University of Nevada, Reno

Aerial Robotic Chain: Modeling, Control, Shape and Motion Planning

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science & Engineering

by

Dinh Huan Nguyen

Prof. Dr. Kostas Alexis - Thesis Advisor
December 2020
THE GRADUATE SCHOOL

We recommend that the thesis prepared under our supervision by

DINH HUAN NGUYEN

entitled

Aerial Robotic Chain: Modeling, Control, Shape and Motion Planning

be accepted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Kostas Alexis, Ph.D.
Advisor

Christos Papachristos, Ph.D.
Committee Member

Antonio Franchi, Ph.D.
Committee Member

M Sami Fadali, Ph.D.
Graduate School Representative

David W. Zeh, Ph.D., Dean
Graduate School

December, 2020
Abstract

Research in aerial robotics is pushing the frontier of overall autonomy, sensing, processing and endurance characteristics of Micro Aerial Vehicles (MAVs). Flying robots are now being integrated in an ever increasing set of applications including those of infrastructure inspection, mine exploration, surveillance, entertainment and more. However, despite this unprecedented progress there still exist challenges and mission profiles that are rendered impossible for current aerial robotic technology. Largely, existing MAV designs are monolithic and thus present certain trade-offs, for example between payload and endurance or size. A traditional quadrotor design for example cannot simultaneously have the ability to navigate through narrow corridors and have the capacity to integrate a significant payload or present long endurance. Aiming to overcome these limitations, this work investigates the advanced potential of a reconfigurable and multilinked system-of-systems of aerial robots in order to simultaneously achieve the ability to cross narrow sections, morph shape, ferry significant payloads, offer distributed sensing and computing, and enable redundancy alongside system extendability. The proposed design, dubbed the Aerial Robotic Chain (ARC), corresponds to a reconfigurable system of systems with individual quadrotor MAVs (ARC-units) connected through rigid links and 3-Degree of Freedom (DoF) joints. The proposed design is generic, applicable to N-connected MAVs, while in this specific work we have realized an experimental prototype consisting of two connected quadrotors (ARC-Alpha). In this work we contribute the design, modeling, control design, shape and motion planning for the aerial robotic chain. In particular, a parallel control design consisting of individual $SO(3)$ controller for angular control of each rigid link, alongside Model Predictive Control for the position control of the system-of-systems is proposed. The shape and motion planner for ARC exploits a library of Aerial Robotic Chain configurations, optimized either for cross-section size or sensor
coverage, alongside a probabilistic strategy to sample random shape configurations that may be needed to facilitate continued collision-free navigation. Experiment and simulation studies demonstrate performance of the proposed controller and motion planner.
Dedication

Dedicated to my parents, brother and teachers.
Acknowledgments

I would like to thank my advisor, Prof. Konstantinos Alexis for his guidance, wisdom, and support. I would also like to thank Tung Dang, Shehryar Khattak, Frank Mascarich and many others for their valuable ideas in numerous discussions, and their patience in helping me complete the experiments.
# Table of Contents

1 Introduction 1

1.1 Motivation .................................................. 1

2 Related Work ................................................. 4

3 Design, Modeling and Control .................................. 7

3.1 System Design ................................................ 7

3.2 Dynamic Model .............................................. 9

3.3 Control Approach .......................................... 13

3.3.1 Model Predictive Position Control for the first ARC-link . 15

3.3.2 Angular Controller for each ARC-link .................... 18

3.3.3 Thrust Vector Assignment for each ARC-unit ............. 20

3.3.4 Upper bound for internal forces between the ARC-links and ARC-units .................................................. 21

3.4 Evaluation Studies ........................................... 28

3.4.1 Simulation Studies ........................................ 28

3.4.2 Experimental Studies ..................................... 30

3.5 Discussions .................................................. 33

4 Shape and Motion Planning .................................... 35

4.1 Problem Statement .......................................... 35
4.2 Shape and Position Control ........................................ 36

4.3 Proposed Approach ........................................... 37
   4.3.1 Translation Planner ....................................... 38
   4.3.2 Library of Shape Configurations .......................... 39
   4.3.3 Set of Random Shapes .................................... 40
   4.3.4 SRS-based Edge Connection ............................... 42
   4.3.5 Constructing the Solution Online ........................ 43
   4.3.6 Computational Complexity ............................... 45

4.4 Evaluation Studies ............................................ 46
   4.4.1 Environment 1: Room with Window ...................... 47
   4.4.2 Environment 2: Maze-like Room .......................... 49
   4.4.3 Comparison against full-state PRM ...................... 49
   4.4.4 Application-driven Demonstration ...................... 50

5 Conclusions and Future Work .................................. 54

References ........................................................ 57
List of Tables

4.1 Key Statistics for Environment 1 ........................................ 48
4.2 Key Statistics for Environment 2 ........................................ 49
4.3 Comparison against full-state PRM ................................. 50
# List of Figures

1.1 Design of ARC system with $N=4$ ARC-units .................................................. 2  
1.2 ARC alpha system with 2 ARC-units ................................................................. 2  

3.1 Illustration of the extendable and reconfigurable Aerial Robotic Chain concept ................................................................. 8  
3.2 Dynamic model diagram of the aerial robotic chain ........................................... 9  
3.3 Control diagram for the Aerial Robotic Chain ..................................................... 14  
3.4 Simulation result for an ARC system with 3 units ............................................. 29  
3.5 Rectangular trajectory tracking experiment ....................................................... 31  
3.6 Helical trajectory tracking experiment .................................................................. 32  
3.7 Window passing experiment .................................................................................. 33  

4.1 Block diagram overview of the proposed Aerial Robotic Chain planner (ARCplanner) ................................................................. 38  
4.2 Instances of the ARC shapes stored in the “Library of ARC Shape Configurations” (LSC) ................................................................. 51  
4.3 Instances of the Gazebo-based simulation of an ARC with $N = 5$ ARC-units navigating a room with a window based on the ARCplanner ........................................ 52  
4.4 ARCplanner solution for an ARC with $N = 5$ ARC-units navigating a room with a window ................................................................. 52
4.5 ARCplanner solution for an ARC with \( N = 5 \) robots navigating a maze-like room

4.6 ARCplanner solution for \( N = 6 \) ARC-units navigating a two-silo structure and co-optimizing for sensor coverage
Chapter 1

Introduction

1.1 Motivation

Research in aerial robotics has enabled their widespread utilization in a multitude of applications including infrastructure inspection [1], exploration [2,3], surveillance [4], entertainment [5] and more. One new trend in the field is to develop flexible, reconfigurable platforms that can adapt to different tasks or environments by morphing their shapes. Previous relevant work has demonstrated quadrotor Micro Aerial Vehicles (MAVs) that can fold their arms to pass through narrow windows [6,7], lattice-based [8] and chain-based [9] modular platforms that can reconfigure their shapes in response to a given task. As compared to single MAV platforms, multi-system modular designs present the advantages that they can integrate significantly more payload, distribute their sensing and processing capabilities and thus also facilitate redundancy, cooperatively exploit the thrust vectors for novel aerial physical interaction. Furthermore, chained multi-linked MAV platforms have the ability to traverse through narrow sections more effectively than either monolithic larger vehicles or lattice-based platforms. Motivated by the aforementioned benefits, in this work we
investigate the problem of control and motion planning for a new type of reconfigurable multi-linked aerial robotic system-of-systems, called the Aerial Robotic Chain (ARC) depicted in Figure 1.1 and a realized prototype system with 2 ARC-units is illustrated in Figure 1.2.

![Figure 1.1: Visualization of the aerial robotic chain with 4 ARC-units.](image1)

![Figure 1.2: The ARC-Alpha prototype of the presented Aerial Robotic Chain.](image2)

The design of ARC consists of individual quadrotor MAVs (“ARC-units”) connected through rigid links and 3-DoF joints. Each ARC-unit is considered to integrate the necessary low-level autopilot functionalities (attitude and thrust control), alongside processing and sensing payloads. These ARC-units are then connected with each other through rigid links that also integrates the necessary communications wiring such that information is exchanged among the chain modules.

This work consists of two parts: in the first part, we contribute the design, modeling and control design for the Aerial Robotic Chain; in the second part, an efficient collision-free motion planner for ARC system is proposed given the controller designed
in the first part.

In particular, the control structure consists of multiple parallel $SO(3)$ controllers for angular control of the rigid links, alongside a Model Predictive Control for the position control of the first link. We investigate in detail when this approach is valid by providing an upper bound for the internal forces between the ARC-units and neighbor links. Simulation studies for system with N ARC-units and experimental evaluation on prototyped ARC-Alpha system with 2 ARC-units demonstrate the performance of the controller.

The proposed motion planning approach (ARCplanner) identifies navigation solutions for the Aerial Robotic Chain with an arbitrary number of units by decoupling the translation planner and the configuration (shape) planner. For the configuration planner, we propose a bifurcated approach of a) utilizing a library of fixed configurations, generated either to minimize cross-section or to maximize sensor coverage given a certain frustum model for the perception system onboard each ARC-unit, as well as b) a random sampling-based search algorithm responsible for identifying collision-free paths when such solutions are not possible on the basis of the library-stored configurations.

The remainder of this thesis is organized as follows: Section 2 overviews related work, while Section 3 describes dynamic model and control structure for ARC system. Section 4 presents ARC motion planner, followed by the conclusions in Section 5.
Chapter 2

Related Work

The ARC corresponds to a new contribution in the domain of shape reconfiguration for flying robots, a domain of research that is attracting increased attention. One of the first system is the Distributed Flight Array \cite{8}, a lattice-based, modular, reconfigurable vehicle composed of hexagonal-shaped modules attached by magnets. This work used a distributed sensing and control where little information is needed to be exchanged between modules. The DRAGON robotic system which was recently introduced \cite{9,10} is a dual-rotor-embedded multilink robot with the ability of multi-DoF transformation. It can control the full pose in $\mathbb{SE}(3)$ and through a prototype consisting of four links it demonstrated the ability to adjust its shape to go through a narrow window. Our design is a chain-based design similar to DRAGON, however the main difference of the ARC design and control is that our joints have no motor and we use the thrust vectors of every ARC-unit to both stabilize the system and change its shape. The work in \cite{11,12} presents a multilinked multirotor with an internal communication system made to enable the transportation of objects of significant size by exploiting form adaptation. As such, the shape of the robot is modified online by adjusting the configuration of its joints in a manner that is best suitable to transfer certain objects. The contribution in \cite{13} presents a dual connected bi-copter with a
controller that allows it to fly at an arbitrary tilt angle. At the same time, this field of research shares ideas with those of multi-robot payload transfer [14][15], and aerial manipulation [16][17]. In comparison with the existing investigations in the field, ARC differs in that its design emphasizes on agile navigation through narrow cross sections, while simultaneously allowing reconfigurability, selectable payload capacity, and the potential for distributed integration of the desired sensing and processing solutions.

Motion planning for robot systems with high degrees of freedom is an interesting research area. ARC involves multiple degrees of freedom hence full sampling of the configuration space online is challenging [18]. Our robot can be approximated as a deformable object for which the dimensionality of the configuration space can be reduced. [19] uses principal component analysis (PCA) to reduce the dimension of deformation space. However, PCA can add narrow passages to the problem and create implausible deformations of the reconstructed robot. [20] assumes certain joint angles to be fixed, while simulating forward dynamics which may not be valid in our case. [21] proposes an efficient representation for deformable linear objects and a path planner for them by reusing a local shape planner in a global roadmap. The community of reconfigurable robots has introduced several path planning methods. [22] uses a leader-follower approach in which the path of the first robot (the head) is computed from a Generalized Voronoi Graph (GVG). [23] proposes a path planner based on differential kinematics but the assumption that obstacles can be represented as primitive geometries may not hold in general. [24] reduces the task of searching the whole configuration space by implementing a base planner for a few modules and reuse it within a hierarchical planner, however this method is only applicable to reconfigurable robots having self-similar structure. Our approach is closest to [25] which uses a high-level planner to determine which of its defined configurations suits best for the current task or environment, combined with a local configuration plan-
ner. However, our approach is different in that a) we focus on cluttered environments, hence if the configuration library fails to find a solution, we utilize the solutions of a local PRM-based configuration planner to find a valid shape, and b) we replan the translation path when the local planner does not find a solution.
Chapter 3

Design, Modeling and Control

3.1 System Design

The Aerial Robotic Chain concept focuses on reconfigurability to facilitate advanced sensing, processing, and endurance capabilities combined with the ability of shape morphing in order to navigate through complex and possibly confined environments. In comparison to monolithic aerial robot designs, it provides an alternative to the typical trade-off between payload/endurance and the minimum environment cross section that the system can navigate through. To achieve this goal, ARC consists of multiple quadrotor MAVs (ARC-units) rigidly connected with each other via lightweight rods and respective 3DoF rotational joints at the connection points. The rigid link serves to facilitate a) definite geometric configurations, b) communications between the robots (through the integration of respective wiring), and c) the (future) ability to estimate the pose of one robot from the others assuming the integration of encoders within each joint. Figure 3.1 visualizes the ARC concept.

In the particular prototype realization of the aerial robotic chain, a system consisting of two ARC-units and a connecting rod was developed. This prototype, ARC-
Figure 3.1: Illustration of the extendable and reconfigurable Aerial Robotic Chain concept. A collision-tolerant quadrotor is considered as the basis of each ARC-unit, while the different units are connected through lightweight rods attached on 3-DoF rotational joints.

Alpha, is built around a collision-tolerant quadrotor design with 5-inch 3-blade propellers (Lumenier 5 × 5 × 3) and high–torque motors (Lumenier LX2205-12 2400kV), integrating a PX4 autopilot responsible for attitude and thrust control, and an UP Core Plus board with an Atom x7-E3950 2GHz responsible for high-level control and further autonomy functionalities. The first ARC-unit of ARC-Alpha further integrates a Realsense T265 visual-inertial Tracking system. On the other hand, the second ARC-unit integrates a FLIR Boson LWIR thermal camera. The distribution of the required sensing payload to two different ARC-units follows the principle design goal of a distributed and reconfigurable system that at the same time can have the payload of a larger unit with the navigation capacity of smaller systems in terms of flying through narrow cross sections. The ARC-units are connected through two 3DoF rotational joints that can be integrated with optical encoders (US digital E4T), while the connecting rod further integrates wiring that allows the UP boards of the robots to be connected with each other. Thus, the design of the ARC-Alpha allows for distributed computation among the two robotic units and sharing of information. In the future, the encoders on the joints will be used to enable the estimation of the pose of the second robot from the first, while the distributed computing will be exploited to allow the control to run on either both of onboard computers or on any of them, while also allowing “hot-swap”. The design of ARC-Alpha is a milestone in
the direction of realizing the extendable aerial robotic chain.

3.2 Dynamic Model

In this section, the model capturing the dynamics of the aerial robotic chain is presented under the assumption that each ARC-unit is a quadrotor MAV as depicted in Figure 3.2.

Let $W$ denote the world frame, $B_i$ the body-fixed frame of ARC-unit $i$, and $B_{L_i}$ be the body-fixed frame of the connecting link $i$ with orthonormal basis $\{L_i e_1, L_i e_2, L_i e_3\}$. Furthermore, let $N$ be the number of ARC-units, $L_i$ the link connecting unit $i$ to $i+1$, $J_{i,j}$ the joint connecting the ARC-unit $i$ to the link between ARC-unit $i$ and ARC-unit $j$, $d_{i,j} \in \mathbb{R}^3$ the vector from the Center-of-Gravity (CoG) of ARC-unit $i$ to the joint $J_{i,j}$ expressed in $B_i$, and $l_i \in \mathbb{R}^3$ the vector from joint $J_{i+1,i}$ to joint $J_{i,i+1}$ expressed in $B_{L_i}$, $l_i \parallel L_i e_1$, let $u_i \in \mathbb{R}^3$ be the thrust vector generated by ARC-unit $i$ expressed in $W$, $g \in \mathbb{R}^3$ be the gravitational force expressed in $W$, $M_i \in \mathbb{R}^3$ the moments generated by ARC-unit $i$ expressed in $B_i$, $m_i$ the mass of ARC-unit $i$, $m_{L_i}$ the mass of link $L_i$, $x_i, v_i$ the position and velocity of ARC-unit $i$ expressed in $W$, $R_i \in SO(3)$ the rotation matrix from $B_i$ frame to $W$, $R_{L_i} \in SO(3)$ the rotation matrix from $B_{L_i}$ to $W$, $\Omega_i \in \mathbb{R}^3$ the angular velocity of ARC-unit $i$ expressed in $B_i$, $\Omega_{L_i} \in \mathbb{R}^3$ the
angular velocity of link $L_i$ expressed in $\mathbb{B}_{L_i}$, $J_i \in \mathbb{R}^{3 \times 3}$ the inertia matrix of ARC-unit $i$ with respect to $\mathbb{B}_i$, $J_{L_i} \in \mathbb{R}^{3 \times 3}$ the inertia matrix of link $L_i$ with respect to $\mathbb{B}_{L_i}$, and $F_{L_i,j}$ the force applied by link $L_i$ to ARC-unit $j$ expressed in $\mathbb{W}$, downwash effect is not considered in this work. Finally, let $\hat{a}$ be the hat operator on $a \in \mathbb{R}^3$ defined by $\hat{a}b = a \times b \ \forall a, b \in \mathbb{R}^3$, and assume that $m_{L_i} = 0$ (weightless link) for $1 < i < N - 1$. Then it holds that:

$$-F_{L_{i,i}} - F_{L_{i,i+1}} \approx 0 \Rightarrow$$

$$F_{L_{i,i}} = F_i, \ F_{L_{i,i+1}} = -F_i, \ F_i \in \mathbb{R}^3$$

From Newton’s second law applied on each ARC-unit:

$$\begin{align*}
\ddot{x}_1 &= \frac{1}{m_1}(u_1 - m_1g + F_1) \\
\ddot{x}_2 &= \frac{1}{m_2}(u_2 - m_2g - F_1 + F_2) \\
&\vdots \\
\ddot{x}_i &= \frac{1}{m_i}(u_i - m_ig - F_{i-1} + F_i), \ 1 < i < N \\
&\vdots \\
\ddot{x}_N &= \frac{1}{m_N}(u_N - m_Ng - F_{N-1})
\end{align*}$$

(3.2)

And applying Euler’s rotation equation on each robot:

$$\begin{align*}
\dot{\Omega}_1 &= J_1^{-1}(-\hat{\Omega}_1J_1\Omega_1 + M_1 + \hat{d}_{1,2}R_1^T F_1) \\
\dot{\Omega}_2 &= J_2^{-1}(-\hat{\Omega}_2J_2\Omega_2 + M_2 - \hat{d}_{2,1}R_2^T F_1 + \hat{d}_{2,3}R_2^T F_2) \\
&\vdots \\
\dot{\Omega}_i &= J_i^{-1}(-\hat{\Omega}_iJ_i\Omega_i + M_i - \hat{d}_{i,i-1}R_i^T F_{i-1} + \hat{d}_{i,i+1}R_i^T F_i), \ 1 < i < N \\
&\vdots \\
\dot{\Omega}_N &= J_N^{-1}(-\hat{\Omega}_NJ_N\Omega_N + M_N - \hat{d}_{N,N-1}R_N^T F_{N-1})
\end{align*}$$

(3.3)
Similarly, applying Euler’s formula for each link:

\[
\dot{\Omega}_L = J^{-1}_L (-\Omega_L J_L \sqrt{\Omega} L_i + \hat{J}_L R^T L_i \mathbf{F}_i), \quad 1 \leq i \leq N - 1
\]  

(3.4)

To derive the nonlinear equations describing the ARC dynamics, we calculate the force \( \mathbf{F}_i \), \((i = 1...N-1)\) in terms of \( \mathbf{R}_k, \Omega_k, \mathbf{u}_k, \mathbf{M}_k, \) \((k = 1...N)\) and \( \mathbf{R}_L, \Omega_L, \) \((j = 1...N - 1)\) and then plug \( \mathbf{F}_i \) into Eqs. (3.2) and (3.3). For \( i = 1...N - 1 \), we have:

\[
\mathbf{x}_i = \mathbf{x}_{i+1} + \mathbf{R}_{i+1} \mathbf{d}_{i+1,i} + \mathbf{R}_L l_i - \mathbf{R}_d \mathbf{d}_{i,i+1}
\]  

(3.5)

⇒ \( \ddot{\mathbf{x}}_i = \ddot{\mathbf{x}}_{i+1} + \mathbf{R}_{i+1} \dot{\Omega}^2_{i+1} \mathbf{d}_{i+1,i} - \mathbf{R}_{i+1} \dot{\mathbf{d}}_{i+1,i} \dot{\Omega}_{i+1} + \)

\[
\mathbf{R}_L \dot{\Omega}^2_L l_i - \mathbf{R}_L \dot{\mathbf{d}}_i \dot{\mathbf{d}}_{i,i+1} + \mathbf{R}_d \dot{\mathbf{d}}_{i,i+1} \dot{\Omega}_i
\]  

(3.6)

Plugging \( \ddot{\mathbf{x}}_i, \ddot{\mathbf{x}}_{i+1} \) in Eq. (3.2), \( \dot{\Omega}_i, \dot{\Omega}_{i+1} \) in Eq. (3.3), and \( \dot{\Omega}_L i \) in Eq. (3.4) results:

\[
a_{i,i-1} \mathbf{F}_{i-1} + a_{i,i} \mathbf{F}_i + a_{i,i+1} \mathbf{F}_{i+1} = \mathbf{b}_i
\]  

(3.7)

where matrices \( a_{i,i-1}, a_{i,i}, a_{i,i+1} \in \mathbb{R}^{3\times3} \) such that:

\[
a_{i,i-1} = \begin{cases} 0_{3\times3}, & \text{if } i = 1 \\ -\frac{1}{m_i} I_{3\times3} + \mathbf{R}_d \mathbf{d}_{i,i+1} J^{-1} \mathbf{d}_{i,i-1} R^T_i, & \text{if } i > 1 \end{cases}
\]  

(3.8)

\[
a_{i,i} = \left( \frac{1}{m_i} + \frac{1}{m_{i+1}} \right) I_{3\times3} - \mathbf{R}_{i+1} \mathbf{d}_{i+1,i} J^{-1} \mathbf{d}_{i+1,i+1} R^T_{i+1} + \]

\[
\mathbf{R}_L l_i J^{-1} \mathbf{R}^T_{L_i} - \mathbf{R}_d \mathbf{d}_{i,i+1} J^{-1} \mathbf{d}_{i,i+1} R^T_i
\]  

(3.9)
\[ a_{i,i+1} = \begin{cases} 0_{3 \times 3} & \text{, if } i = N - 1 \\ -\frac{1}{m_i}I_{3 \times 3} + R_{i+1}\hat{d}_{i+1,i}J_{i+1}^{-1}\hat{d}_{i+1,i+2}R_{i+1}^T & \text{, if } i < N - 1 \end{cases} \tag{3.10} \]

\[ b_i = -\frac{u_i}{m_i} + \frac{u_{i+1}}{m_{i+1}} + R_{i+1}\hat{\Omega}_{i+1}^2\hat{d}_{i+1,i} - R_{i+1}\hat{d}_{i+1,i}J_{i+1}^{-1}(-\hat{\Omega}_{i+1}J_{i+1}\hat{\Omega}_{i+1} + M_{i+1}) \]

\[ + R_L\hat{\Omega}_L^2l_i + R_L\hat{J}_L^{-1}\hat{\Omega}_LJ_L\hat{\Omega}_L - R_L\hat{\Omega}_L^2\hat{d}_{i,i+1} + R_L\hat{d}_{i,i+1}J_i^{-1}(-\hat{\Omega}_J\hat{\Omega}_J + M_i) \]

\[ \tag{3.11} \]

Let us define the vectors \( \mathbf{F} = [\mathbf{F}_1^T, \mathbf{F}_2^T, ..., \mathbf{F}_{N-1}^T]^T \), \( \mathbf{b} = [\mathbf{b}_1^T, \mathbf{b}_2^T, ..., \mathbf{b}_{N-1}^T]^T \), and matrix \( \mathbf{A} \in \mathbb{R}^{3(N-1) \times 3(N-1)} \) such that for \( 0 \leq i, j \leq N \):

\[ \mathbf{A}[3i : 3i + 2, 3j : 3j + 2] = \begin{cases} a_{i,i+1} & \text{, if } j = i + 1 \\ a_{i,i-1} & \text{, if } j = i - 1 \\ 0_{3 \times 3} & \text{, otherwise} \end{cases} \tag{3.12} \]

From Eq. (3.7), a system of equations of the form \( \mathbf{A}\mathbf{F} = \mathbf{b} \) is derived, hence allowing to solve for \( \mathbf{F} \) with each \( \mathbf{F}_i \) expressed as a function of:

\[ \mathbf{F}_i = \mathbf{f}_i(\mathbf{R}_1, \mathbf{\Omega}_1, ..., \mathbf{R}_N, \mathbf{\Omega}_N, \mathbf{R}_{L_1}, \mathbf{\Omega}_{L_1}, ... \mathbf{R}_{L_{N-1}}, \mathbf{\Omega}_{L_{N-1}}, \mathbf{u}_1, ..., \mathbf{u}_N, \mathbf{M}_1, ..., \mathbf{M}_N) \]

\[ \tag{3.13} \]

Then, plugging Eq. (3.13) into Eqs. (3.2), (3.3), and (3.4) leads to the full motion
dynamic equations of the ARC expressed as:

\[
\begin{align*}
\dot{R}_i &= R_i \hat{\Omega}_i \\
\dot{R}_{L_i} &= R_{L_i} \hat{\Omega}_{L_i} \\
\end{align*}
\]  

\text{Eq. (3.2)}

\text{Eq. (3.3)}

\text{Eq. (3.4)}

(3.14)

Subsequently, the ARC control design is presented.

### 3.3 Control Approach

To enable the control of the aerial robotic chain we designed a parallel control architecture, shown in Figure 3.3, consisting of one model predictive controller handling the position dynamics of the leading ARC-link, combined with \( N - 1 \) angular controllers for the \( N - 1 \) ARC-links, thrust assignment and conversion of such references to commands for a low-level attitude controller onboard each ARC-unit. For this analysis, the offsets between the robots’ CoG and the joints are considered negligible \( (d_{i,i+1} \approx 0^{3 \times 1}, d_{i,i-1} \approx 0^{3 \times 1}) \).

From Eq. (3.2), we have:

\[
m_1 \ddot{x}_1 + m_2 \ddot{x}_2 = u_1 + u_2 - (m_1 + m_2)g + F_2
\]  

(3.15)
Figure 3.3: Control diagram for the Aerial Robotic Chain system-of-systems considering $N$ ARC-units. Each robot $i$ runs control w.r.t link $L_i$.

Plugging $\ddot{x}_i, \dddot{x}_{i+1}$ in Eq. (3.2) to (3.6) and replace $d_{i,i+1}, d_{i+1,i}$ as $0^{3 \times 1}$ it holds:

$$
\begin{align*}
R_{L_1} \hat{\Omega}^2_{L_1} l_1 - R_{L_1} \hat{\Omega} L_1 &= \frac{1}{m_1} u_1 - \frac{1}{m_2} u_2 + \left( \frac{1}{m_1} + \frac{1}{m_2} \right) F_1 - \frac{1}{m_2} F_2 \\
&\text{...} \\
R_{L_i} \hat{\Omega}^2_{L_i} l_i - R_{L_i} \hat{\Omega} L_i &= \frac{1}{m_i} u_i - \frac{1}{m_{i+1}} u_{i+1} - \frac{1}{m_i} F_{i-1} + \left( \frac{1}{m_i} + \frac{1}{m_{i+1}} \right) F_i - \\
&\quad \frac{1}{m_{i+1}} F_{i+1}, \quad (i < i < N - 1) \\
&\text{...} \\
R_{L_{N-1}} \hat{\Omega}^2_{L_{N-1}} l_{N-1} - R_{L_{N-1}} \hat{\Omega} L_{N-1} &= \frac{1}{m_{N-1}} u_{N-1} - \frac{1}{m_N} u_N - \\
&\quad \frac{1}{m_{N-1}} F_{N-2} + \left( \frac{1}{m_{N-1}} + \frac{1}{m_N} \right) F_{N-1}
\end{align*}
$$

(3.16)
In order to decouple each equation in Eq. (3.15) and (3.16) and limit the force that each joint must tolerate, we limit each force $F_k$, $(1 < k < N - 1)$ by assigning:

$$\frac{1}{m_i}u_i - \frac{1}{m_{i+1}}u_{i+1} = a_{L_i,\text{rot}} + R_{L_i}\hat{\Omega}_{L_i}^2 l_i, \quad 1 \leq i \leq N - 1$$

(3.17)

where $a_{L_i,\text{rot}}$ is the control output from the angular controller of the $i$-th ARC-unit and is perpendicular to $R_{L_i}l_i$. The term $R_{L_i}\hat{\Omega}_{L_i}^2 l_i$ can be thought as a feedforward term to counteract the centripetal forces caused when the robots follow a curved trajectory.

When the condition in Eq. (3.17) holds, then the magnitude of $F_i$ is limited (as further detailed, alongside some more specific assumptions in section 3.3.4). Thus, we can then ignore the terms related to $F_i$ in Eqs. (3.15) and (3.16) during the control design process and instead treat them as disturbances.

### 3.3.1 Model Predictive Position Control for the first ARC-link

A Model Predictive Controller (MPC) is designed for the position dynamics of the leading ARC-unit and the rest of the motion control and coordination is handled by the parallel angular controllers and the thrust vector assignment for each ARC-unit. The choice of utilizing MPC is based on its ability to account for future references and the expected system response, while simultaneously respecting constraints in the control input vector.

Let us define a fixed point on the first link ($L_1$) with coordinates $x_{L_1}$ and velocity
\( \mathbf{v}_{L_1} \) satisfying:

\[
\mathbf{x}_{L_1} = \frac{m_1 \mathbf{x}_1 + m_2 \mathbf{x}_2}{m_1 + m_2}
\]  
(3.18)

And define:

\[
\mathbf{a}_{MPC} = \frac{\mathbf{u}_1 + \mathbf{u}_2 - (m_1 + m_2) \mathbf{g}}{m_1 + m_2}
\]  
(3.19)

Then it holds that:

\[
\dot{\mathbf{x}}_{L_1} = \mathbf{v}_{L_1}
\]  
(3.20)

\[
\dot{\mathbf{v}}_{L_1} = \ddot{\mathbf{x}}_{L_1} = \frac{m_1 \ddot{\mathbf{x}}_1 + m_2 \ddot{\mathbf{x}}_2}{m_1 + m_2} = \mathbf{a}_{MPC}
\]  
(3.21)

Using an approach similar to the one in [26], we design an MPC for the position dynamics of the leading ARC-unit with the difference here being that we use thrust components in 3-axes as control inputs instead of roll-pitch-thrust references. The model used for the MPC design takes the form:

\[
\dot{\mathbf{x}} = \mathbf{A}_c \mathbf{x} + \mathbf{B}_c \mathbf{u} + \mathbf{B}_{d,c} \mathbf{d}
\]  
(3.22)

where \( \mathbf{x} = [\mathbf{x}_{L_1}^T, \mathbf{v}_{L_1}^T]^T \), \( \mathbf{u} = \mathbf{a}_{MPC} \), \( \mathbf{d} = [d_x, d_y, d_z]^T \) (disturbance vector) and:

\[
\mathbf{A}_c = \begin{pmatrix} 0^{3 \times 3} & \mathbf{I}^{3 \times 3} \\ 0^{3 \times 3} & 0^{3 \times 3} \end{pmatrix}, \quad \mathbf{B}_c = \begin{pmatrix} 0^{3 \times 3} \\ \mathbf{I}^{3 \times 3} \end{pmatrix}, \quad \mathbf{B}_{d,c} = \begin{pmatrix} 0^{3 \times 3} \\ \mathbf{I}^{3 \times 3} \end{pmatrix}
\]  
(3.23)
Discretizing the above model leads to:

\[ x_{k+1} = Ax_k + Bu_k + B_d d_k \] (3.24)

Given the above representation, the MPC optimization problem can be written as:

\[
\begin{align*}
\min_U & \sum_{k=0}^{N-1} (||x_k - x_{ref,k}||^2_{Q_x} + ||u_k - u_{ref,k}||^2_{R_u}) \\
\text{s.t.} & \quad x_{k+1} = Ax_k + Bu_k + B_d d_k \\
& \quad d_{k+1} = d_k \\
& \quad \left( \begin{array}{c} a_{x,min} \\ a_{y,min} \\ a_{z,min} \end{array} \right) \leq u_k \leq \left( \begin{array}{c} a_{x,max} \\ a_{y,max} \\ a_{z,max} \end{array} \right) \\
& \quad x_0 = x(t_0), d_0 = d(t_0)
\end{align*}
\] (3.25)

where \( Q_x \succeq 0 \) is the penalty on the state error, \( R_u \succeq 0 \) is the penalty on the control input, \( R_\Delta \succeq 0 \) is the penalty on control input’s variation, \( P \succeq 0 \) is the terminal state error penalty, \( x_{ref,k} \) and \( u_{ref,k} \) are the target state vector and target control input at time \( k \), respectively. The disturbance \( d_k \), assumed to be constant throughout the prediction horizon, is used to obtain offset-free reference tracking and is estimated at each control sampling time using a Kalman Filter. Furthermore, we can add the desired acceleration (expressed in \( \mathbb{W} \)) to the output of the MPC controller.
to achieve improved tracking:

\[ a_{MPC,k} = u_k + w a_{ref,k} \quad (3.26) \]

This control action is then combined with the output of the angular controller of ARC-unit 1 and through the “mixer and converter” we derive the respective thrust vector assignment which in turn leads to the roll-pitch-yaw and thrust commands to the low-level autopilot of the first ARC-unit.

### 3.3.2 Angular Controller for each ARC-link

Provided the position control for the leading ARC-unit, the rest of the control design relies on a parallel set of angular controllers combined with respective thrust assignment for each ARC-unit. From Eq. (3.16) it holds that:

\[ \hat{\Omega}_{Li}^2 l_i \hat{\Omega}_{Li} - \hat{\Omega}_{Li} = R_{Li}^T \left( \frac{u_i}{m_i} - \frac{u_{i+1}}{m_{i+1}} \right) \quad (3.27) \]

Multiplying the two sides of this equation with \( \hat{l}_i \) leads to:

\[ \hat{l}_i \hat{\Omega}_{Li}^2 l_i - \hat{l}_i \hat{\Omega}_{Li} = \hat{l}_i R_{Li}^T \left( \frac{u_i}{m_i} - \frac{u_{i+1}}{m_{i+1}} \right) \quad (3.28) \]

Using the formula \( a \times (b \times c) = b(a \cdot c) - c(a \cdot b) \), we have:

\[ \hat{l}_i \hat{\Omega}_{Li}^2 l_i = l_i \times (\Omega_{Li} \times (\Omega_{Li} \times l_i)) \Rightarrow \quad (3.29) \]
\[
\hat{l}_i \hat{\Omega} L_i^2 l_i = l_i \times (\Omega L_i (\Omega L_i \cdot l_i) - l_i (\Omega L_i \cdot \Omega L_i)) \Rightarrow (3.30)
\]
\[
\hat{l}_i \hat{\Omega} L_i^2 l_i = \Omega L_i \times (-\hat{l}_i^2) \Omega L_i
\]

Plugging Eq. (3.31) into Eq. (3.28) and defining \(J_{L_i}^* = -\hat{l}_i^2\), we have:

\[
\Omega L_i \times J_{L_i}^* \Omega L_i + J_{L_i}^* \hat{\Omega} L_i = \hat{l}_i R_{L_i}^T \left( \frac{u_i}{m_i} - \frac{u_{i+1}}{m_{i+1}} \right) (3.32)
\]

By observation it is derived that Eq. (3.32) has similar dynamics to the attitude loop of a single quadrotor. Then, using the \(SO(3)\) angular controller described in \[27\], we derive the control law:

\[
\hat{l}_i R_{L_i}^T \left( \frac{u_i}{m_i} - \frac{u_{i+1}}{m_{i+1}} \right) = -K_{R,L_i} e_{R,L_i} - K_{\Omega,L_i} e_{\Omega,L_i} (3.33)
\]

where

\[
e_{R,L_i} = \frac{1}{2} (R_{L_i,d}^T R_{L_i} - R_{L_i}^T R_{L_i,d})^\vee (3.34)
\]
\[
e_{\Omega,L_i} = \Omega L_i - R_{L_i}^T R_{L_i,d} \Omega L_i,d (3.35)
\]

From Eq. (3.33) we choose:

\[
\frac{u_i}{m_i} - \frac{u_{i+1}}{m_{i+1}} = \underbrace{R_{L_i}(-K_{R,L_i} e_{R,L_i} - K_{\Omega,L_i} e_{\Omega,L_i}) \times \frac{l_i}{\|l_i\|^2}}_{a_{L_i,rot}} (3.36)
\]

The right side of this equation is \(a_{L_i,rot}\) as defined in Eq. (3.17), Eq. (3.36) guara-
tees that $\mathbf{a}_{L_i,rot}$ is perpendicular to $\mathbf{R}_{L_i}$. Notice that in Eq. (3.27), $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ cannot affect $\Omega_{L_i,x}$, hence we cannot control the roll angle of the ARC-link, as presented in the evaluation studies in Section 3.4.1.

### 3.3.3 Thrust Vector Assignment for each ARC-unit

The last step in the control design of the aerial robotic chain is the assignment of thrust vector values for each ARC-unit given $\mathbf{a}_{L_i,rot}$ and $\mathbf{a}_{MPC}$. For the first unit we have:

\[
\begin{align*}
\begin{cases}
\mathbf{u}_1 + \mathbf{u}_2 - (m_1 + m_2) \mathbf{g} = (m_1 + m_2) \mathbf{a}_{MPC} \\
\frac{1}{m_1} \mathbf{u}_1 - \frac{1}{m_2} \mathbf{u}_2 = \mathbf{a}_{L_1,rot} + \mathbf{R}_{L_1} \hat{\mathbf{\Omega}}_{L_1}^2 \mathbf{e}_1
\end{cases}
\end{align*}
\]

(3.37)

\[\Rightarrow \begin{cases}
\mathbf{u}_1 = m_1 (\mathbf{g} + \mathbf{a}_{MPC}) + \frac{m_1 m_2}{m_1 + m_2} (\mathbf{a}_{L_1,rot} + \mathbf{R}_{L_1} \hat{\mathbf{\Omega}}_{L_1}^2 \mathbf{e}_1) \\
\mathbf{u}_2 = m_2 (\mathbf{g} + \mathbf{a}_{MPC}) - \frac{m_1 m_2}{m_1 + m_2} (\mathbf{a}_{L_1,rot} + \mathbf{R}_{L_1} \hat{\mathbf{\Omega}}_{L_1}^2 \mathbf{e}_1)
\end{cases}
\]

(3.38)

Similarly, for the $i$-th ARC-unit with $i = 2...N - 1$:

\[
\begin{align*}
\frac{\mathbf{u}_i}{m_i} - \frac{\mathbf{u}_{i+1}}{m_{i+1}} &= \mathbf{a}_{L_i,rot} + \mathbf{R}_{L_i} \hat{\mathbf{\Omega}}_{L_i}^2 \mathbf{e}_i \\
\Rightarrow \mathbf{u}_{i+1} &= \frac{m_{i+1}}{m_i} \mathbf{u}_i - m_{i+1} (\mathbf{a}_{L_i,rot} + \mathbf{R}_{L_i} \hat{\mathbf{\Omega}}_{L_i}^2 \mathbf{e}_i)
\end{align*}
\]

(3.39)

(3.40)

Finally, after deriving the reference thrust vector $\mathbf{u}_{i,ref}$ for each ARC-unit, we need to convert from the thrust vector reference to roll-pitch-thrust commands to be tracked by the low-level autopilot onboard each ARC-unit (e.g., based on the PX4
open-source autopilot). Define $T_i$, $\phi_i$, $\theta_i$, $\psi_i$ as reference thrust, roll, and pitch angles, alongside the yaw angle of the quadrotor $i$, then the following conversion between thrust vector to attitude/thrust commands holds:

$$T_i = ||u_i||_2$$

$$e_i = \frac{u_i}{T_i}$$

$$a_i = e_i(0) \cos(\psi_i) + e_i(1) \sin(\psi_i)$$

(3.41)

$$b_i = e_i(0) \sin(\psi_i) - e_i(1) \cos(\psi_i)$$

$$\theta_i = \arctan2(a_i, e_i(2))$$

$$\phi_i = \arctan2(b_i, \sqrt{a_i^2 + e_i(2)^2})$$

where $e_i(\ell)$ the $\ell$ element of vector $e_i$. Each parallel control component is running onboard each ARC-unit, while the MPC only runs on the first ARC-unit.

### 3.3.4 Upper bound for internal forces between the ARC-links and ARC-units

Let $F_{i,j}^\parallel$ be the projected vector of $F_i$ onto vector $R_{L_i}l_j$, $F_{i,j}^\perp$ the projected vector of $F_i$ onto the plane that is perpendicular to vector $R_{L_i}l_j$, $M = \max_i \{||F_{i,i}^\parallel||_2\}$. From Eq. (3.4), we have

$$J_{L_i}^T \dot{\Omega}_{L_i} + \dot{\Omega}_{L_i} J_{L_i} \Omega_{L_i} = \hat{1}_i R_{L_i}^T F_{i,i}^\perp, \quad 1 \leq i \leq N - 1$$

(3.42)
Assume that every elements in inertia matrix of the link $L_i$, $J_{L_i}$ are small, we have the magnitude of $\mathbf{F}^\perp_{i,i}(1 \leq i \leq N - 1)$ is limited and hence $M$ is limited. We prove that with the conditions

(C1) Eq. (3.17)

(C2) $N\lambda < \pi$, with $\lambda = \cos(\frac{D}{2}) > 0, D = \min \left\{ \min_i \left\{ \frac{(m_i + m_{i+1})^2}{m_i^2 + m_{i+1}^2} \right\} , 2 - \epsilon \right\}, \epsilon$ is small.

we can find an upper bound for the magnitude of $\mathbf{F}^\parallel_{i,i}$ and hence $\mathbf{F}_i$ based on $M$.

Plugging Eq. (3.17) into Eq. (3.16), we have

\[
\begin{align*}
-R_{L_1,\hat{l}_1} \hat{\Omega}_{L_1} &= a_{L_1,\text{rot}} + \left( \frac{1}{m_1} + \frac{1}{m_2} \right) F_1 - \frac{1}{m_2} F_2 \\
-\cdots
\end{align*}
\]

\[
\begin{align*}
-R_{L_i,\hat{l}_i} \hat{\Omega}_{L_i} &= a_{L_i,\text{rot}} - \frac{1}{m_i} F_{i-1} + \left( \frac{1}{m_i} + \frac{1}{m_{i+1}} \right) F_i - \frac{1}{m_{i+1}} F_{i+1}, 1 < i < N - 1 \\
-\cdots
\end{align*}
\]

\[
\begin{align*}
-R_{L_{N-1},\hat{l}_{N-1}} \hat{\Omega}_{L_{N-1}} &= a_{L_{N-1},\text{rot}} - \frac{1}{m_{N-1}} F_{N-2} + \left( \frac{1}{m_{N-1}} + \frac{1}{m_N} \right) F_{N-1}
\end{align*}
\]

(3.43)

Notice that vectors $-\mathbf{R}_{L_i,\hat{l}_i} \hat{\Omega}_{L_i}$ and $a_{L_i,\text{rot}}$ are perpendicular to vector $\mathbf{R}_{L_i,\hat{l}_i}$. Projecting two sides of equation $i(1 \leq i \leq N - 1)$ in system of equations (3.43) onto vector $\mathbf{R}_{L_i,\hat{l}_i}$, we have
\[
\begin{align*}
(\frac{1}{m_1} + \frac{1}{m_2})F_{1,1} - \frac{1}{m_2}F_{2,1} &= 0 \\
...
- \frac{1}{m_i}F_{i-1,i} + (\frac{1}{m_i} + \frac{1}{m_{i+1}})F_{i,i} - \frac{1}{m_{i+1}}F_{i+1,i} &= 0, 1 < i < N - 1 \\
...
- \frac{1}{m_{N-1}}F_{N-2,N-1} + (\frac{1}{m_{N-1}} + \frac{1}{m_N})F_{N-1,N-1} &= 0
\end{align*}
\]

\[
F_{1,1} = \frac{m_1}{m_1 + m_2}F_{2,1}
\]

\[
\begin{align*}
F_{i,i} &= \frac{m_{i+1}}{m_i + m_{i+1}}F_{i-1,i} + \frac{m_1}{m_i + m_{i+1}}F_{i+1,i}, 1 < i < N - 1 \\
...
F_{N-1,N-1} &= \frac{m_N}{m_{N-1} + m_N}F_{N-2,N-1}
\end{align*}
\]

Define \( t_i = ||F_{i,i}||_2^2 \), we have (with \( 1 < i < N - 1 \))

\[
||F_{i-1,i}||_2^2 \leq ||F_{i-1}||_2^2 = ||F_{i-1,i-1}||_2^2 + ||F_{i-1,i-1}^\perp||_2^2 \leq t_{i-1} + M^2
\]

\[
||F_{i+1,i}||_2^2 \leq ||F_{i+1}||_2^2 = ||F_{i+1,i+1}||_2^2 + ||F_{i+1,i+1}^\perp||_2^2 \leq t_{i+1} + M^2
\]
From Eq. (3.45), Eq. (3.46) and Eq. (3.47), we have

\[
\begin{align*}
    t_1 &= \| \mathbf{F}_{1,1} \|_2^2 = \frac{m_1^2}{(m_1 + m_2)^2} \| \mathbf{F}_{2,1} \|_2^2 \leq \frac{1}{D} (t_2 + M^2) \\
    &
\end{align*}
\]

... 

\[
\begin{align*}
    t_i &= \| \mathbf{F}_{i,i} \|_2^2 = \left\| \frac{m_{i+1}}{m_i + m_{i+1}} \mathbf{F}_{i-1,i} + \frac{m_i}{m_i + m_{i+1}} \mathbf{F}_{i+1,i} \right\|_2^2 \\
    &\leq \frac{m_i^2 + m_{i+1}^2}{(m_i + m_{i+1})^2} (\| \mathbf{F}_{i-1,i} \|_2^2 + \| \mathbf{F}_{i+1,i} \|_2^2) \leq \frac{1}{D} (t_{i-1} + M^2 + t_{i+1} + M^2) \\
    , 1 < i < N - 1 \\
\end{align*}
\]

... 

\[
\begin{align*}
    t_{N-1} &= \| \mathbf{F}_{N-1,N-1} \|_2^2 = \frac{m_N^2}{(m_{N-1} + m_N)^2} \| \mathbf{F}_{N-2,N-1} \|_2^2 \leq \frac{1}{D} (t_{N-2} + M^2) \\
    Dt_1 - t_2 &\leq M^2 \\
    &
\end{align*}
\]

... 

\[
\begin{align*}
    -t_{i-1} + Dt_i - t_{i+1} &\leq 2M^2, 1 < i < N - 1 \\
    &
\end{align*}
\]

... 

\[
\begin{align*}
    -t_{N-2} + Dt_{N-1} &\leq M^2 \\
    &
\end{align*}
\]

Let us define matrix \( \mathbf{A} \in \mathbb{R}^{(N-1)\times(N-1)} \) and vector \( \mathbf{b} \in \mathbb{R}^{(N-1)\times1} \) as
A = \begin{pmatrix}
  D & -1 & 0 & \ldots & 0 & 0 \\
  -1 & D & -1 & \ldots & 0 & 0 \\
  0 & -1 & D & \ldots & 0 & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  0 & 0 & 0 & \ldots & D & -1 \\
  0 & 0 & 0 & \ldots & -1 & D 
\end{pmatrix}, b = \begin{pmatrix}
  M^2 \\
  2M^2 \\
  2M^2 \\
  \vdots \\
  2M^2 \\
  M^2 
\end{pmatrix} \quad (3.50)

Furthermore, let $c_i (1 \leq i \leq N - 1)$ be the column vector with $i^{th}$ element equal to 1 and other elements equal to 0, $t = [t_1, t_2, \ldots, t_{N-1}]^T$. In order to find the upper bounds for $t_i (1 \leq i \leq N - 1)$, we solve $N - 1$ LP problems of the form

$$\begin{align*}
\text{max} & \quad c_i^T t \\
\text{s.t.} & \quad At \leq b \\
& \quad t \geq 0
\end{align*} \quad (3.51)$$

**Proposition 1:** With condition (C2), matrix $A$ is invertible and each element in the inverse matrix of $A$ is positive.

*Proof:* Matrix $A$ is a symmetric tridiagonal matrix, the determinant of $A$ is given in [28]

$$|A| = (-1)^{N-1} \sin N\lambda \frac{\sin \lambda}{\sin \lambda} \quad (3.52)$$

Since $0 < \lambda < N\lambda < \pi$, we have $|A| \neq 0$ and $A$ is invertible. Define $R = A^{-1}$, each element of $R$ is given in [28]
\[ R_{ij} = \frac{-\cos (N - |j - i|)\lambda - \cos (N - i - j)\lambda}{2\sin \lambda \sin N\lambda} \]
\[ = \frac{\sin \frac{(i + j - |j - i|)\lambda}{2} \sin \frac{(2N - i - j - |j - i|)\lambda}{2}}{\sin \lambda \sin N\lambda} \]  
(3.53)

(3.54)

It can be seen that \( R_{ij} = R_{ji} \), without loss of generality, assume \( j \geq i \), we have

\[ R_{ij} = \frac{\sin i\lambda \sin (N - j)\lambda}{\sin \lambda \sin N\lambda} \]  
(3.55)

Since \( 0 < \lambda \leq i\lambda, (N - j)\lambda \leq N\lambda < \frac{\pi}{2} \), we have \( R_{ij} > 0 \ \forall i, j = 1, \ldots, N - 1 \).

**Proposition 2**: The upper bound for \( t_i \), i.e. the solution of LP problem (3.51) is \( i \)-th element of vector \( t^* = Rb \)

**Proof**: Karush Kuhn Tucker (KKT) conditions for LP problem (3.51)

\[ \begin{align*}
    A t^* & \leq b \\
    t^* & \geq 0 \\
    c_i - A^T \lambda^* + s^* & = 0 \quad (\lambda^* = [\lambda^*_1, \lambda^*_2, \ldots, \lambda^*_N]^{T}, s^* = [s^*_1, s^*_2, \ldots, s^*_N]^{T}) \\
    \lambda^* & \geq 0 \\
    s^* & \geq 0 \\
    \lambda^*_i (A_i t^* - b_i) & = 0, i = 1, \ldots, N - 1 \\
    s^*_i t^*_i & = 0, i = 1, \ldots, N - 1 
\end{align*} \]  
(3.56), (3.58), (3.60), (3.61), (3.62)

Choose \( t^* = Rb, \lambda^* = Rc_i, s^* = [0, \ldots, 0]^{T} \). Conditions (3.56), (3.58), (3.60), (3.61), (3.62) hold. Since all elements of \( R, b \) and \( c_i \) are non-negative, conditions (3.57) and (3.59)
are also satisfied. KKT conditions are all satisfied, hence \( t^* \) is the global maximizer for (3.51) and \( i-th \) element of vector \( t^* \) or \( t^*_i \) is the maximum value of \( t_i \).

We have

\[
||F_i||^2_2 = ||F_{i,i}||^2_2 + ||F_{i,i}^\perp||^2_2 \leq t^*_i + M^2
\]

\[
\implies ||F_i||_2 \leq \sqrt{t^*_i + M^2}
\]

Now we calculate the upper bounds of \( ||F_i||_2 \) for some specific cases.

**Case 1:** \( D = 1.999 \), which occurs when \( m_i = 0.95m_{i+1} \) or \( m_{i+1} = 0.95m_i \). We have \( \lambda = \acos(\frac{1.999}{2}) = 0.0316 \), number of drones \( N < \frac{\pi}{\lambda} = 99.42 \). Choose \( N = 6 \).

\[
t^* = A^{-1}b = \begin{pmatrix} 1.999 & -1 & 0 & 0 & 0 \\ -1 & 1.999 & -1 & 0 & 0 \\ 0 & -1 & 1.999 & -1 & 0 \\ 0 & 0 & -1 & 1.999 & -1 \\ 0 & 0 & 0 & -1 & 1.999 \end{pmatrix}^{-1} \begin{pmatrix} M^2 \\ 2M^2 \\ 2M^2 \\ 2M^2 \end{pmatrix} = \begin{pmatrix} 4.02M^2 \\ 7.03M^2 \\ 8.03M^2 \\ 7.03M^2 \end{pmatrix}
\]

From Eq. (3.64), we have \( ||F_1||_2 \leq 2.24M, ||F_2||_2 \leq 2.83M, ||F_3||_2 \leq 3M, ||F_4||_2 \leq 2.83M, ||F_5||_2 \leq 2.24M \)

**Case 2:** \( D = 1.75 \), which can happen when \( m_i = 0.45m_{i+1} \) or \( m_{i+1} = 0.45m_i \). We have \( \lambda = \acos(\frac{1.75}{2}) = 0.5 \), number of drones \( N < \frac{\pi}{\lambda} = 6.2 \). Choose \( N = 6 \).
From Eq. (3.64), we have $||F_1||_2 \leq 7.81M, ||F_2||_2 \leq 10.25M, ||F_3||_2 \leq 11M, ||F_4||_2 \leq 10.25M, ||F_5||_2 \leq 7.81M$.

It can be seen that there are two ways to obtain a tighter bound on $||F_i||_2$ and therefore validate the approach of independent control design:

- Making the masses of consecutive drones nearly identical to each other ($m_i \approx m_{i+1}$)
- Limiting desired angular accelerations and desired angular velocities of each link, hence limiting $M$ as can be seen from Eq. (3.42)

3.4 Evaluation Studies

To evaluate the presented control approach, both simulation and experimental studies were conducted.

3.4.1 Simulation Studies

Figure 3.4 presents the closed-loop tracking performance for an ARC system with $N = 3$ units and $m_i = 0.965$kg, $J_i = \text{diag}([0.0048, 0.0048, 0.0084])$kgm$^2$ ($i = 1, 2, 3$),
\( J_{L_j} = \text{diag}([0.00001, 0.0018, 0.0018])\text{kgm}^2 \) \((j = 1, 2)\), \( l_1 = l_2 = [0.353, 0, 0]^T\text{m}\), \( d_{2,1} = d_{1,3} = d_{3,1} = 0^{3 \times 1} \). The reference trajectory takes the form:

\[
x_{L1,d} = [0.5t, 0.3 \cos(\omega t - \frac{\pi}{2}), 0.3(1 + \sin(\omega t - \frac{\pi}{2}))]^T
\]

\[
[\phi_{L1,d}, \theta_{L1,d}, \psi_{L1,d}] = [0, \beta, \alpha]
\]

\[
[\phi_{L2,d}, \theta_{L2,d}, \psi_{L2,d}] = [0, -\beta, -\alpha]
\]

\[
\omega = \frac{2\pi}{10}, [\alpha, \beta] = \begin{cases} 
\left[\frac{\pi t}{60}, 0\right] & \text{if } t \leq 10 \\
\left[\frac{\pi}{6} \cos(\omega(t - 10)), \frac{\pi}{6} \sin(\omega(t - 10))\right] & \text{if } t > 10
\end{cases}
\]

Figure 3.4: Simulation result for an ARC system with 3 units. As shown, the roll angle of the connecting links (not of the robots) is not regulated.
3.4.2 Experimental Studies

A set of experimental studies were conducted using ARC-Alpha, the prototype implementation of the aerial robotic chain incorporating two ARC-units. The first experiment related to the ability of ARC-Alpha to track a rectangular trajectory without change in the robot orientation. Figure 3.5 presents this result indicating the small error from the reference. Within all the experimental studies in this paper, the pose estimate of each ARC-unit and of the connecting link are based on a Vicon Motion Capture system.

Subsequently, the system was commanded to track a helix generated by the relative motion of the two ARC-units, while the CoG of the ARC system as a whole was only commanded to move along the vertical axis (upward then downward). In this experiment, ARC-Alpha is exploiting its ability to create significant yaw moments by commanding opposite roll angles for its two ARC-units (quadrotors) and demonstrate coordinated flight. This result is shown in Figure 3.6.

Finally, a third experiment was conducted that related to ARC-Alpha navigating a trajectory involving the traversal through two windows thus requiring it to change its configuration in space as the planes of the two windows were perpendicular to each other and the width of each was small enough to require ARC-Alpha to traverse it “head-first”. It is noted that in all three experiments, two ARC-units are commanded to follow yaw angle of the link.
Figure 3.5: Rectangular trajectory tracking experiment: a) trajectories of the link and two ARC-units, b) sequence of ARC pose, c) and d) position and angular tracking performance of the link, respectively.
Figure 3.6: Helical trajectory tracking experiment: a) trajectories of the link and two ARC-units, b) sequence of ARC poses, c,d) position and angular tracking performance of the link, respectively.
Figure 3.7: Window passing experiment: a) trajectories of the link and two ARC-units, b) sequence of ARC poses when it pass the windows, c,d) position and angular tracking performance of the link, respectively.

3.5 Discussions

The above three experiments demonstrate reliable tracking accuracy and precision of the ARC system. It is noted that due to the current mechanical limits, the pitch angle of the link cannot be commanded to follow a large reference angle, this will
be left for future improvement in the joint’s design. Simulation results in RotorS environment \cite{29} with number of ARC-units $N = 4, 5, 6$ is given in Section 4.4.
Chapter 4

Shape and Motion Planning

4.1 Problem Statement

This section presents collision-free motion planning for the ARC multi-linked aerial robot. The following environment representation and problem definition are considered.

Definition 1 (Environment Representation) Given an environment of finite volume $V_M$, it is assumed to be represented through a known occupancy map $\mathcal{M}$ based on a discretization of voxels with edge length $r_v$. The map is organized in a subset of obstacle space leading to occupied voxels $\mathcal{M}_{occ}$, alongside collision-free space $\mathcal{M}_{free}$.

Problem 1 (ARC Navigation) Considering a dynamic model representation for the Aerial Robotic Chain system-of-systems $\dot{x} = f(x, u)$, where $x$ is the state and $u$ is the control, $x_0$ the current robot configuration, and given the obstacle space $\mathcal{M}_{occ}$ and the goal region $\mathcal{M}_{goal}$ then we have the associated space of in-collision configures
\( \mathcal{X}_{\text{obs}} \) and the goal configurations \( \mathcal{X}_{\text{goal}} \). Given the above, the objective is to find, if it exists, a sequence of configurations \( \{x\} \) leading to a path \( \sigma \) such that the solution satisfies \( \sigma \notin \mathcal{X}_{\text{obs}} \) and the final configuration \( x_f \) of the path \( \sigma \) is such that \( x_f \in \mathcal{X}_{\text{goal}} \).

### 4.2 Shape and Position Control

In addition to the notation used in section 3.2, we also denote \( [x_r, y_r, z_r]^T \) the position of ARC-unit \( i \) expressed in \( \mathbb{W} \), \([\phi_i, \theta_i, \psi_i]^T\) and \([\phi_l^i, \theta_l^i, \psi_l^i]^T\) respectively as roll, pitch, yaw Euler angles (zyx order) of quadrotor \( i \) and link \( i \); \( R_y, R_z \) the rotation matrices about the corresponding axes. We make two assumptions here: (A1) the offsets between the joints and robots’ COGs are negligible \((d_{i+1,i} \approx 0, d_{i,i+1} \approx 0)\); (A2) in normal operating conditions, tilt angles of every ARC-units are small. Also, \( R_{L_i} \) \( = R_z(\psi_l^i)R_y(\theta_l^i)l_i \) since \( l_i \parallel L e_1 \). Hence for \( 1 \leq i \leq N - 1 \), Eq. \( (3.5) \) can be rewritten as:

\[
[x_{i+1}, y_{i+1}, z_{i+1}]^T = [x_i, y_i, z_i]^T - R_z(\psi_l^i)R_y(\theta_l^i)l_i \tag{4.1}
\]

From assumptions (A1), (A2) and Eq. \((4.1)\), we choose the configuration space \( \zeta \) of the ARC system as \([x_1^r, y_1^r, z_1^r, \phi_1^r, \psi_1^r, \theta_1^l, \psi_1^l, \theta_2^l, \psi_2^l, \ldots, \theta_{N-1}^l, \psi_{N-1}^l]^T\) which has dimension equal to \( 2N + 2 \). We utilize the control structure presented in 3.3 for the shape and position control of ARC with one modification: the reference position is of the first ARC-unit \([x_{i,d}^r, y_{i,d}^r, z_{i,d}^r]^T\), the reference position for the first link can then be
calculated as:

\[
x_{L_1,d} = [x_{1,d}^r, y_{1,d}^r, z_{1,d}^r]^T - \frac{m_2}{m_1 + m_2} R_z(\psi_i^l) R_y(\theta_i^l) l_i
\]  

(4.2)

### 4.3 Proposed Approach

The proposed motion planning approach for the Aerial Robotic Chain integrating an arbitrary number of ARC-units, called ARCplanner, enables such a re-configurable aerial robotic system-of-systems to navigate complex environments in a computationally efficient manner. The ARCplanner employs a 2-step process of first attempting to find path segments that navigate from the current robot location to its goal, and subsequently identifying the best collision-free ARC shape configuration. For each path segment of the translation planning solution, the method first checks within a library of configurations which stores shape solutions if a collision-free solution based on one of the pre-considered configurations is possible. The library of ARC shape configurations contains two sets of solutions, namely a) optimized for the smallest possible cross-section, and b) optimized for maximum sensor coverage given a certain model of the frustum of the perception system onboard each ARC-unit. If a collision-free solution relying on the stored shape configurations is not found, the method proceeds to utilize a pre-sampled set of random configurations found through Probabilistic RoadMap (PRM) search of the configuration space for solutions regarding different ARC shapes in order to identify one that enables collision-free guidance for
the specific path segment at hand. It is important to note that this process is invoked only sparingly and it is also critical to mention that the random configurations are sampled a priori and thus limiting the penalty on online computational cost. Finally, if and only if no shape configuration allows to navigate across the planned path segment, then the method re-samples solutions for the robot translation on the basis that different topologies at that level may facilitate collision-free navigation for its possible shape configurations. The overall architecture of the ARCplanner is depicted in Figure 4.1.

Figure 4.1: Block diagram overview of the proposed Aerial Robotic Chain planner (ARCplanner).

4.3.1 Translation Planner

The first step of the ARCplanner is a PRM-based motion planning process that is responsible for identifying collision-free paths for the leading robot of the Aerial Robotic Chain. In that sense, this planner searches for admissible configurations in the $\xi = [x^r_1, y^r_1, z^r_1]^T$ space (leading robot location) given a roadmap build phase, outlined in Algorithm 1 and subsequent fast (multi-)query requests for paths. Func-
tion SampleFreeT samples configurations $\xi_i$ sampled from a uniform distribution, while NearT returns the other sampled vertices within a radius of $r_T$ from a vertex $v_T$. Function CollisionFreeT evaluates if a connection between two sampled vertices $v_T, u_T$ (corresponding to two leading robot configurations) is collision-free given an occupancy map representation of the environment based on the work in [30]. As Algorithm 1 builds the roadmap, subsequent multi-query collision-free paths can be derived seamlessly through lazy evaluation [18] and thus for any feasible navigation problem a path solution $\sigma_T$ is derived from the graph $\mathcal{G}_T$.

**Algorithm 1 Translation Planner**

1: $V_T \leftarrow \xi_0 \cup \{\text{SampleFreeT}(\xi_i)\}_i=1,\ldots,M_{\xi_0}$; $E_T \leftarrow 0$ $v_T \in V_T$
2: $U_T \leftarrow \text{NearT}(G_T = (V_T, E_T), v_T, r_T)$ \{$v_T$}; $u_T \in U_T$ CollisionFreeT($v_T, u_T$)
3: $E_T \leftarrow E_T \cup \{(v_T, u_T), (u_T, v_T)\}$;
4: $\mathcal{G}_T = (V_T, E_T)$, $\sigma_T$;

### 4.3.2 Library of Shape Configurations

As the dimensionality of the motion planning problem that needs to be solved to enable collision-free navigation is large for an ARC with a large number $N$ of ARC-units (e.g., $N \geq 4$) and since in most cases, certain geometric configurations tend to enable finding an admissible solution, a first step towards computational efficiency is to introduce a Library of ARC Shape Configurations (LSC). LSC consists of solutions optimized for a) narrow cross-section navigation, b) passing through corners, and c) maximizing the collective field-of-view of the robotic system-of-systems given an appropriately placed frustum-constrained sensor onboard each ARC-unit. As such,
for every \( N \) ARC-units, LSC may contain a) a “Line” configuration (LI), b) three serpentine configurations, specifically two (mirrored) on the x-y plane and one on x-z (SE, SM, SV), c) a Circular Arc (CA), and d) a Polygon Shape (PN). In the PN configuration for \( N \) ARC-units, the robots are positioned by constructing the associated polygon through defining an inscribing circle and specifying \( N \) equally-spaced nodes. In the CA configuration, the position of the robots is found by utilizing a \( 90^\circ \) arc and increasing/decreasing the radius as required to fit the ARC system given the length \( l_r \) of the connecting rod. The (SE, SM, SV) configurations are defined based on the maximum angle of the joint which in turn defines the serpentine shape, while their orientations are aligned with the connecting rods. The LI configuration is straightforward. The shapes in LSC are stored in a list denoted \( L_{LSC} \). Finally, it is noted that for every configuration in LSC, a set of \( n_\psi \) discrete yaw rotations \( \{\psi_r\} \) of the head robot are considered during the collision-free path evaluation phase. Figure 4.2 depicts instances of LSC configurations.

### 4.3.3 Set of Random Shapes

The configurations stored in the LSC tend to provide admissible collision-free solutions most of the time and thus lead to a major reduction in computation time, while exploiting the shape-reconfiguration abilities of the ARC. However, it is not impossible that certain motion planning problems for the ARC navigating in complex and confined environments cannot be solved using only LSC shapes (or their rotations around the azimuth). To allow our approach to provide a solution for such cases,
a further sampling-based planning stage is considered. In particular, a PRM-based method for shape configuration sampling is implemented and aims to provide admissible solutions for paths sampled by the translation planner when no LSC-based solution is feasible. Considering a hovering configuration of the leading robot, say in the arbitrary location $\xi = [x^r_1, y^r_1, z^r_1]^T$, the PRM stage will sample for the leading robot and the ARC links and thus propose multiple shape configurations in an effort to identify new attainable solutions. Thus, the configuration space ($\eta$-Space) of this sampling-based planning stage is $\eta = [\psi^r_1, \theta^l_1, \psi^l_1, \theta^l_2, ..., \theta^l_{N-1}, \psi^l_{N-1}]^T$, where $\psi^r_1$ is the heading of the leading robot, $\theta^l_i$, $\psi^l_i$ are the pitch and yaw angle of the connecting link $i$, and $N$ is the number of ARC-units in the overall chain. This step is executed offline and without any consideration of the specific map $\mathcal{M}$, while the $\eta$-space is chosen such that all shape transitions do not self-collide (straight edge in $\eta$-space by linearly varying the joint angles among configurations). Algorithm 2 outlines the main algorithmic steps of the random shape planer for the ARC system-of-systems based on PRM. This step is essential for solution resourcefulness and although in most cases LSC-configurations are sufficient, it was identified that in complex environments the ability to utilize new random shapes is key for successful and fast navigation. In the context of this algorithm, the function SampleFreeC samples random $\eta$ shape configurations corresponding to $v_S$ vertices in the $\eta$-Space graph, and NearC identifies a finite set of vertices that are within a radius $r_S$ from $v_S$. It is critical to highlight that this step is executed a priori, i.e., before the robot is even deployed to any environment, and as a result, a large number $N_{SRS}$ of randomly sampled shape configurations, a
Set of Random Shapes (SRS) ($\mathbb{L}_{SRS}$), are stored in its memory ready to be utilized online when needed.

**Algorithm 2 Random Shape Planner**

1. $\mathbb{V}_S \leftarrow \eta_{\text{init}} \cup \{\text{SampleFreeC}_i\}_{i=1,\ldots,M-1};$ $\mathbb{E}_S \leftarrow 0$ $v_s \in \mathbb{V}_S$
2. $\mathbb{U}_S \leftarrow \text{NearC}(G = (\mathbb{V}_S, \mathbb{E}_S), v_s, r_s) \setminus \{v_s\};$ $u_s \in \mathbb{U}_S$
3. $\mathbb{E}_S \leftarrow \mathbb{E}_S \cup \{(v_s,u_s), (u_s,v_s)\};$
4. $G_{SRS} \leftarrow (\mathbb{V}_S, \mathbb{E}_S), \mathbb{L}_{SRS};$

**4.3.4 SRS-based Edge Connection**

SRS provides the required resourcefulness of random shape configurations to facilitate transition between two points in complex environments. Let $\mathbb{L}_{SRS}$ represent the list of randomly sampled shape configurations in $G_{SRS}$. The associated SRS-based edge connection algorithm is presented in Algorithm 3. In this context, a Valid Shape is one that does not collide with the environment at the considered starting vertex location of the leading robot $\xi_i$. Similarly, an Admissible Shape is one that remains collision free along the edge $(\xi_i, \xi_j)$ that connects two leading robot locations $\xi_i$ and $\xi_j$ ($v_T, u_T$ in the $G_T$ graph). Furthermore, all Valid Shapes are kept as they may correspond to intermediate vertices in the $\eta$-Space allowing to reach the required Admissible Shapes. Thus, the graph $G_{Val}$ is constructed from all valid shapes with a radius of connection $d_c$ in the $\eta$-Space. The function $\text{OptAdmissibleTransition}$ identifies the path corresponding to the minimum distance to an admissible shape in the configuration space, while similarly the method $\text{CollisionFreeE}$ checks for the whole edge $\sigma_{\text{min}}$ to be collision-free. For all the cases that such an edge is not collision-
free, the graph $\mathbb{G}_{Val}$ is pruned. Finally, the method returns $\sigma_{\min}$ starting from $v_C$ configuration in $\zeta = [x^r_1, y^r_1, z^r_1, \psi^r_1, \theta^l_1, \psi^l_1, \theta^l_2, \psi^l_2, \ldots, \theta^l_{N-1}, \psi^l_{N-1}]^T$ and traversing along $v_T, u_T ((\xi_i, \xi_j))$, alongside a flag $s_u^v$ indicating planning success.

**Algorithm 3** SRSConnect($v_S, (v_T, u_T)$)

1. $\mathbb{L}_{Val}^{SRS} \leftarrow$ ValidShapes($\mathbb{L}_{SRS}, \xi_i$)
2. $\mathbb{L}_{Adm}^{SRS} \leftarrow$ AdmissibleShapes($\mathbb{L}_{Val}^{SRS}, (\xi_i, \xi_j)$)
3. $\mathbb{G}_{Val} \leftarrow$ ConstructGraph($\mathbb{L}_{Val}^{SRS}, d_c$)
4. $b_{CF} \leftarrow$ FALSE $b_{CF} = \text{FALSE}$
5. $\sigma_{\min}(v_T, u_T) \leftarrow$ OptAdmissibleTransition($\mathbb{G}_{Val}, \mathbb{L}_{Adm}^{SRS}$)
6. $b_{CF} \leftarrow$ CollisionFreeE($\sigma_{\min}(v_T, u_T)$) $b_{CF} = \text{FALSE}$
7. $\mathbb{G}_{Val} \leftarrow$ PruneCollidingEdges($\mathbb{G}_{Val}, \sigma_{\min}(v_T, u_T)$)
8. $\{\sigma_{\min}(v_T, u_T), s_u^v\}$

### 4.3.5 Constructing the Solution Online

Considering the library of shape configurations (LSC) and $N_{SRS}$ a priori randomly sampled new ARC shapes in SRS, both stored in the memory of the robot, alongside the online identified leading robot path $\sigma_T$ connecting to the desired location $\xi_{goal}$ without the head robot colliding, the algorithm is now ready to proceed to the final steps of identifying an optimized solution for the full ARC. More specifically, in a first step considering all edges $(\xi_i, \xi_j)$ in $\sigma_T$ (connecting $(v_T, u_T)$) in the $\mathbb{G}_T$ graph, the method searches inside LSC to identify such shapes that provide collision-free navigation across $\sigma_T$. For that purpose, let $u_S$ denote each vertex in the shape $\eta$-Space corresponding to a location of the leading robot and thus to a vertex in $\mathbb{G}_T$, and $u_C$ the respective vertex in the $\zeta = [x^r_1, y^r_1, z^r_1, \psi^r_1, \theta^l_1, \psi^l_1, \theta^l_2, \psi^l_2, \ldots, \theta^l_{N-1}, \psi^l_{N-1}]^T$ space. Then, the solutions inside LSC are searched in the following manner: a) from
the current robot shape first check if without any change of shape, a certain edge is admissible (i.e., collision-free for the full ARC consisting of \( N \) ARC-units), b) if this is not possible consider discrete \( \{ \psi_1^r \} \) azimuth rotations of that shape about the leading robot, c) if still no solution is found then search within LSC in such an order such that shapes that are faster to be achieved are evaluated first. For a solution to be admissible, the shape must not be in collision at the beginning of a certain edge, at the end, and during the path. Although LSC configurations tend to provide solutions in most cases, it is still possible that certain path edges remain inadmissible with any LSC shape. When such a condition appears, the method proceeds to evaluate if any of the configurations in SRS provides a solution for the current edge. In this process, solution admissibility is considered in the same sense as when evaluating LSC configurations. The overall steps are presented in Algorithm 4, while when no solution is found for certain edges, the flag \( \{ s_v^u \} \) returns false and thus the translation planner is re-invoked (a rare case) for the remaining path to the destination vertex \( v_{goal} \) (with associated lead robot location \( \xi_{goal} \)). Furthermore, the method MotionPlan reconstructs the path \( \sigma_{ARC} \) in the full configuration space \( \zeta \) from the edges \( E_{sol} \). Identification of the optimal path is based on minimizing the following cost-to-transition \( C_T \):

\[
C_T = \left\{ (N)(\psi_{1,e_1}^r - \psi_{1,e_2}^r)^2 + \sum_{i=1}^{N-1} (N - i)[(\psi_{i,e_1}^l - \psi_{i,e_2}^l)^2 + (\theta_{i,e_1}^l - \theta_{i,e_2}^l)^2] \right\}^{1/2}
\]  

(4.3)
where \( N \) is the number of ARC-units, \( c_1, c_2 \) are the initial and final shape configurations and \( \psi_i^l, \theta_i^l \) correspond to the yaw and pitch angles of link \( i \). It is noted that, during a transition between two vertices in the \( \zeta \)-Space, the robot first makes the shape transition (\( \eta \)-space) and then the translation transition (\( \xi \)-space).

Algorithm 4 Online Solution Construction

1: \( E_{sol} \leftarrow 0 \) edges \((v_T, u_T) \in \sigma_T\)
2: \( v_S \leftarrow \) starting vertex of current edge;
3: \( b_v^u = \) FALSE; \( \{\psi_\eta^l\} \) rotations \( \eta_{lsc} \in \mathbb{L}_{LSC} \)
4: \( u_S \leftarrow \) vertex for rotated \( \eta_{lsc}; \text{Admissible}(v_S, u_S, v_T, u_T)\)
5: \( E_{sol} \leftarrow E_{sol} \cup \{(v_C, u_C)\}; \)
6: \( b_v^u = \) TRUE; \( b_u^v = \) FALSE
7: \( s_v^u = \) TRUE;
8: \( \{\sigma_{\min}(v_T, u_T), s_v^u\} \leftarrow \text{SRSConnect}(v_S, (v_T, u_T)) \); \( s_v^u = \) FALSE
9: \( \sigma_T \leftarrow \text{TranslationPlannerQuery}(v_T, \text{goal}) \)
10: \( E_{sol} \leftarrow E_{sol} \cup \{\sigma_{\min}(v_T, u_T)\}; \)
11: \( \sigma_{ARC} = \text{MotionPlan}(E_{sol}); \)
12: \( \sigma_{ARC}; \)

4.3.6 Computational Complexity

The proposed algorithm is designed so as to provide computationally efficient solutions for the motion planning problem of complex multi-linked aerial robotic chains with multiple ARC-units. Therefore it relies on the a priori stored LSC and SRS. Then online, it further prioritizes attempts to find solutions through shapes that tend to satisfy most common environments stored in LSC, while SRS configurations are considered only when needed. As such, the steps that contribute online computational cost are the translation planner (Section 4.3.1), the SRS-based edge connection (only when invoked) (Section 4.3.4), and the online solution construction
4.3.5. Focusing on the most important cost factors, given an ARC-unit size modeled as a cube with length $D_R$, then, for the translation planner SampleFreeT is $O(V_M/V_{M,\text{free}} \times V_{D_R}/r_v^3 \times \log(V_M/r_v^3))$. Similarly, for the CollisionFreeT function, the cost is $O(V_{D_R}/r_v^3 \times d_{avg}/r_v \times \log(V_M/r_v^3))$, where $d_{avg}$ is the average edge length. Similarly, for the step of online construction of the solution and regardless if an LSC or SRS configuration is considered, the computational cost for the Admissible function given an average edge length $d_{avg}^c$ and $N$ ARC-units is $O(N \times V_{D_R}/r_v^3 \times d_{avg}^c/r_v \times \log(V_M/r_v^3))$.

A key question in terms of worst-case analysis is the number of shape configurations $N_{SRS}$ within SRS. Although the majority of cases are not searched, the absolute worst-case is to scale the previously mentioned computational cost term by $N_{SRS}$.

Lastly, during the SRS-based connection to build $G_{Val}$, a similar computational cost term holds for $N - 1$ robots (as at this point the head robot is not considered). This analysis emphasizes only on the dominant cost terms and thus it is partial though indicative of the factors that dominate in scaling the complexity of the problem given an Octomap-based occupancy-map \[30\] representation.

4.4 Evaluation Studies

In order to comprehensively evaluate the proposed motion planning algorithm for the Aerial Robotic Chain, four sets of simulation studies were conducted using the Gazebo-based RotorS Simulator \[29\]. First, we present two sets of studies that relate to the application of the ARCplanner in two room-like environments, one of which
involved a narrow window. In these studies we do not only present the end result of the ARCplanner but also evaluate the contribution of its components. A third study compares the ARCplanner with a full-state PRM thus demonstrating that given the high dimensionality of the underlying motion planning problem a direct application of PRM over the full state is not practical. Last, we present an application-driven demonstration of our approach. Recorded videos of these results are available at https://youtu.be/wPSMg9_YEh8.

4.4.1 Environment 1: Room with Window

In a first simulation study, two cases of ARC systems, namely with \( N = 4 \) and \( N = 5 \) ARC-units were commanded to navigate a room involving a narrow window and sub-spaces divided with a wall (“Environment 1”). The relevant simulation result for the case of \( N = 5 \) robots in the chain are presented in Figures 4.3 and 4.4. As depicted, the ARCplanner successfully commands the robot to change its shape and pass through the most narrow settings.

To allow us to study in detail the role of LSC and SRS in the solution resourcefulness of ARCplaner we conduct a specific study where ARCplanner either utilizes only LSC, only SRS or both. As this environment is structured in a manner that it has one narrow window whereas the rest is relatively free, we observe from the experiments that it is far easier for the robot to evaluate and decide on a shape from LSC when near relatively free regions. While this works in rather open regions, when ARC plans a path obliquely to a given window, it becomes fairly difficult for it to be
able to translate using the limited LSC shapes. In fact, it is found that the use of SRS is crucial for finding solutions in such challenging cases. However, as discussed in Section 4.3, the process of finding solutions through SRS is expensive (Section 4.3.4) since the planner spends significant time for a solution that can be arrived at easily, hence increasing the planning time. Thus as the conducted simulations show, it is the overall combined use of LSC and SRS that offers both the necessary planning resourcefulness and the computational efficiency for a planner capable of finding solutions in challenging environments within reasonable time. Table 4.1 summarizes the findings of this study indicating how the performance increases in case of combined LSC/SRS use versus each case alone. Within Table 4.1, $M_{VT}$ denotes the number of vertices in the translation planning $G_{T}$ graph, $N$ is the number of ARC-units in the considered ARC robot, $t_P$ is the overall planning time in seconds, and $t_{G_T} = 25.6 s$ is the computational time required to build this graph. The “LSC” column denotes the ARCplanner using only LSC and not SRS, the “SRS” column corresponds to the mirrored case, while “Both” stands for the full ARCplanner utilizing LSC and SRS.

Table 4.1: Key Statistics for Environment 1

<table>
<thead>
<tr>
<th></th>
<th>LSC</th>
<th>SRS</th>
<th>Both</th>
<th>LSC</th>
<th>SRS</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$M_{VT}$</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
</tr>
<tr>
<td>$N_{SRS}$</td>
<td>0</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>$t_P (s)$</td>
<td>11.3</td>
<td>23.0</td>
<td>4.4</td>
<td>46.1</td>
<td>54.4</td>
<td>4.4</td>
</tr>
</tbody>
</table>
4.4.2 Environment 2: Maze-like Room

In a second study in a maze-like room (Environment 2), the robot has to follow a zig-zag path. Being constrained in this manner, the robot has to find an appropriate shape-change path that allows it to cross such complicated surroundings. In this environment, since not enough space is available for the rear part of the robot to move freely, the LSC shapes which are mostly oriented linearly fail to provide solutions. Therefore, the planner uses SRS to find an appropriate solution. A relevant simulation study with $N = 5$ ARC-units is depicted in Figure 4.5. Furthermore, a comparative study analogous to the one conducted in the previous case when only using LSC, only SRS, or the full ARCplanner is summarized in Table 4.2. In this case, the time to build graph $G_T$ is $t_{G_T} = 9.4s$.

<table>
<thead>
<tr>
<th></th>
<th>LSC</th>
<th>SRS</th>
<th>Both</th>
<th>LSC</th>
<th>SRS</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$M_{VT}$</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
</tr>
<tr>
<td>$N_{SRS}$</td>
<td>0</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>$t_P (s)$</td>
<td>1.5</td>
<td>9.5</td>
<td>1.27</td>
<td>32.7</td>
<td>14.2</td>
<td>7.9</td>
</tr>
</tbody>
</table>

4.4.3 Comparison against full-state PRM

An alternative to the proposed approach would be to utilize a full-state PRM, utilizing Lazy evaluation [18], within which an online PRM would sample for the leading robot position, yaw and all the pitch and yaw Euler angles of the links in an effort to find collision-free paths for ARC. A relevant study was conducted based on Environment
1. The results are summarized in Table 4.3 where \( M_F \) stands for the vertices sampled in full-state PRM, \( t_G \) denotes the time (in seconds) for building the graphs needed in the different cases, and \( t_P \) the time (in seconds) spent for planning. As depicted, when the full-state PRM uses a number of vertices sampled leading to a time to build the graph and time to plan similar to that of the ARCplanner, it most often fails (80% of the time) to find a solution, while when we tune the number of vertices to safely find solutions it ends up requiring orders of magnitude greater computational cost. This result is aligned with the theoretical expectation for the cost of full-state PRM for multi-DoF systems.

### 4.4.4 Application-driven Demonstration

Finally, we present an application-driven simulation study where an ARC-unit is commanded to navigate among certain waypoints in a two-silo structure. The robot is commanded to optimize sensing coverage, thus prescribing the use of the Polygon Shape ("PN" configuration) when possible, and only employ shapes that minimize its cross-section when going through constrained regions (e.g., between the two silos). Figure 4.6 depicts the result.
Figure 4.2: Instances of the ARC shapes stored in the “Library of ARC Shape Configurations” (LSC). For the case of $N = 5$ with the joint at the robot’s Center of Gravity, the five possible configurations (LI, SE, SM, CA, PN) are presented, and indicative directions of the onboard frustum-constrained sensor of each robot are also depicted. For $N = 4$ and the joint beyond the center of gravity, the SV configuration is presented. At the bottom of the figure, an arc shape (CA) for $N = 4$ robots, a serpentine (SE) for $N = 4$, and a polygon (PN) for $N = 6$ are also presented. In our simulation studies, we have accounted for joint placements that enable the SV shape.
Figure 4.3: Instances of the Gazebo-based simulation of an ARC with $N = 5$ ARC-units navigating a room with a window based on the ARCplanner.

Figure 4.4: ARCplanner solution for an ARC with $N = 5$ ARC-units navigating a room with a window. The planner makes 9 changes in shape. Each robot is depicted with a coordinate frame, while alternating cyan and magenta paths are used between planning steps.
Figure 4.5: ARCplanner solution for an ARC with $N = 5$ robots navigating a maze-like room. The planner makes in total 10 changes in shape.

Figure 4.6: ARCplanner solution for $N = 6$ ARC-units navigating a two-silo structure and co-optimizing for sensor coverage [video: https://youtu.be/wP8Mg9_YEh8].
Chapter 5

Conclusions and Future Work

This thesis presents a new aerial reconfigurable robotic platform, dubbed ARC, consisting of multiple ARC-units connected with each other through rigid links. This multilinked system-of-systems of aerial robot aims to simultaneously achieve the ability to cross narrow sections, morph shape, ferry significant payloads, execute advanced work-task physical interaction, offer distributed sensing and computing, and enable redundancy alongside system extendability. Dynamic equations governing the system are derived with the assumption that the ARC-links are weightless, then a parallel control structure consisting of one MPC for position control and multiple $SO(3)$ angular controls for every ARC-links is proposed. We also provide detailed analysis for the upper limit of internal forces between the ARC-links and ARC-units given the proposed thrust-vector mixing method. The performance of the controller is demonstrated both in Simulink and RotorS simulations with $N$ ARC-units ($N = 3, 4, 5, 6$) as well as in real-world experiments with ARC-Alpha prototype consisting of $N = 2$ ARC-units.
Given the shape and position controller presented in Chapter 3, we present a path planning method in Chapter 4 to generate collision-free path for the ARC system to navigate in cluttered environment. The proposed planner addresses the challenge of planning in high dimensional configuration space by in several ways. First, we decouple the translation planner from the configuration (shape) planner. Second, for each path segment of the translation planning solution, we first check in the library of shape configurations if a stored shape solution exists that can be reached and traversed through that segment in a collision-free manner. Third, only if such a solution cannot be identified, we invoke a search within a set of random shape configurations sampled a priori (offline) through a Probabilistic RoadMap (PRM) based configuration planner that identifies novel shape configurations for ARC to facilitate collision-free navigation through the specified segment. Finally, if and only if all options of the local planner fail, we remove the specific segment from the solution and rerun the translation planner to identify new topologies in which possible ARC shapes can allow collision-free flight. Ablation studies demonstrate that combining both Library of Shape Configurations (LSC) and Set of Random Shapes (SRS) yields better planning time than using each of them only. Additionally, the proposed planner outperforms the full-state PRM in terms of planning time and success rate given the same planning density in fairly complex environments.

As far as future work is concerned, there are many potential directions. The current parallel control structure can be used as the expert for a neural network-based controller to imitate, then the learning-based controller can be further optimized with
reinforcement learning to improve the speed of changing shape. Since ARC is a complex system with multiple constraints (limit on internal forces, thrust vectors, etc), a new class of imitation and reinforcement learning algorithms with safety guarantee in design, verification and deployment phases need to be proposed. Recent works utilize Semidefinite Programming \[31\] or Constrained Markov Decision Process theory \[32\] demonstrate promising progress in this research direction. Furthermore, the local planner in the above path planning approach can be improved by considering the dynamic of the ARC system though differential flatness theory \[33, 34\]. By generating smoother trajectories that account for the system dynamic, more aggressive and efficient maneuvers can be achieved.
References


[23] M. Zhao, T. Anzai, F. Shi, K. Okada, and M. Inaba, “Path planning based on differential kinematics for passing through small opening by transformable multi-


