Record For Processing IMU Data

Mingze Chen

February 24, 2021

1 Gathered Data Form

A ros package called *playground* was created and inside the package a **Python** script called *listener.py*:

```
#!/usr/bin/env python
import re
import os.path
import argparse
import csv
import datetime
import rospy
from std_msgs.msg import String
from rosbot_ekf.msg import Imu
from nav_msgs.msg import Odometry

args = None


def write_to_csv(filename, dic):
    file_exists = os.path.isfile(filename)
    # print "File exists: ", file_exists
    with open(filename, 'a') as csvfile:
        headers = ['seq', 'secs', 'nsecs', 'x', 'y', 'z', 'w',
                   'av_x', 'av_y', 'av_z', 'la_x', 'la_y', 'la_z', 'time']
        file_is_empty = os.stat(filename).st_size == 0
        header_writer = csv.writer(csvfile, lineterminator='\n')
        writer = csv.DictWriter(csvfile,
                                delimiter=',',
                                lineterminator='\n',
                                fieldnames=headers)

        if file_is_empty:
            header_writer.writerow(headers)
            # writer.writeheader()  # file doesn't exist yet, write a header

        writer.writerow(
            {headers[i]: dic[i]
             for i in range(len(dic))})


def callback(data):
    # 'Imu' object is not iterable
    # rospy.loginfo(rospy.get_caller_id() + " Type of data: %s", type(data))
    rospy.loginfo(rospy.get_caller_id() + "␣I␣heard␣%s", data)
    data = str(data)
    # lin_acc_ind = data.find('linear_acceleration')
    # There are 3 values indicating linear acceleration
    # rospy.loginfo(rospy.get_caller_id() + " Index for linear acceleration is: %s", lin_acc_ind)
    res = re.findall(r"[-+]?\d*\.\d+|\d+", data)
    # combine secs and nsecs to complete timestamp
    res.append(float(res[1]) + float(res[2]) * 10 ** (-9))

    if args.save and args.save.endswith('.csv'):
        write_to_csv(args.save, res)
    else:
        current_date = datetime.datetime.now()
        # name the csv file according to date automatically
        filename = 'data' + str(current_date.day) + \
            str(current_date.month) + str(current_date.year)
        write_to_csv(str(filename + '.csv'), res)


def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber("mpu9250", Imu, callback)
    # rospy.Subscriber("odom", Odometry, callback)
    rospy.spin()

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-s', '--save', required=False,
                        help="save␣to␣<filename>")

    args = parser.parse_args()
    listener()
```

Through running the following command :

```
husarion@husarion:~/pathTo/catkin_ws$ source ./devel/setup.bash
husarion@husarion:~/pathTo/catkin_ws$ roscore
```

start a 2nd. command line window and execute:

```
$ roslaunch rosbot_ekf all.launch rosbot_pro:=true
```

start a 3rd. command line window and execute:

```
$ rosrun playground listener.py –s angularVel.csv
```

At the same time, through running motor controller the robot runs along a trace like in figure 1:

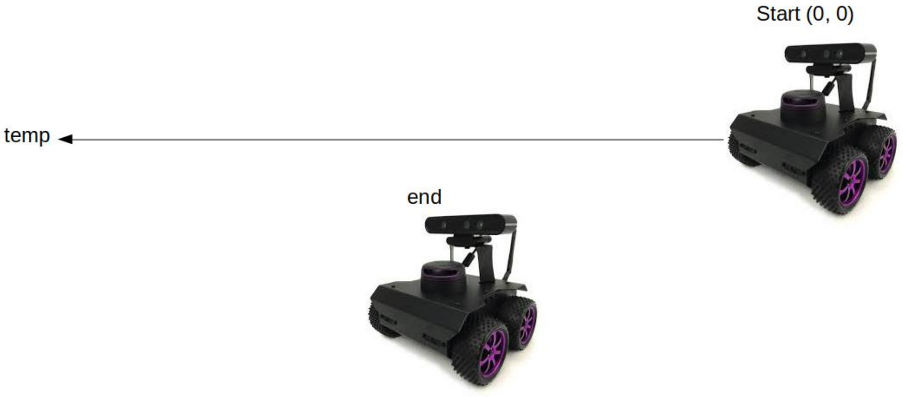

Figure 1: Running process of the robot

The robot moved forward from start to temp and then backward from temp to end.

The partial output from the command line is:

```
  [INFO] [1607701130.617211]: /listener_7171_1607701129181 I heard header:
  seq: 17352
  stamp:
    secs: 1607701130
    nsecs: 603677978
  frame_id: ''
orientation:
  x: 0.00418900698423
  y: -0.00127400737256
  z: 0.296933829784
  w: 0.954888045788
angular_velocity: [2.2560975551605225, 1.2195122241973877, 1.2804877758026123]
linear_acceleration: [-0.01806640625, -0.001953125, -1.0166015625]
```

Collected data was saved in a csv file called *angularVel.csv* and angular velocity along all 3 axes against time were plotted, as shown in figure 2:

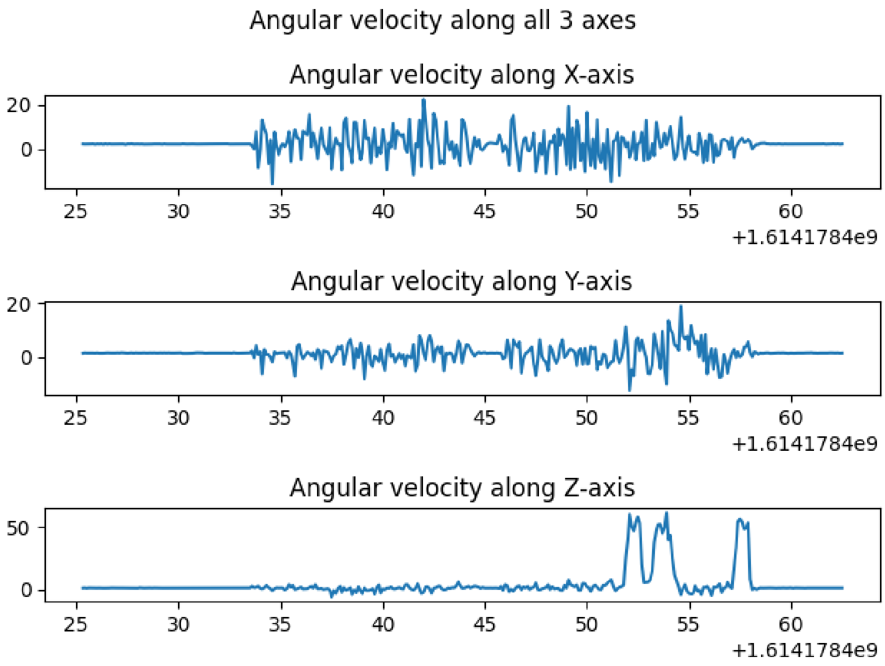

Figure 2: Angular velocity along all 3 axes during running process of the robot

2 Idea for next step

a. Based on collected angular velocity estimate position of the robot and compare it with *Pose Data From PoseStamped.*

b. Running repeated experiment, e.g. line segment trajectory

c. Consider using Graph Neural Networks, not only EKF

d. Consider designing a random walk model for the final demonstration