COMP3431

Robotic Software Architecture

Assignment 2: Report

Nathan ADLER
Aneita YANG

November 9, 2015

Contents

1 Introduction

In this assignment, both the hardware and software aspects of robotics are explored. The overall objective was to create a robot that could drive autonomously in an outdoor environment, whilst avoiding any obstacles. A motorised wheelchair was the baseline from which the robot was constructed.

To achieve the objective, the robot is equipped with a GPS and compass (using an Android phone with ROS). A laser scanner is also attached to the front of the robot, gathering information about the robot's immediate surroundings.

1.1 Modules

1.1.1 Hardware

1.1.2 Software

On the software side, five/six(?????) nodes run in conjunction to operate the robot:

`dest_sender` keeps track of the robot's remaining waypoints.

`gps_drive` publishes the direction in which the robot needs to travel to reach its destination.

`rtk_gps_pub` ...

`laser_safe` publishes movement messages, either directly to the bot's destination, or to avoid an obstacle.

`motordata_arduino_send` ...

`sick_tim` ... (TODO: reference the github)

2 Hardware

3 Software

TODO: Generate rqt_graph of nodes talking to each other

3.1 Planner

The planner module is responsible for keeping track of the robot's waypoints and informs the robot of its next destination.

The `dest_sender` node parses a file containing the GPS coordinates of waypoints and turns each latitude-longitude pair into a NavSatFix message. Each waypoint is stored in a list (as a NavSatFix message), with the head of the list being the robot's next destination. `dest_sender` publishes this message to the `/ugv_nav/waypoints` topic for other modules to subscribe to.

`dest_sender` subscribes to the `/ugv_nav/arrived` topic, which signals when the robot has reached its destination. Messages which are published to the `/ugv_nav/arrived` topic trigger a callback which removes the current waypoint from the list (i.e. the head of the list). If the robot has not reached its final destination and the list is not empty, the robot's next destination is published.

3.2 Localisation

The localisation module is responsible for calculating the heading from the robot's current position to its destination and determines a direction in which the robot should travel to reach its goal.

To assist with the autonomous driving, the robot is equipped with an Android phone running the `android_sensors_driver` app which publishes:

- Camera images: `sensor_msgs/CompressedImage` and `sensor_msgs/Image`
- Fluid pressure data: `sensor_msgs/FluidPressure`
- Illuminance data: `sensor_msgs/Illuminance`
- Accelerometer: `sensor_msgs/Imu`
- Magnetic field: `sensor_msgs/MagneticField`
- GPS fixes: `sensor_msgs/NavSatFix`
- Temperature data: `sensor_msgs/Temperature`

For the purposes of this assignment, data gathered from the phone's GPS fixes and magnetic fields are used to calculate the bearing of the destination from the robot. The `gps_drive` node performs these calculations - the Android device publishes the GPS coordinates of the robot to the `/phone1/android/fix` topic and the GPS coordinates of the destination are published to the `/ugv_nav/waypoints` topic by the `dest_sender` node. The Android device also publishes information about the robot's bearings to `/phone1/android/magnetic_field`.

$$\text{difference\_lat} = \text{destination\_latitude} - \text{source\_latitude}$$
$$\text{difference\_long} = \text{destination\_longitude} - \text{source\_longitude}$$

Supposing the robot is facing true north, then:

$$\theta = \tan^{-1}\left(\frac{\text{difference\_long}}{\text{difference\_lat}}\right),$$

where θ is the angle the robot must turn to reach its destination.

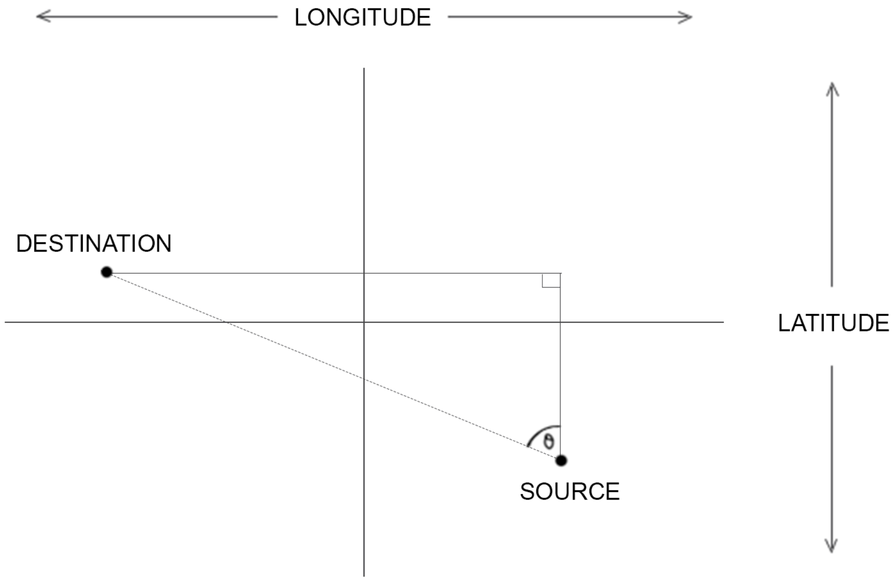

Figure 1: Calculation of the bearing of the destination from the robot, if the robot is facing true north.

As the ???th decimal place provides up to ??? precision:

$$\text{distance\_to\_destination} = 111,000 * \sqrt{(\text{difference\_lat})^2 + (\text{difference\_long})^2}$$

If the distance to the destination is less than 5 (i.e. the robot is within 5 metres of its destination), it is considered to have arrived at its destination. At this stage, the `gps_drive` node will publish a message to the `/ugv_nav/arrived` topic, signalling to the `dest_sender` that the robot has reached its destination and that a new destination is required.

As the robot obtains GPS fixes frequently, it is sensitive to small changes to its position. The frequent updating of the robot's position results in abrupt movement changes as the robot attempts to face the bearing to its destination. In order to reduce the amount of sudden directional changes, rather than continually looking for a straight, direct path to its goal, the direction the robot decides to travel is biased towards a 'smooth path'.

TODO: Explain straight line bias calculations

3.3 Waypoint Traversal

lasersafe, motordata

3.4 Open Source SICK TiM Driver

4 Results

5 Future Work and Improvements

6 Appendix

6.1 sensor_msgs/NavSatFix.msg

```
Header header

/* Satellite fix status information */
NavSatStatus status

/* Latitude [degrees]. Positive is north of equator; negative is south. */
float64 latitude

/* Longitude [degrees]. Positive is east of prime meridian; negative is west. */
float64 longitude

/*
Altitude [m]. Positive is above the WGS 84 ellipsoid
(quiet NaN if no altitude is available).
*/
float64 altitude

/*
Position covariance [m^2] defined relative to a tangential plane
through the reported position. The components are East, North, and
Up (ENU), in row-major order.
Beware: this coordinate system exhibits singularities at the poles.
*/
float64[9] position_covariance

/*
If the covariance of the fix is known, fill it in completely. If the
GPS receiver provides the variance of each measurement, put them
along the diagonal. If only Dilution of Precision is available,
estimate an approximate covariance from that.
*/
uint8 COVARIANCE_TYPE_UNKNOWN = 0
uint8 COVARIANCE_TYPE_APPROXIMATED = 1
uint8 COVARIANCE_TYPE_DIAGONAL_KNOWN = 2
uint8 COVARIANCE_TYPE_KNOWN = 3

uint8 position_covariance_type
```