Development of a robotic arm towards aerial manipulation

Samuel Zeitler[1]

***Abstract* — This thesis continues the developmental work towards a dual-arm robotic system, specifically engineered towards aerial manipulation. Building upon the foundational work by Jaspar Sindermann, this work progresses from conceptual design to a fully operational robotic arm. The manufacturing relies on home-scale 3D printing, which enables rapid design iterations while also proposing mechanical challenges. The original design is adapted and a first version of the prototype is assembled. Subsequently a software-suite was developed in MATLAB. Core mathematical concepts like coordinate transforms, robot kinematics and control theory are combined into a modular framework, used for both visualization and control of the arm. The assembled prototype is successfully equipped for basic control of the end effector's position, proving the feasibility of the original concept. A series of task-specific tests are conducted to review the performance and provide a basis for future development.**

CONTENTS

I. INTRODUCTION

A. Overview of previous work

This thesis continues the foundational work done by Jaspar Sindermann in a preceding thesis. He created the initial concept and mechanical design for a robotic arm prototype designed for aerial manipulation (Fig. 1). The long-term ambition of this project is to engineer two identical and synchronized robotic arms, designed for mounting on drones, enabling them to execute complex dual-handed tasks.

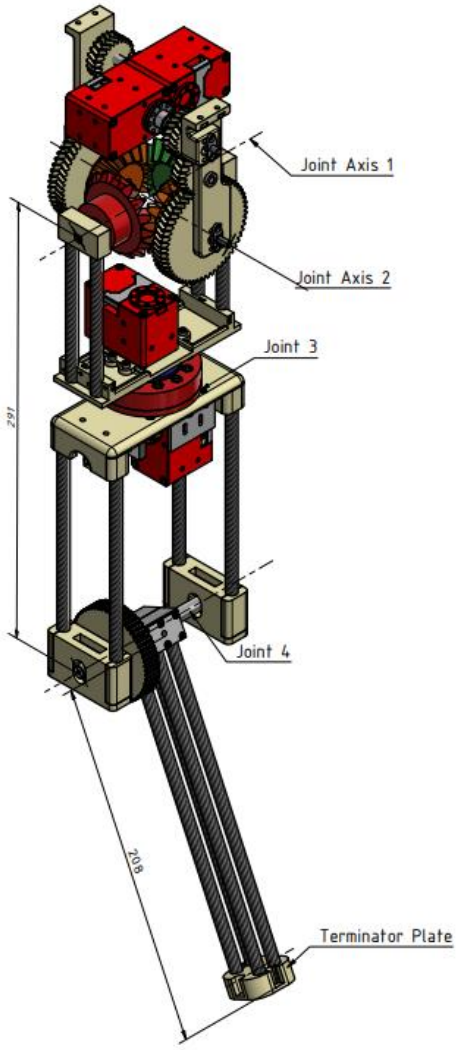

Fig. 1: Total view of the robotic arm as designed by Jaspar Sindermann [1]

Sindermann's design stands out for its dual-arm setup, with each arm having four degrees of freedom. This design choice was made to mimic the human arm and enable a wide range of manipulation tasks, from detailed manipulations to larger movements, while being airborne.

[1] Chair of Autonomous Aerial Systems, Department of Aerospace and Geodesy, Technical University of Munich, Munich, Germany, *samuel.zeitler@tum.de*

A key element of Sindermann's work lies in his focus on optimizing the kinematics. He used a modified inverse condition number as a new measure of dexterity, which helped guide the designed arm to be not just functional, but also optimize the workspace volume and manipulability.

In addition, Sindermann put the design through a thorough dynamic analysis. By testing multiple configurations, he made sure the prototype was a result of empirical analysis and iterative refinement. The design resulted in following specifications: an arm length of 0.5 meters, an estimated weight of 0.9 kilograms, and a commendable payload capacity of 0.2 kilograms. The full original design comprises a shoulder joint (Joint 1 and 2) followed by a yaw joint (Joint 3) and an elbow joint (Joint 4) displayed in Fig. 1. The Terminator Plate is defined as the end effector and should eventually mount a gripper.

The prototype design is mostly composed of 3D printed parts (e.g. gears and connectors) spaced apart by carbon fiber tubes. The axes are designed to be milled from aluminum, done locally at the TUM Department of Aerospace and Geodesy. The motors for each joint were chosen to be Dynamixel XH430 W210-T, a compact and lightweight servo with integrated position and velocity control [2].

B. Scope of this Thesis

Building upon the foundation laid by Jaspar Sindermann, this study delves into transforming a conceptual design into a functional prototype. The primary goal of this work is to verify that Sindermann's concept can be realized relying on home-scale equipment.

The secondary goal is to optimize the design of the prototype so that a second arm identical arm can efficiently be built in the future.

The thesis follows the development of the prototype in chronological order. Starting with the Manufacturing and Assembly (Chapter II), continuing with the Software Development (Chapter III) and ending with Testing and Review (Chapter IV). The ambition is to not only document the progress in the project but also to explore different topics, challenges and solutions in applied robotics.

II. Manufacturing and Assembly

This chapter aims to describe the building process of the robotic arm. The initial task involved cataloging the prints and components which were already manufactured by Sindermann. This process was crucial for identifying the starting point of the project and to identify missing elements or larger discrepancies.

It soon became evident that the original 3D printed parts did not align perfectly with their counterparts, especially the aluminum milled axes that demanded precision fits. The challenge was not just to identify these mismatches but to rectify them without compromising the integrity of Sindermann's kinematic and dynamic analysis. Almost all printed parts had to be updated and reprinted to achieve an acceptable fit, which was highly time consuming.

For simplicity, the decision was made to position the arm with its shoulder joint on a table facing upwards, deviating from its original downward-facing orientation intended for mounting on a drone. The assembly began at the bottom with the shoulder joint and continued upward to the end effector.

A. Shoulder Joint

The process of manufacturing the robotic arm started with the shoulder joint, since it was deemed to be the most straightforward standalone assembly to build. This selection allowed for a manageable starting point, enabling the isolation and understanding of potential challenges within a contained component.

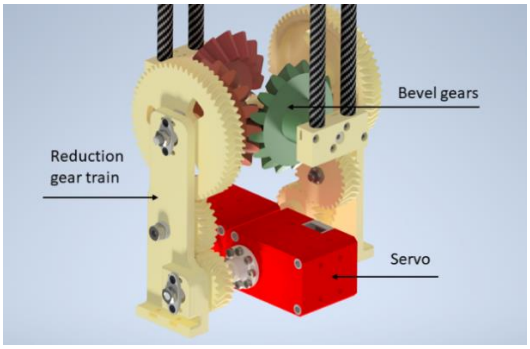

Fig. 2: Shoulder Joint
From the original design by Jaspar Sindermann

The first core component of the Shoulder Joint is the reduction gear train located on the sides of the shoulder joint. The reduction assembly serves as the bridge connecting the servos on the bottom to the bevel gears which are mounted on aluminum axes on the top. Both reduction assembly provide an amplification of the motor torque of $i_{gearTrain} = 2.5$ resulting in a total gearing ratio of $i_{shoulder} = 5$.

The second core component of the Shoulder Joint is the Bevel gear assembly, which creates two rotational axes crossing in the center of the bevel gears (Fig. 3).

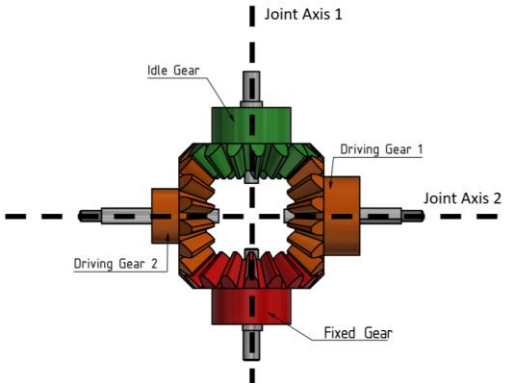

Fig. 3: Bevel gear assembly

Synchronous movement of the driven gears results in a tilt around the driven gears axis (Joint Axis 2). Asynchronous movement of the driven gears results in a tilt around the orthogonal axis (Joint Axis 1). This allows the shoulder joint to point in any direction of the upper hemisphere while having to maintain a minimum elevation of $\varphi_{min} = 45°$ to horizontal plain to avoid self-collision.

During assembly, the biggest challenge was to fix the gears on their metal axes in such a way that they would sit

firmly and still be mountable by hand. This is important so that the movement of the shoulder would precisely follow the rotation of the servos without mechanical play. To achieve this, all the contact surfaces of the 3D printed parts had to be redesigned with minimal tolerance, often requiring several iterations to create a satisfactory friction fit. This proved challenging throughout the whole assembly phase making loose gears the most common failure points of the prototype.

At the bottom of the shoulder gear, an acrylic mounting plate was added to serve as a base for the whole arm. The servos and reduction assemblies where fixed on the mounting plate while allowing the distance between them to be adjustable.

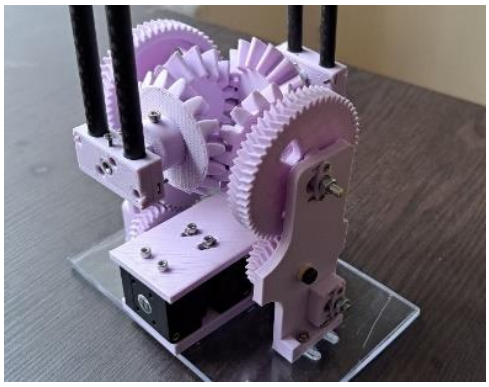

Fig. 4: Complete Shoulder Joint with Servos Mounted on an acrylic plate fixed on the table

Following the completion of the build, the mechanical performance of the shoulder joint was assessed, with an emphasis on the backlash-free operation of the gears and the overall stability of the joint. Notably, the verification of the top joint's functionality was a significant milestone in the development process, confirming the feasibility of home scale 3D printing as a primary manufacturing technique and paving the way for the subsequent stages.

Although the movement of the joint was deemed satisfactory considering the sole reliance on 3D printed parts, one key drawback of the design became evident. The bevel gear assembly, consisting of the four facing gears in the center, must be pressed together with enough clamping force to support the entire weight of the arm and payload. In the original iteration of the joint, the gears barely provided any resistance towards axial forces, due to the elastic nature of the 3D printed parts. This resulted in an unstable joint that would easily come apart under any load. To cope with this issue, the back plate of the reduction gear trains was reinforced with a higher infill ratio and bigger side length. Additionally, a tightly wrapped tensioning tape was provisionally added around the carbon fiber tubes to create even more clamping force between the gears (Fig. 5). The result was that the joint could now support approximal 2 kg of weight while maintaining operation. While this is nowhere near an optimal solution, it was deemed to be an acceptable solution for a first prototype.

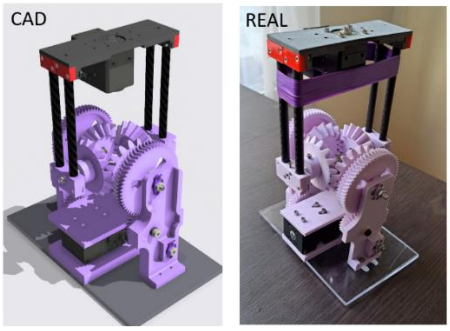

Fig. 5: Shoulder Joint
Tensioning tape provides clamping force for the bevel gears

B. Yaw Joint

The yaw joint connects the middle segment of the arm to the shoulder joint. It allows for rotational movement around the arm's central axis, while carrying the radial and axial load. The original design of the yaw joint intended the use of an intricate cycloidal gearbox to amplify the motor torque. However, for the prototype the use of a plain bearing was deemed more suitable. To validate this major change in design we need to revisit Sindermann's dynamic requirements for the arm.

1. The arm should be able to hold its own weight and a payload of $m_{PL} = 0{,}2$ kg in any joint configuration.
2. The arm shall be able to exert a force of $F_{Test} = 3$ N in any joint configuration.
3. Each joint $i \in 1 \dots k_{\text{joints}}$ shall be capable of providing the maximum design torque $\tau_{\text{max},i}$ at a rotational velocity so that the resulting end effector velocity v_e is at least $v_e = 0{,}15 \frac{\text{m}}{\text{s}}$ if all following joints are at a position $q_{i+1,\dots,k} = 0$.

Based on these requirements, Sindermann deduces the following total required joint torques:

[Nm]	Shoulder 1	Shoulder 2	Yaw	Elbow
$\tau_{\text{dyn,max}}$	0,03	−0,10	0,57	2,67
τ_{grav}	2,33	2,21	1,18	0,63
τ_{test}	1,5	1,5	0,63	0,63
$\boldsymbol{\tau_{\text{total}}}$	**3,86**	**3,61**	**2,38**	**3,93**

The purchased servos have a sufficient stall torque of 3.1 Nm (at 14.8 V, 1.5 A), so a transmission is not strictly needed to meet the dynamic requirements [2]. Additionally, the suggested cycloidal gearbox turned out to be particularly hard to manufacture because of its intricate design. Consequently, it was decided to simplify the prototype yaw joint to a basic 3D printed axial bearing (Fig. 6).

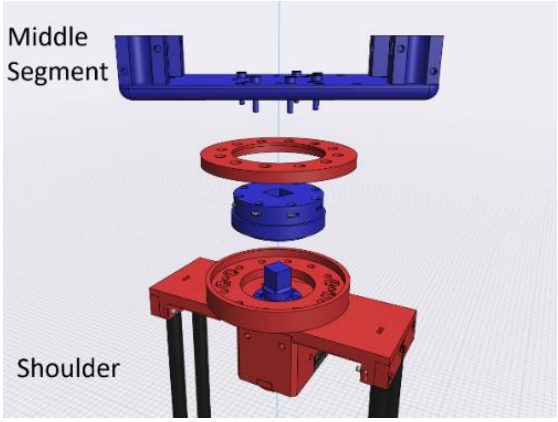

Fig. 6: Simplified Yaw Joint
Parts moving together have the same color

The arm's middle segment (colored blue) is attached to the inner moving part of the plain bearing. This setup essentially creates a transmission for the motor torque, offloading the axial and radial forces from the servo to the housing (colored red).

C. Elbow Joint and Final Assembly

The elbow joint serves as the last joint in the arm's kinematic chain, allowing for flexion and extension movements. In revising the original design, only a minor modification was made to the elbow joint: the size of the pulley gear was increased and it was fitted with a rim to prevent the belt from slipping off. This modification altered the effective transmission from $i_{elbow} = 4$ to $i_{elbow} \cong 2.5$ which leads to $\tau_{elbow,max} \geq 4\ Nm$, which still surpasses the required torque. This was deemed necessary to enhance the reliability and functionality of the joint. No alterations were made to the end effector segment, maintaining its original design specifications.

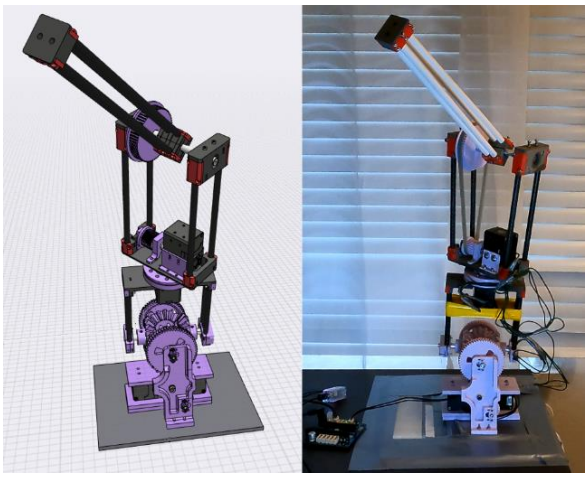

Fig. 7: Complete Prototype
Additional rims hold the belt in place

The final assembly phase included the electrical wiring of the arm. Careful planning was employed in the wire routing to both minimize mechanical interference and ensure an orderly arrangement. All servos were linked in a daisy chain configuration, utilizing the TTL communications protocol for connectivity [4]. A USB-to-TTL adapter provided the interface between the servos and a host PC. In addition, each servo is equipped with an integrated MCU and multiple sensors, including an absolute position encoder.

The final assembly of the prototype is depicted in Fig. 7.

III. Software Development

The software's primary task is to control the arm through a host-PC. To achieve this, the chapter covers robot kinematics, software development, and control theory. A theoretical look at the kinematics is followed by the development of a custom software framework for the robotic arm. The framework contains a kinematic simulation of the arm, housing forward and inverse kinematic methods as well as a 3D visualization. Additionally, an interface between the host pc running MATLAB and the servos is necessary. MATLAB was chosen to be the software environment as it features easy visualization, strong mathematical capabilities and a precompiled library for the used Servos. The full code of this project is publicly available in the GitHub Repository [3].

A. Kinematic foundations

The coordinate system for the robotic arm is defined in Fig. 8. The arm features four joint axes, labeled from 1 to 4 with rotation in the mathematically positive direction. While the first joint axis is consistently aligned with the global y-direction, the remaining axes are kinematically dependent. These joints form a kinematic chain that originates at the global origin (O) and culminates at the end effector, also referred to as Tool Center Point (TCP). The global origin is defined atop the acrylic mounting plate, below the intersection of the first and second joint axes.

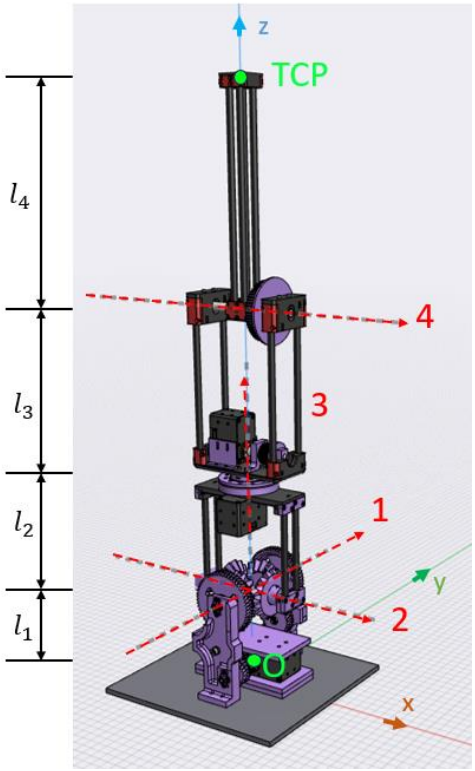

Fig. 8: Arm Coordinate System

The four rotational degrees of freedom (DOF) of the arm are arranged in a y-x-z-x sequence. Although a 3-DOF configuration would suffice for positioning the end effector within the workspace, the inclusion of a fourth DOF enables the arm to approach the same point from multiple orientations. This added flexibility enhances the arm's dexterity and optimizes the utilization of its workspace. Sindermann's design, including the link lengths l_1 to l_4, was informed by a comprehensive kinematic analysis, elaborated in his thesis [1].

Forward kinematics

The position of the TCP in global coordinates (g) is denoted as ${}_g\boldsymbol{x}_{TCP}$ and derived from joint angles $\mathbf{q}$ using a chain of elementary transformation matrices:

$$ {}_g\boldsymbol{x}_{TCP} = \boldsymbol{R_1} * (\boldsymbol{R_2} * (\boldsymbol{R_3} * (\boldsymbol{R_4} * \boldsymbol{x_4} + \boldsymbol{x_3}) + \boldsymbol{x_2}) + \boldsymbol{x_1}) + \boldsymbol{x_0}\,. \quad (1) $$

With elementary transformation matrices denoted like:

$$ \boldsymbol{R_1} \mathrel{\widehat{=}} {}_g\boldsymbol{R_y}(q_1)_1 = \begin{pmatrix} cos(q_1) & 0 & sin(q_1) \\ 0 & 1 & 0 \\ -sin(q_1) & 0 & cos(q_1) \end{pmatrix}. \quad (2) $$

And relative position vectors denoted like:

$$ \boldsymbol{x_4} \mathrel{\widehat{=}} {}_4\boldsymbol{x}_{4E} = (0 \quad 0 \quad l_4)^T. \quad (3) $$

The rotation of the TCP can be represented as a rotation matrix defined in global frame, which represents the orientation of the end-effector with respect to the unrotated origin frame:

$$ {}_g\boldsymbol{R}(\boldsymbol{q})_E = \boldsymbol{R_1} * \boldsymbol{R_2} * \boldsymbol{R_3} * \boldsymbol{R_4}\,. \quad (4) $$

Together, the position and rotation define a coordinate frame. The equations and notation are elaborated in [5] and form the basis of the kinematic simulation in subchapter B.

Inverse Kinematics

The inverse kinematics formulate the basis for the intended position control of the TCP in subchapter C. Starting from the forward kinematic equation:

$$ {}_g\boldsymbol{x}_{TCP} := \boldsymbol{f}(\boldsymbol{q}). \quad (5) $$

Derivation w.r.t time gives:

$$ \dot{\boldsymbol{x}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \underbrace{\left(\frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\boldsymbol{q}}\right)}_{\boldsymbol{J}(\boldsymbol{q})} \dot{\boldsymbol{q}}. \quad (6) $$

Solving for $\dot{\boldsymbol{q}}$:

$$ \dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})^{-1}\, \dot{\boldsymbol{x}}. \quad (7) $$

This equation states how a desired velocity at the TCP $\dot{\boldsymbol{x}}$ is achieved by the setting the joint velocities $\dot{\boldsymbol{q}}$ accordingly.

In the case of 4 joints, the Jacobian $\boldsymbol{J}$ is of dimension $\mathbb{R}^{3\times4}$. The excess DOF ($\boldsymbol{q} \in \mathbb{R}^{\boldsymbol{4}} > {}_g\boldsymbol{x}_{TCP} \in \mathbb{R}^3$) generally results in infinitely many possible solutions to the inverse kinematics problem and $\boldsymbol{J}(\boldsymbol{q})^{-1}$ does not exist. Classically the Moore-Penrose-Inverse is used instead:

$$ \boldsymbol{J}(\boldsymbol{q})^{-1} := \boldsymbol{J}(\boldsymbol{q})^{+} \quad (8) $$

Utilizing the Moore-Penrose-Inverse implicitly optimizes $\dot{\boldsymbol{q}}$ to minimize the Euclidean norm $\| \dot{\boldsymbol{q}} \|_2$. This means it finds the "smallest" joint velocity vector that achieves the desired end-effector velocity [6].

Notably, the Jacobian can become rank-deficient in certain singular configurations of the arm. In such cases, the system loses its ability to move the end-effector in specific directions, leading to unbound joint velocities. This demands the explicit handling of singular positions (e.g. a fully stretched out arm) during operation [7].

B. Kinematic Simulation in MATLAB

Utilizing the derived equations, a kinematic simulation of the arm was developed. The simulation is intended to mirror the real arm, equipping it with predeveloped methods for forward and inverse kinematics.

While many libraries already exist that manage the relationship between dependent coordinate frames (e.g. TF for ROS [8]), it was decided to instead self-develop a small framework for the simulation. This approach was chosen to provide an independent and comprehensible code base tailored towards the prototype.

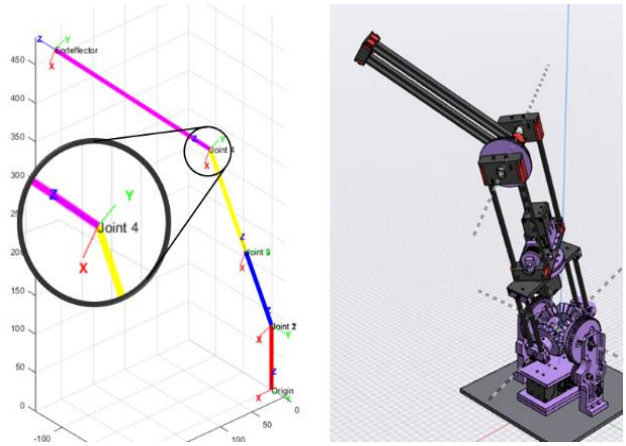

Fig. 9: Simulated Robot
MATLAB (left) and CAD (right), Frame of Joint 4 magnified

The simulation is comprised of the following classes:

- The *SimulatedRobot* class defines a robotic arm containing Coordinate-Frames, Joints and Links. It provides the *forwardKinematic* method, which returns the current position of the end effector and the *getJacobian* method, which returns the current Jacobian. These methods will later be reused for the real prototype.
- The *Frame* class is the core concept of the simulation. A series of Frames form a kinematic chain, executed as a sequence of parent-child relationships, commencing from a fixed Origin. Every Frame is defined by the position relative to its parent and a rotational matrix denoted in the global Frame (${}_g\boldsymbol{R}_f$). The *rotate* method rotates the Frame around a given local axis alongside all its descendants.
- The *Joint* class inherits from *Frame* and defines a fixed rotation axis. It confines the *rotate* method to only rotate about its joint axis (e.g. Joint 3 only rotates around its local z-Axis)

- The *Link* class simply connects two Frames or Joints, visualized with a different color for every link segment.

The following UML-Class diagram shows the relationship between the classes (Fig. 10).

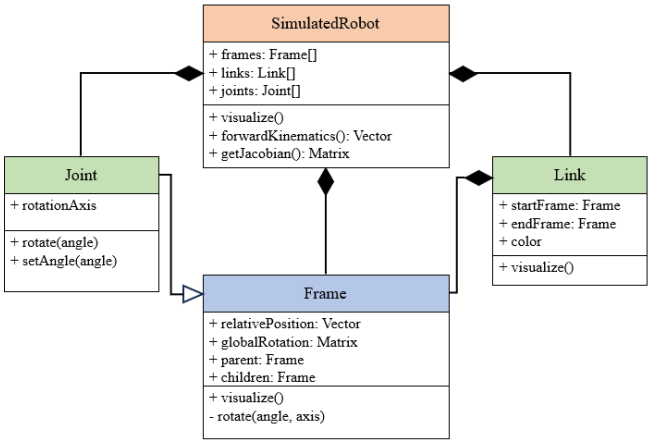

Fig. 10: UML-Diagram of the Kinematic Simulation in MATLAB

Frame rotation:

The *rotate* method was the most challenging concept to implement and shall be discussed in the following paragraphs:

Each Frame **f** stores their rotational matrix denoted in the global Frame ${}_g\boldsymbol{R}_{\mathrm{f}}$. This trivializes the visualization of the Frame (Fig. 9) since the basis axes of **f** are the columns of ${}_g\boldsymbol{R}_{\mathrm{f}}$.

After a rotation, ${}_g\boldsymbol{R}_{\mathrm{f}}$ must be updated alongside all descendants ($\boldsymbol{f_{desc}}$, ${}_g\boldsymbol{R}_{\boldsymbol{f_{desc}}}$). As **f** can only rotate around one of its basis axes with angle α, ${}_g\boldsymbol{R}_{\mathrm{f}}$ can be updated with by post-multiplying the basic 3D rotational matrices $\boldsymbol{R}_{x,y,z}$:

$$ {}_g\boldsymbol{R}'_f = {}_g\boldsymbol{R}_f * \boldsymbol{R}_{x,y,z}(\alpha). \qquad (9)$$

Unfortunately, this cannot directly be used to rotate the descendent Frames because the post-multiplication of $\boldsymbol{R}_{x,y,z}(\alpha)$ would result in a rotation in the descendent Frames coordinate system [9]. To perform a kinematically correct rotation, the dependent Frames must be rotated around the parents' rotation axis. Thus, a pre-multiplied $\boldsymbol{R}$ is calculated that represents the correct rotation in global coordinates, applying to all Frames:

$$ {}_g\boldsymbol{R}'_f = \boldsymbol{R} * {}_g\boldsymbol{R}_f. \qquad (10)$$

Equating (9) and (10) yields:

$$ {}_g\boldsymbol{R}_f * \boldsymbol{R}_{x,y,z}(\alpha) = \boldsymbol{R} * {}_g\boldsymbol{R}_f $$

$$ \rightarrow \boldsymbol{R} = {}_g\boldsymbol{R}_f * \boldsymbol{R}_{x,y,z}(\alpha) * {}_g\boldsymbol{R}_f{}^{-1}. \qquad (11)$$

The inverse is simplified using the orthogonality property of rotational matrix ${}_g\boldsymbol{R}_f$:

$$ \boldsymbol{R} = {}_g\boldsymbol{R}_f * \boldsymbol{R}_{x,y,z}(\alpha) * {}_g\boldsymbol{R}_f{}^{T}. \qquad (12)$$

Finally, pre-multiplying $\boldsymbol{R}$ is used to update **f**:

$$ {}_g\boldsymbol{R}'_f = \boldsymbol{R} * {}_g\boldsymbol{R}_f \qquad (13)$$

and $\boldsymbol{f_{desc}}$ recursively for all descendent Frames:

$$ {}_g\boldsymbol{R}'_{\boldsymbol{f_{desc}}} = \boldsymbol{R} * {}_g\boldsymbol{R}_{\boldsymbol{f_{desc}}}. \qquad (14)$$

Resulting in a kinematically correct rotation of the kinematic chain.

Efficient calculation of the Jacobian

Equation (6) shows an expression $\boldsymbol{J}(\boldsymbol{q})$ for the Jacobian can be found by symbolic derivation of the forward kinematic equations. However, the Jacobian for the current configuration must be numerically calculated at every simulation timestep by substituting in the values for the current joint angles, which is computationally expensive.

Alternatively, the Jacobian can be approximated differently. In this approach, the Jacobian is computed based on the concept of approximating derivatives using small perturbations [10]:

1. For each joint angle q_i, perturb it by a small amount Δq to get $q_{i+} = q_i + \Delta q$ and $q_{i-} = q_i - \Delta q$.
2. Compute the end effector positions for these perturbed joint angles using the forward kinematics $\boldsymbol{f}$:

$$ \boldsymbol{x}^{+}_{TCP_i} = \boldsymbol{f}(q_{i+}) $$
$$ \boldsymbol{x}^{-}_{TCP_i} = \boldsymbol{f}(q_{i-}) $$

3. Approximate the derivative $\frac{\partial_g \boldsymbol{x}_{TCP}}{\partial q_i}$ as:

$$ \frac{\partial_g \boldsymbol{x}_{TCP}}{\partial q_i} \approx \frac{\boldsymbol{x}^{+}_{TCP} - \boldsymbol{x}^{-}_{TCP}}{2\Delta q} $$

4. Populate the Jacobian matrix $\boldsymbol{J}$ by setting its i^{th} column to $\frac{\partial_g \boldsymbol{x}^{+}_{TCP}}{\partial q_i}$:

$$ \boldsymbol{J} = \left[\frac{\partial_g \boldsymbol{x}^{+}_{TCP}}{\partial q_1}, \frac{\partial_g \boldsymbol{x}^{+}_{TCP}}{\partial q_2}, \dots, \frac{\partial_g \boldsymbol{x}^{+}_{TCP}}{\partial q_n} \right] $$

Measuring the computation time shows that the numeric approach is about 40 times faster than the symbolic substitution and should therefore be used for the end effector control.

C. Control interface

After laying down the theoretical groundwork and developing the kinematic simulation, the next step was to bring the robot to life in a practical setting. The communication between MATLAB running on the main computer and the attached servos is facilitated by the DynamixelSDK library [11]. Written in C, this library is used for communication with the servos using a serial connection.

The Dynamixel XH430-W210 Servos that were acquired come with several built-in control modes, which are as follows:

- Mode for Current Control
- Mode for Velocity Control
- Position Control Mode (from 0° to 360°)
- Extended Position Control Mode (Multi-turn)
- Current-Driven Position Control Mode
- PWM Control Mode

Ideally, torque control would be the most suitable, as it enables the robotic arm to adapt and respond flexibly when interacting with its surroundings, such as during grasping activities. However, the XH430-W210 model lacks a built-in torque sensor. As a result, torque can only be approximated based on the motor's current, which may not be entirely accurate. For the time being, the prototype will operate using the velocity control mode (Fig. 11).

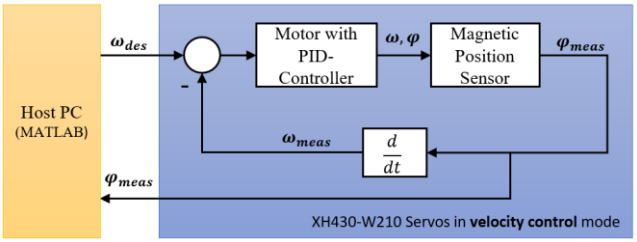

Fig. 11: Velocity Control of the Servos

Here $\boldsymbol{\omega}$ is the vector of angular velocities of the servos and $\boldsymbol{\varphi}$ is the angle of the servos.

The developed simulation already provides kinematic methods as well as a visualization. In order to use these with the real robotic arm, the joint angles of $\boldsymbol{q}$ must be obtained from the measured servo angles angles $\boldsymbol{\varphi_{meas}}$[2]. Then the configuration of the real robot could be visualized using the converted servo angles (Fig. 12).

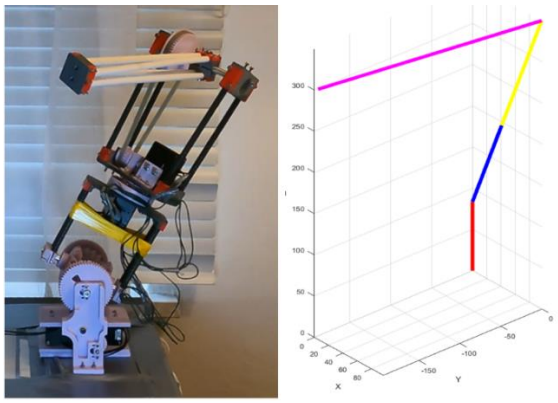

Fig. 12: Real arm with visualization in MATLAB Using the converted servo angles

Angle conversion $\boldsymbol{q(\varphi)}$:

In the case of the Yaw and Elbow Joint (Joint 3 and Joint 4), the relationship between $\boldsymbol{\varphi}$ and the $\boldsymbol{q}$ is straightforward:

$$q_3 = \varphi_3 - \varphi_{3,0}$$

$$q_4 = -(\varphi_4 - \varphi_{4,0})/i_{elbow}$$

With offsets $\varphi_{x,0}$ defined such that $\mathbf{q} = \mathbf{0}$ results in a fully stretched out arm. In the case of the Shoulder Joint (Joint 1 and Joint 2), the relationship is non-trivial. Due to the bevel gear configuration, each Joint angle is influenced by both servos. The servo located in x-direction in the arm coordinate system () is denoted with subscript 1:

$$\Delta\varphi_1 = (\varphi_{1,0} - \varphi_1)$$

$$\Delta\varphi_2 = (\varphi_{2,0} - \varphi_2)$$

$$q_1 = (\Delta\varphi_1 - \Delta\varphi_2)/i_{shoulder}$$

$$q_2 = -(\Delta\varphi_1 + \Delta\varphi_2)/i_{shoulder}$$

Velocity conversion $\boldsymbol{\omega(\dot{q})}$:

In Order to use the Jacobian calculations of the developed simulation, the joint velocities $\dot{\boldsymbol{q}}$ need to be converted to the servo velocities $\boldsymbol{\omega}$. The conversion follows from derivating $\boldsymbol{q(\varphi)}$ w.r.t time:

$$\omega_3 = \dot{q}_3$$

$$\omega_4 = -\dot{q}_4 * i_{elbow}$$

Derivation of the formulas for the shoulder joint results in:

$$\omega_1 = \frac{1}{2} * (\dot{q}_2 - \dot{q}_1) * i_{shoulder}$$

$$\omega_2 = \frac{1}{2} * (\dot{q}_2 + \dot{q}_1) * i_{shoulder}$$

End effector control interface

By combining the conversion functions $\boldsymbol{q(\varphi)}$ and $\boldsymbol{\omega(\dot{q})}$ with the forward kinematics $\boldsymbol{f(q)}$ and inverse kinematics $\boldsymbol{J(q)^{-1}}$ from Subchapter A, the end effector control interface can be formulated in following diagram:

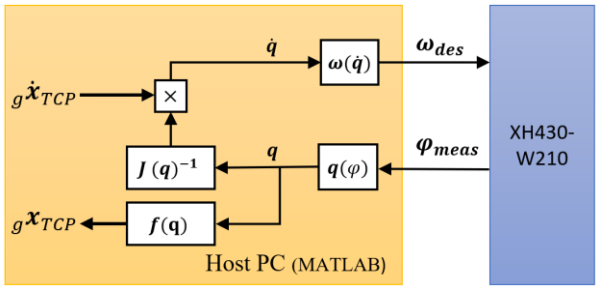

Fig. 13: end effector control interface

The output ${}_g\boldsymbol{x}_{TCP}$ represents the current position of the end effector in global coordinates, obtained from the measured servo angles. The input ${}_g\dot{\boldsymbol{x}}_{TCP}$ is used to command a desired joint velocity, utilizing the calculated Jacobian.

D. End effector position control

The last step in achieving precise control over the end effector's position is to develop a control loop. The task is to move the end effector from any current position ${}_g\boldsymbol{x}_{TCP}$ to a desired goal position ${}_g\boldsymbol{x}_{Goal}$. The control variable is the end effector velocity ${}_g\dot{\boldsymbol{x}}_{TCP}$. This defines a classic control problem that can be solved using a simple PID controller.

[2] In fact, in velocity control mode, the servos output $\boldsymbol{\varphi_{meas}}$ as values between -1,048,575 and 1,048,575. These values are multiplied by the factor 0.0015 to obtain the current servo angles $\boldsymbol{\varphi}$ in radiant. The angles are tracked over max. 512 rotations and the zero position is arbitrary.

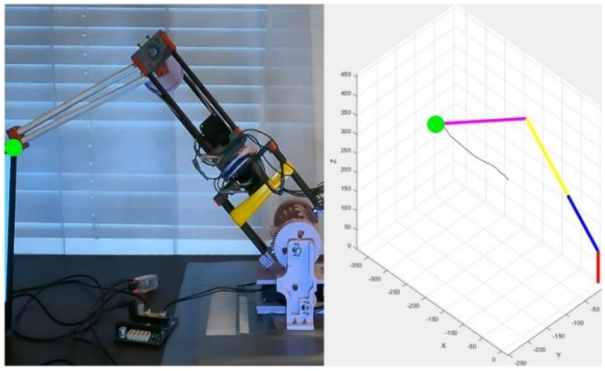

Fig. 14: End effector position control using a PID-Controller
Visualized goal position (green) and trajectory (black)

If the target position is beyond the arm's reach, the existing control algorithm extends the arm to its maximum length. This stretched position, known as a singularity, can cause mathematical issues as the inverse Jacobian diverges. To counter this, the algorithm constantly monitors the *condition* number of the Jacobian throughout the movement [12]. Should this number exceed an empirical threshold, indicating the system is near a singularity, the controller stops the motion:

$$cond(\boldsymbol{J}) = \| \boldsymbol{J} \| \cdot \| \boldsymbol{J^{-1}} \| < threshold\ value$$

With:

- $\| \boldsymbol{J} \|$ is the norm of the matrix $\boldsymbol{J}$, calculated as the largest singular value[3] of $\boldsymbol{J}$.
- $\| \boldsymbol{J^{-1}} \|$ is the norm of the inverse of $\boldsymbol{J}$, calculated as the largest singular value of $\boldsymbol{J^{-1}}$.

Additionally, the elevation angle θ of the shoulder joint is determined through trigonometric calculations. If θ falls below 45 degrees, there is a risk of the arm colliding with itself, leading to another scenario where the controller stops the action.

$$\theta(\boldsymbol{q}) = \frac{\pi}{2} - cos^{-1}(cos(q_2) \cdot cos(q_1)) > \frac{\pi}{4}$$

While preventing catastrophic failures, the safeguards also restrict the arm from accessing its full range of motion.

IV. Testing and Review

The last chapter of this thesis is focused on formulating an objective review of the results and highlighting areas in need of improvement. The thesis concludes with an outlook on the continuation of the project

A. Basic Functional Testing

For the initial testing of the prototype, a small number of simple tasks were developed. For the first task the torque was disabled for all servos, meaning the joints would not provide resistance to external forces. This allowed the arms end effector to be guided by hand. The task for the robot was to correctly draw the hand-navigated path in the built-in visualization. As the measured servo angles are the only available information, the calculated path of the end effector has to match the actual path if the conversion of the servo angles to joint angles (Chapter III C) and the forward kinematics (Chapter II A) had to be correct.

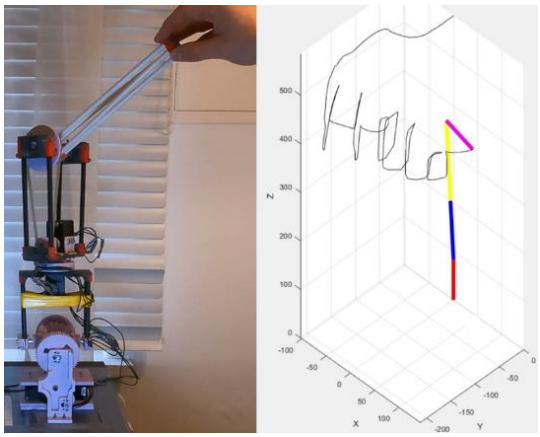

Fig. 15: Robot guided by hand
End effector path calculated from the measured servo angles

The word “Hello” was drawn in the air, and the robot accurately calculated the end effector path. While this test doesn't shed much light on the absolute accuracy of the calculated path, it does validate the correctness of the used forward kinematic equations.

Up next, the speed control of the end effector was to be tested. For this, the desired velocity was defined via user input. If the user was able to control the velocity of the end effector, the conversion from desired joint velocities to servo velocities as well as the inverse kinematic methods had to be correct.

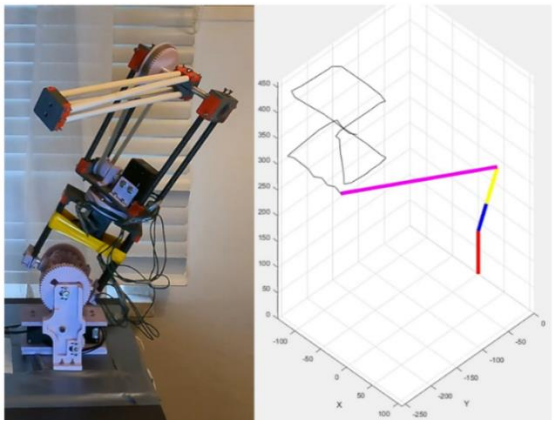

Fig. 16: Robot controlled with keyboard

As shown in Fig. 16, where the user instructed the end effector to draw a cube, the implementation appears correct. The most significant problem was linked to the Moore-Penrose-Inverse of the Jacobian. The calculated joint velocities correctly result in the desired end effector velocity, but the configuration of the robot is ignored. This causes the robot to move into unfavorable configurations that might exceed the minimum elevation limit of the shoulder joint or approach singularities.

[3] The largest singular value of $\boldsymbol{J}$ is calculated as the square root of the maximum eigenvalue λ_{max} of Matrix $\boldsymbol{M}$.
With square Matrix $\boldsymbol{M}$ being the product of $\boldsymbol{J}$ with its conjugate transpose $\boldsymbol{J^H}$.

B. *Laboratory testing using motion capturing*

For a more detailed analysis of the robotic arm's movement, the robot was placed inside the *Flight Lab* of the supervising Chair for *Autonomous Aerial System* at TUM Ottobrunn. Prof. Markus Ryll kindly assisted in setting up the motion capture environment for the robotic arm. To track the movement of the robot's end effector, a constellation of infrared markers was attached (Fig. 17).

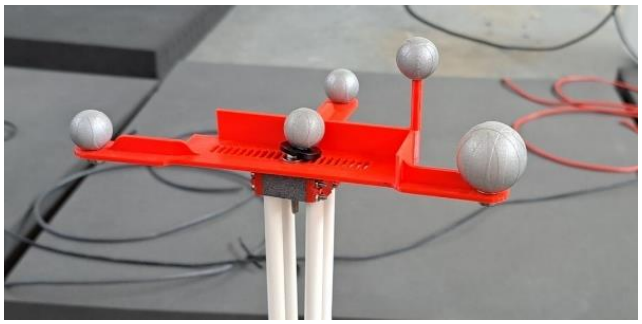

Fig. 17: End effector with attached tracking markers

Utilizing the position control algorithm outlined in Chapter III D, the robotic arm was programmed to navigate a circle. The circumference was delineated using 10 waypoints within the global coordinate system. The robot approached the points one after another, forming the circular pattern in Fig. 18. The end effector trajectory was calculated based on the measured servo angles and compared to the positions recorded by the motion capture system.

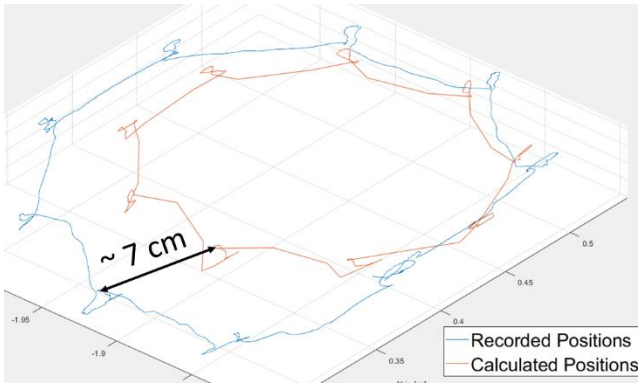

Fig. 18: End effector trajectory (recorded vs. calculated)

The positions that form the intended circular path are easily discernible as spirals appearing around the circumference. At these points, the robotic arm switches from one target position to another. The starting and stopping motions cause slight fluctuations in the end effector's movement.
As the robotic arm successfully completes the circular path, significant position errors become evident with a maximum of 7 cm. These error increases as the arm's center of gravity shifts away from its mounting point during motion. This generates an amplified moment on the shoulder joint, especially when the arm is fully extended. Consequently, this leads to a discernible sag in the robotic arm, causing the end effector to be positioned lower than intended. A measurement of this phenomenon is depicted in Fig. 19.

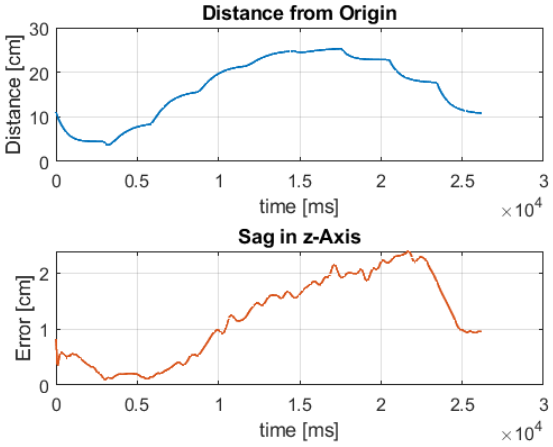

Fig. 19: Correlation between distance from origin and sag

The identified reason for the discrepancy is the mechanical play in the arm, especially in the bevel gear configuration of the shoulder joint.

C. *Starting Point for Optimization*

This subchapter outlines potential areas for improvement in the robotic arm prototype.

Mechanical Review

1. *The assembly process* of the robotic arm prototype presents the first set of challenges. The complexity of the assembly, involving a multitude of parts and hard-to-mount modules, often makes the process cumbersome and error-prone. To alleviate this, a design review focused on simplification is recommended. Specifically, the redesign should aim to make screwing connections more accessible, thereby streamlining the assembly process.
2. *The interface between plastic parts and aluminum axes* is an area of issue. The plastic parts must be printed with perfect tolerance to ensure a tight fit, a condition that is difficult to maintain over time. Moreover, the small contact area tends to fail when transmitting large torques. To address this, the area, should be increased. The use of ball bearings should also be discussed for better guidance behavior.
3. *The bevel gear of the shoulder joint* is particularly problematic as the primary source of sagging. It's recommended to experiment with various bevel gear types, as well as gears with larger diameters and more teeth. A more structured method of providing clamping force, should replace the current tensioning tape. A comparable solution was developed in [14].
4. *The mechanical testing* should be intensified, especially regarding the arm's load-bearing capacity. Currently the arm has only been tested in an inverted configuration without a gripper or loads. As the arm is supposed to be suspended from a drone, constructing a mounting stand would result in more realistic testing is recommended.

Control Review

5. *The inverse kinematics algorithm* currently employed has several limitations. It does not account for the configuration of the arm, which can lead to orientation issues, especially given the arm's limited Degrees of Freedom. Moreover, the algorithm overlooks singularity configurations and joint limits, posing a risk of self-collisions during complex maneuvers. This challenge is tied to the Moore-Penrose pseudoinverse of the Jacobian which minimizes joint velocity but fails to consider the robots configuration Advanced methods such as trajectory optimization or motion planning can navigate these issues. A practical example is the open motion planning library (OMPL) for sampling-based motion planning [13].
6. *The end effector control* currently relies on a position-based PID-Controller. This approach restricts the system's ability to follow trajectories, requiring any path to be broken down into a series of discrete points. To overcome this limitation, the development of a controller capable of following continuous trajectories is recommended, an example can be found in [15].
7. *The control of the servos* is velocity-based, omitting any information about the joint torque. This makes the arm unresponsive to external forces and prevents it from exhibiting compliant behavior. Experimentation with current-based torque estimation for the servos is advised, along with the exploration of a dynamic model for the robot.

V. Outlook

In this thesis, the original blueprint of Jaspar Sindermann has been seamlessly brought to life, culminating in a tangible, operational prototype with the capability of end effector position control. Realizing this vision demanded a semester filled with dedicated effort, inclusive of multiple redesigns, 3D printing, mechanical tweaks, in-depth kinematic studies, and intricate programming. The achievements are especially commendable when considering the initial obstacles faced, notably the sole dependence on a household 3D printer.

As we cast our eyes forward, the project is set to evolve into a Master's Thesis spearheaded by the same author. The subsequent phase, which involves crafting a second arm, is anticipated to open doors to the complex world of robotic interaction and dual-handed dexterity. This work serves as a foundational step towards the grand objective: to design and engineer two harmoniously synchronized robotic arms, tailor-made to be affixed on drones, thereby empowering them to undertake intricate dual-handed operations.

VI. Appendices

[1] J. Sindermann, „Design and construction of a robotic arm towards aerial manipulation," *Semesterthesis - TUM,* 2021.

[2] Robotis, „E-Manual XH430-W210 Servo," 2022. [Online]. Available: https://github.com/ROBOTIS-GIT/emanual/blob/master/docs/en/dxl/x/xh430-w210.md.

[3] S. H. Zeitler, „GitHub," [Online]. Available: https://github.com/Friedsam2000/Robotic-Arm-Prototype.

[4] www.seeedstudio.com, „RS232 vs TTL: Beginner Guide to Serial Communication," 2019. [Online]. Available: https://www.seeedstudio.com/blog/2019/12/11/rs232-vs-ttl-beginner-guide-to-serial-communication/.

[5] J. J. Craig, Introduction to Robotics, Mechanics and Control, Upper Saddle River, NJ: Perason Education International, 2005.

[6] A. A. M. Rodney G. Roberts, A Comparison of Two Methods for Choosing Repeatable Control Strategies for Kinematically Redundant Manipulators, Nice, France: School of Electrical Engineering, Purdue University, 1992.

[7] R. Y. A., „Geometric Jacobians Derivation and Kinematic," *5th International Conference on Robotics and Mechatronics (ICROM),* 2017.

[8] T. Foote, „tf: The transform library," *Technologies for Practical Robot Applications (TePRA),* 2013.

[9] T. Milligan, „More applications of Euler rotation angls," *IEE Antennas and Propagation Magazin,* Bd. 41, Nr. 4, pp. 78-83, 1999.

[10] J. W. T. F. Heng-Bin An, „On finite difference approximation of a matrix-vector product in the," *Journal of Computational and Applied,* 2010.

[11] Dynamixel, „GitHub," [Online]. Available: https://github.com/ROBOTIS-GIT/DynamixelSDK.

[12] J.-P. Merle, „Jacobian, Manipulability, Condition Number, and Accuracy of Parallel Robots," *ASME. J. Mech. Des.,* p. 199–206, 2005.

[13] M. M. a. L. E. K. I. A. Sucan, „The Open Motion Planning Library," *IEEE Robotics & Automation Magazine,* Bd. 19, Nr. 4, pp. 72-82, 2012.

[14] J. &. G. M. &. L. H. &. H. G. Butterfass, „ DLR-Hand II: next generation of a dextrous robot hand.," *Proceedings 2001 ICRA, IEEE International Conference on Robotics and Automation.,* 2001.

[15] J. &. S.-N. G. &. C.-L. R. Peza-Solis, „Trajectory Tracking Control in a Single Flexible-Link Robot," *Journal of Applied Research and Technology,* 2015.