Flight tests of the path following vector field method on the formation of Crazyflie 2.1 nano quadcopters

Tagir Muslimov

December 20, 2022

Flight tests of the path following vector field method on the formation of Crazyflie 2.1 nano quadcopters. The Lighthouse positioning system is used for experiments: https://www.bitcraze.io/documentation/lighthouse/

1 Basic Algorithm

1.1 Notation

(CX, CY) (or (c_e, c_n)) $-$ circle center coordinates
$k > 0$ $-$ smoothness coefficient for the path following control law
$R = const$ (or ρ) $-$ radius of the circular path
$v_f = const$ $-$ the maximum value of the additional speed component
v_{cruis} $-$ final cruising speed of a Crazyflie formation
$D_{12} = const$ $-$ the desired angular distance between the 1st and 2nd copters (we also call this value the desired phase shift)
$D_{23} = const$ $-$ the desired angular distance between the 2nd and 3rd copters
Required condition for testing on three UAVs: $D_{12} + D_{23} > 2\pi$

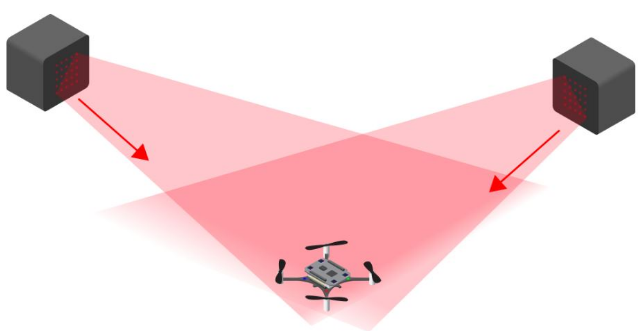


Figure 1: Lighthouse Positioning System

$k_f > 0$ – smoothness coefficient for the speed control law
T_Z – specified altitude of formation flight
v_z – vertical speed for takeoff or landing
d_i – distance from the ith Crazyflie to a circle center
phi (or φ_i) – phase angle of the ith Crazyflie
angle (or χ_i^c) – command course angle of the ith Crazyflie
vx (or v_x) – speed of Crazyflie along the x-axis in the world (global) coordinate system
vy (or v_y) – speed of Crazyflie along the y-axis in the world (global) coordinate system
vz (or v_z) – speed of Crazyflie along the z-axis in the world (global) coordinate system
px (or p_e) – coordinate of Crazyflie along the x-axis in the world (global) coordinate system
py (or p_n) – coordinate of Crazyflie along the y-axis in the world (global) coordinate system
kalman.stateX – estimation of the copter's position with the Kalman filter along the x-axis in the world (global) coordinate system
kalman.stateY – estimation of the copter's position with the Kalman filter along the y-axis in the world (global) coordinate system

1.2 Course Angle Control Law

Control law for the Crazyflie course angle:

$$\chi_i^c = \varphi_i + \lambda \left[\frac{\pi}{2} + \operatorname{atan}(k(d_i - \rho)) \right], \tag{1}$$

where $\lambda = 1$ means clockwise motion and $\lambda = -1$ means counterclockwise motion. This law is based on the one presented in the monograph [1].

1.3 Speeds Control Law

Control law for the Crazyflie speeds:

$$\begin{bmatrix} v_1^c \\ v_2^c \\ v_3^c \end{bmatrix} = \begin{bmatrix} v_{cruis} \\ v_{cruis} \\ v_{cruis} \end{bmatrix} + \begin{bmatrix} v_f\,(2/\pi)\arctan\left(k_f\,(\Delta\varphi_{12} - D_{12})\right) \\ v_f\,(2/\pi)\arctan\left(k_f\,(-\Delta\varphi_{12} + \Delta\varphi_{23} + D_{12} - D_{23})\right) \\ v_f\,(2/\pi)\arctan\left(k_f\,(-\Delta\varphi_{23} + D_{23})\right) \end{bmatrix}, \tag{2}$$

where
v_i^c - linear speed command for Crazyflie;
$v_{cruis} = const$ - final cruising speed of the formation;
$v_f = const$, $v_f \leq v_{cruis}$ - the maximum value of the additional speed component;
$k_f > 0$ - adjustable coefficient;
$\Delta\varphi_{i,j}$ (or $p_{i,j}$) - current phase shift between the i-th and j-th drone; in the code, this value is denoted as p_{12} for the phase shift between the 1st and 2nd drone

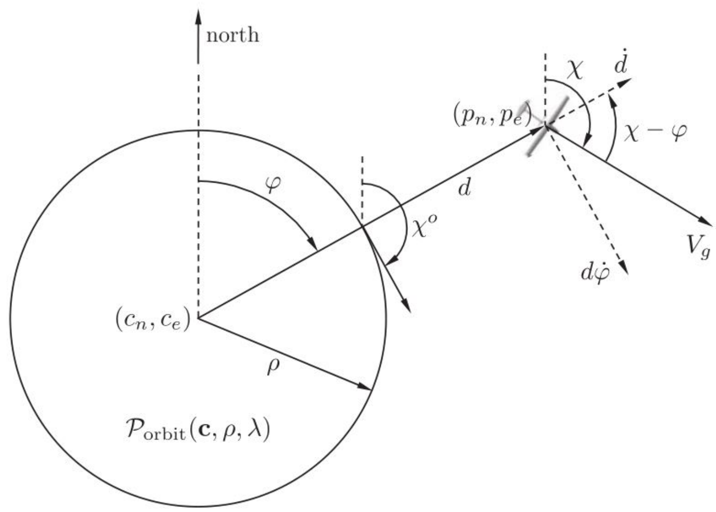


Figure 2: Notation used. Picture from [1]

and as p_{23} for the phase shift between the 2nd and 3rd drone.
$D_{i,j} = const$ - desired phase shift between the i-th and j-th drone.
Required condition for testing on three UAVs: $D_{12} + D_{23} > 2\pi$

How do we calculate the current phase shift (for example, $\Delta\varphi_{12}$)?

$\text{dot}_{product} = (p_{e_1} - c_e) \cdot (p_{e_2} - c_e) + (p_{n_1} - c_n) \cdot (p_{n_2} - c_n);$

$\text{magnitude}_1 = ((p_{e_1} - c_e)^2 + (p_{n_1} - c_n)^2)^{1/2};$

$\text{magnitude}_2 = ((p_{e_2} - c_e)^2 + (p_{n_2} - c_n)^2)^{1/2};$

$\text{triple}_{product} = (p_{e_1} - c_e) \cdot (p_{n_2} - c_n) - (p_{e_2} - c_e) \cdot (p_{n_1} - c_n);$

$\cos \Delta\varphi_{12} = \text{dot}_{product} / (\text{magnitude}_1 \cdot \text{magnitude}_2) \Rightarrow$

$\Rightarrow \Delta\varphi_{12} = \arccos(\text{dot}_{product} / (\text{magnitude}_1 \cdot \text{magnitude}_2));$

if $\text{triple}_{product} > 0$
then $\Delta\varphi_{12} := 2\pi - \Delta\varphi_{12};$ **end**
out $= [\Delta\varphi_{12}]$

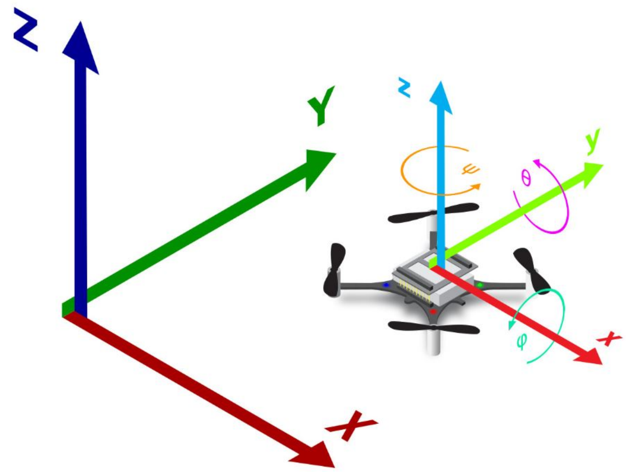


Figure 3: The Coordinate System of the Crazyflie 2.X https://www.bitcraze.io/documentation/system/platform/cf2-coordinate-system/

1.4 Crazyflie 2.1 Body Frame and Rotation Matrices

We calculate commands in the global (world) coordinate system. To get commands in the body coordinate system, we need to do a transformation with rotation matrices.

- **roll** and **yaw** are clockwise rotating around the axis looking from the origin (**right-hand-thumb**)
- **pitch** are counter-clockwise rotating around the axis looking from the origin (**left-hand-thumb**)

Formula (2.5) from [1] is modified:

$$\mathbf{R}(\varphi, \theta, \psi) = \mathbf{R}(\varphi) \cdot \mathbf{R}^T(\theta) \cdot \mathbf{R}(\psi) =$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \mathbf{R}(\varphi, \theta, \psi) \cdot \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix}, \tag{3}$$

where $\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$ is a vector of commands in body-frame and $\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix}$ is a vector of commands in global(world)-frame.

1.5 Velocity Control via Python API

If we use **def send hover setpoint(self, vx, vy, yawrate, zdistance)**, then the value z is taken as a *constant*. Therefore v_z from the above vector (3) does not end up being used. However, in order to correctly calculate v_x and v_y, we must use 3-by-3 matrices.

The resulting control law for implementation based on (1)-(2):

$$\begin{aligned} \dot{p}_x &= v_i^c \cdot \sin(\chi_i^c) \\ \dot{p}_y &= v_i^c \cdot \cos(\chi_i^c) \end{aligned} \tag{4}$$

Using (3):

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \mathbf{R}(\varphi, \theta, \psi) \cdot \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ 0 \end{bmatrix},$$

These values are fed to the function **def send hover setpoint(self, vx, vy, yawrate, zdistance)**

1.6 Setpoint Control via Python API

If we use **def send position setpoint(self, x, y, z, yaw)**:

$$\begin{aligned} x &= x_{t+1} = x_t + \dot{p}_x \cdot \Delta t \\ y &= y_{t+1} = y_t + \dot{p}_y \cdot \Delta t \\ z &= const \end{aligned} \tag{5}$$

where
x_t and y_t are the Crazyflie positions at the current moment in time, which can be obtained from the Kalman filter through the values of the logged variables **kalman.stateX** and **kalman.stateY**;
Δt is a fairly small time step;
$\dot{p}_x$ and $\dot{p}_y$ are the global frame speed commands from (4). These values from (5) are fed to the function **def send position setpoint(self, x, y, z, yaw)**

2 The Results of the Experiments

The following presents the experimental results for the method described in section 1.6. The code from the file **CircularMotion setpos 2 copters.py** was used for the experiments.

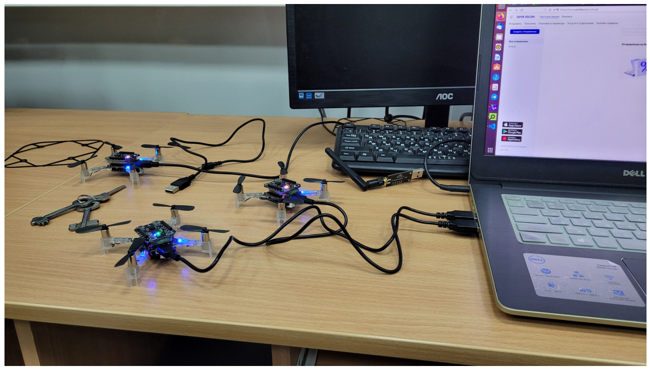
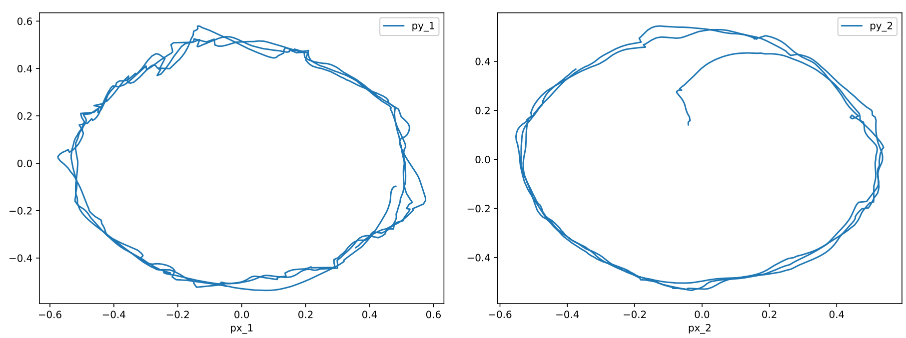


Figure 4: Crazyflie 2.1 copters used in the experiments

2.1 Experiments on two copters

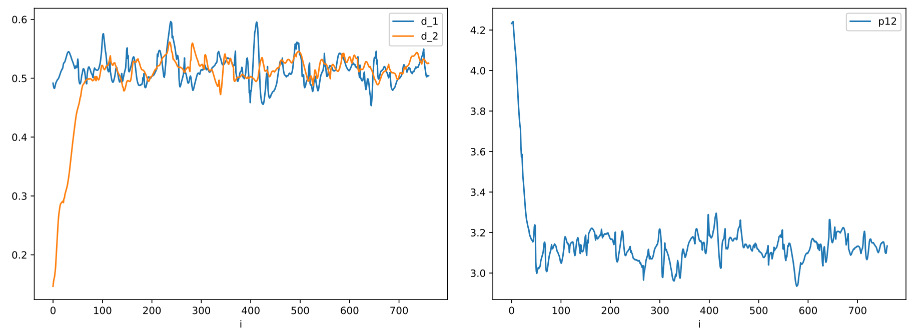
Preliminary video of the experiments is available at the link:
`https://youtu.be/Ushn_94qVtE`

2.1.1 Stationary center of the circular path



(a) (b)

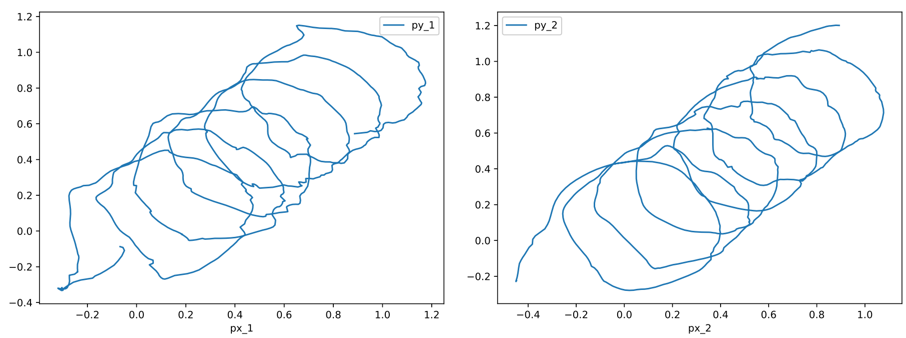
Figure 5: Two copters flight with a stationary center. (a) Trajectory of the 1st copter; (b) trajectory of the 2nd copter



(a) (b)

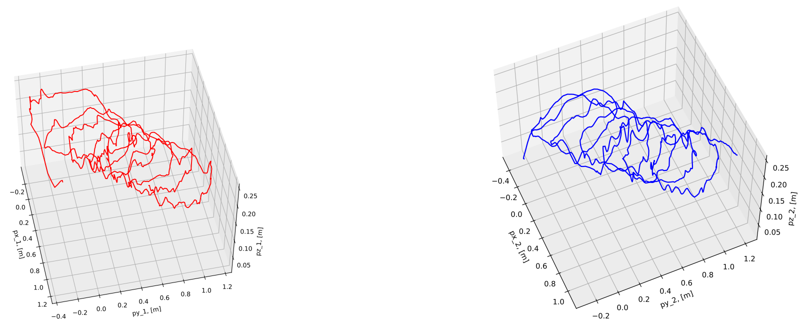
Figure 6: Two copters flight with a stationary center. (a) Distances to the circle center; (b) phase shifts

2.1.2 Moving center of the circular path



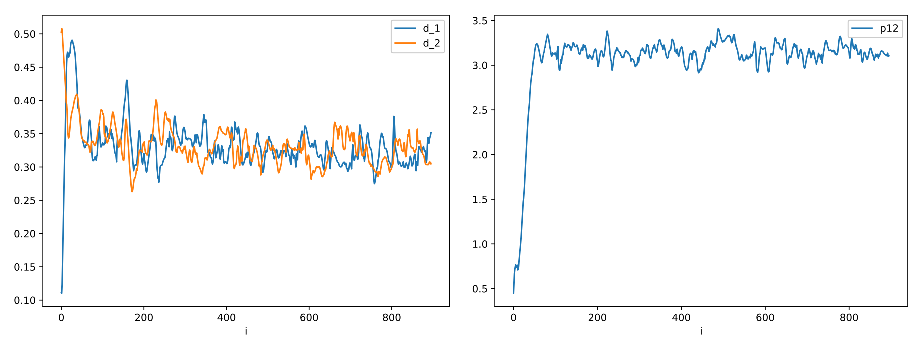
(a) (b)

Figure 7: Two copters flight with a moving center. (a) Trajectory of the 1st copter; (b) trajectory of the 2nd copter



(a) (b)

Figure 8: Two copters flight with a moving center. (a) 3D trajectory of the 1st copter; (b) 3D trajectory of the 2nd copter



(a) (b)

Figure 9: Two copters flight with a moving center. (a) Distances to the circle center; (b) phase shifts

2.2 Experiments on three copters

2.2.1 Stationary center of the circular path

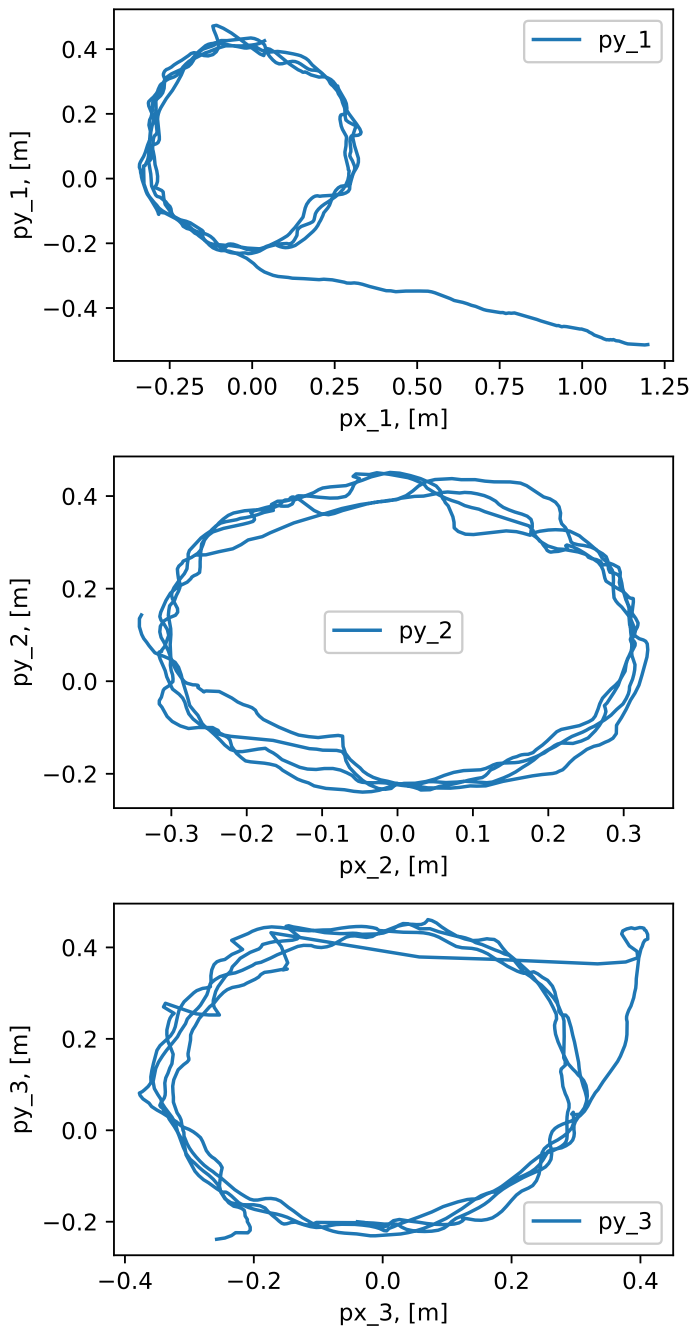
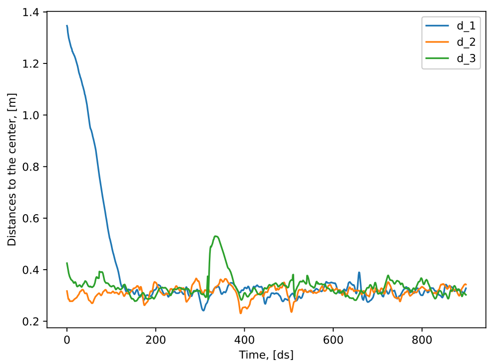
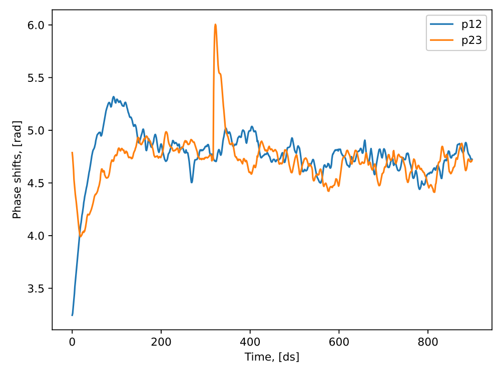


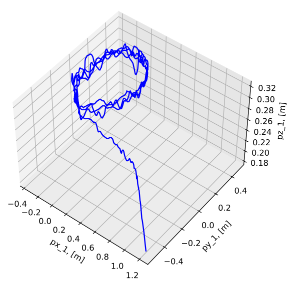
Figure 10: Three copters flight with a stationary center. Trajectories

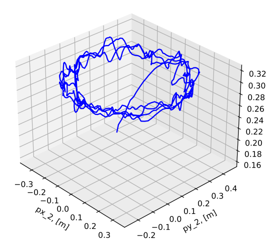




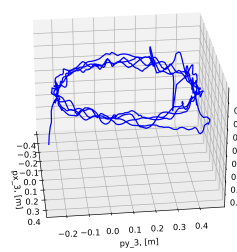
(a) (b)

Figure 11: Three copters flight with a stationary center. (a) Distances to the circle center; (b) phase shifts





(a) (b)



(c)

Figure 12: Three copters flight with a stationary center. (a) 3D trajectory of the 1st copter; (b) 3D trajectory of the 2nd copter; (c) 3D trajectory of the 3rd copter

2.2.2 Moving center of the circular path

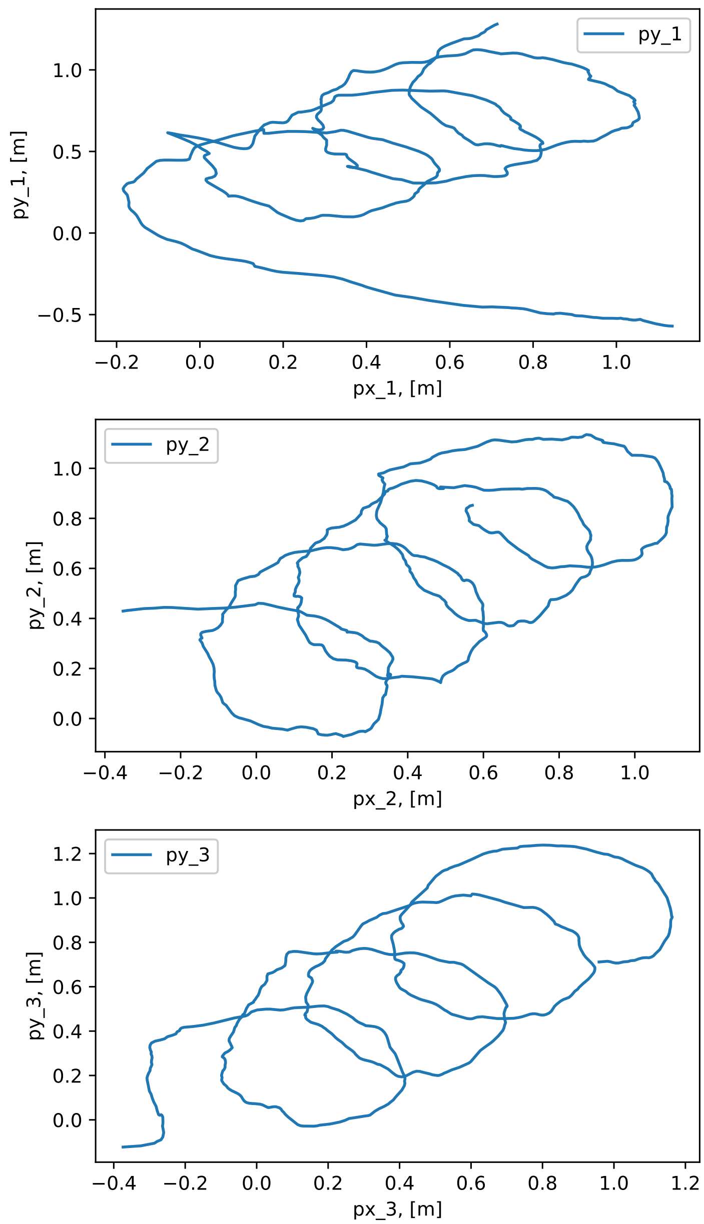
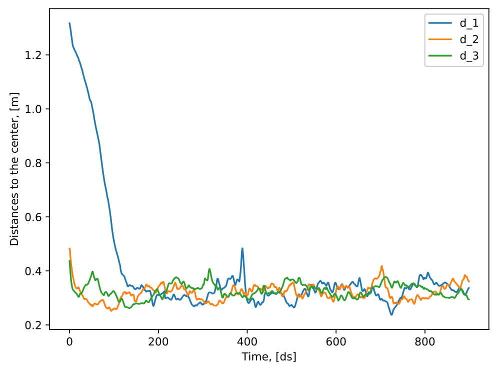
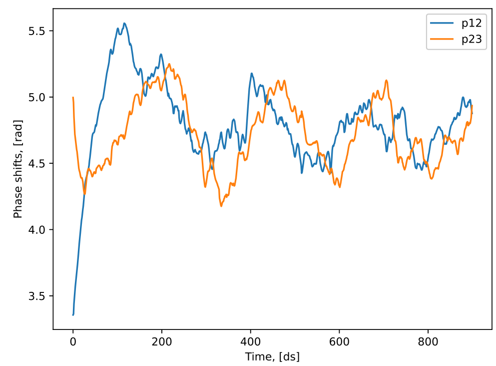


Figure 13: Three copters flight with a moving center. Trajectories

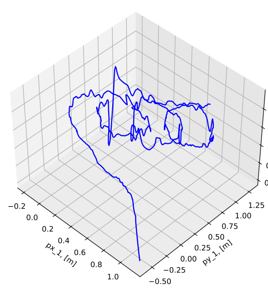


(a)

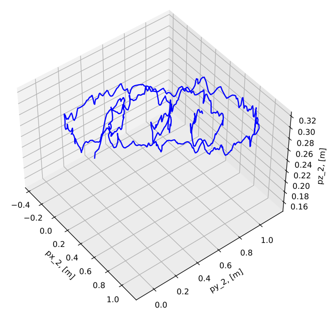


(b)

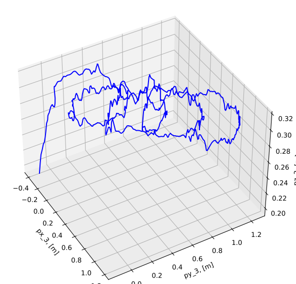
Figure 14: Three copters flight with a moving center. (a) Distances to the circle center; (b) phase shifts



(a)



(b)



(c)

Figure 15: Three copters flight with a moving center. (a) 3D trajectory of the 1st copter; (b) 3D trajectory of the 2nd copter; (c) 3D trajectory of the 3rd copter

References

[1] Beard, R. W., McLain, T. W. (2012). Small unmanned aircraft: Theory and practice. Princeton university press.