

Cooperative Motion Planning for Multiple UAVs via the Bézier Curve Guided Line of Sight Techniques

Warunyu Hematulin[†], Patcharin Kamsing^{†*}, Peerapong Torteeka[‡], Thanaporn Somjit[†],
Thaweerath Phisannupawong[†], Tanatthep Jarawan[†]

[†]*Air-Space Control, Optimization, and Management Laboratory, Department of Aeronautical Engineering,
International Academy of Aviation Industry, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand*

[‡]*National Astronomical Research Institute of Thailand, ChiangMai 50180, Thailand.*

*Email: h.warunyu@gmail.com, patcharin.ka@kmitl.ac.th, peerapong@narit.or.th, newpai1.12@gmail.com,
thaweerath2009@gmail.com, Tanatthep001@gmail.com*

Abstract—Multiple motion planning plays an essential role in several vehicle motions. This paper proposes a cooperative method between the Line of Sight techniques and the Bézier curve, applying this to motion planning for unmanned aerial vehicles. The experiment is implemented using the AirSim plugin on Unreal Engine 4. The results of the proposed method are compared with those for the conventional Line of Sight techniques to control multiple unmanned aerial vehicles. The results illustrate that the proposed method takes more time to process than the conventional one. However, the proposed method can reach a higher performance by addressing the target unmanned aerial vehicles and the pre-defining path more than the conventional method does, which is shown in all three simulation cases.

Index Terms—Multiple UAVs, motion planning, Guidance Law, LOS(Line-of-Sight), Bézier curve, AirSim

I. INTRODUCTION

Recently, unmanned aerial vehicles (UAVs) have proved to be useful in fields like remote sensing (e.g., mining), real-time monitoring, disaster management, border and crowd surveillance, military applications, the delivery of goods, precision agriculture, infrastructure inspection, and the media and entertainment [1], [2]. Some companies try to implement UAVs to automatically control their inventory by deploying indoor UAVs to carry their product from one point to another. This can decrease the human error effect on stock control. Motion planning for UAVs should ensure that their mission is performed perfectly through an automatic process. Motion planning such as this also has an impact on space missions, for instance, when a spacecraft is docking [3], [4]. Furthermore, motion planning can improve aircraft trajectory and enable an airport reach its highest capacity [5].

AirSim is a plugin that can be used on Unreal Engine 4 to simulate the realistic movements of UAVs. AirSim has a library for the control of UAVs, which this experiment applies using a quadrotor. Guidance systems are important when considering the performance of aerial, surface, and underwater vehicles, regardless of the motion control scenario involved. A guidance system is used in many processes, such as target tracking, path following, path tracking, and path maneuvering. Many guidance systems have been introduced by different

researchers, such as Line of Sight (LOS) guidance, Pure Pursuit (PP) guidance, and Constant Bearing (CB) guidance. Regardless of their size and mission, all UAVs share the need for a Guidance, Navigation, and Control (GNC) system. LOS is a guidance law that extends the pursuit guidance law, in which several improvements are presented to allow for operation in the real world. The LOS guidance method is used in many fields of work, especially to guide underactuated marine vessels and vehicles, operating on or below the surface of the sea, in path-following tasks [9]. The LOS guidance method can steer a simplified vessel to and along a desired parameterized curve path [10]. Due to the embedded integral action, the guidance law makes the vessels follow straight lines by compensating for the drift effect of environmental disturbances, such as currents, wind, and waves [11].

Motion planning is a major area in robotics as it gives way to autonomous vehicles. In that sense, the motion planning of UAVs is of importance. As UAVs come to be applied in a growing number of different environments, motion planning plays one of the most important roles in providing UAVs with truly autonomous capabilities. The parametric curve used in the computer graphic is referred to as the Bézier curve. In vector graphics, the Bézier curve is used to model smooth curves that can be scaled indefinitely. Nowadays, the Bézier curve has been applied in much research. [6] used the Bézier curve in a motion planning algorithm for autonomous vehicles with waypoints and corridor constraints. [7] proposed Bézier curve-based smooth motion planning algorithms for a mobile robot. [8] presented a new motion planning algorithm for a mobile robot that combined the Genetic Algorithm with the Bézier curve, thereby demonstrating that it is appropriate to combine the Bézier curve with another algorithm.

In this paper, the Bézier curve will be implemented in a motion planning algorithm using a set of points that target UAV moved pass along the waypoint; these have been generated to act as a set of control points. Then, this will be combined with a guidance system, namely, LOS. The experiment is conducted on Unreal Engine 4 and the AirSim plugin is employed to make the simulation realistic. This paper is organized as follows: the next section is related to the field of work, introducing the Unreal Engine, AirSim, LOS, and Bézier

This work was supported by King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand.

Curve. Section III describes the proposed method. Section IV presents the results of the three test cases and a conclusion is drawn in section V.

II. RELATED WORK

A. Unreal Engine

Since photo-realistic rendering is an essential requirement for aerial informatics and robotics platforms, the popular, recently open-sourced, advance real-time 3D creation platform for real photo visuals and immersive experiences, Unreal Engine (UE4) [12], is used to run the simulation platform, AirSim. Because Unreal Engine 4 is open-source software and is available on many operation systems, such as Linux, Windows, and MacOS, it is considered to be the right choice for this work. Moreover, the Unreal Engine platform offers an elaborate marketplace with hundreds of pre-made detailed environments, many created using photogrammetry techniques to generate a reasonably faithful reconstruction of real-world scenes. These can be used perfectly with AirSim; the AirSim code base is implemented as a plugin for the Unreal Engine. It can be dropped into any Unreal project, as shown in Fig. 1.



Fig. 1. Image from an AirSim simulation including a UAV model, front-facing of the UAV and sensors.

B. AirSim

Today, simulation modeling is utilized to solve real-world problems safely and efficiently. There are many platforms used to solve these kinds of problems, such as: Gazebo [13], Hector [14], RotorS [15], and jMavSim [16]. However, AirSim is favored for this work. It is a new choice for making the guidance algorithms in realistic UAVs dynamic. Furthermore, in artificial intelligence (AI) research, AirSim is developed as a platform for experimentation with deep learning computer vision and reinforcement learning algorithms for autonomous vehicles, which is interesting and may be of use for our work in the future.

AirSim supports various flight controllers (FCs), which can be either hardware-in-the-loop or completely software-simulated. In this paper, we use the default software-simulated flight controller provided by AirSim. However, AirSim also supports other FCs, such as the Pixhawk [17], ROSFlight [18], Hackflight [19], and PX4 [20] software stack. AirSim supports

any FC that can communicate using MAVLINK, a widely-used micro aerial vehicle message marshaling library.

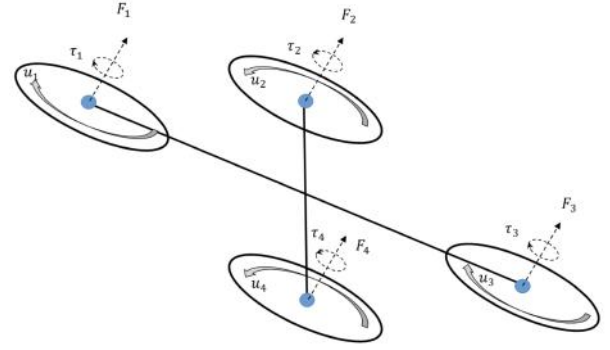


Fig. 2. Vehicle model for the quadrotor. The four blue vertices experience the controls $u_1, \dots, u_4$, which, in turn, result in the forces $F_1, \dots, F_4$ and the torques $\tau_1, \dots, \tau_4$, as shown in [22].

AirSim provides an interface for defining a vehicle as a rigid body that may have an arbitrary number of actuators generating forces and torques. The vehicle model includes parameters such as mass, inertia, coefficients for linear and angular drag, coefficients of friction, and restitution, which are utilized by the physics engine to compute rigid body dynamics. Formally, a vehicle is defined as a collection of K vertices placed at positions $r_1, \dots, r_k$ and normals $n_1, \dots, n_k$, each of which experience a unitless vehicle-specific scaler control input $u_1, \dots, u_k$. The forces and torques from these vertices are assumed to be generated in the direction of their normals. However, note that the positions, as well as the normals, are allowed to change during the simulation. Fig. 2 shows how a quadrotor can be depicted as a collection of four vertices. The control input u_i drives the rotational speed of the propellers located at the four vertices. The forces and torques can be computed by propellers as

$$F_i = C_T \rho \omega_{max}^2 D^4 u_i, \quad (1)$$

and

$$\tau = \frac{1}{2\pi} C_{pow} \rho \omega_{max}^2 D^5 u_i. \quad (2)$$

Here, C_T and C_{pow} are the thrust and the power coefficients, respectively. These are based on the physical characteristics of the propeller. ρ is the air density and D is the propeller's diameter. ω_{max} is the maximum angular velocity in revolutions per minute. This vehicle model provides a way to specify the cross-sectional area in the body frame that, in turn, can be used by the physics engine to compute the linear and angular drag on the body.

The vehicle is exposed to various physical phenomena including gravity, air density, air pressure, and magnetic field. The model of AirSim is accurate enough to allow a real-time operation with hardware-in-the-loop. For the gravity, AirSim approximates a gravitational acceleration g at height h by

applying the binomial theorem on Newton's law of gravity and neglecting the higher powers, as

$$g = g_0 \frac{R_e^2}{(R_e + h)^2} \approx g_0 \left(1 - 2\frac{h}{R_e}\right). \quad (3)$$

In which R_e is the Earth's radius and g_0 is the gravitational constant measured at the surface. The relationship between the altitude and the pressure of the Earth's atmosphere is complicated due to the presence of many distinct layers, each with its own individual properties. Standard Temperature T and Standard Pressure P , are defined using the 1976 U.S. Standard Atmosphere model for altitude below 51 kilometers, switching to the model in [21] beyond that and up to 86 kilometers. Then, the air density is defined by

$$\rho = \frac{P}{RT}. \quad (4)$$

where R is the specific gas constant.

In AirSim, the kinematic state of the body is expressed using six quantities: position, orientation, linear velocity, linear acceleration, angular velocity, and angular acceleration. Since the vehicle moves in the presence of air, linear and angular drag have a significant effect on the dynamics of the body. The simulator computes the magnitude $|F_d|$ of the linear drag force on the body following the drag equation

$$|F_d| = \frac{1}{2} \rho |v|^2 C_{lin} A, \quad (5)$$

where C_{lin} is the linear air drag coefficient, A is the vehicle cross-section, and ρ is the air density. This drag force acts in the direction opposite to the velocity vector v . Calculating the angular drag for an arbitrary shape remains a complex and computationally-intensive task. Therefore, AirSim used simple but improved approximations to model the angular drag. Consider an infinitesimal surface area ds in the extremity of the body experiencing the angular velocity ω . As the linear velocity dv experienced by ds is given by $r_{ds} \times \omega$, we can now use the following linear drag equation for ds :

$$|dF| = \frac{1}{2} \rho |r_{ds} \times \omega|^2 C_{lin} ds. \quad (6)$$

where the direction of dF is $-r_{ds} \times \omega$. In addition to the drag forces and torques, AirSim computes the net forces and torques following the equations

$$F_{net} = \sum_i F_i + F_d, \quad (7)$$

and

$$F_{net} = \sum_i [\tau_i + r_i \times F_i] + \tau_d. \quad (8)$$

AirSim obtains the linear acceleration by applying Newton's second law and then adding a gravity vector to compute the net acceleration, $a = F_{net}/m + g$. The angular acceleration in the body frame is given by Euler's rotation equation

$$\alpha = I^{-1}(\tau_{net} - (\omega \times (I\omega))). \quad (9)$$

where I is the inertia tensor and ω is angular velocity, both in body frame.

In addition, AirSim offers sensor models for an accelerometer, gyroscope, barometer, magnetometer, and global positioning system (GPS), which are implemented as a C++ header-only library and can be independently used outside of AirSim. The sensor models can replace others or add new sensors.

C. Line of Sight Guidance Law

This paper describes the guidance law to generate a lateral acceleration command, to track a moving UAV target in the first algorithms, and to track the path generated by the Bézier curve of a moving UAV target in the second algorithm. Conceptually, this guidance law steers the ground velocity vector towards an aim point, which is the moving UAV target in this work, chosen to be a specified look-ahead distance along the desired path in front of the quadrotor.

Based on the geometry shown in Fig. 3, the Line of Sight guidance law is designed from the azimuth angle of a quadrotor velocity (Ψ_q), which is defined as the angle between the north and collector planes, to converge to the azimuth angle of the LOS (Ψ_{LOS}), connecting the quadrotor location (P_q) and the final point (P_f). Thus, the azimuth angle of a LOS (Ψ_{LOS}) is used to command the heading angle of the quadrotor, as shown in (11).

$$\Psi_{LOS} = \tan^{-1}\left(\frac{x_f - x_q}{y_f - y_q}\right), \quad (10)$$

$$\Psi_{cmd} = \Psi_{LOS}. \quad (11)$$

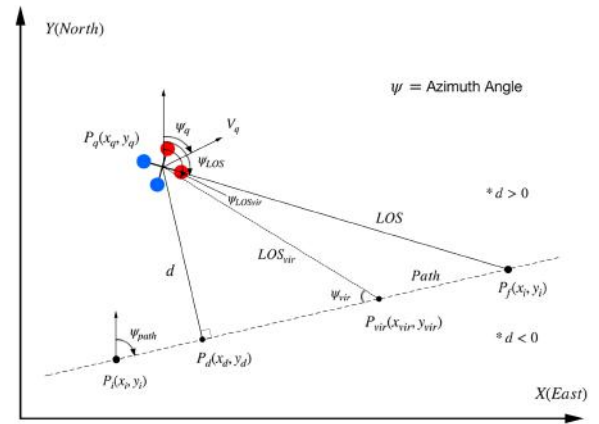


Fig. 3. The Structure of the LOS Guidance Laws.

We can calculate the virtual point (P_{vir}) on the straight path to the final point from the initial point. So, the command heading angle of the aircraft (Ψ_{cmd}) is equal to the azimuth angle of the virtual LOS ($\Psi_{LOS_{vir}}$), connecting the aircraft location (P_q) and the virtual point (P_{vir}), as given by the equations

$$\Psi_{LOS_{vir}} = \tan^{-1}\left(\frac{x_q - x_{vir}}{y_q - y_{vir}}\right), \quad (12)$$

$$\Psi_{cmd} = \Psi_{LOS_{vir}}. \quad (13)$$

The distance between the quadrotor location and the position of the perpendicular-projection, which is the path from the

quadrotor position, is as shown in Fig. 3. If the quadrotor is located on the left side of the flight path, d is positive ($d > 0$). If the aircraft is on the right side, d is negative ($d < 0$). The angle of the virtual (Ψ_{vir}) between the quadrotor position (P_q) and the virtual position (P_{vir}) can be obtained by applying a PID control to the cross-track-error, as shown in (12). To sum up, the azimuth angle of the quadrotor velocity and the command heading angle of the quadrotor can be written as (11) and (13).

D. Bézier Curve

A Bézier curve is a famous parametric curve used in computer graphics and related fields. It is materialized as a function in many famous computer programs, such as Adobe Photoshop. A Bézier curve was proposed by a French engineer named Pierre Bézier in 1962. He applied this curve to design an automobile. Since then, the Bézier curve has been applied in many fields, especially in computer graphic design and missile movement trajectory planning.

A Bézier curve is defined by a set of control points. The control points include a start point, terminal point, and shape-defined point. The first and last control points are always the end points of the curve. The order of a Bézier curve depends on the number of control points. If the number of control points is n , then the order will be $n-1$. The curve moves through the start and terminal points but does not necessarily move through the shape-defined points, which means the shape-defined points can be moved to reshape and modify the curve without changing the initial and terminal boundary conditions. Each shape-defined point can bend the curve once. More shape-defined points can define a complex curve. A Bézier curve $P(s)$ of degree n , obtained from $n+1$ control points ($P_0, P_1, \dots, P_n$), is written as

$$P(s) = \sum_{i=0}^n B_i^n(s) P_i \quad s \in [0, 1], \quad (14)$$

where $B_i^n(s)$ is the i^{th} Bernstein polynomial of degree n , which can be defined by

$$B_i^n(s) = \binom{n}{i} s^i (1-s)^{n-i}, \quad (15)$$

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \quad i \in 0, 1, \dots, n, \quad (16)$$

where the binomial theorem to expand $(1-t)^{n-i}$ is as seen in Fig. 4. For the sake of simplicity, it can be rewritten in the monomial form as follows:

$$P(s) = [P_0 \quad P_1 \quad \dots \quad P_n] \begin{bmatrix} m_{0,0} & \dots & m_{0,n} \\ m_{1,0} & \dots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{n,0} & \dots & m_{n,n} \end{bmatrix} \begin{bmatrix} t^0 \\ t^1 \\ \vdots \\ t^n \end{bmatrix}, \quad (17)$$

where

$$m_{i,j} = (-1)^{j-i} \binom{n}{j} \binom{j}{i}. \quad (18)$$

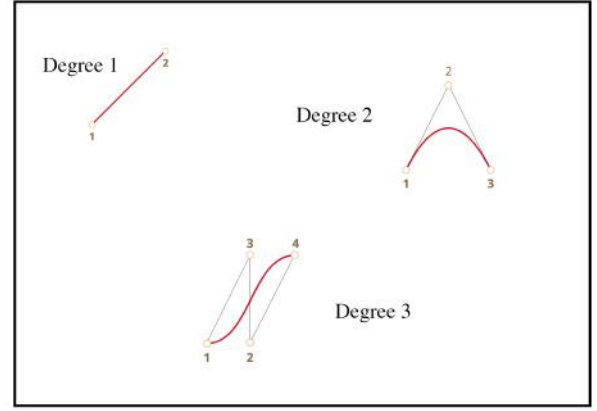


Fig. 4. Differential degree of a Bézier curve.

III. METHOD

The AirSim flight controller API is used for the simulation in this work with both the proposed method and conventional method. The concept for the proposed method is to combine the LOS guidance law and a motion planning algorithm, namely, the Bézier curve, to command the quadrotor to pursue the moving target UAVs. At the same time, the conventional method applies only the LOS guidance law to pursue the moving target UAVs.

A. LOS technique

The first basic algorithm uses the LOS guidance law, which is discussed in section II. The following pseudocode (Algorithm 1:) is the sequence used for the following quadrotor guidance to pursue the moving target quadrotor using LOS guidance law. The concept sequence begins by defining the positions for the following and target quadrotors using the AirSim flight controller API, then the heading angle for the LOS is calculated.

Algorithm 1: LOS moving target pursuit algorithm

Data: Position of quadrotor follower, Position of target quadrotor;

Result: Velocity command for following quadrotor;

while target is moving **do**

 Get the position of the target quadrotor;

 Get the position of the following quadrotor;

 Calculate the LOS heading angle;

 Command the following quadrotor to follow the target;

end

B. LOS-based Bézier

The proposed method collects the positions of moving target quadrotors and sets them as the control points. The Bézier curve will generate a path based on those control points and the LOS will use that to drive the following quadrotor along the generated path. For example, in the implementation, if we set the number of control points (n) to five, that means

the proposed method will collect five positions for the target quadrotors every 0.5 seconds and use these as the set of control points to generate a path based on the Bézier curve, to dictate the movement of the following quadrotor.

Algorithm 2: LOS-based Bézier moving target pursuit algorithm

Data: Position of follower quadrotor, Position of target quadrotor;

Result: Velocity command for following quadrotor;
Collect the target position n times every 0.5 seconds to be a set of control points;

while *target is moving* **do**

 Generate a path from the set of control points based on Bézier curve theory;

 Use the LOS algorithm to command the following quadrotor and collect the new set of control points at the same time;

end

IV. SIMULATION RESULTS AND DISCUSSION

The experimental results when using the proposed method (combining the processing of the LOS algorithm with motion planning using the Bézier curve) have been compared with those obtained when using the conventional LOS guidance law algorithm. Both have been applied in the same scenario, beginning with defining the waypoints that the target quadrotor moves along. Then, the following quadrotor uses the LOS or the proposed method to move and follow the target quadrotor. To see the differences between results in different environments, the experiment has been conducted with three different simulation cases for the flight path, namely: 1) a 90-degree turning flight path, 2) a straight flight path with wind, and 3) a random flight path.

A. 90-degree turning flight path

In the first simulation, the target quadrotor created a flight path from the waypoint. In this experiment, there were a total of three waypoints, starting from WP0 (0,0) m, moving to WP1 (0, 15) m, and then to WP2 (15, 30)m, as shown in Fig. 5. The target quadrotor was set to follow the waypoint using the flight controller API of AirSim, with a velocity of 2 m/s. The following quadrotor started at (-3, 0) behind the target. The proposed method has been set with eight controls points, which is the best value for this kind of flight path. Fig. 5 shows the comparable results of the two methods.

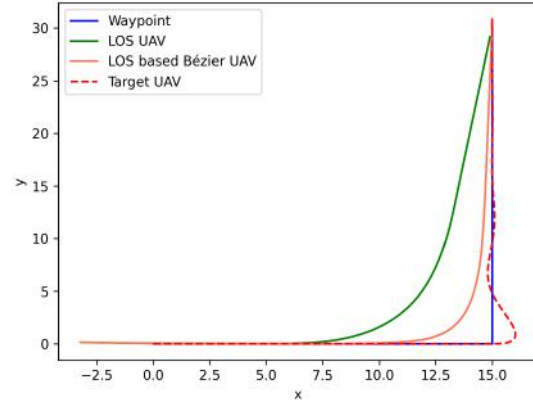


Fig. 5. The actual paths of the following quadrotor and target quadrotor: A comparison of the two algorithms for the 90-degree turning flight path.

at a speed of 6 m/s. Fig. 6 shows how the target quadrotor moved from the (0, 0)m point to the (30, 0)m point with 2 m/s of velocity, using the flight controller API of AirSim. The following quadrotor started at (-3, 0) behind the target. For the proposed method, the number of controls points was set to seven, which is the best value for this kind of flight path. Fig. 6 show the comparable results of the two methods.

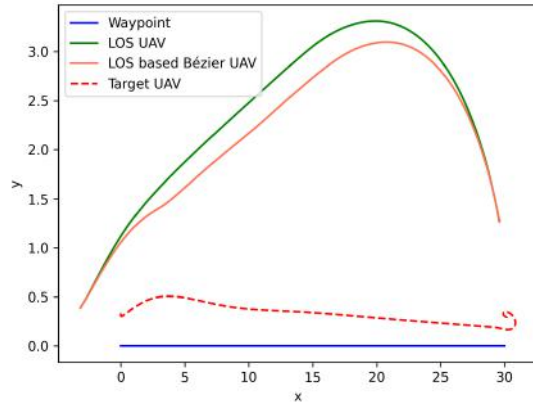


Fig. 6. The actual paths of the following quadrotor and target quadrotor: A comparison of the two algorithms for the added wind straight flight path.

TABLE I
TIMES TAKEN IN THE 90-DEGREE TURNING FLIGHT PATH EXPERIMENT.

	Time (mm:ss)
LOS algorithm	00:47.30
LOS-based Bézier algorithm	00:49.01

B. Straight flight path with wind

In the second experiment, the flight path was a straight line with added the wind to the north, or along the y-axis,

TABLE II
TIMES TAKEN IN THE ADDED WIND STRAIGHT FLIGHT PATH EXPERIMENT.

	Time (mm:ss)
LOS algorithm	00:43.93
LOS-based Bézier algorithm	00:48.44

C. Random Flight Path

The random flight path had a defined total of four waypoints: WP0 (0, 0)m, WP1 (10, 2)m, WP2 (13, 15)m, and WP3 (-6, 15)m. The target quadrotor was set to move along

the waypoints using the flight controller API of AirSim with a velocity of 2 m/s. The following quadrotor started at (-3, 0) behind the target. For the proposed method, the number of controls points was set to seven, which is the best value for this kind of flight path. Fig. 7 shows the comparable results of the two methods.

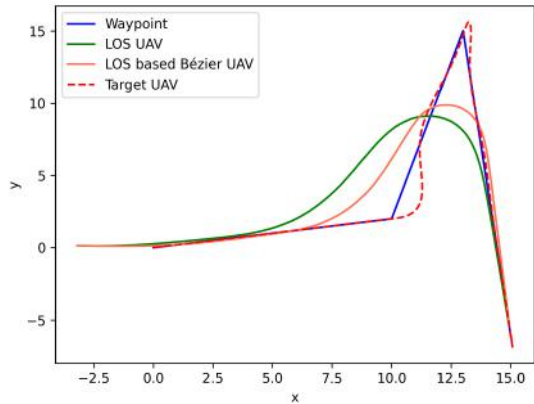


Fig. 7. The actual paths of the following quadrotor and target quadrotor: A comparison of the two algorithms for the random flight path.

TABLE III
TIMES TAKEN IN THE RANDOM FLIGHT PATH EXPERIMENT.

	Time (mm:ss)
LOS algorithm	00:54.35
LOS-based Bézier algorithm	00:59.09

V. CONCLUSION

Cooperative motion planning is increasingly essential nowadays as part of the growth of many industries. In this paper, UAVs were selected as the objects of study for cooperative motion planning. The experiment employed the Unreal Engine 4 software and the AirSim plugin. The experiment compared the results between the conventional LOS method for controlling UAVs and the proposed method, which contributes a novel process for calculating a path before performing the LOS. Although the proposed method took more time to process, the results show that the proposed method can address pre-defined waypoints and target UAVs more so than the LOS method is solely applied. This paper clearly demonstrates the results using three different test cases. Future research should further considered implementing the proposed method as part of an automatic system.

REFERENCES

- [1] Shakhathreh, H.; Sawalmeh, A.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges. arXiv 2018, arXiv:1805.00881.
- [2] Hassanalian, M.; Abdelkefi, A. Classifications, applications, and design challenges of drones: A review. Prog. Aerosp. Sci. 2017, 91, 99–131.
- [3] T. Phisannupawong, P. Kamsing, P. Tortceka and S. Yooyen, "Vision-based attitude estimation for spacecraft docking operation through deep learning algorithm," 2020 22nd International Conference on Advanced Communication Technology (ICACT), Phoenix Park, PyeongChang., Korea (South), 2020, pp. 280-284, doi: 10.23919/ICACT48636.2020.9061445.
- [4] Phisannupawong, T.; Kamsing, P.; Tortceka, P.; Channumsin, S.; Sawangwit, U.; Hematulin, W.; Jarawan, T.; Somjit, T.; Yooyen, S.; Delahaye, D.; Boonsrimuang, P. Vision-Based Spacecraft Pose Estimation via a Deep Convolutional Neural Network for Noncooperative Docking Operations. Aerospace 2020, 7, 126.
- [5] P. Kamsing et al., "Aircraft trajectory recognition via statistical analysis clustering for Suvarnabhumi International Airport," 2020 22nd International Conference on Advanced Communication Technology (ICACT), Phoenix Park, PyeongChang., Korea (South), 2020, pp. 290-297, doi: 10.23919/ICACT48636.2020.9061368.
- [6] J. Choi, R. Curry and G. Elkaim, "Path Planning Based on Bézier Curve for Autonomous Ground Vehicles," Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, San Francisco, CA, 2008, pp. 158-166, doi: 10.1109/WCECS.2008.27.
- [7] F. Zhou, B. Song, G. Tian, "Bézier Curve Based Smooth Path Planning for Mobile Robot", Journal of Information and Computational Science, vol. 8, no. 12, pp. 2441-2450, 2011.
- [8] Yang Linqun, Luo Zhongwen, Tang Zhonghua and Lv Weixian, "Path planning algorithm for mobile robot obstacle avoidance adopting Bézier curve based on Genetic Algorithm," 2008 Chinese Control and Decision Conference, Yantai, Shandong, 2008, pp. 3286-3289, doi: 10.1109/CCDC.2008.4597937.
- [9] W. Caharija et al., "Integral Line-of-Sight Guidance and Control of Underactuated Marine Vehicles: Theory, Simulations, and Experiments," in IEEE Transactions on Control Systems Technology, vol. 24, no. 5, pp. 1623-1642, Sept. 2016, doi: 10.1109/TCST.2015.2504838.
- [10] R. Skjetne, U. Jørgensen and A. R. Teel, "Line-of-sight path-following along regularly parametrized curves solved as a generic maneuvering problem," 2011 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, 2011, pp. 2467-2474, doi: 10.1109/CDC.2011.6161364.
- [11] W. Caharija, K. Y. Pettersen, M. Bibuli et al., "Integral line-of-sight guidance and control of underactuated marine vehicles: theory, simulations, and experiments," IEEE Transactions on Control Systems Technology, vol. 24, no. 5, pp. 1623-1642, 2016.
- [12] Karis, B., Games, E.: Real shading in unreal engine 4. In: Proc. Physically Based Shading Theory Practice (2013)
- [13] Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IROS (2004)
- [14] Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., Von Stryk, O.: Comprehensive simulation of quadrotor uavs using ros and gazebo. In: SIMPAR, pp. 400-411. Springer (2012)
- [15] Furrer, F., Burri, M., Achtelik, M., Siegwart, R.: Rotors modular gazebo mav simulator framework. In: Robot Operating System (ROS), pp. 595-625. Springer (2016)
- [16] Babushkin, A.: Jmavsim. <https://pixhawk.org/dev/hil/jmavsim>
- [17] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In Robotics and automation (ICRA), 2011 IEEE international conference on, pages 2992-2997. IEEE, 2011.
- [18] Jackson, J., Ellingson, G., McLain, T.: Rosflight: A lightweight, inexpensive mav research and development tool. In: ICUAS, pp. 758-762 (2016). DOI 10.1109/ICUAS.2016.7502584
- [19] Levy, S.: Hackflight Simple quadcopter flight control firmware and simulator for c++ hackers. <https://github.com/simondlevy/hackflight>
- [20] Meier, L., Tanskanen, P., Fraundorfer, F., Pollefeys, M.: Pixhawk: A system for autonomous flight using onboard computer vision. In: ICRA, pp. 2992-2997. IEEE (2011)
- [21] Braeunig, R.: Atmospheric models. <http://www.braeunig.us/space/atmmode1.html> (2014)
- [22] Shital Shah, Debadepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In Marco Hutter and Roland Siegwart, editors, Field and Service Robotics, pages 621-635, Cham, 2018. Springer International Publishing. ISBN 978-3-319-67361-5. doi: 10.1007/978-3-319-67361-5_40.



Warunyu Hematulin is currently a bachelor degree student, major in Aeronautical Engineering and Commercial Pilot (International Program), from the Department of Aeronautical Engineering, International Academy of Aviation Industry. His research interests are included the simulation, guidance and control system on aeronautical and aerospace engineering.



Patcharin Kamsing received doctoral degree from the University of Chinese Academy of Sciences, Beijing, China sponsor by CAS-TWAS president fellowship. She currently is lecturer and assistant dean of International Academy of Aviation Industry, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. Her research is image processing and remote sensing applications.



Peerapong Torteeka received doctoral degree in Celestial Mechanics and Applied Astrometry Engineering at National Astronomical Observatories of Chinese Academy of Sciences (NAOC), University of Chinese Academy of Sciences, Beijing, China. He currently works at National Astronomical Research Institute of Thailand, research interests include passive optical-based small and dim space debris recognition and tracking system, moving object extraction, autonomous detection, robotics telescope for the space observation.



Thanaporn Somjit is currently a bachelor degree student, major in Aeronautical Engineering and Commercial Pilot (International Program), from the Department of Aeronautical Engineering, International Academy of Aviation Industry.



Thaweerath Phisannupawong is currently a bachelor degree student, major in Aeronautical Engineering and Commercial Pilot (International Program), from the Department of Aeronautical Engineering, International Academy of Aviation Industry. His research interests are included the image processing and application of deep learning on aeronautical engineering.



Tanattthep Jarawan is currently a bachelor degree student, major in Aeronautical Engineering and Commercial Pilot (International Program), from the Department of Aeronautical Engineering, International Academy of Aviation Industry. His research interests are included Indoor Localization Method via WiFi Receive Signal Strength Indicator.