Proximity-informed robot control through whole-body artificial skin

by

A. Aranburu Fernndez

B.A., UPV/EHU Euskal Herriko Unibertsitatea, 2017

M.S., UPC Universitat Politecnica de Catalunya, 2020

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

2020

This thesis entitled:
Proximity-informed robot control through whole-body artificial skin
written by A. Aranburu Fernndez
has been approved for the Department of Computer Science

Prof. Alessandro Roncone

needed?

needed?

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Aranburu Fernndez, A. (M.S., Industrial Engineering)

Proximity-informed robot control through whole-body artificial skin

Thesis directed by Prof. Alessandro Roncone

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

After conclusions.

Conclusions smaller.

Often the abstract will be long enough to require more than one page, in which case the macro "\OnePageChapter" should *not* be used.

But this one isn't, so it should.

Contents

Chapter 1

Introduction

1.1 Human-Robot Interaction

We are on the threshold of a new era in robotic technology that will change our lives in the way we live and work. This robotic revolution has the potential to surpass the ubiquity and usefulness of the computer revolution, if only because robots can change not only our virtual world, but the physical world also. (https://h2r.cs.brown.edu/about/)

One of the main fields of research working on the expansion of these abilities to interact with the physical world is human-robot interaction (HRI). HRI deals with the general problem of integrating robots in human-populated environments. There are a number of ways in which robotics can be impactful to society in the short- mid- term. Some examples:

- Hospital robots will check on patients and report their status to nurses, saving time and improving patient outcomes.
- Childcare robots will help parents with chores such as assisting at diaper changing or feeding, so that families can spend high-quality time together.
- Manufacturing robots will collaborate with people to assemble complex objects on reconfigurable assembly lines, increasing the efficiency and flexibility of factory floors.

The transition of robots from restricted access areas to human-populated environments comes with a number of problems that research needs to overcome, being safety one of the key issues that need to be addressed. Traditionally, safety in industrial robots has been achieved isolating them in a cell with safety interlocks to prevent direct interaction. HRI contributions will only be feasible once the coexistence of robots and humans does not come with serious injury risk.

1.2 Human senses and sensors in robotics

In order to achieve the safety requirements of HRI, robots are expected to adapt to the changes in their environment the way humans do. This is why it is important to study how humans interact with their environment.

Interaction between a human being and its environment is accomplished through its senses. One important aspect research in robotics has focused on has been to look for ways to emulate and exploit these human senses. For example, visual sensing has been widely explored in this field. One of the reasons is the development of new low-cost depth sensors such as the Microsoft Kinect $^{\text{TM}}$ has allowed to meet the many requirements of vision application at an affordable price. These depth sensors are capable of acquiring 2D images as well as the distances of the objects represented in each pixel to the camera. The usual way this data is used is to reassemble representations of obstacles in a robot-oriented space.

However, other human senses are of great importance for the acquisition of data relevant to the interaction with their surroundings. Touch is one of them. Tactile sensing is the most fundamental sense when contacting the external world and it is the biggest and oldest of the sensorial organs. An experiment that consisted of exploring objects after anesthetizing the hands of a control group demonstrated that the simple task of maintaining a stable grasp of objects becomes surprisingly difficult when sense of touch is suppressed. Their hand movements are inaccurate and unstable [1]. In another experiment carried on by astronauts at the International Space Station, sense of touch was proven to be an important indicator of direction and spatial disorientation [2]. Sense of touch is crucial when experiencing object properties, such as size, shape, texture and temperature. It informs about slip. It is essential to develop awareness of the body and in consequence, to differentiate ones body from the rest of ones surroundings. Its absence seriously hinders the interaction of an individual with its environment [3].

However, tactile sensing in robotics has not been able to achieve the penetration this evidence suggests it should have had. In traditional industrial robots, this importance has been ignored. Engineers have been able to avoid the issue of developing a artificial sensor that emulates touch by using prior knowledge about the object to be manipulated and the environment. The limitations of this approach are obvious: robots are only capable of working in structured and controlled environments. https://www.sciencedirect.com/science/article/pii/S0921889015001621

The result is that research and technology in artificial tactile sensors is not as well developed as other perception modalities. Tactile sensors in robotic applications are represented by:

- **Pressure sensing arrays.** Pressure sensor matrix that provides information about the location and amount of pressure exerted on a surface.
- **Force-torque sensors.** They give feedback about the forces and torques that are applied on a given point in the 3 geometric axes.
- **Dynamic tactile sensors.** If you press your finger against an objects corner, you can feel it for as long as you hold your finger in place. If you slowly rest your finger on the table, in contrast, you feel relatively little, until you start moving it gently back and forth. Suddenly, you are able to feel the texture, dustiness, scratches and a breadth of the properties the object has. These sensations are provided by the dynamic, fast acting tactile sensors that are embedded in the skin.
- **Detect changing light levels.** https://www.wired.com/story/this-clever-robotic-finger-feels-with-light/

1.3 Whole-body artificial skins

Current research is focused on developing tactile skins that cover robot end effectors and hands with a number of tactile sensors, in parallel with devising new algorithms that make use of these new sensors for dexterous object manipulation ([4] [5]).

However, whole-body artificial skins have not gained the same attention as end-effector coverings. In fact, this type of skins can be of great importance in the transition of robots to operating alongside humans and the new safety issues that arise with it.

Traditional robots have been designed for precision and performance at expenses of compliance. Compliance is essential for interaction and extensive research has been conducted to deal with the issue, leading to two paradigms: **passive** and **active compliance**. Passive compliance aims to achieve safety through physical adaptation to the environment, by the use of flexible parts and mechanisms or by limiting capabilities such us speed. Active compliance, in contrast, relies on algorithms and control laws for reactive interaction with the environment. Both approaches are not mutually exclusive.

Both approaches have some important limitations, though. Passive compliance achieves safety at the cost of reducing accuracy, in the case of using flexible parts, and agility when the capabilities of the robot are reduced. Active compliance makes use of not suitable sensors, which provide either a high amount of low-quality information regarding the task, as in the case of the aforementioned depth cameras or a limited amount of relevant data, as force-torque sensors located at the robots joints. In addition, both solutions are currently lacking in flexibility. New problems require completely new designs. This may be one of the reasons why none of the solutions have become the established standard in robotics.

Whole-body artificial skins have the potential to solve all of these issues, providing high-density relevant information, that is as close to the particular task of collision avoidance as it gets. At the same time, it can also be designed to be mechanically compliant to the environment.

Adaptable to the form of the robot.

In addition, the skin can be equipped with heterogeneous sensing so that a greater amount of high-quality information is collected. Tactile data is only useful when contact has been established between the robot and the environment. However, collisions can be actively avoided before establishing undesired contact by choosing suitable pre-impact strategies. That is why adding proximity sensing capabilities to the skin opens up new possibilities in terms of safety robot control. The addition of proximity sensors allows a enables a compact, external motion capture system free system, with highly informative prior-to-contact data. This combined with lighter data, the main problems of depth-sensing are solved: occlusion is never again a problem and computational costs are greatly reduced.

1.4 Control and collision avoidance

From the control perspective, using proximity data for the safe movement of a robot might prevent performing the main task when any undesired object obstructs the desired trajectory. Nonetheless, this does not have to be the case if a redundant robot is used.

A robot is redundant if the number of degrees of freedom (DoF) n is bigger than the degrees of freedom needed for the task m. For example, if the specifications of a given task only require positioning the end-effector of a robot at a certain point, i.e. $m = 3$ a robot with $n = 3$ DoF would be able to accomplish the task and one with $n = 4$ DoF would be task redundant. A redundant robot has the advantage of being able to accomplish the task in infinitely many possible ways.

This means that safe motion does not imply stopping the main task and in fact, the main task should only halt when there is no way to make use of redundancy to avoid a collision while completing it.

1.5 Objectives and scope

Chapter 2

Related Work

Chapter 3

Technical Approach

In this chapter, we will explain the technical means by which the objective of this thesis has been approached. Firstly, in section 3.1 we start talking about the basic elements that have been used during the development this work. In order to make the robot avoid obstacles with proximity data, obstacles need to be located with respect to the robots kinematic chain. That's why, in section 3.2.1 we briefly present a framework for automatic kinematic calibration that leverages an IMU to accurately estimate the pose (position and orientation) of skin units along the surface of a robot. The main section is section 3.3, where the methods for collision avoidance are presented.

3.1 Tools

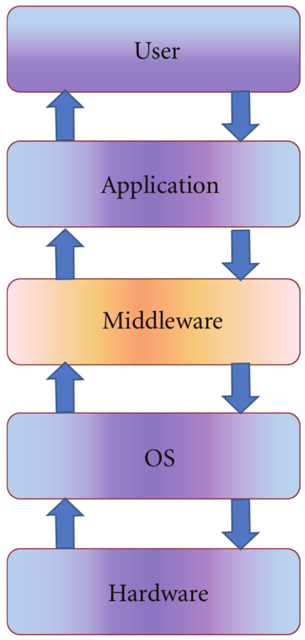
General presentation of the tools

3.1.1 ROS

The Robot Operating System (ROS) is an open source robot middleware, robot framework or robot development environment (RDE), one of many available in the market. Robot middleware [6] is an abstraction layer that resides between the operating system (OS) and software applications (figure 3.1). Its purpose is to provide a framework and take care of several important parts of applications development, so that the developer needs only to build the logic or algorithm as a component.

The main goal of ROS in particular is to provide a framework for easy code reuse in research.

Figure 3.1: Robotic middleware [6]



This is why ROS is most extended in the academic context. It also provides APIs for Python and C++ and codes written in both languages can be used interchangeably and at the same time.

ROSs main concept is its runtime graph: a peer-to-peer network of loosely coupled components or nodes that use the ROS communication infrastructure. ROS implements two main communication types:

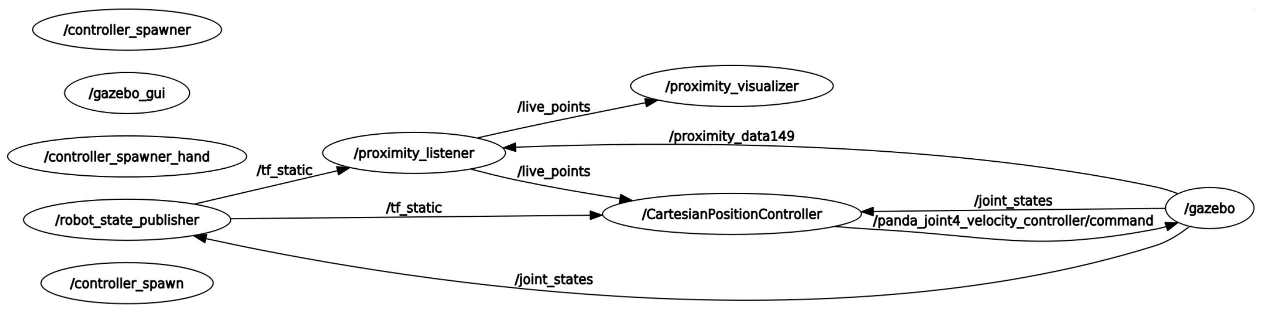
- **Topics.** A publisher node publishes a message to a topic. Another node subscribes to this topic, meaning that every time there is a new message published in the topic this subscriber node will receive it and therefore will we able to utilize and manipulate the data. Several nodes can publish to a given topic, as well as several nodes can also be subscribers of the same topic.

- **Services.** Sometimes the asynchronous model topics provide is not suitable. When synchronous behavior is desired services are the way to go. Services have two sides: client and service. From the client side calling a service is equivalent to calling a function. However, this function is managed in a different node: the server side.

In figure 3.2 the runtime graph of this work is presented as an example. The real robot is replaced by a simulation. The /gazebo node is calculating the physics of the real robot, as well as sensor readings. /CartesianPositionController is subscribed to the topic /joint_states, as it needs this information for control purposes. Sensor data is processed in the /proximity_listener node and then is published to the topic /live_points. This topic has two subscribers in these examples. On the one hand, /proximity_visualizer uses the data published in /live_points to visualize the sensor data in rviz (the 3D visualization tool provided by ROS to visualize several types of visual data). On the other hand, /CartesianPositionController uses the data for collision avoidance control, as explained later in this chapter (CROSS-REFERENCES).

/robot_state_publisher is an example of the benefits of the framework ROS provides for easy code reuse. This node is part of a widely used ROS package that uses the robots description (described in a .urdf file) and messages published in /joint_states to provide transformation matrices between any two joints in the robots kinematic chain. It publishes this data in /tf topics.

Figure 3.2: ROS graph



In addition, bags in ROS enable real data storage coming from sensors in the robot. These allow developers to test their algorithms with real data without having to collect new data every time they need to run their code.

The ROS Master enables nodes to find each other and to communicate. It also holds the parameter server, which provides a central storage location for data that is relevant for several nodes. It serves, broadly speaking, as a place where global variables can be stored and retrieved by nodes. Examples of things that are stored in the parameter server are PID control parameters, the robots urdf description and frequencies at which different components living in the graph work.

Last, it should be mentioned that despite the fact that ROS is widely used in the academic context, it is lacking the reliability and robustness safety-critical systems in industrial or commercial contexts require. Some of the industrial fields, such as the automotive and the aerospace, have their own standards for the development of safety-critical software. These standards need to be certified for every software component written in any of these fields. ROS was not developed to any of these standards and it actually depends on many libraries that were not developed to these standards. ROS 2, in contrast, is based on components that are safety-certified (DDL communication) that make ROS 2 certifiable. Indeed, Apex.AI is working on Apex.OS: an API compatible to ROS 2 that is being certified ISO 26262 for safe automotive applications [7].

3.1.2 Franka Emika Panda

The robot utilized for the development of the ideas presented in this work has been the Panda, manufactured by the company Franka Emika based in Munich, Germany. It is well known for its performance and usability as well as for its affordability, starting at 10,000$.

The Franka Emika Panda (FEP) is equipped with 7 revolute joints, i.e. it has 7 degrees of freedom (DOF). Each of the revolute joints mounts a torque sensor. The total weight of FEP is around 18 kg. It can handle payloads of up to 3 kg.

It has a control interface capable of providing joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ and link side torques vector τ with a refresh rate of 1 kHz. The control interface also provides numerical values of the inertia matrix $\bar{\mathbf{M}}(\mathbf{q})$, the gravity vector $\bar{\mathbf{g}}(\mathbf{q})$, the end effector Jacobian matrix $\bar{\mathbf{J}}(\mathbf{q})$ and the Coriolis term $\bar{\mathbf{c}}(\mathbf{q}, \dot{\mathbf{q}})$.

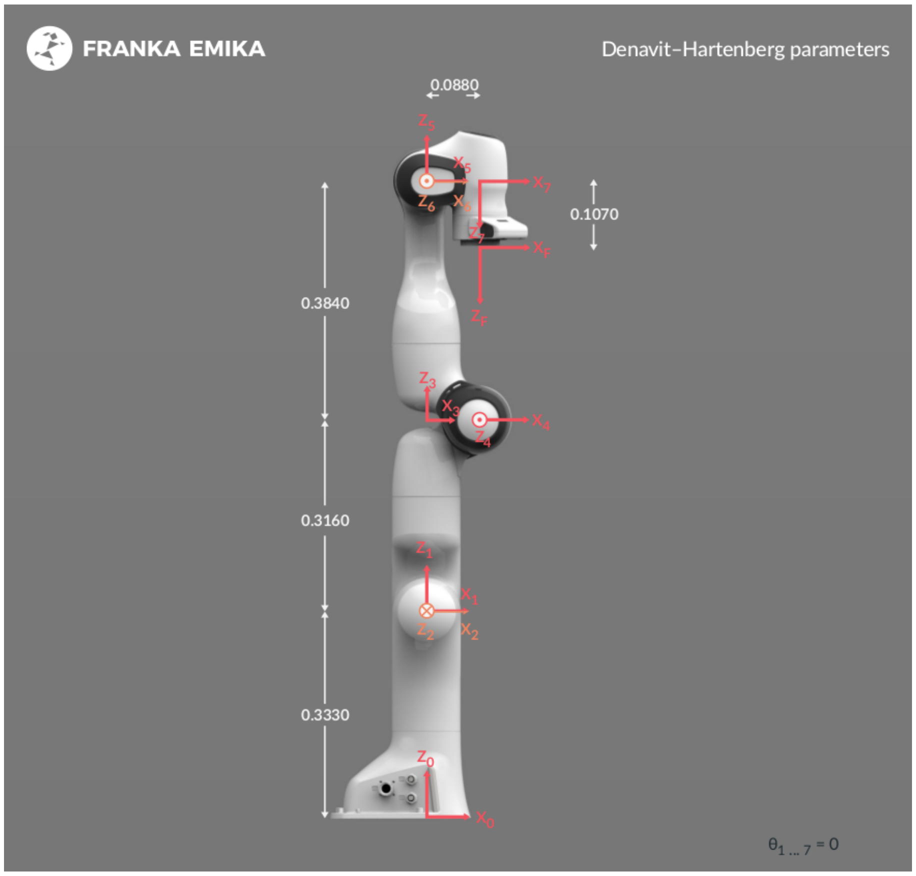
There are 3 control modes available. At the lowest level, there is the torque-mode $\tau_{\mathbf{d}}$. The joint velocity $\dot{\mathbf{q_d}}$ and joint position $\mathbf{q_d}$ modes provide a higher level control and are the ones that are going to be used throughout this work.

The Denavit Hartenberg (DH) parameters provided by the manufacturer [8] and depicted in figure 3.3 are shown in table 3.1.

Table 3.1: Denavit Hartenberg parameters [8]

Joint	$\mathbf{a}$ (m)	$\mathbf{d}$ (m)	$\boldsymbol{\alpha}$ (rad)	$\boldsymbol{\theta}$ (rad)
1	0	0.333	0	θ_1
2	0	0	$-\frac{\pi}{2}$	θ_2
3	0	0.316	$\frac{\pi}{2}$	θ_3
4	0.0825	0	$\frac{\pi}{2}$	θ_4
5	-0.0825	0.384	$-\frac{\pi}{2}$	θ_5
6	0	0	$\frac{\pi}{2}$	θ_6
7	0.088	0	$\frac{\pi}{2}$	θ_7
F	0	0.107	0	0

Figure 3.3: FEP's kinematic chain [8]



The manufacturer also provides the physical limits of the robot, both in the joint and the cartesian spaces (tables 3.2 and 3.3 respectively).

Table 3.2: Joint space limits [8]

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	
$\mathbf{q_{max}}$	2.8973	1.7628	2.8973	-0.0698	2.8973	3.7525	2.8973	rad
$\mathbf{q_{min}}$	-2.8973	-1.7628	-2.8973	-3.0718	-2.8973	-0.0175	-2.8973	rad
$\mathbf{\dot{q}_{max}}$	2.1750	2.1750	2.1750	2.1750	2.6100	2.6100	2.6100	$\frac{\text{rad}}{\text{s}}$
$\mathbf{\ddot{q}_{max}}$	15	7.5	10	12.5	15	20	20	$\frac{\text{rad}}{\text{s}^2}$

Table 3.3: Cartesian space limits [8]

	Translation	**Rotation**	**Elbow**
$\mathbf{\dot{p}_{max}}$	1.7000 $\frac{\text{m}}{\text{s}}$	2.5000 $\frac{\text{rad}}{\text{s}}$	2.1750 $\frac{rad}{\text{s}}$
$\mathbf{\ddot{p}_{max}}$	13.0000 $\frac{\text{m}}{\text{s}^2}$	25.0000 $\frac{\text{rad}}{\text{s}^2}$	10.0000 $\frac{rad}{\text{s}^2}$

3.2 Perception via proximity sensing

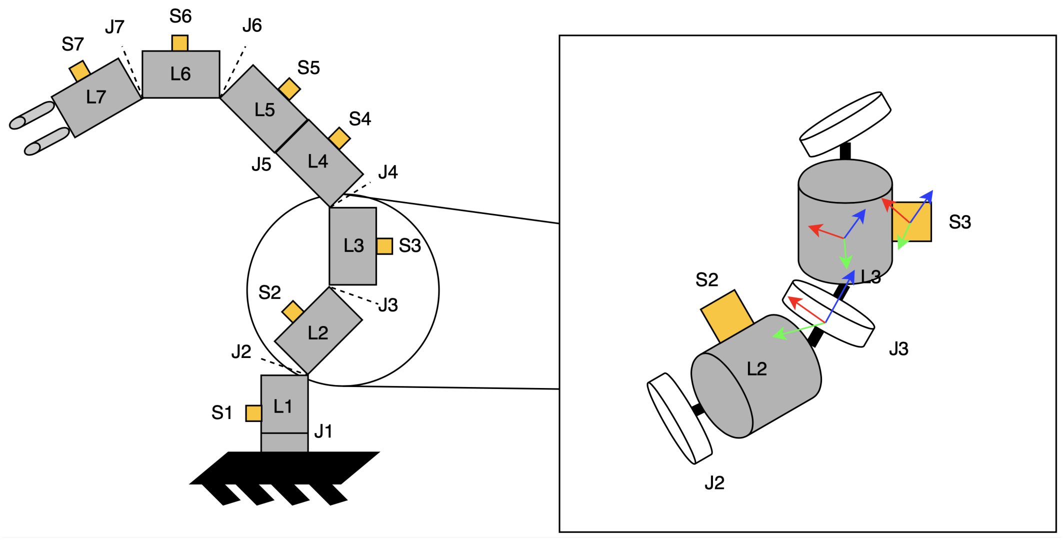
3.2.1 Kinematic Calibration

My work wouldn't have validity in the real world if it wasn't for calibration.

We present a framework for automatic kinematic calibration that leverages an IMU to accurately estimate the pose (position and orientation) of skin units, of arbitrary number, along the surface of a robot.

To automatically locate skin units along the surface of a robot, we utilize the angular velocity and linear acceleration measurements from the IMUs. Our optimization algorithm estimates the pose (position and orientation) of an SU using *modified Denavit-Hartenberg (DH) parameters* [1] as illustrated in figure 3.4. The pose of each SU is estimated by six DH parameters with respect to the previous joint in the kinematic chain: four parameters from the joint to a virtual joint, and two additional parameters from the virtual joint to the SU (other two parameters are set to 0). A virtual joint is located within the link that is orthogonal to the joints z axis and the SUs z axis. This solution is necessary to adhere to the Denavit-Hartenberg notation, so that each transformation can be expressed with no more than four parameters.

Figure 3.4: Depiction of multiple skin units (S) placed on the robots links (L) and separated by joints (J). We estimate the Denavit-Hartenberg parameters of each joint in order to calculate the pose of each skin unit along the surface of the robot.



Our optimization algorithm is composed of the following **four steps**:

(1) **Initialize a kinematic chain with randomized values.** We represent each skin unit coordinate using a transformation matrix.

$$^{0}T_{SU_i} = {}^{0}T_1 \cdot {}^{1}T_2 \ldots {}^{i-1}T_i \cdot {}^{i}T_{SU_i}, \quad \forall i$$

, where:

$$^{i}T_{i+1} = \left[\begin{array}{c|c} {}^{i}R_{i+1} & {}^{i}\mathbf{p}_{i+1} \\ \hline \mathbf{0} & 1 \end{array}\right].$$

(2) **Collect Data.** First, static forces applied to the IMU (that is, the constant acceleration due to gravity) are measured and compensated for. Then, each reference joint is moved

through its operational range in a constant rotation pattern and the resulting acceleration as measured by the IMU is stored.

(3) **Define an error function.** Acceleration exerted on each SU ${}^{SU_i}a_{u,d}$ can be estimated as a composition of local acceleration, tangential acceleration and centripetal acceleration:

$${}^{RS}\mathbf{a}_{tan_{u,d}} = {}^{RS}\alpha_d \times {}^{RS}\mathbf{r}_{u,d}$$

$${}^{RS}\mathbf{a}_{cp_{u,d}} = {}^{RS}\omega_d \times \left({}^{RS}\omega_d \times {}^{RS}\mathbf{r}_{u,d}\right)$$

$${}^{SU_i}\mathbf{a}_{u,d} = {}^{SU_i}\underline{R}_{RS} \cdot \left({}^{RS}\mathbf{g} + {}^{RS}\mathbf{a}_{tan_{u,d}} + {}^{RS}\mathbf{a}_{cp_{u,d}}\right).$$

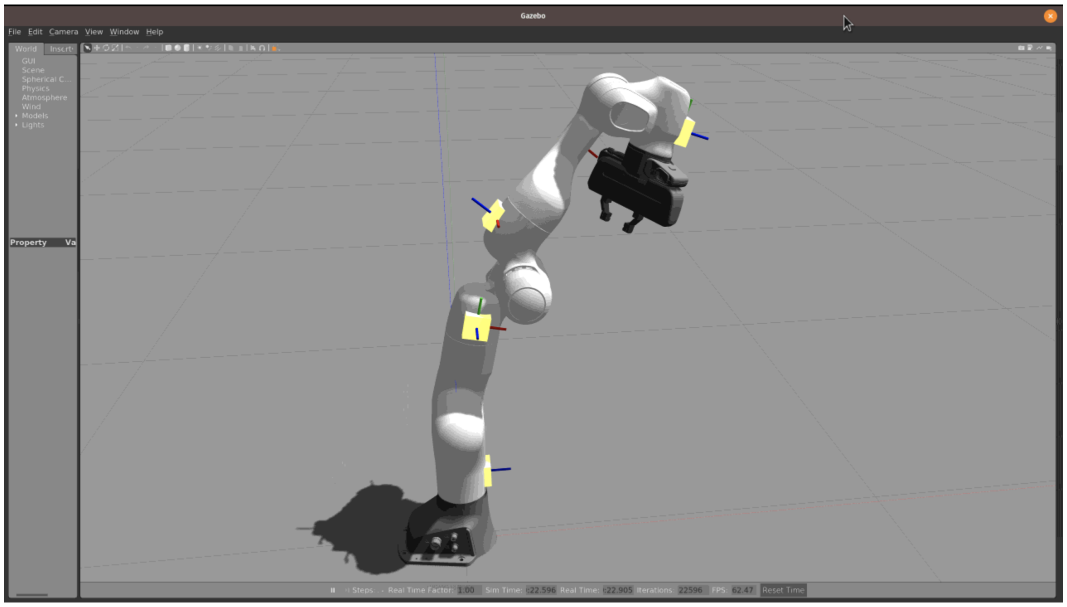
Angular velocity ω and angular acceleration α are measured during data collection, whereas rotation matrix R and position vector $\mathbf{p}$ can be computed using the current estimated DH parameters.

One example error function could be seen as the error between the measured accelerations from the IMUs and the estimated accelerations using the kinematic chain model for n_{pose} poses [2]:

$$E = \sum_{i=1}^{n_{pose}} \sum_{j=1}^{n_{joint}} ||a_{i,j}^{model} - a_{i,j}^{IMU}||^2$$

(4) **Minimize the error function with a global optimizer.** A global optimizer optimizes the DH parameters by minimizing the chosen error function. We combine several different error functions to estimate both rotational and translational parameters. The final result in simulation can be seen in figure 3.5.

Figure 3.5: Calibrated IMU positions on a simulated Franka Emika Panda robot.


Gazebo
File Edit Camera View Window Help
World Insert
GUI
Scene
Spherical C...
Physics
Atmosphere
Wind
Models
Lights
Property Va
Steps Real Time Factor: 1.00 Sim Time: 22.596 Real Time: 22.905 Iterations: 22596 FPS: 62.47 Reset Time

3.2.2 Location of points itself

3.3 Control

3.3.1 The Kinematic Control Problem

In this section one of the most fundamental problems in robotics will be defined and solved: the Kinematic Control Problem. It can be formulated as the problem of obtaining a joint space trajectory $\mathbf{q}(t)$ that satisfies a desired trajectory in the cartesian space $\mathbf{x}(t)$.

This problem contains several elements required in order to solve the collision avoidance problem in subsequent sections. Moreover, the collision avoidance will be built as a modification to the more fundamental method described in this section.

3.3.1.1 Geometric representation of the kinematic chain of a robot

Kinematics is the branch of physics that studies the motion of the bodies. It does that with no consideration of the causes that originate that motion, forces and torques. Kinematics is required and fundamental in robotics for design, analysis, control and simulation.

In order to study the kinematics of a robot its geometric representation must be defined mathematically beforehand. Rigid body kinematics deals with the problem of studying relative movement of two non-deformable elements by attaching a coordinate frame to each of them. Robot manipulators and robots in general can be seen as a series of rigid links connected by joints. Therefore, attaching a frame to each of the links of a robot in a convenient manner is how robots' geometry is usually represented.

However, there are infinitely many ways in which this 3-dimensional frames can be assigned to the links. The origin and rotation of the frames could be located anywhere in the space as long as its movement was fixed to the respective links.

That is where conventions for geometric representation come in place. The most well known and extended convention was introduced by Denavit and Hartenberg (DH) in 1955 [9]. Although there have been numerous adaptations, the representations serve the same purpose and DH parameters are still widely used. In fact, the manufacturer of the robot employed in this work (3.1.2) provides DH parameters in its documentation [8].

DH suggest a set of rules to place the frames corresponding to each robot link:

- The n links of the mechanism are numbered from 0 to N, being link 0 the fixed base.
- The n joints of the mechanism are numbered from 1 to N, with joint i located between links $i-1$ and i.
- The $\mathbf{z_i}$ axis is located along the axis of joint i.
- The $\mathbf{x_{i-1}}$ axis is located along the common perpendicular between $\mathbf{z_{i-1}}$ and $\mathbf{z_i}$ axis.

Then, they define 4 parameters that completely define the position and rotation of a link with respect to the previous one. Note that full position and orientation are usually defined by at least 6 parameters. However, geometric constraints inherent to robotic arms allow us to define them with just 4 parameters.

The four parameters are:

- $\mathbf{a_i}$: the distance from $\mathbf{z_{i-1}}$ to $\mathbf{z_i}$ along $\mathbf{x_{i-1}}$.
- $\alpha_\mathbf{i}$: the angle from $\mathbf{z_{i-1}}$ to $\mathbf{z_i}$ about $\mathbf{x_{i-1}}$.
- $\mathbf{d_i}$: the distance from $\mathbf{x_{i-1}}$ to $\mathbf{x_i}$ along $\mathbf{z_i}$.
- $\theta_\mathbf{i}$: the angle from $\mathbf{x_{i-1}}$ to $\mathbf{x_i}$ about $\mathbf{z_i}$.

Most common joint types in robot manipulators are revolute and prismatic joints. In the case of **revolute joints** all the parameters but $\theta_\mathbf{i}$ are defined by the geometric design of the robot, while $\theta_\mathbf{i}$ is the variable of the movement. If the joint is **prismatic**, all the parameters but $\mathbf{d_i}$ are defined by the geometry of the robot, while $\mathbf{d_i}$ is the variable of the movement.

Franka Panda Emika (described in section 3.1.2) is equipped with 7 revolute joints. Its DH parameters are given by the manufacturer and shown in figure 3.3.

In ROS, the robots' geometry is internally described in *urdf* files. *urdf* files make use of *Extensible Markup Language (XML)*, the same markup language that is used for web pages in *html* files. The following fragment taken from the complete descripion of Franka Panda (by justagist [10]) Emika describes the transformation from frame 6 to frame 7.

```
<joint name="${arm_id}_joint7" type="revolute">                    1
      <safety_controller k_position="100.0"                         2
                         k_velocity="40.0"                          3
                         soft_lower_limit="-2.8973"                 4
                         soft_upper_limit="2.8973"/>                5
      <origin rpy="${pi/2}␣0␣0" xyz="0.088␣0␣0"/>                   6
      <parent link="${arm_id}_link6"/>                              7
      <child link="${arm_id}_link7"/>                               8
      <axis xyz="0␣0␣1"/>                                           9
      <limit effort="12" lower="-2.8973"                            10
                         upper="2.8973"                             11
                         velocity="2.6100"/>                        12
      <dynamics damping="1.0" friction="0.5"/>                      13
</joint>                                                            14
```

Listing 3.1: Fragment of panda_arm.urdf

The relative positions and orientations are defined under the tag `<origin/>` in line 6. DH parameters are not used in this tag, even though *urdf* could be adapted to use them through the language *xacro*, that permits to do math inside textiturdf. Rotation is defined through *Euler Angles*: rotations are defined by applying rotation about x, y and z axes, in this order. Translation is defined with a 3×1 vector that goes from frame $i-1$ to frame i, from 6 to 7 in this case.

The use of DH parameters would have been beneficial and less error prone through the process of writing the *urdf* description.

3.3.1.2 Forward Kinematics

Once the robot has been represented with its DH parameters, a mathematical tool is needed to find the position and orientation of the end effector (or any other point in the kinematic chain) given all the joint variables' values ($\theta_\mathbf{i}$s for revolute joints and $\mathbf{d_i}$s for prismatic joints). This is the definition of the forward kinematics problem that will be exposed in this section.

A common way to represent the rotation and translation from one axis to another are homogeneous transformation matrices. They are 4×4 matrices defined as in equation 3.1.

$$^{i-1}\mathbf{T}_i = \begin{pmatrix} ^{i-1}\mathbf{R}_i & ^{i-1}\boldsymbol{p}_i \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \tag{3.1}$$

The top left 3×3 matrix $^{i-1}\mathbf{R}_i$ defines the rotation from axis $i-1$ to axis i. $^{i-1}\mathbf{R}_i$ is an orthogonal matrix, this meaning that the basis formed by its columns is orthonormal. Each of the column vectors are required to have unit length. One nice property of orthogonal matrices is that their inverse is exactly the same as their transpose:

$$(^{i-1}\mathbf{R}_i)^{-1} =^i \mathbf{R}_{i-1} = (^{i-1}\mathbf{R}_i)^{\intercal}$$

On the other hand, the top right 3×1 position vector $^{i-1}\boldsymbol{p}_i$ defines the translation vector from axis $i-1$ to axis i.

Homogeneous transformation matrices are an easy and convenient way to calculate several linked transformations. $^{i-2}\mathbf{T}_i$ would be calculated as follows:

$$^{i-2}\mathbf{T}_i =^{i-2} \mathbf{T}_{i-1}^{i-1}\mathbf{T}_i = \begin{pmatrix} ^{i-2}\mathbf{R}_{i-1} & ^{i-2}\boldsymbol{p}_{i-1} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} ^{i-1}\mathbf{R}_i & ^{i-1}\boldsymbol{p}_i \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} = \begin{pmatrix} ^{i-2}\mathbf{R}_{i-1}^{i-1}\mathbf{R}_i & (^{i-2}\boldsymbol{p}_{i-1} +^{i-2} \mathbf{R}_{i-1}^{i-1}\boldsymbol{p}_i) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}$$

This is the desired behaviour because $^{i-2}\mathbf{R}_{i-1}^{i-1}\mathbf{R}_i$ represents the combination of applying the rotation $^{i-2}\mathbf{R}_{i-1}$ first and $^{i-1}\mathbf{R}_i$ after. $(^{i-2}\boldsymbol{p}_{i-1} +^{i-2} \mathbf{R}_{i-1}^{i-1}\boldsymbol{p}_i)$ takes the vector $^{i-1}\boldsymbol{p}_i$ to the $i-2$ basis and them it sums the origin of the frame $i-1$, $^{i-2}\boldsymbol{p}_{i-1}$, already written in the $i-2$ axis.

The homogeneous transformation matrix $^{i-1}\mathbf{T}_i$ can be built from the DH parameters representation with $\mathbf{a_i}$, $\alpha_\mathbf{i}$, $\mathbf{d_i}$ and $\theta_\mathbf{i}$. For that purpose the following steps are combined in order:

(1) $\alpha_\mathbf{i}$ rotation about the axis x_{i-1}.

(2) $\mathbf{a_i}$ translation through the axis x_{i-1}.

(3) $\theta_\mathbf{i}$ rotation about the axis z_i.

(4) $\mathbf{d_i}$ translation about the axis z_i.

$$^{i-1}\mathbf{T}_i = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_i \\ \sin\theta_i\cos\alpha_i & \cos\theta_i\cos\alpha_i & -\sin\alpha_i & -\sin\alpha_i d_i \\ \sin\theta_i\sin\alpha_i & \cos\theta_i\sin\alpha_i & \cos\alpha_i & \cos\alpha_i d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Once all the the transformation matrices $^{i-1}\mathbf{T}_i$ are defined for $i = 1, \ldots, n$, the full transformation from the base $i = 0$ to the end effector $i = n$ transformation can easily be calculated:

$$^0\mathbf{T}_n =^0 \mathbf{T}_1^1\mathbf{T}_2 \ldots^{n-1} \mathbf{T}_n$$

Forward Kinematics are calculated in the robot_state_publisher package [11]. The robot's description is read from a *urdf* file loaded in the parameter server. Then, the joints' values are read from a topic named */joint_states* and used to compute forward kinematics of all the joints. The results are published in a topic named */tf*.

3.3.1.3 Inverse Kinematics

The inverse Kinematics problem is the opposite to the one of Forward Kinematics (section 3.3.1.2). The objective in this case is not to locate the robot's parts once known joint values, but to find the joint values take the end effector (or any other point in the kinematic chain) to a given position and orientation.

Given the nature of the homogeneous transformation matrices defined in section 3.3.1.2, whose elements contain sin and cos functions, the problem requires to solve a set of non linear equations defined by:

$$^{0}\mathbf{T}_n(q_1, \ldots, q_n) = (^{0}\mathbf{T}_n)^d$$

It is trivial to see that solving the position $^{0}\boldsymbol{p}_n(q_1, \ldots, q_n) = (^{0}\boldsymbol{p}_n)^d$ describes 3 equations. On the other hand, since rotation matrices provide six auxiliary relationships (3 coming from the fact that column vectors to have unit length, and 3 coming from the requirement of its column vectors being mutually orthogonal), and they contain 9 elements, $9 - 6 = 3$ more equations are coming from $^{0}\mathbf{R}_n(q_1, \ldots, q_n) = (^{0}\mathbf{R}_n)^d$.

In order a solution to exist, the desired location and rotation $(^{0}\mathbf{T}_n)^d$ needs to be in the robot's workspace.

In addition, depending on the number of DoF n of the manipulator, these equations will have **no solution**, will have **only one solution** or will have **infinitely many solutions**. Having into account that the problem provides 6 equations, if $n < 6$ then there will exist no solutions. If $n = 6$ there will only exist one solution. If $n > 6$ there will be infinitely many solutions that will satisfy all 6 equations.

A closed form of the equation does not always exist. In fact, in the common case there is no way to do this. Consequently, numerical methods are needed.

For Franka Panda Emika $n = 7$, so there are infinitely many joint values that satisfy the Inverse Kinematics equations.

3.3.1.4 Forward Instantaneous Kinematics

Forward instataneous Kinematics relates the motion rates of joints $\mathbf{q} = \begin{bmatrix} q_1 & \ldots & q_n \end{bmatrix}^\intercal$ with the velocities in the 3D Cartesian Space of the end effector.

Note that everything defined for the specific point in the kinematic chain *end effector* will also be adaptable to any other point in the kinematic chain, by simply considering a reduced kinematic chain.

We start by redefining what we got in section 3.3.1.2. This was our final result, given in homogeneous transformation matrices:

$$^{0}\mathbf{T}_n =^0 \mathbf{T}_1^1\mathbf{T}_2 \ldots ^{n-1}\mathbf{T}_n = \begin{pmatrix} ^{0}\mathbf{R}_n & ^{0}\boldsymbol{p}_n \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}$$

The position of the end effector is well defined by the position vector $^{0}\boldsymbol{p}_n$. However, the rotation matrix $^{0}\boldsymbol{p}_n$, by definition of rotation matrix, contains six auxiliary relationships (3.3.1.2). In consequence, rotation matrices are not minimal representations. For Forward Instantaneous Kinematics we will use a minimal representation of orientation, such as Euler angles (3.3.1.1), where orientation is completely described with just three elements.

$$\mathbf{t}(t) = \mathbf{f}(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \\ f_5(t) \\ f_6(t) \end{bmatrix}$$

$\mathbf{f}$ is a differentiable nonlinear vector function. $f_1(t)$, $f_2(t)$ and $f_3(t)$ describe the position of the end effector and $f_4(t)$ $f_5(t)$ and $f_6(t)$ its orientation.

Thus, the rate of change of this parameters is given by $\dot{\mathbf{t}}(t)$,

$$\dot{t}_i(t) = \frac{dt_i(t)}{dt} = \sum_{i=1}^{n} \frac{\partial t_i}{\partial q_i} \cdot \frac{dq_i}{dt} = (\nabla_{\mathbf{q}} t_i)^{\intercal} \cdot \dot{\mathbf{q}}$$

In matrix form,

$$\dot{\mathbf{t}} = \frac{d\mathbf{t}}{dt} = \frac{\partial \mathbf{t}}{\partial \mathbf{q}} \cdot \dot{\mathbf{q}} = \mathbf{J_t}(\mathbf{q}) \cdot \dot{\mathbf{q}}$$

$\mathbf{J_t}(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is called the *Jacobian.* It relates joint space velocities to cartesian space velocities and it is a function of the joint angles (or distances in the case of prismatic joints) $\mathbf{q}$.

It is convenient to remark that, with this formulation, the components of the velocity vector $\dot{\mathbf{t}}(t)$ express the rate of change of the parameters that form the minimal representation vector $\mathbf{t}(t)$. In Rigid Body Kinematics, the velocities of all the points in a rigid body (a link in our case) are completely defined by two vectors [12].

(1) The *angular velocity* of the body $\boldsymbol{\omega}$.

(2) The velocity of one point pertaining to the body $\mathbf{v}_O$.

Given those two vectors the velocity $\mathbf{v}_P$ of any other point P pertaining to the same body can be calculated as:

$$\mathbf{v}_P = \mathbf{v}_O + \boldsymbol{\omega} \times \overrightarrow{\mathbf{OP}} \tag{3.2}$$

However, while the end effector velocity represented by the first three elements of $\dot{\mathbf{t}}(t)$ is suitable for $\mathbf{v}_O$, the last three elements of $\dot{\mathbf{t}}(t)$ don't express the angular velocity $\boldsymbol{\omega}$ of the end effector link. They represent the rate of change of Euler Angles or any other minimal representation parameters chosen to describe orientation.

The actual Jacobian that relates the joint space velocities to $\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v}_{\text{end effector}} \\ \boldsymbol{\omega}_{\text{end effector}} \end{bmatrix}$ is:

$$\mathbf{J}(\mathbf{q}) = \mathbf{T}(t) \cdot \mathbf{J}_{\mathbf{t}}(\mathbf{q})$$

$$\mathbf{T}(t) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}(t) \end{bmatrix}$$

$\mathbf{T}(t)$ only depends on time t and its expression depends on the minimal representation chosen to describe orientation. The resulting equation is:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q}) \cdot \dot{\mathbf{q}} \tag{3.3}$$

There are many methods that compute the values of the jacobian matrix $\mathbf{J}(\mathbf{q})$ [13]. The general idea underlying is similar in all of them, with differences in efficiency.

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \boldsymbol{\mathcal{J}}_{P1} & \boldsymbol{\mathcal{J}}_{P2} & \cdots & \boldsymbol{\mathcal{J}}_{Pn} \\ \boldsymbol{\mathcal{J}}_{O1} & \boldsymbol{\mathcal{J}}_{O2} & \cdots & \boldsymbol{\mathcal{J}}_{On} \end{bmatrix}$$

$\boldsymbol{\mathcal{J}}_{Pi} \in \mathbb{R}^{3\times1}$ denotes the contribution of $\dot{q}_i$ per radian to $\mathbf{v}_{\text{end effector}}$. $\boldsymbol{\mathcal{J}}_{Oi} \in \mathbb{R}^{3\times1}$ denotes the contribution of $\dot{q}_i$ per radian to $\boldsymbol{\omega}_{\text{end effector}}$. Their value is computed as follows:

$$\boldsymbol{J}_{Pi} = \begin{cases} {}^{0}\mathbf{z}_{i-1} & \text{if joint } i \text{ is prismatic} \\ {}^{0}\mathbf{z}_{i-1} \times^{i-1} \boldsymbol{p}_{\text{end effector}} & \text{if joint } i \text{ is revolute} \end{cases}$$

$$\boldsymbol{J}_{Oi} = \begin{cases} \mathbf{0} & \text{if joint } i \text{ is prismatic} \\ {}^{0}\mathbf{z}_{i-1} & \text{if joint } i \text{ is revolute} \end{cases}$$

As described in section 3.3.1.1, ${}^{0}\mathbf{z}_{i-1}$ is joint i's z axis, which is a unitary vector in the direction of the joint. ${}^{0}\mathbf{z}_{i-1}$ coincides both the direction in which a prismatic joint contributes to the end effector velocity $\mathbf{v}_{\text{end effector}}$ and with the contribution of a revolute joint to the angular velocity of the end effector $\boldsymbol{\omega}_{\text{end effector}}$. The magnitude $\mathbf{z}_{i-1} \times^{i-1} \boldsymbol{p}_{\text{end effector}}$ is due to the $\boldsymbol{\omega} \times \overrightarrow{\mathbf{OP}}$ part in equation 3.2. It is obvious that the contribution of a prismatic joint to $\boldsymbol{\omega}_{\text{end effector}}$ is $\mathbf{0}$.

KDL

3.3.1.5 Inverse Instantaneous Kinematics

As defined at the beginning of 3.3.1, the kinematic control problem deals with the problem of obtaining a joint space trajectory $\mathbf{q}(t)$ that satisfies a desired trajectory in the cartesian space $\mathbf{x}(t)$. Several approaches can be taken in order to find a solution to this problem, but according to [14], the most relevant ones for redundant manipulators (sections 3.1.2 and 3.3.1.2) solve the problem at the joint velocity level.

Therefore, joint space velocities $\dot{\mathbf{q}}(t)$ that result in end effector velocities $\dot{\mathbf{x}}_{\mathbf{d}}$ that satisfy a desired trajectory in the cartesian space $\mathbf{x}_{\mathbf{d}}(t)$ must be found. This is the opposite problem to the one presented in the previous section (3.3.1.4) that was resolved in equation 3.3. The equation presents a linear system of equations in the unknowns $\dot{\mathbf{q}}$. This contrasts with the fact that, as stated in section 3.3.1.3, the Inverse Kinematics problem is non-linear. The jacobian $\mathbf{J}(\mathbf{q})$ is a local linearization of the instantaneous problem, which is valid only for the specific time instant t and joint values $\mathbf{q}$ for which it was calculated. In the general task that than be identified by m variables:

$$\underset{m\times n}{\mathbf{J}(\mathbf{q})} \cdot \underset{n\times 1}{\dot{\mathbf{q}}} = \underset{m\times 1}{\dot{\mathbf{x}}_{\mathbf{d}}} \tag{3.4}$$

It is important to note that the maximum number of variables a task can be identified with is $m = 6$. This implies that for a redundant robot for which $n >= 7$, $m > n$ will always remain true.

In the following paragraphs we will utilize some definitions and theorems of Linear Algebra to verify that the linear system of equations defined by equation 3.4 has a set of solutions that can only be either empty or infinite. We also show that the maximum rank of $\mathbf{J}(\mathbf{q})$ is its number of rows m. When the $\mathbf{J}(\mathbf{q})$ is full rank, i.e. when its rank is exactly m, the vector space of solutions has dimension $(m - n)$. When the rank is $< m$, there exists no solution.

We start by listing some definitions and theorems of Linear Algebra (taken from [15]. However this are well known results and definitions and can be found in any general linear algebra book).

Definition 1. *The **row space/column space** of a matrix is the span of the set of its rows/columns. The **row rank/column rank** is the dimension of this space, the number of linearly independent rows/columns.*

Theorem 2. *For any matrix, the row rank and column rank are equal.*

Definition 3. *The **rank** of a matrix is its row rank or its column rank.*

Theorem 4. *No linearly independent set can have a size greater than the dimension of the enclosing space.*

Therefore, by 6, it is obvious that the maximum rank for a given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the minimum value between m and n. The rank of this matrix can only be <= than this maximum rank. In the case of $\mathbf{J}(\mathbf{q})$ in a redundant robot this is given by the dimension of the row space m.

Theorem 5. *For linear systems with n unknowns and with matrix of coefficients* $\mathbf{A}$*, the statements*

(1) the rank of $\mathbf{A}$ *is* r

(2) the vector space of solutions of the associated homogeneous system has dimension $(n\text{-}r)$

are equivalent.

The homogeneous system associated to equation 3.4, $\mathbf{J}(\mathbf{q}) \cdot \dot{\mathbf{q}} = \mathbf{0}$, consequently has a solution space of dimension $(\mathrm{n} - \mathrm{r}) \geq (\mathrm{n} - \max(\mathrm{r})) = (\mathrm{n} - \mathrm{m})$. In the case of Panda Franka Emika, the solution space of the associated homogeneous system will always be greater than or equal to 1: $(\mathrm{n} - \mathrm{r}) \geq (\mathrm{n} - \max(\mathrm{m})) = (7 - 6)$.

Therefore when the rank of $\mathbf{J}(\mathbf{q})$ becomes smaller than m, the ability to control one axis either of translation or rotation is lost. The positions q in which this situation is given are called singularities. The reader may think that it would be useful to have a measure of when one of this situations is about to take place. [16] introduced a measure that indicates how close a robot is from a singularity:

$$\sqrt{\mathbf{J}^{\intercal} \cdot \mathbf{J}}$$

This value tends to 0 when the vector q tends to a position in which the rank of $\mathbf{J}$ will be decreased.

Summing up, the problem has been defined as a set of linear equations (equation 3.4) and the form of the solutions has been studied. Next, we must solve the set of equations.

Pseudoinverse

However, doesn't avoid singularities.

Leageouis

3.3.1.6 Control Problem

```cpp
void moveToPosition(const Eigen::Vector3d xd)
{
    Eigen::VectorXd qDot;
    positionErrorVector = xd - x;

    while (positionErrorVector.norm() > position_error_threshold
            && ros::ok())
    {
        readEndEffectorPosition();
        readControlPointPositions();
        positionErrorVector = xd - x;
        xdDot = pGain * positionErrorVector;
        ros::spinOnce();
        qDot = IIK(xdDot)
        jointVelocityController.sendVelocities(qDot);
        rate.sleep();
    }
    qDot = Eigen::VectorXd::Constant(7, 0.0);
    jointVelocityController.sendVelocities(qDot);
}

Eigen::VectorXd IIK(Eigen::Vector3d xdDot)
{
    J = kdlSolver.computeJacobian("end_effector", q);
    Jpinv = J.completeOrthogonalDecomposition().pseudoInverse();
    Eigen::VectorXd qDot1, qDot2;
    qDot1 = Jpinv * xdDot;
    qDot2 = secondaryTaskGain *
            ((Eigen::MatrixXd::Identity(7,7) - Jpinv*J) *
            secondaryTaskFunctionGradient(q));
    return qDot1 + qDot2;
}

Eigen::VectorXd secondaryTaskFunctionGradient(Eigen::VectorXd q)
{
    Eigen::VectorXd qMid, qRanges;
    qMid = 0.5 * (jointLimitsMax + jointLimitsMin);
    qRanges = jointLimitsMax - jointLimitsMin;
    return 2.0/7.0 * (q - qMid).cwiseQuotient(qRanges);
}
```

Listing 3.2: CartesianPositionController.cpp

3.3.2 Flacco

3.3.3 Quadratic programming

Chapter 4

Results

youtube links, charts... from flacco, QP simulations and real robot if posible.

illustrations, block diagrams (http://dia-installer.de/) powerpoint

Wait for this last part.

Chapter 5

Conclusions

introduction summary

one short paragraph about

results

future work

Bibliography

[1] G Westling and RS Johansson. Factors influencing the force control during precision grip. Experimental brain research, 53(2):277–284, 1984.

[2] Jan BF van Erp and Hendrik AHC van Veen. Touch down: the effect of artificial touch cues on orientation in microgravity. Neuroscience Letters, 404(1-2):78–82, 2006.

[3] Ravinder S Dahiya, Giorgio Metta, Maurizio Valle, and Giulio Sandini. Tactile sensingfrom humans to humanoids. IEEE transactions on robotics, 26(1):1–20, 2009.

[4] S. Tian, F. Ebert, D. Jayaraman, M. Mudigonda, C. Finn, R. Calandra, and S. Levine. Manipulation by feel: Touch-based control with deep predictive models. In 2019 International Conference on Robotics and Automation (ICRA), pages 818–824, 2019.

[5] Achu Wilson, Shaoxiong Wang, Branden Romero, and Edward Adelson. Design of a fully actuated robotic hand with multiple gelsight tactile sensors. arXiv preprint arXiv:2002.02474, 2020.

[6] Ayssam Elkady and Tarek Sobh. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. Journal of Robotics, 2012, 05 2012.

[7] Apex.AI. Apex.OS^{R}, 2020. https://www.apex.ai/apex-os.

[8] Franka Emika. Franka emika panda robot and interface specifications, 2020. https://frankaemika.github.io/docs/control_parameters.html#.

[9] Richard S Hartenberg and Jacques Denavit. A kinematic notation for lower pair mechanisms based on matrices. Journal of applied mechanics, 77(2):215–221, 1955.

[10] justagist. Franka emika panda robot description, 2020. https://github.com/justagist/franka_panda_description.

[11] Robot state publisher ros package, 2020. wiki.ros.org/robot_state_publisher.

[12] José María Goicolea Ruigómez. Curso de Mecánica. 2010.

[13] David E. Orin and William W. Schrader. Efficient Computation of the Jacobian for Robot Manipulators. The International Journal of Robotics Research, 3(4):66–75, 1984.

[14] Bruno Siciliano. Kinematic control of redundant robot manipulators: A tutorial. Journal of intelligent and robotic systems, 3(3):201–212, 1990.

[15] Jim Hefferon. Linear algebra, 3rd.

[16] Tsuneo Yoshikawa. Dynamic manipulability of robot manipulators. Transactions of the Society of Instrument and Control Engineers, 21(9):970–975, 1985.

[17] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. IEEE Transactions on Robotics, 33(6):1292–1312, 2017.

[18] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3356–3363, 2008.