Autonomous GPS Waypoint Navigation Project

The system objective is to steer differential-drive vehicle from its current position to the next waypoint, utilizing the GPS signals. The project developed in ROS and consists of one node that subscribes to two topics and publishes two topics, and one class.

The Node has a class and three functions plus the main function, it includes the following files as dependences however it uses the geometry_msgs/TwistStamped to steer the vehicle.

```
#include <ros/ros.h>
#include <sensor_msgs/NavSatFix.h>
#include <nav_msgs/Path.h>
#include <tf/transform_broadcaster.h>
#include <geometry_msgs/TwistStamped.h>
#include <geometry_msgs/TransformStamped.h>
#include <visualization_msgs/Marker.h>
#include <visualization_msgs/MarkerArray.h>
#include <gps_navigation/GpsConversion.h>
```

Main function

In this function I initialized all the variables and the ROS Publishers used in the system, and creating an ROS timer. I used an two arrays to populate the Latitude and the Longitude and push them to the GPS navigation class to get the East North UP coordinates, and populate these points on global arrays X and Y to use on other functions, and used for the Marker message to show the waypoints on RVIZ.

GpsConversion Class

The main functionality for this class is to receive GPS Latitude, Longitude and Altitude, and provide a North East UP X, Y and Z coordinates, it is a replica from the one created in the class.

recvGpsVel

This function is a call back function for subscribing to the gps_vel topic, it receives a TwistStamped. Its main functionality is to compute the heading angle by receiving the vehicle current X, Y coordinates and compute the angle by taking the arctan for the X and Y and store it in a global variable to use in other function.
And create Quaternion for the Yaw and populate the transformation message.

recvGps

This function is a call back function for subscribing to gps_fix topic, it receives sensor_msgs. This function dose the following:

1- Receiving the msg GPS points and convert them to ENU.
2- Push the points to the path_msg to populate the path that appears in RVIZ.
3- Compute the angle between the vehicle and the waypoint by computing the point's difference for both and then take the arctan of the new X and Y point.

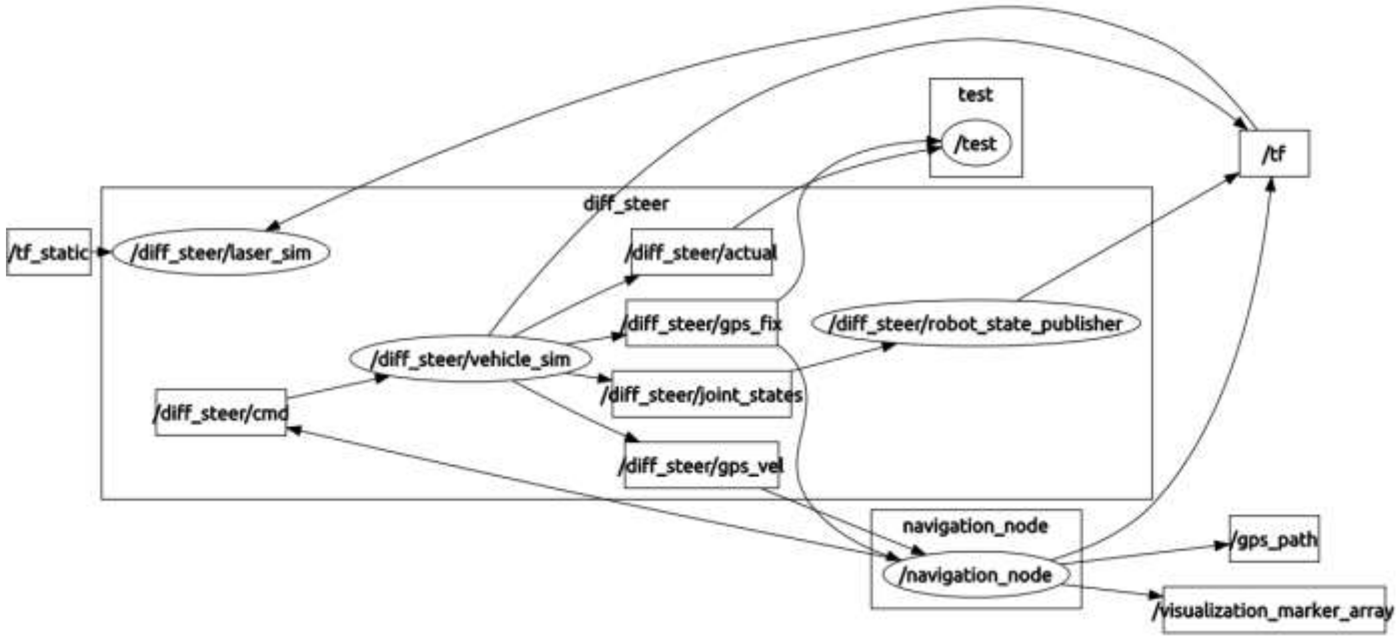
4- From the above angle we can compute the error angle by subtracting it from the heading angle.
5- Check if the vehicle reached the waypoint or within 0.5 meter from it if so change the waypoint to the next one by increasing the array index.
6- Check if the vehicle reached the end point and set a flag.
7- Populate the TF transform message with the ENU coordinate points.

timerCallback

this function is called every (0.05) second as we specified in the creation on the main function, in this function we do the following.

1- Stamp the transform message and specify parent and child frames.
2- Send the transform to the TF.
3- Check the flag we set in the recvGps function to indicate that the vehicle reached the end point or not, and set the twist angular Z with either the error angle multiplied by some gain to steer the vehicle or to zero to stop the vehicle.
4- Populate ROS publishers for the entire system.

Following is the rqt_graph



Project variables

1- **Package name: gps_navigation**
2- **Launch file name: gps_navigation.launch**