Vision-Based Navigation for Control of Micro Aerial Vehicles

Xavier WeiJie Leong and Henrik Hesse
Aerospace Sciences Division
University of Glasgow
Singapore
Xavierleong94@gmail.com, henrik.hesse@glasgow.ac.uk

***Abstract*—This paper presents the use of an external vison-based positioning system for navigation and flight control of micro aerial vehicles. The motion capture system Optitrack is used to localize the vehicle which is a nano-quadcopter Crazyflie 2.0. An interface was created to pass positioning data to the Robot Operating Software (ROS). ROS acts as communication layer to bridge between Optitrack and available control nodes for Crazyflie 2.0 platforms.**

Keywords-Vision-Based Navigation; Robot Operating System (ROS); Aerial Robotics, Micro Aerial Vehicles (MAVs)

I. INTRODUCTION

In the last decade, Micro Aerial Vehicles (MAVs) have been used extensively for research and, recently, also find applications in areas such as inspection surveillance [8,9,13]. To achieve autonomous operation for such applications, we require robust positioning and control performance of MAVs. To localize the MAV and estimate its state, common sensor fusion approaches typically rely on inertial sensors and GPS to obtain a position measurement. However, in GPS-denied environments the vehicle has no sense of its position [18,19]. This lack of position information naturally hinders the control performance and is especially relevant in recent applications of MAVs for inspection of enclosed areas. For such industry applications, we typically require a reliable, inexpensive positioning system [9].

The choice of the MAV in this work is the open-sourced, open-hardware nano-quadcopter Crazyflie 2.0 as shown in Figure 1. The Robot Operating Software (ROS) is used as the communication layer to demonstrate the flexible and modular code design for MAVs using ROS. To obtain external position measurements we use the motion capture system Optitrack.

The main objective of this work is to implement a ROS framework for the inexpensive localization and control of MAVs using vision-based position data from Optitrack.

A. MAV Platform: Crazyflie 2.0

The research platform Crazyflie 2.0 is a nano-quadcopter which was developed by a team of software engineers at Bitcraze. It weighs 27g and can be easily flown indoors with the dimensions of 92mm x 92mm x 29mm.

The MAV comes equipped with a micro-processor which reads measurements from a 10-degree-of-freedom (10-DOF) inertial measurement unit (IMU). The IMU consists of a 9-DOF instrument with gyroscope, accelerometer and magnetometer and 1-DOF unit with a pressure sensor. The IMU therefore provides measurements of angular velocity, linear acceleration, orientation and lastly altitude. However, the Crazyflie 2.0 has no 3D position measurements and the pressure sensor tends to drift providing inaccurate altitude measurements due to external disturbances [1,6].

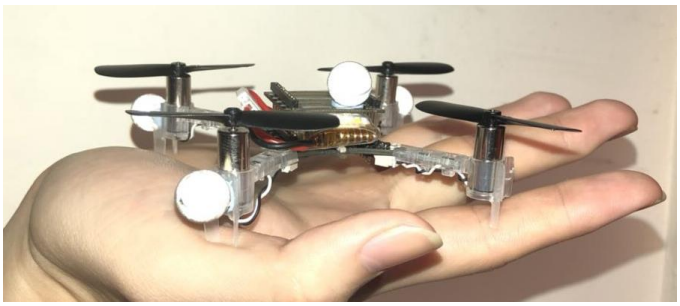

Figure 1. The palm-sized nanoquadcopter, Crazyflie 2.0, attached with reflective markers

B. External Positioning System: Optitrack

In the aerial robotics research domain, quadcopters are typically localized using the external positioning system VICON system [14,18,21]. Recent work [5], however, has demonstrated the use of the motion capturing system Optitrack as an inexpensive solution to provide 6 DOF accuracy. Optitrack is used to detect infrared light reflected from markers on the MAV. After calibration and so-called wanding [7], the system is able to stream data in real time to a 3D virtual reality graph.

C. Robot Operating System: ROS

ROS is a communication interface extensively used in the robotics community to program robots. Its strength is the modular nature which allows code reuse and compatibility. It is operated on an Ubuntu operating platform and features open source libraries and tools for robotic applications. ROS nodes are bits of codes C++ or Python to perform specific tasks. In Figure 2, ROS is illustrated as a communication layer between nodes, where nodes can publish and subscribe to topics. These topics are messages sent to ROS and usually that subscribe to these published topics can make use of this data [7,20].

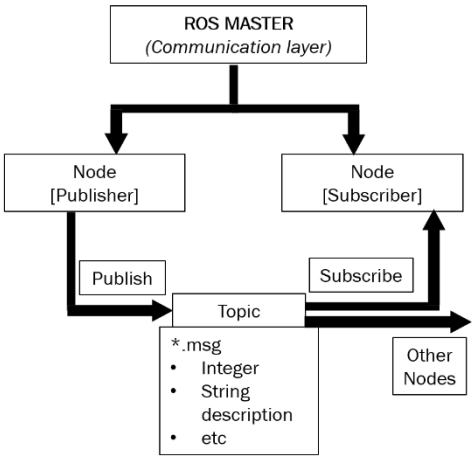

Figure 2. ROS flowchart of nodes in publishing and subscribing

II. MATHEMATICAL MODELLING

A. *Crazyflie Dynamics*

The dynamics of the Crazyflie 2.0 quadcopter is modelled with respect to an inertial Earth frame as defined in Figure 3. The body reference frame is also presented with the X, Y and Z axes and the Euler angles for roll, pitch and yaw denoted as φ, θ and ψ respectively [17]. Using these Euler angles, we can derive a transformation between the quadcopter body-attached frame B and the inertial frame E as,

$$H_B^E = \begin{bmatrix} C\theta\, C\psi & -C\theta\, S\psi & S\theta \\ S\varphi\, S\vartheta\, C\psi + C\varphi\, S\psi & C\varphi\, C\psi + S\varphi\ S\theta\ S\psi & -S\varphi\, C\theta \\ -\,C\varphi\, S\vartheta\, C\psi + S\varphi\, S\psi & S\varphi\, C\psi + C\varphi\, S\theta\, S\psi & C\varphi\, C\theta \end{bmatrix} \quad (1)$$

where C is the cosine and S the sine operator.

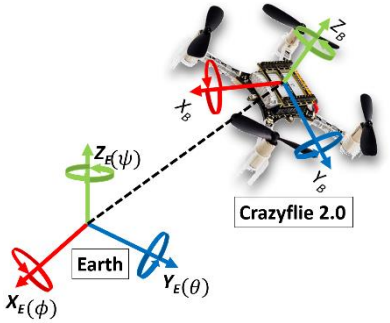

Figure 3. Coordinate system for Crazyflie 2.0

The dynamic equations are derived next under the following assumptions:

1. The quadcopter is a rigid body that cannot deform.
2. The quadcopter is symmetrical in terms of mass distribution and geometry.
3. The mass of the quadcopter is constant.

Based on Newton's 2nd Law we can find the translational equations of motion (EoM) in the body frame B as [22],

$$\sum \vec{F}_B = m\vec{a} = \left(\dot{\vec{v}}_B + \vec{\omega}_B \times \vec{v}_B\right) = H_E^B \vec{F}_g^E + \vec{F}_a^B$$

where m is mass, $\vec{a}$ is total acceleration vector, $\dot{\vec{v}}_B$ is linear acceleration vector, $\vec{\omega}_B = [p, q, r]$ is the angular velocity vector, $\vec{v}_B = [u, v, w]$ is linear velocity vector, $\vec{F}_g^E$ is the gravity vector of the Crazyflie 2.0 in the Earth from E and $\vec{F}_a^B = \left[F_x, F_y, F_z\right]$ is the vector of total motor forces in the body frame B resulting from all four propellers. The translational EoM can then be expressed in vector form as,

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_z/m \end{bmatrix} - H_E^B \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Next, the rotational dynamics are derived using the theorem of angular momentum,

$$\sum \vec{M}_B = \dot{\vec{h}} = \dot{\vec{h}}_B + \vec{\omega}_B \times \vec{h}_B$$

where $\vec{h}$ is the total angular momentum around the center of gravity, $\vec{h}_B$ the angular momentum with respect to the rotating body frame B and $\vec{M}_B$ the vector of resultant torques from the motors. With $\vec{h}_B = \mathcal{J}\vec{\omega}_B$, we can find the angular EoM as,

$$\sum \vec{M}_B = \mathcal{J}\dot{\vec{\omega}}_B + \vec{\omega}_B \times \mathcal{J}\vec{\omega}_B$$

with the inertia matrix $\mathcal{J}$ of the Crazyflie 2.0 assumed to be diagonal due to symmetry, such that

$$\mathcal{J} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

where $I_{xx} = 1.395 \times 10^{-5} kgm^2$, $I_{yy} = 1.436 \times 10^{-5} kgm^2$ and $I_{zz} = 2.173 \times 10^{-5} kgm^2$ [14].

The relationship between the translational and angular EoM is given by the transformation H_E^B defined in (1). The Euler angles are propagated through the relationship between $\vec{\omega}_B$ and Euler rates,

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sin\theta \\ 0 & cos\theta & sin\phi cos\theta \\ 0 & -sin\phi & cos\phi cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

which can be re-arranged to compute the Euler rates,

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi \tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi/cos\theta & cos\phi/cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \text{ for } \theta \neq \frac{\pi}{2}$$

where the angular velocities $\vec{\omega}_B = [p, q, r]$ are now the inputs and can come from IMU measurements of the Crazyflie 2.0. To avoid gimbal lock, pitching cannot reach 90 degrees ($\theta \neq \pi/2$). Quaternions will be introduced in Section III.B. to resolve the gimbal lock.

III. MOTION TRACKING WITH THE OPTITRACK SYSTEM

Figure 3 shows the Optitrack system with six Flex 13 cameras which can detect reflected infrared light from reflective

markers on the Crazyflie 2.0. With the six measurements the Optitrack System can then compute the 3D coordinates of the quadcopter with high precision and frequency. The software *Motive* is used to process the 3D position data as the virtual graph and must be calibrated through a process called *wanding* [7].

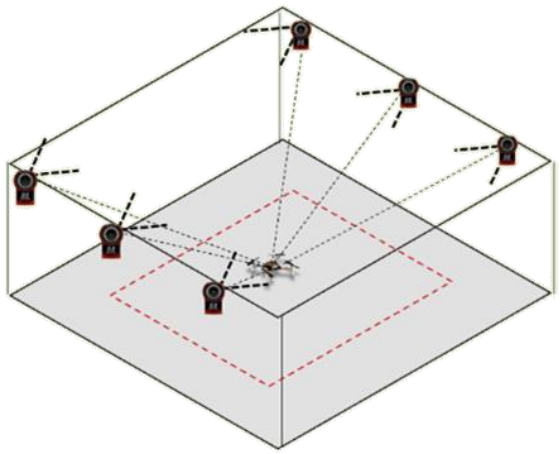

Figure 4. Six Flex 13 cameras detecting the reflective markers on the Crazyflie 2.0. The red area is the effective field of vision.

The red-dotted area in Figure 4 is measured 3m x 2m and is the effective area the system can operate. The Optitrack system is validated in Motive by placing reflective balls around the borders of the effective area and calculating the distance to match the above dimensions. The axis coordinate system for the Optitrack is right-handed North-Up-East (NUE) as show in Figure 5 (left).

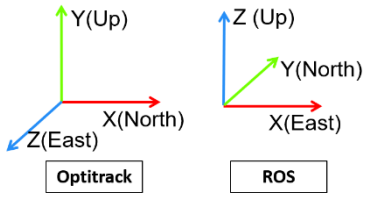

Figure 5. Optitrack North-Up-East (left) versus and ROS East-North-Up (right) coordinate frame convention.

The six Flex 13 Cameras are connected to hub (Optihub 2) which captures the data and sends it to Motive. Motive then computes the 3D virtual graph in real time from the measurements of the reflective markers. This process is illustrated in the top part of the schematic in Figure 6.

As shown in the bottom part of Figure 6, the data is then streamed to a router which broadcasts the position data over to ROS. Motive is Virtual Reality Peripheral Network (VRPN) compatible which allows the data streaming to produce a report identifiable as a movable tracker in ROS with position and orientation. However, since ROS standards follow a different axis coordinate system, East-North-Up (ENU) as illustrated in Figure 5 (right), it is necessary to convert the position and orientation data. The VRPN node therefore has additional code to transform between the Optitrack and ROS frame conventions as explained next [7,12].

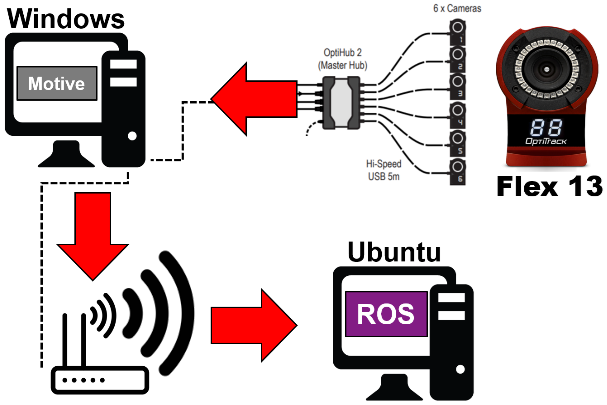

Figure 6. Flow of Optitrack to Motive and streaming through Wi-Fi to ROS

A. *Coordinate Axis Conversion for Position*

Figure 7 illustrates the transformation required to rotate between Optitrack and ROS frames [11]. This is done by applying a 90-degree rotation on the X axis. Using the transformation matrix in (1) and applying a roll rotation of $\varphi = \pi/2$, we can find the required transformation to be

$$H_{Opti}^{ROS} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

The transformation effectively means swapping the Y and Z coordinates and applying a negative sign to the Z coordinate.

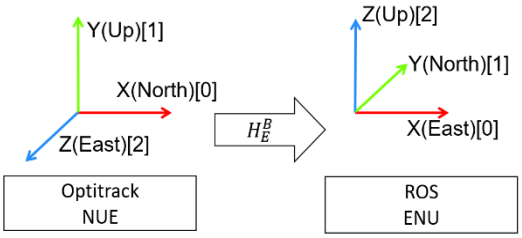

Figure 7. Axis Conversion from NUE to ENU defined in Figure 5.

B. *Coordinate Axis Conversion for Orientation*

To transform the orientation of the Crazyflie 2.0 from Optitrack to ROS frame conventions, we will use quaternions. Quaternions are the generalization of complex numbers to 3D and commonly used in robotics to describe rotations [17]. Following the Euler Theorem of Rotation, quaternions can be introduced as the magnitude of rotation (the angle α) and a vector as [17],

$$q^{opti} = \left[\cos\left(\frac{\alpha}{2}\right), n_1 \sin\left(\frac{\alpha}{2}\right), n_2 \sin\left(\frac{\alpha}{2}\right), n_3 \sin\left(\frac{\alpha}{2}\right)\right] = [w, [x, y, z]]$$

where the vector

$$\vec{n}^{Opti} = [\, n_1, n_2\, n_3] = \sin\left(\frac{\alpha}{2}\right)[x, y, z]$$

defines axis of rotation. To convert the quaternion representation in Optitrack, q^{Opti}, to ROS standards, we need to first isolate the angle part α by taking inverse cosine of w in q^{Opti}. With the angle of rotation, α, we can then compute the vector $\vec{n}^{Opti}$.

Finally, to convert from Optitrack to ROS standards we simply need to apply the transformation H_{Opti}^{ROS} to $\vec{n}^{Opti}$ such that

$$\vec{n}^{ROS} = H_{Opti}^{ROS} \cdot \vec{n}^{Opti} = [N_1, N_2, N_3]$$

with $\vec{n}^{ROS}$ computed in ROS standards, we can then form the new quaternion q^{ROS} as

$$q^{ROS} = \left[\cos\left(\frac{\alpha}{2}\right), N_1 \sin\left(\frac{\alpha}{2}\right), N_2 \sin\left(\frac{\alpha}{2}\right), N_3 \sin\left(\frac{\alpha}{2}\right)\right]$$

which describes the orientation of the Crazyflie 2.0 with respect to the Earth frame E in ROS-based (ENU) convention.

TABLE I. EXAMPLE OF QUATERNION TRANSFORMATION BETWEEN OPTITRACK AND ROS CONVENTIONS FOR $\alpha = \pi/2, \vec{n}^{Opti} = [0,1,2]$.

Orientation	***w***	***x***	***y***	***z***
Optitrack	-1	0	1	2
ROS	-1	0	-2	1

Similar to the position transformation, the orientation transformation effectively swaps the Y and Z coordinates and negates the Z coordinate as illustrated in Table I.

IV. SOFTWARE INTEGRATION IN ROS FOR PATH CONTROL OF THE CRAZYFLIE 2.0

Next, we will demonstrate the integration of the vision-based navigation of the Crazyflie 2.0 in ROS for a simple waypoint tracking control exercise. The software structure for the controller in ROS is illustrated in Figure 8 where the red arrows represent the topics that are subscribed by the controller and the green-colored arrows are published by the controller. Note that the software integration for the control of Crazyflie 2.0 has been adapted from [12].

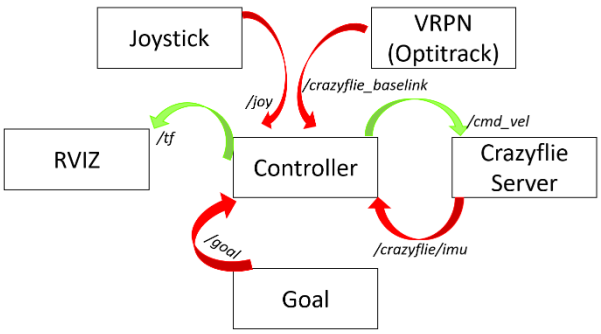

Figure 8. Software structure with ROS for communication.

The VRPN node accepts the position and orientation data from Optitrack and transforms the data to ROS standards. The published data */crazyflie_baselink* has the same structure as */tf* which is subscribed to by RVIZ [7,12]. RVIZ is a ROS tool that displays the 3D data on a ROS graphical user interface. The input */joy* allows the user to command take-off, landing or emergency landing using a joystick. The input */goal* is the desired path and coordinate point which can also be defined by the user and is in ROS coordinates. With published IMU data */crazyflie/imu* from the Crazyflie 2.0 and the waypoints */goal*, the controller node is able to publish the desired velocity */cmd_vel* to the Crazyflie 2.0. These commanded velocity inputs are subsequently processed by the onboard controller as described next.

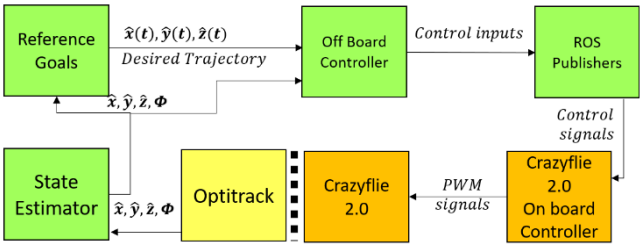

Figure 9. Generic flow of the Crazyflie 2.0 controller.

Figure 9 shows the generic flow of the software integration from a control perspective, i.e. the position and orientation data is extracted from Optitrack which finally lead to control inputs on the Crazyflie 2.0 itself. The boxes in green represent ROS computing, yellow represents Motive processing and orange are operations executed directly onboard the Crazyflie 2.0.

Based on the desired waypoints and current position data the offboard controller in ROS produces the velocity control inputs. This input is then transmitted to the Crazyflie 2.0 using the CrazyRadio and processed by the onboard controller [23]. Finally, the onboard controller computes the motor inputs in the form of Pulse-Width Modulation (PWM) signals for the four motors as [21],

$$PWM_{motor_1} = Thrust - \phi/2 - \theta/2 - \psi$$
$$PWM_{motor_2} = Thrust + \phi/2 - \theta/2 + \psi$$
$$PWM_{motor_3} = Thrust - \phi/2 + \theta/2 + \psi$$
$$PWM_{motor_4} = Thrust + \phi/2 + \theta/2 - \psi$$

V. EXPERIMENTS & TEST RESULTS

The following experimental results demonstrate the software integration of the Optitrack system with ROS, including the transformation of position and orientation data for the first test. The following tests show hover and waypoint tracking to demonstrate the ROS integration for autonomous flight of the Crazyflie 2.0.

A. Position and Orientation Axis Conversion Experiment

The first experiment is to test the conversion between Optitrack and ROS axis systems. Table II shows the results for a position and orientation experiment. For the position test, a location of the Crazyflie 2.0 was measured by hand. Table II compares the measured data to the Optitrack results and the converted ROS data.

Second part of Table II demonstrates the transformation of the quaternion orientation extracted from Optitrack to ROS. The MATLAB test uses the corresponding Euler angles, which can also be extracted from Optitrack, converts the Euler angles to a Direct Cosine Matrix and lastly converts these to quaternions. The MATLAB exercise demonstrates the validity of the quaternion transformation proposed in Section III.B.

TABLE II. EXPERIMENTAL RESULTS TO VALIDATE THE POSITION AND ORIENTATION TRANSFORMATION FROM OPTITRACK TO ROS.

Position		***X* [m]**	***Y* [m]**	***Z* [m]**
Measured		-0.5	0.5	-0.5
Optitrack		-0.502	0.497	-0.503
ROS		-0.502	0.503	0.497
Orientation	***w***	***x***	***y***	***z***
Optitrack	0.754	-0.0227	-0.653	0.0593
MATLAB	0.756	-0.0290	-0.0598	-0.6518
ROS	0.754	-0.0227	-0.0593	-0.653

B. *Hover Experiment*

For the hover test, the Crazyflie 2.0 is commanded to fly to a single waypoint at 0.8 m height, i.e. $[X, Y, Z] = [0, 0, 0.8\,m]$. This simple experiment demonstrates the software integration and the correct interfacing between the different ROS nodes introduced in Figure 8. The hover results in Figure 10 and Figure 11 further show that the Optitrack position and orientation data can be used reliably for the offboard controller to command the Crazyflie 2.0 to fly to the defined waypoint.

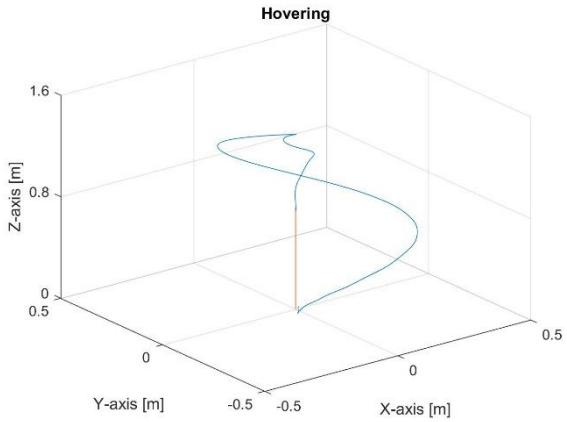

Figure 10. 3D trajectory (blue) for hover experiment to hold 0.8 m height.

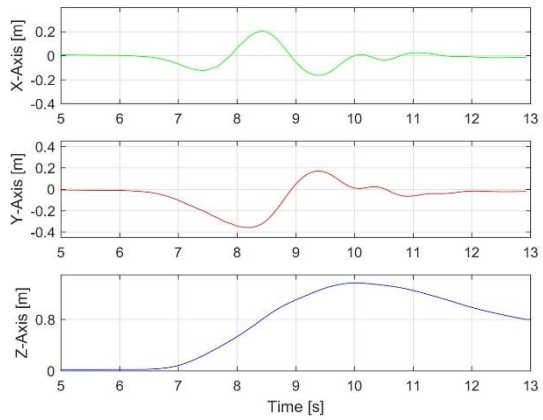

Figure 11. Time history of 3D position components for hover experiment.

Figure 11 shows the corresponding time history of the Crazyflie 2.0 during hover. The measured Z data shows that the quadcopter accelerates first up to 1.2 m before leveling at 0.8m. The X and Y position data show that the controller is able to command the Crazyflie 2.0 to the desired waypoint at $[X, Y, Z] = [0, 0, 0.8\,m]$. The initial deviation from the reference is to overcome the ground effect.

C. *Waypoint Trajectory*

The final experiment demonstrates the waypoint tracking capability of the implemented vision-based control framework for the Crazyflie 2.0. Figure 12, Figure 13 and Figure 14 show the corresponding experimental results for waypoint tracking. Note that the control task here is *not* to follow the trajectory indicated by the orange path in Figure 12 but to fly to a waypoint and hold its position for 3 s at this waypoint before moving to the next waypoint. The colored circles in Figure 12 are the waypoints defined in the code by the user.

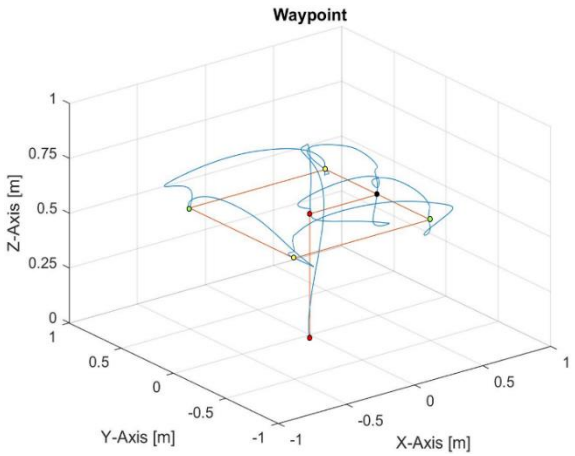

Figure 12. 3D trajectory (blue) for waypoint tracking experiment with circles identifying the waypoints.

Figure 13 shows the corresponding time history of the 3D position components for the waypoint tracking experiment. The dotted curve in Figure 13 represents the reference position of the instantaneous waypoint that is followed at specific time instance. It is clear that the Crazyflie 2.0 is able to reach each waypoint, hover around the waypoint for 3 s and then proceed to the next one.

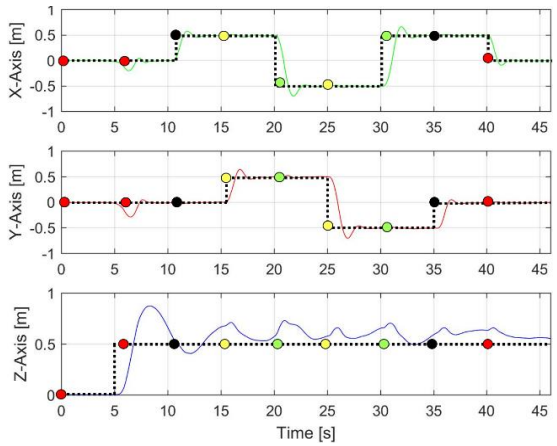

Figure 13. Time history of 3D position components for waypoint tracking experiment. Current waypoint in circles correspond to definition in Figure 12.

The oscillations in the Z direction in Figure 13 arise as the Crazyflie 2.0 is *not* commanded to follow a specific trajectory but to reach the next waypoint. To accelerate in the X and Y directions, the onboard controller tends to also apply excess thrust in the Z direction.

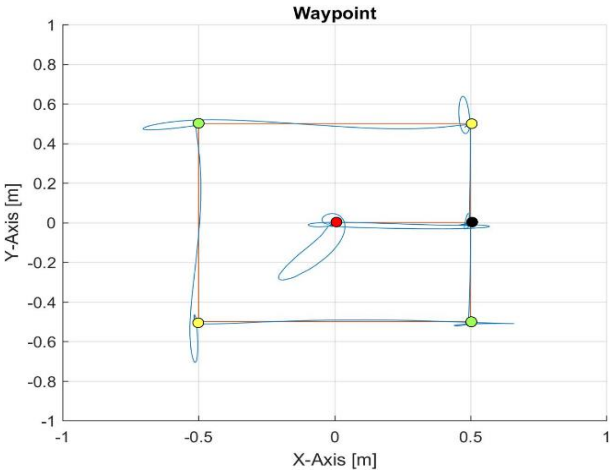

Figure 14. 2D projection of the trajectory in Figure 12.

To demonstrate the waypoint tracking capabilities, Figure 14 shows the projection of the Crazyflie 2.0 trajectory on the $X-Y$ plane only. Figure 14 highlights that the controller can successfully command the Crazyflie 2.0 in dynamic situations with only minor overshoots. The waypoint tracking can be further extended to trajectory control by discretizing the trajectory with multiple waypoints and considering the vehicle dynamics in the discretization.

VI. Conclusion

The objective of this work was to use the motion capture system Optitrack as an external localization system for autonomous flight control of the nano-quadcopter Crazyflie 2.0. This paper provides a comprehensive documentation on how to integrate the Optitrack system with ROS and it especially addresses the transformation between coordinate conventions in ROS and Optitrack. This enables the use of the Optitrack localization system, which is more affordable than the standard VICON positioning system, for educational purposes.

The integration of Optitrack with the communication platform ROS further makes for a versatile platform which can rely on the modular and open-source nature of ROS for a wide range of robot applications. In this work, we have demonstrated this capability by using open-source controller nodes to enable waypoint tracking of the Crazyflie 2.0 nano-copter. This simple integration further highlights the benefits of ROS in combination with Optitrack as an education platform in aerial robotics.

Acknowledgement

The authors would like to thank Dr. Spot Srigrarom from the University of Glasgow Singapore to provide access to the Optitrack UAV facility. The authors would also like to acknowledge the technical advice from Paul Beuchat at ETH Zurich.

References

[1] J. Foerster, “System Identification of the Crazyflie 2.0 Nano Quadrocopter”, Bachelor Thesis, Swiss Federal Institute of Technology (ETH) Zürich, Switzerland.

[2] M. W. Mueller and R. D’Andrea, "A model predictive controller for quadrocopter state interception", in European Control Conference (ECC), Zürich, Switzerland, 2013, pp. 1383-1389

[3] Landry, B. , “Planning and control for quadrotor flight through cluttered environments”, Master’s Thesis, Massachusetts Institute of Technology, USA, 2015.

[4] M. Binder, L. Bieri, M. Meier and N. Melchior, "Indoor Localization Using Ultra-Wide-Band", Master’s Thesis, ETH Swiss Federal Institute of Technology Zurich, Zurich, Switzerland, 2017.

[5] C. Hansen, D. Gibas, J. Honeine, N. Rezzoug, P. Gorce, B. Isableu, "An inexpensive solution for motion analysis”, Journal of Sports Engineering and Technology. Compiegne, France, 2014.

[6] L. Edwards, “Open source robotics and process control cookbook designing and building robust dependable real time system,” Elsevier, New York, USA, 2011.

[7] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R Wheeler, A. Ng “ROS: an open-source Robot Operating System,” ICRA Workshop on Open Source Software, Kobe, Japan, 2009.

[8] D. WONG, "Plan for drones to deliver parcels takes flight", The Straits Times, Feb 9, 2018.

[9] G. Warwick, "Aircraft Inspection Drones Entering Service With Airline MROs", MRO network, Apr 04, 2018.

[10] "Cutting-edge drones and unmanned vehicles to boost Singapore's surveillance capabilities", Channel NewsAsia, 2018.

[11] Evan G. Hemingway “Perspectives on Euler Angle Singularities, Gimbal Lock, and the Orthogonality of Applied Forces and Applied Moments.”Article, University of California. Berkeley, CA, USA, 2017.

[12] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, N. Ayanian “Mixed Reality for Robotics”, IEEE Intl Conf. on Intelligent Robots and Systems, Hamburg, Germany. 2015.

[13] K. Kaur, The Straits Times, "Five projects to kick off unmanned aircraft system trials at Singapore's first drone estate", 2018.

[14] C. Luis, J. Le Ny, "Design of a Trajectory Tracking Controller for a Nanoquadcopter" Technical Report, Polytechnique Montreal. Montreal, Canada, 2016.

[15] J. Diebel, “Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors”, Stanford University. Stanford, California, USA, 2006

[16] J. A. Preiss, W. Hoenig, G. S. Sukhatme and N. Ayanian “Crazyswarm: A Large Nano-Quadcopter Swarm”, IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017.

[17] A. Hanson, “Visualizing quaternions,” Morgan Kaufmann, San Francisco, CA, USA, 2006, pp. 43-56.

[18] J. Bjorn, M. Kjaegaard, J. A. Larsen “Autonomous Hover for a quad rotor helicopter”, Aalborg University, master’s Thesis, Aalborg, Denmark, 2007.

[19] M. Wierema, “Design, implementation and flight test of indoor navigation and control system for a quadrotor UAV” TU Delft, Master of Science Thesis, Delft, Netherlands, 2008.

[20] J. Mišeikis, “INF3480 - Introduction to Robot Operating System,” Lecture Slides, University of Oslo, Norway, 2017.

[21] G.P. Subramanian, “Nonlinear control strategies for quadrotors and CubeSats”, Master Thesis, University of Illionis, USA, 2015.

[22] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, “Design, modeling, estimation and control for aerial grasping and manipulation,” IEEE Intl Conf. on Intelligent Robots and Systems, San Francisco, CA, USA, 2011.