Lab 3 Report

Robotics Integration Group Project I

Yuwei ZHAO (23020036096)
Group #31 2025-11-26

Abstract

See Resources on github.com/RamessesN/Robotics_MIT.

1 Introduction

2 Procedure

2.1 Individual Work

2.1.1 Transformations in Practice

1. **MESSAGE VS. TF**

- **Assume we have an incoming `geometry_msgs::Quaternion quat_msg` that holds the pose of our robot. We need to save it in an already defined `tf2::Quaternion quat_tf` for further calculations. Write one line of C++ code to accomplish this task.**

  Solution:
  To convert a `geometry_msgs::Quaternion` into a `tf2::Quaternion`, simply initialize the latter with the x, y, z, w components of the incoming message:

```
quat_tf = tf2::Quaternion(quat_msg.x, quat_msg.y, quat_msg.z, quat_msg.w);
```

```
tf2::Quaternion::Quaternion ( const tf2Scalar & x,
                              const tf2Scalar & y,
                              const tf2Scalar & z,
                              const tf2Scalar & w
                            )                    [inline]
```

Constructor from scalars.

Definition at line 36 of file Quaternion.h.

Figure 1: tf2 Quaternion doc

- **Assume we have just estimated our robot's newest rotation and it's saved in a variable called `quat_tf` of type `tf2::Quaternion`. Write one line of C++ code to convert it to a `geometry_msgs::Quaternion` type. Use `quat_msg` as the name of the new variable.**

Solution:

```
geometry_msgs::Quaternion quat_msg = tf2::toMsg(quat_tf);
```

- If you just want to know the scalar value of a `tf2::Quaternion`, what member function will you use?

2. **CONVERSION**

- Assume you have a `tf2::Quaternion quat_t`. How to extract the yaw component of the rotation with just one function call?

- Assume you have a `geometry_msgs::Quaternion quat_msg`. How to you convert it to an Eigen 3-by-3 matrix? Refer to this for possible functions. You probably need two function calls for this.

2.1.2 Modelling and control of UAVs

1. STRUCTURE OF QUADROTORS
2. CONTROL OF QUADROTORS

2.2 Team Work

2.2.1 Trajectory tracking for UAVs

2.2.2 Launching the TESSE simulator with ROS bridge

2.2.3 Implement the controller

2.2.4 Simulator conventions

2.2.5 Geometric controller for the UAV

3 Reflection and Analysis

4 Conclusion

5 Source Code

•

•