Control and Guidance of an Autonomous Quadrotor Landing Phase on a Moving Platform

M. S. Ale. Isaac*, A. Naghash, and S. H. Mirtajedini
Amirkabir University of Technology, 424 Hafez Ave, Tehran, Iran

ABSTRACT

This summary describes an application of a vision-based implementation of three control algorithms to a rather light quadrotor to land on a moving platform with a random path and unknown velocity. Comparing sliding mode (SM), PID, and model reference adaptive (MRAC) controllers in both MATLAB SimScape synthetic space and real-world, we proved the superiority of the former one. The guidance method is an online tracking which uses a linear regression to estimate the landing point. We used a state-of-the-art visual odometry algorithm, SVO, augmented by IMU to correct the path angle. No prior information about the quadrotor or the landing platform is required.

1 INTRODUCTION

Quadrotor landing phase has been a salient research challenge in recent years. The challenge will arise when the landing platform is a moving object which constantly changes its course on a randomly generated path. By enabling a quadrotor to land on such a platform in a robust and smooth manner, this will be more prepared to be deployed by moving machines, such as next generation of autonomous cars, boats an even planes. Besides, This ability has various benefits, include, faster charging for more flight endurance, mapping, search, rescue, and assistance mobile objects [1, 2, 3]; explicitly, this has been used in robotic challenges like IMAV competitions. The bottlenecks contain, first, detecting the platform and find its position, as well as estimating a reliable spot to be considered as a goal for our robot to touch the moving platform; second, implementing a control algorithm which is able to track the platform and in the meantime, immune to disturbances that, in practice, are imposed to the plant during the landing.

1.1 Related Work

There are a few thrived projects in the same submission, Lee et al. ponder a line of sight (LOS) algorithm for the guidance section, compounding with a conventional controller (e.g. PID) will flourish [4]; however, following in this situation requires a fairly large camera with a wide field of view, in order not to miss the moving platform, and consequently, is not operational easily. Falanga et al. worked on a cascade controller (compounding of two PID controllers with the feedback of states, velocities, and accelerations), compiled on an onboard computer, equipping with state estimation and path planning [1], but lacking random target estimation.

1.2 Contribution

In this paper we propose a quadrotor system which detects, follows, and lands on a moving platform just using an onboard computer. The merit of our work is summarized in measuring the platform's random positions online and then, fitting a convenient curve on previous points. By calculating the polynomial coefficient on the curve, it will produce the new point, based on the time step and estimated velocity of the platform. No prior information about neither velocity nor position and not even the path of the moving platform or the quadrotor is needed. Every point under the quadrotor will be covered and mapped, then the localization process will start.

2 SYSTEM OVERVIEW

Here, we will describe following items:

- Control and guidance;
- Position estimation;
- Landing platform detection;
- Virtual platform;
- Experimental platform.

2.1 Control and Guidance

This subsection is devided into 4 segments:

- Sliding Mode algorithm;
- PID algorithm;
- MRAC algorithm;
- Navigation.

Comparing aforementioned controllers, we investigate which ones fulfill the following requirements: stability in both descending and landing phase, resistance to the probable noises, either internal or external, faster response to a sudden and random path deviation, and ability to thrust again right after landing. Consequently, we put our attention into considering

*The authors are with the Micro Air Vehicle Group, Amirkabir University of Technology—http://autmav.com Email address(es): sadegh_al@aut.ac.ir, naghash@aut.ac.ir, sehomi@aut.ac.ir

all nonlinear terms of the system, not just in controllers nor the main dynamic model of the system, except the PID controller, which has a linear base and so conducive to linearizing the system. Worthwhile, the main strategy for compounding each controller with the navigation algorithm is based on immediate reaction of the plant to fly over the moving platform and keep lock its sight on until landing. Besides, the controllers are divided into inner and outer loops [5]; the reference values are x_{ref}, y_{ref}, z_{ref}, and ψ_{ref} which are determined by the guidance law and computing the angle of plant trajectory, using tangent inverse. Moreover, state variables, x and y are controlled in the outer loop and in counterpart ϕ, θ, and ψ are controlled in the outer loop, and z is controlled separately, but not in this division. Thereby, the slow dynamic equations could be considered [3, 6] as:

$$\begin{cases} \ddot{x} = \frac{F_z}{m}(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ \ddot{y} = \frac{F_z}{m}(\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi) \end{cases} \tag{1}$$

Solving equations (1), we have:

$$\begin{cases} \phi_{des} = \frac{m}{F_z}(\ddot{x}\sin\psi - \ddot{y}\cos\psi) \\ \theta_{des} = \frac{m}{F_z}(\ddot{x}\cos\psi + \ddot{y}\sin\psi) \end{cases} \tag{2}$$

Equations (2) state that the outer controller loop could be nonlinear and the desired attitudes depend on the thrust force, longitudinal and lateral accelerations, and yaw angle; if so, using a PD controller instead all above will satisfy our criteria; however, using PID, MRAC or SM methods have latency because of their integral components and is so against our goal to be fast responding in transient phase. For fast dynamic [3] we have:

$$\begin{cases} \ddot{z} = \frac{U_1}{m}(\cos\phi\cos\theta) \\ \ddot{\phi} = \dot{\theta}\dot{\psi}\frac{Iy-Iz}{Ix} + \frac{J_r}{I_x}\dot{\theta}\Omega_r + \frac{l}{I_x}U_2 \\ \ddot{\theta} = \dot{\phi}\dot{\psi}\frac{Iz-Ix}{Iy} + \frac{J_r}{I_y}\dot{\phi}\Omega_r + \frac{l}{I_y}U_3 \\ \ddot{\psi} = \dot{\phi}\dot{\theta}\frac{Ix-Iy}{Ix} + \frac{1}{I_z}U_4 \end{cases} \tag{3}$$

U_1 is the total thrust force which equals F_z and other $U_2, U_3, and U_4$ are the roll moment, pitch moment, and yaw moment respectively. Meanwhile, J_r is the rotor gyroscopic inertia and Ω_r is the rotor angular velocity.

2.1.1 *Sliding Mode Algorithm*

To compute the switiching surface for any system with the degree of n, we have:

$$x^{(n)} + f(x) = u \rightarrow S(x,t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} e \tag{4}$$

The quadrotor is a second order system, consequently, all the nonlinearities which refer to the $f(x)$ equal zero [7]. Computing the equal energy for reaching the switching surface u_{eq}, and determining total energy u we have:

$$\begin{cases} u_{eq} : \dot{S} = 0 \rightarrow x^{(n-1)} - {x^{(n-1)}}_d + \lambda\dot{e} = 0 \\ \rightarrow u_{eq} = {x^{(n-1)}}_d - \lambda\dot{e} \\ u = u_{eq} - Ktanh(S) \rightarrow u = {x^{(n-1)}}_d - \lambda\dot{e} - K\tanh(S) \end{cases} \tag{5}$$

K value refers to a discontinuous component against system noises, which is calculated from try and error, means if it more than a determined magnitude, the system will be stable. This is derived from the fact that how far negative is the Lyapunov function derivative, it will converge to a value more negative and so will be stable faster. In addition, instead of $sign$, we use $tanh$ function to make the chatterings of the switching surface more smooth, so nor requires integration of swithcing surface. Totally, the controller will be designed as hereunder:

$$\begin{cases} \ddot{x}_d = -\lambda\dot{e}_x - K\tanh(S_x) \\ \ddot{y}_d = -\lambda\dot{e}_y - K\tanh(S_y) \\ U_1 = m(\ddot{z}_d - \lambda\dot{e}_z) - K\tanh(S_z) \\ U_2 = \frac{I_x}{l}(\ddot{\phi}_d - \lambda\dot{e}_\phi) - K\tanh(S_\phi) \\ U_3 = \frac{I_y}{l}(\ddot{\theta}_d - \lambda\dot{e}_\theta) - K\tanh(S_\theta) \\ U_4 = I_z(\ddot{\psi}_d - \lambda\dot{e}_\psi) - K\tanh(S_\psi) \end{cases} \tag{6}$$

λ values are computed in the simulation and then corrected by implementation results.

2.1.2 *PID Algorithm*

Using a PD as outer loop and PID for inner one, we have:

$$\begin{cases} \ddot{x}_d = K_{d_x}\dot{e}_x + K_{p_x}e_x \\ \ddot{y}_d = K_{d_y}\dot{e}_y + K_{p_y}c_y \\ U_1 = K_{d_z}\dot{e}_z + K_{p_z}e_z + K_{i_z}\int e_z \\ U_2 = K_{d_\phi}\dot{e}_\phi + K_{p_\phi}e_\phi + K_{i_\phi}\int e_\phi \\ U_3 = K_{d_\theta}\dot{e}_\theta + K_{p_\theta}e_\theta + K_{i_\theta}\int e_\theta \\ U_4 = K_{d_\psi}\dot{e}_\psi + K_{p_\psi}e_\psi + K_{i_\psi}\int e_\psi \end{cases} \tag{7}$$

The two former equations are related to the outer controller loop which helps us computing the desired longitudinal and lateral accelerations.

2.1.3 *MRAC Algorithm*

The adaptation law is based on the trajectory following. We introduce a second order system which must adapt the model to the reference model [7, 6]. The chosen model, reference model, and the adaptation law are, respectively:

$$G(s) = \frac{1}{s(s+a)} \tag{8}$$

$$G_m(s) = \frac{w^2}{s^2+2\xi\omega s+\omega^2}$$
$$\rightarrow \ddot{x}_m + \underbrace{2\xi\omega}_{a_1} x_m + \underbrace{\omega^2}_{a_2} x_m = \underbrace{\omega^2}_{b} u_c \quad (9)$$
$$u = \theta_1 u_c - \theta_2 \dot{x} - \theta_3 x \quad (10)$$

a is the estimation parameter to adapt our model to the reference model. ξ and ω are damping ration and system frequency, and θ_is are values we compute to update the adaptation law. Differencing equations (5) and (6), adding $a_1\dot{x}+a_2x$ term to both sides of the equation, and simplifying, we have:

$$e = \frac{1}{s^2+2\xi\omega s+\omega^2}\left(\begin{bmatrix} \dot{x} & -x & u_c \end{bmatrix}\begin{bmatrix} \widetilde{\theta}_2 \\ \widetilde{\theta}_3 \\ \widetilde{\theta}_1 \end{bmatrix}\right) \quad (11)$$

The system is not strictly positive real (SPR); therefore, we cannot use Kalman Yakubovich Lemma [5] and use state space equations to solve the system. Hence:

$$A = \begin{bmatrix} O_{6\times6} & I_{6\times6} \\ -a_2 & -a_1 \end{bmatrix} = \begin{bmatrix} O_{6\times6} & I_{6\times6} \\ -w_i^2 & -2\xi_i\omega_i \end{bmatrix} \quad (12)$$

ξ_i and ω_i magnitudes are exploited by various tests. To calculate errors in state space form, we have:

$$\begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} = A\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + B\underbrace{\begin{bmatrix} -\dot{x} & -x & u_c \end{bmatrix}}_{\bar{\Phi}}\begin{bmatrix} \widetilde{\theta}_2 \\ \widetilde{\theta}_3 \\ \widetilde{\theta}_1 \end{bmatrix} \quad (13)$$

$$\begin{bmatrix} \dot{\widetilde{\theta}}_2 \\ \dot{\widetilde{\theta}}_3 \\ \dot{\widetilde{\theta}}_1 \end{bmatrix} = -\Gamma\bar{\Phi}^T B^T P\begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} \quad (14)$$

Γ and P are symmetric positive-definite matrixes which defined to satisfy $A^TP + PA = -Q$, and Q is the same as P. The Lyapunov candidate function proof of stability comes in the Appendix.

2.1.4 *Navigation in cases of visible platform and temporarily lost platform*

We use a simple method for both tracking and landing on moving platform. Explicitly, because of unknown velocity or path pattern of the object, we consider a minuscule component of the time, in order to reduce computation, especially in implementation. For sake of estimating the touch down spot (the estimated landing point), a second order regression is implemented, which takes a buffer of last 15 positions of the platform trajectory into account, and as a result, the coordinates on the fitted curve at a certain number of time-steps (in our case, 5) after the current platform position is considered as the estimated landing point. In other words, it is expected that the quadrotor and the moving platform will meet, on this calculated position. Calculating such a point seems to be an obligation due to the fact that in vision-based, precise landing, scenarios such as the present work, the platform will get out of sight as soon as the camera gets closer than a threshold (in our case, it is 0.3 m). From this point onward, the robot needs to blindly reach the estimated landing point and the reliability of this estimation as well as the accuracy of both, robot estimated position and platform detected position comes into play, which will be explained in proceeding sections.

2.2 *Position Estimation*

A very necessary objective of our drone is to maintain its stability even in case of losing its landing target so position control is still active and the procedure of testing will be less hazardous. The prerequisite of a great position control is to have position feedback. Many common sensors used for having position feedback are GPS, IMUs, Infrared Markers, Radio Beacons and motion capture systems (MOCAP). For a quadrotor to be truly autonomous, all computations of position estimation must be onboard and since we are planning for the precise landing on a moving platform, the position feedback also needs to be both precise and enough accurate. Here we implement one of the most absolute methods of position and attitude estimation, close to ground or rather featureful environments, referred to as visual inertial odometry (VIO) methods. A comparison between state-of-the-art VIO approaches could be found in [1].
Here we chose the Semi-Direct Visual Odometry (SVO) [8] algorithm for position estimation of our quadrotor. This algorithm is compiled on an Odroid XU4 companion computer and a forward-looking camera. The reason for the forward-looking camera setup is to prevent any position estimation error when the drone is close to the landing platform because in down-looking setup most of the camera field of view will be filled with the platform itself.
We have tested the precision of the system with two configurations in a scenario close to our objective, to test the limits of the estimation algorithm for our specific setup. The scenario is moving the camera on a sine-like path close to the ground with changing the camera heading so that the camera view is being changed constantly.

- **First configuartion**: the algorithm runs in monocular configuration and the trajectory in the XY plane is shown in "Figure 3". The trajectory is compared to a ground truth waypoint path and the result is showing that there is a huge difference between two trajectories. The SVO trajectory is scaling down as the camera is moving further which could result in any unpredictable behavior of the control system. Similar issues have been reported for implementations of the algorithm with different camera models, especially for non-global shutter cameras. This could be explained by the blurriness occurring in images when the non-global shutter cameras are rotating and changing the

view and the algorithm is unable to estimate the camera rotations properly. Generally, the visual odometry (VO) has problems in "Pure Rotation" movements. A discussion about it can be found in [9, 10].

- **Second configuartion**: one way to work around the errors due to camera rotations and changes in camera view (especially in non-global shutter cameras) is the integration of an IMU system as the SVO has its own extended Kalman filter (EKF) [8] running at 200 Hz. So in this configuration, an IMU data at 150 Hz is provided to the filters and with the same dataset, the results are shown in Figures 4, 5. Note that the trajectory of SVO is more close to the ground truth waypoint path and its scale is not decreasing as the camera moves further. Only the trajectory is slightly getting away from the Ground Thruth but still, this position feedback is good enough for controlling the quadrotor positions in landing phase because the quadrotor target is to follow the landing platform and because of that, even small drifts in position will not make our precise landing fail.

2.3 Landing Platform Detection

For the purpose of detecting the Moving Platform, a down-looking camera is mounted on the quadrotor capturing images at a rate of 10 Hz. The target on the moving platform must be a standard and easy to detect marker, so we have used an Augmented Reality markers board (AR), which is shown in fig 4, and the ArUco module of OpenCV library[5] to detect both the position and orientation of the landing platform in the down-looking Camera frame $\{\mathbf{C_d}\}$ By performing a homogeneous transformation from down-looking camera frame $\{\mathbf{C_d}\}$ to the forward-looking camera $\{\mathbf{C_f}\}$ (the camera which is used for SVO) and then using another homogeneous transformation from forward-looking camera $\{\mathbf{C_f}\}$ to the world frame $\{\mathbf{W}\}$, obtained from position estimations of SVO, we will have the landing platform position in the world frame.

2.4 Virtual Platform

We built a complete process of our mission in the powerful MATLAB SimScape simulator. All the three controllers, moving platform, and cameras are simulated to compare their performance and achieve a precise implementation [11, 12]. Meanwhile, most of the controllers' gains are set in the simulator and then, corrected in real-world. Besides of the pros of MATLAB simulator, such can be easily done, friction and noise included, model-based dynamics and so forth, there are a few defects, like weak and difficult collision avoidance simulation, no detection probability. Considering all merits and demerits we suppose optimum detection of the downward camera and make the dynamic model of the quadcopter in the platform. No dynamic equation is needed to build the model, just by exporting from CATIA or SolidWorks, either a .xml or .STL file, to the MATLAB software. Based on our knowledge, the sliding mode then, PID, and finally the MRAC controllers keep the plant more stable, respectively. Specifically, when the stochastic noises grow or the object velocity increases, or even when the standard deviation of the random path (σ) moves upright, comparison result will be more observable that the sliding mode controller works really spectacular. Some of the best virtual results are shown in Table 1, and Figure 1 shows the the drones which are based on three controller methods in the simulator; besides, the results of Simulink with $2.5m/s$ are shown in Figures 6, 7:

A (m/s)	Controller	B (cm)	C (cm)
0.5	SM	8	2
	PID	10	3
	MRAC	15	13
1.5	SM	10	8
	PID	15	12
	MRAC	26	15
2.5	SM	11	13
	PID	14	17
	MRAC	66	82

Table 1: Results of the virtual test. Note that **A** refers to the moving platform velocity, **B** mentions the longitudinal deviation with the platform center, and **C** refers to the lateral deviation with the platform center.

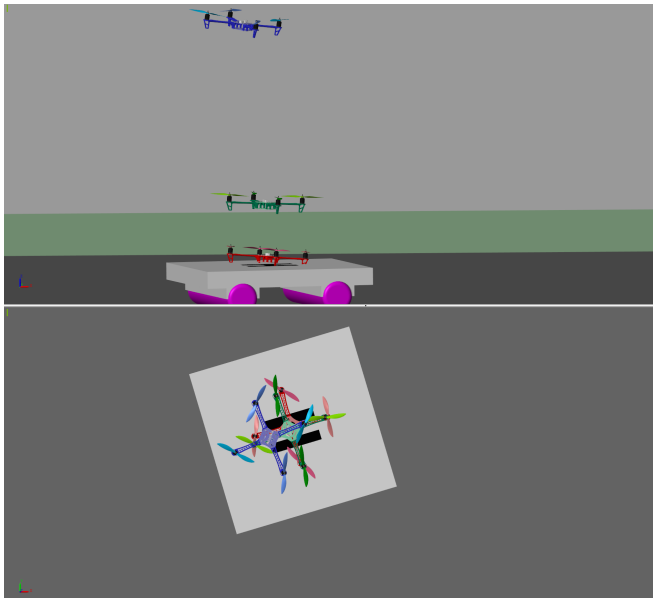

Figure 1: Comparison among virtual drones with three controllers in MATLAB Simscape dynamic space. The red drone refers to PID, the green one to SLD, and blue drone to MRAC controller.

2.5 Experimental Platform

In this submission, we put all our attempts into finding the best practical controller; regardless of any experimental result, the PID works perfect because of fewer coefficients we must tune and thanks to the proper sensors giving feedback of position, velocity, and accelerations. The most challenging section of implementation we faced, could be summarized in localization of the drone and if so very well, we will be able to do various missions. Therefore, we could not test very random cases because a rectangular or a circular random, supposedly, are difficult for our algorithms so we limited deviations. Moreover, when the trajectory of the object is random, its velocity impacts a lot on detection because it might conducive to losing the platform and causing miscalculation even in the landing process of the drone, so we did not test with more than $1.5m/s$ velocity. Some of the best results are shown in Table 2 and Figures 8, 9, 10, 11; also, the landing platform pattern is shown in Figure 2.

A (m/s)	Controller	B (cm)	C (cm)
0.5	PID	7	5
	SM	10	5
	MRAC	25	16
1.5	PID	12	10
	SM	21	12
	MRAC	30	42

Table 2: Results of the practical test in real-world. **A** refers to the moving platform velocity, **B** refers the longitudinal deviation with the platform center, and **C** refers to the lateral deviation with the platform center.

3 CONCLUSION

In this paper, we introduced a fully autonomous quadrotor landing on a moving platform even if it moves on a random path. During the work, three control algorithms are compared to find the best one. To the best of our knowledge, compounding of a PD controller (for inner loop) and SM (for outer one) does better in the simulation, but a little lax in the implementation because of its inordinate coefficients those cannot be tunned practically wholly. Notwithstanding, the fair performance of the SM and MRAC, PID is hardly deniable; this works perfectly in both virtual and real-world.
To continue, we compiled a fantastic visual odometry algorithm (SVO) on the plant for mapping, detecting the ARCode installed on the surface of the moving platform, and tracking. There is no need for any prior information about platform velocity, quadrotor location, and even the path line. No need to a special strategy, when missing the object; just using a simple 2D regeression based on last considerations and estimating a new point as landing target.

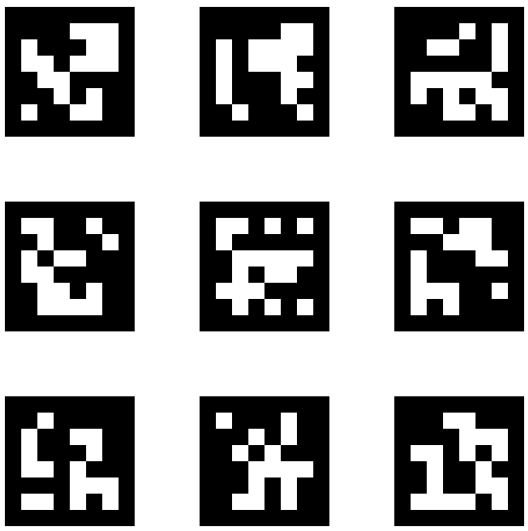

Figure 2: The ARCode icon sheet which is installed on the moving platform to be detected

ACKNOWLEDGEMENTS

The authors would like to thank N. Shahsavari for camparing the PTAM algorithm with our chosen counerpart, A. Yazdanshenas for creating a base CAD model, the Aerospace AUTMAV laboratory adminisrator for the use of a long time indoor testbed, and all of its members for their influential insights.

REFERENCES

[1] Davide Falanga, Alessio Zanchettin, Alessandro Simovic, Jeffrey Delmerico, and Davide Saramuzza. Vison-based autonomous quadrotor landing on a moving platform. In *International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2017.

[2] Matthias Schreier. Modeling and adaptive control of a quadrotor. In *IEEE International Conference on Mechatronics and Automation*, 2012.

[3] Zachary T. Dydek, Anuradha M. Annaswamy, and Eugene Lavretsky. Adaptive control of control uavs: A design trade study with flight evaluation. *IEEE Transactions on Control Systems Technology*, 42(13566251):1400–1406, 2012.

[4] Hanseob Lee, Seokwoo Jung, and David Hyunchul Shim. Vison-based uav landing on the moving platform. In *International Conference on Unmanned Aircraft System (ICUAS)*, 2016.

[5] Tammaso Bresciani. *Modeling, Identification, and Control of a Quadrotor Helicopter*. MSc Theses, Department of Automatic Control, Lund University, 2008.

[6] Andrew Zulu and Samuel John. A review of control algorithms for atonomous quadrotors. *Open Journal of Applied Sciences*, 28(12524328):77–89, 2011.

[7] Eugene Lavretsky and Kevin Wise. *Robust and Adaptive Control*. Springer, Springer Nature Switzerland, AG, 2013.

[8] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometery. In *IEEE International Conference onRobotics and Automation (ICRA)*, 2014.

[9] Georg Klein and David Muuray. Parallel tracking and mapping for small ar workspaces. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[10] Bruno Herisse, Tarel Hamel, Robert Mahony, and Francois-Xavier Russotto. Landin a vtol unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on Robotics*, 28:77–89, 2012.

[11] John Olaya, Nicolas Pintor, Dscar F. Aviles, and Juan Chaparro. Analysis of 3 rps robotic platform motion in simscape and matlab gui environment. *International Journal of Applied Engineering Research*, 12:1460–1468, 2017.

[12] Viliam Fedak, Frantisek Durovsky, and Robert Uveges. Analysis of robotic system motion in simmechanics and matlab gui environment. *MATLAB Applications for the Practical Engineer*, 3, 2014.

APPENDIX A: DATA

The proof of Lyapunov statbility comes hereupon, we prove that with the chosen $P \cdot D\cdot$ candiadate function, its derivative will be negetive. The Lyaponov candidtae is similar to one used in [3].

$$\dot{V} = \dot{e}^T P e + e^T P \dot{e} + tr\left(\dot{\widetilde{\theta}}^T \Gamma^{-1} \widetilde{\theta} + \widetilde{\theta}^T \Gamma^{-1} \dot{\widetilde{\theta}}\right) \tag{15}$$

tr in equation (14) refers to the trace function, means the summation of main diagonal terms of the matrix.

$$\begin{aligned} &\rightarrow \dot{V} = (\dot{x} - \dot{x}_m)^T P e + \\ &e^T P (\dot{x} - \dot{x}_m) + \frac{\Gamma^{-1}\tilde{\theta}^T d\left(\tilde{\theta}+\tilde{\theta}\right)}{dt} \end{aligned} \tag{16}$$

$$\begin{aligned} &\rightarrow \dot{V} = \left(\overbrace{A_m x - A_m x_m}^{A_m e} + \overbrace{\dots}^{\alpha\tilde{\theta}}\right)^T P e + \\ &e^T P \left(A_m e + \alpha\tilde{\theta}\right) + 2\Gamma^{-1}\tilde{\theta}^T \frac{d\tilde{\theta}}{dt} \end{aligned} \tag{17}$$

$$\begin{aligned} &\rightarrow \dot{V} = e^T A_m^T P e + \widetilde{\theta}^T \alpha^T P e + \\ &e^T P A_m e + e^T P \alpha\tilde{\theta} + 2\Gamma^{-1}\tilde{\theta}^T \frac{d\tilde{\theta}}{dt} \end{aligned} \tag{18}$$

$$\begin{aligned} &\rightarrow \dot{V} = e^T \overbrace{\left(A_m^T P e + P A_m e\right)}^{-Q} + \widetilde{\theta}^T \alpha^T P e + \\ &\overbrace{\left(\widetilde{\theta}^T \alpha^T P e\right)^T}^{e^T P \alpha\tilde{\theta}} + 2\Gamma^{-1}\widetilde{\theta}^T \frac{d\tilde{\theta}}{dt} \end{aligned} \tag{19}$$

$$\rightarrow \dot{V} = \underbrace{-e^T Q}_{N\cdot D\cdot} + \underbrace{2\widetilde{\theta}^T \left(\Gamma \alpha^T P e + \frac{d\widetilde{\theta}}{dt}\right)}_{=0} \tag{20}$$

To prove strictly negetiveness of the equation (19), $2\widetilde{\theta}^T \left(\Gamma \alpha^T P e + \frac{d\widetilde{\theta}}{dt}\right)$ must equals zero. The first term is $N \cdot D\cdot$ clearly because we have supposed the negetiveness of the Q matrix, before. Hence, either $\widetilde{\theta}^T = 0$ or $\dot{\widetilde{\theta}} = -\Gamma \alpha^T P e$ is true. The former phrase cannot be true so the latter is correct.

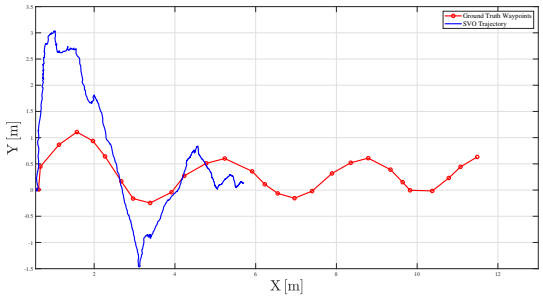

Figure 3: The trajectory of camera vs. the ground thruth path

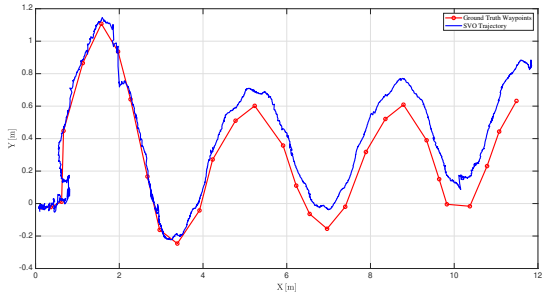

Figure 4: The trajectory of camera + IMU vs. the ground thruth path

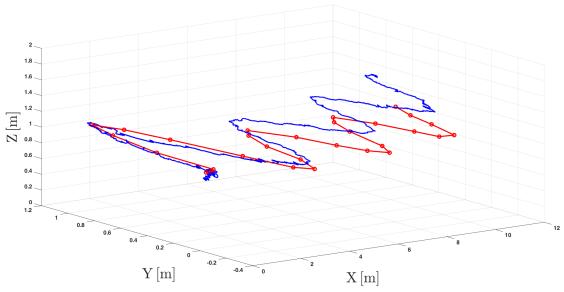

Figure 5: 3D trajectory of camera vs. the ground thruth path

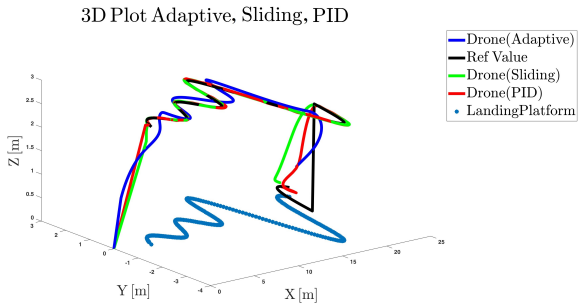

Figure 6: The 3D comparison of three controllers with $2.5m/s$ velocity in MATLAB Simscape

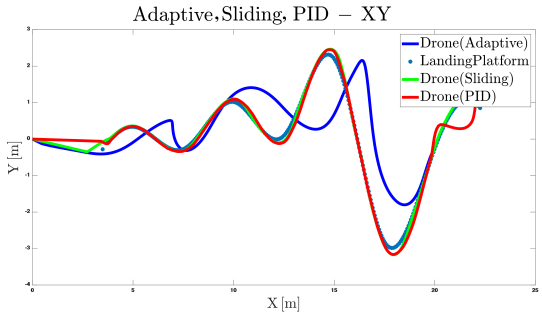

Figure 7: The XY comparison of three controllers to the landing platform with $2.5m/s$ velocity in MATLAB Simscape

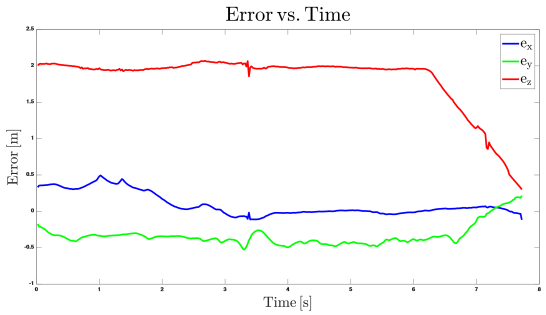

Figure 8: The XY errors of the quadrotor center with the moving platform center with $0.5m/s$ velocity with PID controller

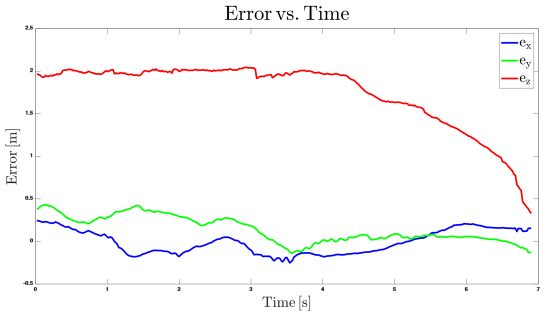

Figure 9: The XY errors of the quadrotor center with the moving platform center in with $0.5m/s$ velocity with sliding controller

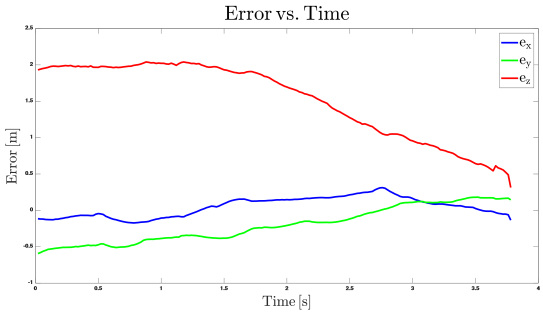

Figure 10: The XY errors of the quadrotor center with the moving platform center with $1.5m/s$ velocity with PID controller

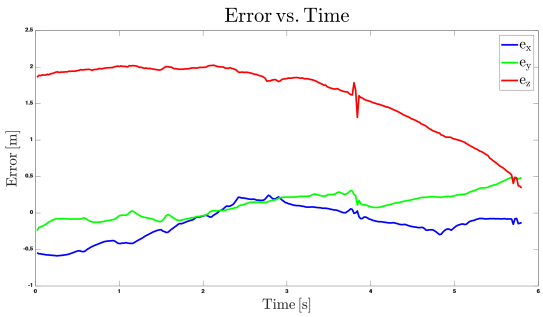

Figure 11: The XY errors of the quadrotor center with the moving platform center with $1.5m/s$ velocity with sliding controller