Summer Internship
Report

Combination of homing and boundary following for Near-Optimal navigation

Submitted by

Ridhi Puppala
Undergraduate Researcher
Indian Institute of Technology Madras
Adyar, Chennai 600036

Under the guidance of

Leena Vachhani
Associate Professor

Systems and Control Engineering
Indian Institute of Technology Bombay
Powai, Mumbai 400076

Preface

The author is a pre-final year undergraduate student enrolled in a five year Interdisciplinary dual degree program at Indian Institute of Technology Madras (IITM) through which he will be awarded a B.Tech in Mechanical Engineering and an M.Tech in Robotics. He will be graduating in July 2020. This report documents his summer research internship at ARMS Lab associated with Systems and Control Engineering, Indian Institute of Technology Bombay (IITB). The ten week internship was accomplished during the summer of 2019.

Internship was mainly focused on formulation of novel navigation controller for robots under limited access to sensor measurements. The work attempts to solve the motion planning problem of mobile robots which have access to only limited local information to reach a static or moving target while avoiding obstacles. The proposed methods uses a combination of homing and boundary tracking for solving target tracking and obstacle avoidance problems independent. Proofs of convergence guarantee for such systems has been partially included in this report. The main focus of this report will be on the intricate details of practical implementation of proposed method on simple differential drive robots simulated in Gazebo (ROS) platform. It will explain the procedures of simulated experiments, difficulties faced while implementation and how they were resolved. It also attempts to examine the theoretical connection of these implementation issues.

The report discuss the limitations of the proposed method based on different environment settings, robot specifications or methods of implementation. The report also encompasses motivation, objectives, problem formulation and learning outcomes apart from theoretical understanding, proofs and practical implementation details in order to give a complete overview of the work accomplished during the internship. Report is written with an intention to document the work to an extent and also serve as a reference for future extension of this work.

Contents

Chapter 1

Disclaimer

The report contains partial details of research work accomplished which is yet to be published. The work is confidential! Please refrain from using any of the contained ideas for your research or projects. This report was to mainly focused on expounding the details of practical implementation. It specifically highlights practical implementations issues and how they were solved for simulations. This report is yet to finished due to my busy schedule.

Chapter 2

Introduction

2.1 Motivation

Motion and path planning play a significant role in autonomous mobile robotics. Global pose and map information is considered to be crucial for autonomous navigation. Environments populated with obstacles that lack accurate GPS or accurate map information like indoor environments pose a great challenge for navigation. In several practical scenarios, noise in sensor measurements or mapping can also hinder planning and navigation. In the recent past, research community has shown great interest in simultaneous localization and mapping (SLAM) and other related techniques for solving localization and planning problems in completely unknown environments. The downside of these solutions are a need for sophisticated sensors, high-end processors with large memory support for storing and processing information real-time which in turn demand higher power consumption.

Consider a scenario where robot has to approach a target in an obstacle populated environment but has access to only local information of what is around the robot, but has no knowledge of the environment or global pose information of target or itself. It would be impossible to prove that robot will converge to the target always. But if we introduce an additional information of relative orientation or sense of orientation of robot with respect to the target at every instant there is a possibility to solve this problem and prove that robot will successfully reach the target while dodging the obstacles. Through this work we aim to solve robot motion planning problem efficiently even under extreme limitations of sensory information and computational processing power of robot. Consequently, this would lead to lower power consumption which in turn reduces the robot's battery weight requirements. This serves as a great advantage for aerial or underwater robots.

2.2 Homing and Boundary Following - An Introduction and Comparative Study

In biology, ***Homing*** *is the inherent ability of an animal to navigate towards an original location through unfamiliar areas.* Inspired by this behavior, researchers have been applying similar ideas for autonomous navigation [1]. Homing vector is defined as a unit vector originating from centre of the robot and points directly towards the home (target) position for navigation problem or previously known location for kidnapped robot localization. Mobile ground robots or aerial robots are commanded to orient towards homing vector and then move towards it through visual feedback which directly or indirectly encodes relative bearing information [2, 3, 4, 5]. Due to environmental constraints underwater robots can use acoustic beacon based methods for homing [6, 7, 8]. Robust steering control law for autonomous homing using panoramic images has been proposed in [9]. This control law has been adopted for homing as it just uses sign information of rate of change of bearing estimated directly from panoramic cameras. This robust control method ensures finite time convergence under known maximum percentage errors in commanded inputs.

Visual homing can be conveniently implemented on ground or aerial robots under the constraint that occlusion while viewing the goal is minimal or non-existent. We can use probabilistic approach for visual homing under dynamic obstacles proposed in [5] to overcome the issue of partial occlusions. Two or more hydrophones can be attached to underwater robots receiving acoustic signal from source (goal) for homing which is not significantly affected by obstacles.

Boundary following has been used as an obstacle avoidance technique in autonomous navigation working in combination with various path planning algorithms. Bug algorithms were the earliest planners that used contact sensor for boundary following algorithm. An **exit** or **leave** condition is necessary to move out of boundary following mode when used for global motion or path planning of robot. Boundary following algorithm proposed in [10] works in conjunction with artificial potential fields to move the robot to goal. A boundary following algorithm with instant goals has been used to achieve globally convergent path planning behavior in [11]. The proposed boundary following algorithm in both of these works are complicated and difficult to implement practically. Another significant drawback of these works is their exit condition's dependence on global pose information of robot and target, or at least the information of distance to goal.

Gyroscopic control based boundary following has been implemented for single robots [13] and formation of multiple robots [14]. Noise prone curvature estimation from range sensor measurements affects the performance of control law proposed in [13] under practical settings. Elegant boundary following model and shape dynamics proposed in this work has been adopted to devise a simple and practically efficient boundary following algorithm for our near-optimal finite-time globally convergent steering control law. Finite time convergence of proposed boundary following controller has been proved. Exit condition depends only on sign change in rate of change of bearing or relative bearing estimated as a part of homing but not on any global information. The proposed steering control law is globally convergent in finite time with access to only local information.

2.3 Problem Definition and Research Objectives

2.3.1 Problem Definition and Simulation Setup

In order to test the developed theoretical formulation, we implement the control law on an ideal simulation of differential drive robot in Gazebo. *Gazebo environment provides a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces for real-time simulations.* Main objective of my work was to simulate experiments with differential drive robots in obstacle populated worlds. Proofs have been presented for any value of linear velocity, but the simulations have been performed with unit linear velocity for simplicity. Control law was devised such that the commanded linear velocity is unit while steering control law was dependent on feedback from local information and relative bearing collected by sensors.
Differential drive robot specifications:

1. Circular Chassis with diameter of 0.3m attached with two drive wheels and two castor wheels for balancing
2. Hokuyo Laser Range Finder for local obstacle information
3. Visual/acoustic homing sensor, processors, control and power electronics

Modelling of inertia, collision and contacts for the robot has been done in correlation with actual physical models. A lower level velocity differential drive controller operates on two wheeled unicycle robot model with linear and angular velocity of robot's geometric centre as inputs. Simulations have been setup on ROS Indigo and Gazebo-7 running on Ubuntu 14.04. The Gazebo robot model as shown in Fig. 2.1 has been adopted from Github

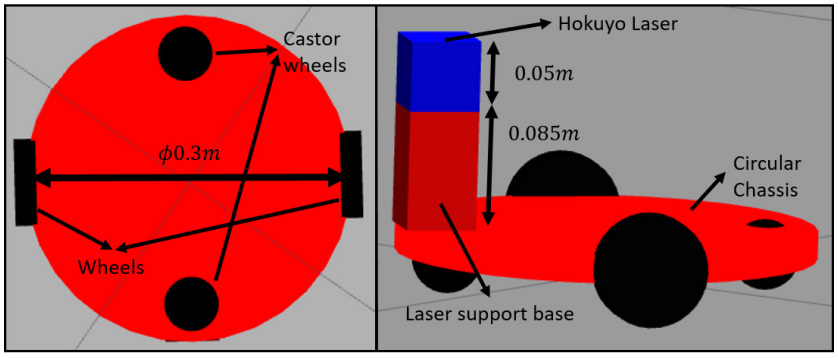


Figure 2.1: Differential drive robot description

repository of [15]. The robot description can be found in this Xacro file which is an extended version of URDF format.

2.3.2 Research Objectives

1. Theoretical Objectives
   (a) Develop globally convergent controllers for homing and boundary following modes of operation
   (b) Prove finite time convergence of both modes independently using concepts of Lyapunov Stability and Sliding Mode Control (SMC)
   (c) Prove stability of the overall system
   (d) Repeat the above three for moving target and static obstacles combination and then for moving target and dynamic obstacles.
2. Experimental Objectives
   (a) Practical implementation of control law in gazebo.
   (b) Analyze the implementation issues and resolve them while examining theoretical connection of these issues.
   (c) Iteratively modify and validate theoretical control law based on experimental results
   (d) Repeat the above three for moving target and static obstacles combination and then for moving target and dynamic obstacles.
   (e) Generate comparative results of proposed algorithm with existing optimal algorithms.

Chapter 3

Homing

Homing provides a sense of direction to robot in order to reach its goal. Visual and acoustic homing are two main types of homing for robots. Other possible methods might be based on triangulation of electromagnetic signals for estimating relative pose or bearing. In this chapter we present two different scenarios of homing as elucidated by Table 3.1.

State of the Target	Sensor Information
Static	Signum of gradient of bearing
Dynamic	Signum of relative bearing

Table 3.1: Two different scenarios of homing

3.1 Static Target Homing

Steering control law as proposed in [9] has been adopted for static target homing. *The paper's key contribution is a robust steering control law for autonomous homing where the robot's aligns its orientation along direction of the homing vector without explicitly estimating the homing vector but using only coarse information of bearing. Finite time convergence of robot to a small circle centered around the robot under bounded random errors is also proved.* A practical approach of estimating sign of gradient of bearing from panoramic images has also been presented. It also addresses practical issues like unknown image distance function, variation in environments, method of capturing panoramic images, camera parameters and execution errors.

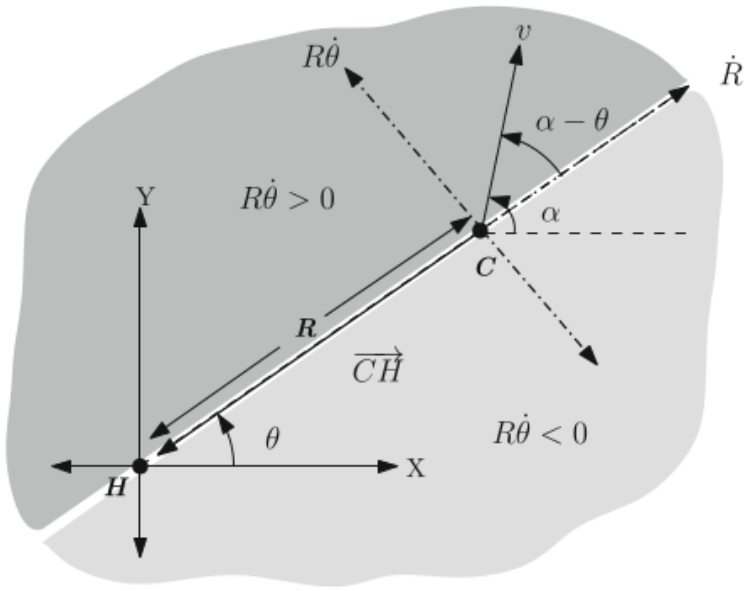


Figure 3.1: Engagement geometry with static target [9]

3.1.1 Theory

Let homing vector between current position $C(x, y)$ and home position $H(0, 0)$ denoted by $\overrightarrow{CH}$ can be described by $R(t) \in \mathbb{R}^+$ and $\theta(t) \in (-\pi, \pi]$ as shown in Fig. 3.1. Differential drive robot kinematics is identical to that of conventional unicycle kinematic model as shown in (3.1). Let the angle between instantaneous velocity vector $v(t)$ and X axis be $\alpha(t) \in (-\pi, \pi]$. Random errors in linear velocity $v(t) \in \mathbb{R}^+$ and angular velocity $u(t) \in \mathbb{R}$ are represented by $\delta_v, \delta_u \in \mathbb{R}$ respectively.

$$\begin{aligned} \dot{R} &= (v + \delta_v) \cos(\alpha - \theta) \\ R\dot{\theta} &= (v + \delta_v) \sin(\alpha - \theta) \\ \dot{\alpha} &= u + \delta_u \end{aligned} \tag{3.1}$$

Consider an ideal model with $\delta_v = \delta_v = 0$, Theorem 1 proposes a steering control law for homing and finite time convergence is proved. A more detailed and different approach to prove robustness and convergence of the same homing controller can be found in [9]. Partial sections of theorems and proofs from this paper have been reproduced here. The proof presented in this chapter is based on finite time reachability of sliding manifold borrowed from Sliding mode control theory.

Theorem 1 Steering control law u for system (3.1) is defined as

$$u = K_{hom} sgn(\dot{\theta}) \tag{3.2}$$

where R_{th} is radius of circle centered at home position, β is a positive free parameter, $\dot{\theta}$ is gradient of bearing angle and $sgn(x)$ is defined as,

$$sgn(x) = \begin{cases} -1 & x < 0 \\ +1 & x \geq 0 \end{cases}$$

Control law defined by (3.2) steers the robot the $K_{hom} > \frac{v_{max}}{R_{th}}$

Proof Let $s : \pi - (\alpha - \theta) = 0$ be the sliding surface of homing system of model described in 3.1. Physically this sliding surface represent the direction of homing vector. We can select $\sigma = \pi - (\alpha - \theta)$ as a switching function that forces the system to evolve to the selected sliding surface. For simplicity we limit the angular range of $\pi - (\alpha - \theta) to (-\pi, \pi]$ as a result of which $\alpha - \theta \in [0, 2\pi)$.

$$\begin{aligned} sgn(\dot{\theta}) &= sgn(\sin(\alpha - \theta)) \\ &= sgn(\sin(\sigma)) \\ &= sgn(\sigma) \end{aligned} \tag{3.3}$$

Consider a Lyapunov function candidate

$$\begin{aligned} V &= \frac{1}{2}\sigma^T\sigma \\ &= \frac{1}{2}(\pi - (\alpha - \theta))^2 \\ &= \frac{1}{2}\sigma^2 \end{aligned} \tag{3.4}$$

From 3.1 and $\pi - (\alpha - \theta) \in (-\pi, \pi]$, we arrive at 3.3. Differentiating 3.4 on both sides and using the above equations we get

$$\begin{aligned} \dot{V} &= \sigma^T\dot{\sigma} \\ &= |\sigma| sgn(\sigma)\dot{\sigma} \\ &= \sqrt{2V} sgn(\sigma)(\dot{\theta} - \dot{\alpha}) \\ &= -\sqrt{2V}(K_{hom} sgn(\dot{\theta}) sgn(\sigma) - \frac{v \sin(\alpha - \theta)}{R} sgn(\sigma)) \\ &= -\sqrt{2V}(K_{hom} - \frac{v \sin(\alpha - \theta)}{R} sgn(\sigma)) \end{aligned} \tag{3.5}$$

Since $\frac{v\sin(\alpha-\theta)}{R}sgn(\sigma) \leq |\frac{v\sin(\alpha-\theta)}{R}| \leq \frac{v_{max}}{R_{th}}$, we define $\mu = K_{hom} - \frac{v_{max}}{R_{th}}$.
Let v_{max} be maximum linear velocity and R_{th} be the radius of smallest circle around home position that the robot can enter. Using 3.5 and above inequality,

$$\begin{aligned} \dot{V} &< -\sqrt{2V}(K_{hom} - \frac{v_{max}}{R_{th}}) \\ &= -\sqrt{2V}\mu \end{aligned} \tag{3.6}$$

Using finite time sliding manifold reachability theorem borrowed from [16] and $\dot{V} < -\mu^{'}\sqrt{V}$ (where $\mu^{'} = \sqrt{2}\mu$), we can prove the finite time convergence to sliding surface which means the robot aligns to homing vector orientation in finite time. Since the robot can move to target after homing in finite time, we can conclude that system converges to static target or home position in finite time.

3.1.2 Implementation

ROS package called **mastering_ros_robot_description_pkg** contains Xacro files for robot and wheel description individually. All other files are unnecessary in this package. The robot description parameters like size of wheels or chassis; hokuyo laser parameters like angular range, distance range, resolution, samples and topic update frequency; differential drive parameters like wheel torque, acceleration, topic update frequency and publishing topic can be changed through **diff_wheeled_robot.xacro** file. Hokuyo laser range finder and differential drive plugin parameters are shown in Fig. 3.2 for reference. Another package **diff_wheeled_robot_gazebo** contains launch file called **diff_wheeled_gazebo_full.launch** to launch the Gazebo server, client (GUI) and spawn the robot in the world of our choice as specified in this file. Package named **diff_wheeled_robot_control** contains scripts of controller nodes for robot's navigation. In order to validate the above theoretical results, a circular robot as shown in Fig. 2.1 was programmed with static target homing control law. The robust steering control law in [9] capable of handling bounded random input errors is given as follows

$$u = \beta\frac{v}{R_{th}}sgn(\dot{\theta}) \tag{3.7}$$

where $\beta > 1 + \frac{max(\delta_v)}{v} + \frac{max(\delta_u)R_{th}}{v}$ for $R > R_{th}$. A unit linear velocity input and angular velocity as shown in 3.7 is commanded to the robot through the **cmd_vel** topic.

```
<gazebo>
  <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">

    <rosDebugLevel>Debug</rosDebugLevel>
    <publishWheelTF>false</publishWheelTF>
    <robotNamespace>/</robotNamespace>
    <publishTf>1</publishTf>
    <publishWheelJointState>false</publishWheelJointState>
    <alwaysOn>true</alwaysOn>
    <updateRate>100.0</updateRate>
    <leftJoint>front_left_wheel_joint</leftJoint>
    <rightJoint>front_right_wheel_joint</rightJoint>
    <wheelSeparation>${2*base_radius}</wheelSeparation>
    <wheelDiameter>${2*wheel_radius}</wheelDiameter>
    <broadcastTF>1</broadcastTF>
    <wheelTorque>30</wheelTorque>
    <wheelAcceleration>1.8</wheelAcceleration>
    <commandTopic>cmd_vel</commandTopic>
    <odometryFrame>odom</odometryFrame>
    <odometryTopic>odom</odometryTopic>
    <robotBaseFrame>base_footprint</robotBaseFrame>

  </plugin>
</gazebo>
```

```
</link>
<joint name="hokuyo_joint" type="fixed">
  <origin xyz="${base_radius - hokuyo_size/2} 0 ${base_height+2*hokuyo_size}" rpy="0 0 0" />
  <parent link="base_link"/>
  <child link="hokuyo_link" />
</joint>

<gazebo reference="hokuyo_link">
  <material>Gazebo/Blue</material>
  <turnGravityOff>true</turnGravityOff>
  <sensor type="ray" name="head_hokuyo_sensor">
    <pose>${hokuyo_size/2} 0 0 0 0 0</pose>
    <visualize>false</visualize>
    <update_rate>100</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>721</samples>
          <resolution>1</resolution>
          <min_angle>-3.141592</min_angle>
          <max_angle>3.141592</max_angle>
        </horizontal>

      </scan>
      <range>
        <min>0.01</min>
        <max>5.0</max>
        <resolution>0.001</resolution>
      </range>
    </ray>
    <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
      <topicName>scan</topicName>
      <frameName>hokuyo_link</frameName>
    </plugin>
  </sensor>
</gazebo>
```

Figure 3.2: Hokuyo laser range finder plugin and Differential drive control plugin parameters

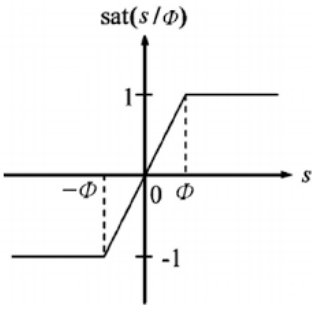


Figure 3.3: Saturation function with width of ϕ

In order to simplify the simulation, we use odometry information to calculate the gradient of bearing. The robot moves on XY plane, with x, y, α denoting the position and heading of robot w.r.to global frame is directly obtained from **nav_msgs/Odometry**. Here angle made by vector joining home position to robot centre $\overrightarrow{R}$ with X axis, $\theta = \arctan(y/x)$. Using Eq. 3.1 gradient of bearing $(\dot{\theta})$ is calculated. In a practical real world setting, they can be directly estimated from alignment sensors, cameras or beacons. For visual homing robot should be devoid of occlusions or it should be minimal.
Chattering is an undesirable consequence of Sliding mode control. In order to resolve this issue researchers have tries to replace sgn function with saturation functions denoted by sat, an example of which is shown in Fig. 3.3 [17, 18, 19]. Variation in path traced by robot for $sgn(\dot{\theta})$ and $sat(\dot{theta})$ is shown in Fig. 3.4. Practical implementation of saturation function is difficult for homing, but its effect on smoothening of robot's path can be observed through this simulated comparison. According to [9], $sgn(\dot{\theta})$ can be computed using minimal computations over panoramic images taken from the robot without actually calculating gradient of bearing $\dot{\theta}$.

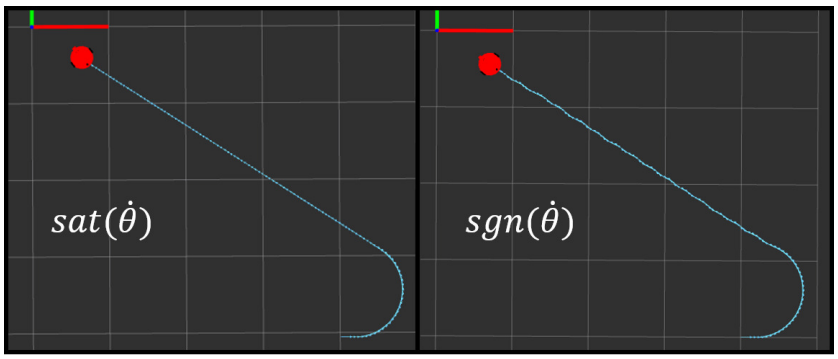


Figure 3.4: Comparison of path traced by robots for $sgn(\dot{\theta})$ and $sat(\dot{theta})$

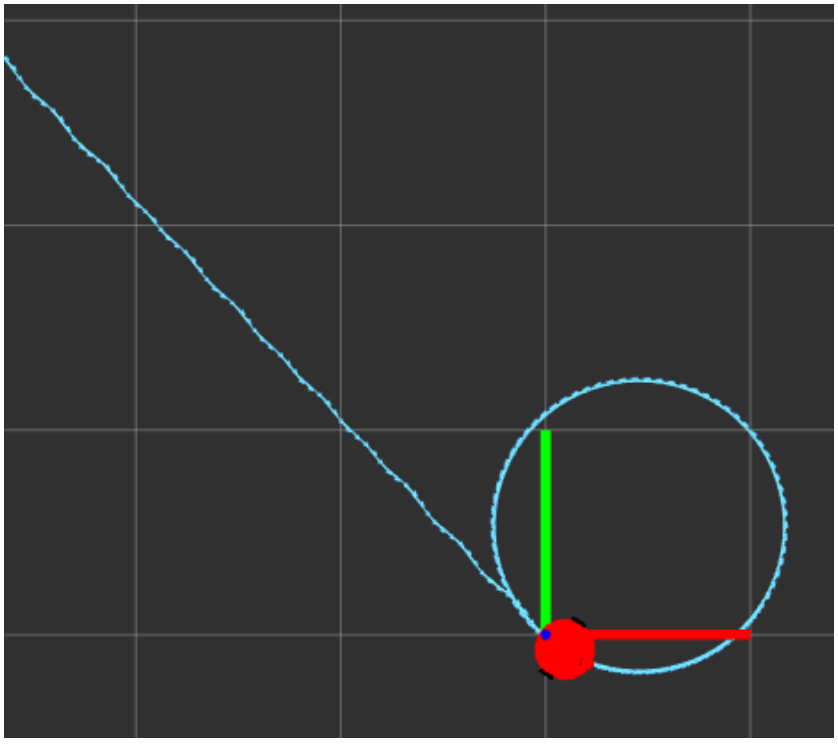


Figure 3.5: Path traced by robot after reaching the target

ROS package **satfunc_controller** contains python script **satfunc_controller_onlyBT.py** for implementation of only homing using saturation function. Package named **satfunc_controller** contains scripts of controller nodes for robot's navigation using Saturation functions for all the different cases implemented with just Sign functions in the package named **diff_wheeled_robot_control**. The consequence of steering controller for homing is that robot settles in a circular orbit of radius $R_{ss} = \frac{R_{th}}{\beta}$ it reaches very close to home position as seen in Fig. 3.5. This is the actual R_{th}. But practically, R_{th} is chosen to be sufficiently small for homing to a close proximity of the robot. It the target at home position has a finite size, then that should also be taken into account for selecting an appropriate R_{th}. Evident from 3.7, if a very small value of R_{th} is chosen then u is very high . It will be saturated to robot's maximum achievable angular velocity in experimental implementation. In the Gazebo simulation, transient dynamics in linear and angular velocity can be observed due to dynamics of lower level differential drive controller. Another observation is that steering controller becomes more aggressive with increasing β which is obvious from 3.7.

3.2 Dynamic Target Homing

3.2.1 Theory

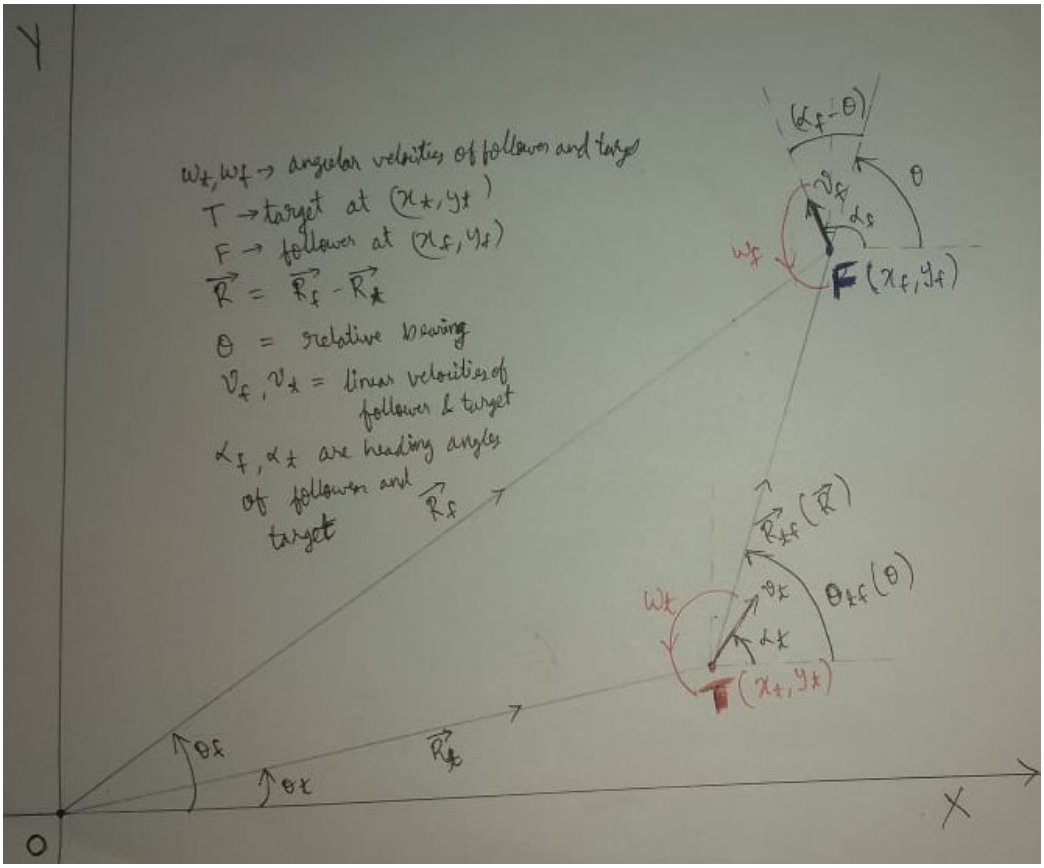


Figure 3.6: Engagement geometry with dynamic target

Dynamic target homing is modelled as a system with a follower robot homing towards a moving target. Kinematic model of such system is different from the model described in 3.1 due to differences in engagement geometry as shown in Fig. 3.6. Let target and follower robot denoted by $T(x_t, y_t)$ and $F(x_f, y_f)$ in the global XY plane centered at origin O. Let v_t and v_f be non-negative instantaneous velocities making angles $\alpha_t, \alpha_f \in (-\pi, \pi]$ with the X axis. $\overrightarrow{R_t}$ and $\overrightarrow{R_f}$ denote the global position vectors which make angles $\theta_t, \theta_f \in (-\pi, \pi]$ with the X axis. Angular velocities of the robots are represented as ω_t and ω_f. Similar to static homing, we propose a steering controller for follower and prove its finite time convergence using finite time reachability of sliding manifold borrowed from [16]. Instantaneous homing vector $\overrightarrow{FT}$ is

anti parallel to relative position vector R which makes an angle $\theta \in (-\pi, \pi]$ with X axis. Kinematic model for this system is described in 3.8.

$$\begin{aligned} \dot{R} &= v_f \cos(\alpha_f - \theta) - v_t \cos(\alpha_t - \theta) \\ R\dot{\theta} &= v_f \sin(\alpha_f - \theta) - v_t \sin(\alpha_t - \theta) \\ \dot{\alpha_f} &= u \quad \text{and} \quad \dot{\alpha_t} = \omega_t \end{aligned} \tag{3.8}$$

Theorem 2 Steering control law u for system (3.8) is defined as

$$u = K_{hom} sgn(\pi - (\alpha_f - \theta)) \tag{3.9}$$

where $\pi - (\alpha_f - \theta)$ denotes the relative bearing angle of follower w.r.to target and $K_{hom} > \frac{max(v_f + v_t)}{R_{th}}$.

Proof Let $s : \pi - (\alpha_f - \theta) = 0$ be the sliding surface of homing system of model described in 3.8. We select a switching function $\sigma = \pi - (\alpha_f - \theta)$ and limit the angular range of $\pi - (\alpha_f - \theta) to (-\pi, \pi]$. Using the fact that v_f and v_t are independent linear velocities and $R > R_{th}$, we can arrive at the following inequality.

$$\begin{aligned} \Big(\frac{v_f \sin(\alpha_f - \theta)}{R} - \frac{v_t \sin(\alpha_t - \theta)}{R}\Big) sgn(\sigma) &\leq \Big|\frac{v_f \sin(\alpha_f - \theta)}{R} - \frac{v_t \sin(\alpha_t - \theta)}{R}\Big| \\ &\leq \Big|\frac{v_f \sin(\alpha_f - \theta)}{R}\Big| + \Big|\frac{v_t \sin(\alpha_t - \theta)}{R}\Big| \\ &\leq \frac{max(v_f + v_t)}{R_{th}} \end{aligned} \tag{3.10}$$

Consider a Lyapunov function candidate identical to 3.5, $V = \frac{1}{2}\sigma^2$. Differentiating 3.4 on both sides and using 3.8 and the inequality 3.10 we get the following:

$$\begin{aligned} \dot{V} &= \sigma^T \dot{\sigma} \\ &= |\sigma| sgn(\sigma) \dot{\sigma} \\ &= \sqrt{2V} sgn(\sigma)(\dot{\theta} - \dot{\alpha_f}) \\ &= -\sqrt{2V} sgn(\sigma)\Big(K_{hom} sgn(\sigma) - \Big(\frac{v_f \sin(\alpha_f - \theta)}{R} - \frac{v_t \sin(\alpha_t - \theta)}{R}\Big)\Big) \\ &= -\sqrt{2V}\Big(K_{hom} - \Big(\frac{v_f \sin(\alpha_f - \theta)}{R} - \frac{v_t \sin(\alpha_t - \theta)}{R}\Big) sgn(\sigma)\Big) \\ &\leq -\sqrt{2V}\Big(K_{hom} - \frac{max(v_f + v_t)}{R_{th}}\Big) \end{aligned} \tag{3.11}$$

Using finite time sliding manifold reachability theorem borrowed from [16] and result obtained in 3.11, $\dot{V} < -\mu_{mt}\sqrt{V}$ where $\mu_{mt} = \sqrt{2}\Big(K_{hom} - \frac{max(v_f + v_t)}{R_{th}}\Big)$, we can prove the finite time convergence to sliding surface which means the robot aligns to homing vector orientation in finite time. Since the robot can move to target after homing in finite time, we can conclude that system converges to moving target in finite time.

3.2.2 Implementation

ROS package called **moving_target_gazebo** contains controller scripts and launch files for robots. For validating the control law as shown in 3.9, we have setup a simulation with follower robot homing towards target robot. Target and followers are controlled using **target_controller** and **follower_controller** python scripts in this package. Target is commanded with random angular velocity and a constant linear velocity with obstacle avoidance capability through boundary following controller. Follower is governed by the moving target homing controller and a constant linear velocity with obstacle avoidance capability through boundary following controller. Files for launching multiple followers and target in custom worlds (populated with static obstacles) have been written to easily scale the simulation to a swarm aggregation problem. In the **launch** directory of this package, **single_robot.launch** calls the robot description Xacro file; **all_robots.launch** assigns name space to multiple robots being spawned; custom Gazebo world client and server set with required parameters are launched using the **main.launch** and; ros nodes for follower and target controllers are launched using **follower_controller.launch** and **target_controller.launch**.
An implementation issue with ROS tf was encountered for moving target homing. This is common issue faced when working with multiple robot simulations in Gazebo. Each robot needs to assigned a name space and tf_prefix to distinguish the identically named topics like $/odom$, $/scan$, $/cmd\_vel$. In order to define a transformation between these two branches of tf tree a static transform publisher is introduced as shown in Fig. 3.7. Starting pose of robots can also be initialized as shown in Fig. 3.7. The tf trees for static and moving target homing have been attached at the end of this report (7).
One of the important observation is that follower robot's linear velocity should be greater than that of target to be able to converge in finite time always which can be easily concluded through intuition. Laser scanners are places at a height such that obstacle avoidance (boundary following) would not be possible around another robot. Due to the behavior of the follower in

all_robots.launch

```
<launch>
  <!-- No namespace here as we will share this description.
       Access with slash at the beginning -->

  <!-- BEGIN ROBOT 1-->
  <group ns="target">
    <param name="tf_prefix" value="target_tf" />
    <include file="$(find moving_target_gazebo)/launch/single_robot.launch" >
      <arg name="init_pose" value="-x 5.0 -y 0.0" />
      <arg name="robot_name"  value="target" />
    </include>
    <node pkg="tf" type="static_transform_publisher" name="stat_tf"
    args="5.0 0.0 0.0 0.0 0.0 0.0 /map /target_tf/odom 10" />
  </group>

  <!-- BEGIN ROBOT 2-->
  <group ns="follower">
    <param name="tf_prefix" value="follower_tf" />
    <include file="$(find moving_target_gazebo)/launch/single_robot.launch" >
      <arg name="init_pose" value="-x -5.0 -y -5.0" />
      <arg name="robot_name"  value="follower" />
    </include>
    <node pkg="tf" type="static_transform_publisher" name="stat_tf"
    args="-5.0 -5.0 0.0 0.0 0.0 0.0 /map /follower_tf/odom 10" />
  </group>
</launch>

```

Figure 3.7: Static tf publisher in *all_robots.launch*

proximity to target as shown in Fig. 3.5 and the previous laser range finder placement issue, there arises a need to implement a hard stop condition to follower keeping in mind the relative sizes of target and follower. In the simulation code, we use odometry to calculate the separation and stop the follower when the separation is less than three times robot's chassis diameter. This problem would have not occurred if we were dealing with simulation on point robots. After follower reaches a hard stop and the separation increases due to random motion of target, the follower starts following the target again. Similar to Section Fig. 3.1 we have implemented saturation function for follower homing to achieve improved performance with lesser chattering.

Put results

Chapter 4

Boundary Following

Boundary following algorithms or control laws for have been used on robots to effectively avoid obstacles in planning problems. The devised control law can't handle concave surfaces. Combination of homing and boundary following will establish a globally convergent planner using local information. A robot enters boundary following mode when it encounters an obstacle. A simple yet elegant exit condition which uses information from homing has been developed to drive the robot out of boundary following mode and continue homing towards target. A distance metric in the steering control law can drive the robot into an orbit with constant offset around the obstacle. A comparative study on path lengths for obstacle avoidance with and without distance metric has been presented in the following section of this chapter. Other theoretical and practical issues along with implementation details have been discussed.

4.1 Boundary following with Static Obstacles

Gyroscopic control based boundary following is a promising technique as proposed in [13].The obstacle and robot engagement model and shape dynamics model developed in this paper has been borrowed to develop a simpler and effective control law which uses *sgn* function to steer the robot continuously to a heading tangential to the surface of obstacle around which it is performing boundary following on $\mathbb{R}^2$. Finite time convergence of boundary following without any distance regulating function has been proved using the same sliding mode control theorem used previously in this report. A few sections of this paper have been reproduced for completeness.

4.1.1 Theory

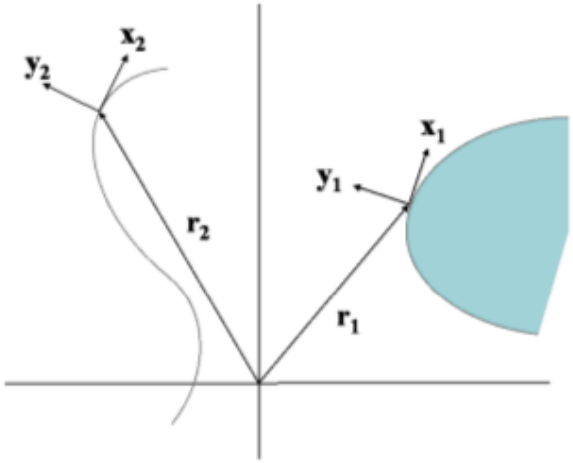


Figure 4.1: Positions and frames for the trajectory of the robot (r_2, x_2, y_2) and for the closest point on boundary curve (r_1, x_1, y_1) [13]

Boundary Following Model
In the planar setting, consider a robot moving at constant linear velocity (and subject to steering control) in the presence of a single obstacle. Suppose that at each instant of time, the point on the obstacle boundary which is closest to (i.e., the minimum Euclidean distance from) the moving vehicle is unique. This point on the obstacle boundary, which we will call the closest point (or shadow point), moves along the boundary curve. (We assume uniqueness of the closest point in order to streamline the discussion and bring out the key ideas. Of course, in dealing with real-world obstacles, non-uniqueness of the closest point is an important issue.)
Let r_1 denote the position of the closest point, let x_1 denote the unit tangent vector to the boundary curve at the closest point, and let y_1 denote the unit normal vector. Using the convention that a unit normal vector completes a right-handed orthonormal frame with the corresponding unit tangent vector. κ_1 is the plane curvature function for the boundary curve and s denoting the arc-length parameter. Because the closest point depends on the motion of the robot, $\nu_1 = \frac{ds}{dt}$ depends on both the boundary curve and on the trajectory of the robot. Letting r_2 denote the position of the robot, x_2 the unit tangent vector, y_1 the unit normal vector, and u and v denote the steering control and linear velocity for the robot respectively, following system of equations for the formation consisting of the robot and the closest point can be developed:

$$
\begin{aligned}
\dot{r_1} &= \nu_1 x_1 & \dot{r_2} &= v x_2 \\
\dot{x_1} &= \nu_1 \kappa_1 y_1 & \dot{x_2} &= u v y_2 \\
\dot{y_1} &= -\nu_1 \kappa_1 x_1 & \dot{y_1} &= -u v x_2
\end{aligned} \tag{4.1}
$$

where κ_1 may be considered given (in practice, κ_1 can be derived from sensor data, e.g., from a laser rangefinder); ν_1 is a deterministic function of (r_1, x_1, y_1) and (r_2, x_2, y_2) andκ_1; and u is the control input applied to avoid colliding with the obstacle and to achieve boundary following (see Fig. 4.1).

Shape Variables

Let us define $r = r_2 - r_1$ which is vector from the closest point on the boundary curve to the robot with the assumption that $|r| > 0$ initially. Let ϕ define the angle between the heading direction of the robot and the tangent vector to the boundary curve at the closest point such that, $x_1 \cdot y_2 = \sin\phi$ and $x_1 \cdot x_2 = \cos\phi$. Closest point and robot can be treated as two interacting agents and they converge to a steady state formation governed by shape dynamics. Shape variables (ρ, ϕ) where $\rho = |r|$ and $\phi \in (-\frac{\pi}{2}, \frac{\pi}{2})$ can be used to develop shape dynamics models (see [13] for derivation) as follows:

$$
\begin{aligned}
\dot{\rho} &= \mp v \sin\phi \\
\dot{\phi} &= v\left[\left(\frac{\kappa_1}{1 \pm |\kappa_1|\rho}\right) - u\right]
\end{aligned} \tag{4.2}
$$

where plus sign is used when the boundary curves away from robot, and the minus sign is used when the boundary curves inward toward robot. There is a singularity in the model when $\rho = \frac{1}{\kappa_1}$ and the boundary curves inward toward the robot.

Theorem 3 Steering control law u for system (4.2) is defined as

$$
u = K_{bf} sgn(\phi) \text{ where } K_{bf} > \frac{1}{\rho_{min}} \tag{4.3}
$$

Proof Let $s : \phi = 0$ be the sliding surface of shape dynamics model described in 4.2. Selecting $\sigma = \phi$ as a switching function that forces the system to evolve to the selected sliding surface, Lyapunov function candidate as shown in 3.3 gives $V = \frac{1}{2}\phi^2$. Differentiating Lyapunov candidate on both sides while

considering only positive sign (convex shapes) in 4.2,

$$\begin{aligned}\dot{V} &= \dot{\phi}\phi \\ &= v\Big[\Big(\frac{\kappa_1}{1+|\kappa_1|\rho}\Big) - u\Big]\phi \\ &= v|\phi|sgn(\phi)\Big[\Big(\frac{\kappa_1}{1+|\kappa_1|\rho}\Big) - K_{bf}sgn(\phi)\Big] \\ &= -v\sqrt{2V}\Big[K_{bf} - \Big(sgn(\phi)\frac{\kappa_1}{1+|\kappa_1|\rho}\Big)\Big]\end{aligned} \tag{4.4}$$

Using the following inequality:

$$\begin{aligned}sgn(\phi)\frac{\kappa_1}{1+|\kappa_1|\rho} &\leq \Big|\frac{\kappa_1}{1+|\kappa_1|\rho}\Big| \\ &= \frac{1}{\frac{1}{|\kappa_1|}+\rho} \\ &\leq \frac{1}{\rho_{min}}\end{aligned} \tag{4.5}$$

From 4.4, it can be showed that $\dot{V} < -\mu_{bf}\sqrt{V}$ where $\mu_{bf} = \sqrt{2}v\Big(K_{bf} - \frac{1}{\rho_{min}}\Big)$. Hence finite time convergence to the chosen sliding surface is proved.

4.1.2 Exit Conditions

For a globally convergent planner system undergoes switching between homing and boundary following. There is a need for an exit condition in the boundary following mode. These exit conditions have been designed such that there is no need of any additional information as it uses information from homing as described in Table 4.1.

Target type	**Exit Condition**
Static	Sign change of $sgn(\dot{\theta})$
Dynamic	Sign change of $sgn(\pi - (\alpha_f - \theta))$

Table 4.1: Exit condition for boundary following under two different cases

The advantage of this novel exit condition is it uses coarse sign information of bearing or its gradient instead of global information or new sensor information unlike existing planners in literature. The physical interpretation of these exit conditions boils down sign change of relative bearing which happens at the intersection of tangent from home onto the curve traced by boundary following path as illustrated in Fig. 4.2. Consider a case of static target

homing towards home (H) along with boundary following around a single obstacle. A and B are path traced by robot in boundary following mode without exit condition. In path A $\dot{\theta} > 0$, path B $\dot{\theta} < 0$ and point T is a local maxima (or local minima on tangent to obstacle on other side) of θ. The exit condition in Table 4.1 forces the robot to exit tangentially and enter homing mode. Exit condition for dynamic target homing exhibits similar behavior.

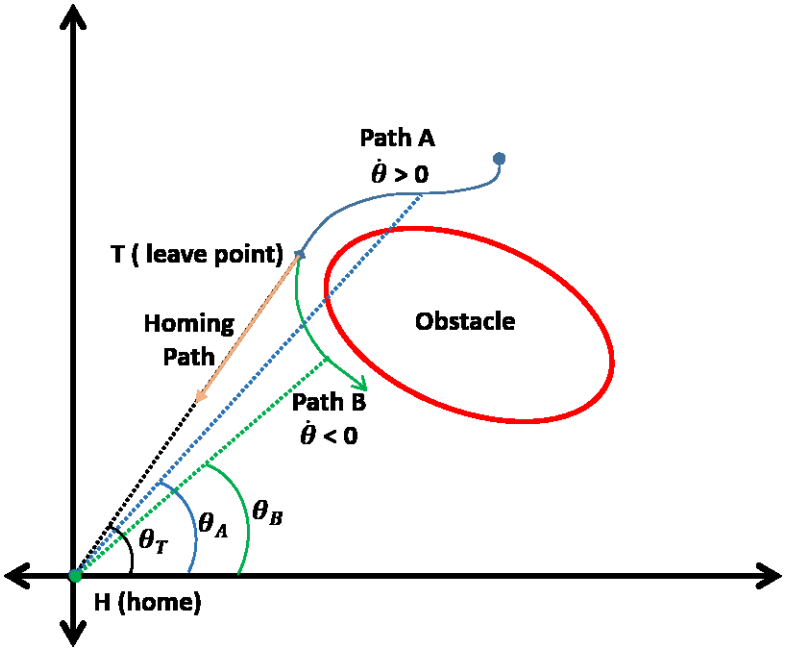


Figure 4.2: Exit condition geometry

4.1.3 Implementation

In practice estimating ϕ is not straight forward. Consider the laser rangefinder has an angular range of $(\delta_{min}, \delta_{max})$. In Xacro file containing the Hokuyo laser's parameters the ranges can be changed according to the requirement. In the current setting, resolution set to 1, number of samples is set to 720, and range is $(-\pi, \pi]$ resulting in a least count of 0.5°. The laser rangefinder is placed facing the robot's heading direction. Defining δ as an angle from robot's heading direction to line joining robot centre (neglecting laser and robot centre offset) to closest point as shown in Fig. 4.3. From the chosen angle and frame conventions with counter clockwise as positive direction for angles, the following is the relation between ϕ and δ,

$$\phi = \begin{cases} \delta + \frac{\pi}{2} & \delta \in (-\pi, 0] \\ \delta - \frac{\pi}{2} & \delta \in (0, \pi] \end{cases}$$

Using the above relation, we con modify $sgn(\phi)$ term in the steering control law 4.3 to the following:

$$u = -K_{bf} sgn(\cos\delta) sgn(\delta) \tag{4.6}$$

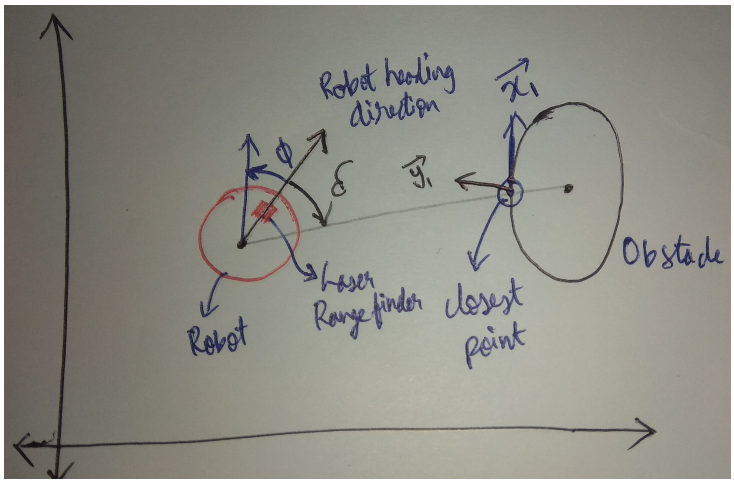


Figure 4.3: Definition of δ and ϕ

Hokuyo laser rangefinder outputs a $1D$ array called **ranges** of length equal to number of samples containing the obstacle distances at each resolution step around the sensor in a $2D$ plane. A simple $min()$ function was used to search for minimum distance or closest point. δ is estimated using a remapping function between index of the closest point in the array to angle of that point from robot heading (robot heading is always in the centre of the array or angle range). ρ is the array value at the index of the closest point in **ranges**. Hokuyo laser rangefinder gives inf values when it doesn't detect any obstacle within the LR circle around the robot.
When the rangefinder was placed at lower heights with complete angular range, laser rays reflected from wheels of the robot to throw junk values in the **ranges**. A simple solution was to increase the height of sensor from the chassis. Due to differential drive controller, the robot experienced pitching motion which resulted in laser rays getting reflected from ground and giving junk values in **ranges**. A filtering algorithm can be used on the **ranges** data to resolve this issue which has been left for future work.

A proper choice of ρ_{min} is essential for effective performance of boundary following evident from 4.3. A reasonable value for ρ_{min} would be two times the robot diameter. At the steady state, robot will try to settle in an orbit of radius $\frac{v}{K_{bf}}$ centered around obstacle irrespective of the dimension of obstacle. There is a possibility of collision if the size of obstacle is larger than this steady state orbit (equilibrium). This issue can be resolved using a distance regulating function which will change its equilibrium or sliding surface. In practice the robot takes more than 3 revolutions to reach this state. The boundary following condition can be used along with exit condition can be used to avoid reach this situation. But this exposes a imminent shortcoming in the designed controller. ROS package **satfunc_controller** contains python script **satfunc_controller_onlyBT.py** for implementation of only boundary tracking.

4.1.4 Comparative study on distance regulating function

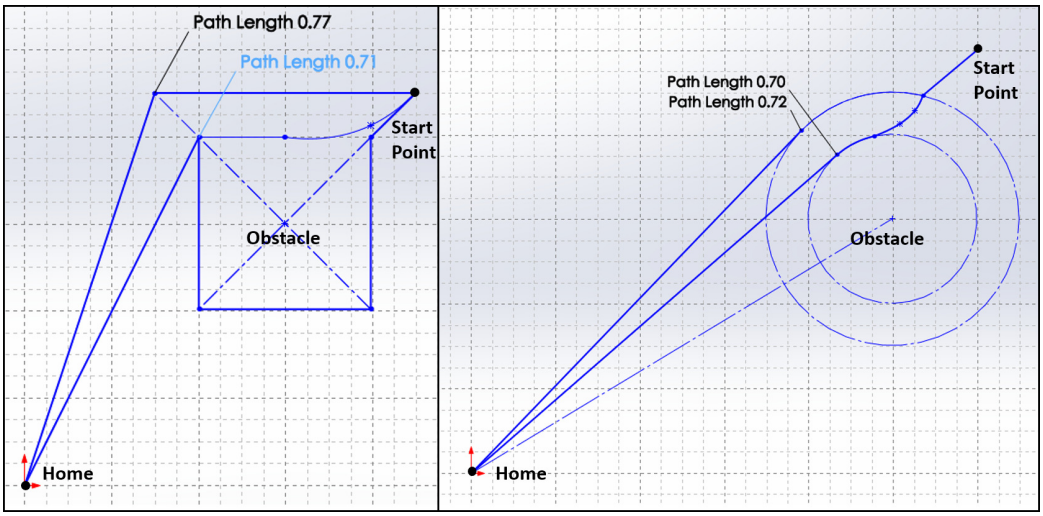


Figure 4.4: Path length comparison on SolidWorks sketch mode

A distance metric can be included in steering control law for boundary following to regulate the distance of robot from obstacle. This can be helpful in maintaining a safe distance from obstacle. The drawback of such a distance regulating function is it will tighten the motion of robot around obstacle or make it unstable, erratic or very aggressive. There is also a possibility that the robot can get stuck in boundary following mode forever if the obstacles are closer than a limit that can be determined from robot characteristics.
Consider a robot with LR: laser range in distance (typically 2 - 10 metres),

SD: safe distance between robot centre and obstacle. In terms of separation from closest point limits for ρ are established as, $\rho_{max} = LR$, $\rho_{min} = SD$, ρ_o is the desired distance/separation from boundary such that $\rho_{max} > \rho_o > \rho_{min}$. When there is no distance function involved, robot moves in level sets of decreasing radius centred around the geometric centre of obstacle until it reaches its equilibrium orbit with radius of $\frac{v}{K_{bf}}$. It's path depends on the control gain K_{bf}, entry angle of robot within the laser range and linear velocity. In contrast, robot settles in an orbit of desired distance ρ_o under the influence of a distance regulating function the robot irrespective of other parameters. Illustrated by 4.2 path length using distance regulating function is shorter than the case when there is no distance regulating function generally (The robot's transient turning dynamics has been neglected). It's difficult to prove this theoretically or geometrically due to presence of several variables. Based on the above discussions, it is clear that distance regulating function is necessary. A good choice of steering controller with distance regulating function can be:

$$u = -K_{bf} sgn(\cos\delta) sgn(\delta) + K_\rho \Big[1 - \Big(\frac{\rho_o}{\rho}\Big)^2\Big] \tag{4.7}$$

The control gains K_{bf} and K_{rho} can be tuned to adjust the aggressiveness of distance regulation and boundary following control independently. The steering controller in 4.7 has singularity at $\rho = 0$. In practice, the controller will never allow the robot to enter into a situation where ρ is very small given that controller gains are well tuned. Theoretical proof of steering controller in 4.7 is pending but it is found to work well in developed simulations. Clipping or saturation of steering control value has been implemented in simulation setup to avoid aggressive or unstable behavior of robot in simulated environment and replicating practical conditions.

Put results

Chapter 5

Combination of homing and boundary following

The proper functioning of exit condition is a challenge due to hysteresis in exit condition signal and chattering of robot dynamics.
Clipping the steering values to avoid aggressive or unstable behavior.

Chapter 6

Future Work

Boundary following for dynamic obstacles. Combination of dynamic target homing and dynamic boundary following.
Boundary following and homing in R^3 for all the combination of static and dynamic target or obstacles.

Chapter 7

Conclusion

Acknowledgment

References

[1] G. Bianco, R. Cassinis, A. Rizzi, N. Adami and P. Mosna, "A bee-inspired robot visual homing method," Proceedings Second EUROMICRO Workshop on Advanced Mobile Robots, Brescia, Italy, 1997, pp. 141-146. doi: 10.1109/EURBOT.1997.633620

[2] M. Liu, C. Pradalier and R. Siegwart, "Visual Homing From Scale With an Uncalibrated Omnidirectional Camera," in IEEE Transactions on Robotics, vol. 29, no. 6, pp. 1353-1365, Dec. 2013. doi: 10.1109/TRO.2013.2272251

[3] A. A. Argyros, K. E. Bekris and S. C. Orphanoudakis, "Robot homing based on corner tracking in a sequence of panoramic images," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. II-II. doi: 10.1109/CVPR.2001.990917

[4] T. Goedeme, T. Tuytelaars, L. Van Gool, D. Vanhooydonck, E. Demeester and M. Nuttin, "Is structure needed for omnidirectional visual homing?," 2005 International Symposium on Computational Intelligence in Robotics and Automation, Espoo, 2005, pp. 303-308. doi: 10.1109/CIRA.2005.1554294

[5] A. Sabnis, A. G. K., V. Dwaracherla and L. Vachhani, "Probabilistic Approach for Visual Homing of a Mobile Robot in the Presence of Dynamic Obstacles," in IEEE Transactions on Industrial Electronics, vol. 63, no. 9, pp. 5523-5533, Sept. 2016. doi: 10.1109/TIE.2016.2569496

[6] J . Vaganay, P. Baccou and B. Jouvencel, "Homing by acoustic ranging to a single beacon," OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No.00CH37158), Providence, RI, USA, 2000, pp. 1457-1462 vol.2. doi: 10.1109/OCEANS.2000.881809

[7] P. Baccou and B. Jouvencel, "Homing and navigation using one transponder for AUV, postprocessing comparisons results with long

base-line navigation," Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), Washington, DC, USA, 2002, pp. 4004-4009 vol.4. doi: 10.1109/ROBOT.2002.1014361

[8] G. Vallicrosa, P. Ridao, D. Ribas and A. Palomer, "Active Range-Only beacon localization for AUV homing," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, 2014, pp. 2286-2291. doi: 10.1109/IROS.2014.6942871

[9] A. G. K., A. Sabnis,L. Vachhani, "Robust Steering Control for Autonomous Homing and its Application in Visual Homing under Practical Conditions,", J. of Intel. and Robot. Syst., vol. 89, no. 3, pp. 403-419, March 2008. doi: 10.1007/s10846-017-0561-2

[10] S. Charifa and M. Bikdash, "Adaptive boundary-following algorithm guided by artificial potential field for robot navigation," 2009 IEEE Workshop on Robotic Intelligence in Informationally Structured Space, Nashville, TN, 2009, pp. 38-45. doi: 10.1109/RIISS.2009.4937904

[11] S. S. Ge, Xuecheng Lai and A. A. Mamun, "Boundary following and globally convergent path planning using instant goals," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 35, no. 2, pp. 240-254, April 2005. doi: 10.1109/TSMCB.2004.842368

[12] S. S. Ge, Xuecheng Lai and A. A. Mamun, "Boundary following and globally convergent path planning using instant goals," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 35, no. 2, pp. 240-254, April 2005. doi: 10.1109/TSMCB.2004.842368

[13] F. Zhang, E. W. Justh and P. S. Krishnaprasad, "Boundary following using gyroscopic control," 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601), Nassau, 2004, pp. 5204-5209 Vol.5. doi: 10.1109/CDC.2004.1429634

[14] Fumin Zhang and Salman Haq, "Boundary following by robot formations without GPS," 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, 2008, pp. 152-157. doi: 10.1109/ROBOT.2008.4543201

[15] Lentin Joseph, "Mastering ROS for Robotics programming," Packt Publishing Ltd., December 2015, Available at: http://it-ebooks.info/

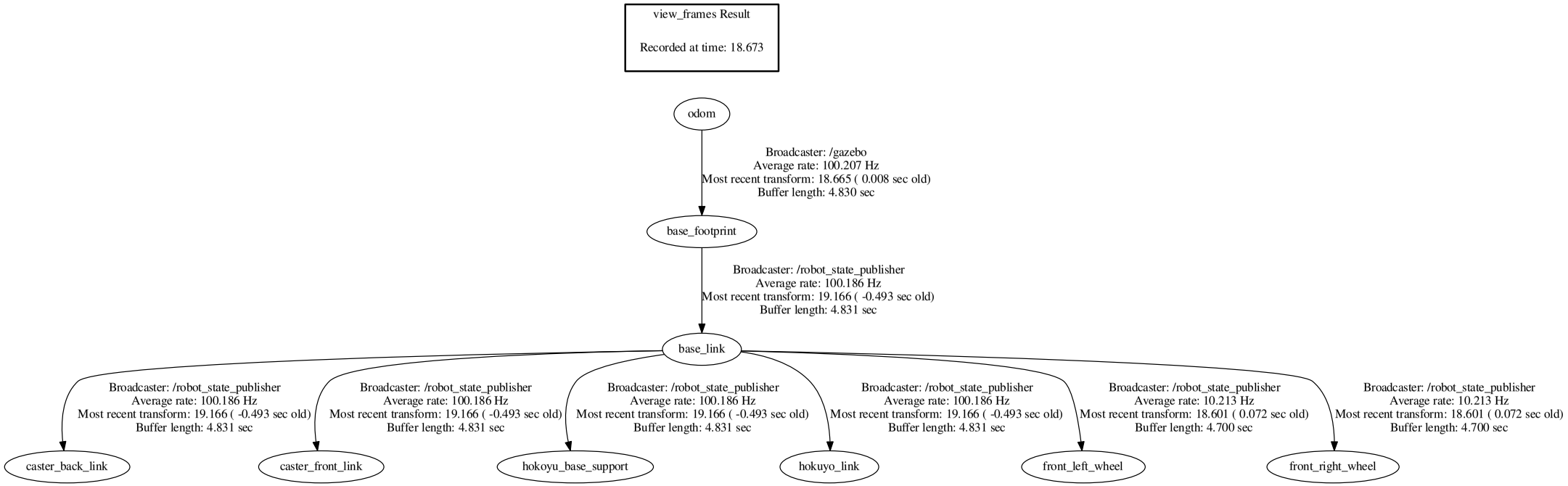
https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Second-Edition.git

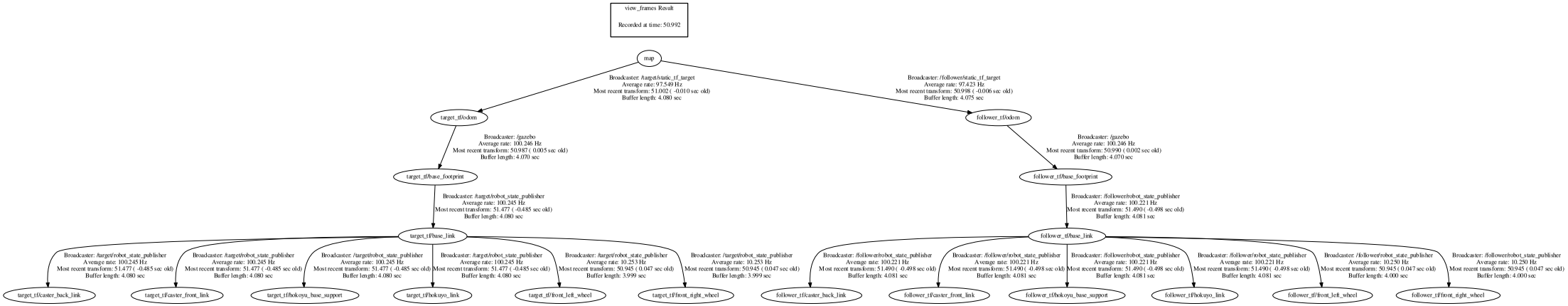
[16] Perruquetti W., Barbot, J.P.,"Sliding Mode Control in Engineering", Marcel Dekker Hardcover, 2002. ISBN 978-0-8247-0671-5

[17] J. Y. Hung, R. M. Nelms and P. B. Stevenson, "An output feedback sliding mode speed regulator for DC drives," in IEEE Transactions on Industry Applications, vol. 30, no. 3, pp. 691-698, May-June 1994. doi: 10.1109/28.293718

[18] A. Saghafinia, H. W. Ping, M. N. Uddin and K. S. Gaeid, "Adaptive Fuzzy Sliding-Mode Control Into Chattering-Free IM Drive," in IEEE Transactions on Industry Applications, vol. 51, no. 1, pp. 692-701, Jan.-Feb. 2015. doi: 10.1109/TIA.2014.2328711

[19] Yong Feng, Xinghuo Yu and Zhihong Man, "Non-singular terminal sliding mode control and its application for robot manipulators," ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196), Sydney, NSW, 2001, pp. 545-548 vol. 2. doi: 10.1109/ISCAS.2001.921368


view_frames Result
Recorded at time: 18.673
odom
Broadcaster: /gazebo
Average rate: 100.207 Hz
Most recent transform: 18.665 (0.008 sec old)
Buffer length: 4.830 sec
base_footprint
Broadcaster: /robot_state_publisher
Average rate: 100.186 Hz
Most recent transform: 19.166 (-0.493 sec old)
Buffer length: 4.831 sec
base_link
Broadcaster: /robot_state_publisher
Average rate: 100.186 Hz
Most recent transform: 19.166 (-0.493 sec old)
Buffer length: 4.831 sec
caster_back_link
Broadcaster: /robot_state_publisher
Average rate: 100.186 Hz
Most recent transform: 19.166 (-0.493 sec old)
Buffer length: 4.831 sec
caster_front_link
Broadcaster: /robot_state_publisher
Average rate: 100.186 Hz
Most recent transform: 19.166 (-0.493 sec old)
Buffer length: 4.831 sec
hokoyu_base_support
Broadcaster: /robot_state_publisher
Average rate: 100.186 Hz
Most recent transform: 19.166 (-0.493 sec old)
Buffer length: 4.831 sec
hokuyo_link
Broadcaster: /robot_state_publisher
Average rate: 10.213 Hz
Most recent transform: 18.601 (0.072 sec old)
Buffer length: 4.700 sec
front_left_wheel
Broadcaster: /robot_state_publisher
Average rate: 10.213 Hz
Most recent transform: 18.601 (0.072 sec old)
Buffer length: 4.700 sec
front_right_wheel


view_frames Result
Recorded at time: 50.992
map
Broadcaster: /target/static_tf_target
Average rate: 97.5.40 Hz
Most recent transform: 51.002 (-0.010 sec old)
Buffer length: 4.080 sec
Broadcaster: /follower/static_tf_target
Average rate: 97.4.23 Hz
Most recent transform: 50.998 (-0.006 sec old)
Buffer length: 4.075 sec
target_tf/odom
follower_tf/odom
Broadcaster: /gazebo
Average rate: 100.246 Hz
Most recent transform: 50.987 (0.005 sec old)
Buffer length: 4.070 sec
Broadcaster: /gazebo
Average rate: 100.246 Hz
Most recent transform: 50.990 (0.002 sec old)
Buffer length: 4.070 sec
target_tf/base_footprint
follower_tf/base_footprint
Broadcaster: /target/robot_state_publisher
Average rate: 100.245 Hz
Most recent transform: 51.477 (-0.485 sec old)
Buffer length: 4.080 sec
Broadcaster: /follower/robot_state_publisher
Average rate: 100.221 Hz
Most recent transform: 51.490 (-0.498 sec old)
Buffer length: 4.081 sec
target_tf/base_link
follower_tf/base_link
Broadcaster: /target/robot_state_publisher
Average rate: 100.245 Hz
Most recent transform: 51.477 (-0.485 sec old)
Buffer length: 4.080 sec
Broadcaster: /target/robot_state_publisher
Average rate: 100.245 Hz
Most recent transform: 51.477 (-0.485 sec old)
Buffer length: 4.080 sec
Broadcaster: /target/robot_state_publisher
Average rate: 100.245 Hz
Most recent transform: 51.477 (-0.485 sec old)
Buffer length: 4.080 sec
Broadcaster: /target/robot_state_publisher
Average rate: 100.245 Hz
Most recent transform: 51.477 (-0.485 sec old)
Buffer length: 4.080 sec
Broadcaster: /target/robot_state_publisher
Average rate: 10.253 Hz
Most recent transform: 50.945 (0.047 sec old)
Buffer length: 3.999 sec
Broadcaster: /target/robot_state_publisher
Average rate: 10.253 Hz
Most recent transform: 50.945 (0.047 sec old)
Buffer length: 3.999 sec
Broadcaster: /follower/robot_state_publisher
Average rate: 100.221 Hz
Most recent transform: 51.490 (-0.498 sec old)
Buffer length: 4.081 sec
Broadcaster: /follower/robot_state_publisher
Average rate: 100.221 Hz
Most recent transform: 51.490 (-0.498 sec old)
Buffer length: 4.081 sec
Broadcaster: /follower/robot_state_publisher
Average rate: 100.221 Hz
Most recent transform: 51.490 (-0.498 sec old)
Buffer length: 4.081 sec
Broadcaster: /follower/robot_state_publisher
Average rate: 100.221 Hz
Most recent transform: 51.490 (-0.498 sec old)
Buffer length: 4.081 sec
Broadcaster: /follower/robot_state_publisher
Average rate: 10.250 Hz
Most recent transform: 50.945 (0.047 sec old)
Buffer length: 4.000 sec
Broadcaster: /follower/robot_state_publisher
Average rate: 10.250 Hz
Most recent transform: 50.945 (0.047 sec old)
Buffer length: 4.000 sec
target_tf/caster_back_link
target_tf/caster_front_link
target_tf/hokuyu_base_support
target_tf/hokuyo_link
target_tf/front_left_wheel
target_tf/front_right_wheel
follower_tf/caster_back_link
follower_tf/caster_front_link
follower_tf/hokuyu_base_support
follower_tf/hokuyo_link
follower_tf/front_left_wheel
follower_tf/front_right_wheel