SDRE-Based Control Strategy For Quadrotor Landing On A Moving Target

Rishab Balasubramanian, Alvika Gautam, Mandeep Singh and P.B. Sujit

***Abstract*— Vision-based landing can enable precision landing. However, ability to achieve this performance depends on the developed landing controller. In this paper, we present a State Dependent Ricatti Equation (SDRE)-based control strategy for robust landing of a quadrotor on a moving target using vision. The SDRE formulation is simpler and computes the optimal control inputs for the vehicle. Simulations in software-in-the-loop are carried out for different target speeds to shows its ability to land.**

I. Introduction

Quadrotors have become an integral part of many applications like search and rescue [**?**], mapping [**?**], agriculture [**?**], logistics delivery [**?**], etc. Accurate landing is an essential feature that is required in these applications. Apart from landing in constrained environments, it is required that the vehicle needs to land on moving vehicles (targets). Due to the motion of the target, global positioning system (GPS) based landing is inaccurate [**?**] including RTK-GPS [**?**]. Therefore, external aids in the form of IR sensors [**?**] or Lidar [**?**], or camera vision [**?**] is required. The Lidar is bulky and expensive for most quadrotor operations and IR sensors are sensitive to environments features. The camera vision is the most popular due to low cost, passive and ease in integrating with the existing autopilots.

Autonomous landing of aerial vehicles on moving targets involves (a) detection of the landing target using computer vision techniques and (b) determining guidance or controller commands to the vehicle for accurate landing. The are several computer vision techniques that can be used to detect the landing like optical flow [**?**], stereo vision [**?**] [**?**], image segmentation [**?**] [**?**] [**?**] and edge detection [**?**]. Once detected, the target information is then used to determine appropriate guidance or control commands for landing on the target.

Over the years, several landing controllers have been developed. Serrra et al. [**?**] developed an image-based visual servo control to land a quadrotor on a moving target. Falanga et al. [**?**] use vision to detect a tag for quadrotor landing on a moving rover. Cocchioni et al. [**?**] developed different vision algorithms to detect the landing pad patterns for robust landing. Li et al. [**?**] developed a vision algorithm to contain drift for a hover by taking the initial take-off image for a vertical landing. Ghommam and Saad [**?**] developed an adaptive tracking control scheme with backstepping and dynamic surface control for landing a moving target whose position is assumed to be known. Vlantis et al. [**?**] proposed a nonlinear model predictive controller for a quadotor to land on moving robot having inclined landing surface. Hoang et al. [**?**] use visual-bag-of-words method to detect the landing pad and develop a controller for tracking and landing using feedback linearization. Wang and Bai [**?**] use red LEDs to mark the landing pad and use color segmentation to detect the landing pad using vision. A PID control law is used to land on the vertically on stationary target whose pose is simulated as that of a ship. Most of the controllers are nonlinear controller with no optimality guarantees.

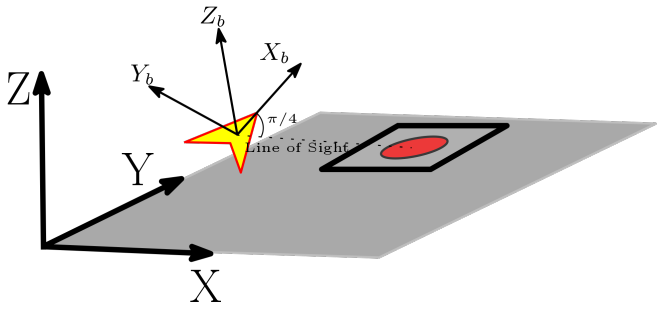


Fig. 1: The figure shows the global axes, and those in the body fixed frame, as well as the orientation of the camera in the setup used

Model Predictive Controller (MPC) is the most common optimal controller that is used for quadrotor landing [**?**], [**?**]. Although, MPC determines optimal solutions, it is computationally intensive and hence difficult to implement for real-time application in the autopilot. Hence, an off-board computer computes the controls and commands the vehicle. Alternatively, the dynamics can be linearized for real-time implementation on a small onboard companion computer [**?**]. In this paper, we consider the complete nonlinear model and determine optimal control inputs by using the nonlinear optimal control framework on State-Dependent Ricatti Equation approach (SDRE) [**?**], [**?**].

In this paper, we propose a vision-assisted SDRE based controller for landing of a quadrotor on a moving target. The SDRE controller determines optimal control inputs and the structure is sufficiently simple for real-world implementation on a onboard computer. The stability of the controller is given and we show the efficacy of the approach through software-in-the-loop simulations.

The paper is organized as follows. In Section II we define the objective and the problem statement. In Section III, we determine the SDRE controller formulation and in Section IV we determine the position of the target. In Section VI

Rishab Balasubramanian is an under graduate student at NIT Thiruchirapalli, Email: rishab.edu@gmail.com

Alvika Gautam is a Post-Doctoral Fellow at Brigham Young University, Provo, Utah, UT-84602, Email:

we implement the determined SDRE controller in Gazebo software-in-the-loop simulation and show the results. We conclude in Section VII.

II. Problem Formulation

Consider a UAV that needs to land on a target as shown in Figure 1. A camera is mounted on the vehicle which is inclined at an angle of $\pi/4$ with respect to the body of the UAV. We use the image information from the camera with a Kalman filter to estimate the position and velocity of the moving landing target on the ground. The errors in the body-fixed frame of the UAV are computed to generate the desired controller commands.

We consider the problem as an infinite horizon regulator problem. The distance errors in the body frame of the UAV can be defined as states of a regulator system, which should converge to zero. To ensure that the UAV lands on the target, it is necessary for the relative velocities of the two to be equal at the time of landing. We develop an optimal guidance strategy based on SDRE formulation. We assume a decoupled form of the control to simplify its application.

The states are errors in the x, y, and z directions of the body fixed frame are e_x, e_y, e_z respectively. The input to UAV are $u, \theta,$ and ϕ representing normalized thrust, pitch and roll angles respectively. We assume that the autopilot has the feature to convert the thrust, and desired roll and pitch angles to the speed of the four rotors.

Let us consider frame A to be the global frame and frame B to be the body fixed frame. The only force acting on the UAV, apart from the thrust generated by its rotors, is the force due to gravity in the z direction of the global frame $-mg$ (assuming no external disturbances). It can be transformed to the body fixed frame B by multiplying with a rotational matrix R_A^B to get:

$$W_B = R_A^B \begin{bmatrix} mg\sin(\theta) \\ -mg\sin(\phi)\cos(\theta) \\ -mg\cos(\phi)\cos(\theta) \end{bmatrix}, \tag{1}$$

where, W_B is the force due to gravity in frame B and the R_A^B is the rotation matrix from Global frame A to the body frame B, which is given as

$$R_A^B = \begin{bmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + s(\phi)s(\theta)c(\psi) & \\ & s(\psi)s(\phi) + c(\phi)c(\psi)s(\theta) & \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\phi)s(\theta)s(\psi) & \\ & -s(\phi)c(\psi) + s(\psi)s(\theta)c(\phi) & \\ -s(\theta) & c(\theta)\sin(\phi) & c(\theta)c(\phi) \end{bmatrix}, \tag{2}$$

The angles θ and ϕ represent pitch and roll angles respectively. The notions $\cos(x)$ and $\sin(x)$ are denoted as $c(x)$ and $s(x)$ respectively.

Using small angle approximation, we can rewrite Equation (1) as

$$W_B = \begin{bmatrix} mg\theta \\ -mg\phi \\ -mg \end{bmatrix}. \tag{3}$$

We now consider the thrust (T) that acts only along the z direction of frame B. Thus,

$$F_z = T - mg = mu_1, \tag{4}$$

where $u_1 = (T - mg)/m$. u_1 is one of the control inputs.

The accelerations thus become

$$a_B = \begin{bmatrix} g\theta \\ -g\phi \\ u_1 \end{bmatrix}. \tag{5}$$

We then write out linearized state equations as

$$\dot{x} = Ax + Bu, \tag{6}$$

where,

$$x = \begin{bmatrix} e_x \\ \dot{e}_x \\ e_y \\ \dot{e}_y \\ e_z \\ \dot{e}_z \end{bmatrix}, u = \begin{bmatrix} u_1 \\ \theta \\ \phi \end{bmatrix}, A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -g & 0 \\ 0 & 0 & 0 \\ 0 & 0 & g \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix},$$

where, g is the gravitational constant ($|g| = 9.8$) and e_x, e_y, e_z are the distance to the target along X, Y, Z axes of frame B, and $\dot{e}_x, \dot{e}_y, \dot{e}_z$ are their first derivatives (error in velocity between the target and the UAV) respectively.

III. SDRE Control Law

We obtain the optimal control input to the system by solving the Ricatti equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0, \tag{7}$$

$$u = -R^{-1}B^T Px, \tag{8}$$

where u is the desired input.

The R matrix is selected so as to limit the pitch and roll angles to a small range such that the small angle approximation holds valid. The Q matrix is selected so as to reduce overshoots in x and y directions and also to ensure that the drone descends as it follows the target to ensure optimal landing.

During the simulations we use Q and R as:

$$Q = \begin{bmatrix} \frac{25e_x^2}{|e_z|}+1 & 0 & 0 & \\ 0 & 0 & 0 & 0 \\ & & & \\ 0 & \frac{150(0.5+|e_z|)\dot{e}_z}{0.1e_x} & 0 & \\ & 0 & 0 & 0 \\ & & & \\ 0 & 0 & \frac{25e_y^2}{|e_z|}+1 & \\ & 0 & 0 & 0 \\ & & & \\ 0 & 0 & 0 & \\ & \frac{150(0.5+|e_z|)\dot{e}_z}{0.1e_y} & 0 & 0 \\ & & & \\ 0 & 0 & 0 & \\ & 0 & 1+\frac{10e_z}{\sqrt{0.01(e_x^2+e_y^2)}} & 0 \\ & & & \\ 0 & 0 & 0 & \\ & 0 & 0 & \frac{1}{|e_z|} \end{bmatrix},$$

$$R = \begin{bmatrix} 800 & 0 & 0 \\ 0 & 75000 & 0 \\ 0 & 0 & 75000 \end{bmatrix}.$$

For the distance errors e_x and e_y the weighting parameters in Q are chosen such that they are inversely proportional to e_z so that the UAV does not move very close to the target when it is very high up and subsequently lose the it from the field of view. The weights for the velocity errors were then chosen so as to reduce overshoots

IV. Computing the position of the target

A grey colored landing space is placed on top of the rover, on which a red colored target is attached. The input video from the camera is converted from RGB to HSV and YCrCb color spaces. This is because of the robustness of these color spaces to lighting conditions over RGB. In the HSV model, colors of each hue are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top. Hue thus defines the dominant color of an area. The saturation dimension resembles various tints of bright color measuring the colorfulness, and the value dimension resembles luminance. The Hue, Saturation and Value are calculated as:

$$H = \frac{\arccos(0.5(R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}, \quad (9)$$

$$S = 1-3\frac{\min(R,G,B)}{R+G+B} \quad (10)$$

$$V = \frac{R+G+B}{3} \quad (11)$$

In the YCrCb color space the color is represented by luma Y, constructed as a weighted sum of RGB values, and two color difference values Cr and Cb that are formed by subtracting luma from RGB red and blue components.

$$Y = K_rR+K_gG+K_bB, \quad (12)$$

$$Cr = R-Y, \quad (13)$$

$$Cb = B-Y. \quad (14)$$

In each computed space, a mask is applied to remove only the regions which contain red color. To remove noise, morphological transforms are applied. The two resulting masks are then combined to obtain the final mask of the image. On this, blob detection algorithm is applied to find the largest contour as in Figure 3(b)

After blob detection, the center of the contour is calculated. Once the center of the target is found, the coordinates received in pixel values are then transformed to coordinates in the body fixed frame by a homography matrix H.

A. Computing the Homography matrix

Let us define the homographic matrix H as:

$$H = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix},$$

and $H^{'}$ as:

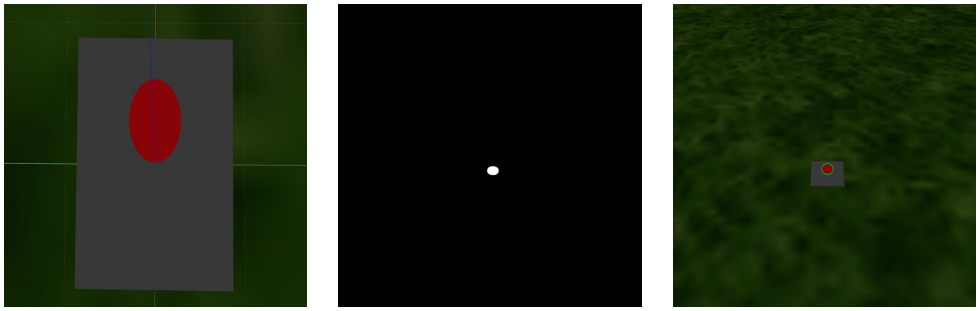
$$H^{'} = \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix}.$$

Let the image co-ordinates be (a_1, b_1) and the global co-ordinates be (X_1, Y_1). Then we relate the two as:

$$\begin{bmatrix} a_1 \\ b_1 \\ 1 \end{bmatrix} \approx \begin{bmatrix} wa_1 \\ wb_1 \\ w1 \end{bmatrix} = H \begin{bmatrix} X_1 \\ Y_1 \\ 1 \end{bmatrix}.$$

The homography matrix H has 9 variables, and hence to solve for these values we need 9 equations. This is done by using five directional vectors, in the body frame, that correspond to (0,0), (w,0), (0,h), (w,h), (w/2, h/2) in the image frame, where w and h are the width and height in pixels of the image. These vectors are multiplied with the rotational matrix, R_A^B to obtain the directional vectors in the world frame. Once the directional vectors are found in the world frame, their intersection with the $z = z_0$ plane (where z_0 is the height of the rover) is computed. Consider the intersection points as (X_i, Y_i, z_0) and the corresponding image coordinate as (a_i, b_i). The matrix C is then computed using each point i as follows:

$$C = \begin{bmatrix} X_i & Y_i & 1 & 0 & 0 & 0 & -aX_i & -aY_i & -a \\ 0 & 0 & 0 & X_i & Y_i & 1 & -bX_i & -bY_i & -b \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \end{bmatrix}. \quad (15)$$



(a) (b) (c)

Fig. 2: a) The landing zone atop the rover with a red circular visual target b) Extracted mask of the red target c) The detected position of the target

Once generated the entries of the homography matrix are then computed by solving the homogeneous equation:

$$CH^{'} = 0. \tag{16}$$

This can be done by using Singe Variable Decomposition (SVD). Once the homography matrix is determined, the image coordinates can be transformed to determine the position of the target as:

$$X = \frac{(m_{00}a + m_{01}b + m_{02})}{(m_{20} + m_{21} + m_{22})}, \tag{17}$$

$$Y = \frac{(m_{10}a + m_{11}b + m_{12})}{(m_{20} + m_{21} + m_{22})}, \tag{18}$$

where (X, Y) are the errors (distance between UAV and target) in X and Y of the world frame and (a, b) are the image coordinates of the target.

V. Kalman Filter

A Kalman Filter is applied over the global coordinates obtained from the vision system:

$$\underbrace{\begin{bmatrix} x_{k+1} \\ vx_{k+1} \\ y_{k+1} \\ vy_{k+1} \end{bmatrix}}_{X_1} = \underbrace{\begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 1 & 0 & 1 \end{bmatrix}}_{\mathrm{F}} \underbrace{\begin{bmatrix} x_k \\ vx_k \\ y_k \\ vy_k \end{bmatrix}}_{X_k} \tag{19}$$

where $k+1$ denotes the next time step and x, vx, y, vy denote the global x position, x velocity, y position, y velocity respectively. F is the co-efficient matrix, X_k is the previous estimated states and X_1 is the prediction matrix. Then

$$P_1 = FP_kF^T + Q, \tag{20}$$

where P is the co-variance of the filter output and Q is the noise matrix for the filter. The Kalman gain is then computed as:

$$K = P(P + R)^{-1}, \tag{21}$$

where R is the noise in the measurement. The new states and co-variance are thus calculated as:

$$X_{k+1} = X_1 + K(Z - X_1), P_{k+1} = P_1(1 - K), \tag{22}$$

where, X_{k+1} is the new predicted state and P_{k+1} is the co-variance.

Figure 3a shows the errors between estimated and actual position. We can observe that the error is negligible and approaches to zero as we near the rover. Figure 3b shows the errors between estimated and actual velocity. We can observe larger errors towards the starting and towards the end of the flight. These are positions where the drone rotates to accelerate/decelerate, causing the camera to rotate as well. This produces larger errors, which reduce over time due to the kalman filter.

VI. Simulation Results

We have validated the performance and effectiveness of the developed SDRE control using Gazebo-ROS.

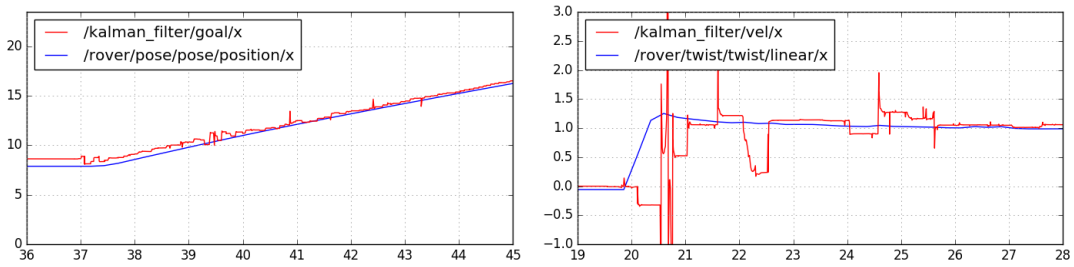
A. Simulation Setup

In order to validate the controller we use ROS (Robotics Operating System) for developing the algorithm and Gazebo for the 3D visualization and simulations. OpenCV is used for the computer vision section. All the simulations were run on a HP laptop with Intel i5 processor and 8 GB RAM.

The algorithms were developed using python scripts and the detection, estimation and control algorithms were run parallely. In the simulations, the quadrotor used has a on-board IMU and a camera to detect and estimate the position and velocity of the ground vehicle. We assume that it is not possible to establish communication with the rover and hence in these experiments we do not rely on the GPS or odometry data from the rover but rather just the camera feedback from the drone. The camera was oriented at and angle of 45°with respect to the UAV, facing downwards. The resolution was assumed as 1000*1000 with the FOV being 60°.

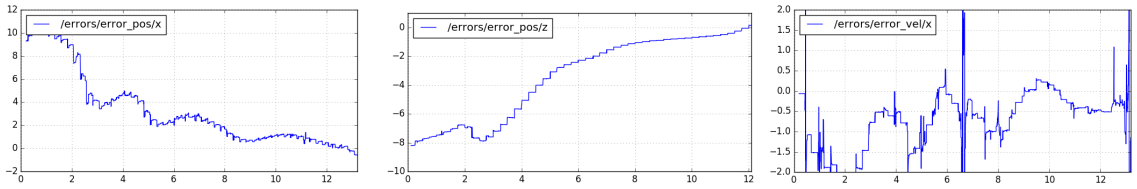
B. Results

Figure 5a shows the position error of the target to the UAV along the X axis. We can observe that this error rapidly decreases. There are however some sudden peaks in the error calculated. These are the points where the UAV is decelerating so that the velocity of the target can be matched at the time of landing. Figure 5b shows the vertical position error, which can be seen to be largely negative initially, and



(a) (b)

Fig. 3: Estimated and actual (a) Position (b) Velocity when landing on a rover moving along the X axis with speed 1m/s



(a) (b) (c)

Fig. 4: Errors (a) in position along X direction (b) in position along Z direction (c) in velocity along X direction between the rover and drone while landing. The velocity of the rover is 1m/s.

gradually decreases to zero. Figure 6 shows the trajectory followed by the UAV as it lands on a rover moving straight with a velocity of 1,2,3 and 4m/s respectively.

VII. Conclusions

A decoupled SDRE-based controller was developed for a quadrotor landing on a moving target. The proposed control law was verified in simulation through software-in-the-loop. The simulations show the ability of the proposed method to follow and land on different target speeds. The future work involves extending the algorithm to path following for the quadrotor, and including the yaw, so as to allow landing on the target along a particular direction. Also, perform outdoor experiments to validate the approach.

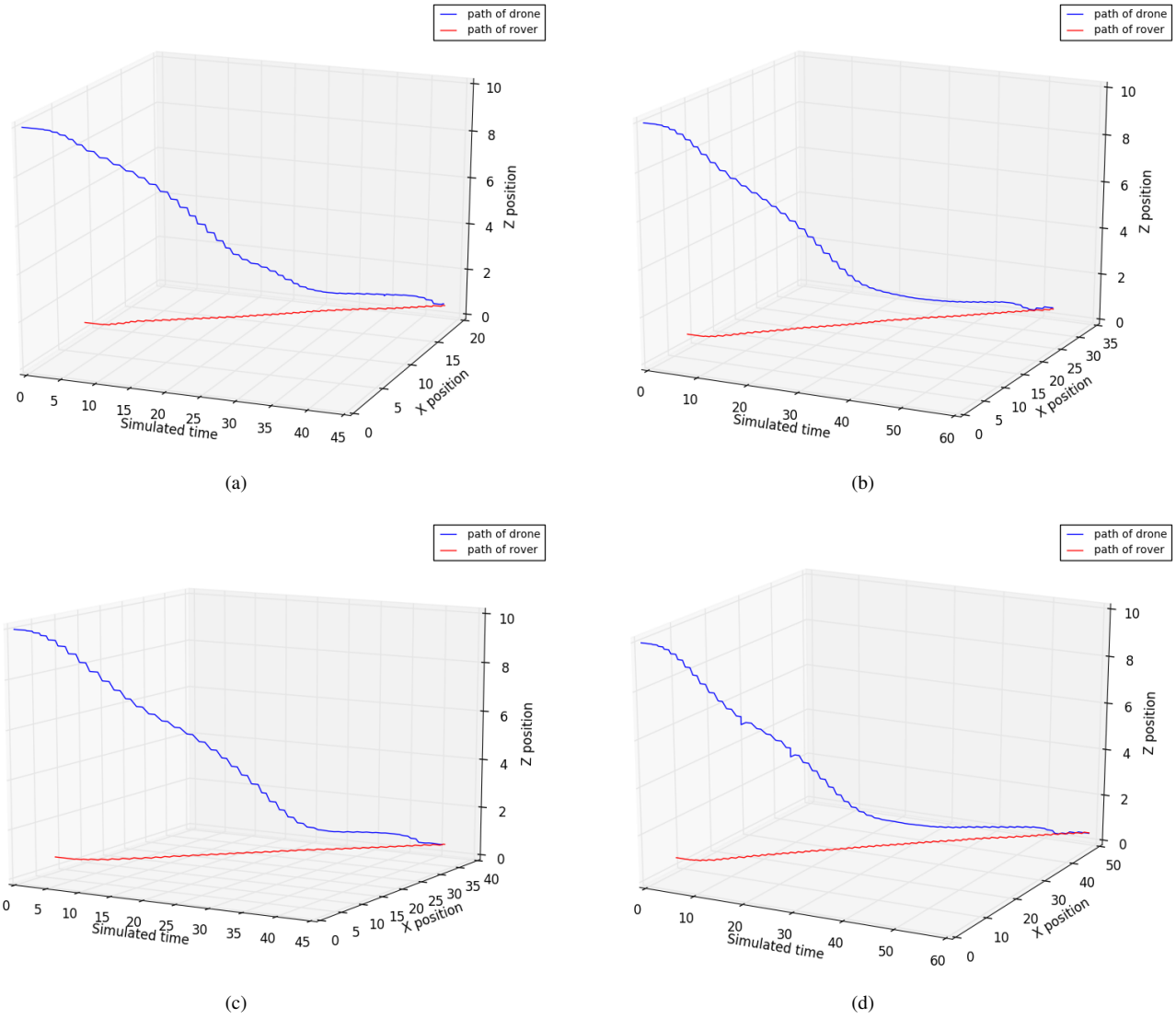


Fig. 5: Path followed by the quadrotor and ground rover for linear velocity of 1, 2, 3, 4 m/s for the rover. The time shown here is the simuation time. The real time is $t \approx 0.23t_0$ where t is in seconds and t_0 is the simulation time