Pursuit and Evasion

Jordan Lane and Brad Ofrim

1. PURPOSE

Pursuit and Evasion is a competition consisting of two Turtlebots, a pursuer, and an evader. The pursuer bot is tasked with chasing the evader bot for as long as possible. The evader bot is tasked with attempting to lose the chaser bot. Each turtlebot plays both roles, and a round ends when a pursuer loses the evader, the two bots crash, or the evader follows the pursuer for the full round.

This report aims to explain the logic behind both implementations, and how to run both the evader (robber) and pursuer (cop) on the Turtlebot 2.

2. PRE-REQUISITES

This project is built using python 2.7, ros-kinetic, ros-kinetic kobuki, and turtlebot libraries for Ubuntu Xenial 16.04. If these are not installed, please refer to their installation pages on the official ROS wiki or the official python installation page.

- http://wiki.ros.org/kobuki/Tutorials/Installation/kinetic
- http://wiki.ros.org/kinetic/Installation
- https://www.python.org/downloads/

The source code for our project can be found at https://github.com/bofrim/CMPUT_412

Create or navigate to an existing catkin workspace and clone our repository. Our repository consists of multiple catkin packages so you will need to copy the files (or just the packages you want; in this case, “comp1”) from our repository into your catkin workspace.

3. EXECUTION

Open a console and source the catkin workspace that the package was placed. For example, if your workspace was called catkin_ws:

```
$ cd catkin_ws/
$ source ./devel/setup.bash
```

Next, build the project:

```
$ catkin_make
```

3.1 Simulated Robber

To start the robber simulation, run the simulated robber launch file

```
$ cd catkin_ws/
$ roslaunch comp1 simulated_rob.launch
```

This will launch the robber in the default gazebo world.

3.2 Physical Robber

Start by turning on the Turtlebot and connecting it to your computer.

To start the physical robber, run the simulated robber launch file.

```
$ cd catkin_ws/
$ roslaunch comp1 physical_rob.launch
```

Launching this file will start the robber, and it will immediately begin running it's evasion algorithm

3.3 Simulated Cop

To start the robber simulation, open two consoles

In the first console, run the keyboard control node. This will act as our controller for the Simulated Cop.

```
$ cd catkin_ws
$ rosrun comp1 keys_to_joy.py
```

In the second console launch the simulated cop.

```
$ cd catkin_ws/
$ roslaunch comp1 simulated_cop.launch
```

This will launch the cop in the default gazebo world. To control the cop, switch to the first terminal, and press the corresponding key:

Key	Function
X	Start the following mode
A	Start the tracking mode
B	Start idle mode (default)

3.4 Physical Cop

Start by turning on the Turtlebot and connecting it to your computer.

To start the physical cop, run the physical robber launch file.

```
$ cd catkin_ws/
$ roslaunch comp1 physical_cop.launch
```

Turn on a compatible Joy controller and control the robot using the same buttons used to control the cop in the simulated environment.

4. CONCEPTS AND CODE

4.1 Cop - Software Tracking Gimbal

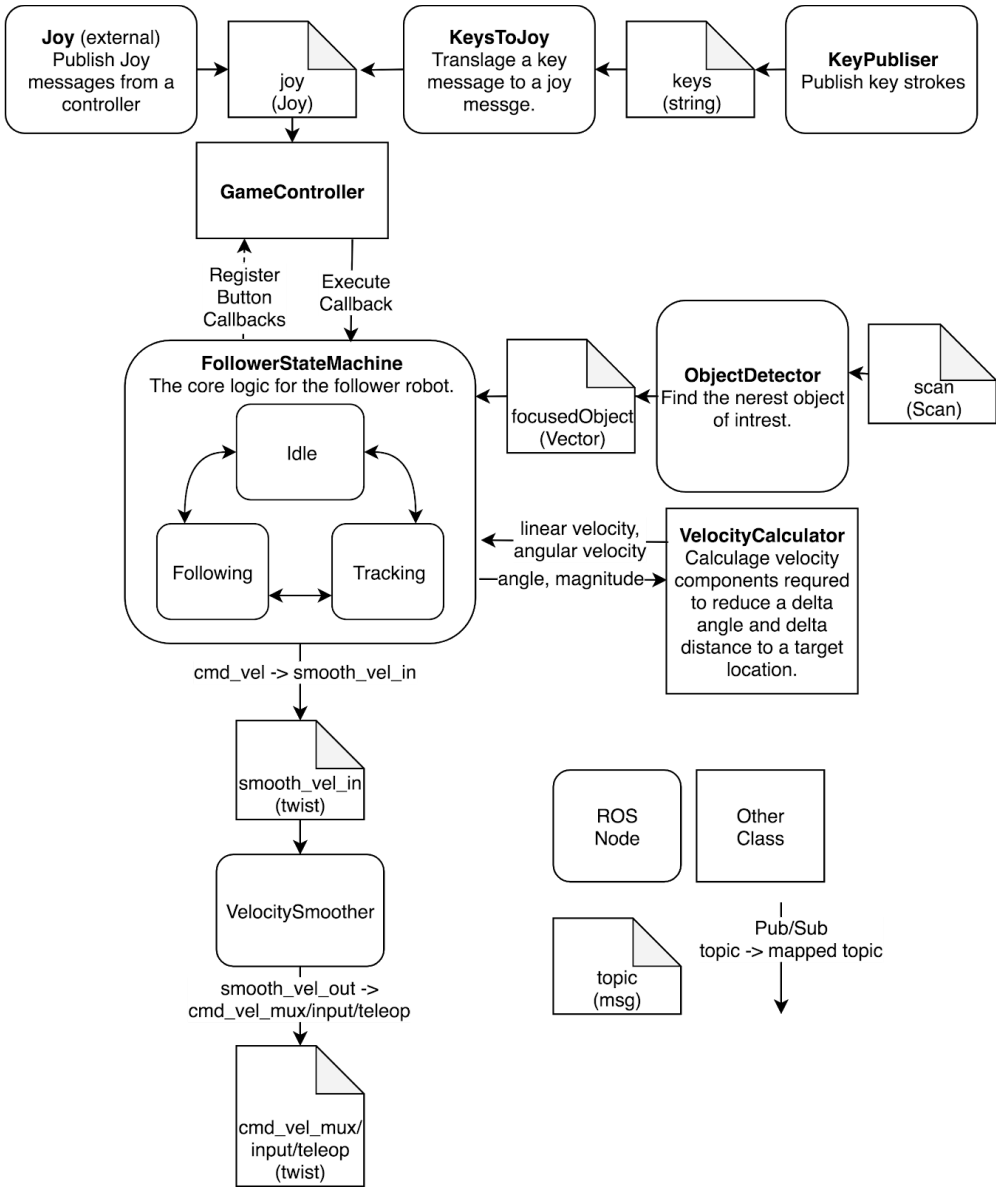

Figure: Cop System design

The pursuer system consists of a state machine, an incoming data path, and an outgoing data path. Data received from the laser scan is processed by an “ObjectDetector” node which attempts to find and track the nearest object of interest. The object detector uses a “software gimbal” concept to continuously track the robber within a narrow range of scan data. As the robber moves across the cop’s field of view, the cop keeps track of what angle it is at relative to

the robot's base. The object detector creates a buffer of plus/minus 10 degrees around this angle, which is the total range that will be used for the next scan. The object detector then repeats the cycle by determining the closest object within this range and updating the buffer. This allows the robot to track objects within the narrow band and ignore all distractions such as walls and other objects.

The "FollowerStateMachine" is where the main control of the robber takes place. Angle and distance data from the "ObjectDetector" is combined with data from the game controller and "VelocityCalculator" utility class to control the motion of the robot. The "VelocityCalculator" class calculates an appropriate velocity in order to minimize the relative angle between the robot and the object of interest, as well as to keep the robot a specific distance from that object. Velocity data is smoothed and passed to the robot base.

4.2 Robber - Zones of Interest

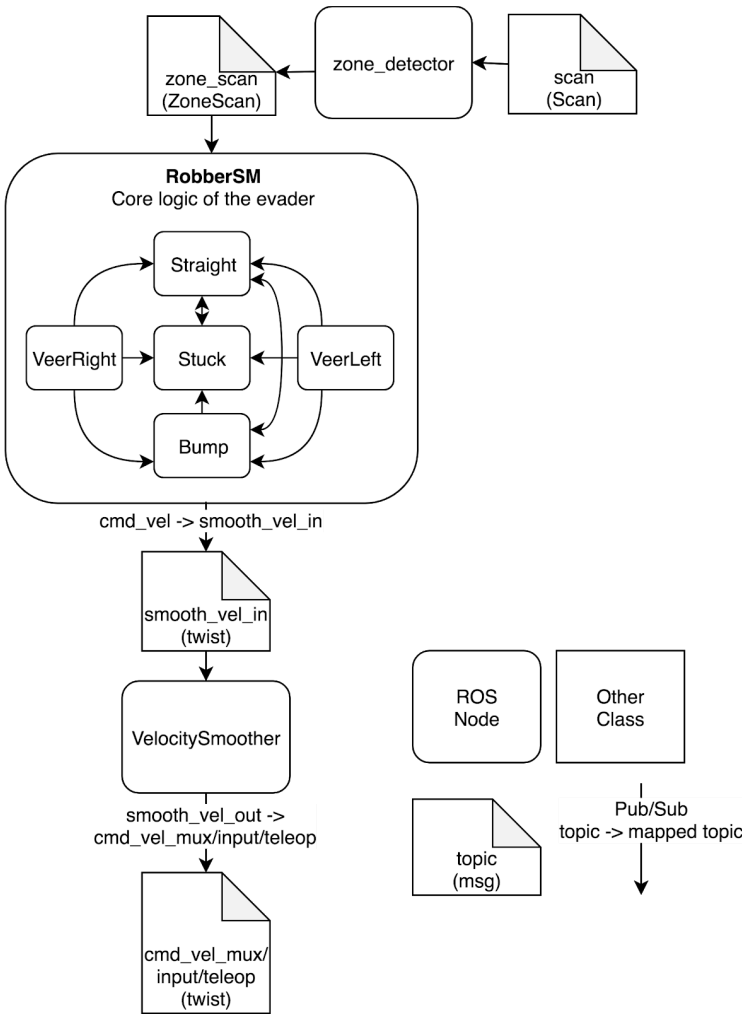

Figure: Robber System Design

The robber robot consists of a state machine with one input data stream and one output data stream. The input scan data from the laser scanner is processed by a “ZoneScanner” which partitions the scan into discrete zones and computes the distance to the nearest object in each zone. This zone data is passed to the robber state machine which determines an appropriate state based on the distance to the nearest objects in each zone. Each state will then output a velocity command which represents the operation of that state (i.e. move forward, veer slightly, stop and turn). Velocity data is smoothed and passed to the robot base.