UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

MODELLING AND CONTROL OF MANIPULATORS

Third Assignment

Jacobian Matrices and Inverse Kinematics

Author:

Delucchi Manuel
Cappellini Matteo

Student ID:

s4803977
s4822622

Professors:

Giovanni Indiveri
Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo
George Kurshakov

January 8, 2024

Contents

Mathematical expression	Definition	MATLAB expression
$< w >$	World Coordinate Frame	w
${}^{a}_{b}R$	Rotation matrix of frame $< b >$ with respect to frame $< a >$	aRb
${}^{a}_{b}T$	Transformation matrix of frame $< b >$ with respect to frame $< a >$	aTb
${}^{a}O_{b}$	Vector defining frame $< b >$ wit respect to frame $< a >$	aOb

Table 1: Nomenclature Table

1 Assignment description

The third assignment of Modelling and Control of Manipulators focuses on the definition of the Jacobian matrices for a robotic manipulator and the computation of its inverse kinematics.

The third assignment consists of three exercises. You are asked to:

- Download the .zip file called MOCOM-LAB3 from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling in the predefined files. In particular, you will find two different main files: *"ex1.m"* for the first exercise and *"ex2_ex3.m"* for the second and third exercises.
- Write a report motivating your answers, following the predefined format on this document.
- **Putting code in the report is not an explanation!**

1.1 Exercise 1

Given the CAD model of the robotic manipulator from the previous assignment and using the functions already implemented:

Q1.1 Compute the Jacobian matrices for the manipulator for the following joint configurations:

- $\mathbf{q}_1 = [1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8]$
- $\mathbf{q}_2 = [0.3, 1.4, 0.1, 2.0, 0, 1.3, 0]$
- $\mathbf{q}_3 = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.]$
- $\mathbf{q}_4 = [1, 1, 1, 1, 1, 1, 1]$

1.2 Exercise 2

In the second exercise the model of a Panda robot by Franka Emika is provided. The robot geometry and jacobians can be easily retrieved by calling the following built-in functions: *"getTransform()"* and *"geometricJacobian()"*.

Q2.1 Compute the cartesian error between the robot end-effector frame ${}^b_eT$ and the goal frame ${}^b_gT$.
${}^b_{ge}T$ must be defined knowing that:

- The goal position with respect to the base frame is ${}^bO_g = [0.55, -0.3, 0.2]^\top (m)$
- The goal frame is rotated of $\theta = \pi/6$ around the y-axis of the robot end-effector initial configuration.

Q2.2 Compute the desired angular and linear reference velocities of the end-effector with respect to the base: ${}^b\nu^*_{e/0} = \alpha \cdot \begin{bmatrix} \omega^*_{e/0} \\ v^*_{e/0} \end{bmatrix}$, such that $\alpha = 0.2$ is the gain.

Q2.3 Compute the desired joint velocities. (Suggested matlab function: *"pinv()"*).

Q2.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"*.

1.3 Exercise 3

Repeat the Exercise 2, by considering a tool frame rigidly attached to the robot end-effector according to the following specifications:

$$eRt = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 & 0 \\ sin(\phi) & cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & \end{bmatrix}, \phi = -44.98(deg), {}^eO_t = [0, 0, 21.04]^\top (cm)$$

Q3.1 Compute the cartesian error between the robot tool frame ${}^b_tT$ and the goal frame ${}^b_gT$.
${}^b_{gt}T$ must be defined knowing that:

- The goal position with respect to the base frame is ${}^bO_g = [0.55, -0.3, 0.2]^\top (m)$
- The goal frame is rotated of $\theta = \pi/6$ around the y-axis of the robot tool frame initial configuration.

Q3.2 Compute the angular and linear reference velocities of the tool with respect to the base: ${}^{b}\nu^{*}_{e/0} = \alpha \cdot \begin{bmatrix} \omega^{*}_{e/0} \\ v^{*}_{e/0} \end{bmatrix}$, such that $\alpha = 0.2$ is the gain.
Q3.3 Compute the desired joint velocities. (Suggested matlab function: *"pinv()"*).
Q3.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"*.
Q3.5 Comment the differences with respect to Exercise2.

2 Exercise 1

The aim of kinematic analysis in a manipulator is to understand how changes in joint configurations, represented by the vector q, affect the velocity of the end-effector, denoted by $\dot{x}_{e/o}$, when subjected to a set of joint space velocities $\dot{q}$. This relationship is linear and defined by a configuration-specific matrix known as the ***Jacobian*** $J_{e,o}$, represented as a $6 \times n$ matrix, where n is the number of joints in the manipulator. The six rows typically represent the velocities in three transnational axes (x, y, z) and three rotational axes (roll, pitch, yaw) of the end-effector.

$$\dot{x}_{e/o} = J_{e,o}(q) * \dot{q} \tag{1}$$

The angular and linear velocities, respectively represented as $w_{e/o}$ and $v_{e/o}$ of the end-effector are computed as sums across intermediate frames (from 1 to n):

$$w_{e/o} = \sum_{i=1}^{n} w_{i/i-1} \tag{2}$$

$$v_{e/o} = \sum_{i=1}^{n} v_{i/i-1} + (w_{i/i-1} \times r_{e,i}) \tag{3}$$

The angular velocities between each couple of intermediate frames $w_{i/i-1}$ are determined based on the joint velocities $\dot{q}_i$ and joint types (**RJ** for revolute joints, **TJ** for prismatic joints).

$$w_{i/i-1} = \begin{cases} k_i * \dot{q}_i & \text{if } i = \textbf{RJ} \\ 0 & \text{if } i = \textbf{TJ} \end{cases} \tag{4}$$

Similarly, the linear velocities between each couple of intermediate frames $v_{i/i-1}$ are determined based on the joint velocities $\dot{q}_i$, joint types and the distance vector $r_{e,i}$ between the end-effector and the i-th joint.

$$v_{i/i-1} = \begin{cases} k_i * \dot{q}_i \times r_{e,i} & \text{if } i = \textbf{RJ} \\ k_i * \dot{q}_i & \text{if } i = \textbf{TJ} \end{cases} \tag{5}$$

Combining the angular and linear velocities together for all joints gives a unified representation:

$$\dot{x}_{e/o} = J_{e,o}(q) * \dot{q} \rightarrow \begin{cases} w_{e/o} = J^A(q) * \dot{q} \\ v_{e/o} = J^L(q) * \dot{q} \end{cases} \tag{6}$$

where J^A and J^L are a collection of terms representing the angular and linear components for all joints in the manipulator.

$$J^A_{i,o} = \begin{cases} k_i & \text{if } i = \textbf{RJ} \\ 0 & \text{if } i = \textbf{TJ} \end{cases} \tag{7}$$

$$J^L_{i,o} = \begin{cases} k_i \times r_{e,i} & \text{if } i = \textbf{RJ} \\ k_i & \text{if } i = \textbf{TJ} \end{cases} \tag{8}$$

3 Q1.1

For the first task, we computed the Jacobian matrices for the manipulator given specific joint configurations using the functions implemented for the CAD model from the previous assignment:

- ***BuildTree()*** function to create the tree of frames from the CAD model (observable in **Appendix**) to establish the structural hierarchy of frames within the manipulator.
- ***GetDirectGeometry()*** and ***GetTransformationWrtBase()*** functions to obtain the transformation matrices corresponding to each joint configuration from the tree of frames.

Then, we implemented the ***GetJacobian()*** function to calculate the end-effector Jacobian matrices for each specific joint configuration.

The joint configurations considered are:

- $\mathbf{q}_1 = [1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8]$
- $\mathbf{q}_2 = [0.3, 1.4, 0.1, 2.0, 0, 1.3, 0]$
- $\mathbf{q}_3 = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.]$
- $\mathbf{q}_4 = [1, 1, 1, 1, 1, 1, 1]$

The corresponding Jacobian matrices are:

$$J_1 = \begin{bmatrix} 0 & -0.9738 & -0.0516 & 0.4367 & 0.8629 & 0.1484 & 0.2744 \\ 0 & -0.2272 & 0.2213 & -0.8720 & 0.4756 & 0.0849 & -0.9607 \\ 1 & 0 & 0.9738 & 0.2213 & 0.1710 & -0.9853 & -0.0414 \\ -84.6839 & -112.0562 & 26.6565 & -68.7635 & 22.1181 & -145.3631 & 0 \\ 251.4504 & 480.3023 & 270.3338 & 60.8825 & 12.6505 & -40.4250 & 0 \\ 0 & -25.3391 & -60.0074 & 375.6585 & -146.8038 & -25.3846 & 0 \end{bmatrix}$$

$$J_2 = \begin{bmatrix} 0 & -0.2955 & -0.1624 & -0.3880 & -0.8925 & -0.3880 & -0.0172 \\ 0 & 0.9553 & -0.0502 & 0.9215 & -0.3711 & 0.9215 & 0.0112 \\ 1 & 0 & 0.9854 & -0.0170 & 0.2563 & -0.0170 & 0.9998 \\ 66.8925 & 668.0313 & 30.7961 & 228.1334 & -66.6141 & -66.6141 & 0 \\ 119.7951 & 678.7165 & 82.3672 & 237.5423 & -57.2052 & 140.9873 & 0 \\ -315.5555 & 209.9516 & -195.6046 & 107.5337 & 135.8502 & 59.4009 & 0 \\ 0 & 336.8636 & 3.6019 & 407.6740 & -2.5016 & 1.7672 & 0 \end{bmatrix}$$

$$J_3 = \begin{bmatrix} 0 & 0 & -0.9950 & -0.0198 & 0.9216 & 0.1326 & 0.6340 \\ 0 & 1 & 0 & 0.9801 & -0.0587 & 0.9766 & 0.0476 \\ 1 & 0 & 0.0998 & -0.1977 & -0.3836 & 0.1692 & -0.7719 \\ -11.5371 & -94.2237 & -1.1518 & -277.0574 & 9.7293 & -116.5666 & 0 \\ 68.9866 & 0 & -86.8658 & -101.2317 & 71.6365 & 32.0793 & 0 \\ 0 & -68.9866 & -11.4794 & -474.1005 & 12.4132 & -93.7665 & 0 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} 0 & -0.8415 & -0.2919 & -0.8372 & 0.5468 & -0.4559 & -0.2954 \\ 0 & 0.5403 & -0.4546 & -0.3039 & -0.4589 & 0.5384 & -0.8427 \\ 1 & 0 & 0.8415 & -0.4546 & -0.7003 & -0.7087 & -0.4501 \\ 396.0151 & 23.0076 & 313.8749 & -35.9272 & -58.6935 & -128.4543 & 0 \\ -8.7764 & 35.8322 & 5.0460 & -387.7714 & 69.3126 & 0.6435 & 0 \\ 0 & 337.9771 & 111.6171 & 325.3533 & -91.2477 & 83.1148 & 0 \end{bmatrix}$$

4 Exercise 2

In this part of the assignment we will study how we can solve the *inverse kinematic problem* on the *Panda robot* model by Franka Emika, so that we can compute the needed joint positions in space such that the end-effector can reach a certain goal.

4.1 Q2.1

The initial configuration is defined by the following joint angles:

$$q = [0.0167305, 0.762614, 0.0207622, 2.34352, 0.0305686, 1.53975, 0.753872]$$

The goal position for the end-effector is specified in the base frame as ${}^bO_g = [0.55, -0.3, 0.2]^T$. To achieve this goal position accurately, it is necessary to perform a transformation that involves both translation and rotation. The rotation component involves rotating the end-effector around the y-axis by an angle $\theta = \frac{\pi}{6}$. This rotation is applied to the initial configuration to achieve the desired end-effector orientation at the goal position.

To accomplish this, we execute the following steps:

1. Form a rotation matrix ${}^{e}R_{ge}$ to execute the desired rotation around the y-axis by $\theta = \frac{\pi}{6}$.

$$ {}^{e}R_{ge} = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} = \begin{bmatrix} 0.866 & 0 & 0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0.866 \end{bmatrix} $$

2. Multiply the rotational part of the initial transformation matrix ${}^{b}_{e}T$, obtained using the built-in function *getTransform()*, with the previously calculated rotation matrix ${}^{e}R_{ge}$ to derive the rotation matrix ${}^{b}R_{ge}$ within the frame $< b >$, indicating the desired rotation to reach the goal.

3. Calculate the overall transformation matrix ${}^{b}T_{ge}$ by combining ${}^{b}R_{ge}$ with the end-effector's goal position ${}^{b}O_{g}$. This combination produces the transformation matrix that define the transition from the initial end-effector position to the desired goal position, while taking into account the necessary rotation. The final matrix structure looks like this:

$$ {}^{b}T_{ge} = \begin{bmatrix} {}^{b}R_{ge} & {}^{b}O_{g} \\ \vec{0} & 1 \end{bmatrix} = \begin{bmatrix} 0.6526 & -0.6824 & 0.3292 & 0.55 \\ -0.5837 & -0.7299 & -0.3558 & -0.3 \\ 0.4831 & 0.0401 & -0.8747 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Now, we can compute the linear component of the Cartesian error between the robot end-effector frame ${}^{b}T_{e}$ and the goal frame ${}^{b}T_{ge}$, by simply extracting their translation vectors and checking their difference at each iteration.
To compute the angular part of the Cartesian error (which represents the desired change in orientation that needs to be applied to the current rotation matrix of the end-effector, in order to achieve the desired orientation), we will need to extract the desired angle and axis of rotation by using the function *ComputeInverseAngleAxis()*, developed in the first assignment, on the matrix ${}^{e}R_{ge}$, since it contains the information relative to the desired rotation to reach the goal. Then we just multiply these two results by the rotational part of the transformation matrix ${}^{b}T_{e}$, which is the current rotation matrix of the end-effector, to obtain the angular error.

4.2 Q2.2

Finally we can compute the vector of reference velocities ${}^{b}\nu^{*}_{e/0} = \alpha \cdot \begin{bmatrix} \omega^{*}_{e/0} \\ v^{*}_{e/0} \end{bmatrix}$ simply by multiplying the obtained error by the given proportional gain ($\alpha = 0.2$).

4.3 Q2.3

Now we can compute the desired joint velocities that will be used to control the manipulator by multiplying the pseudo-inverse of the Jacobian matrix ${}^{b}J_{e}$, obtained via the MATLAB function *pinv()*, with the vector of reference end-effector velocities obtained in the previous step. The pseudoinverse is used when the Jacobian may not have a unique and exact inverse and this approach allows us to find a solution that minimizes the error in the least squares sense. The result is a vector of joint velocities that is crucial in inverse kinematics control as it determines the adjustments needed for each joint to achieve the desired end-effector motion.

4.4 Q2.4

The simulation of the robot's motion is implemented by the function *KinematicSimulation()*. This function dynamically updates the position of each joint using a straightforward kinematic formula:

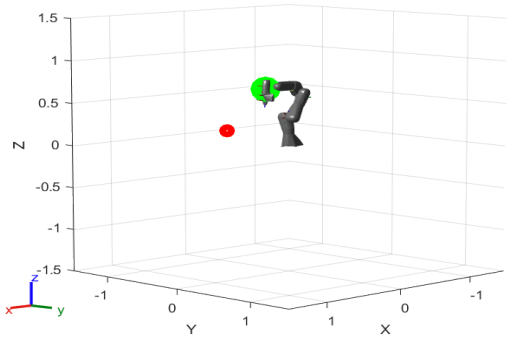
$$ q_i = q_i + \dot{q_i} t $$

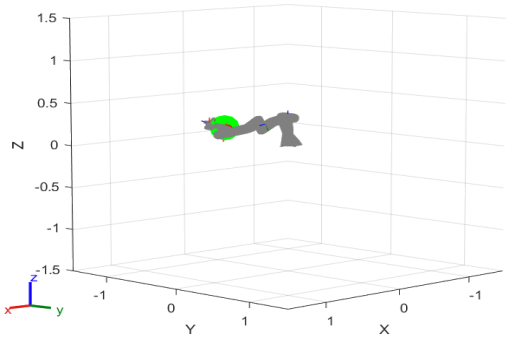
where q_i represents the updated joint position, $\dot{q_i}$ denotes the previously obtained desired joint velocity and t is a specified time interval between each iteration. By iteratively applying this formula, the function dynamically progresses the robot's joint positions, effectively simulating the motion over time. This approach ensures the adherence of joint angles within predetermined operational limits:

$$ q_{min} = [-2.8973; -1.7628; -2.8973; -3.0718; -2.8973; -0.0175; -2.8973] $$

$$ q_{max} = [2.8973; 1.7628; 2.8973; -0.0698; 2.8973; 3.7525; 2.8973] $$

In this image we can observe the initial configuration of the robot and how the controller acts on each joint to reach the desired target.





5 Exercise 3

For this last exercise, the task slightly deviates from the previous one as the robot is tasked to reach the same goal position, but we are now considering a tool frame rigidly attached to the robot's end-effector.

5.1 Q3.1

An important distinction lies in the rigid attachment of the tool frame to the robot, characterized by the following specifications:

$$eRt = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \phi = -44.98(deg), {}^{e}O_t = [0, 0, 21.04]^\top (cm)$$

Based on those information, we built the new transformation matrix ${}^{e}T_t$, an homogeneous transformation matrix that describes the pose of the tool frame wrt the end-effector frame:

$${}^{e}T_t = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 & 0 \\ sin(\phi) & cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0.2104 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, following a similar methodology employed in the previous exercise, we executed the previously cited sequential steps to derive the following transformation matrix:

$${}^{b}T_{gt} = \begin{bmatrix} 0.8599 & 0.2432 & 0.4489 & 0.55 \\ 0.2183 & 0.9700 & 0.1072 & -0.3 \\ 0.4615 & 0.0058 & -0.8871 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now, as seen in exercise 2, we can compute the Cartesian error between the robot tool frame ${}^{b}T_t$ and the goal frame ${}^{b}T_{gt}$, by simply extracting the linear and angular error, as already done previously.

5.2 Q3.2

Here the steps to follow are the same described in section **2.2**, as we use the errors extracted in the previous part to calculate the desired angular and linear reference velocities of the tool wrt the base.

5.3 Q3.3

The main difference is highlighted in this part of the exercise, as we need to take into account the change of the structure when studying the Jacobian matrix from base to the rigid tool. To do so, we consider the *Rigid Body Transformation Matrix* from the end-effector to the tool frame, which we can obtain by studying the skew-symmetric matrix constructed from the translation vector of the tool frame ${}^{e}T_t$, as follows:

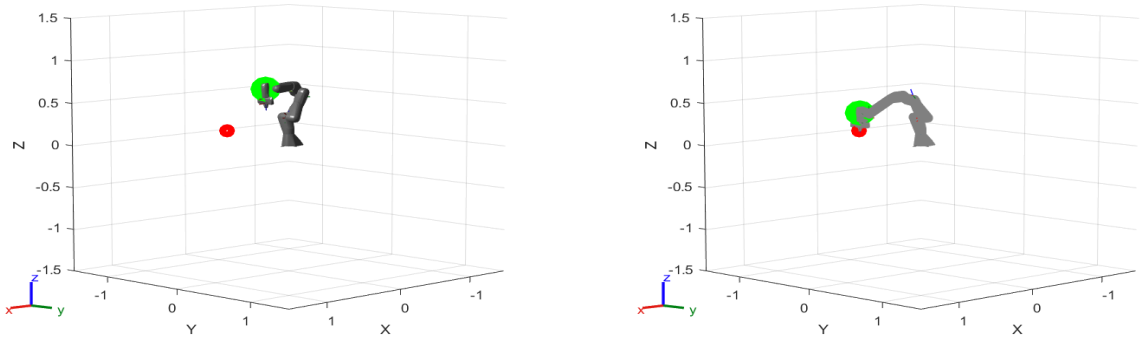
$$skew = \begin{bmatrix} 0 & -{}^{e}T_t(3) & {}^{e}T_t(2) \\ {}^{e}T_t(3) & 0 & -{}^{e}T_t(1) \\ -{}^{e}T_t(2) & {}^{e}T_t(1) & 0 \end{bmatrix}$$

$$ {}^{ee}T_t = \begin{bmatrix} I_3 & 0_3 \\ skew & I_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -0.2104 & 0 & 1 & 0 & 0 \\ 0.2104 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} $$

Since ${}^{b}J_e$ is the Jacobian matrix representing the end-effector motion wrt the base and ${}^{ee}J_t$ can be used to transform the Jacobian from the end-effector frame to the tool frame, the matrix resulting from their multiplication, denoted as ${}^{b}J_t$, represents the Jacobian matrix corresponding to the motion of the rigid tool wrt the base, which we can use to determine the desired joint velocities $\dot{q}$, computed by taking the pseudo-inverse of the Jacobian matrix ${}^{b}J_t$ with the MATLAB function *pinv()* and multiplying it by the vector of desired end-effector velocities $\dot{x}$.
The result is a vector of joint velocities that is crucial in inverse kinematics control as it determines the adjustments needed for each joint to achieve the desired end-effector motion.

5.4 Q3.4

In this image we can observe the initial configuration of the robot and how the controller acts on each joint to reach the desired target when considering the rigid tool attached to the end-effector.



5.5 Q3.5

The difference with the second part of the assignment lies in the presence of the tool attached to the end-effector of the robot, this allows the robot to reach the target in a more natural way, since it doesn't need to orientate the end-effector in parallel with the target's position, but it can reach it from above, changing the joints orientation.

6 Appendix

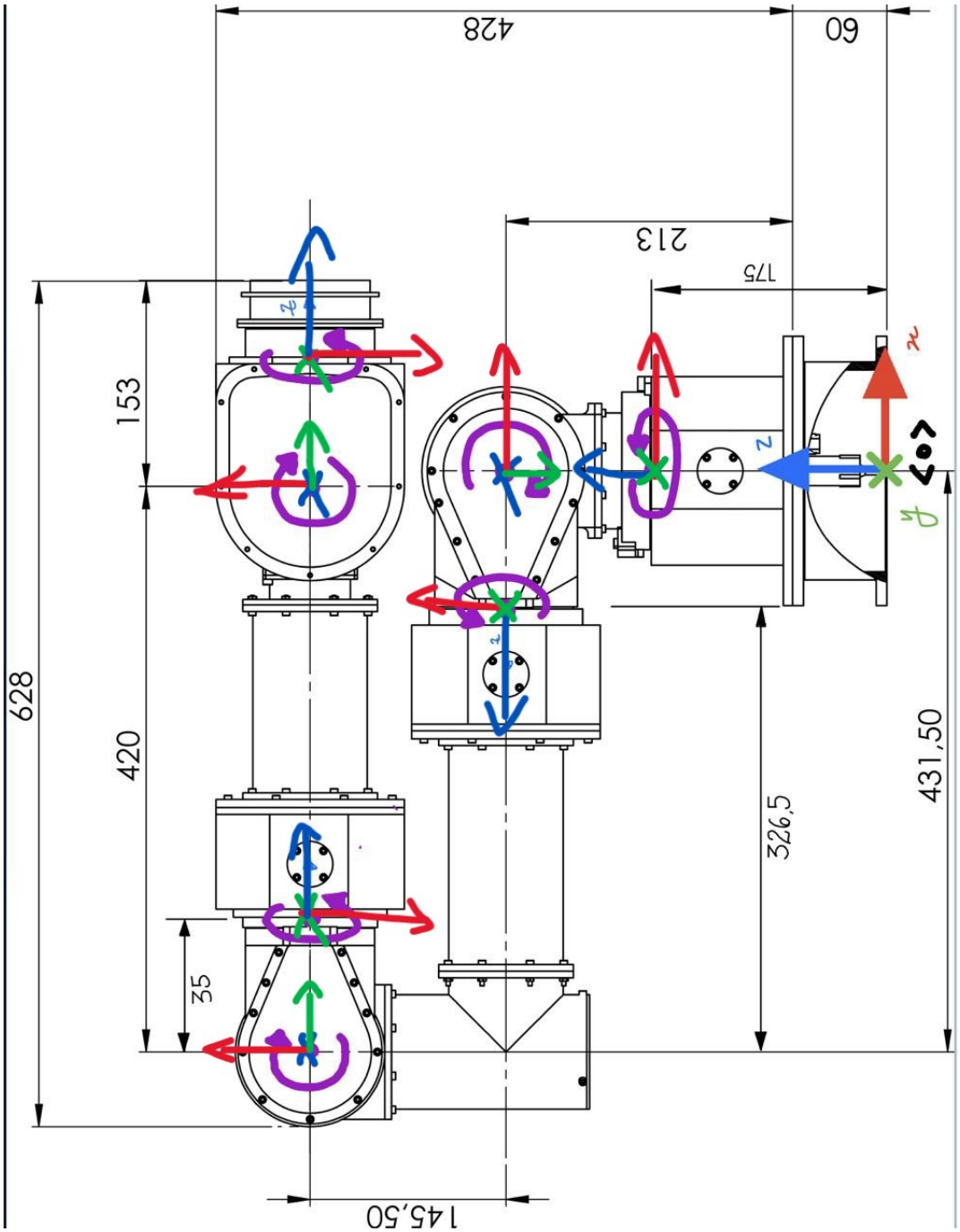


Figure 1: CAD model of the robot

6.1 Appendix A

6.2 Appendix B