \\ g_t^T \end{bmatrix}(\theta_d + \varepsilon_d) \quad (1)$$

where η_t is internal state and α_z, β_z, τ are time constants. $x_{d,t}$, $\dot{x}_{d,t}$, $\ddot{x}_{d,t}$ respectively determine dth component of the position, velocity and acceleration at time t. The generated trajectories are sampled over a uniform subdivision of time $\{t_0, t_1, \ldots, t_N\}$. The basis functions $g_t \in R^{p\times 1}$ are defined by a piecewise linear function approximator with Gaussian weighting kernels [36] as follows:



Initial θ_d (θ_d^{init})
Noisy DMP Parameters
Generate Initial Trajectory
Generate $\varepsilon_{d,k=1,\ldots,K}$
$\theta_d + \varepsilon_{d,k}$
Execute DMPs
$\theta_d(\theta_d^{new})$
Generate K rollouts
Updating θ_d ($\theta_{d,new}$)
Parameter Update
Calculate Cost
$S_{k,i}$
Robot Model
Controller
Trajectories
$x_{d,t}, \dot{x}_{d,t}, \ddot{x}_{d,t}$

Fig. 1. The block diagram of our policy improvement algorithm

$$| g_t |_j = \frac{w_j \eta_t}{\sum_{k=1}^{p} w_k}\left(goal_d - x_{d,0}\right), \quad w_j = exp\left(-0.5h_j\left(\eta_t - c_j\right)^2\right) \quad (2)$$

Where h_j is bandwidth and c_j is the center of the Gaussian kernels.

In Equation (1), the parameter θ_d denotes the shape of the dth dimension of the absorption trajectory in which there is a scalar component in θ_d^j for each component of the basis function $| g_t |_{j=1,\ldots,L}$. This formulation allows a DMP to generate arbitrary smooth trajectories (see [48] and [49]). The features of DMP include [50]:

1- Each DMP generates a dimension of the trajectory $[x_{d,t}, \dot{x}_{d,t}, \ddot{x}_{d,t}]_{t\in\{t_0,\ldots,t_{N-1}\}}$. Therefore, m dimensional trajectory can be generated with m DMPs $[x_{d,t}, \dot{x}_{d,t}, \ddot{x}_{d,t}]_{d\in\{1,\ldots,m\}}$.
2- Each DMP converges from the starting point $x_{d,0}$ to the goal point $goal_d$.
3- The general shape of the trajectory is determined by the parameters $\theta_{d=1,\ldots,m}$:

$$\theta_d = \begin{bmatrix} \theta_d^1 \\ \theta_d^2 \\ \vdots \\ \theta_d^L \end{bmatrix} \quad (3)$$

In the learning problem, exploration begins with adding noise $\varepsilon_{d,k}$ to the parameter θ_d. The noise $\varepsilon_{i,k}$ is drawn from a zero-mean Gaussian distribution with variance Σ_d.

θ_d^{init} in Figure (1) denotes the initial trajectory parameter which is generated by a GPP algorithm. DMP execution repeats K times with $\theta_d + \varepsilon_{d,k}$ which results in K noisy trajectories $[x_{d,t_i,k}, \dot{x}_{d,t_i,k}, \ddot{x}_{d,t_i,k}]_{k=1,\ldots,K}$, which are called rollouts. Each of the rollouts contains N waypoints and robot should track all of them to specify $S_{k,i}$ which is the costs of the kth rollout from ith waypoint to $(N-1)$th waypoint, $i = 0,1,\ldots,N-1$1. A controller is needed to steer the robot along the trajectory with minimum error. It is crucial to obtain a safe trajectory through learning, since tracking an unsafe trajectory leads to robot damage. To solve the problem, we have used the robot's dynamic model to simulate trajectory tracking. In more details, the controller is applied to the robot model to track the desired trajectory, and the output of the model is used to calculate the costs $S_{k,i}$. Furthermore, the system transfer function is identified and used instead of the model to make the simulation closer to the reality.

The costs $S_{k,i}$ are defined as Equation (4) [47]:

1 - In the rest of this paper, i is used instead of t_i for notational compactness

$$S_{k,i} = \sum_{j=i}^{N-1} C_{k,j} \tag{4}$$

$$S\lambda_{k,i} = -10 \frac{S_{k,i} - \min\limits_{j=1,\dots,K} S_{j,i}}{\max\limits_{j=1,\dots,K} S_{j,i} - \min\limits_{j=1,\dots,K} S_{j,i}} \tag{5}$$

$$P_{k,i} = \frac{exp(S\lambda_{k,i})}{\sum_{j=1}^{K} exp(S\lambda_{j,i})} \tag{6}$$

Where $C_{k,i}$ is the instant cost of the ith waypoint of kth rollout, and it is equal to the sum of all the costs that the robot has spent to reach the ith point of the trajectory from $(i-1)$th point. $S\lambda_{k,i}$ is the normalized version of $S_{k,i}$. $P_{k,i}$ denotes the importance of the kth rollout from ith waypoint to $(N-1)$th waypoint. Each of the K rollouts will have a specific contribution in updating the parameters θ_d and as well as determining the size and direction of the noise according to the weights $P_{k,i}$. In other words, the new values of θ_d and Σ_d for the time t_i (i.e., $\theta_{d,i}^{new}$ and $\Sigma_{d,i}^{new}$) are calculated as follow [47]:

$$\theta_{d,i}^{new} = \sum_{k=1}^{K} P_{k,i}\theta_{d,k} \tag{7}$$

$$\Sigma_{d,i}^{new} = \sum_{k=1}^{K} P_{k,i}(\theta_{d,k} - \theta_d)(\theta_{d,k} - \theta_d)^T, \quad i.e. \Sigma_{d,i}^{new} = \sum_{k=1}^{K} P_{k,i}\varepsilon_{d,k}\varepsilon_{d,k}{}^T \tag{8}$$

Then, the parameter θ_d and the noise variance Σ_d is updated as [47]:

$$\theta_d^{new} = \left[\sum_{i=0}^{N-1} (N-i)\theta_d^{new}\right] / \sum_{i=0}^{N-1} (N-i) \tag{9}$$

$$\Sigma_d^{new} = \frac{\left[\sum_{i=0}^{N-1} (N-i)\Sigma_{d,i}^{new}\right]}{\sum_{i=0}^{N-1} (N-i)} + \lambda_{min} I \tag{10}$$

Where I is identity matrix and λ_{min} is a positive scalar constant. In Equation (10), early convergence is avoided by adding $\lambda_{min} I$ to the covariance matrix at each updating step [51]. With the updating method, the learning tends towards less costly trajectories. Equation (8) shows, the contribution of sample $\theta_{d,k}$ in $\Sigma_{d,i}^{new}$ is $P_{k,i}\varepsilon_{d,k}\varepsilon_{d,k}{}^T$. In fact, the effect of sample $\theta_{d,k}$ on the expansion of the distribution in $\varepsilon_{d,k}$ direction is proportional to $P_{k,i}$ and $|\varepsilon_{d,k}|$. It means that the distribution will be expanded towards less costly samples. This process is repeated until the algorithm converges to a suitable trajectory for moving the robot.

2-2. Proposed Path Tracking Algorithm

The proposed path planning algorithm generates a trajectory which includes the waypoints namely $wayPoint_{i=0,\dots,N-1}$. The quadrotor can pass along the trajectory with the sequential chasing of these waypoints. Each waypoint is a vector including three components: instantaneously required position, velocity and acceleration, i.e. $(p_{t_i}, \dot{p}_{t_i}, \ddot{p}_{t_i})$, where $p_{t_i} = (x_1, x_2, x_3)$ shows the position of the robot in three-dimensional space. Trajectory tracking is improved using the desired velocity and acceleration in the controller. Also for every moment, a suitable target point should be determined by the tracking algorithm due to low-rate control command and position updating. In the next part, we present the details of the proposed trajectory tracking algorithm and its controller.

In the trajectory tracking algorithm, the starting waypoint is considered as the first target point $(p_T, \dot{p}_T, \ddot{p}_T)$, and the controller brings the quadrotor to the target point neighborhood[2]. Then the target point is replaced by the next waypoint, and the controller steers the quadrotor toward it. The process of replacing the target points will be continued until the quadrotor reaches to the end of the trajectory. By this approach, if the quadrotor passes the target point with a distance more than r_a, the target point does not change; therefore reciprocating movement is unavoidable. For this issue, another algorithm is proposed to determine the suitable target points after lift-off and before landing. The pseudocode is given in Table 1. The execution of the algorithm is done after each position and angle measurement. The algorithm starts tracking by selecting $wayPoint_0$ as the target point; therefore the quadrotor moves in the neighborhood of $wayPoint_0$. If the quadrotor stays nf successive time steps at the neighborhood of $wayPoint_0$, the target point changes to $wayPoint_1$ (rows 3 to 8 Table 1). In the next time steps, if the quadrotor is in the neighborhood of a target point,

[2] - Reaching to the target point is impossible duo to noise, disturbances and error in position measurement; therefore we consider a neighborhood with radius r_a around the target point. Placing the quadrotor in the neighborhood means to reach the point.

the target point is replaced by the first waypoint outside of its neighborhood. In other words, the waypoints might be dense and close together in the vicinity of the target point; hence some of the next waypoints are also in the neighborhood. In this case, the target point will be replaced by the first waypoint in the outside of the neighborhood (rows 14 to 18 in Table 1). The rows 10 to 13 in Table 1 are considered for reciprocating avoidance during trajectory tracking. In other words, if the quadrotor is getting away from the current target point, which is assumed to be $wayPoint_i$, the distances of quadrotor from $2n_p + 1$ waypoints, i.e. $wayPoint_{i-n_p}$, $\ldots, wayPoint_{i+n_p}$ are calculated. The target point is replaced by the waypoint with minimum distance, minDist. ra is set to minDist+Δ where Δ is a small value. In this way,the quadrotor returns to the most appropriate waypoint and continues passing the trajectory. The target point is not changed anymore after setting to $wayPoint_{N-1}$, in which the desired velocity and acceleration are zero.

Waypoints contain velocities and accelerations of the trajectory and using the velocities and accelerations can improve the trajectory tracking as

Table 1
The target point selection algorithm for mobile robots with low-rate control command

```
     Initialization: Trajectory={$wayPoint_{i=0,\ldots,N-1}$}, $p = Position\ of\ robot$, $(p_T, \dot{p}_T, \ddot{p}_T) = wayPoint_0$, count=0, i=1.
1.   $dist = \sqrt{(p_T.x_1 - p.x_1)^2 + (p_T.x_2 - p.x_2)^2 + (p_T.x_3 - p.x_3)^2}$
2.   IF $(p_T, \dot{p}_T, \ddot{p}_T) == wayPoint_0$ THEN
3.   |   IF count<$n_f$ && dist< $r_a$ THEN
4.   |   |   count++;
5.   |   ELSE IF count>=$n_f$ && dist< $r_a$ THEN
6.   |   |   $(p_T, \dot{p}_T, \ddot{p}_T) = wayPoint_1$
7.   |   ELSE
8.   |   |   count=0;
9.   ELSE
10.  |   IF dist > previousDist THEN   // Reciprocating avoidance  code
11.  |   |   [$(p_T, \dot{p}_T, \ddot{p}_T)$, NearDist] = FindNearestTarget($wayPoint_{i-n_p}, \ldots, wayPoint_{i+n_p}$);
12.  |   |   $r_a$ = NearDist+Δ;
13.  |   |   dist= NearDist;
14.  |   WHILE dist< $r_a$
15.  |   |   i=i+1
16.  |   |   $(p_T, \dot{p}_T, \ddot{p}_T) = wayPoint_i$
17.  |   |   IF i==(n-1) THEN break;
18.  |   |   $dist = \sqrt{(p_T.x_1 - p.x_1)^2 + (p_T.x_2 - p.x_2)^2 + (p_T.x_3 - p.x_3)^2}$
19.  previousDist = dist;
20.  IF  i>=n-1 Then target = wayPoint$_{(n-1)}$
21.  Calculate control signal and change robot input
```

Various controllers such as PID [4] [52], PD [53], PI [52], Takagi-Sugeno fuzzy controller [54] and sliding mode controller [55] are used by researchers to navigate the quadrotor along the trajectory. In this paper, a two degrees of freedom PID controller based on the controller in [53], is used as shown in Equation (10). The advantages of the PID controller are: simple adjustment of its coefficients, robustness and capability of industrial implementation.

$$u = K_P(p_T - p) + K_I \int (p_T - p)dt + K_D(b\dot{p}_T - \dot{p}) + c\ddot{p}_T \tag{10}$$

$$\dot{p} = \frac{M}{1 + \frac{M.T_S}{Z-1}} p \tag{11}$$

Where $u = (u_{x_1}, u_{x_2})$ is control signal in which u_{x_1} and u_{x_2} are control signals for $p.x_1$ and $p.x_2$ axes. p_T, $\dot{p}_T$ and $\ddot{p}_T$ are respectively, the position, velocity, and acceleration of the target point which is determined by the proposed algorithm, p and $\dot{p}$ are respectively, the quadrotor position and velocity. T_S is time step, M is a constant coefficient that determines the bandwidth of the highpass filter which acts as an differentiator, Z is z transfer variable, K_P, K_I, K_D, b and c are PID controller coefficients. The first and second term of the controller reduce the position error. The quadrotor velocity converges to the desired velocity due to the third component. The third term, i.e., velocity error is used instead of the derivative of the error signal. The error is equal the difference between the target point and the quadrotor position. Since the target point is constant until the quadrotor reaches to its neighborhood, we have $\dot{p}_T = 0$ in the error derivative that is not true. With this approach, the velocity is also controlled. Quadrotor velocity is calculated by the time derivative of its position. For this purpose, a high-pass filter in Equation (11) has been used. The filter also reduces the effect of the noise in the computation of the derivative. The term $c\ddot{p}_T$ is the prediction of the desired acceleration, and it can significantly improve performance for trajectories with large accelerations or controllers with soft gains [53].

The yaw angle affects the quadrotor direction according to Equation (12).

$$\begin{aligned} u_\theta &= u_{x_1}\cos\psi + u_{x_2}\sin\psi \\ u_\varphi &= -u_{x_1}\sin\psi + u_{x_2}\cos\psi \end{aligned} \tag{12}$$

Where u_θ and u_φ are control signals of pitch angle and roll angle. ψ is quadrotor yaw angle. To control the yaw angle, PID controller is used as:

$$\begin{aligned} u_\psi = K_{P3} \times (\psi_T - \psi) + K_{I3}\int (\psi_T - \psi)dt \\ + K_{D3} \times \frac{Z-1}{Z}(\psi_T - \psi) \end{aligned} \tag{13}$$

Where u_ψ is yaw angle control signal. ψ_T is desired yaw angle. K_{D3}, K_{I3} and K_{P3} are PID controller coefficients. Block diagram of the system controller is shown in Figure (2).

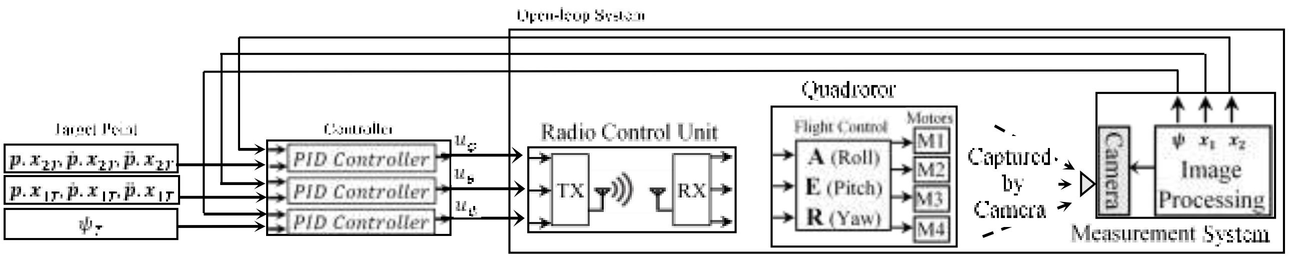


Fig. 2. Block Diagram of Closed-Loop System

The control signals u_φ, u_θ and u_ψ respectively determine the desired roll angle(feed to input A[3] of flight controller[4]), the desired pitch(feed to input E) and the desired yaw angle(feed to input R) for the flight controller as Figure (2) shows.

3. Implementation

We did a low-cost implementation of our proposed algorithm and evaluated its performance within a real configuration containing a quadrotor, a webcam, and a computer. Furthermore, a two-dimensional trajectory is considered in an arena that could be seen via the webcam. On the arena, the robot's attitude $(p.x_1, p.x_2, \psi)$ and obstacles are obtained by processing the webcam frames.

3-1. Specifications of the Robot and the Arena

Figure (3) shows the specifications of the arena as well as the webcam's view. The color of the floor is white, and the color of the obstacles are set to black. Our webcam, which is a TSCO TW 100K, sends the captured images to a computer via its USB port. The rate of data transmission is 30 frames per second (30 fps). The size of quadrotor with its propeller guards is (approximately) equivalent to a square with a side length of 35cm. Table 2 shows the quadrotor parts list. NAZA M-Lite flight controller has some parameters which called gains and we adjust them as shown in

Table 3. We have made a radio control unit which can communicate with the computer and quadrotor via two modules namely HLK-RM04 and MRF24J40. The radio control unit runs in two modes: either manual or automatic. Running in manual mode allows the operator to control the quadrotor manually, but the automatic mode enables the quadrotor to receive the commands from a computer automatically.

3 - A, E and R are abbreviation for Aileron, Elevator and Rudder, respectively.

4 - Our flight controller is NAZA M-Lite [58]

3-2. Our Software Program

Using Visual C++ 2012, we developed a processing unit by which the robot is controlled and the calculations are performed. The processing unit has three modules:

The first module is responsible for performing the following stages:

1- Capturing an image from the webcam.
2- Processing the image using OpenCV library to identify the obstacle positions
3- Constructing the spherical Bounding Volume Hierarchy (BVH) tree, by BVH algorithm to enable checking the collision occurrence in the next stages [56]
4- Generating an initial trajectory [57] by RRT* algorithm.

In the second module, the initial trajectory is represented using the DMPs and improved by the proposed method which is done offline.
The third module:

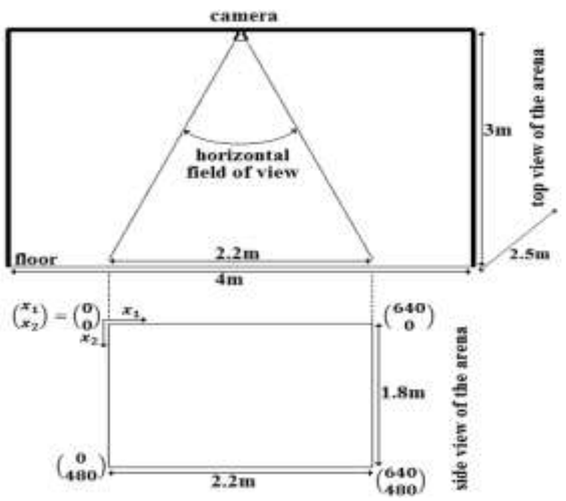


Fig. 3. Schematic of the Arena Map

Table 2
Quadrotor parts list

Frame	HMF X240 Quadcopter Frame
Motor	T-Motors 1806
ESC	XC3012BA dualsky
weight	750 grams
Flight controller	NAZA M-Lite
Propeller	3 blade 5×3
Center to motor dis	13cm
Battery	3000mAH

Table 3
The flight Controller Gain

Basic Gain Pitch	125
Basic Gain Roll	125
Basic Gain Yaw	115
Basic Gain Vertical	140
Attitude Gain Pitch	125
Attitude Gain Roll	125

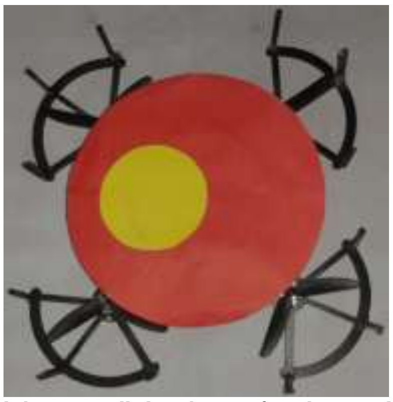


Fig. 4. Two disks are installed on the top of quadrotor to determine the position and angle of the robot.

1- Capturing frames with 30fps from the webcam
2- Calculating the position and angle of the robot in each frame.
3- Sending control commands to quadrotor for tracking the trajectory.

The red and small yellow disks in Figure (4) are installed on the quadrotor to determine the position and angle (yaw) of the robot. The center of disks is calculated by using the image processing. The quadrotor position is indicated by the center of the red disk. The angle between x_1 axis and the line segment, which connects the disk centers, is yaw. The units of the yaw and position are degree and pixel respectively.

3-3. Identification of the System

Figure (2) shows the open loop system whose inputs are applied to the flight controller inputs and its outputs are position and angle, ($p.x_1$, $p.x_2$) and ψ, of the quadrotor in the arena. The inputs of the flight controller (E,R,A) are 50 Hz PWM signals whose pulse-width is determined as shown in Table 4. The control signal u_φ, u_θ and u_ψ are pulse-width in the range [1000, 2000] which are transformed to the corresponding PWM pulses in radio controller unit. The radio controller unit transmits the pulses to the flight controller as inputs.
According to Equation (12), position control is simplified as $u_\theta = u_{x_1}$ and $u_\varphi = u_{x_2}$ by setting

$\psi_T = 0$ in the yaw controller. We used Matlab identification toolbox to identify the transfer functions of the open-loop system including position transfer functions, i.e. $H_{p.x_1}$and $H_{p.x_2}$, and yaw transfer function, i.e. H_ψ. For this purpose, u_θ, u_φ and u_ψ signals are fed to the inputs E, A, R of the flight controller and $p.x_1$, $p.x_2$ and ψ are measured as outputs which are shown in Figure (5). $H_{p.x_1}$, $H_{p.x_2}$ and H_ψ are identified as:

$$\begin{aligned} H_{p.x_1} &= \frac{-0.009862Z + 0.011745481275}{Z^2 - 2.0075Z + 1.007589555} \\ H_{p.x_2} &= \frac{-0.0012214Z + 0.00270739686}{Z^2 - 2.034 + 1.035011534} \\ H_\psi &= \frac{0.001539Z}{Z^2 - 1.9Z + 0.9002} \end{aligned} \tag{14}$$

and the response of the identified systems to the inputs are plotted in Figure (5).

Result In order to evaluate the performance of our proposed algorithm, two arenas with different obstacles have been created. First, the obstacles regions are specified, and the BVH tree is generated to check the collision occurrence. In our experiment, we use two DMPs for the variables $p.x_1$ and $p.x_2$ in the two-dimensional arena. The initial parameters $\theta_{d=1,2}^{init}$ of DMPs are calculated from the initial trajectory.

The PI2-CMA algorithm adjusts the parameter vectors $\theta_{d=1,2}$ such that the cost function is minimized. We defined the cost function according to the following criteria:

Unsafe distances from obstacles during trajectory are punished.

The trajectories with shorter length are preferred.

Reaching to the goal point is rewarded.

According to the above-mentioned criteria, the cost function for the piece of kth rollout starting from $wayPoint_i$ is defined as:

$$\begin{aligned} C_{k,i} &= \xi_1 \times TLC_{k,i} + \xi_2 \times CUDO_{k,i} \\ C_{k,i} &= C_{k,i} + \xi_3 \times CRGP_{k,i}\,,\ \ for\ i > \frac{N-1}{2} \\ S_{k,i} &= \sum_{j=i}^{N-1} C_{k,j} \end{aligned} \tag{15}$$

Table 4
Pulse width specifications of the flight controller Inputs

Input	Minimum pulse width	Middle pulse width	Maximum pulse width
A(roll)	1000µs	1500µs	2000µs
E(pitch)	1000µs	1500µs	2000µs
R(yaw)	1000µs	1500µs	2000µs

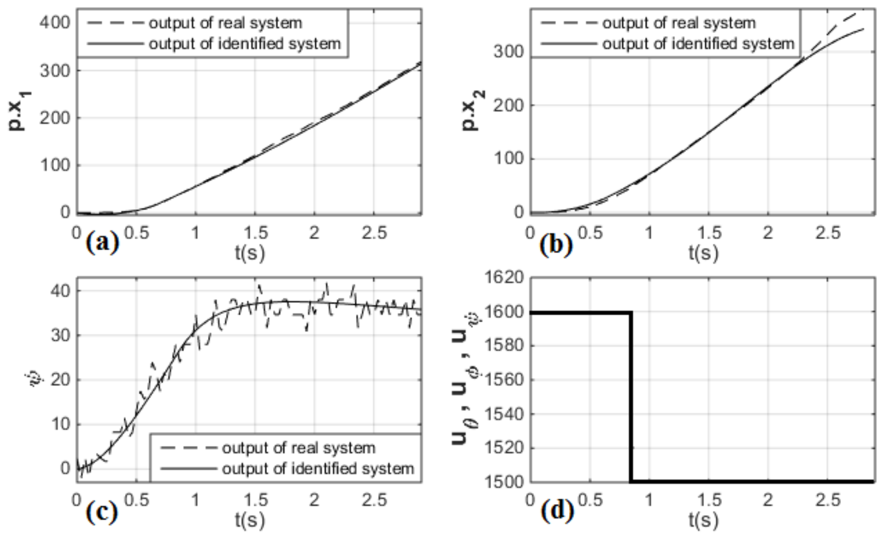


Fig. 5. a) The response of $H_{p.x_1}$ and the position system $p.x_1$ to u_θ. b) The response of $H_{p.x_2}$ and the position system $p.x_2$ to u_φ. c) The response of H_ψ and the yaw system to u_ψ. d) The input signals u_θ, u_φ and u_ψ

where $TLC_{k,i}$ is the distance between current and previous robot positions (note that the kth rollout length is equal to $\sum_{j=0}^{N-1} TLC_{k,j}$) and is defined as:

$$TLC_{k,i} = (p.x_1 - p_{prev}.x_1)^2 + (p.x_2 - p_{prev}.x_2)^2 \quad (16)$$

where the current and the previous position of the quadrotor are denoted by p and p_{prev}. $CRGP_{k,i}$ is the distance between current robot position and the goal point and is defined as:

$$CRGP_{k,i} = (p.x_1 - p_{goal}.x_1)^2 + (p.x_2 - p_{goal}.x_2)^2 \quad (17)$$

Where p_{goal} denotes the last waypoint which also called goal point. $CUDO_{k,i}$ denotes the cost of the robot distance from obstacles and is defined as follows:

$$CUDO_{k,i} = \sum_{l=1}^{Q} e^{10\left(\frac{mindist - obsdist_l}{mindist}\right)} \quad (18)$$

Where Q is the number of obstacles. Minimum safe distance from the obstacles is $mindist$. The distance between the quadrotor and the lth obstacle is denoted by $obsdist_l$, hence the distance $obsdist_l$ is safe if it is greater than $mindist$. ξ_1, ξ_2 and ξ_3 are determined according to the importance of each criterion and changing the value of the coefficients affect the result.

The control signal for a target point is applied to the $H_{p.x_1}$ and $H_{p.x_2}$ models and the output, i.e., the quadrotor positions during tracking, are calculated. Hence after tracking, we have all the values which are needed for calculating the cost. Table 5, Table 6, Table 7, Table 8 and Table 9 respectively show the parameter values for DMP, yaw and position controller tracking algorithm and cost function. $T_S = 1/fps$ and fps denotes the frame rate of the webcam. α_x, β_z, α_z and τ are set as in [36]. The controller coefficients K_P, K_I, K_D, K_{P3}, K_{I3} and K_{D3} are initially calculated by Matlab's PID tuner and then readjusted in order to reach a better response. The other values (ξ_1, ξ_2, ξ_3, A, b, L, nf, np and Δ) are obtained by trial and error.

Table 5
The values of the DMP Parameters

DMP Parameters	α_x	β_z	α_z	τ(s)	L
Value	8.333	6.25	25	0.05	25

Table 6
The values of the position controller coefficients

Position controller coefficients	K_P	K_I	K_D	b	c	M	T_S(s)
Value	0.5	0.3	1.2	0.5	0.1	20	0.033

Table 7
The values of the yaw controller coefficients

Yaw controller coefficients	K_{P3}	K_{I3}	K_{D3}
Value	3	0.3	7

Table 8
The values of the tracking algorithm Parameters

Tracking algorithm Parameters	n_f	n_p	Δ
Value	5	10	5

Table 9
The values of the cost function parameters

Cost function parameters	ξ_1	ξ_2	ξ_3	$mindist$
Value	0.1	75	10	100

Figure (6) and Figure (7) show initial and improved trajectory of an example leaning run in two different arenas, A and B. Figure (8) and Figure (9) show the average learning curves of 10 different learning runs and illustrate how the cost is reduced during the learning process.

By a real experiment in arena A, we compare the performance of the proposed tracking algorithm with/without reciprocating avoidance code (rows 10 to 13 in Table 1) in Figure (10) and Figure (11). In Figure (10) the improved trajectory in arena A is tracked based on our proposed tracking algorithm three times that the quadrotor movement during one of them is shown in Figure (11).

Figure (12) shows that the algorithm with reciprocating avoidance code has a higher performance. Figure (13) shows the effects of the desired speed and acceleration terms in the controller behavior of the proposed tracking algorithm.

At the beginning of a real experiment, the quadrotor is manually steered into the neighborhood of the starting point and automatic control is activated. During the real experiment, the neighborhood radius, ra, for the first waypoint is set to 10 and for next waypoints, it is set to 15. Also, the yaw angle controller is used to hold $\psi_T = 0$.

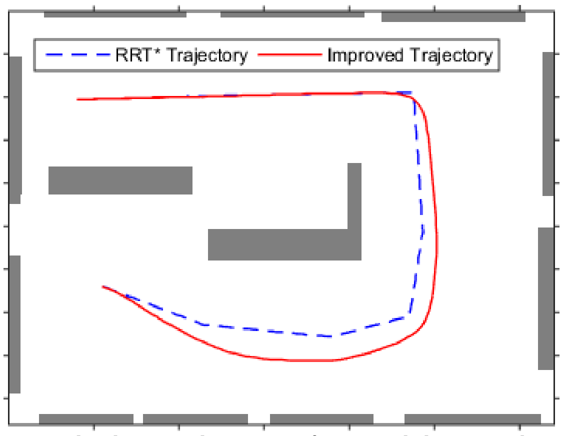


Fig. 6. Initial and improved trajectory of an example leaning in the arena A.

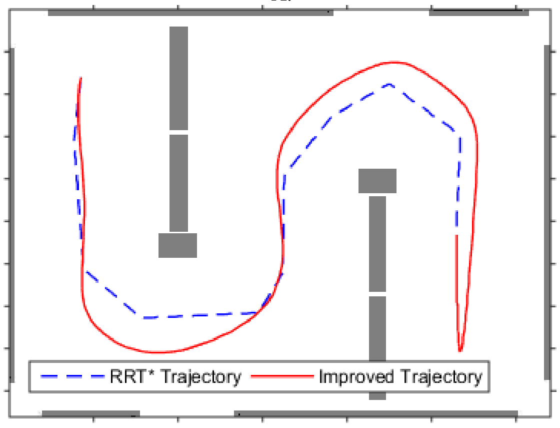


Fig. 7. Initial and improved trajectory of an example leaning in the arena B.

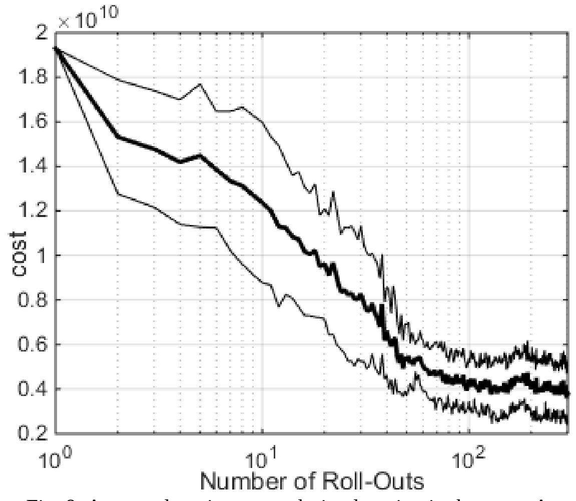


Fig. 8. Average learning curve during learning in the arena A.

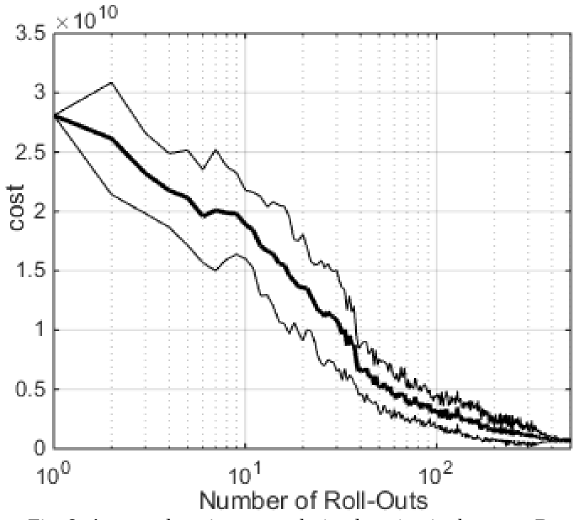


Fig. 9. Average learning curve during learning in the arena B.

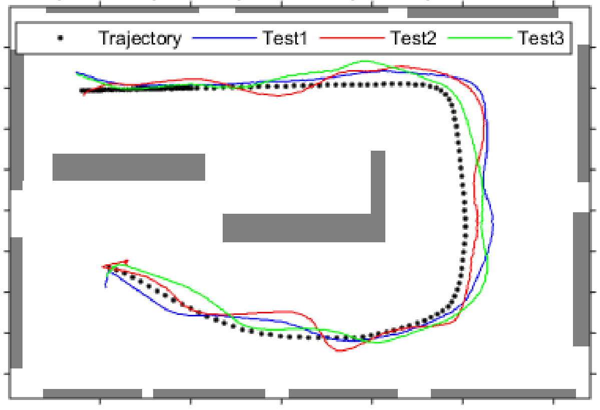


Fig. 10. The dotted line denotes the learned trajectory in arena A that is tracked three times by the proposed tracking algorithm in real experiment.

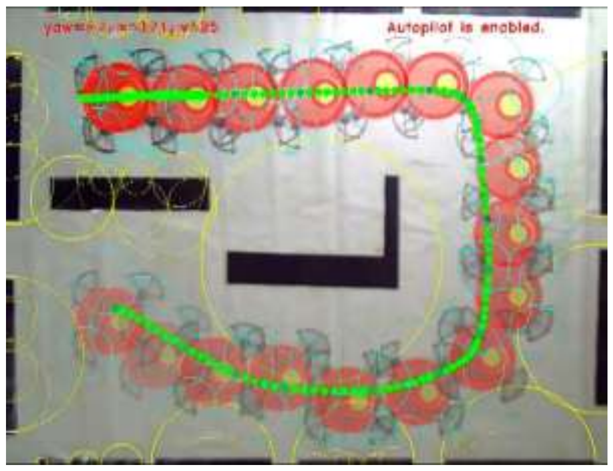


Fig. 11. The quadrotor movement during the tracking experiment which is labeled Test1 in Figure (10).

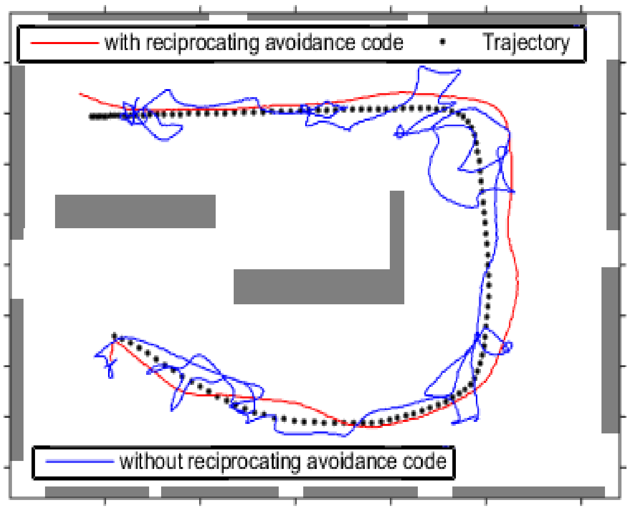


Fig. 12. The tracking result of the proposed tracking algorithm with/without rows 10 to 13 in Table 1 in the real experiment.

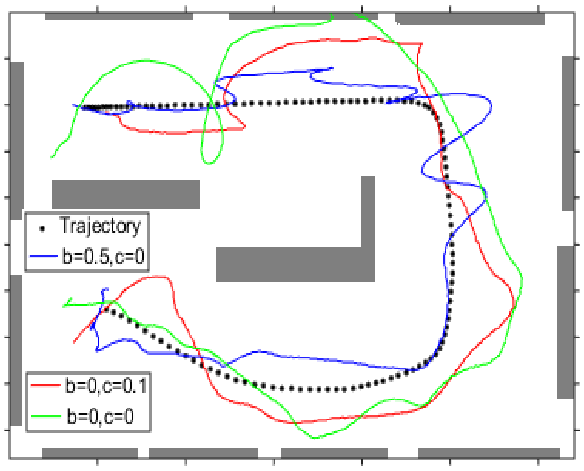


Fig. 13. The effects of ignoring desired speed and acceleration on the controller performance in the real experiment.

At the end, we want to design a path by the proposed algorithm for a more complex problem which is presented in paper [22]. The problem in the paper is designing a path which should be passed through the vicinity of some certain points. The designed path in [22] is shown in Figure (14).

To track such a path, the robot have to move slowly in the path corners, otherwise it may not be able to navigate the path successfully. If the dynamics of the robot is taken into account in the design of the path, it will be possible to navigate the path at higher speeds. We designed the same path with our proposed algorithm regarding to the robot dynamics, as shown in Figure (15).

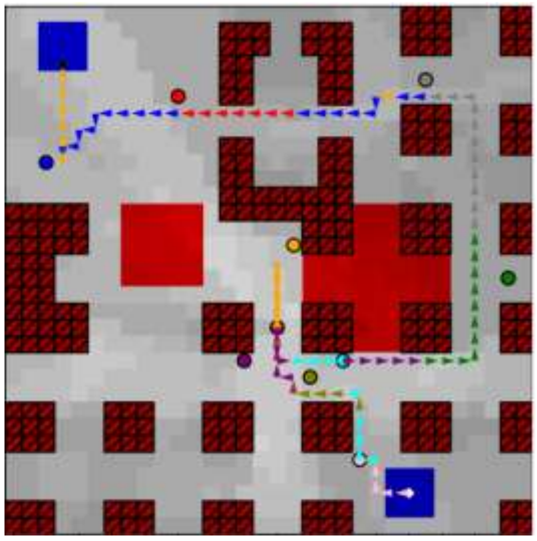


Fig. 14. The designed path in [22]. The colored circles denote the points that the robot should pass through their vicinity. The blue squares denote the start/end points of the path. The red area denotes the no-fly zone. The striped red squares denote the obstacles.

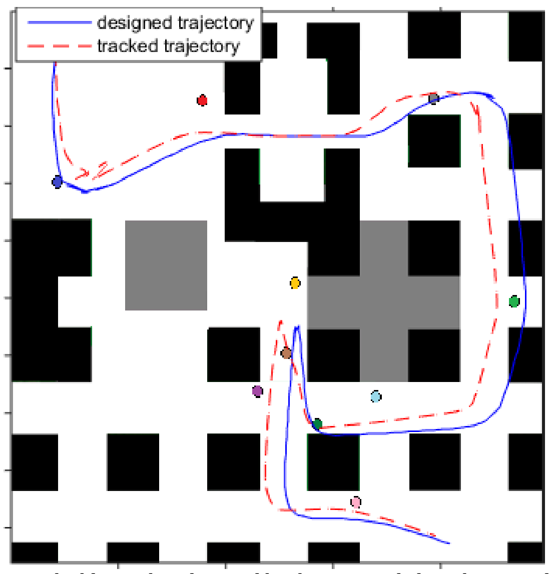


Fig. 15. The blue path is designed by the proposed algorithm regarding to the robot dynamics. The red path denotes the path that the robot pass

The figure shows that the designed path has passed through some obstacles and during tracking the path, the robot can pass through safe areas at higher speed as shown in the figure.

4. Conclusion

In this paper, a learning-based GPP algorithm is proposed that are applicable to mobile robots. Advantages of using our proposed algorithm are: 1- Using the identified model, the effects of the robot's dynamic and error/delay in the position measurement system are considered, 2- DMP is used to generate desired velocities and accelerations as well as desired positions that improve tracking.

To evaluate the performance of the proposed algorithm, a processing unit is developed for processing images, performing our algorithm calculations and controlling the robot movement. Also, a radio control unit is made for communication between the computer and the quadrotor (or operator in the manual mode).

Results show that the proposed algorithm improves the initial trajectory concerning the robot's dynamics and PI^2-CMA with the designed cost function tend to shorter and safer trajectory and the proposed tracking algorithm and controller improve the performance of tracking.

5. Acknowledgment

The authors would like to thank Omid Maldar, Dr. Moosa Ayati and Dr. Javad Mashayekhi Fard.

References

[1] S. Bortoff, "Path planning for UAVs," in *IEEE Proceedings of the American Control Conference*, Chicago, IL, USA, 2000.

[2] RotorCopters, "Rotor Copters," 2017. [Online]. Available: http://www.rotorcopters.com/sub-50-multirotor-drone-mini-reviews/. [Accessed 21 September 2017].

[3] G. V. Raffo and M. M. d. Almeida, "Nonlinear robust control of a quadrotor UAV for load transportation with swing improvement," in *American Control Conference (ACC)*, Boston, MA, USA, 2016.

[4] S. Emelianov, A. Bulgakow and D. Sayfeddine, "Aerial laser inspection of buildings facades using quadrotor," *Procedia Engineering*, vol. 85, pp. 140-146, 2014.

[5] K. Zainuddin, N. Ghazali and Z. M. Arof, "The feasibility of using low-cost commercial unmanned aerial vehicle for small area topographic mapping," in *Aerospace Electronics and Remote Sensing Technology (ICARES), 2015 IEEE International Conference on*, Bali, Indonesia, 2015.

[6] P. Sujit and R. Beard, "Multiple UAV exploration of an unknown region," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2, p. 335–366, April 2008.

[7] N. K. Yang, K. T. San and Y. S. Chang, "A Novel Approach for Real Time Monitoring System to Manage UAV Delivery," in *Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on*, Kumamoto, Japan, 2016.

[8] S. Verykokou, A. Doulamis, G. Athanasiou, C. Ioannidis and A. Amditis, "UAV-based 3D modelling of disaster scenes for Urban Search and Rescue," in *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*, Chania, Greece, 2016.

[9] K. Sedighi, K. Ashenayi, R. Wainwright and H. Tai, "Autonomous local path planning for a mobile robot using a genetic algorithm," in *IEEE Congress on volutionary Computation,2004*, Portland, USA, 2004.

[10] H. Yu, R. Sharma, R. W. Beard and C. N. Taylor, "Observability-based local path planning and obstacle avoidance using bearing-only measurements," *Robotics and Autonomous Systems*, vol. 61, no. 12, p. 1392–1405, 2013.

[11] A. Signifredi, B. Luca, A. Coati, J. S. Medina and D. Molinari, "A General Purpose Approach for Global and Local Path Planning Combination," Las Palmas, Spain, 2015.

[12] K. Yang and S. Sukkarieh, "An analytical continuous-curvature pathsmoothing algorithm," *Robotics, IEEE Transactions on*, vol. 26, no. 3, p. 561–568, 2010.

[13] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuouscurvature paths," *IEEE Trans. Robot.*, vol. 20, no. 6, p. 1025–1035, 2004.

[14] L. E. Kavraki, M. N. Kolountzakis and J. -C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166 - 171, 1998.

[15] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," in *TR 98–11, Computer Science Dept., Iowa State University*, 1998.

[16] I. Noreen, A. Khan and Z. Habib, "Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, pp. 97-107, 2016.

[17] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *4th International Symposium on Voronoi Diagrams in Science and Engineering, 2007. ISVD*, Glamorgan, UK, 2007.

[18] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos and L. Petrou, "A Review of Global Path Planning Methods for Occupancy Grid Maps Regardless of Obstacle Density," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, p. 829–858, 2016.

[19] J. Latombe, Robot Motion Planning, Boston, MA: Kluwer Academic Publishers, 1991.

[20] J. Ou, X. Guo, W. Lou and M. Zhu, "Quadrotor

Autonomous Navigation in Semi-Known Environments Based on Deep Reinforcement Learning," *Remote Sens,* vol. 13, no. 21, p. 4330, 2021.

[21] E. Camci and E. Kayacan, "End-to-End Motion Planning of Quadrotors Using Deep Reinforcement Learning," *CoRR abs/1909.13599,* 2019.

[22] M. Theile, H. Bayerlein, R. Nai, D. Gesbert and M. Caccamo, "UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning," in *20th International Conference on Advanced Robotics (ICAR),* 2021.

[23] K. Yang and S. Sukkarieh, "Planning Continuous Curvature Paths for UAVs Amongst Obstacles," in *Australasian Conference on Robotics and Automation 2008(ACRA 2008),* Canberra, Australia, 2008.

[24] Y.-J. Tsai, C.-S. Lee, C.-L. Lin and C.-H. Huang, "Development of Flight Path Planning for Multirotor Aerial Vehicles," *Aerospace,* vol. 2, no. 2, pp. 171-188, 2015.

[25] A. Boeuf, J. Cort´es, R. Alami and T. Sim´eon, "Planning agile motions for quadrotors in constrained environments," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on,* Chicago, IL, USA, 2014.

[26] K. Yang, S. Moon, S. Yoo, J. Kang, N. L. Doh, H. B. Kim and S. Joo, "Spline-Based RRT Path Planner for Non-Holonomic Robots," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, p. 763–782, 2014.

[27] L. Matthies, R. Brockers, Y. Kuwata and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *2014 IEEE International Conference on Robotics & Automation (ICRA),* Hong Kong, China, 2014.

[28] Y. Kuwata, G. Fiore and E. Frazzoli, "Real-time Motion Planning with Applications to Autonomous Urban Driving," *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY,* vol. 17, no. 5, pp. 1105-1118, 2009.

[29] A. L. Jennings, R. Ordonez and N. Ceccarelli, "Dynamic programming applied to UAV way point path planning in wind," in *Computer-Aided Control Systems, 2008. CACSD 2008. IEEE International Conference on*, San Antonio, TX, USA, 2008.

[30] P. Lin, S. Chen and C. Liu, "Model Predictive Control-based Trajectory Planning for Quadrotors with State and Input Constraints," in *2016 16th International Conference on Control, Automation and Systems (ICCAS 2016),* HICO, Gyeongju, Korea, 2016.

[31] L. Campos-Mac´ıas, D. G´omez-Guti´errez, R. Aldana-L´opez, R. d. l. Guardia and J. I. Parra-Vilchis, "A Hybrid Method for Online Trajectory Planning of Mobile Robots in Cluttered Environments," *IEEE ROBOTICS AND AUTOMATION LETTERS,* vol. 2, no. 2, pp. 935-942, 2017.

[32] C. Yongbo, Y. Jianqiao, M. Yuesong, Z. Siyu, A. Xiaolin and J. Zhenyue, "Trajectory optimization of multiple quad-rotor UAVs in collaborative assembling task," *Chinese Journal of Aeronautics,* vol. 29, no. 1, p. 184–201, 2016.

[33] V. Roberge, M. Tarbouchi and G. Labonté, "Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning," *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS,* vol. 9, no. 1, pp. 132-141, 2013.

[34] S. A. Gautam and N. Verma, "Path planning for unmanned aerial vehicle based on genetic algorithm & artificial neural network in 3D," in *2014 International Conference on Data Mining and Intelligent Computing (ICDMIC),* New Delhi, India, 2014.

[35] A. Perez, R. Platt, G. Konidaris, L. Kaelbling and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, 2012.

[36] A. Evangelos, J. Buchli and S. Schaal, "A Generalized Path Integral Control Approach to Reinforcement Learning," *Journal of Machine Learning Research,* vol. 11, pp. 3137-3181, 2010.

[37] J. Buchli, F. Stulp, E. Theodorou and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research,* vol. 30, no. 7, pp. 820-833, 2011.

[38] F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti and S. Schaal, "Learning Motion Primitive Goals for Robust Manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, USA, 2011a.

[39] M. Anjidani, M. R. Jahed-Motlagh and M. a. N. A. M. Fathy, "A novel online gait optimization approach for biped robots with point-feet," *ESAIM: Control, Optimisation and Calculus of Variations,* vol. 25, no. ESAIM: COCV, p. 29, 2019.

[40] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning,* vol. 8, no. 3, p. 229–256, 1992.

[41] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks,* vol. 21, no. 4, p. 682–697, 2008a.

[42] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research,* vol. 15, no. 1, pp. 319-350, 2001.

[43] R. S. Sutton, D. McAllester, S. Singh and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning

with Function Approximation," in *NIPS'99 Proceedings of the 12th International Conference on Neural Information Processing Systems*, Denver, CO, 1999.

[44] J. Peters and S. Schaal, "Natural Actor-Critic," *Neurocomputing,* vol. 71, no. 7-9, p. 1180–1190, 2008b.

[45] J. Kober and J. Peters, "Policy search for motor primitives in Robotics," *Advances in Neural Information Processing Systems (NIPS 2008),* vol. 21, pp. 297-304, 2008.

[46] F. Stulp and O. Sigaud, "Policy improvement methods: Between blackbox optimization and episodic reinforcement learning," in *Journ´ees Francophones sur la Planification, la D´ecision et l'Apprentissage pour la conduite de syst`emes (JFPDA)*, 2012a.

[47] F. Stulp and O. Sigaud, "Path Integral Policy Improvement with Covariance Matrix Adaptation," in *29 th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012b.

[48] L. P. Selen, D. W. Franklin and D. M. Wolpert, "Impedance control reduces instability that arises from motor noise," *J. Neurosci.,* vol. 7, no. 40, p. 12606–12616, 2009.

[49] F. Stulp, J. Buchli, A. Ellmer, M. Mistry, E. Theodorou and S. Schaal, "Reinforcement learning of impedance control in stochastic force fields," in *2011*, Frankfurt am Main, Germany, 2011b.

[50] F. Stulp, J. Buchli, A. Ellmer, M. Mistry and E. A. T. a. S. Schaal, "Model-Free Reinforcement Learning of Impedance Control in Stochastic Environments," *IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT,* vol. 4, no. 4, pp. 330-341, 2012c.

[51] M. Kobilarov, "Cross-Entropy Randomized Motion Planning," in *In Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.

[52] G. Hoffmann, S. Waslander and C. Tomlin, "Quadrotor Helicopter Trajectory Tracking Control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, 2008.

[53] D. Mellinger, M. Shomin and V. Kumar, "Control of Quadrotors for Robust Perching and Landing," in *International Powered Lift Conference*, Philadelphia, 2010.

[54] F. Jurado, G. Palacios, F. Flores and H. M. Becerra, "VISION-BASED TRAJECTORY TRACKING SYSTEM FOR AN EMULATED QUADROTOR UAV," *Asian Journal of Control,* vol. 16, no. 3, p. 729–741, 2014.

[55] M. Reinoso, L. I. Minchala, J. P. Ortiz, D. Astudillo and D. Verdugo, "Trajectory Tracking of a Quadrotor Using Sliding Mode Control," *IEEE Latin America Transactions,* vol. 14, no. 5, pp. 2157-2166, 2016.

[56] S. Quinlan, "Efficient distance computation between non-convex objects," in *Proceedings IEEE International Conference on Robotics and Automation*, 1994.

[57] M. Jordan and A. Perez, "Optimal Bidirectional Rapidly-Exploring Random Trees," Tech. Rep. MIT-CSAIL-TR-2013-021, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, 2013

[58] Dji, "NAZA-M LITE," 2014. [Online]. Available: https://www.dji.com/naza-m-lite/download. [Accessed 21 September 2017].