ÉCOLE CENTRALE DE NANTES

MASTER ARIA-ROBA

"AUTOMATIQUE, ROBOTIQUE ET INFORMATIQUE APPLIQUÉE"

2017 / 2018

Master Thesis Report

Presented by

LI Zhongmou

21/07/2017

Motion planning and control of a Flying Parallel Robot

President:	Philippe Martinet	Full Prof., LS2N, ECN
Evaluators:	Philippe Martinet	Full Prof., LS2N, ECN
	Damien Six	Dr., LS2N, ECN
	Abdelhamid Chriette	Ass. Prof.,LS2N, ECN
	Sébastien Briot	Researcher,LS2N,ECN
Supervisor(s):	Damien Six	Dr., LS2N, ECN
	Abdelhamid Chriette	Ass. Prof.,LS2N, ECN
	Sébastien Briot	Researcher,LS2N,ECN

Laboratory: Laboratoire des Sciences du Numérique de Nantes LS2N

Acknowledgements

Je suis arrivé en France et cela fait deux ans que j'étudie *à* Centrale. Pendants ces deux années, j'ai appris entre autre la robotic, la commande et la physique des drones.

Je voudrais remercier ma famille qui m'a soutenu durant mon master et specialement ma sœur, qui soigne notre père malade depuis une année. Sans leur soutien, je n'aurais pas pu continuer à étuder en France et finir mon mémoire.

J'ai reçu beaucoup d'aide de la part de mes directeurs : SIX Damien, CHRIETTE Abdelhamid et BRIOT Sébastien, sur mon projet. Ils m'ont conseillé sur mon mémoire, specialement sur la méthodologie de la recherche et comment faire une presentation. Grace à leur aide et leur patience, j'ai appris pas à pas à m'ameliorer professionnellement.

Durant mon mémoire, l'équipe ARMEN nous a proposé des ateliers sur des sujets variés qui m'ont permit de me renseigner sur l'état de l'art des robots. Je tiens à remercier mes amis et collègues Benjamin, Max, Rabbia et toutes les personnes de ARIA et EMARO ainsi que Pauline de l'INSA, qui ont travaillé sur leur mémoires et leur stages en même temps que moi nous permettant de discuter ensemble pendant les pauses. Cela, ma permis de relacher la pression et de rester concentré et motivé durant ces six mois.

Finalement, j'aimerais remercier toutes les personnes que je n'ai pas cité précédemment et qui m'ont aidé sur mon projet.

Abstract

This master thesis report presents a motion optimisation method for a new aerial robot Flying Parallel Robot (FPR) with respects to its dynamics and payloads constraints.

In the Chapter 1, an overview of a quadrotor and its potential research interest are discussed at first. New design conceptions of quadrotor and platform are presented in a brief way. As the object robot FPR is a new robot design, the trajectory generation of a single drone is addressed and analysed as references. We explore the on-broad system navigation guidance control system of a drone and the typical control system structure, which are the backgrounds and influence trajectory generation. This is followed by state of art about the trajectory generation. Also, we address introduction of Type 2 singularity for parallel robots and methods for crossing such singularity. We introduce the object robot FPR which consists of a passive chain and two drones as well as its advantages in applications.

In the Chapter 2 Dynamic model and control system of FPR, we recall the dynamic model of FPR and the controller proposed by designers of FPR. To begin with, the dynamic model consists of two parts: passive chain dynamic model and attitude dynamic model. Computed torque control are applied in those two loops and one coupling term in the dynamic model is treated as a perturbation in the controller. This perturbation term plays an significant part in the following motion optimisation.

In the Chapter 3 Optimisation motion planning for FPR, we first formulate the motion planning problem statement and choose 9-degree polynomial to generate a trajectory. Then, a optimisation motion planning is presented: way points without its positions and via points work as decision variables (a point in defined as position, velocity, acceleration, third and fourth derivatives); the objective function is built based on the energy cost and the perturbation in the controller. Then, the parameters in the optimisation algorithm and results are analysed later. Subsequently, this optimisation method is validated by the simulation in Matlab. A prototype robot built in LS2N is used to test the optimised motion in real experiment.

In the Chapter 4, two approaches to cross the Type 2 singularity for FPR prototype are discussed and analysed. This chapter is began by addressing the problem statement of crossing singularity for the PFR robot prototype. Then, two different approaches are used to cross the singularity and final results are presented and summarised.

Notations

BS	backstepping
DOF	degree of freedom
FBL	feedback linearisation
FPR	flying parallel robots
GNC	guidance, navigation, and control
RUAV	rotor unmanned aircraft vehicle
CoM	centre of mass
RUAS	rotor unmanned aircraft system
SMC	sliding mode control
MDPT	minimisation derivatives of the position trajectory
UAV	unmanned aircraft vehicle
Llink	length of FPR's link
d_i	the distance between the joint i and the CoM of quadrotor i
$\mathbf{f}$	input force of the FPR
$\mathbf{fp}_1, \mathbf{fp}_2$	force applied by drone 1 and drone 2 to the passive chain of FPR
f_{1z}, f_{2z}	thrust force of drone 1 and drone 2 of FPR
f_{im}	the force produced by a rotor of a done
F_{max}	the maximum thrust force of a drone
$\mathbf{g}$	acceleration of gravity
$\mathbf{J}$	objective function in the motion optimisation
$\mathbf{J}_d$	Jacobian matrix relating $\mathbf{q}_d$ and $\mathbf{q}_a$
$\mathbf{J}_e$	energy consumption in the objective function in the motion optimisation
$\mathbf{J}_\phi$	perturbation index in the objective function in the motion optimisation
$\mathbf{J}_p$	Jacobian matrix relating the $\mathbf{t}_p$ and $\mathbf{q}_a$
$\mathbf{J}_{pr}$	Jacobian matrix relating $[\dot{y}_{j3}, \dot{z}_{j3}]^T$ and $[\dot{q}_1, \dot{q}_2]^T$
m_{di}	the mass of the quadrotor i
n_{vp}	number of via points
p, q, r	rotational rates about body axes of a drone
$\mathbf{p}$	perturbations in the controller of FPR

q_1, q_2	angles of joint 1 and joint 2 of FPR prototype
q_{11}, q_{21}	actuated joints of 5 bar mechanism
q_{12}, q_{13}, q_{22}	passive joints of 5 bar mechanism
q_{21}	the relative angle between two links of FPR
$\mathbf{q}$	state variable of FPR and $\mathbf{q} = \begin{bmatrix} x & y & z & \psi & \theta & \phi & q_{21} & \phi_1 & \phi_2 \end{bmatrix}^T$
$\mathbf{q}_p$	coordinate of the passive chain of FPR and $\mathbf{q}_p = \begin{bmatrix} y & z & \phi & q_{21} \end{bmatrix}$
$\mathbf{q}_a$	coordinate of the attitude of FPR and $\mathbf{q}_a = \begin{bmatrix} \psi & \theta & \phi_1 & \phi_2 \end{bmatrix}$
$\mathbf{r}_i$	positions of joint i in the vertical plane in the world frame
$\mathbf{R}_i$	is the rotation matrix about ϕ_1 in the vertical plane
$\mathbf{s}$	configuration state of FPR
$\mathbf{S}$	points in motion planning of FPR and $\mathbf{S} = \begin{bmatrix} \mathbf{s} & \dot{\mathbf{s}} & \ddot{\mathbf{s}} & \mathbf{s}^{(3)} & \mathbf{s}^{(4)} \end{bmatrix}$
$\mathbf{S}_{d1}, \mathbf{S}_{d2}$	decision variables in the motion optimisation
$\mathbf{t}_p$	the platform twist of 5 bar mechanism
$\mathbf{v}_1, \mathbf{v}_2$	linear velocity of drone 1 and drone 2 of FPR
x_z, y_z, z_z	translational coordinate of a drone in the world frame
x_e, y_e	coordinates of the end-effector of 5 bar mechanism
x, y, z	translational coordinate of the FPR end-effector in the world frame
y_{j1}, y_{j2}, y_{j3}	horizontal coordinate of joint 1, 2, 3 of the FPR prototype
z_{j1}, z_{j2}, z_{j3}	vertical coordinate of joint 1, 2, 3 of the FPR prototype
$\mathbf{x}_z$	$\mathbf{x}_z = [x_z, y_z, z_z]$
$\mathbf{x}_e$	$\mathbf{x}_e = [x_e, y_e]$
$\mathbf{x}$	$\mathbf{x} = [x, y, z]$
α	the weight in the objective functions of optimisation process
$\boldsymbol{\gamma}$	$\boldsymbol{\gamma} = [p, q, r]$
ψ_z	yaw angle of a drone in the world frame
ϕ_1, ϕ_2	roll angles of drone 1 and drone 2 of FPR
ϕ, θ, ψ	roll angle, pitch angle and yaw angle of the end-effector
τ_{im}	the torque produced by the rotor m of the done i
$\tau_{bx}, \tau_{by}, \tau_{bz}$	are roll torque, pitch torque and yaw torque of a drone.
$\boldsymbol{\tau}_e$	the real robot input efforts of 5 bar mechanism

$\boldsymbol{\tau}_{ta}$	the virtual input efforts in the actuated joints of 5 bar mechanism
$\boldsymbol{\tau}_{td}$	the virtual input efforts in the passive joints of 5 bar mechanism
$\boldsymbol{\tau}_{b}$	drag torque $\boldsymbol{\tau}_b = [\tau_{bx}, \tau_{by}, \tau_{bz}]$
$\boldsymbol{\delta}, \boldsymbol{\delta}_p$	perturbations representations in the close-loop of the passive chain
ϖ_{im}	the rotor angular speed of a drone
$\boldsymbol{\xi}$	configuration state of FPR prototype and $\boldsymbol{\xi} = \begin{bmatrix} q_1 & q_2 \end{bmatrix}$
$\boldsymbol{\Xi}$	$\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi} & \dot{\boldsymbol{\xi}} & \ddot{\boldsymbol{\xi}} & \boldsymbol{\xi}^{(3)} & \boldsymbol{\xi}^{(4)} \end{bmatrix}$

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter is dedicated to introduce Flying Parallel Robot and the state of art trajectory generation for a single quadrotor.

First of all, a quadrotor and its application as a robot platform are introduced. By giving a brief and a big overview of quadrotor applications, we aim to show the potential research interests of quadrotors. Then, an overview of the on-board system of a quadrotor, navigation, guidance and control (NGC), is given to understand trajectory generation for a quadrotor system. The whole NGC structure and typical control system are introduced then. In the third part, trajectory generation and its state of art of the methods are explored, especially two main approaches: minimisation derivatives of the position trajectory and optimal control approach. Their models are built and analysed in details in this part. Crossing Type 2 singularity is another important task in this thesis. State of art about methods for crossing Type 2 singularity for parallel robots are stated. This is followed by the introducing the conception of FPR that is our object robot.

1.1 State of art quadrotor design and application

Generally speaking, a quadrotor or drone is an aerial mechanism with six DOF, i.e three translations and three rotations. It consists of four individual rotors attached to a rigid cross airframe [23]. One example is XAircraft X650 shown in the Fig.1.1.

Fig. 1.1. XAircraft X650

A quadrotor has been deeply investigated in recent years and it has been applied in many different areas, because of its rapid manoeuvrability, easy maintainability, low-cost manufacturability with reduced mechanical complexity, stable hovering capability [29, 28]. Now, researchers now are trying to apply drones as versatile platforms. Quadrotors and some mechanics like a manipulator are used to build a complex flying platform. Main applications can be regrouped into three categories: a flying grasper, a cable-suspended transportation and an aerial manipulation with a tool.

A flying grasper

A high speed and avian-inspired grasping quadrotor is able to perform with pick-up velocities at 2m/s and 3m/s in the real experiment[35].

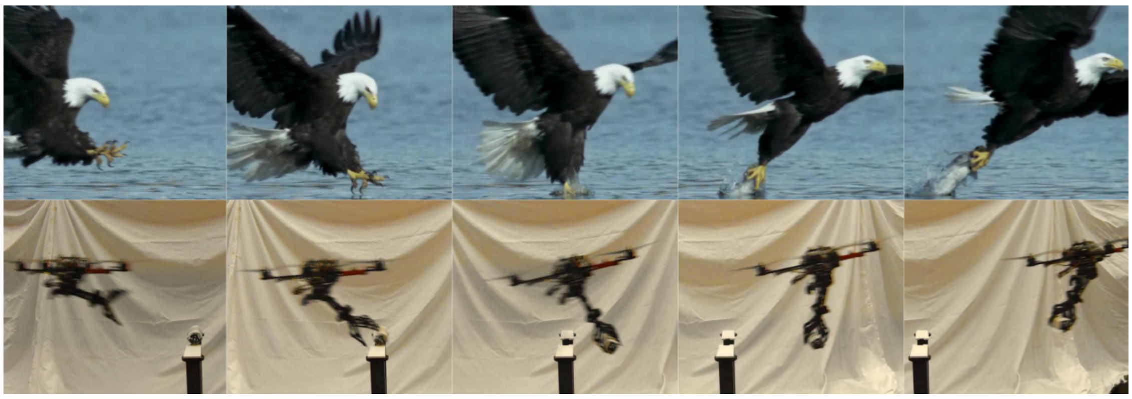


Fig. 1.2. A still image comparison between the eagle and the quadrotor for a trajectory with the quadrotor moving at 3 m/s (9 body lengths/s) at pickup.[35]

A cable-suspended transportation

Montserrat Manubens et al. propose the robot FlyCrane consisting of three aerial robots connected to a platform by three pairs of cables, as illustrated in Fig.1.3.

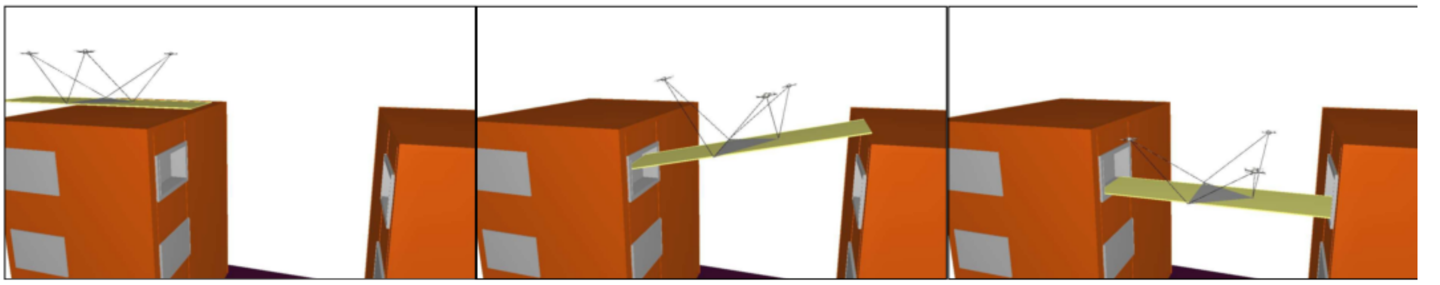


Fig. 1.3. The Rescue problem: the FlyCrane has to install a lightweight footbridge between two buildings for a rescue operation.[24]

An aerial manipulation with a tool

Fig.1.4 shows that Christopher Korpela et al.[20] built an aerial vehicle with dual multi-degree of freedom manipulators. They also developed control methods and tested them in the flight tests.

To sum up, the quadrotor is not only working as a flying robot, but also can be a working

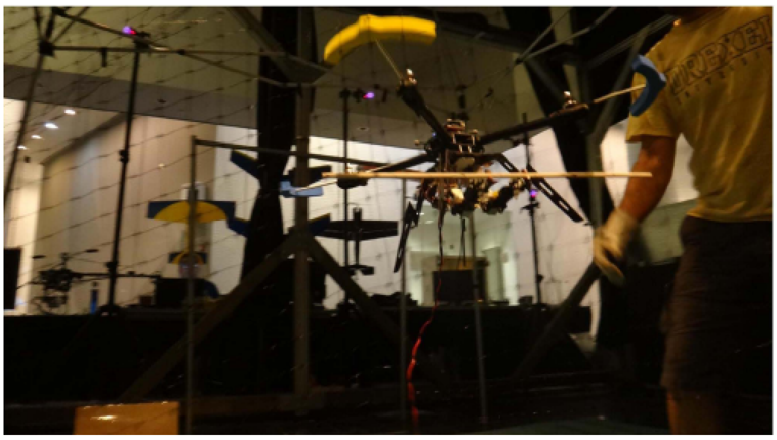


Fig. 1.4. MM-UAV carrying a long rod[20]

platform on which other devices can be put. In such a way, the the workspace of a robot can extended dramatically.

1.2 Navigation, guidance, control system of a quadrotor

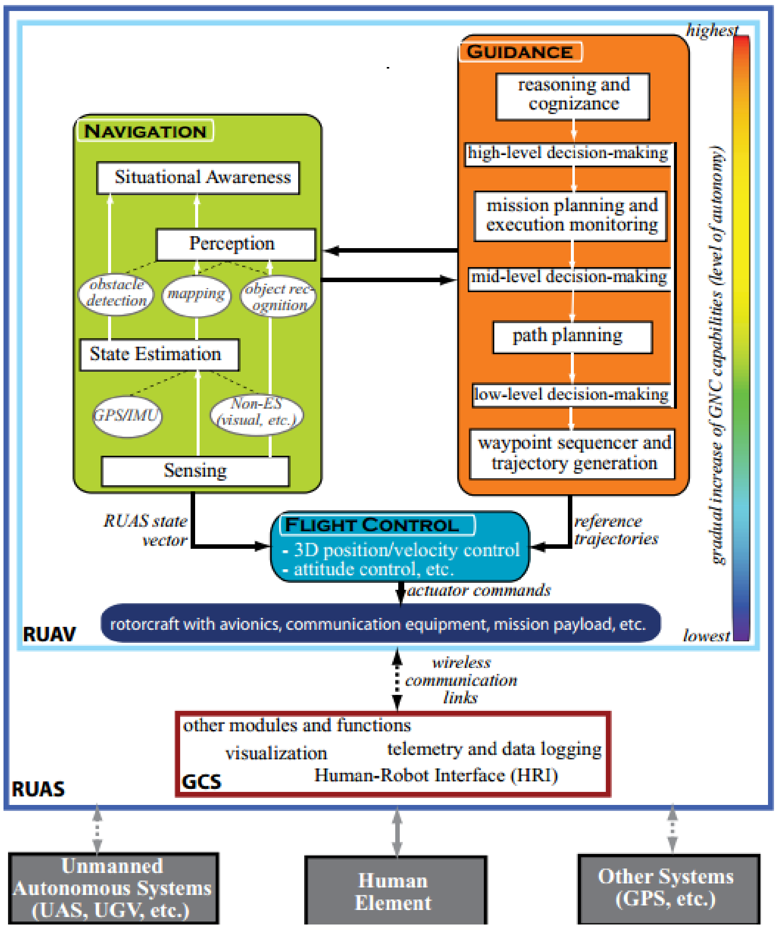
In order to have a big picture of the trajectory generation of a drone, we briefly discuss the on-board system including navigation, guidance and control subsystems and address the state of art about the trajectory generation methods.

An on-board system of a quadrotor can be regrouped into three main categories: navigation, guidance and control (NGC). NGC are expected to finish missions without direct or continuous human control. Especially, the **guidance system** has the highest priority compared to the **navigation system** and the **control system**. The components of **navigation system** and **guidance system** are shown in the Fig.1.5.

Navigation system is to figure out "where we are". For autonomous vehicles such as robots and UAVs, navigation can be defined as the process of extracting information about the vehicle's self-motion and also the surrounding environment [28]. Typically, most of the sensors of UAV work in the frame of navigation system. Navigation system can estimate the vehicle's attitude, angular rates, height, velocity and position with respect the word frame or relative to the target. In some advanced navigation system, mapping, obstacles detection and localization can also be done.

Guidance system works like a driver of UAV finishing the mission and ensuring the safety at the same time. It takes information "where we are" and "where the target is" from the navigation system and generate a trajectory that UAV will track to reach the destination respecting the obstacles and constrains of UAV.

In the guidance system,



highest
GUIDANCE
reasoning and cognizance
high-level decision-making
mission planning and execution monitoring
mid-level decision-making
path planning
low-level decision-making
waypoint sequencer and trajectory generation
NAVIGATION
Situational Awareness
Perception
obstacle detection
mapping
object recognition
State Estimation
GPS/IMU
Non-ES (visual, etc.)
Sensing
gradual increase of GNC capabilities (level of autonomy)
RUAS state vector
FLIGHT CONTROL
- 3D position/velocity control
- attitude control, etc.
reference trajectories
actuator commands
rotorcraft with avionics, communication equipment, mission payload, etc.
RUAV
lowest
wireless communication links
other modules and functions
visualization
telemetry and data logging
Human-Robot Interface (HRI)
GCS
RUAS
Unmanned Autonomous Systems (UAS, UGV, etc.)
Human Element
Other Systems (GPS, etc.)

Fig. 1.5. Overall structure of guidance navigation and control systems onboard of a RUAS [18].

1. Mission Planning refers to the process of generating tactical goals, a route (general or specific), a commanding structure, coordination, and timing for a RUAS or a team of unmanned systems [17, 18]. The mission, the destination for example, can be pre-defined and uploaded to the UAV or can be generated online.

2. Path Planning is to find the best and safest way, usually a curve, to reach a goal position/configuration or to accomplish a specific task [18]. The path does not contain any time information.

3. Trajectory Generation produces different time-respecting motion functions (reference position, reference heading, etc.) that are physically possible, satisfy RUAS dynamics and constraints. and can be directly used as reference trajectories for the flight controller [18]. In other words, it add the time information to the path.

1.3 Trajectory generation for a quadrotor

1.3.1 Problem statement of trajectory generation

A trajectory generation problem can be presented as: with given initial and final sates, a trajectory leads a quadrotor to the final destination and is feasible with respect to dynamics constrains and control input constraints.

A *configuration* is the three position and three orientation coordinates. The *configuration space* or *C-space* is the set of all possible configurations of a vehicle. A *state* consists of the configuration and rates of the configuration change. Similarly, all the possible states are named after *state space*.

The whole configuration or state world could be divided into *free* space and *obstacle* space. Free space is where the vehicle can move without contacting obstacles. Obstacle space is the subset of points representing a collision between the vehicle and an obstacle [8]. UAV is represented by a point vehicle which is a basic assumption for trajectory planning to vastly simplifies the problem [8]. An example is given by the Fig.1.6.

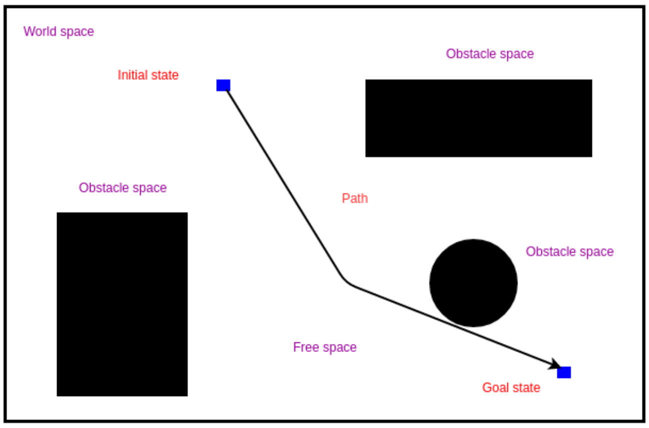


Fig. 1.6. Point vehicle representation

Fig.1.6 shows that a *path* is a curve that the vehicle is going to follow in the configuration space, while a *trajectory* refers to a combination of the path and the time information along the path.

Planing the UAV flight path or trajectory is the chief problem in autonomous UAV deployment [28]. The main challenges can be summarised as[22, 21, 7, 28, 8]

- Computation complexity caused by mathematical optimization methods which are computationally too demanding.
- Dependency between time and the state space introduced by the differential constraints
- UAV have differential constrains like limited speed and maximum acceleration.

- Uncertainty in vehicle dynamics and limited precision in command tracking.
- Uncertainty in the knowledge of the environment (e.g., obstacle locations).
- Uncertainty in pose information.
- Disturbances in the operational environment (e.g., wind, atmospheric turbulence).

1.3.2 Sate of art of trajectory generation

Trajectory generation for a drone has been a research interest for many years. However, due to difficulties and challenges mentioned before, the exact solution is generally impossible to find. Nearly all the algorithms designed to solve this problem are approximative [8].

In 2015, Hehn, Markus and D'Andrea, Raffaello [14] classify three main categorisations in trajectory generation algorithms for quadrotors:

1. A decoupling geometric and temporal planning approach.
   (a) first, a geometric trajectory without time information is constructed. For example, those path primitives can be constructed with lines in [16], polynomials in [6] or splines in [3].
   (b) then, parametrising the generated path in time such that the dynamic constraints of the quadrocopter are enforced.
2. An another approach is minimisation derivatives of the position trajectory (MDPT). These derivatives are linked to control input constraints and feasibility of the trajectory.
3. A numerically optimal control (OP) approach that directly considers the nonlinear dynamics of the drone. Optimal solutions are found through optimal control methods.

1.3.3 Minimisation derivatives of the position trajectory

MDPT considers the dynamics constrains and control inputs by optimising the derivatives of the position trajectory. Those derivatives are associated with the control and dynamics.

In 2012, Daniel Mellinger and Vijay Kumar [25] proposed and tested a real-time method minimising derivatives of the position trajectory to generate a sequence of 3D positions and yaw angles.

At first, a keyframe $\boldsymbol{\sigma}$ is defined as a combination of the position $\mathbf{x}_z$ and the yaw angle ψ_z in the space.

$$\boldsymbol{\sigma} = [\mathbf{x}_z, \psi_z] \tag{1.1}$$

where $\mathbf{x}_z = \begin{bmatrix} x_z & y_z & z_z \end{bmatrix}^T$ and x_z, y_z, z_z are translational coordinate of the drone.

The quadrotor is expected to pass some keyframes at certain time. Fig.1.7 shows an example in which the quadrotor should begin from keyframe 0 and pass by keyframe 1, keyframe 2 and then till keyframe n.

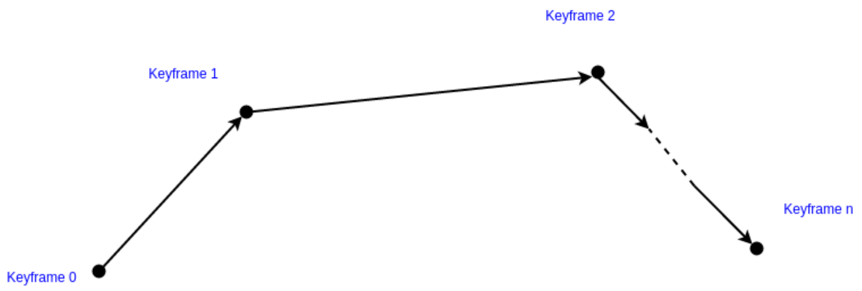


Fig. 1.7. Key frames in the space

The decision variables are the keyframe sequences: $\begin{bmatrix} \boldsymbol{\sigma_1} & \boldsymbol{\sigma_2} \ldots \boldsymbol{\sigma}_{n-1} \end{bmatrix}$.

Then, a m-order polynomial function is used to generate a trajectory between each two keyframes. The choice of m depends on the mission. For instance, if we want to specify the the position, velocity and acceleration at the initial and target point, the order of polynomial functions should be 5.

Then the whole trajectory is represented as

$$\boldsymbol{\sigma}(t) = \begin{cases} \sum_{i=0}^{m} \boldsymbol{\sigma}_{i1} t^i & t_0 \leq t \leq t_1 \\ \sum_{i=0}^{m} \boldsymbol{\sigma}_{i2} t^i & t_1 \leq t \leq t_2 \\ : & : \\ \sum_{i=0}^{m} \boldsymbol{\sigma}_{in} t^i & t_{n-1} \leq t \leq t_n \end{cases} \tag{1.2}$$

where $\boldsymbol{\sigma}_{i1} t^i (\sigma_{i2} t^i ...)$ are parameters in the polynomial function that define a trajectory between keyframe σ_0 and σ_1.

Then, $\boldsymbol{\sigma}(t)$ acts as a basis function for the following optimisation. In the model of Daniel Mellinger and Vijay Kumar, control inputs u_2 and u_3 are functions of the 4th derivatives of the positions and u_4 is related to the 2nd derivative of the yaw angle. Thus, the objective function is designed to minimise the integral of the square of the snap.

$$min \int_{t_0}^{t_f} u_{\mathbf{x}} \parallel \frac{d^4 \mathbf{x}_z}{dt^4} \parallel^2 + u_\psi \parallel \frac{d^2 \psi_z}{dt^2} \parallel^2 dt \tag{1.3}$$

where $u_{\mathbf{x}}, u_\psi$ are constants that make the integrand nondimensional.

Each keyframes should be passed by the quadrotor at a specified time and the corresponding constrain is formulated as

$$\boldsymbol{\sigma}(t_i) = \boldsymbol{\sigma}_i, \qquad i = 0, ..., m \tag{1.4}$$

Sometimes, the quadrotor might be asked to stop or meet a special conditions at some of keyframes. Other constrains about elements of $\boldsymbol{\sigma}(t)$ x_z, y_z, z_z, ψ_z are defined depending on the mission.

$$\frac{d^j x_z}{dt^j}\,|_{t=t_i} = 0 \quad \textit{or free}, \qquad i = 0, ..., m; j = 1, 2, 3, 4 \tag{1.5}$$

$$\frac{d^j y_z}{dt^j}\,|_{t=t_i} = 0 \quad \textit{or free}, \qquad i = 0, ..., m; j = 1, 2, 3, 4 \tag{1.6}$$

$$\frac{d^j z_z}{dt^j}\,|_{t=t_i} = 0 \quad \textit{or free}, \qquad i = 0, ..., m; j = 1, 2, 3, 4 \tag{1.7}$$

$$\frac{d^j \psi_z}{dt^j}\,|_{t=t_i} = 0 \quad \textit{or free}, \qquad i = 0, ..., m; j = 1, 2 \tag{1.8}$$

Also, some extensions like corridor constraints and optimal segment times were done in their work. Real time experiments were conducted to test this method in [25].

Then the developments of MDPT are introduced in the Table 1.1.

Table 1.1. **Sate of art of MDPT**

Time	Researchers	Validation
2012	Soonkyum Kim et al.[19] considered the trajectory generation as a Mixed-Integer Quadratic Program (MIQP) respecting kinematic constraints, avoiding obstacles, and constraining the trajectory to a particular homology class.	Numerical Simulation
2012	Daniel Mellinger et al.[26] used MIQP approach to generate trajectories for four quadrotors.	Real time experiments
2013	Charles Richter et al.[31]. who were inspired by the work in [25] developed a objective function minimising a weighted sum of derivatives	Real time experiments
2015	Sarah Tang and Vijay Kumar[34] applied MIQP trajectory generation for a quadrotor with a cable-suspended payload	Numerical Simulation and Real time experiments

1.3.4 Optimal control approach

Markus Hehn and Raffaello D'andrea developed a dynamic algorithm for trajectory generation that directly incorporates the dynamics constrains at the planning stage and is fast enough in real-time planning [13] in 2011.

In their work, a trajectory generator computes a trajectory from a given state x_{z0}, y_{z0}, z_{z0} to a given target state in order to reach in the minimal time. It is defined as $x_z(t), y_z(t), z_z(t)$. Then control inputs are calculated from this trajectory generated. The control inputs are set as the rotational rates about body axes $\boldsymbol{\gamma} = \begin{bmatrix} p & q & r \end{bmatrix}^T$ and the mass-normalised collective thrust F_a.

Constrains of thrust and rotational rates are analysed first. The trajectory generation can be done in each translational coordinate after decoupling. Jerk of three translational DOF $\dddot{\mathbf{x}}_z$ are planning inputs considering the acceleration and control inputs constrains. Then, trajectory generated is checked and replanned with reduced jerk constrains if it is infeasible.

The whole process is shown in the Fig.1.8.

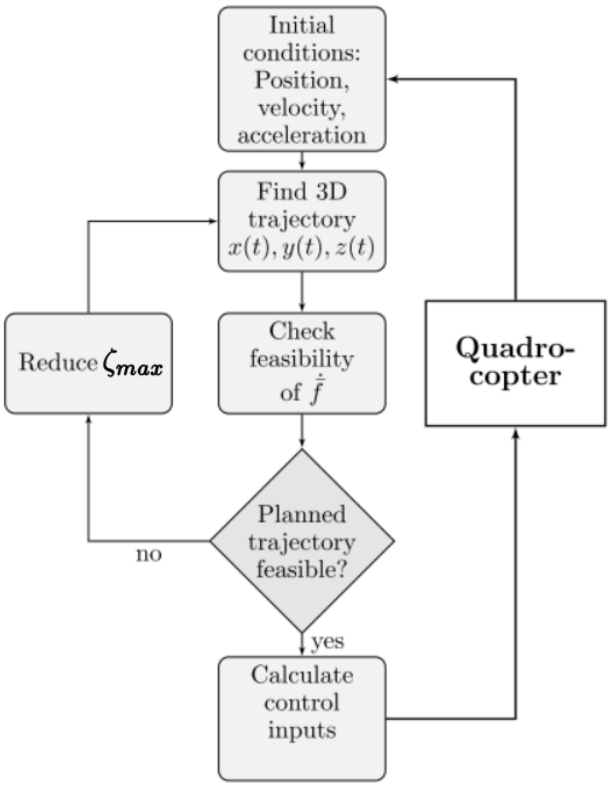

Initial conditions: Position, velocity, acceleration
Find 3D trajectory $x(t), y(t), z(t)$
Reduce ζ_{max}
Check feasibility of $\dot{f}$
Quadro-copter
Planned trajectory feasible?
no
yes
Calculate control inputs

Fig. 1.8. Closed-loop control using the trajectory planning [13]

Feasibility Conditions

Feasibility constrains for trajectory include two parts: collective thrust and rotational rates.

In terms of collective thrust, a vector $\mathbf{f}_a$ is introduced to represent the total mass-normalised force required to track the trajectory.

$$\mathbf{f}_a := \begin{bmatrix} \ddot{x}_z \\ \ddot{y}_z \\ \ddot{z}_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = {}^f\mathbf{R}_b \begin{bmatrix} 0 \\ 0 \\ F_a \end{bmatrix} \tag{1.9}$$

The direction of thrust can be described as

$$\bar{\mathbf{f}}_a = \frac{\mathbf{f}_a}{\parallel \mathbf{f}_a \parallel} \tag{1.10}$$

The limits for thrust force are assumed to be $[F_{min}, F_{max}]$. The constrains of the thrust force can be represented by $\mathbf{f}_a$ in the following way

$$\begin{gathered} F_{min} \leqslant \parallel \mathbf{f}_a \parallel \leqslant F_{max} \\ F_{min} \leqslant \sqrt{(\ddot{x}_z)^2 + (\ddot{y}_z)^2 + (\ddot{z}_z + g)^2} \leqslant F_{max} \end{gathered} \tag{1.11}$$

The control inputs p, q have to be bounded with the unit norm propoerty of the rotation matrix[1, 13].

$$\begin{gathered} p \leqslant \parallel \dot{\bar{\mathbf{f}}}_a \parallel \\ q \leqslant \parallel \dot{\bar{\mathbf{f}}}_a \parallel \end{gathered} \tag{1.12}$$

Constrains on r can be user-defined, for instance, $r = 0$.

Decoupling

In order to simplify the problem, three DOF are decoupled into x_z, y_z, z_z independently in trajectory generation.

As a consequence of decomposition, collective thrust and rotational rate constrains for the whole system have to be transformed for each translational coordinate.

For collective thrust constrains, limits user-defined $(\ddot{x}_{z_{max}}, \ddot{y}_{z_{max}}, \ddot{z}_{z_{max}})$ are set for each coordinate separately.

$$|\ddot{x}_z| \leqslant \ddot{x}_{z_{max}}, |\ddot{y}_z| \leqslant \ddot{y}_{z_{max}}, |\ddot{z}_z| \leqslant \ddot{z}_{z_{max}} \tag{1.13}$$

The values of $(\ddot{x}_{z_{max}}, \ddot{y}_{z_{max}}, \ddot{z}_{z_{max}})$ should meet conditions expressed in Eq.1.11.

$$\begin{gathered} \sqrt{(\ddot{x}_{z_{max}})^2 + (\ddot{y}_{z_{max}})^2 + (\ddot{z}_{z_{max}} + g)^2} \leqslant F_{max} \\ g - \ddot{z}_{z_{max}} \geqslant F_{min} \end{gathered} \tag{1.14}$$

The transformation of rotational rate constrains to each coordinate is complicated. Planning inputs are $\dddot{\mathbf{x}}_z$, the constrains on p, q are transformed to constrains for the $\dddot{x}_z$, $\dddot{y}_z$, $\dddot{z}_z$. Authors show that even it is possible to find relations between $\dddot{\mathbf{x}}_z$ and $\mathbf{f}_a$, it is difficult to get allowable magnitudes for elements due to the complexity.

Consequently, the constrains, for instance $\dddot{x}_{z_{max}}$, are defined by users and will be iterated for obtaining a feasible trajectory in the checking feasibility part.

Trajectory generation

An example of x_z trajectory generation is presented with the target being the origin. Let $\boldsymbol{\varrho} = (\varrho_1, \varrho_2, \varrho_3) = (x_z, \dot{x}_z, \ddot{x}_z)$. The decision variable is ζ the third derivative of the x_z. The purpose is to minimise the reaching time t_f. So, the time-optimal problem can be rewritten as:

$$\zeta^{\diamondsuit} = argmin \quad t_{fx} \tag{1.15}$$

subject to states dynamics

$$\begin{aligned} \dot{\varrho}_1 &= \varrho_2 \\ \dot{\varrho}_2 &= \varrho_3 \\ \dot{\varrho}_3 &= \zeta \end{aligned} \tag{1.16}$$

The initial and target state are

$$\begin{aligned} \varrho(t=0) &= x_{z0} \\ \dot{\varrho}(t=0) &= \dot{x}_{z0} \\ \ddot{\varrho}(t=0) &= \ddot{x}_{z0} \\ \varrho(t=t_{fx}) &= 0 \end{aligned} \tag{1.17}$$

The control input corresponds to the constrains on rotational rates.

$$|\zeta| \leqslant \zeta_{max} \tag{1.18}$$

where $\zeta_{max} = \dddot{x}_{z_{max}}$. That might be modified in the checking feasibility steps.

Acceleration constrains from Eq.1.13 will appear as a state constrain.

$$|\varrho_3| \leqslant \ddot{x}_{z_{max}} \tag{1.19}$$

With some mathematics in applied optimistic in [2, 11], the author claims that the solution $\zeta^{\diamondsuit}$ has a form shown in Fig.1.9. The corresponding $\ddot{x}, x$ can be integrated by Eq.1.16.

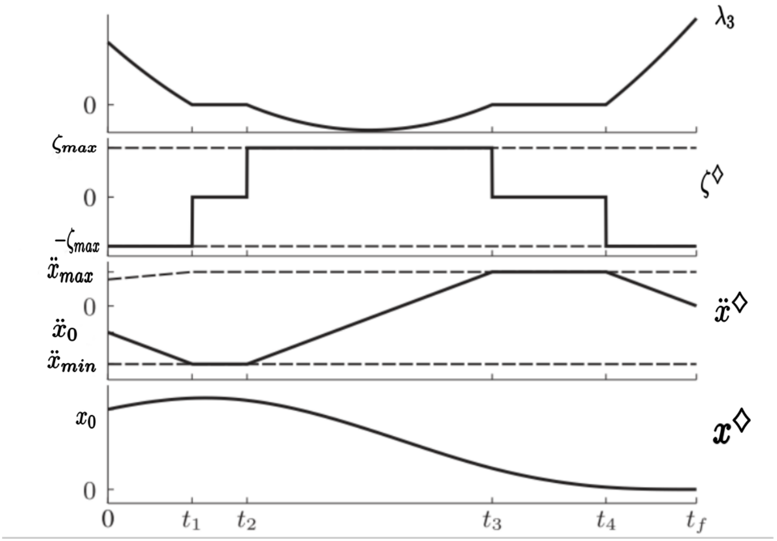


Fig. 1.9. Optimal trajectory including $\lambda_3, \zeta^{\diamondsuit}, \ddot{x}^{\diamondsuit}, x^{\diamondsuit}$[14]

The whole process consists of five sections:

1. $t \in [0, t_1]$, $\zeta^{\diamondsuit} = -\zeta_{max}$,

2. $t \in [t_1, t_2]$, $\zeta^{\diamondsuit} = 0$,
3. $t \in [t_2, t_3]$, $\zeta^{\diamondsuit} = \zeta_{max}$,
4. $t \in [t_3, t_4]$, $\zeta^{\diamondsuit} = 0$,
5. $t \in [t_4, t_{fx}]$, $\zeta^{\diamondsuit} = -\zeta_{max}$

Obviously, $\zeta^{\diamondsuit}$ is bang-singular and fully defined by its switching times $t_1, ..., t_{fx}$. It is fully specified by the five times $t_1, t_2, t_3, t_4, t_{fx}$ and the initial control input. The decision variables change to $t_1, t_2, t_3, t_4, t_{fx}$ instead of ζ.

$$\{t_1, t_2, t_3, t_4, t_{fx}\}^{\diamondsuit} = argmin \quad t_{fx} \tag{1.20}$$

A bisection algorithm is used to compute t_f based on which we can directly calculate t_1, t_2, t_3, t_4. Then the whole trajectory is obtained.

Applying this method to y_z, z_z coordinates to obtain t_{fy}, t_{fz}. The quadrotor ends at rest at the origin at time $t_f = max(t_{fx}, t_{fy}, t_{fz})$.

Check feasibility

The trajectory generated would be checked whether it meets the collective force and rotational rates constrains in Eq.1.11,1.12. It can act as a reference trajectory for the controller if it satisfies the constrains. Otherwise, we need to re-generate the trajectory with smaller values of ζ_{max}.

This method was tested at an indoor aerial vehicle development platform at ETH Zurich. The 3D result is shown Fig.1.10.

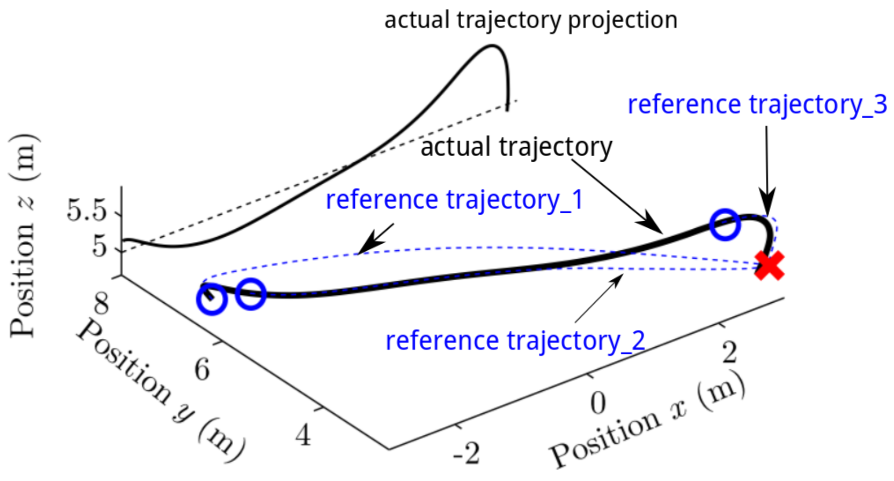


Fig. 1.10. 3D flight trajectory during experiment

In the Fig.1.10, the target point is marked by a red cross. The beginning state includes nonzero velocity and acceleration. At each blue circle, the quadrotor generates a reference trajectory which is a dotted blue line. The actual trajectory is the bold black line.

The quadrotor does not follow the trajectory_1 at the beginning. Then a trajectory_2 is computed and the quadrotor follows this trajectory quite well till nearly the end. A large error along axis z occurs at the end of the trajectory_2 and a trajectory_3 is computed, which is thought to be caused by strong aerodynamics effect. Finally the quadrotor reaches the target by following the trajectory_3.

Remarks

This method incorporates the thrust force and rotational rates in the optimal form. The decision variable are the jerks of translation coordinates and they are represented by 5 time variables. Experiments show that this method can be conducted in real time.

State of art

Table 1.2 shows the state of art of optimal approach for trajectory generation for a quadrotor.

Table 1.2. **Sate of art of optimal control approach**

Time	Researchers	Validation
2012	Markus Hehn and Raffaello D'Andrea[12] improved the method presented by crossing a specified position at a specified time.	Real-time experiment
2013	Mark W. Mueller et al.[27] introduced a cost function built with jerk of the quadrotor into the optimal framework in[13].	Real-time experiment
2015	Markus Hehn and Raffaello D'Andrea[14] proposed an iterative method for optimally choosing the decoupling parameters for the method in[13].	Real-time experiment

This method incorporates thrust force and rotational constrains in the optimal problem formulation directly. But there is not an efficient way for tuning parameters such as $\ddot{x}_{max}$.

1.4 Type 2 singularity of parallel robots

Singularity and crossing singularity have been the research interest for decades in terms of parallel robots. A 5 bar mechanism is a typical parallel robot shown in the Fig.1.11 whose actuated joints are q_{11} and q'_{21}.

Dynamic model of this parallel robot 5 bar mechanism is presented as

$$\boldsymbol{\tau}_e = \boldsymbol{\tau}_{ta} + \mathbf{J}_p^T \mathbf{w}_p + \mathbf{J}_d^T \boldsymbol{\tau}_{td} \tag{1.21}$$

where $\boldsymbol{\tau}_e$ is the vector of the real robot input efforts.

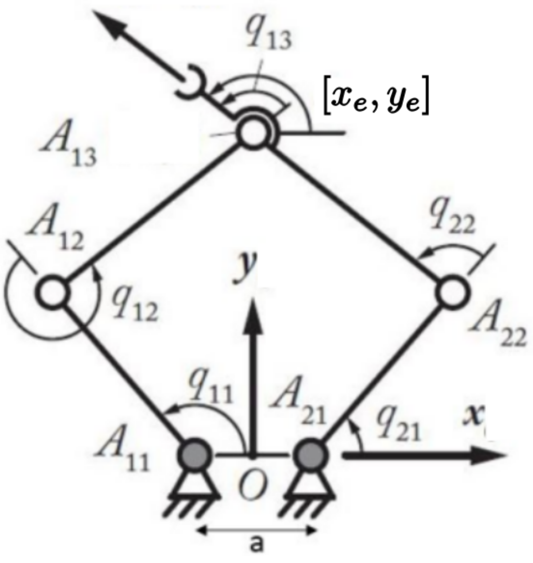


Fig. 1.11. A 5 bar mechanism

$\boldsymbol{\tau}_{ta}$ corresponds to the virtual input efforts in the actuated joints of the parallel robot related to the Lagrange L_t of the virtual tree structure.

$$\boldsymbol{\tau}_{ta} = \frac{d}{dt}\left(\frac{\partial L_t}{\partial \dot{\mathbf{q}}_a}\right)^T - \left(\frac{\partial L_t}{\partial \mathbf{q}_a}\right)^T \tag{1.22}$$

where $\mathbf{q}_a$ represents the actuated joints and $\mathbf{q}_a = \begin{bmatrix} q_{11} & q_{21}' \end{bmatrix}^T$

$\mathbf{w}_p$ corresponds to the wrench of the free platform expressed in the base frame and related with the Lagrange L_p of the moving platform

$$\mathbf{w}_p = \frac{d}{dt}\left(\frac{\partial L_p}{\partial \dot{\mathbf{x}}_e}\right)^T - \left(\frac{\partial L_p}{\partial \mathbf{x}_e}\right)^T \tag{1.23}$$

where $\mathbf{x}_e = \begin{bmatrix} x_e & y_e \end{bmatrix}$ is the coordinate of the end-effector.

$\boldsymbol{\tau}_{td}$ is a vector, which corresponds to the virtual input efforts in the passive joints of the parallel robot related to the Lagrange L_t of the virtual tree structure, and can be computed as:

$$\boldsymbol{\tau}_{td} = \frac{d}{dt}\left(\frac{\partial L_t}{\partial \dot{\mathbf{q}}_d}\right)^T - \left(\frac{\partial L_t}{\partial \mathbf{q}_d}\right)^T \tag{1.24}$$

where $\mathbf{q}_d = \begin{bmatrix} q_{12} & q_{22} & q_{13} \end{bmatrix}$ and q_{12}, q_{22}, q_{13} are passive joints.

$\mathbf{J}_p = \mathbf{A}_r^{-1}\mathbf{B}$ is the Jacobian relating the platform twist $\mathbf{t}_p$ and the active joint velocities $\dot{\mathbf{q}}_a$, which is obtained from

$$\mathbf{A}_r\mathbf{t}_p + \mathbf{B}\dot{\mathbf{q}}_a = 0 \tag{1.25}$$

$\mathbf{J}_d$ allows to express the passive joint velocities $\dot{\mathbf{q}}_d$ as a function of the active joint velocities $\dot{\mathbf{q}}_a$. More details could be found in [5].

Singularities in parallel robots can be classified into following types of singularities according to the kinematic models, which is proposed by Grosselin and Angeles[10].

1. When matrix $\mathbf{B}$ is rank-deficient, such kind of singularity is called Type 1 singularities, in which case the robot loses the ability to move in one given direction. In other words, there is a direction in which no task space velocity can be generated in such configuration. Motion of the actuators does not lead to the displacement of the robot platform

2. When matrix $\mathbf{A}_r$ is rank-deficient, it means Type 2 singularities. It refers to the the fact the parallel mechanism loses its ability to change from assembly mode. The robot gains one (or more) uncontrollable motion.

   One Type 2 singularity of the 5 bar mechanism is given in the Fig.1.11. In such configuration, the parallel robot gains an uncontrollable motion perpendicular to $\overrightarrow{A_{12}A_{13}}$, $\overrightarrow{A_{22}A_{13}}$ (under-actuation in the system).

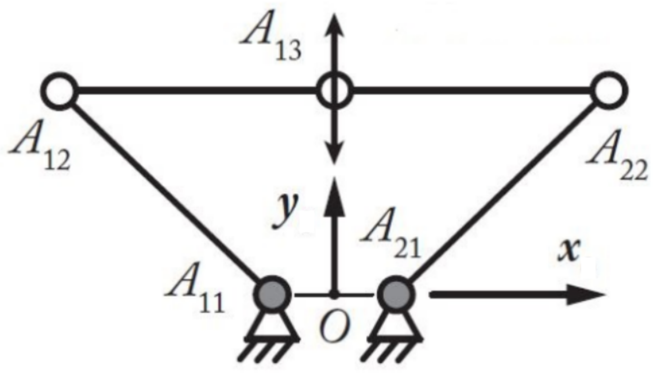


Fig. 1.12. One Type 2 singularity configuration of a 5 bar mechanism

3. When both Type 1 and Type 2 singular postures appear together ($\mathbf{A}_r, \mathbf{B}$ are rank-deficient), it means Type 3 singularity. In such configurations, the robot loses locally the ability to perform a motion along one direction of the workspace and also gains one or more uncontrollable motions along another direction.

4. Other singularities such as LPJTS singularities and its introduction can be found in [5]

One approach is to develop a physical criterion deduced from the degeneracy conditions of the dynamic model [4]. The configuration at the singularity will be specified off-line.

Since the matrix $\mathbf{A}_r$ is rank-deficient, we can find a non-null vector $\mathbf{t}_s$ in the kernel of $\mathbf{A}_r$.

$$\mathbf{t}_s^T \mathbf{w}_p = 0 \tag{1.26}$$

That is the condition to cross the type 2 singularity. It means that the wrench applied on the platform by the legs and external forces $\mathbf{w}_p$ must be reciprocal to the twist $\mathbf{t}_s$ of uncontrollable motion in the singularity locus.

A multi-model computed torque control for tracking optimal trajectories that respect the physical criterion is proposed in [30]. Damien SIX et al implement a controller in Cartesian space to track a trajectory that crosses a Type 2 singularity in [32] After that, Rafael Balderas Hill et al. [15] propose a controller integrated in a multi-control architecture in order to switch between a classical computed torque control far from the singularity and the virtual-constraint-based control law near to the singularity locus. This method is tested in real experiments.

1.5 A Flying parallel robot

Damien Six, Abdelhamid Chriette, Sebastien Briot and Philippe Martinet [33] explore a new flying structure in the Fig.1.13. It is built from two quadrotors linked by a passive kinematic chain. From the perspective of the whole robot, it is a parallel robot where actuators are replaced by two drones. In other words, the end-effector is driven by the two quadrotors which are under-actuated.

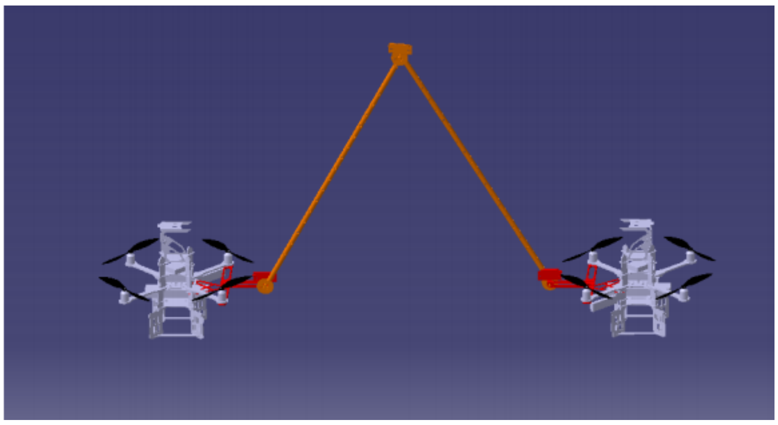


Fig. 1.13. Flying parallel robot[33]

FPR has several advantages compared to the existing flying platforms [33]:

- It can deal with tasks where the end-effector is under or above the quadrotors by changing the configuration of the system.
- The whole efforts are spread over two quadrotors such that FPR's total payload is enhanced.
- The absence of additional embedded motors to actuate the effector reduces the load of the system itself and maintain the energetic autonomy of the drones.
- Choices of leg topology [9] leads to a variety of physical properties of potential interest.

As a combination of quadrotors and a passive chain, FPR's dynamics is more complicated and difficult to analyse than ordinary quadrotors. That is, the under-actuation and dynamic limits, such as thrust limits, of quadrotors cause a difficulty for designing controllers for FPR.

The motion optimisation for the robot needs to consider, under-actuation of quadrotors, the dynamics constrains and two quadrotors payload limitations.

1.6 Master thesis contents

In terms of FPR, the modelling part and a controller system have already been done in [33].

The main purposes and contributions of the thesis are

- to propose an optimal motion planning with the respect of dynamic constraints linked to the drones payload limitations for the RPR.
- to design a trajectory crossing Type 2 singularity of FPR.

This report is organised as follows:

1. Chapter 2 recalls the dynamic modelling process of FPR and introduce the controller designed in [33].
2. Chapter 3 addresses the main contribution of this master thesis optimal motion planning for FPR. Optimisation results are validated in simulation/Matlab. This method is also tested with a FPR prototype robot.
3. Chapter 4 analyses Type 2 singularity of FPR and the approaches to crossing Type 2 singularity for the FPR prototype.

Chapter 2

Dynamic model and control system of FPR

This Chapter illustrates main steps for developing the dynamic model for FPR and the CTC controller in [33]. Two parts of the whole dynamic model are demonstrated: passive chain dynamic model and attitude dynamic model. Based on that, one CTC controller is introduced.

2.1 Dynamic model of FPR

The robot is designed to perform motions restricted to a vertical plane in order to simplify and reduce the study complexity. The chain motion is constrained by the three revolute joints to a planar motion and FRP becomes a planar parallel mechanism. Two passive joints connect the chain and drones which are named after drone 1 and drone 2. Fig.2.1 shows the frames and the parametrisation.

The world frame is defined as $\mathcal{F}$ with axis $\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}$. The local frame attached to the end-effector is named after $\mathcal{F}'$. Two local frames $\mathcal{D}_1, \mathcal{D}_2$ are set at the centers of quadrotor 1 and quadrotor 2.

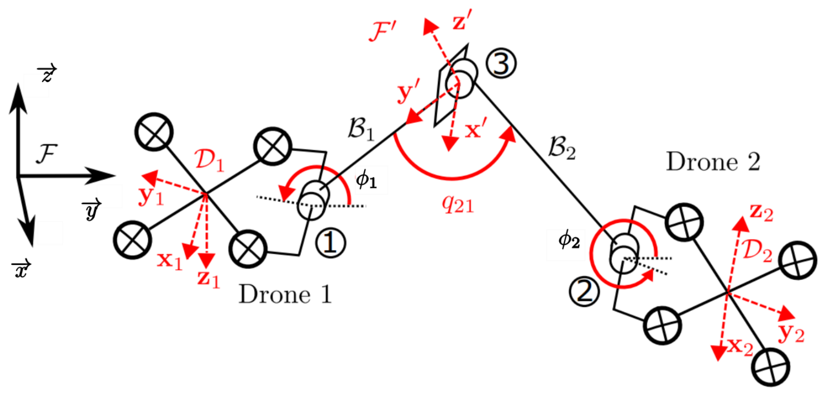


Fig. 2.1. Parametrisation of the FPR[33]

The coordinate vector $\mathbf{q} = \begin{bmatrix} x & y & z & \psi & \theta & \phi & q_{21} & \phi_1 & \phi_2 \end{bmatrix}^T$ where

- $\begin{bmatrix} x & y & z \end{bmatrix}$ means the origin of the local frame $\mathcal{F}'$ expressed in world frame $\mathcal{F}$
- $\begin{bmatrix} \psi & \theta & \phi \end{bmatrix}$ represent orientation of the local frame (Yaw/Pitch/Roll angles) in the world frame.
- q_{21} is the relative angle between two links.
- $\begin{bmatrix} \phi_1 & \phi_2 \end{bmatrix}$ are the roll angles of respectively drone 1 and drone 2 in the world frame.

The inputs of the FPR are from two quadrotors and each quadrotors provides one thrust force and three torques. They can be represented by two actuation wrenches:

$$\mathbf{w}_1 = \begin{bmatrix} 0 & 0 & 0 & f_{1z} & \tau_{1x} & \tau_{1y} & \tau_{1z} \end{bmatrix} \tag{2.1}$$

$$\mathbf{w}_2 = \begin{bmatrix} 0 & 0 & 0 & f_{2z} & \tau_{2x} & \tau_{2y} & \tau_{2z} \end{bmatrix} \tag{2.2}$$

where f_{1z}, f_{2z} are two thrust forces and $\tau_{1x}, \tau_{1y}, \tau_{1z}, \tau_{2x}, \tau_{2y}, \tau_{2z}$ are six torques defined in the local frames of quadrotors $\mathcal{D}_1, \mathcal{D}_2$. We define

$$\mathbf{f} = \begin{bmatrix} f_{1z} & f_{2z} \end{bmatrix}^T \tag{2.3}$$

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{1x} & \tau_{1y} & \tau_{1z} & \tau_{2x} & \tau_{2y} & \tau_{2z} \end{bmatrix}^T \tag{2.4}$$

$$\tag{2.5}$$

The dynamic model of FPR is decoupled into two parts: passive chain dynamic model and attitude dynamic model.

Passive chain dynamic model

$\mathbf{q}_p = \begin{bmatrix} y & z & \phi & q_{21} \end{bmatrix}$ parametrises the configuration of the passive chain.

$\mathbf{f}_{p1}$ and $\mathbf{f}_{p2}$ are the forces applied to the chain by the quadrotors. For instance, $\mathbf{f}_{p1}$ is shown in the Fig.2.2

The relationship among $\ddot{\mathbf{q}}_p$, $\mathbf{f}_{p1}$ and $\mathbf{f}_{p2}$ is obtained by applying the Euler-Lagrange equations to the passive chains.

$$\mathbf{M}_p \ddot{\mathbf{q}}_p + \mathbf{c}_p = \mathbf{J}_p^T \begin{bmatrix} \mathbf{f}_{p1} \\ \mathbf{f}_{p2} \end{bmatrix} \tag{2.6}$$

$\mathbf{M}_p$ is the definite positive generalised inertial matrix (4 by 4) of the planar passive chain depending on $\mathbf{q}_p$; $\mathbf{c}_p$ refers to a 4 dimensional vector of Coriolis and centrifugal effects depending on $\mathbf{q}_p$ and $\dot{\mathbf{q}}_p$.

Then, links among the forces applied by quadrotors, input thrust forces and roll angles are developed. This is done through expressing the translational dynamics of the quadrotors.

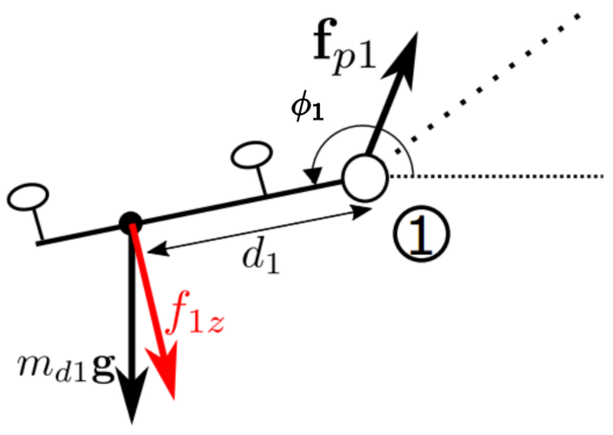


Fig. 2.2. Forces acting on quadrotor 1[33]

For quadrotor i(i=1.2), we have

$$\begin{bmatrix} 0 \\ f_{iz} \end{bmatrix} = \mathbf{R}_i[\mathbf{f}_{pi} + m_{di}(\ddot{\mathbf{r}}_i - \mathbf{g})] + \mathbf{p}_i \tag{2.7}$$

$$\mathbf{p}_i = \begin{bmatrix} -m_{di}d_i\dot{\phi}_i^2 \\ m_{di}d_i\ddot{\phi}_i \end{bmatrix} \tag{2.8}$$

where

- $\mathbf{R}_i = \begin{bmatrix} cos(\phi_i) & sin(\phi_i) \\ -sin(\phi_i) & cos(\phi_i) \end{bmatrix}$ is the rotation matrix about ϕ_i in the vertical plane,
- $\mathbf{r}_i$ positions of joint i in the vertical plane in the world frame,
- m_{di} represents the mass of the quadrotor i,
- $\mathbf{g}$ is acceleration of gravity,
- d_i means the distance between the joint i and the CoM of quadrotor i.

Combing Eq.2.6 and Eq.2.7 results in

$$\mathbf{R}_{inv}(\Psi\mathbf{f} - \mathbf{p}) = \mathbf{M}_t\ddot{q}_p + \mathbf{c}_t \tag{2.9}$$

with $\mathbf{R}_{inv} = \begin{bmatrix} \mathbf{R}_1^{-1} & 0 \\ 0 & \mathbf{R}_2^{-1} \end{bmatrix}$, $\Psi = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$

The term $\mathbf{p}$ is considered as a perturbation of the passive chain dynamics, such that dynamic equations of passive kinematic chain depends only on the input thrust forces $\mathbf{f}$ and roll angles of two drones ϕ_1, ϕ_2.

Attitude dynamic model

The coordinates for attitude are $\mathbf{q}_a = \begin{bmatrix}\psi & \theta & \phi_1 & \phi_2\end{bmatrix}^T$.

Applying Euler-Lagrange equations to the whole FPR and selecting acceleration of attitudes results in

$$\ddot{\mathbf{q}}_a = \mathbf{M}_{inv,a}\left(\mathbf{J}^T \begin{bmatrix}\mathbf{w}_1 \\ \mathbf{w}_2\end{bmatrix} - \mathbf{c}\right) \tag{2.10}$$

in which

- $\mathbf{M}_{inv,a}$ is the rows of the inverse of the matrix $\mathbf{M}$ restricted to the attitude coordinates and $\mathbf{c}$ corresponds to the vector of Coriolis and centrifugal effects:

$$\ddot{\mathbf{q}} = \mathbf{M}^{(-1)}\left(\mathbf{J} \begin{bmatrix}\mathbf{w}_1 \\ \mathbf{w}_2\end{bmatrix} - \mathbf{c}\right) \tag{2.11}$$

- $\mathbf{J}$ is the Jacobian matrix from the twists of drone 1, 2 and coordinates velocities $\dot{q}$
- $\mathbf{w}_1, \mathbf{w}_2$ are input wrenches in quadrotors' local frames, defined in Eq.2.1 and Eq.2.2.

Then, actuation wrenches are expressed in terms of $\mathbf{f}$ and $\boldsymbol{\tau}$. Eq.2.10 can be represented as

$$\ddot{\mathbf{q}}_a = \mathbf{M}_{inv,a}\left(\mathbf{J}_\tau^T \boldsymbol{\tau} + \mathbf{J}_f^T \mathbf{f} - \mathbf{c}\right) \tag{2.12}$$

$$\ddot{\mathbf{q}}_a = \mathbf{A}_\tau \boldsymbol{\tau} + \mathbf{M}_{inv,a}\left(\mathbf{J}_f^T \mathbf{f} - \mathbf{c}\right) \tag{2.13}$$

2.2 Control methods for FPR

The controller scheme of the FPR is illustrated in the Fig.2.3.

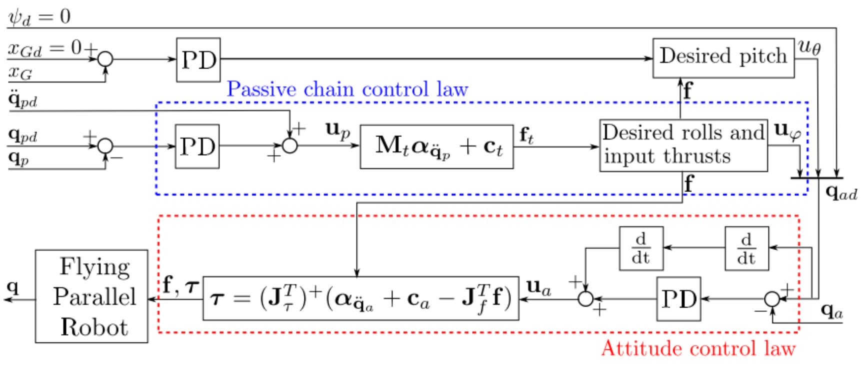

$\psi_d = 0$
$x_{Gd} = 0$
x_G
$\ddot{\mathbf{q}}_{pd}$
$\mathbf{q}_{pd}$
$\mathbf{q}_p$
PD
Desired pitch
u_θ
Passive chain control law
$\mathbf{f}$
PD
$\mathbf{u}_p$
$\mathbf{M}_t\boldsymbol{\alpha}_{\ddot{\mathbf{q}}_p} + \mathbf{c}_t$
$\mathbf{f}_t$
Desired rolls and input thrusts
$\mathbf{u}_\varphi$
$\mathbf{q}_{ad}$
$\mathbf{f}$
$\frac{\mathrm{d}}{\mathrm{dt}}$
$\frac{\mathrm{d}}{\mathrm{dt}}$
$\mathbf{q}$
Flying Parallel Robot
$\mathbf{f}, \boldsymbol{\tau}$
$\boldsymbol{\tau} = (\mathbf{J}_\tau^T)^+(\boldsymbol{\alpha}_{\ddot{\mathbf{q}}_a} + \mathbf{c}_a - \mathbf{J}_f^T\mathbf{f})$
$\mathbf{u}_a$
PD
$\mathbf{q}_a$
Attitude control law

Fig. 2.3. General scheme of the FPR controller [33]

The whole control system is a combination of two cascaded loops: one outer loop for passive chain and one inner loop for attitude. The outer loop generates desired roll angles $\mathbf{u}_\phi$ and input thrusts $\mathbf{f}$ for two drones. Also, The desired angles of θ and ψ are set to guarantee the vertical planar motion. A CTC controller is designed as

$$\mathbf{u}_p = \ddot{\mathbf{q}}_p^d - \mathbf{K}_{pp}(\mathbf{p}_p - \mathbf{p}_p^d) - \mathbf{K}_{dp}(\dot{\mathbf{p}}_p - \dot{\mathbf{p}}_p^d) \tag{2.14}$$

Submitting $\mathbf{u}_p$ and considering $\mathbf{p}$ as disturbances in Eq.2.9, we can compute the input force $\mathbf{f}$

$$\mathbf{f} = \begin{bmatrix} -sin(\phi_1) & cos(\phi_i) & 0 & 0 \\ 0 & 0 & -sin(\phi_2) & cos(\phi_2) \end{bmatrix} \begin{bmatrix} \mathbf{f}_{u1} \\ \mathbf{f}_{u2} \end{bmatrix} \tag{2.15}$$

$$\begin{bmatrix} \mathbf{f}_{u1} \\ \mathbf{f}_{u2} \end{bmatrix} = (\mathbf{M}_t \mathbf{u}_p + \mathbf{c}_t) \tag{2.16}$$

The closed-loop equation of the passive chain is obtained by applying the computed input forces Eq.2.15 to its dynamic model Eq.2.9

$$\mathbf{e}_p + \mathbf{K}_{pp}\mathbf{e}_p + \mathbf{K}_{dp}\dot{\mathbf{e}}_p + \mathbf{M}_t^{-1}(\boldsymbol{\delta} + \boldsymbol{\delta}_p) = 0 \tag{2.17}$$

where

$$\boldsymbol{\delta} = \mathbf{R}_{inv} \begin{bmatrix} f_{u1y}cos(\phi_1) + f_{u1z}sin(\phi_1) \\ 0 \\ f_{u2y}cos(\phi_2) + f_{u2z}sin(\phi_2) \\ 0 \end{bmatrix} \tag{2.18}$$

$$\boldsymbol{\delta}_p = \mathbf{R}_{inv}\mathbf{p} \tag{2.19}$$

The perturbation term $\boldsymbol{\delta}$ is caused by the drones under-actuation and f_{uiy}, f_{uiz} are the distributions of force $\mathbf{f}_{ui}$ along $\overrightarrow{y}$ and $\overrightarrow{z}$ directions. This perturbation term $\boldsymbol{\delta}$ can be cancelled by setting roll angles for drones and two auxiliary inputs are computed as

$$\mathbf{u}_\phi = \begin{bmatrix} u_{\phi 1} \\ u_{\phi 2} \end{bmatrix} = \begin{bmatrix} atan2(f_{u1y}, -f_{u1z}) \\ atan2(f_{u2y}, -f_{u2z}) \end{bmatrix} \tag{2.20}$$

One part of $\boldsymbol{\delta}_p$ in Eq.2.19 can be partly rejected by modifying the thrust inputs in Eq.2.15 as

$$\mathbf{f} = \begin{bmatrix} -sin(\phi_1) & cos(\phi_i) & 0 & 0 \\ 0 & 0 & -sin(\phi_2) & cos(\phi_2) \end{bmatrix} \begin{bmatrix} \mathbf{f}_{u1} \\ \mathbf{f}_{u2} \end{bmatrix} + \begin{bmatrix} m_{d1}d_1\ddot{\phi}_1 \\ m_{d2}d_2\ddot{\phi}_2 \end{bmatrix} \tag{2.21}$$

In this way, the perturbation term $\boldsymbol{\delta}_p$ finally becomes

$$\boldsymbol{\delta}_p = \mathbf{R}_{inv} \begin{bmatrix} -m_{d1}d_1\dot{\phi}_1^2 \\ 0 \\ -m_{d2}d_2\dot{\phi}_2^2 \\ 0 \end{bmatrix} \tag{2.22}$$

The inner loop operates much faster than the outer loop. The inner controller loop computes input torques, receiving desired angles $\mathbf{q}_a^d = [u_\theta, \mathbf{u}_\phi, \psi_d]$. The control law is

$$\mathbf{u}_a = \ddot{\mathbf{q}}_a^d - \mathbf{K}_{pa}(\mathbf{p}_a - \mathbf{p}_a^d) - \mathbf{K}_{da}(\dot{\mathbf{p}}_a - \dot{\mathbf{p}}_a^d) \tag{2.23}$$

We can obtain input torques $\boldsymbol{\tau}$ linking Eq.2.23 and Eq.2.13.

$$\boldsymbol{\tau} = \mathbf{A}_\tau^+(\mathbf{u}_a + \mathbf{M}_{inv,a}\mathbf{c} - \mathbf{M}_{inv,a}\mathbf{J}_f^T\mathbf{f}) \tag{2.24}$$

where $\mathbf{A}_\tau^+$ is the pseudo-inverse of $\mathbf{A}_\tau$.

Chapter 3

Optimisation motion planning for FPR

3.1 Motion generation for FPR

A controller system based on the CTC has been designed and tested with a square trajectory in the vertical plane in the paper [33]. However, an optimal trajectory for this new FPR robot is still necessary.

An optimal trajectory should take into account the following points:

1. The dynamics of FPR in which the actuators are two quadrotors. Each quadrotor is driven by four rotors and the thrust force has a linear relationship with the square of rotor angular velocities. Thus a feasible trajectory should consider the rotors dynamics. Also, the thrust force produced by one quadrotor is always along the $\mathbf{z}_i$ direction in its local frame. If a oblique input force is needed by the robot, quadrotors are supposed to have certain roll angles ϕ_i.
2. Perturbation $\boldsymbol{\delta}_p$ in Eq.2.22 in the controller which we discuss in section 2.2. The optimal trajectory should minimise terms that cause perturbations in the controller.
3. Under-actuation of quadrotors. The attitudes of the two drones are not known.
4. Energy consumption is usually involved in trajectory optimisation. Especially, many quadrotors have their on-board batteries but durations of quadrotors are limited by these batteries.

A vector to design the motion in the vertical plane is defined as

$$\mathbf{s} = \mathbf{q}_p = \begin{bmatrix} y & z & \phi & q_{21} \end{bmatrix} = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \end{bmatrix} \tag{3.1}$$

Basic assumptions

1. trajectory generation for each coordinate is independent for $\mathbf{s}$.

2. Robot does not change its configuration or cross singularity during the motion.
3. Navigation and control systems work in perfect ways. Or, the locations and orientation information of the robot are available without any error and controller tracks the reference signals perfectly without any time delay.

Trajectory generation with 9-degree polynomial

One drone is driven by four on-board rotors which are controlled by mic-controllers. The dynamic model of a drone is to be analysed to understand a feasible robot motion.

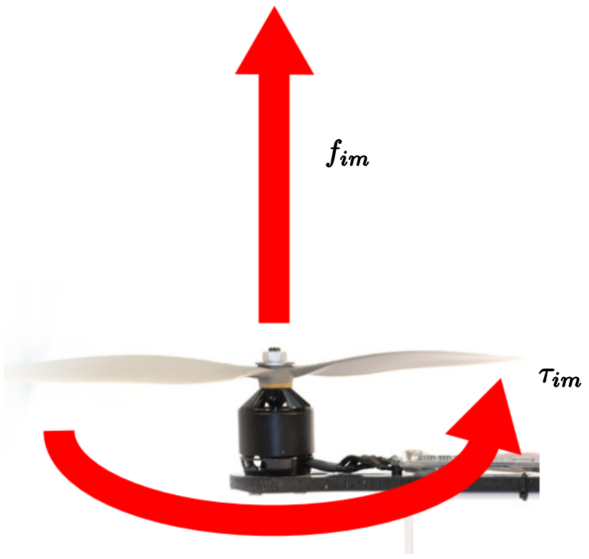


Fig. 3.1. The force and torque generated by one rotor

Fig.3.1 shows the force f_{im} and the torque τ_{im} produced by a rotor of the done i $(m = 1, 2, 3, 4)$. We assume that the rotor m rotates at the speed ϖ_{im}, the thrust f_{im} and torque τ_{im} generated are

$$f_{im} = c_T \varpi_{im}^2 \tag{3.2}$$
$$\tau_{im} = c_Q \varpi_{im}^2 \tag{3.3}$$

where c_T and c_Q are positive constants and we can obtain their values from static thrust tests.

So, the thrust force generated by a whole drone can be modelled as

$$f_{iz} = \sum_{m=1}^{4} f_{im} = c_T \sum_{m=1}^{4} (\varpi_{im}^2) \tag{3.4}$$

Second, the roll torque depends on the front and the rear rotors $(m = 2, 4)$, which is

$$\tau_{i_x} = f_{i2} \cdot d - f_{i4} \cdot d = dC_T(\varpi_{i2}^2 - \varpi_{i4}^2) \tag{3.5}$$

where d is the distance between the CoM of the drone i and one rotor.

Similarly, we can get the dynamic model for pitch torque $(m = 1, 3)$

$$\tau_{i_y} = f_{i1} \cdot d - f_{i3} \cdot d = dC_T(\varpi_{11}^2 - \varpi_{13}^2) \tag{3.6}$$

The dynamic model of yaw torque is different from the others. Yaw rotation is because of the reaction torque (caused by rotor drag). The yaw dynamic model is developed as

$$\tau_{i_z} = \tau_{i2} + \tau_{i4} - \tau_{i1} - \tau_{i4} = c_Q(\varpi_{i2}^2 + \varpi_{i4}^2 - \varpi_{i1}^2 - \varpi_{i3}^2) \tag{3.7}$$

Thus, the drag torque $\boldsymbol{\tau}$ is modelled as

$$\boldsymbol{\tau}_i = \begin{bmatrix} \tau_{i_x} & \tau_{i_y} & \tau_{i_z} \end{bmatrix}^T \tag{3.8}$$

The dynamic equation of one drone is

$$\begin{bmatrix} f_{iz} & \boldsymbol{\tau}_i \end{bmatrix}^T = \mathbf{R}_f \boldsymbol{\varpi}_i \tag{3.9}$$

in which

$$\mathbf{R}_f = \begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & dc_T & 0 & -dc_T \\ -dc_T & 0 & dc_T & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \tag{3.10}$$

$$\boldsymbol{\varpi}_i = \begin{bmatrix} \varpi_{i1}^2 & \varpi_{i2}^2 & \varpi_{i3}^2 & \varpi_{i4}^2 \end{bmatrix}^T \tag{3.11}$$

As each rotor is controlled by a mic-controller, it is necessary to guarantee that the ϖ_{im} is continuous. This can be done by design a trajectory with continuous $\begin{bmatrix} \mathbf{f} & \boldsymbol{\tau} \end{bmatrix}^T$.

With a planned motion, we can represent required input force of FPR from Eq.2.14, 2.21, 2.16 in a simplified way. :

$$\mathbf{f} = (\mathbf{R}_{inv}\boldsymbol{\Psi})^{(-1)} (\mathbf{M}_t \ddot{\mathbf{u}}_p + \mathbf{c}_t) + \begin{bmatrix} m_{d1} d_1 \ddot{\phi}_1 \\ m_{d2} d_2 \ddot{\phi}_2 \end{bmatrix} \tag{3.12}$$

$$\mathbf{f} = g_1(\ddot{\mathbf{q}}_p^d, \dot{\mathbf{q}}_p^d, \mathbf{q}_p^d) \tag{3.13}$$

$\ddot{\mathbf{q}}_p^d$ needs to be designed in order to have a continuous $\mathbf{f}$. Based on the $\mathbf{f}$, we recall the required roll angles of two drones in Eq.2.20 and represent it as

$$\mathbf{u}_\phi = g_2(\mathbf{f}) = g_3(\ddot{\mathbf{q}}_p^d, \dot{\mathbf{q}}_p^d, \mathbf{q}_p^d) \tag{3.14}$$

In the second controller loop of $\mathbf{u}_a$, u_θ are computed by PD controller with $x^d = 0$ and $\psi_d = 0$. Since $\mathbf{u}_a$ depends on the $\ddot{\mathbf{q}}_a^d, \dot{\mathbf{q}}_a^d, \mathbf{q}_a^d$ and $\mathbf{q}_a^d = [u_\theta, \mathbf{u}_\phi, \psi_d]$, we can represent $\mathbf{u}_a$ as

$$\mathbf{u}_a = g_4(\mathbf{q}_p^{d(4)}, \mathbf{q}_p^{d(3)}, \ddot{\mathbf{q}}_p^d, \dot{\mathbf{q}}_p^d, \mathbf{q}_p^d) \tag{3.15}$$

Then, we can obtain the expression for $\boldsymbol{\tau}$ in Eq.2.24 as following:

$$\boldsymbol{\tau} = g_5(\mathbf{u}_a, \mathbf{f}) \tag{3.16}$$

$$\boldsymbol{\tau} = g_6(\mathbf{q}_p^{d(4)}, \mathbf{q}_p^{d(3)}, \ddot{\mathbf{q}}_p^d, \dot{\mathbf{q}}_p^d, \mathbf{q}_p^d) \tag{3.17}$$

It is clear from Eq.3.17 that we must specify $\mathbf{q}_p^{d(4)}$, i.e. $\mathbf{s}^{(4)}$ in the motion planning. Consequently, a 9-degree polynomial is used to generate a trajectory. One trajectory for coordinate s_j ($j = 1, 2, 3, 4$) can be expressed as

$$s_j(t) = a_{j0} + a_{j1}t + a_{j2}t^2 + a_{j3}t^3 + a_{j4}t^4 + a_{j5}t^5 + a_{j6}t^6 + a_{j7}t^7 + a_{j8}t^8 + a_{j9}t^9 \tag{3.18}$$

where t refers to the time and $a_{j0}, ...a_{j9}$ are parameters. Computing those parameters requires the travelling time t_f, the initial and the final position and their 1st, 2nd, 3rd and 4th derivatives.

$$\begin{aligned} s_j(t=0) &= s_{jb} & s_j(t=t_f) &= s_{jf} \\ \dot{s}_j(t=0) &= \dot{s}_{jb} & \dot{s}_j(t=t_f) &= \dot{s}_{jf} \\ \ddot{s}_j(t=0) &= \ddot{s}_{jb} & \ddot{s}_j(t=t_f) &= \ddot{s}_{jf} \\ s_j^{(3)}(t=0) &= s_{jb}^{(3)} & s_j^{(3)}(t=t_f) &= s_{jf}^{(3)} \\ s_j^{(4)}(t=0) &= s_{jb}^{(4)} & s_j^{(4)}(t=t_f) &= s_{jf}^{(4)} \end{aligned} \tag{3.19}$$

where $s_{jb}, \dot{s}_{jb}, \ddot{s}_{jb}, s_{jb}^{(3)}, s_{jb}^{(4)}$ are position, velocity, acceleration, 3rd and 4th derivatives of the robot at the initial position. The information corresponding to the final destination are presented by $s_{jf}, \dot{s}_{jf}, \ddot{s}_{jf}, s_{jf}^{(3)}, s_{jf}^{(4)}$.

Motion planning problem statement

Usually, a robot is asked to pass some designed positions in the task space or in the joint space. Those designed positions by the mission are named after way position s_w.

The way positions of the trajectory in [33] are represented in the Fig.3.2. The mission can be described as

1. $s_0 \Rightarrow s_{w1}$
2. $s_{w1} \Rightarrow s_{w2}$
3. $s_{w2} \Rightarrow s_{w3}$
4. $s_{w3} \Rightarrow s_0$

where s_0 is the Beginning\Ending position $\begin{bmatrix} 0 & 0 & \frac{5\pi}{4} & \frac{\pi}{2} \end{bmatrix}$ and s_{wk} are way positions.

3.2 Motion planning optimisation for FPR

According to the previous section, the trajectory generation needs not only positions but also velocities, accelerations, 3rd and 4th derivatives.

We define

Position : s. s_w means way positions.

Point : $\mathbf{S} = \begin{bmatrix} \mathbf{s} & \dot{\mathbf{s}} & \ddot{\mathbf{s}} & \mathbf{s}^{(3)} & \mathbf{s}^{(4)} \end{bmatrix}$. $\mathbf{S}_w$ refers to way points.

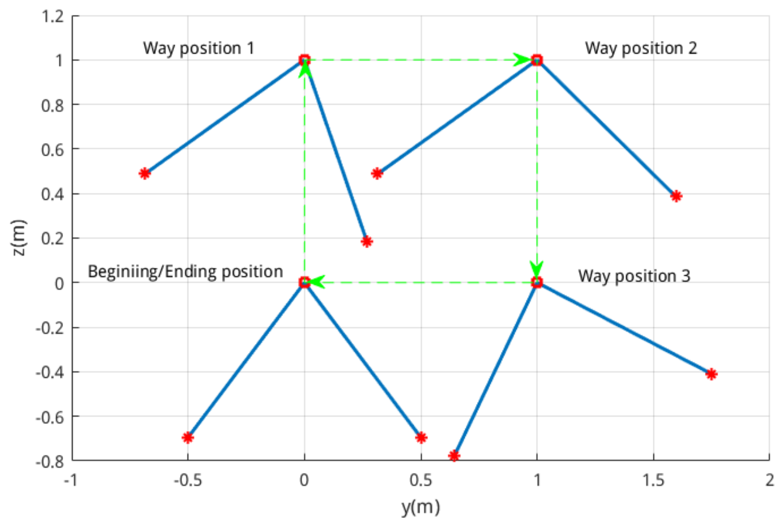


Fig. 3.2. Way positions in the task space

3.2.1 Decision variables

Way points

Clearly, we can see from Eq.3.19 that a trajectory depends on way points and travelling time for segments. Only the way positions are given by user or decided by the task. Instead of setting $\dot{\mathbf{s}}_w = \ddot{\mathbf{s}}_w = \mathbf{s}_w^{(3)} = \mathbf{s}_w^{(4)} = 0$ for every way point, we set $\begin{bmatrix}\dot{\mathbf{s}}_w & \ddot{\mathbf{s}}_w & \mathbf{s}_w^{(3)} & \mathbf{s}_w^{(4)}\end{bmatrix}$ as decision variables in the optimisation. Thus, the robot does not have to ”stop” at every way position.

A decision variable is defined as

$$\mathbf{S}_{d1} = \begin{bmatrix} \dot{\mathbf{s}}_{w_1} & \ddot{\mathbf{s}}_{w_1} & \mathbf{s}_{w_1}^{(3)} & \mathbf{s}_{w_1}^{(4)} \\ & \cdots & & \\ \dot{\mathbf{s}}_{w_k} & \ddot{\mathbf{s}}_{w_k} & \mathbf{s}_{w_k}^{(3)} & \mathbf{s}_{w_k}^{(4)} \\ & \cdots & & \\ \dot{\mathbf{s}}_{w_{n_{wp}}} & \ddot{\mathbf{s}}_{w_{n_{wp}}} & \mathbf{s}_{w_{n_{wp}}}^{(3)} & \mathbf{s}_{w_{n_{wp}}}^{(4)} \end{bmatrix} \tag{3.20}$$

Via points

We aim to optimise the trajectory and improve the robot performance further. Especially, way positions given by the task are far away from each other.

We introduce via point $\mathbf{S}_v$ defined as

$$\mathbf{S}_v = \begin{bmatrix}\mathbf{s}_v & \dot{\mathbf{s}}_v & \ddot{\mathbf{s}}_v & \mathbf{s}_v^{(3)} & \mathbf{s}_v^{(4)}\end{bmatrix} \tag{3.21}$$

One example of via points is shown in the Fig.3.3. Way point 1 and way point 2 are too far away from each other. Two via points are introduced in this trajectory segment, which can possibly change the shape of the path drastically.

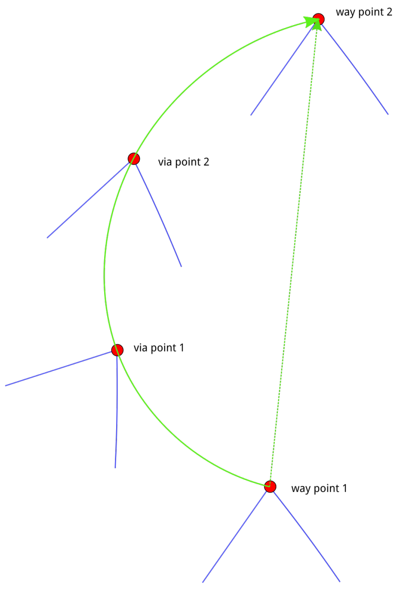


Fig. 3.3. One example of via points

As a consequence, we define another decision variable as

$$\mathbf{S}_{d2} = \begin{bmatrix} \mathbf{S}_{v_1} \\ \cdots \\ \mathbf{S}_{v_k} \\ \cdots \\ \mathbf{S}_{v_{n_{vp}}} \end{bmatrix} = \begin{bmatrix} \mathbf{s}_{v_1} & \dot{\mathbf{s}}_{v_1} & \ddot{\mathbf{s}}_{v_1} & \mathbf{s}^{(3)}_{v_1} & \mathbf{s}^{(4)}_{v_1} \\ & & \cdots & & \\ \mathbf{s}_{v_k} & \dot{\mathbf{s}}_{v_k} & \ddot{\mathbf{s}}_{v_k} & \mathbf{s}^{(3)}_{v_k} & \mathbf{s}^{(4)}_{v_k} \\ & & \cdots & & \\ \mathbf{s}_{v_{n_{vp}}} & \dot{\mathbf{s}}_{v_{n_{vp}}} & \ddot{\mathbf{s}}_{v_{n_{vp}}} & \mathbf{s}^{(3)}_{v_{n_{vp}}} & \mathbf{s}^{(4)}_{v_{n_{vp}}} \end{bmatrix} \tag{3.22}$$

where n_{vp} is the number of via points in the optimisation for one segment trajectory.

In order to simplify the case, we make two assumptions

1. the number of via points n_{vp} can be decided by a user or his task.
2. the travelling time for one segment is distributed to the travelling time between two via points evenly.

Finally, the decision variables of the optimisation are via points in the every segment and $\begin{bmatrix} \dot{\mathbf{s}}_w & \ddot{\mathbf{s}}_w & \mathbf{s}^{(3)}_w & \mathbf{s}^{(4)}_w \end{bmatrix}$ of way points of the whole trajectory.

$$\mathbf{S}_{d1}, \mathbf{S}_{d2} \tag{3.23}$$

3.2.2 Objective function

The main purpose of the motion optimisation is to minimise two terms: energy cost of the robot and perturbations in the controller.

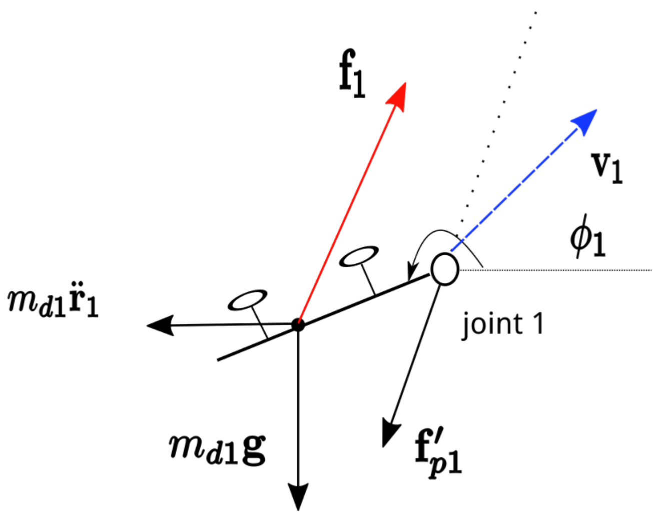


Fig. 3.4. Forces acting on the quadrotor 1 and the velocity of joint 1

We can see from the Fig.3.4 that input force on the quadrotor 1 has to compensate the gravity force $m_{d1}\mathbf{g}$, the centripetal force $m_{d1}\ddot{\mathbf{r}}_1$ ($\mathbf{r}_1$ is positions of joint 1 in the vertical plane) and the reaction force from the chain $\mathbf{f}'_{p1}$ that is decided by the dynamic of motion.

As the linear velocity $\mathbf{v}_1$ can be easily computed through DKM (direct kinematic model) of the parallel robot, we can approximatethe energy consumption of quadrotor 1 with $|\mathbf{f}_1 \cdot \mathbf{v}_1|$.

In this case, the energy consumption of FPR is approximated with

$$J_e = \int |\mathbf{f}_1 \cdot \mathbf{v}_1| + |\mathbf{f}_2 \cdot \mathbf{v}_2| dt \tag{3.24}$$

We recall the expression of $\boldsymbol{\delta}_p$ in 2.22

$$\boldsymbol{\delta}_p = \mathbf{R}_{inv} \begin{bmatrix} -m_{d1}d_1\dot{\phi}_1^2 \\ 0 \\ -m_{d2}d_2\dot{\phi}_2^2 \\ 0 \end{bmatrix}$$

It is suggested that the roll angular velocity $\dot{\phi}$ plays a significant part in the perturbation and it induces a challenge for the controller. Even though the controller used in [33] are proven to be robust, it is still beneficial if the perturbation $\dot{\phi}$ can be optimised in the motion planning. Thus, we introduce a perturbation index:

$$J_\phi = \int |\dot{\phi}_1^2| + |\dot{\phi}_2^2| dt \tag{3.25}$$

Energy J_e and perturbation J are two objectives in our optimisation.

1. we introduce a weight α to transform this problem from a multi-objective optimization into a single objective optimisation. Users can choose the priority term form energy and perturbation to optimise according to their missions and tasks.
2. J_e and J_ϕ share different units and distinct orders of magnitude, from which the optimisation process is influenced. Both of J_e and J_ϕ are normalised by the case that we

set

$$\dot{\mathbf{s}}_w = \ddot{\mathbf{s}}_w = \mathbf{s}_w^{(3)} = \mathbf{s}_w^{(4)} = 0$$

which means the robot is asked to stop at the each way position.

Finally, the objective function is modelled as

$$J = \alpha \cdot \frac{J_e}{J_{e0}} + (1-\alpha) \cdot \frac{J_\phi}{J_{\phi 0}} \tag{3.26}$$

where J_{e0} and $J_{\phi 0}$ are the energy and perturbation when $\dot{\mathbf{s}}_w = \ddot{\mathbf{s}}_w = \mathbf{s}_w^{(3)} = \mathbf{s}_w^{(4)} = 0$.

3.2.3 Constraints

Several constrains should be involved in the motion planning.

1. Safety flying conditions

Quadrotors should fly above the ground and never get too close to the ground. We can use the location of the joints linked to quadrotors to approximate the drone. The geometry of the passive chain is shown in Fig.3.5 .

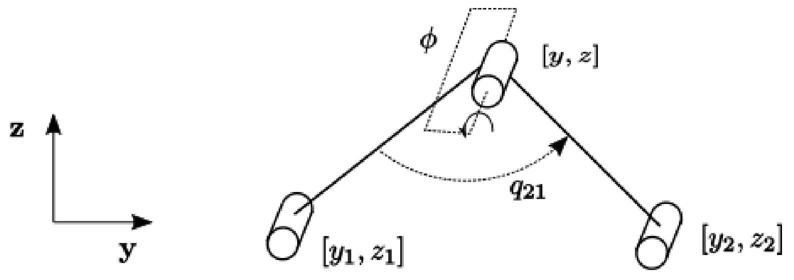


Fig. 3.5. Geometry of the passive chain

Constraints on the locations of the quadrotors can be modelled as

$$z_i \geq -Llink + m_{gd} \tag{3.27}$$

where $Llink$ stands for the length of the link and $m_{gd} = 0.1$ refers to the distance margin used in the optimisation.

2. Not cross the type 2 singularity.

$$\pi + m_{ga} \leq \phi \leq 2\pi - m_{ga} \tag{3.28}$$
$$m_{ga} \leq q_{21} \leq \pi - m_{ga} \tag{3.29}$$

where m_{ga} is an angle margin in this optimisation and $m_{ga} = 0.2$.

3. Dynamic limits of quadrotors

The force generated by one rotor of each drone, f_{im}, should respect the corresponding limit. we define $\boldsymbol{\rho}_i = [f_{i1}, f_{i2}, f_{i3}, f_{i4}]$ as a vector consisting of four rotor forces generated of drone i.

$$\boldsymbol{\rho}_i = \mathbf{c}_T \boldsymbol{\varpi}_i \tag{3.30}$$

where $\mathbf{c}_T = \begin{bmatrix} c_T & c_T & c_T & c_T \end{bmatrix}$. Based on the Eq.3.9 and Eq.3.30, we can get

$$\boldsymbol{\rho}_i = \mathbf{c}_T \mathbf{R}_f^{(-1)} \begin{bmatrix} f_{iz} \\ \boldsymbol{\tau}_i \end{bmatrix} \tag{3.31}$$

We can obtain $f_{iz}, \boldsymbol{\tau}_i$ by computing the required input force and torques from Eq.3.12 and Eq.2.24 of a trajectory. After that, the constraint for every rotor force can be expressed as

$$|f_{im}| \leq f_{max} - m_{gf}, m = 1, 2, 3, 4, i = 1, 2 \tag{3.32}$$

$f_{max} = 9N$ which depends on the exact drones used in the FPR and $m_{gf} = 1.5N$ is defined as a force margin.

4. Quadrotors should not roll over.

A drone is under-actuated and usually ϕ_i works as virtual control in the backstepping controller. In other words, we can not know the ϕ_i directly in the motion planning. Furthermore, the drone is not supposed to roll over just to simplify the robot motion and guarantee a safety flight.

$$|\phi_i| \leq \frac{\pi}{2} - m_{ga} \tag{3.33}$$

3.2.4 Motion optimisation formulation

Decision variables:

$$\mathbf{S}_{d1}, \mathbf{S}_{d2}$$

where $\mathbf{S}_{d1}$ in defined in Eq.3.20, $\mathbf{S}_{d2} = \begin{bmatrix} \mathbf{S}_{v1} & \dots & \mathbf{S}_{vk} & \dots & \mathbf{S}_{vn_{vp}} \end{bmatrix}^T$ defined in Eq.3.39.

Objective function:

$$J = \alpha \cdot \frac{J_e}{J_{e0}} + (1 - \alpha) \cdot \frac{J_\phi}{J_{\phi 0}}$$

where J_{e0} and $J_{\phi 0}$ are the energy and perturbation when $\dot{\mathbf{s}}_w = \ddot{\mathbf{s}}_w = \mathbf{s}_w^{(3)} = \mathbf{s}_w^{(4)} = 0$.

Constraints:

$$\begin{aligned} z_i &\geq -Llink + m_{gd} \\ \pi + m_{ga} \leq \phi &\leq 2\pi - m_{ga} \\ m_{ga} \leq q_{21} &\leq \pi - m_{ga} \\ |f_{im}| &\leq f_{max} \\ |\phi_i| &\leq \frac{\pi}{2} - m_{ga} \end{aligned}$$

This optimisation problem is solved in a numerical way.

Method *fmincon* in Matlab.

fmincon is a non-linear programming solver.

Algorithm *interior-point.*

interior-point is the default and first recommended algorithm in the *fmincon* and it distinguishes itself by solving large-scale problems.

In practice, we propose a 2-step optimisation method.

1. the 1st step is to carry out the optimisation with the decision variables being $\mathbf{S}_{d1}$. It means the optimisation only improve the trajectory by changing the velocities, accelerations, 3rd and 4th derivatives of the robot at every way positions.
2. the 2nd step is to conduct the optimisation for each segment separately whose the decision variable is $\mathbf{S}_{d2}$. This optimisation is based on the new way points which are the results of the optimisation in the 1st step.

As the computation time will increases drastically, if more decision variables are in the optimisation process. For instance, if we introduce one via point in every segment of trajectory in [33], the number of decision variables will be nearly a 32 by 4 matrix in the original method. With the two-step optimisation, decision variables will be just a 16 by 4 matrix in the first step and they will be a 5 by 4 matrix in each segment in the second step.

Obviously, an initial guess plays an essential part in the optimisation solver. Sometimes, a good initial guess can help the optimisation convergence fast. By contrast, a bad initial guess can leads to a infeasible solution. It is more difficult to find a proper initial guess for the optimisation case with more decision variables.

As a consequence, the 2-step optimisation is taken in the practice. It is believed that this 2-step optimisation is a suboptimal approach and some precisions could be lost but it has less computation complexity and easy initial guesses in a general way. An examples in Appendix shows that 2-step optimisation even causes a better result than the original method. But differences between those two methods will depend on the tasks.

3.2.5 Influences of number of via points and weights in objective functions

A user has to choose two parameters in the optimisation process: number of via points n_{vp} and weight α.

The effects given by the n_{vp} are investigated in the following mission:

- the initial position is $[0, 0, 3.9270, 1.5708]$ and final position is $[1, 1, 3.9270, 1.5708]$, which are shown in Fig.3.6. The 2nd-4th derivatives of the initial and final points are set to be zeros.
- the travelling time is set to be $3s$
- $\alpha = 0.5$

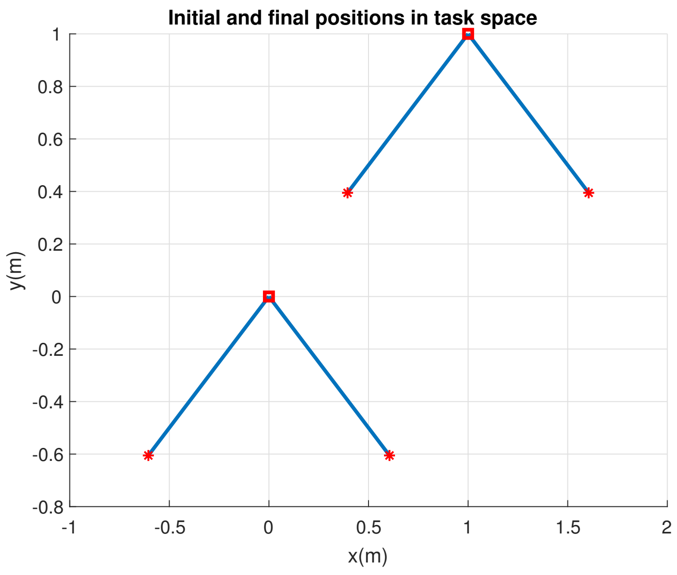


Fig. 3.6. A task to test optimisation parameters

Optimisation is conducted with different n_{vp}. In order to illustrate the effectiveness, we show $\frac{J_e}{J_{e0}}$ and $\frac{J_\phi}{J_{\phi 0}}$ in the Fig.3.7 and Fig.3.8, where J_{e0} and $J_{\phi 0}$ refer to the no via point case.

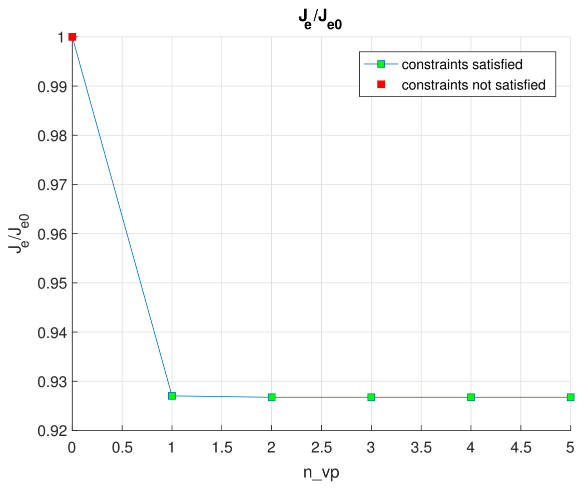


Fig. 3.7. Energy consumption with different number of via points

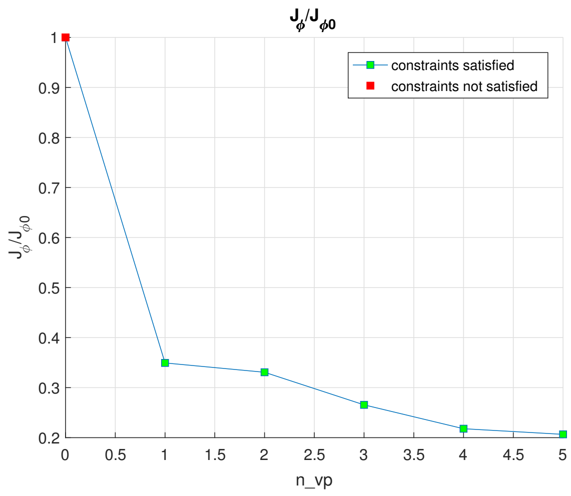


Fig. 3.8. Perturbation index with different number of via points

We can see from the Fig.3.7 and Fig.3.8 that introducing the via points can improve the robot performance drastically. For instance, the trajectory without any via point does not meet the constraints, while one via point can make this trajectory feasible. Also, it is clear that with more via points introduced as decision variables, the J_e and J_ϕ will decline more sharply, especially the drop of J_ϕ. However, the perturbation are easier to be optimised or changed than energy.

Then, influences of different α are investigated by the same task with n_{vp}:

- the initial position is is $[0, 0, 3.9270, 1.5708]$ and final position is $[1, 1, 3.9270, 1.5708]$, which are shown in Fig.3.6.
- the travelling time is set to be $3s$.
- $n_{vp} = 1$.

Several optimisations with $\alpha = 0, 0.3, 0.6, 0.9, 1$ are conducted. We compute $J_e/J_{e|\alpha=0}$ and $J_\phi/J_{\phi|\alpha=1}$ from the results in order to present the differences caused by different α in the optimisation process. $J_{e|\alpha=0}$ and $J_{\phi|\alpha=1}$ are J_e and J_ϕ when $\alpha = 0$ and $\alpha = 1$ respectively.

It is clear from the results in Fig.3.9 and Fig.3.10 that optimisation focus can be adjusted by tuning α. A larger α means that user hopes to minimise energy cost of robot more than the perturbation during the motion, and the minimisation of the perturbation index is more important if a smaller α is set.

To sum up,

1. A larger n_{vp} can cause a more optimal motion. However, the computation time will increase significantly if we choose a too large n_{vp}. A balance needs to be found between the optimisation and the computation cost.

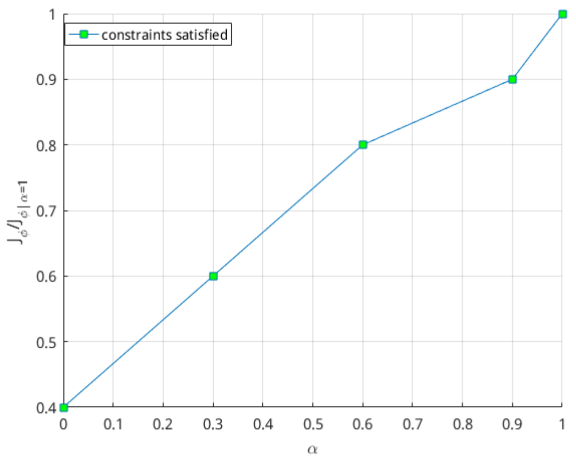


Fig. 3.9. Energy consumption with different weights

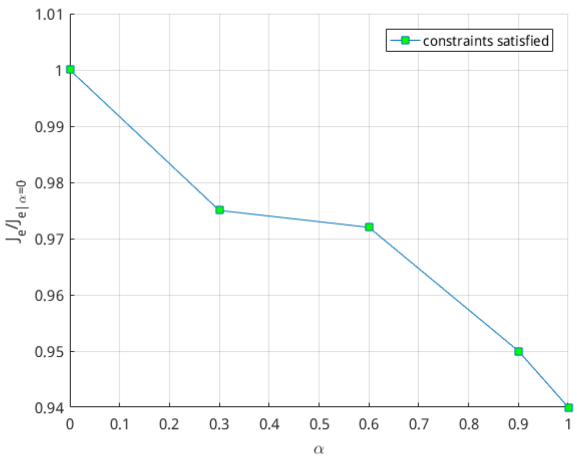


Fig. 3.10. Perturbation index with different weights

2. Energy computation is not improved markedly, not only in with different $n_v p$, but also with different α. But it is still believed to keep the energy consumption in the objective function, as energy consumption depends missions and as more flexibilities are provided to users.

3.2.6 Optimal motion VS no-optimal motion

First of all, an optimised motion is generated with the same way positions, beginning\ending position, same travelling time with that in [33]. We set $n_{vp} = 1, \alpha = 0.5$ during the optimisation.

The trajectory of the end-effector in task space is shown in the Fig.3.11 and the trajectory of each coordinate is illustrated in the Fig.3.12

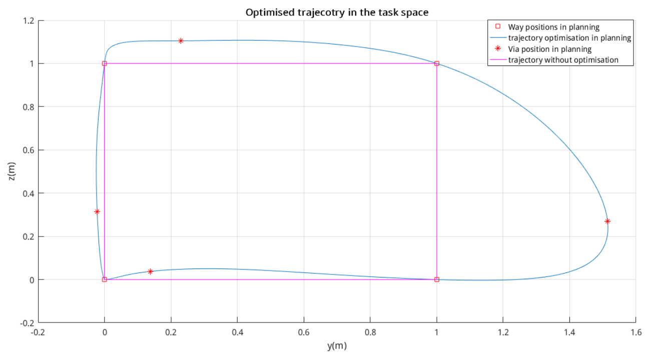


Fig. 3.11. An optimised trajectory of the end-effector in the task space

It is clear from Fig.3.11 and Fig.3.12 and that the trajectory after optimisation is smoother than the trajectory used in [33].

Fig.3.13 and Fig.3.14 show the J_e and J_ϕ of the optimised trajectory.

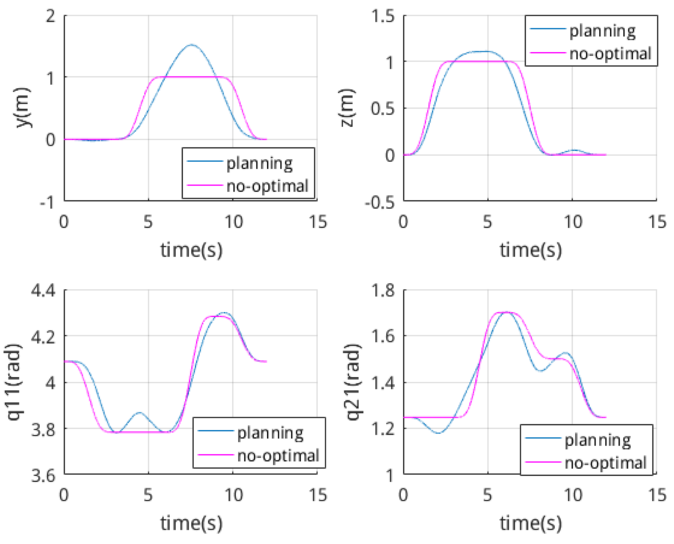


Fig. 3.12. An optimised trajectory of each coordinate

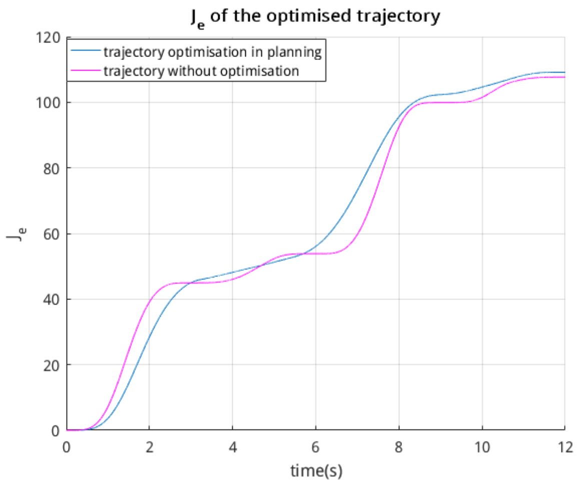


Fig. 3.13. Energy consumption in the trajectory

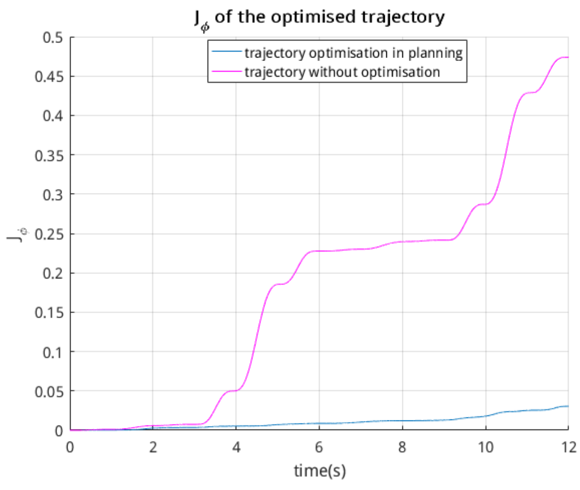


Fig. 3.14. Perturbation index in the trajectory

Fig.3.15, 3.16, 3.17, 3.18 show the $|\mathbf{ft}_1|, |\mathbf{ft}_2|, \phi_1, \phi_2$ of the optimised trajectory. We are able to find that the optimised trajectory have less aggressive $|\mathbf{ft}_1|, |\mathbf{ft}_2|, \phi_1, \phi_2$ than the non-optimal trajectory .

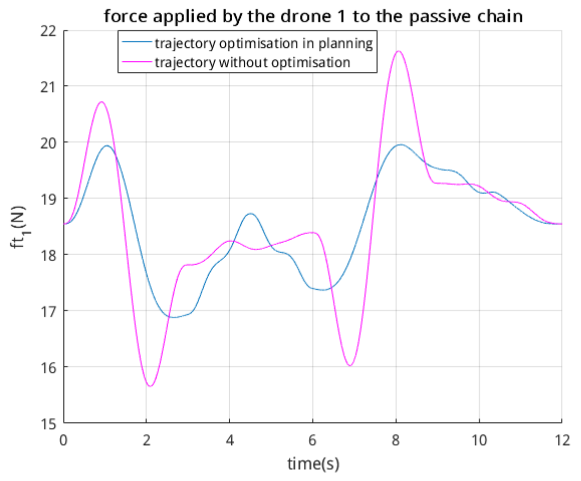


Fig. 3.15. Norm of input force of drone 1

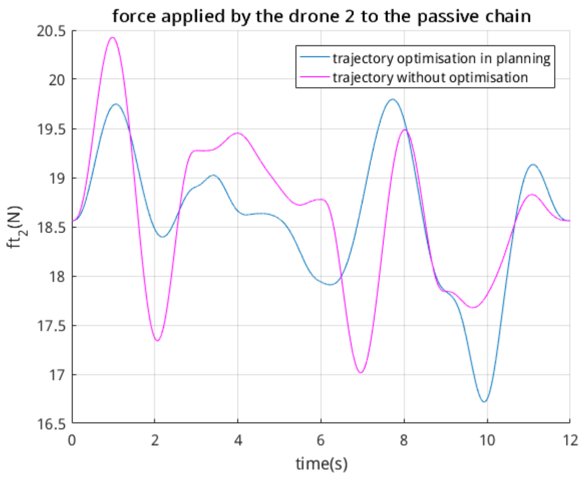


Fig. 3.16. Norm of input force of drone 2

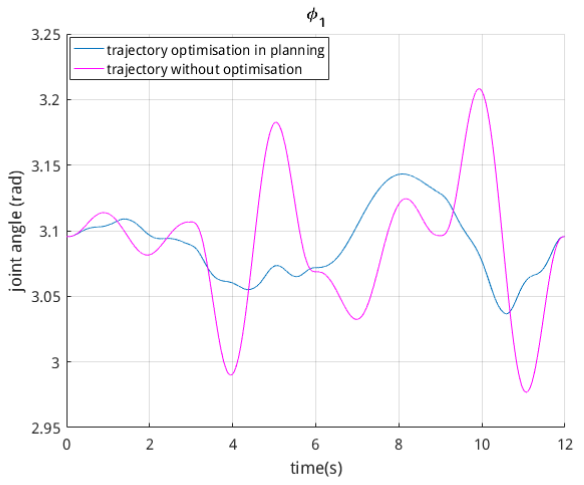


Fig. 3.17. Roll angle of drone 1

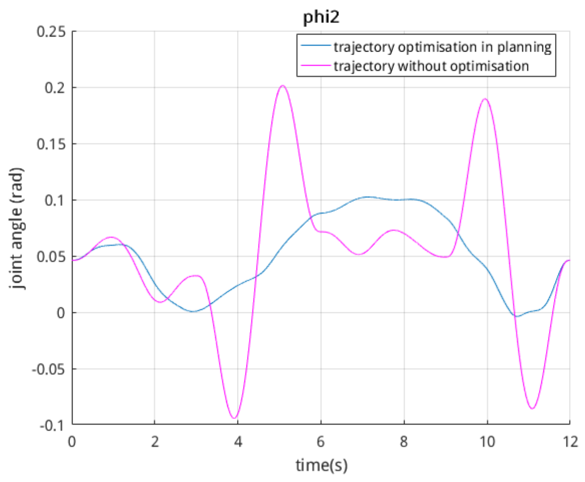


Fig. 3.18. Roll angle of drone 2

Then, we conduct the simulation in SimulinkMatlab and results are shown as follows.

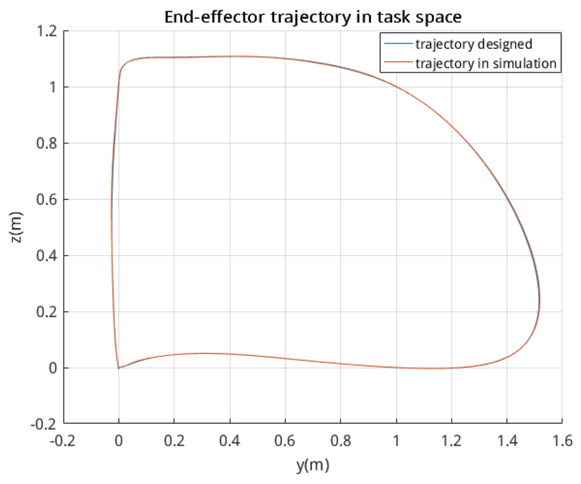


Fig. 3.19. Trajectory of the end-effector in task space

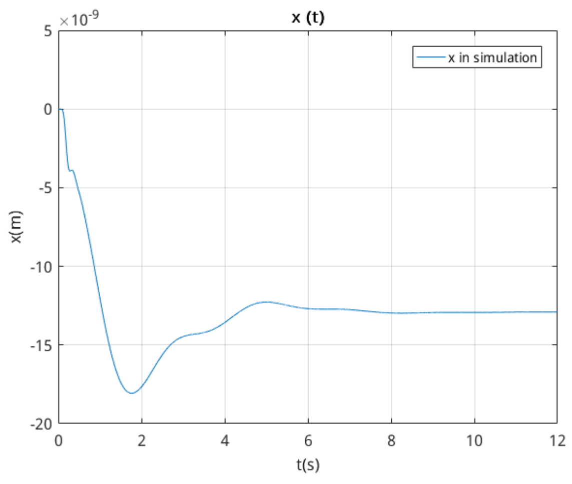


Fig. 3.20. End-effector in x coordinate

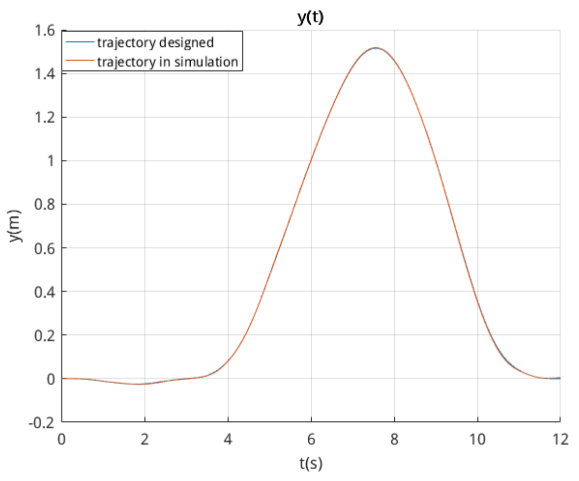


Fig. 3.21. End-effector in y coordinate

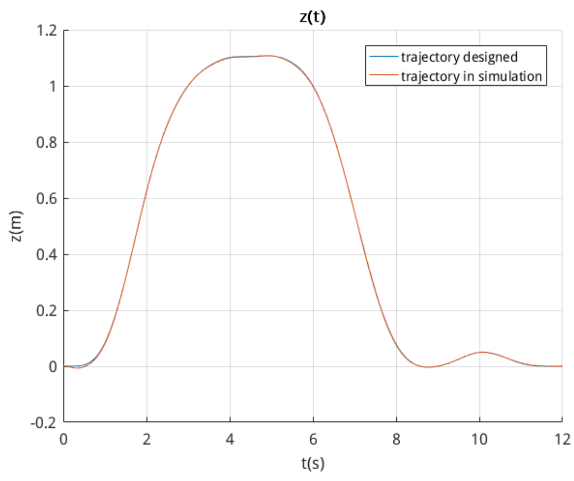


Fig. 3.22. End-effector in z coordinate

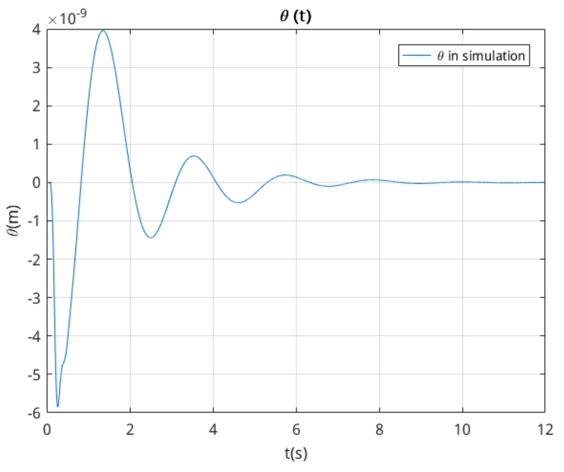


Fig. 3.23. Trajectory of end-effector's pitch angle

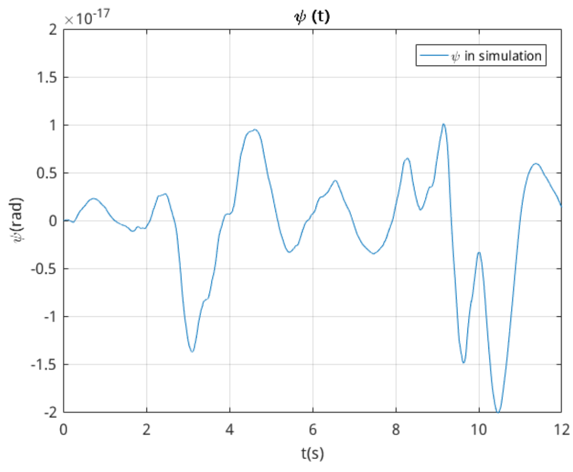


Fig. 3.24. Trajectory of end-effector's yaw angle

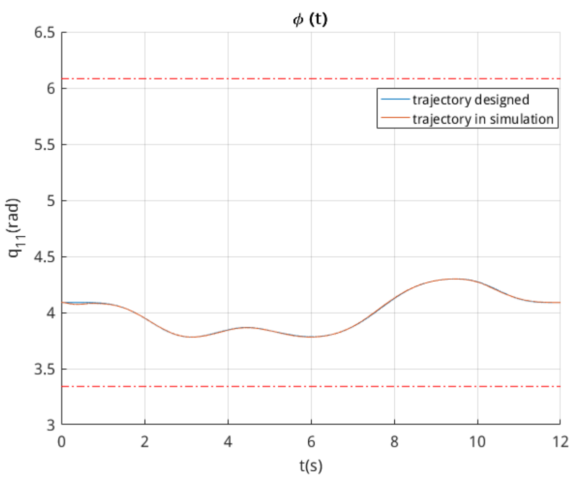


Fig. 3.25. Roll angle of end-effector

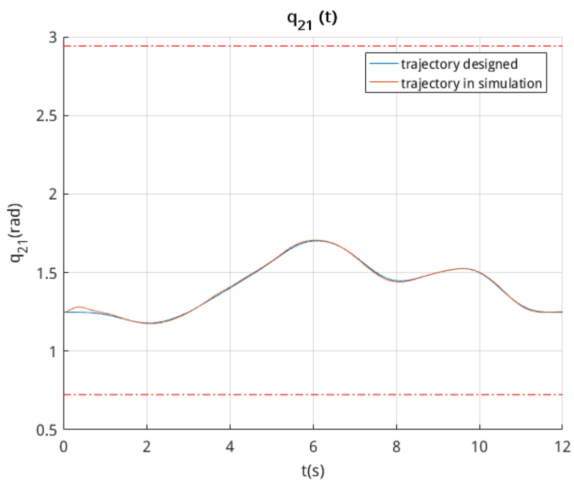


Fig. 3.26. Angle between two links q_{21}

Several analyses can be driven form the results presented in Fig.3.19-.3.32.

1. The motion of FRP is mainly in the vertical plane. $x(t), \theta(t), \psi(t)$ of PFR in Fig.3.20, 3.23, 3.24 are approximately level at around zeros.

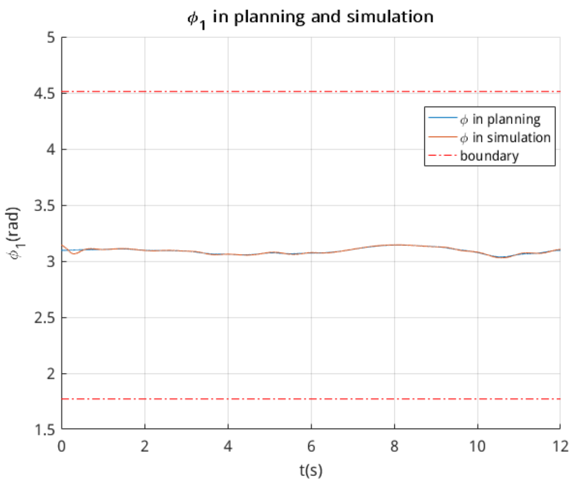


Fig. 3.27. Roll angle of drone 1

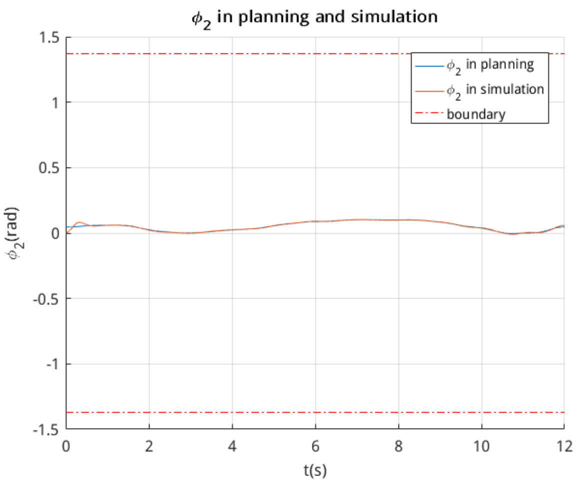


Fig. 3.28. Roll angle of drone 2

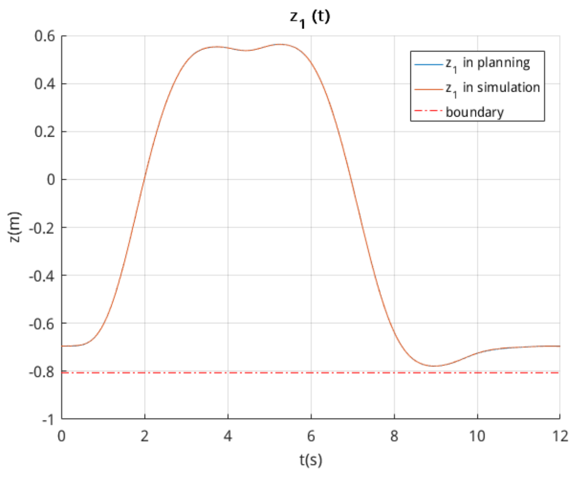


Fig. 3.29. Vertical coordinate of drone 1

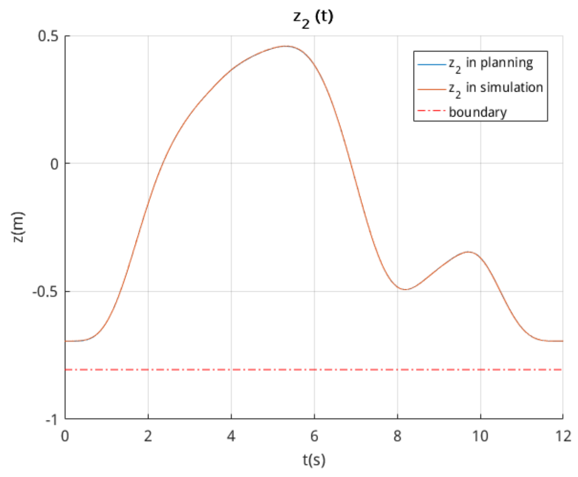


Fig. 3.30. Vertical coordinate of drone 2

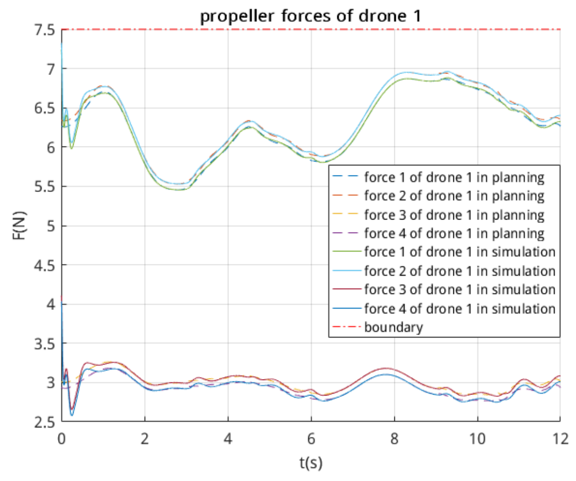


Fig. 3.31. Rotor forces of drone 1

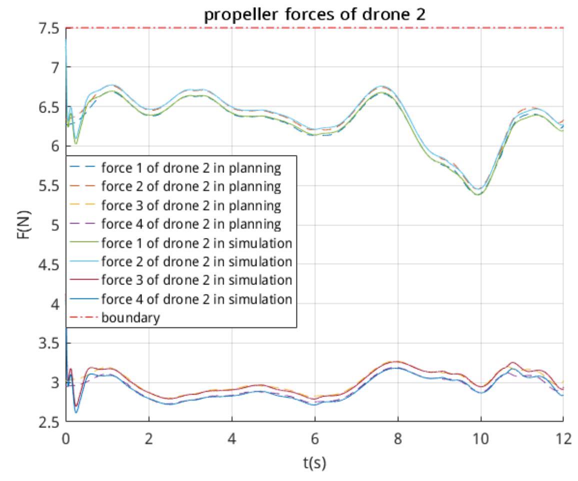


Fig. 3.32. Rotor forces of drone 2

2. Generally speaking, the controller designed in the [33] enables the robot to follow the trajectory designed as we can find from Fig.3.19, 3.21, 3.22, 3.25, 3.26. Constraints about the safety flight and rotors' limits are satisfied in the robot's motion in Fig.3.27, 3.28, 3.29, 3.30, 3.31, 3.32.

3. FPR takes about one third second to take off from the beginning\ending position.

3.3 Methods validation with a FPR prototype

3.3.1 Motion optimisation for the prototype

A FPR prototype has been developed in the LS2N lab as shown in Fig.3.33 and its kinematic structure is shown in the Fig.3.34.

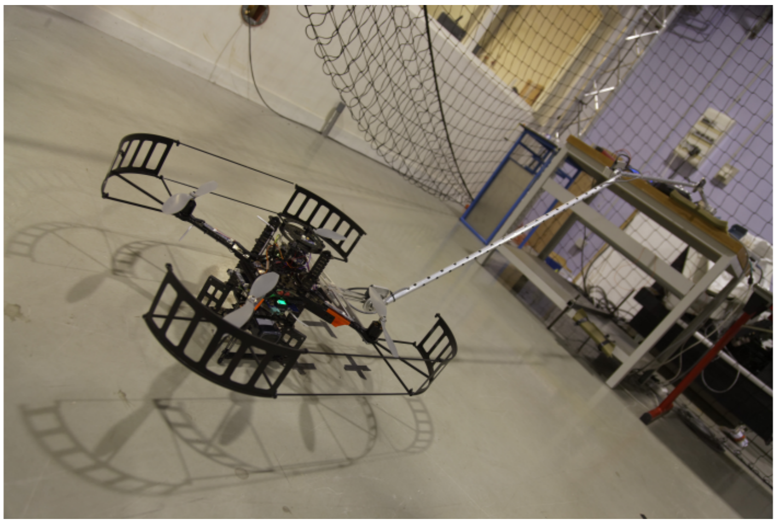


Fig. 3.33. FPR prototype in the lab

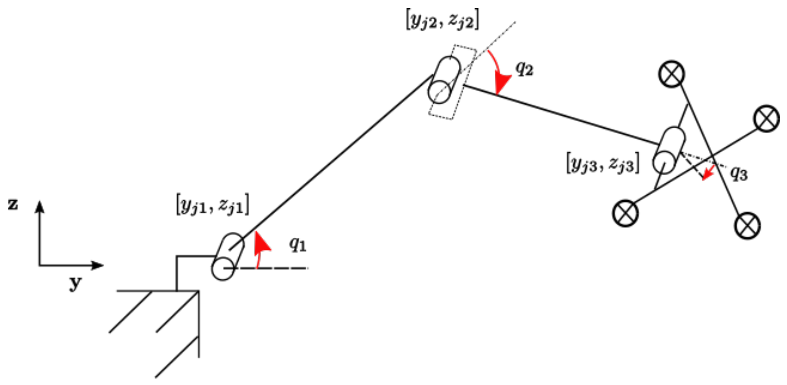


Fig. 3.34. Kinematic structure of prototype

Fig.3.35 shows the motion capture system built in the LS2N . It is used to obtain the position information of this robot in the experiment, from which we compute q_1, q_2.

This robot can be considered as a simplification of the FPR. Compared to FPR, one "actuator" in the prototype is replaced by a fixed point. As all the revolute joints are passive, the DOF of this robot is 2.

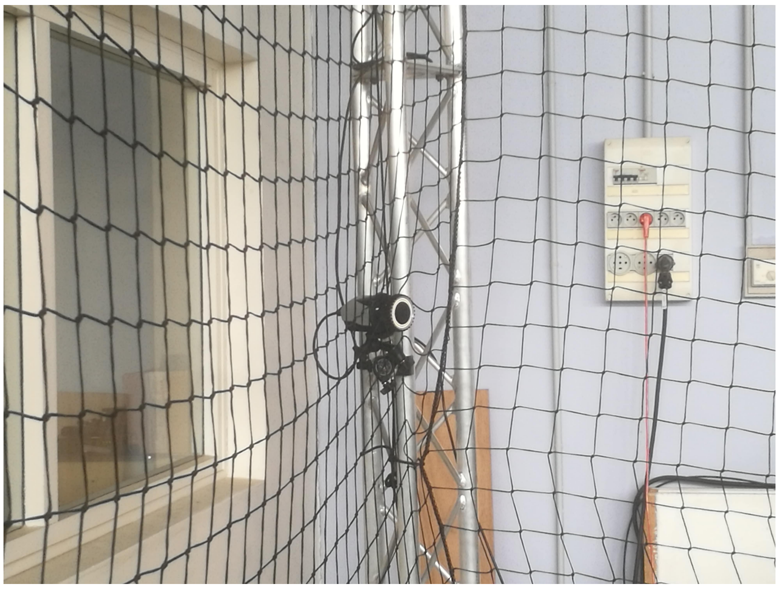


Fig. 3.35. Motion capture system

The vector for motion planning is defined as

$$\boldsymbol{\xi} = \begin{bmatrix} q_1 & q_2 \end{bmatrix} \tag{3.34}$$

A point is defined as $\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi} & \dot{\boldsymbol{\xi}} & \ddot{\boldsymbol{\xi}} & \boldsymbol{\xi}^{(3)} & \boldsymbol{\xi}^{(4)} \end{bmatrix}$.

Basic assumption are made as:

1. Trajectory generation for each coordinate is independent in $\boldsymbol{\xi}$
2. The robot does not change its configuration or cross singularity. The robot is set to be "arm down" configuration ($q_2 \geq 0$) without losing generality.
3. Navigation and control systems work in perfect ways.

Jacobian matrix $\mathbf{J}_{pr}$ of the prototype robot is defined as

$$\begin{bmatrix} \dot{y}_{j3} \\ \dot{z}_{j3} \end{bmatrix} = \mathbf{J}_{pr} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \tag{3.35}$$

and

$$\mathbf{J}_{pr} = \begin{bmatrix} Llink \cdot [sin(q_2 - q_1) - sin(q_1)] & -Llink \cdot sin(q_2 - q_1) \\ Llink \cdot [cos(q_2 - q_1) + cos(q_1)] & -Llink \cdot cos(q_2 - q_1) \end{bmatrix} \tag{3.36}$$

With similar attempts in Chapter 2, the dynamic model of the prototype robot is

$$\mathbf{J}_{pr}^T \mathbf{f}_1 = \mathbf{M}_{pt}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \mathbf{c}_{pt}(\dot{\boldsymbol{\xi}}, \boldsymbol{\xi}) \tag{3.37}$$

- $\mathbf{f}_1$ and $\boldsymbol{\tau}_1$ are the input force and torques of the robot.

- $\mathbf{M}_{pt}$ is the definite positive generalised inertial matrix depending on $\boldsymbol{\xi}$; $\mathbf{c}_{pt}$ refers to a vector of Coriolis and centrifugal effects depending on $\boldsymbol{\xi}$ and $\dot{\boldsymbol{\xi}}$. This work is contributed by SIX Damien.

The 2-step optimisation is conducted to find a optimal motion for the robot.

- the 1st step is to carry out the optimisation with the decision variables being Ξ_{d1}. It means the optimisation only improve the trajectory by changing the velocities, accelerations, 3rd and 4th derivatives of the robot at every way position.

$$\Xi_{d1} = \begin{bmatrix} \dot{\boldsymbol{\xi}}_{w_1} & \ddot{\boldsymbol{\xi}}_{w_1} & \boldsymbol{\xi}^{(3)}_{w_1} & \boldsymbol{\xi}^{(4)}_{w_1} \\ & \cdots & & \\ \dot{\boldsymbol{\xi}}_{w_k} & \ddot{\boldsymbol{\xi}}_{w_k} & \boldsymbol{\xi}^{(3)}_{w_k} & \boldsymbol{\xi}^{(4)}_{w_k} \\ & \cdots & & \\ \dot{\boldsymbol{\xi}}_{w_{n_{wp}}} & \ddot{\boldsymbol{\xi}}_{w_{n_{wp}}} & \boldsymbol{\xi}^{(3)}_{w_{n_{wp}}} & \boldsymbol{\xi}^{(4)}_{w_{n_{wp}}} \end{bmatrix} \tag{3.38}$$

  where $\dot{\boldsymbol{\xi}}_w, \ddot{\boldsymbol{\xi}}_w, \boldsymbol{\xi}^{(3)}_w, \boldsymbol{\xi}^{(4)}_w$ are velocity, acceleration, 3rd and 4th derivatives corresponding to the way points.

- the 2nd step is to conduct the optimisation for each segment separately whose the decision variable is Ξ_{d2}. This optimisation is based on the new way points which are the results of the optimisation in the 1st step.

$$\Xi_{d2} = \begin{bmatrix} \Xi_{v_1} \\ \cdots \\ \Xi_{v_k} \\ \cdots \\ \Xi_{v_{n_{vp}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\xi}_{v_1} & \dot{\boldsymbol{\xi}}_{v_1} & \ddot{\boldsymbol{\xi}}_{v_1} & \boldsymbol{\xi}^{(3)}_{v_1} & \boldsymbol{\xi}^{(4)}_{v_1} \\ & & \cdots & & \\ \boldsymbol{\xi}_{v_k} & \dot{\boldsymbol{\xi}}_{v_k} & \ddot{\boldsymbol{\xi}}_{v_k} & \boldsymbol{\xi}^{(3)}_{v_k} & \boldsymbol{\xi}^{(4)}_{v_k} \\ & & \cdots & & \\ \boldsymbol{\xi}_{v_{n_{vp}}} & \dot{\boldsymbol{\xi}}_{v_{n_{vp}}} & \ddot{\boldsymbol{\xi}}_{v_{n_{vp}}} & \boldsymbol{\xi}^{(3)}_{v_{n_{vp}}} & \boldsymbol{\xi}^{(4)}_{v_{n_{vp}}} \end{bmatrix} \tag{3.39}$$

The objective function is modified as follows:

$$J = \alpha \cdot \frac{J_e}{J_{e0}} + (1-\alpha) \cdot \frac{J_\phi}{J_{\phi 0}}$$

where

- $J_e = \int |\mathbf{f}_1 \cdot \mathbf{v}_1| dt$ and $J_\phi = \int |\dot{\phi}_1^2| dt$
- J_{e0} and $J_{\phi 0}$ are the energy and perturbation when $\dot{\boldsymbol{\xi}}_w = \ddot{\boldsymbol{\xi}}_w = \boldsymbol{\xi}^{(3)}_w = \boldsymbol{\xi}^{(4)}_w = 0$

In terms of constraints for this robot:

1. the drone should not touch the ground or the fixed point.

$$z_{j2} \geq -Llink + m_{gd} \tag{3.40}$$
$$z_{j3} \geq -Llink + m_{gd} \tag{3.41}$$
$$y_{j2} \geq m_{gd} \tag{3.42}$$
$$y_{j3} \geq m_{gd} \tag{3.43}$$
$$|q_1| \leq \frac{\pi}{2} - m_{ga} \tag{3.44}$$

2. the drone should not roll over.

$$|\phi_1| \leq \frac{\pi}{2} - m_{ga} \tag{3.45}$$

3. the robot should not cross the singularity.

$$m_{ga} \leq q_2 \leq \pi - m_{ga} \tag{3.46}$$

4. When it comes to the rotor limits, the four rotors angular speeds are considered rather than forces produced by them. That is much easier to measure rotors angular speeds in the real experiment.

$$\boldsymbol{\varpi}_1 = \mathbf{R}_f^{(-1)} \begin{bmatrix} f_{1z} \\ \boldsymbol{\tau}_1 \end{bmatrix} \tag{3.47}$$

   where f_{1z} and $\boldsymbol{\tau}_1$ can be computed from Eq.2.7 and Eq.3.37 .

   Constraints are presented as

$$0 \leq \varpi_{ij} \leq \varpi_{max} - m_{gr}, i = 1, 2, j = 1, 2, 3, 4 \tag{3.48}$$

   where $\varpi_{max} = 900$ and the margin $m_{gr} = 150$

3.3.2 Simulation in Matlab

An optimal motion for the robot is generated and the trajectory of the drone is shown in the Fig.3.36, in which travelling time between two way points are $\begin{bmatrix} 5s & 8s & 8s & 8s & 8s & 8s & 5s \end{bmatrix}^T$. In the optimisation, we set $\alpha = 0.5$ and n_{vp} in each segment to be 1.

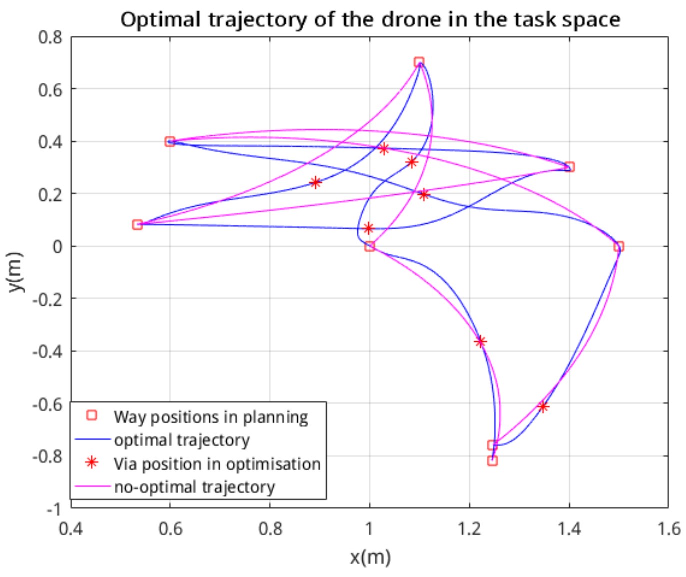


Fig. 3.36. An optimal trajectory of the drone of FPR prototype

Fig.3.37 shows the corresponding $q_1(t), q_2(t)$ of the motion planned. Then, J_e and J_ϕ in the optimal trajectory are shown in the Fig.3.38 and Fig.3.39. The $|\mathbf{f}_1|$ and ϕ_1 are shown in the Fig.3.40 and Fig.3.41.

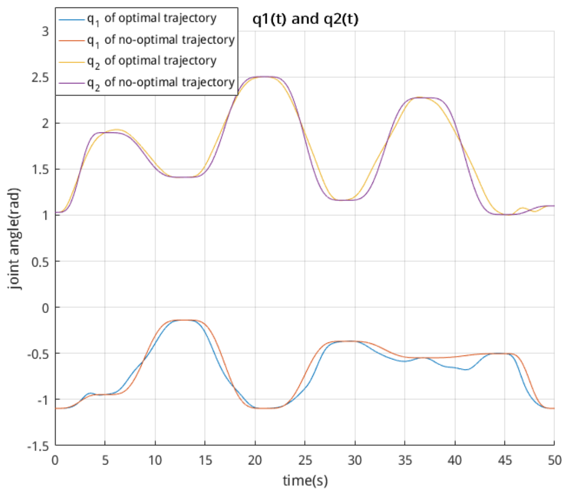


Fig. 3.37. Joint angles in an optimal trajectory of FPR prototype

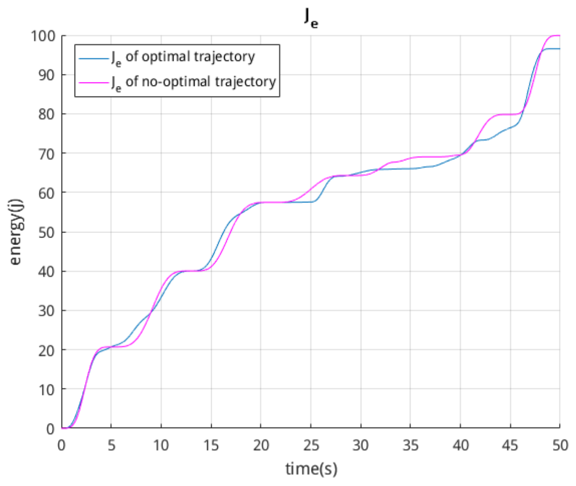


Fig. 3.38. Energy consumption of FPR prototype

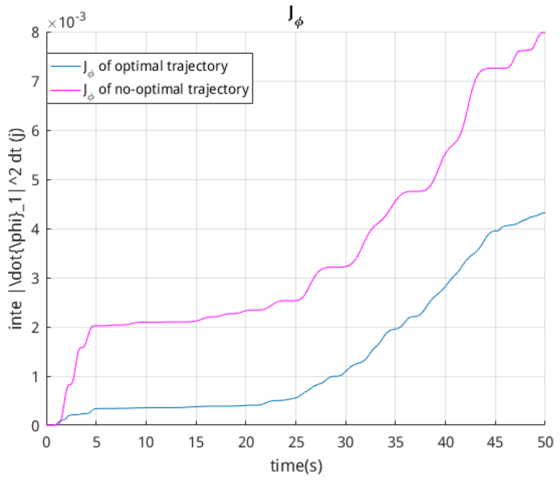


Fig. 3.39. Perturbation index of FPR prototype

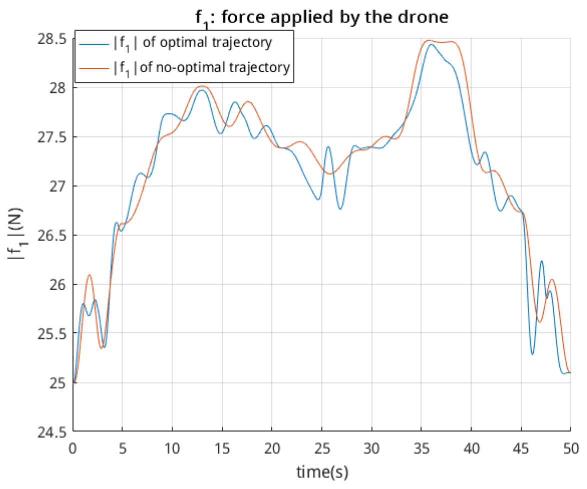


Fig. 3.40. Norm of input force of drone of FPR prototype

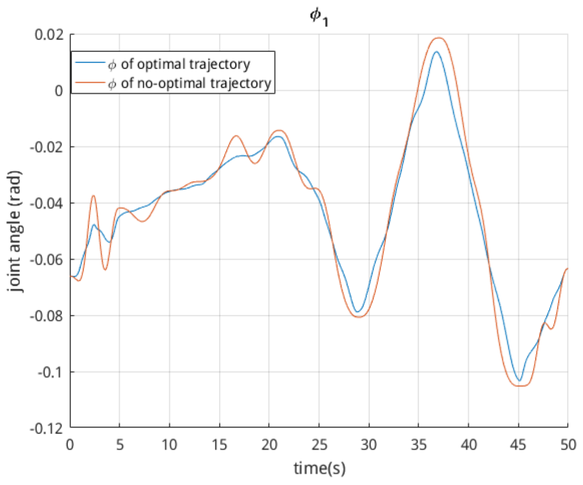


Fig. 3.41. Roll angle of FPR prototype

Simulation with the optimal trajectory in Matlab is conducted to test the controller of this prototype. The drone trajectory in simulation is shown in the Fig.3.42 and the error for q_1, q_2 are demonstrated in Fig.3.43

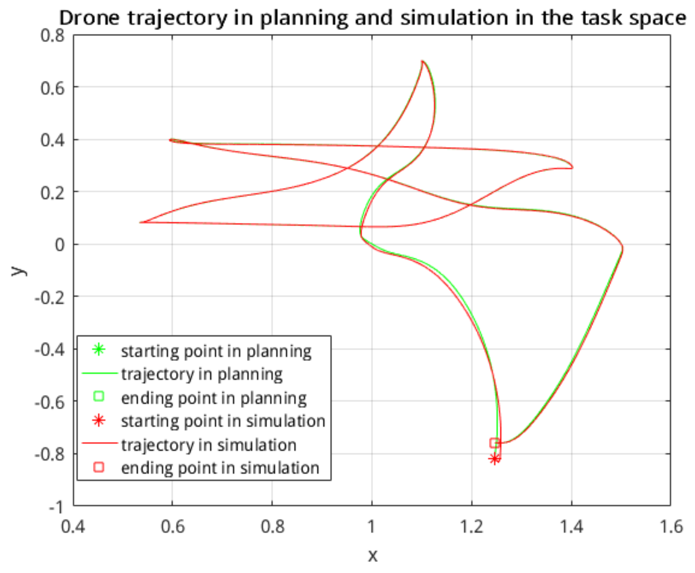


Fig. 3.42. Drone trajectory in planning and simulation

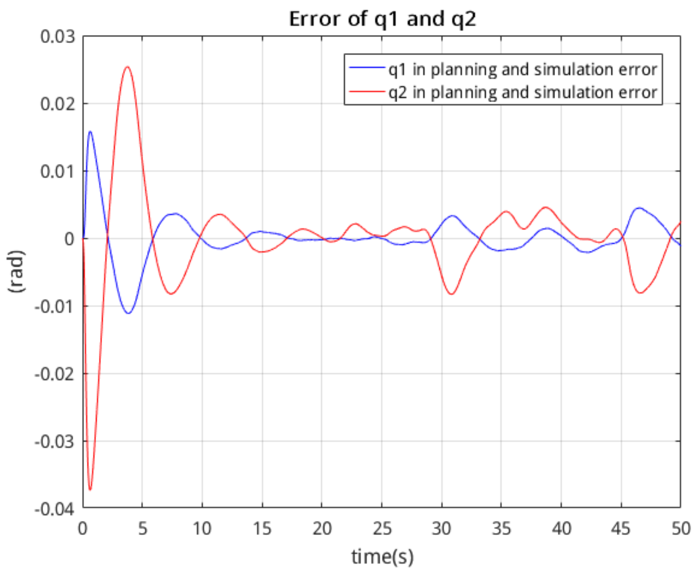


Fig. 3.43. Joints error in simulation

According to the results shown in Fig.3.42-3.51, we can obtain the conclusions as follows:

1. The controller can drive the robot to follow the optimal trajectory properly. Tracking error of q_1 and q_2 in the Fig.3.43 converge and level off at a certain neighbour of origin.
2. All the constraints modelled by the Eq.3.40-4 in the Fig.3.44-.3.49.
3. As we can find that from Fig.3.50, J_e is almost the same in planning and simulation. There is a large jump at the beginning of J_ϕ during the simulation. This is due to the drone takes off from the initial position which is not stable.

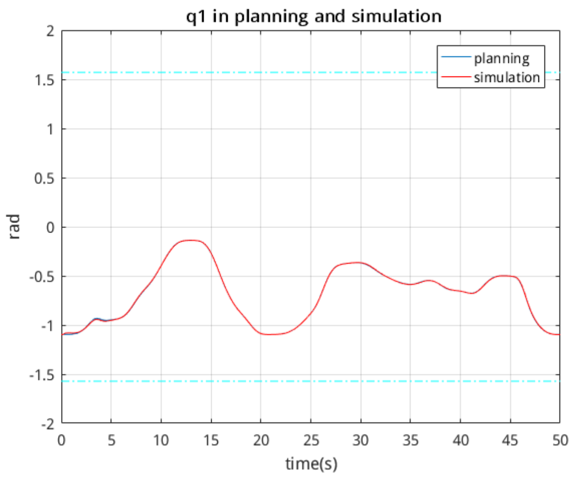


Fig. 3.44. Joint 1 of FPR prototype in simulation

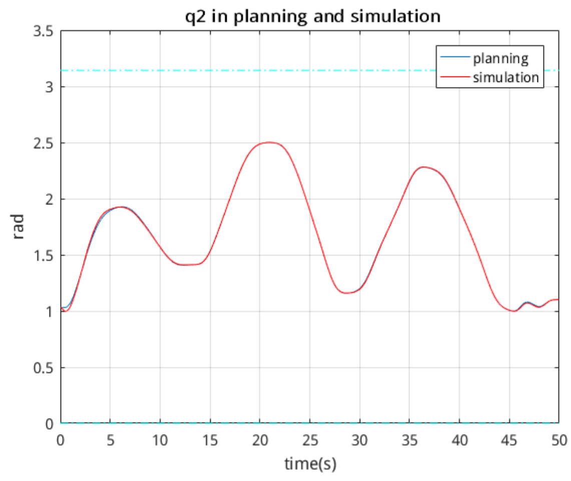


Fig. 3.45. Joint 2 of FPR prototype in simulation

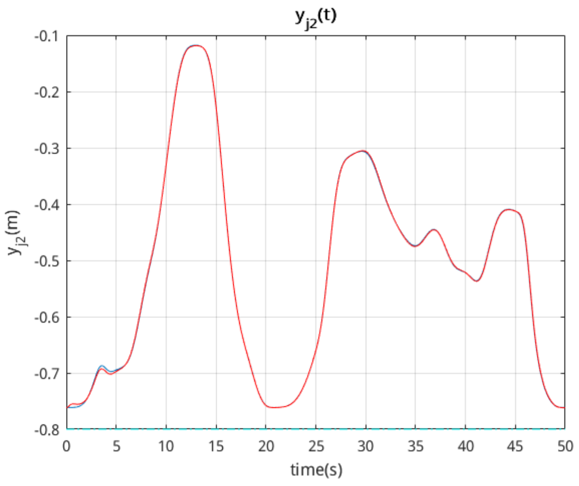


Fig. 3.46. Vertical coordinate of prototype joint 2

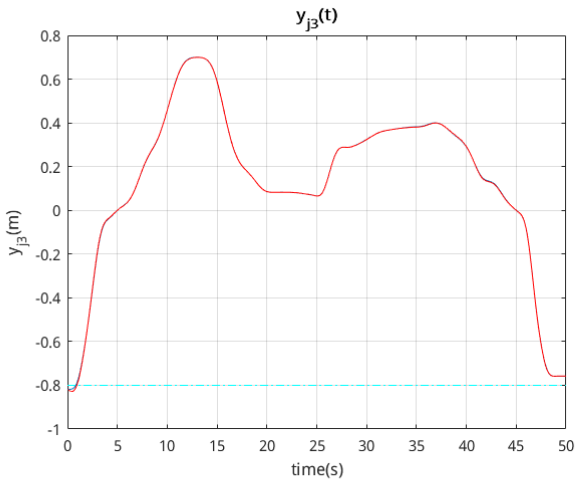


Fig. 3.47. Vertical coordinate of prototype joint 3

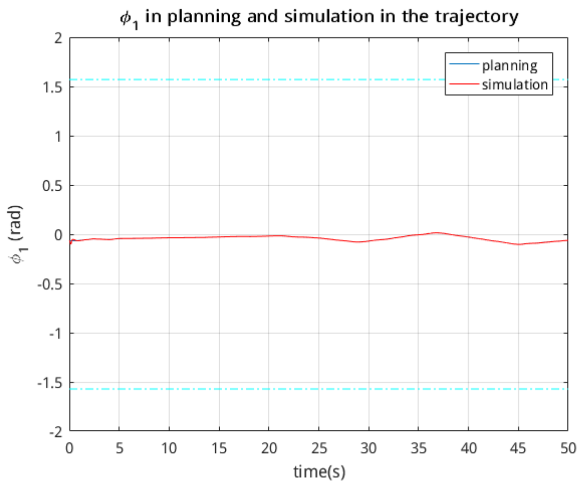


Fig. 3.48. Roll angle of FPR prototype in simulation

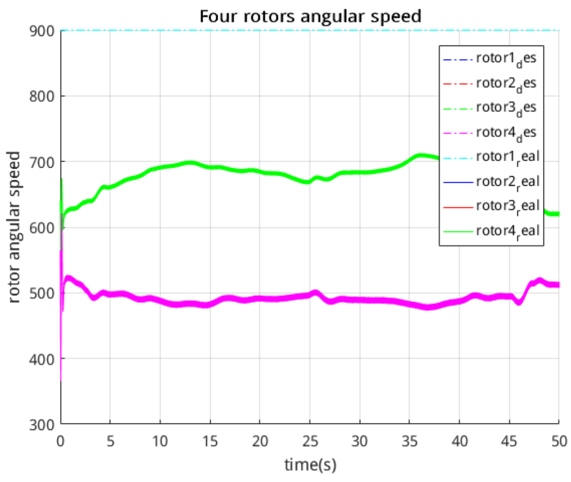


Fig. 3.49. Rotor forces of the drone in simulation

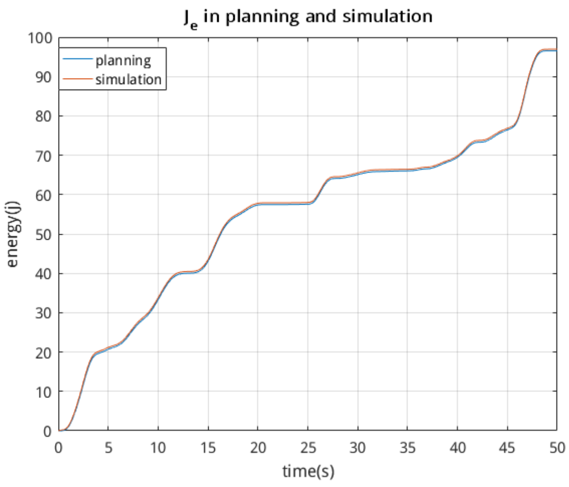


Fig. 3.50. Energy consumption in simulation

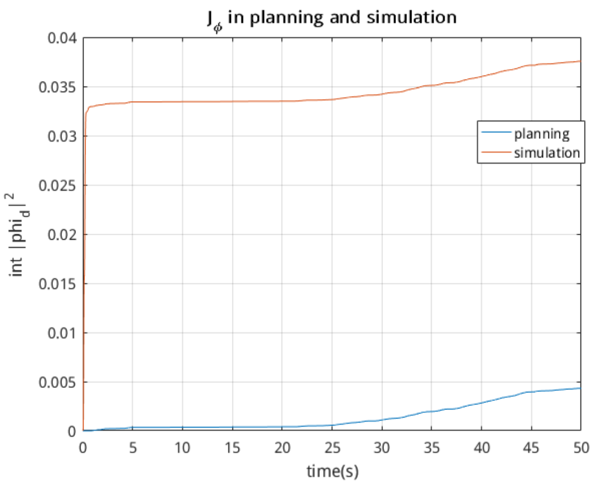


Fig. 3.51. Perturbation index in simulation

3.3.3 Real experiment with the prototype robot

The real experiments are conducted with this prototype robot in order to test our optimal trajectory. The tracking of $\boldsymbol{\xi}$ is shown in Fig.3.52 and Fig.3.53. Another two more examples in the real experiments are presented in the Appendix.

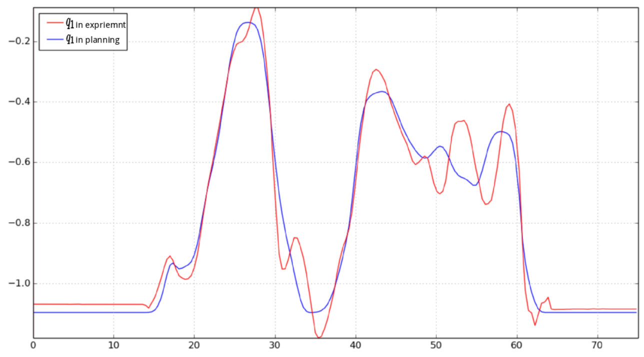


Fig. 3.52. Joint 1 tracking in the real experiment

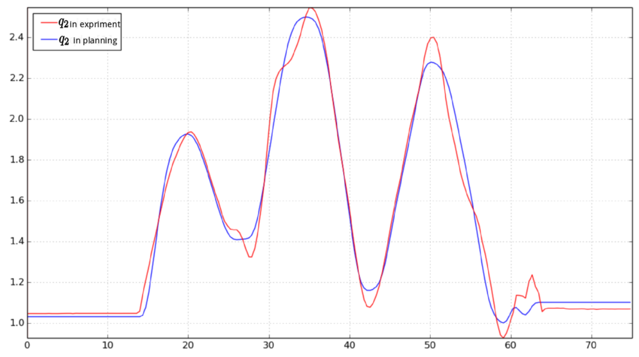


Fig. 3.53. Joint 2 tracking in the real experiment

It is clear from the Fig.3.52 and Fig.3.53 that the prototype robot tracks the motion planned properly in a general way with its controller. Still, we should notice two things:

1. when the drone is taking off and landing, the dynamics of robot, or the drone, is significantly affected by the ground effects. The thrust force rebound from the ground will impact the drone when the drone is just above the ground. The dynamic model of the robot does not consider the ground effects, so the performance is not perfect.
2. if the drone is far away from the fixed point, for instance during the $40s - 50s$, the tracking is a little worse than the before. The reason behind this is the challenge for the controller is much larger when the drone is far away from the base.

The θ is designed to force the robot motion to be in the vertical plane, which should be in

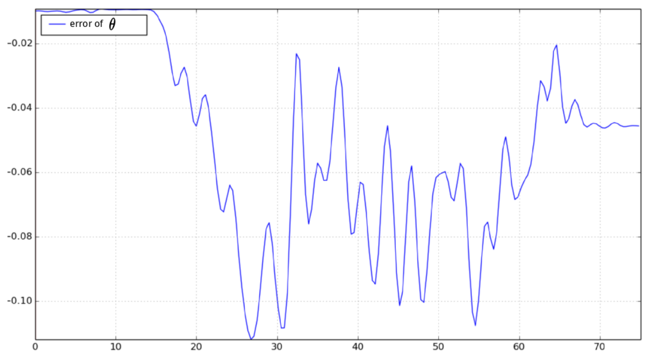


Fig. 3.54. Pitch angle error in the real experiment

the neighbour of zero. Comparing Fig.3.54 with Fig.3.52 and Fig.3.53, we can find that the absolute value of θ error reaches the peak around $0.1rad$ at $28s$ when the drone is at the highest position during the trajectory.

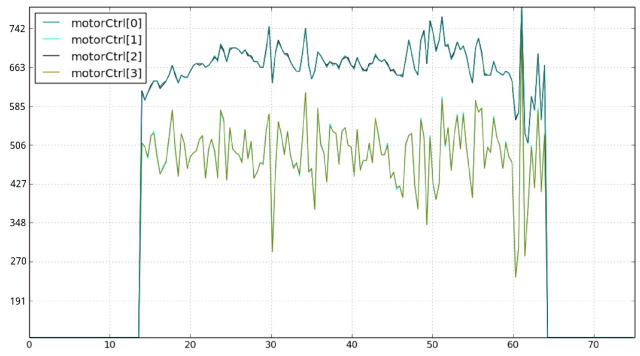


Fig. 3.55. Four motors of the drone in the real experiment

Fig.3.55 gives the performances of four motors in the experiment. That the trajectory does not meet the saturation of the actuator.

To sum up, the optimisation motion is tested in the real experiment. With the controller and the motion optimisation method, the robot is able to finish the mission properly.

Chapter 4

Cross Type 2 singularity for FPR prototype

4.1 Type 2 singularity of FPR prototype

Crossing Type 2 singularity of FPR is a challenge that is totally different from the typical parallel robot such as 5 bar mechanism.

- The actuators are two movable drones in the task space, while the actuated joints are fixed at certain positions in the 5 bar mechanism.
- Inputs of the 5 bar mechanism are torques of the actuated joints, but two under-actuated drones, more precisely the thrust forces and torques of the drones, are inputs of FPR.
- More constraints are on the inputs of FPR. the forces $\mathbf{f}_p$ applied to the passive chain depend on the thrust forces and the roll angles of drones which are limited for both of them. More precisely, the limits of thrust force of a drone is $F_{max} = 36N$ and the roll angles ϕ_1 should be smaller than $\frac{\pi}{2}$ to ensure that the robot is able to follow the motion planned.

The robot prototype can be considered as one special case of FPR when one drone is fixed. Thus, we propose to solve the type 2 singularity crossing with the robot prototype instead of the FPR. That mean we are trying to solve the crossing singularity with one drone first. The main approach is to find a feasible singularity configuration satisfying all the constraints of the robot prototype . Or to be more precise, the input force, torques and roll angle expected meet the constraints.

We recall the dynamic model of robot prototype in Eq.3.37

$$\mathbf{J}_{pr}^T \mathbf{f}_{t1} = \mathbf{M}_{pt}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \mathbf{c}_{pt}(\dot{\boldsymbol{\xi}}, \boldsymbol{\xi})$$

When the robot is at the type 2 singularity, $q_2 = 0$ and the Jacobian matrix becomes

$$\mathbf{J}_{pr} = \begin{bmatrix} -Llink \cdot 2sin(q_1) & -Llink \cdot sin(q_1) \\ Llink \cdot 2cos(q_1) & -Llink \cdot cos(q_1) \end{bmatrix} \tag{4.1}$$

From which we can see that $rank(\mathbf{J}_{pr}) = 1$ and $\mathbf{J}_{pr}$ becomes rand-deficient. In the controller, we compute the $\mathbf{f}_{t1}$ through Inverse Dynamic Mode in the way

$$\mathbf{f}_{t1} = \mathbf{J}_{pr}^{(-T)}\mathbf{M}_{pt}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \mathbf{c}_{pt}(\dot{\boldsymbol{\xi}}, \boldsymbol{\xi}) \tag{4.2}$$

in which $\mathbf{J}_{pr}$, however, is not invertible.

Thus, when the robot is in such a singular configuration, the matrix $\mathbf{J}_{pr}$ cannot be inverted, and then the dynamic model degenerates and cannot be solved. Moreover, in the neighbourhood of the singularity, the force $\mathbf{f}_{t1}$ increases as its expression is proportional to the inverse of the determinant of $\mathbf{J}_{pr}$, which is close to zero in the singularity locus.

The main challenges for crossing this kind of singularity for robot prototype are

- to decide the configuration at the singularity such that expected input force not only be finite but also meet the drone limits.
- to find a balance among singularity configuration and other parameters to make whole trajectory meet the constraints.

An initial and final positions are introduced in the joint space in the crossing singularity task and the two positions in the task space are shown in the Fig.4.1 .

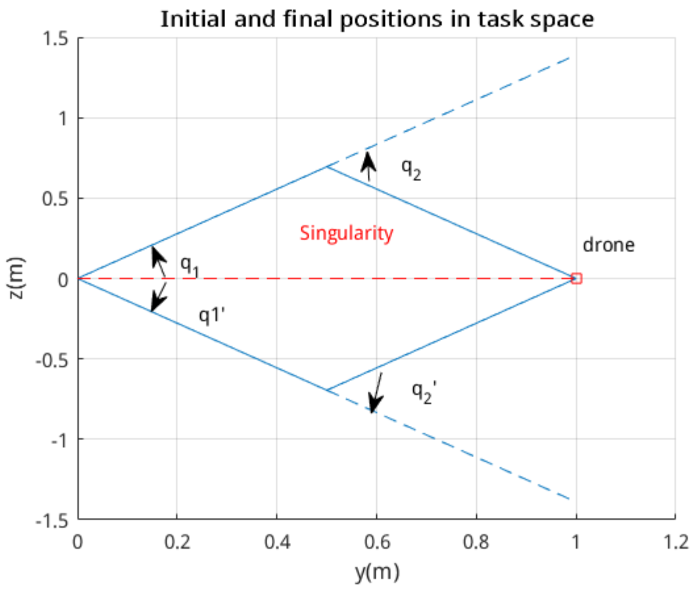


Fig. 4.1. Task presentation in task space to cross singularity

1. initial position $\boldsymbol{\xi}_1 = \begin{bmatrix}-0.9470 & 1.8940\end{bmatrix}$, final position $\boldsymbol{\xi}_2 = \begin{bmatrix}0.9470 & -1.8940\end{bmatrix}$ and singularity position $\boldsymbol{\xi}_s = [10^{-4}, 10^{-4}]$.
2. the whole motion is composed of two parts:
   - $\boldsymbol{\xi}_1 \Rightarrow \boldsymbol{\xi}_s$ during travelling time tf_1

- $\boldsymbol{\xi}_s \Rightarrow \boldsymbol{\xi}_2$ during travelling time tf_2

3. singularity point is defined as $\Xi_s = [\boldsymbol{\xi}_s, \dot{\boldsymbol{\xi}}_s, \ddot{\boldsymbol{\xi}}_s, \boldsymbol{\xi}_s^{(3)}, \boldsymbol{\xi}_s^{(4)}]$.

4.2 Crossing type 2 singularity

4.2.1 1st optimal approach for robot prototype

The first optimal approach is to design a trajectory which satisfies a certain condition such as the Eq.1.26 for 5 bar mechanism, at the singularity. This approach can be summarised as

1. find $\dot{\boldsymbol{\xi}}_s, \ddot{\boldsymbol{\xi}}_s$ to make $\mathbf{f}_{t_1}$ be finite at singularity configuration
2. tune $\mathbf{t}_f = [tf_1, tf_2]$ to generate an initial guess for the trajectory optimisation and the main purpose is to make the trajectory cross singularity only once.
3. optimise the $\boldsymbol{\xi}_s^{(3)}, \boldsymbol{\xi}_s^{(4)}$ (with Via Points) to find a feasible solution to make the whole trajectory meet the constraints.

A vector $\dot{\mathbf{q}}_n \in null(\mathbf{J}_{pr})$ such that

$$\mathbf{J}_{pr}\dot{\mathbf{q}}_n = 0 \tag{4.3}$$

We can obtain one $\dot{\mathbf{q}}_n = \begin{bmatrix}-1 & 2\end{bmatrix}^T$. Then the dynamic model in Eq.3.37 is multiplied by $\dot{\mathbf{q}}_n^T$ and it leads to

$$\dot{\mathbf{q}}_n^T \mathbf{J}_{pr}^T \mathbf{f}_{t1} = \dot{\mathbf{q}}_n^T (\mathbf{M}_{pt}(\boldsymbol{\xi}_s)\ddot{\boldsymbol{\xi}}_s + \mathbf{c}_{pt}(\dot{\boldsymbol{\xi}}_s, \boldsymbol{\xi}_s)) \tag{4.4}$$
$$(\mathbf{J}_{pr}\dot{\mathbf{q}}_n)^T \mathbf{f}_{t1} = \dot{\mathbf{q}}_n^T (\mathbf{M}_{pt}(\boldsymbol{\xi}_s)\ddot{\boldsymbol{\xi}}_s + \mathbf{c}_{pt}(\dot{\boldsymbol{\xi}}_s, \boldsymbol{\xi}_s)) \tag{4.5}$$
$$0 = \dot{\mathbf{q}}_n^T (\mathbf{M}_{pt}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}}_s + \mathbf{c}_{pt}(\dot{\boldsymbol{\xi}}_s, \boldsymbol{\xi}_s)) \tag{4.6}$$

The input force will not be infinite, if $\mathbf{q}_s, \dot{\mathbf{q}}_s, \ddot{\mathbf{q}}_s$ meet the criterion

$$\begin{bmatrix}-1 & 2\end{bmatrix} (\mathbf{M}_{pt}(\boldsymbol{\xi}_s)\ddot{\boldsymbol{\xi}}_s + \mathbf{c}_{pt}(\dot{\boldsymbol{\xi}}_s, \boldsymbol{\xi}_s)) = 0 \tag{4.7}$$

To meet the criterion in Eq.4.7, we set

$$\dot{\boldsymbol{\xi}}_s = [-1, 2] \tag{4.8}$$
$$\ddot{\boldsymbol{\xi}}_s = [-10.7243, 24.6296] \tag{4.9}$$

With the dynamic model in Eq.3.37 we can compute the expected input force at the singularity configuration

$$0 \le \mathfrak{f}_1 \le F_{max} = 36N \tag{4.10}$$

Then, travelling time for each segment influences the path of trajectory. With some values, the trajectory of $\boldsymbol{\xi}(t)$ would cross singularity more than once.

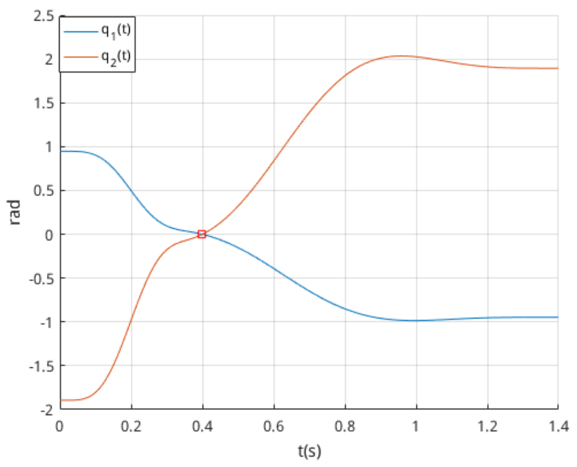


Fig. 4.2. Initial guess of trajectory

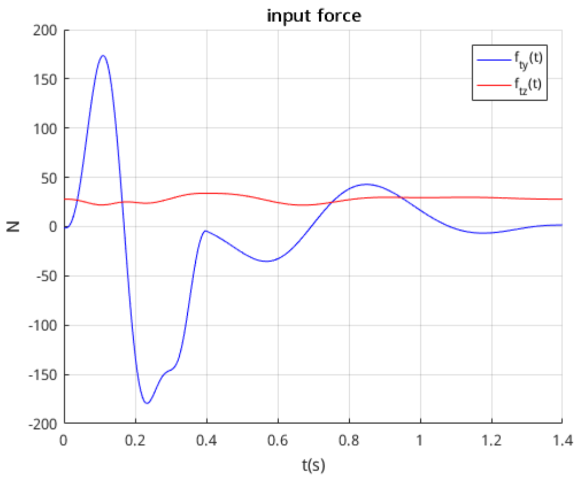


Fig. 4.3. Input force of initial guess trajectory

An initial guess is made with $\mathbf{t}_f = [0.4, 1.5]$:

From Fig.4.2 and Fig.4.3, we can see that the robot crosses the singularity at 0.5s and the input force at that singularity configuration is finite. However, the input force is beyond the limits during the robot motion at some point.

One optimisation approach is conducted with its optimisation formation being:

Decision variables:

- the 1st step optimisation is to find proper $\boldsymbol{\xi}_s^{(3)} and \boldsymbol{\xi}_s^{(4)}$ such that trajectory linking $\boldsymbol{\xi}_1, \boldsymbol{\xi}_s, \boldsymbol{\xi}_2$ is optimal.

- the 2nd step optimisation is conducted for each segment of the whole trajectory with decision varaibles being the via points, i.e. Ξ_{v1}, Ξ_{v2}.

where Ξ_{v1}, Ξ_{v2} means the via points in the first segment and the second segment. $\Xi_1 \Rightarrow \Xi_s$ is the first segment and $\Xi_s \Rightarrow \Xi_2$ is the second segment.

Objective function:

$$J = \alpha \cdot \frac{J_e}{J_{e0}} + (1 - \alpha) \cdot \frac{J_\phi}{J_{\phi 0}}$$

where J_{e0} and $J_{\phi 0}$ are the energy and perturbation index when $\boldsymbol{\xi}_s^{(3)} = \boldsymbol{\xi}_s^{(4)} = 0$ and $\alpha = 0.5$

Constraints:

1. The drone should not touch the ground or the fixed point.
$$\begin{aligned} z_{j2} &\geq -Llink + m_{gd} \\ z_{j3} &\geq -Llink + m_{gd} \\ y_{j2} &\geq m_{gd} \\ y_{j3} &\geq m_{gd} \\ |q_1| &\leq \frac{\pi}{2} - m_{ga} \end{aligned}$$
2. The drone should not roll over.
$$|\phi_1| \leq \frac{\pi}{2} - m_{ga}$$
3. The limits of the four rotors of the drone are
$$0 \leq \varpi_{ij} \leq \varpi_{max}$$
where ϖ_{ij} is the angular speed of the rotor j of the drone i. ϖ_{max} is the limit of the rotor and $\varpi_{max} = 900$.

However, the optimisation with one via point failed. The last failed approach in the optimisation case is shown in Fig.4.4 and Fig.4.5.

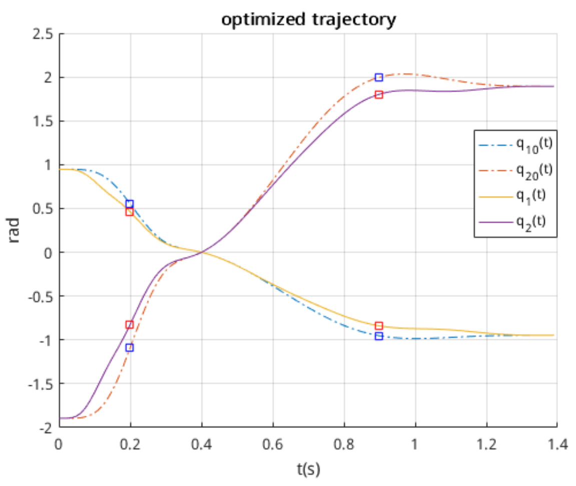


Fig. 4.4. Trajectory in a failed approach

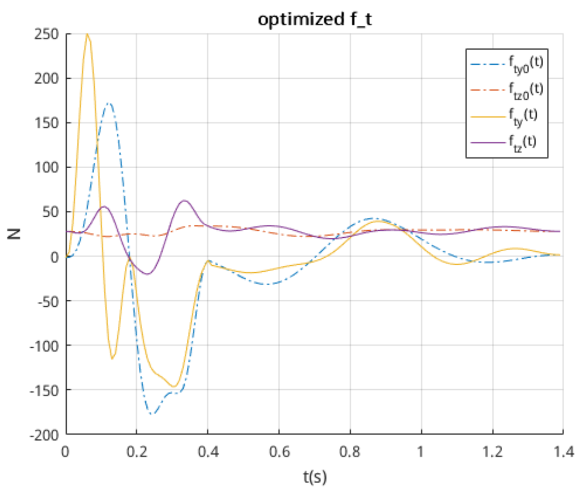


Fig. 4.5. Input force in a failed approach

From the results above, we can see that even with the help of optimisation, the input force required is still beyond the limit of the drone.

4.2.2 2nd optimal approach for robot prototype

The previous approach is able to guarantee that the input force is not infinite but can not specify the exact value of the input force. In contrast, the 2nd approach aims to obtain $\boldsymbol{\xi}_s, \dot{\boldsymbol{\xi}}_s, \ddot{\boldsymbol{\xi}}_s$ by setting the input force.

Instead of setting the input force $\mathbf{f}_{t1}$, we attempt to decide the force applied by the drone to the chain $\mathbf{f}_{p1}$. We can get the direct dynamic model of the passive chain from Eq.2.6 that

$$\ddot{\boldsymbol{\xi}}_s = \mathbf{M}_{pp}^{-1}(\boldsymbol{\xi}_s)(\mathbf{J}_{pr}\mathbf{f}_{p1} - \mathbf{c}_{pr}(\boldsymbol{\xi}_s, \dot{\boldsymbol{\xi}}_s)) \tag{4.11}$$

With

$$\dot{\boldsymbol{\xi}}_s = [-3, 3]^T \tag{4.12}$$

$$\mathbf{f}_{p1} = [0, 5]^T \tag{4.13}$$

We obtain

$$\ddot{\boldsymbol{\xi}}_s = [-12.64, 18.58]^T \tag{4.14}$$

and

$$\mathbf{f}_{t1} = [-6.4, 11.01]^T \tag{4.15}$$

$$|\mathbf{f}_{t1}| \leq F_{max} \tag{4.16}$$

Then, an initial trajectory guess is generated with $\mathbf{t}_f = [1.25, 1.2]$

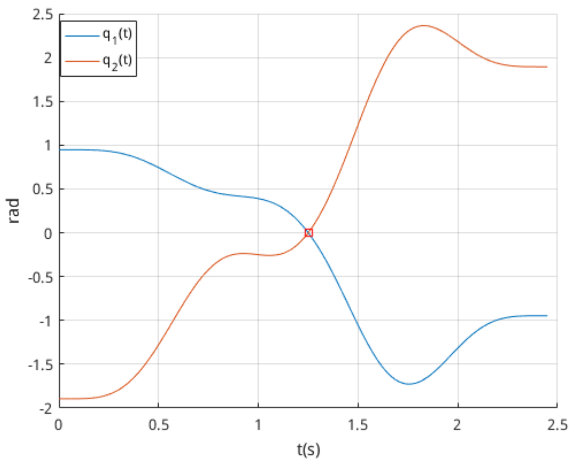


Fig. 4.6. Initial guess for a trajectory

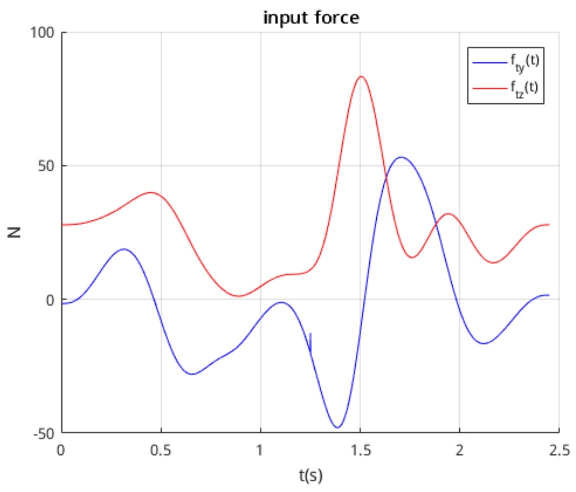


Fig. 4.7. Input force of the initial guess trajectory

Decision variables:

- the 1st step optimisation is to find proper $\boldsymbol{\xi}_s^{(3)}$ and $\boldsymbol{\xi}_s^{(4)}$ such that trajectory linking $\boldsymbol{\xi}_1, \boldsymbol{\xi}_s, \boldsymbol{\xi}_2$ is optimal.
- the 2nd step optimisation is conducted for each segment of the whole trajectory with decision varaibles being the via points, i.e. $\boldsymbol{\Xi}_{v1}, \boldsymbol{\Xi}_{v2}$.

where $\boldsymbol{\Xi}_{v1}, \boldsymbol{\Xi}_{v2}$ means the via points, in the first segment $\boldsymbol{\Xi}_1 \Rightarrow \boldsymbol{\Xi}_s$ and the second segment $\boldsymbol{\Xi}_s \Rightarrow \boldsymbol{\Xi}_2$ respectively.

Objective function:

$$J = \alpha \cdot \frac{J_e}{J_{e0}} + (1 - \alpha) \cdot \frac{J_\phi}{J_{\phi 0}}$$

where J_{e0} and $J_{\phi 0}$ are the energy and perturbation index when $\boldsymbol{\xi}_s^{(3)} = \boldsymbol{\xi}_s^{(4)} = 0$ and $\alpha = 0.5$

Constraints:

1. The drone should not touch the ground or the fixed point.
$$\begin{aligned} z_{j2} &\geq -Llink + m_{gd} \\ z_{j3} &\geq -Llink + m_{gd} \\ y_{j2} &\geq m_{gd} \\ y_{j3} &\geq m_{gd} \\ |q_1| &\leq \frac{\pi}{2} - m_{ga} \end{aligned}$$
2. The drone should not roll over.
$$|\phi_1| \leq \frac{\pi}{2} - m_{ga}$$
3. The limits of the four rotors of the drone are
$$0 \leq \varpi_{ij} \leq \varpi_{max}$$
where ϖ_{ij} is the angular speed of the rotor j of the drone i. ϖ_{max} is the limit of the rotor and $\varpi_{max} = 900$.

Unfortunately, this optimisation approach fails and the last case in the process is shown in Fig.4.8 and Fig.4.9.

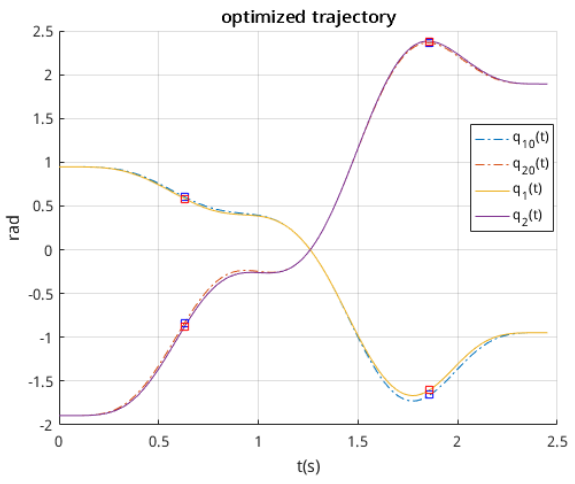


Fig. 4.8. Trajectory in a failed approach

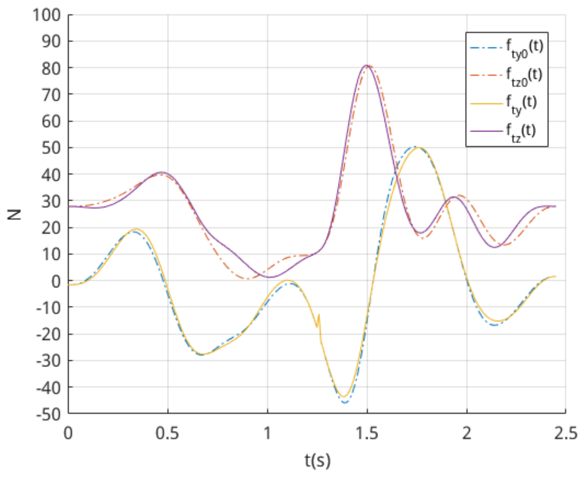


Fig. 4.9. Input force in a failed approach

There are two things we can notice from the Fig.4.8 and Fig.4.9:

- the expected input force is still large than the limit for instance at $0.5s$
- a sudden change or discontinuity happens at crossing singularity both in the before and after optimisation cases.

4.2.3 Summary

Crossing type 2 singularity for FPR or even FRP prototype is a challenge different from typical parallel robots.

Typical approaches with optimisation can make the input force be finite or specified, however, the input force is still beyond the limit somewhere between the initial point and the singularity point, or somewhere else.

The main problems can be summarised as

- trajectory depends not only on the configuration at the singularity point, but also the travelling time. And thus we realised that it is hard to tune both at the same time.
- a good initial guess is necessary to begin an optimisation.
- input force can be provided by a drone limits the mobility.

Conclusion and future work

The main topic of this thesis was to develop a motion optimisation planning for a new aerial robot FPR: a method for motion optimisation and approaches to Type 2 singularity.

FPR is a new robot combining a passive chain and two drones. It can be considered as a parallel robot whose two actuators are drones. Three main challenges are addressed in this thesis: (1) no motion planning has been done for this robot. (2) dynamics of drones, for instance under-actuation, must be taken into account in the motion planning. (3) required input force for crossing Type 2 singularity are beyond the limits of drones.

Chapter 1 states present research work and gives a general description of drones, its advantages and applications with other robots at the beginning. A overview of drones application in the robotics domain are stated: a flying grasper, a cable-suspended transportation and an aerial manipulation with a tool. An example in each domain is introduced, such as a high speed and avian-inspired grasping quadrotor, FlyCrane and an aerial vehicle with dual multi-degree of freedom manipulators. The on-board system and the state of art about the trajectory generation for a single drone is presented. Only the methods which having been tested in the real experiments are chosen. Thus, we present two methods: minimisation derivatives of the position trajectory and optimal control approach and their models in details. Crossing type 2 singularity is one of the task for FPR. The dynamic model of 5 bar mechanism is stated as an example of parallel robots. This is followed by the state of art of type 2 singularity crossing. Finally, the conception of FPR and its potential research interest are introduced.

Chapter 2 first demonstrate developments of FPR dynamic model done by SIX Damien et al. The whole model is composed of two parts: a passive chain part and an attitude dynamic model part. CTC are applied into controller of the passive chain part and attitude dynamic part. A coupling terms associated to the drones angular velocities and accelerations is treated as a perturbation in the closed- loop equation of the passive structure control. Simulations showed the performance and the robustness of the controller designed.

Chapter 3 shows one main contributions of this master thesis. To begin with, the vector for motion planning is defined from the configuration of the passive chain. Trajectory generation is conducted independently for each configuration sate. Considering the rotor dynamics of a drone, a 9-degree polynomial is used to generate a trajectory to guarantee the continuity of the rotor angular speed. A off-line planning optimisation is proposed to improve the robot motion. Way points and via points are defined as decision variables. The objective function includes an energy consumption term and a perturbation term in order to minimise the energy cost of robot during the motion and the perturbation in the controller. Investigation about two parameters in the optimisation, number of via points and weights in the objective function, are carried after that. Simulation in Matlab validates the motion from optimisation. A prototype robot is built in the lab of LS2N. We formulate the optimisation methods for this prototype and conduct simulation and real experiments to test the optimisation results.

Chapter 4 presents two approaches for motion planning based on the optimisation to cross Type 2 singularity. The degenerating in the dynamic model of FPR prototype, Jacobian matrix, is first analysed to understand the type 2 singularity of this robot. Based on the condition of FPR crossing type 2 singularity, an optimisation with the decision variables being the 3rd, 4th derivatives of singularity point and two via points. Even though we can guarantee the input force at the singularity is finite, but the input force in the motion are beyond the limits of robot with the help of introducing via points. Then, we specify the exact input force at the singularity in the 2nd approach. Optimisation with the 3rd, 4th derivatives of singularity point and two via points are considered, but still no feasible solution is found. The main challenge is that the thrust force of the drone can not afford the expected input force of the robot during crossing the singularity.

Some research can be done as continuation of motion planning for this robot:

1. optimisation algorithm can be simplified and optimised for less computation complexity such that this algorithm can perform on-line motion planning.
2. an feasible trajectory crossing singularity with the optimisation method proposed should be investigated continuously.

Appendix

Comparison between the 2-step optimisation and the original optimisation for FPR

An example is presented to show that 2-step optimisation is more effective than the original optimisation method.

The mission is to start at the Beginning/Ending position, pass the way position 1, 2, 3 and finally come back to the Beginning/Ending position shown in the 4.10. The travelling time for each segment trajectory is $3s$.

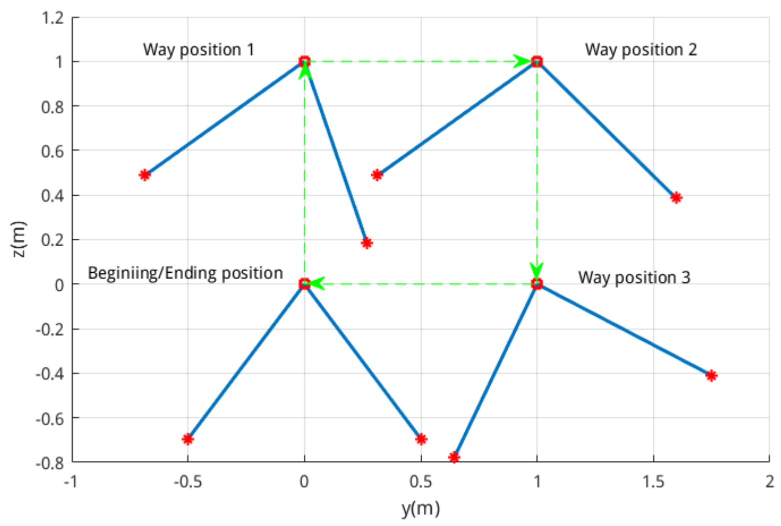


Fig. 4.10. Way positions in the task space

The trajecoties of the original optimisation method and 2-step optimisation are shown in the Fig.4.12 and 4.12.

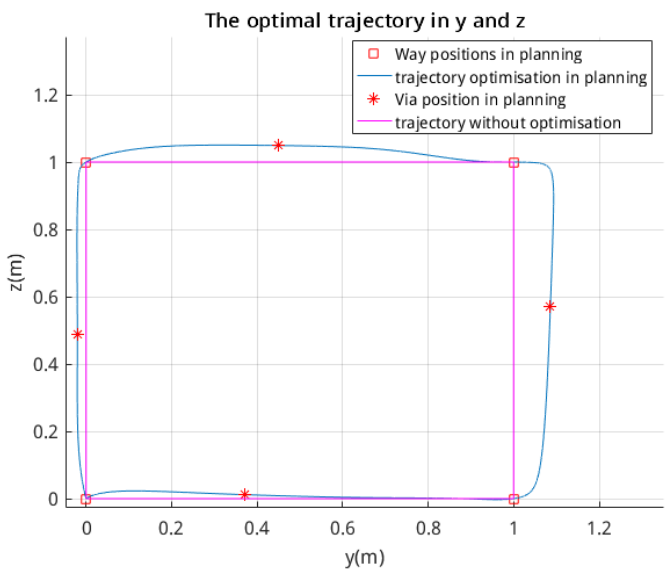


Fig. 4.11. The trajectory of the original optimisation method in the task space

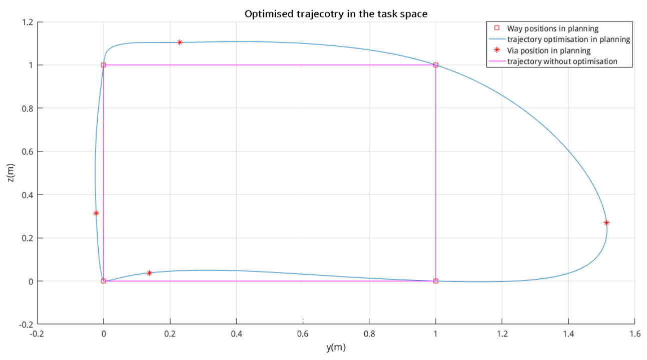


Fig. 4.12. An optimised trajectory of the end-effector in the task space

Their trajectories of each coordinate are presented by Fig.4.13 and Fig.4.14.

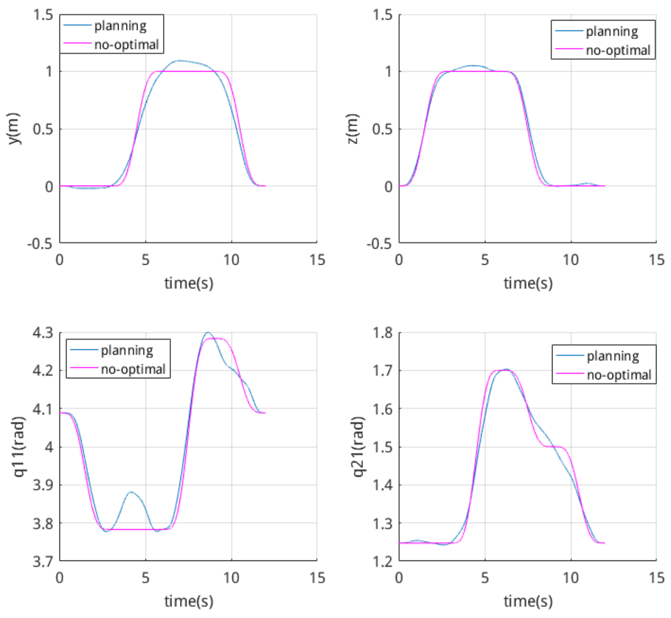


Fig. 4.13. The trajectory of each coordinate of the original optimisation method

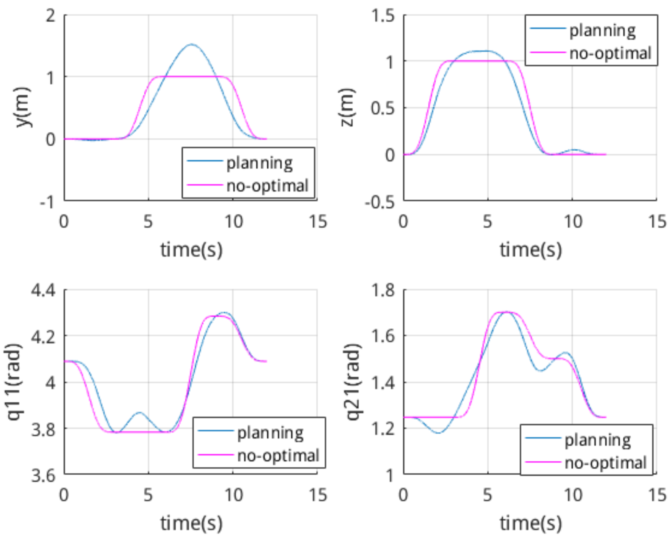


Fig. 4.14. The trajectory of each coordinate of the 2-step optimisation method

As we can see from Fig.4.13, Fig.4.14, Fig.4.13 and Fig.4.14, the 2-step optimisation changes the trajectory shape more than the original method.

Their J_e and J_ϕ are shown in Fig.4.15, Fig.4.16, Fig.4.18 and Fig.4.18.

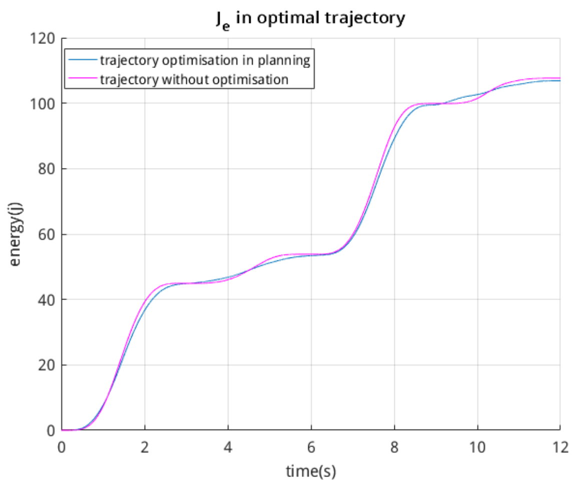


Fig. 4.15. Energy consumption in the original method

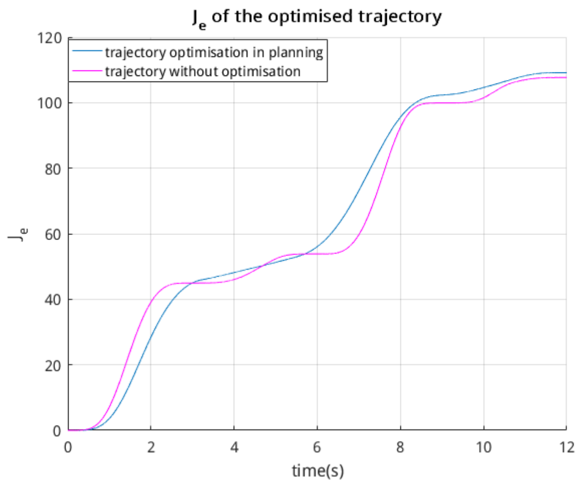


Fig. 4.16. Energy consumption in the 2-step method

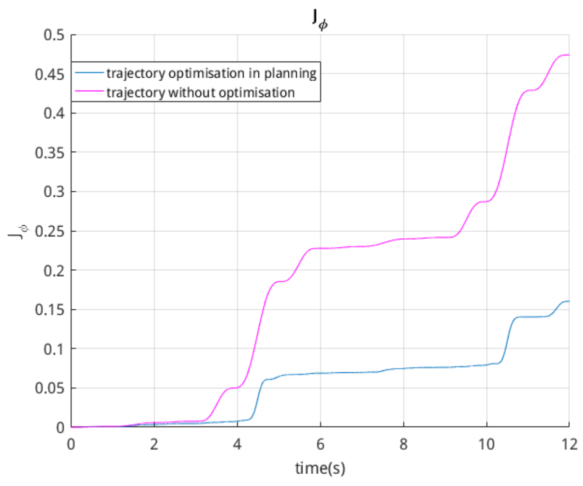


Fig. 4.17. Perturbation index in the original method

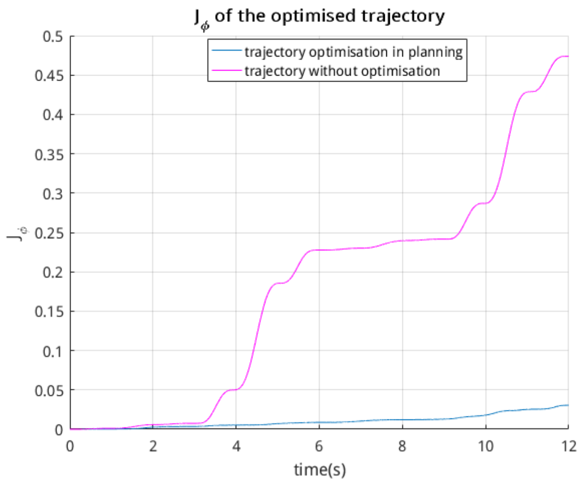


Fig. 4.18. Perturbation index in the 2-step method

Comparing the J_ϕ in Fig.4.17 and Fig.4.17, we can see that 2-step optimisation can obtain a smaller J_e nearly one third of the that of original method. However, the advantage of the 2-step optimisation is not obvious form the perspective of J_e.

Also, it takes the original optimisation programs nearly 40 minutes to finish the optimisation process and find a feasible solution, while the 2-step optimisation just needs around 15 minutes.

Examples in the real experiments

The task is set as the same in the section 3.3.2 as the Fig.3.36.

When the travelling time is $\begin{bmatrix} 5s & 5s & 5s & 5s & 5s & 5s & 5s \end{bmatrix}^T$, the results in the real experiment are shown in Fig.4.19, Fig.4.20,

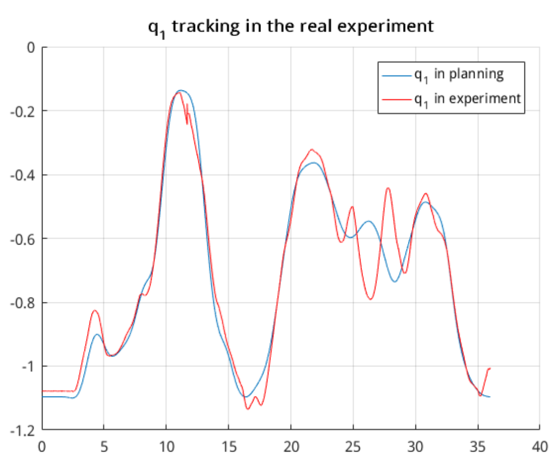


Fig. 4.19. Joint 1 tracking in the real experiment

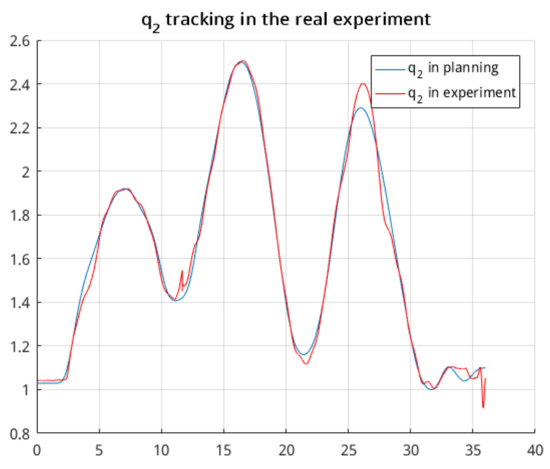


Fig. 4.20. Joint 2 tracking in the real experiment

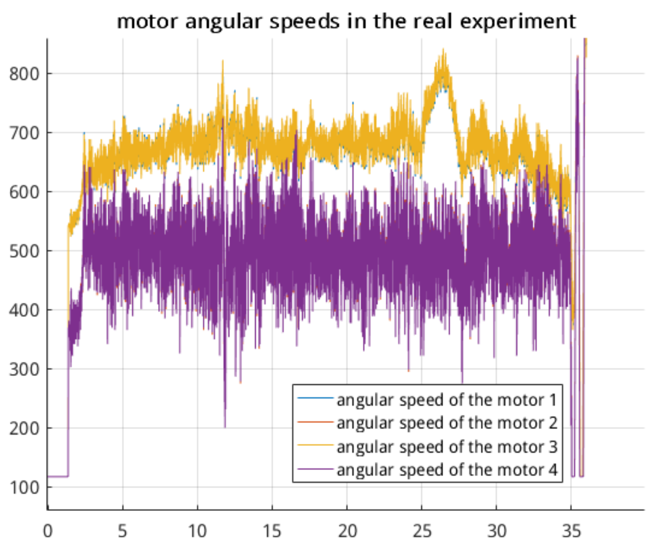


Fig. 4.21. motor angular speeds in the real experiment

When the travelling time is $\begin{bmatrix} 5s & 3s & 3s & 3s & 3s & 3s & 5s \end{bmatrix}^T$, the results in the real experiment are shown in Fig.4.22, Fig.4.23.

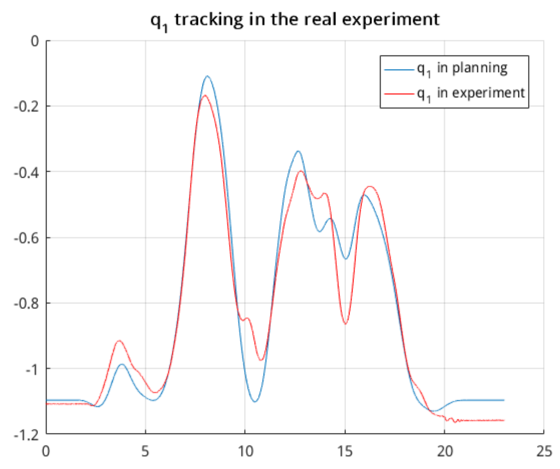


Fig. 4.22. Joint 1 tracking in the real experiment

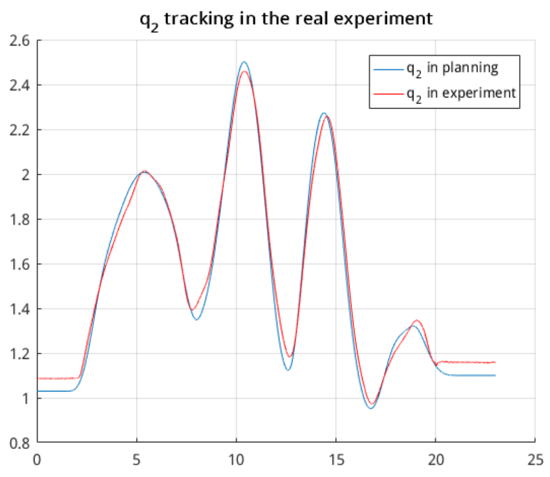


Fig. 4.23. Joint 2 tracking in the real experiment

Technical Annex

Motion planning and optimisation is validated with simulation and the real experiment on the FPR prototype robot.

Three main programs in Matlab are contributed in this master thesis.

1. Motion optimisation program for FPR prototype robot.
2. Motion optimisation program for FRR.
3. Singularity crossing optimisation approaches for FPR prototype robot.

All these programs have been normalised to be user-friendly. In such a case, beginners of this project will find it easy to use these codes to repeat the results in this thesis.

Main structure of programs

The structure and functions of the whole program of *Motion optimisation program for FPR prototype robot* can be presented in Fig.4.24

Run_optimisation Main program.

InterFace() take inputs from users

Optimisation4WayPoints() optimise the trajectory with way points

Optimisation4Segment()() optimise each segment of the trajectory by via points

ReConstuctTrajectory()/TimeSericesTrajectory() generate trajectory for simulation

call simulation call the simulation in Simulink

PlotSimulationResults() show results

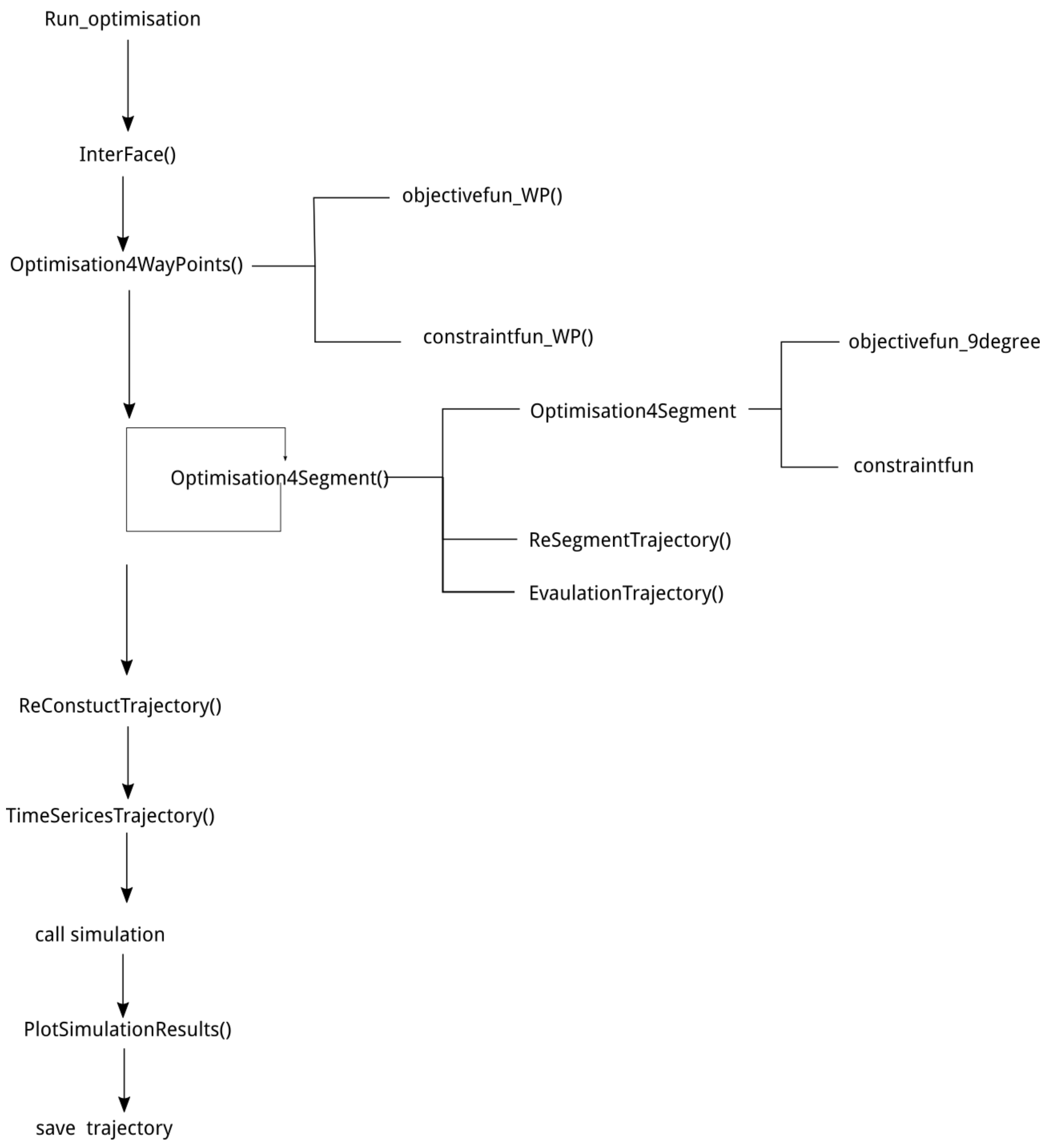
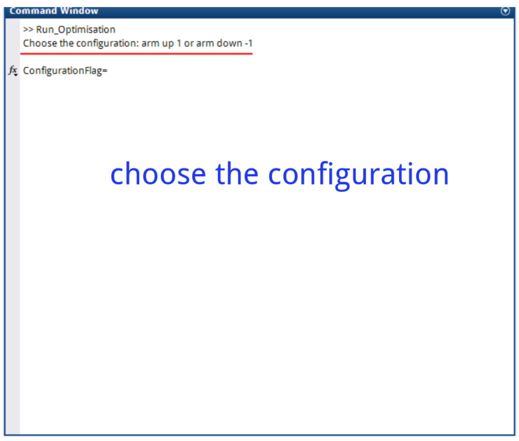


Fig. 4.24. Code structure of FPR prototype motion planning

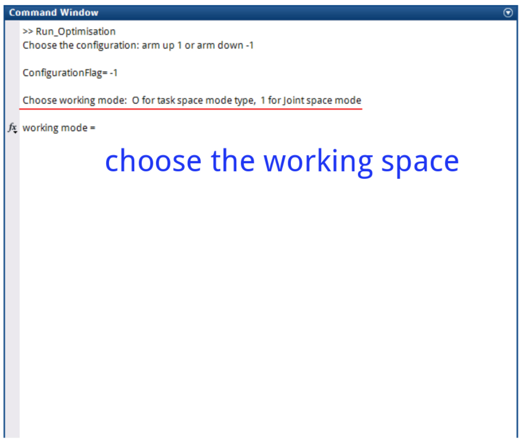
How to use a program to generate a optimal trajectory

1. Run *Run_Optimisation.m* in the path
2. Decide to way points in task space or in joint space
3. Input the way positions that the drone(prototype) or FPR end-effector will pass
4. Choose travelling time and number of via points for each segment
5. Show way positions in task space



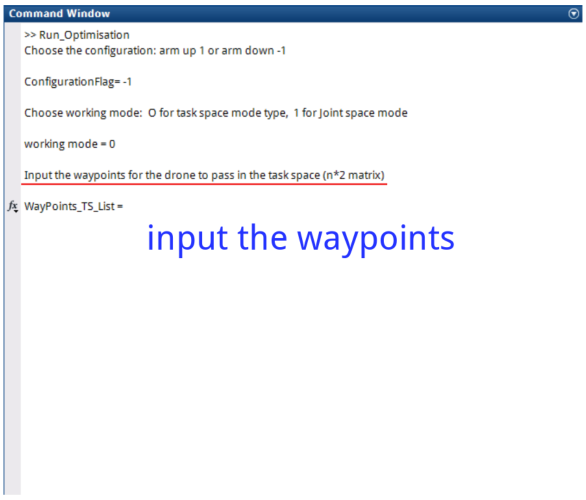
```
Command Window
>> Run_Optimisation
Choose the configuration: arm up 1 or arm down -1

fx ConfigurationFlag=
```

Fig. 4.25. Choose configuration



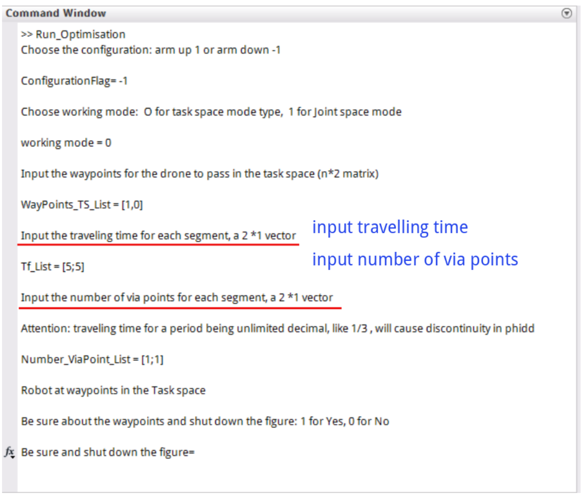
```
Command Window
>> Run_Optimisation
Choose the configuration: arm up 1 or arm down -1

ConfigurationFlag= -1

Choose working mode:  0 for task space mode type,  1 for Joint space mode

fx working mode =
```

Fig. 4.26. Choose working space



```
Command Window
>> Run_Optimisation
Choose the configuration: arm up 1 or arm down -1

ConfigurationFlag= -1

Choose working mode:  0 for task space mode type,  1 for Joint space mode

working mode = 0

Input the waypoints for the drone to pass in the task space (n*2 matrix)

fx WayPoints_TS_List =
```

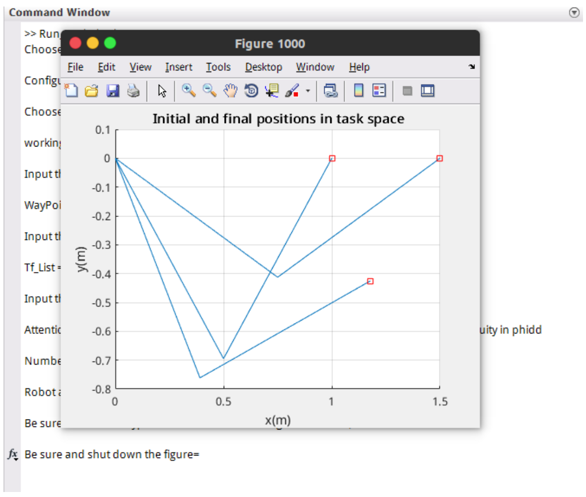
Fig. 4.27. Choose the way positions



```
Command Window
>> Run_Optimisation
Choose the configuration: arm up 1 or arm down -1

ConfigurationFlag= -1

Choose working mode:  0 for task space mode type,  1 for Joint space mode

working mode = 0

Input the waypoints for the drone to pass in the task space (n*2 matrix)

WayPoints_TS_List = [1,0]

Input the traveling time for each segment, a 2 *1 vector

Tf_List = [5;5]

Input the number of via points for each segment, a 2 *1 vector

Attention: traveling time for a period being unlimited decimal, like 1/3 , will cause discontinuity in phidd

Number_ViaPoint_List = [1;1]

Robot at waypoints in the Task space

Be sure about the waypoints and shut down the figure: 1 for Yes, 0 for No

fx Be sure and shut down the figure=
```

Fig. 4.28. Choose $\mathbf{t}_f$ and n_{vp}

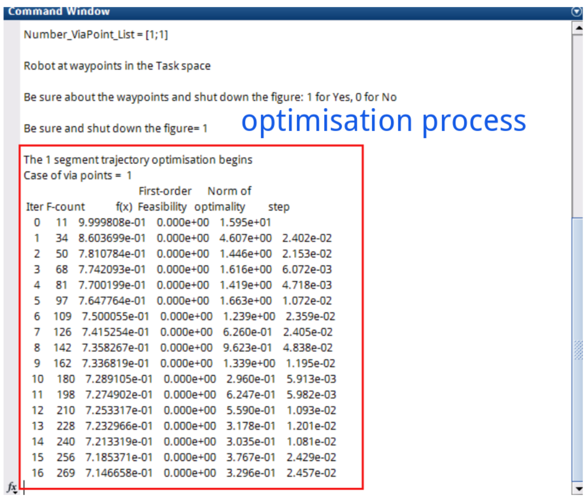
6. Begin the optimisation



```
Command Window
>> Run
Choos
Config
Choose
workin
Input t
WayPoi
Input t
Tf_List
Input t
Attentic                                                          uity in phidd
Numbe
Robot a
Be sure
fx Be sure and shut down the figure=
```

Figure 1000
File Edit View Insert Tools Desktop Window Help
Initial and final positions in task space
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
y(m)
0
0.5
1
1.5
x(m)

Fig. 4.29. Choose the way positions

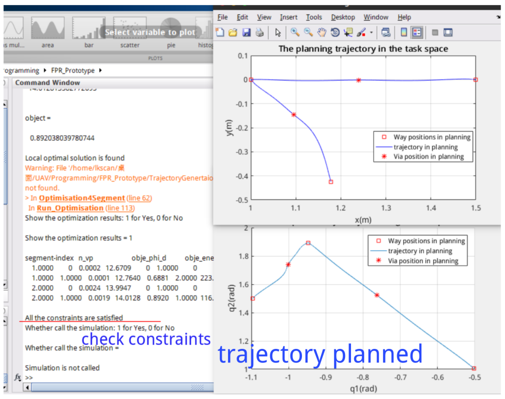


```
Command Window
Number_ViaPoint_List = [1;1]

Robot at waypoints in the Task space

Be sure about the waypoints and shut down the figure: 1 for Yes, 0 for No

Be sure and shut down the figure= 1

The 1 segment trajectory optimisation begins
Case of via points =  1
                                  First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0      11    9.999808e-01    0.000e+00    1.595e+01
    1      34    8.603699e-01    0.000e+00    4.607e+00    2.402e-02
    2      50    7.810784e-01    0.000e+00    1.446e+00    2.153e-02
    3      68    7.742093e-01    0.000e+00    1.616e+00    6.072e-03
    4      81    7.700199e-01    0.000e+00    1.419e+00    4.718e-03
    5      97    7.647764e-01    0.000e+00    1.663e+00    1.072e-02
    6     109    7.500055e-01    0.000e+00    1.239e+00    2.359e-02
    7     126    7.415254e-01    0.000e+00    6.260e-01    2.405e-02
    8     142    7.358267e-01    0.000e+00    9.623e-01    4.838e-02
    9     162    7.336819e-01    0.000e+00    1.339e+00    1.195e-02
   10     180    7.289105e-01    0.000e+00    2.960e-01    5.913e-03
   11     198    7.274902e-01    0.000e+00    6.247e-01    5.982e-03
   12     210    7.253317e-01    0.000e+00    5.590e-01    1.093e-02
   13     228    7.232966e-01    0.000e+00    3.178e-01    1.201e-02
   14     240    7.213319e-01    0.000e+00    3.035e-01    1.081e-02
   15     256    7.185371e-01    0.000e+00    3.767e-01    2.429e-02
   16     269    7.146658e-01    0.000e+00    3.296e-01    2.457e-02
fx
```

Fig. 4.30. Input travelling time and number of via points

7. Call the simulation in Simulink

8. Show results



```
object =

  0.892038039780744

Local optimal solution is found
Warning: File '/home/lkscan/桌
面/UAV/Programming/FPR_Prototype/TrajectoryGenertaio
not found.
> In Optimisation4Segment (line 62)
  In Run_Optimisation (line 113)
Show the optimization results: 1 for Yes, 0 for No

Show the optimization results = 1

segment-index  n_vp           obje_phi_d      obje_ene
   1.0000       0   0.0002  12.6709      0   1.0000      0
   1.0000  1.0000   0.0001  12.7640  0.6881   2.0000  223.
   2.0000       0   0.0024  13.9947      0   1.0000      0
   2.0000  1.0000   0.0019  14.0128  0.8920   1.0000  116.

All the constraints are satisfied
Whether call the simulation: 1 for Yes, 0 for No

Whether call the simulation =

Simulation is not called
>>
```

Fig. 4.31. Show the optimised trajectory

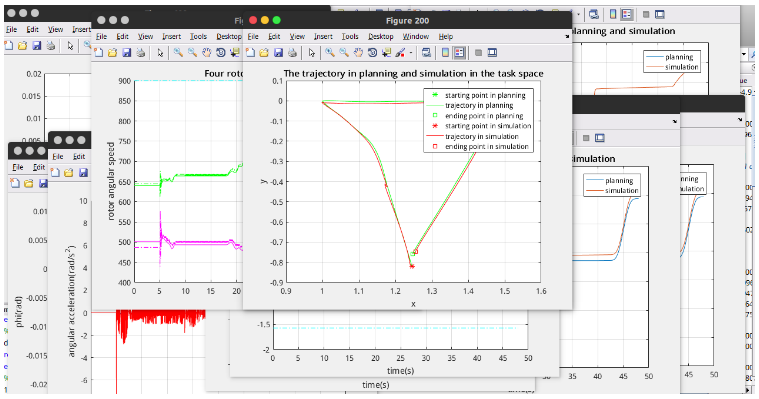
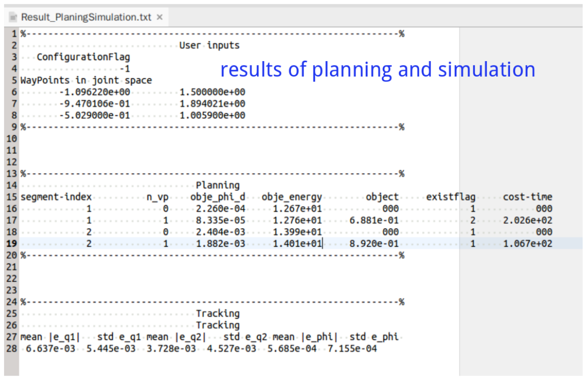


Figure 200
The trajectory in planning and simulation in the task space
starting point in planning
trajectory in planning
ending point in planning
starting point in simulation
trajectory in simulation
ending point in simulation
Four roto
rotor angular speed
angular acceleration(rad/s²)
phi(rad)
planning and simulation
planning
simulation
time(s)
x
y

Fig. 4.32. Call the simulation

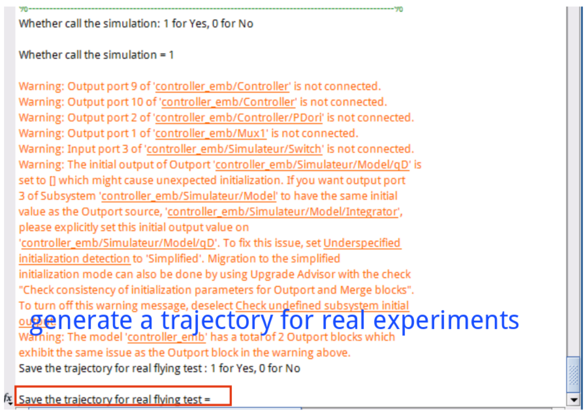
9. Results are recorded and saved in txt

10. Generate trajectory data file for real expriment


Result_PlaningSimulation.txt
results of planning and simulation

```
%-----------------------------------------------------------------------%
                                  User inputs
   ConfigurationFlag
                  -1
WayPoints in joint space
        -1.096220e+00        1.500000e+00
        -9.470106e-01        1.894021e+00
        -5.029000e-01        1.005900e+00
%-----------------------------------------------------------------------%



%-----------------------------------------------------------------------%
                                     Planning
segment-index          n_vp    obje_phi_d   obje_energy       object     existflag     cost-time
            1             0     2.260e-04     1.267e+01          000             1           000
            1             1     8.335e-05     1.276e+01    6.881e-01             2     2.026e+02
            2             0     2.404e-03     1.399e+01          000             1           000
            2             1     1.882e-03     1.401e+01    8.920e-01             1     1.067e+02
%-----------------------------------------------------------------------%



%-----------------------------------------------------------------------%
                                     Tracking
                                     Tracking
mean |e_q1|   std e_q1 mean |e_q2|   std e_q2 mean |e_phi|  std e_phi
 6.637e-03  5.445e-03  3.728e-03  4.527e-03  5.685e-04  7.155e-04
```

Fig. 4.33. Save optimisation results



```
Whether call the simulation: 1 for Yes, 0 for No

Whether call the simulation = 1

Warning: Output port 9 of 'controller_emb/Controller' is not connected.
Warning: Output port 10 of 'controller_emb/Controller' is not connected.
Warning: Output port 2 of 'controller_emb/Controller/PDori' is not connected.
Warning: Output port 1 of 'controller_emb/Mux1' is not connected.
Warning: Input port 3 of 'controller_emb/Simulateur/Switch' is not connected.
Warning: The initial output of Outport 'controller_emb/Simulateur/Model/qD' is
set to [] which might cause unexpected initialization. If you want output port
3 of Subsystem 'controller_emb/Simulateur/Model' to have the same initial
value as the Outport source, 'controller_emb/Simulateur/Model/Integrator',
please explicitly set this initial output value on
'controller_emb/Simulateur/Model/qD'. To fix this issue, set Underspecified
initialization detection to 'Simplified'. Migration to the simplified
initialization mode can also be done by using Upgrade Advisor with the check
"Check consistency of initialization parameters for Outport and Merge blocks".
To turn off this warning message, deselect Check undefined subsystem initial
ou[illegible]
Warning: The model 'controller_emb' has a total of 2 Outport blocks which
exhibit the same issue as the Outport block in the warning above.
Save the trajectory for real flying test : 1 for Yes, 0 for No

Save the trajectory for real flying test =
```

Fig. 4.34. Generate trajectory for real experiment

Bibliography

[1] D. S. Bernstein, *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory.* Princeton University Press Princeton, 2005, vol. 41.

[2] D. P. Bertsekas, *Dynamic programming and optimal control.* Athena scientific Belmont, MA, 1995, vol. 1, no. 2.

[3] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *Control and Automation, 2008 16th Mediterranean Conference on.* IEEE, 2008, pp. 1258–1263.

[4] S. Briot and V. Arakelian, "Optimal force generation in parallel manipulators for passing through the singular positions," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 967–983, 2008.

[5] S. Briot, W. Khalil *et al.*, *Dynamics of Parallel Robots.* Springer, 2015.

[6] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor uav," in *Control Conference (ECC), 2007 European.* IEEE, 2007, pp. 4001–4008.

[7] N. Dadkhah and B. Mettler, "Survey of motion planning literature in the presence of uncertainty: considerations for uav guidance," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 233–246, 2012.

[8] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, pp. 65–100, 2010.

[9] G. Gogu, *Structural synthesis of parallel robots.* Springer, 2008.

[10] C. Gosselin and J. Angeles, "Singularity analysis of closed-loop kinematic chains," *IEEE transactions on robotics and automation*, vol. 6, no. 3, pp. 281–290, 1990.

[11] R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A survey of the maximum principles for optimal control problems with state constraints," *SIAM review*, vol. 37, no. 2, pp. 181–218, 1995.

[12] M. Hehn and R. D Andrea, "Real-time trajectory generation for interception maneuvers with quadrocopters," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4979–4984.

[13] M. Hehn and R. DAndrea, "Quadrocopter trajectory generation and control," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1485–1491, 2011.

[14] M. Hehn and R. D'Andrea, "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, 2015.

[15] R. B. Hill, D. Six, A. Chriette, S. Briot, and P. Martinet, "Crossing type 2 singularities of parallel robots without pre-planned trajectory with a virtual-constraint-based controller," in *2017 IEEE International Conference on Robotics and Automation (ICRA 2017)*, 2017.

[16] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, pp. 1–14.

[17] H.-M. Huang, "Autonomy levels for unmanned systems (alfus) framework: safety and application issues," in *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*. ACM, 2007, pp. 48–53.

[18] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[19] S. Kim, K. Sreenath, S. Bhattacharya, and V. Kumar, "Optimal trajectory generation under homology class constraints," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 3157–3164.

[20] C. Korpela, M. Orsag, M. Pekala, and P. Oh, "Dynamic stability of a mobile manipulating unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[21] S. LaValle and R. Sharma, "A framework for motion planning in stochastic environments: Applications and computational issues," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3. IEEE, 1995, pp. 3063–3068.

[22] S. M. LaValle and R. Sharma, "A framework for motion planning in stochastic environments: modeling and analysis," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3. IEEE, 1995, pp. 3057–3062.

[23] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[24] M. Manubens, D. Devaurs, L. Ros, and J. Cortés, "Motion planning for 6-d manipulation with aerial towed-cable systems," in *Robotics: Science and Systems (RSS)*, 2013, p. 8p.

[25] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2520–2525.

[26] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on.* IEEE, 2012, pp. 477–483.

[27] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 2013, pp. 3480–3486.

[28] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles.* Springer Science & Business Media, 2010.

[29] N. S. Özbek, M. Önkol, and M. Ö. Efe, "Feedback control strategies for quadrotor-type aerial robots: a survey," *Transactions of the Institute of Measurement and Control*, vol. 38, no. 5, pp. 529–554, 2016.

[30] G. Pagis, N. Bouton, S. Briot, and P. Martinet, "Enlarging parallel robot workspace through type-2 singularity crossing," *Control Engineering Practice*, vol. 39, pp. 1–11, 2015.

[31] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research.* Springer, 2016, pp. 649–666.

[32] D. Six, S. Briot, A. Chriette, and P. Martinet, "A controller for avoiding dynamic model degeneracy of parallel robots during type 2 singularity crossing," in *New Trends in Mechanism and Machine Science.* Springer, 2017, pp. 589–597.

[33] D. Six, S. Briot, A. Chriette, and P. Martinet, "Dynamic modeling and trajectory tracking controller of a novel flying parallel robot," in *20th IFAC World Congress*, 2017.

[34] S. Tang and V. Kumar, "Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload," in *2015 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2015, pp. 2216–2222.

[35] J. Thomas, J. Polin, K. Sreenath, and V. Kumar, "Avian-inspired grasping for quadrotor micro uavs," in *ASME International Design Engineering Technical Conference (IDETC)*, 2013.