FIRST ROBOTICS PROJECT

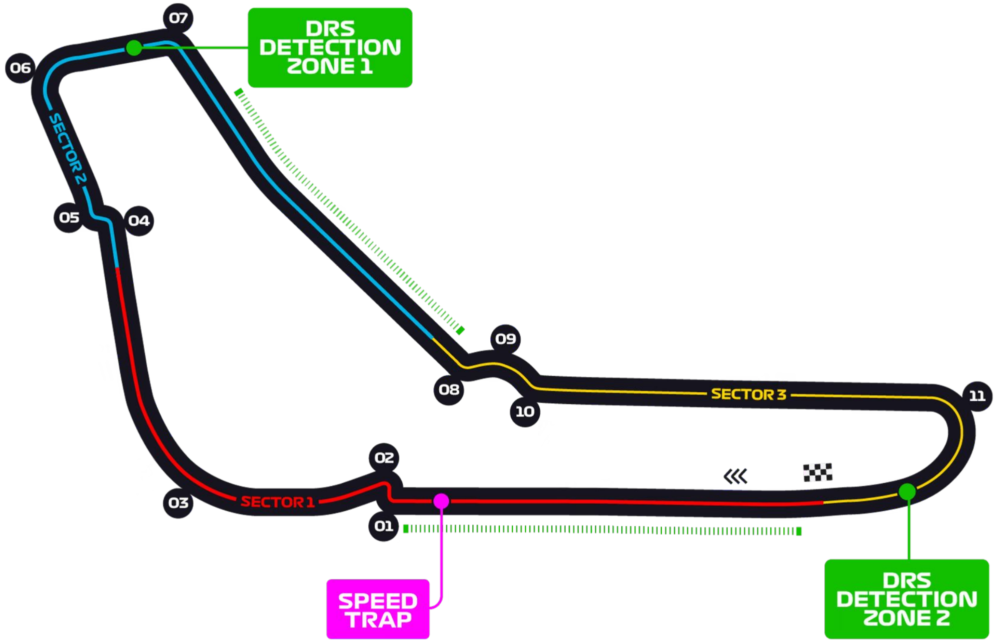
ROBOTICS

POLITECNICO
MILANO 1863

THE PROBLEM

The car

The circuit



DATA

Format: ROS Bag file

play the bag with the command:

rosbag play --clock project.bag

Data:

- /speedsteer vehicle data
- /swiftnav/front/gps_pose gps front data
- /swiftnav/rear/gps_pose gps rear data

VEHICLE

Vehicle data

rear wheels baseline: 130 cm

distance from front to rear wheels:

176.5 cm

steering factor: ~32 (check it with GPS)

/speedsteer topic:

y -> speed (km/h)

x -> steer at the steering wheel (deg)

The car

THE PROJECT (First Node)

- Create a ROS package called ***first_project***
- Create a ROS node to compute the odometry from vehicle data:
  - write a node called ***odometer*** which
    - subscribe to:
      - vehicle status message:
        - type: ***geometry_msgs/PointStamped***
        - topic name: ***/speedsteer***
    - publish:
      - odometry message:
        - type: ***nav_msgs/Odometry***
        - topic name: ***/odom***
      - **tf odom-vehicle**

THE PROJECT (Second Node)

- Create a ROS node to compute the odometry from gps data:
  - write a node called ***gps_odometer*** which
    - subscribe to:
      - vehicle status message:
        - type: ***sensor_msgs/NavSatFix***
        - topic name: ***/swiftnav/front/gps_pose***
    - publish:
      - odometry message:
        - type: ***nav_msgs/Odometry***
        - topic name: ***/gps_odom***
      - **tf odom-gps**

THE PROJECT (Second Node)

- to convert gps data to odometry:
  - convert (latitude-longitude-altitude) to Cartesian ECEF
  - convert Cartesian ECEF to ENU
  - ENU is a relative position, so you need to specify the reference point
- gps_to_odom should have three parameters :
  - lat_r
  - lon_r
  - alt_r
- These parameters should be set in a launch file, for this project you can set manually to the first value from GPS:

THE PROJECT (Second Node)

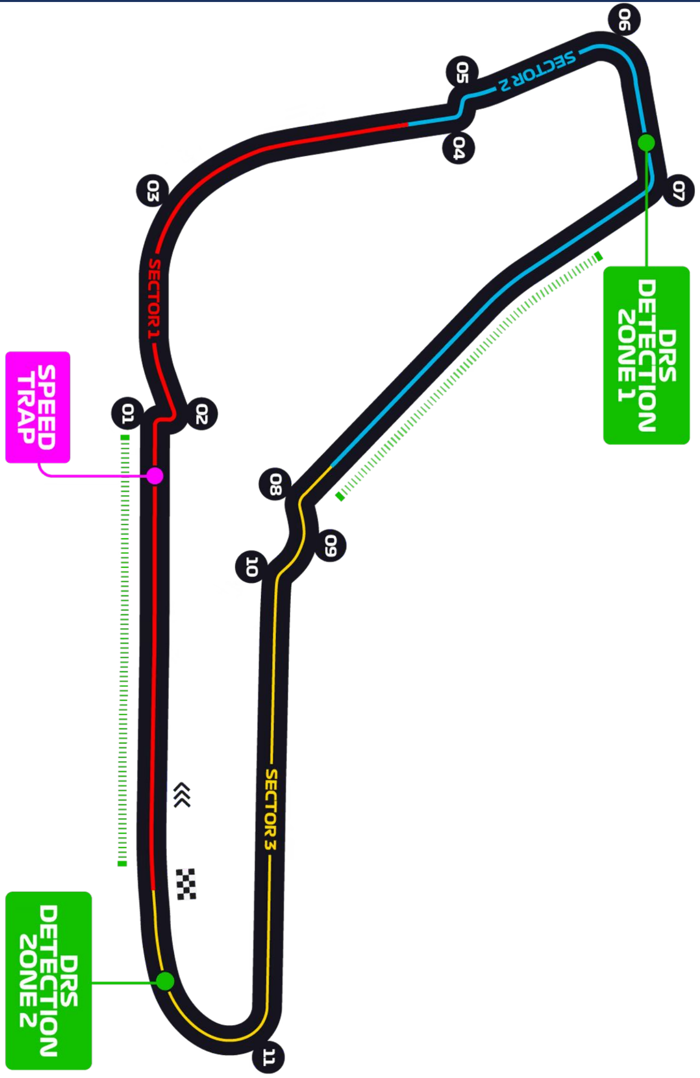
- GPS gives you only position
- For the odometry you also want orientation
- In this scenario we work on a 2D plane, so we want the car heading
- After computing the car position in ENU you can use consecutive poses to estimate the robot heading (if you want you can also add a smoothing filter)

THE PROJECT (Third Node)

- Create a ROS node to compute the sector times and speed:
  - write a node called ***sector_times*** which
    - subscribe to:
      - vehicle status message and gps:
        - topic name: ***/speedsteer***
        - topic name: ***/swiftnav/front/gps_pose***
    - publish custom message:
      - sector_time message:
        - type: ***first_project/secotor_times***
        - topic name: **/sector_times**

THE PROJECT (Third Node)

- ***first_project/secotor_times structure:***
  - *int: current_sector*
  - *float: current_sector_time*
  - *float: current_sector_mean_speed*



THE PROJECT (Launch file)

- The three node should all start from a single launch file called **launch.launch**
- The launch file should also open rviz loading a configuration file which shows the two tfs and the odometry topics, with a blue arrow for odom and red for gps_odom
- the only command to start the project should be:
  roslaunch first_project launch.launch

Deadlines and requested files

-Send **only** a zip file

-Upload on Webeep

-Inside the archive:

- info.txt file (details next slide)
- folders of the nodes you created (with inside CmakeLists.txt, package.xml, etc...)
- **do not send** the entire environment (with build and devel folders)
- **do not send** the bag files

Deadlines and requested files

File txt must contain only the group names with this structure

codice persona;name;surname

You can add another file called readme.txt with additional info. I will not always look for it. But if something goes wrong I'll check for explanations.

Some more requests

Name the archive with your codice persona

Don’t use absolute path

The project need to be written using c/c++

Deadlines and requested files

Deadline: 1 May (4 weeks)

Max 3 student for team

Questions:

-write to me via mail (simone.mentasti@polimi.it)

- do not write only to Prof. Matteucci

Formulas

Latitude-longitude to ECEF:

$$X = (N(\phi) + h)\cos\phi\cos\lambda$$
$$Y = (N(\phi) + h)\cos\phi\sin\lambda$$
$$Z = \big(N(\phi)(1 - e^2) + h\big)\sin\phi$$

Where Φ is latitude, λ is longitude, *h* is altitude, N is defined as

$$N(\phi) = \frac{a}{\sqrt{1 - e^2\sin^2\phi}},$$

and *a* is the semi major axis of the equatorial radius, *b* is the semi minor axis of the polar radius, and e^2 is defined as

$$e^2 = 1 - \frac{b^2}{a^2}$$

where a=6378137 m

b=6356752 m

Formulas

ECEF to ENU:

You need both robot position and reference position (lat_r, lon_r, alt_r) in ECEF Coordinates

Then you can apply the formula:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\sin\lambda_r & \cos\lambda_r & 0 \\ -\sin\phi_r\cos\lambda_r & -\sin\phi_r\sin\lambda_r & \cos\phi_r \\ \cos\phi_r\cos\lambda_r & \cos\phi_r\sin\lambda_r & \sin\phi_r \end{bmatrix} \begin{bmatrix} X_p - X_r \\ Y_p - Y_r \\ Z_p - Z_r \end{bmatrix}$$

Where Φ_r is latitude, λ_r is longitude of the reference point (n.b., you are using sin and cos, so make sure they are in radians/degree, depending on the library you use)