The Swarm Robotic Game

Cuddalore Parthasarathy Sridhar
Instructor: Dr. Vikram Kapila

MS Project May 2018
Outline

- Introduction
- Hardware
- Software
- Architecture
- Blob Detection
- Initialization, Localization and Heading estimate
- Tracking
- Control
- Swarming
- 1 Vs 1 escape
- Public Demonstration
Introduction

- We are building an interactive swarm robotic game where autonomous robots play with humans

- Cameras track robots

- LiDARS track humans
Gameplay

• After three public demonstrations and feedbacks two gameplay were finalized.
  
• Kids - Human chases robots and robots move in formation around the human
  
• Adults - Robots try to bump on the human foot while the human tries to dodge them
Hardware

• Motion of omni-directional robots are hard to predict and makes them a perfect candidate for the game.

• Some of the considered configurations:
With a higher priority to building the system over building the robots, Sphere Sprk+ is selected as the robot platform for this project.

<table>
<thead>
<tr>
<th></th>
<th>Cost</th>
<th>Build/Purchase</th>
<th>Time to build</th>
<th>Symmetry</th>
<th>Customizable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangle</td>
<td>$700</td>
<td>Build</td>
<td>2 months</td>
<td>Cuboid 2-axis symmetry</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>$1500</td>
<td>Purchase</td>
<td>10 days</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triangle</td>
<td>$500</td>
<td>Build</td>
<td>2 months</td>
<td>Delta 2-axis symmetry</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>$1400</td>
<td>Purchase</td>
<td>10 days</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphero Sprk+</td>
<td>$100</td>
<td>Purchase</td>
<td>1 day</td>
<td>Sphere 3-axis symmetry</td>
<td>No</td>
</tr>
</tbody>
</table>
Experimental Setup

Logitech webcam is mounted on the ceiling facing down perpendicular to the black floor

Camera feed
We use ROS (Robot Operating System) framework.

Our key utility in ROS is the publishing/subscribing plumbing as this eliminated the need to write multithreading in python and handles all the backend in C which is 5 times faster.

Language: Python 2.7 and 3.14

Image Processing: OpenCV in Python2.7
Architecture
Blob Detection

- Blob Detection is done using OpenCV
- Maps output from pixels to global frame of reference in 100x100 units
- Output type:
  - Polygon message type
  - `geometry_msgs/Point32[]`
Blob Detection

- Problem:
  - Robots cannot get closer
Blob Detection

- Fix:
  - Increase erode kernel size
  - Polaroid Filter on camera
Initialization - Localization

• As all robots look the same an initialization sequence was built to create blob-IP pair

• Sequence:
  • Turn off LEDs in all robots
  • Send command to flash LED on each IP
  • Record blob location on each flash
  • Turn on LEDs on all robots
Initialization - Heading

• Sphere considers the heading it faces on power up as its zero reference heading.

• Once the initial heading is captured, we can offset all our controls from the initial heading to map the robot orientation to our global frame.

• Sequence:
  • Check if initial localization is complete
  • Move all robots forward
  • Use minimum cartesian distance between robot and all blobs to estimate new robot position
  • Once robot has moved 5 units from the initial position, calculate arctan2(dy/dx) to capture robot initial heading
Tracking

Kalman Filter

\[
\begin{bmatrix}
X_{\text{new}} \\
Y_{\text{new}} \\
x_{\text{velocity new}} \\
y_{\text{velocity new}}
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & dt & 0 \\
0 & 1 & 0 & dt \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
x_{\text{velocity}} \\
y_{\text{velocity}}
\end{bmatrix}
\]
Distance based tracking
KF based tracking

Robot(t-1)

Blobs

Best possible location at time t

Robot position estimate(t)
• PD control is preferred for similar control situation.

• The reason a PID controller is used is because we might have uncertainty in our initial heading estimate resulting in a constant drift. The Integral term accounts for this drift.
PID Control

- Variables: Steering angle, Velocity
  - Steering angle is decided based on angle error
    - Angle error = $\Delta$(current heading - heading to goal)
  - Linear velocity is decided based on distance error
    - Distance error = $\Delta$(position_xy - goal_xy)
Swarming

- Each formation is written as a set of constrains.

```
corner_centre = [[20.0,20.0],[80.0,80.0],[20.0,80.0],[80.0,20.0],[50.0,50.0]]
line_horizontal = [[10.0,50.0],[30.0,50.0],[50.0,50.0],[70.0,50.0],[90.0,50.0]]
```

```
formations = {
    "corner_centre" : corner_centre,
    "line_horizontal" : line_horizontal,
}
```

- Cost matrix is calculated using scipy.distance.cdist between every robot location and all goals.

- Least cost goal is assigned to each robot by using scipy.linear_sum_assignment and is published to the robot nodes.
Swarming

• Swarming node also is programmed with the following functions:

  1. Hold a list of formations to cycle through
  2. Check of formation consensus is achieved
  3. Auto switch formations
  4. Scale formations
  5. Rotate formation
  6. Move formation about any point
  7. Camera angle compensation
  8. Take user input in the command terminal to switch to any formation
Swarming

• Moving formations
  • Detect human position and convert that to a translation matrix.
  • Add the translation matrix to the goal.
• Rotate formations
  • Multiply goals by rotation matrix
• Move and Rotate formations
  • Multiply goals by a homogeneous transformation matrix
Swarming

Formation of alphabet A with 5 robots
1 vs 1 escape

• An occupancy grid is created for the floor area.

• The human player is simulated by another Sphero that is manually driven with a joystick.

• Few changes made to the sphero script recognizes the human substitute robot and tracks it in realtime.
1 vs 1 escape

• Gaussian
  • Apply a Multivariate Gaussian around the human substitute with a radius of 1f and also along the walls.
  • The robot picks us the velocity applied of its grid location on the matrix and moves in the direction with minimum cost.

• Kalman Filter + Broder High Weights + Quadrant Escape
  • KF estimate the quadrant to which the human substitute is heading
  • Robots randomly chooses a safe quadrant and moves towards it
  • Corners and edges are given first and second highest priority
Public Demonstration

- USA Science and Engineering Festival | Non technical audience, mostly kids
- Centre for Advanced Telecommunication Technologies | Technical audience
- NYU Tandon School of Engineering - Research Expo | Mixed audience

Some pictures from the first public demo
VIDEOS

https://youtu.be/9qRJkepvQ-c

Scripts, Report and All videos are uploaded in the drive