NYU TANDON SCHOOL OF ENGINEERING

The Swarm Robotic Game

Cuddalore Parthasarathy Sridhar
Instructor: Dr. Vikram Kapila

MS Project May 2018

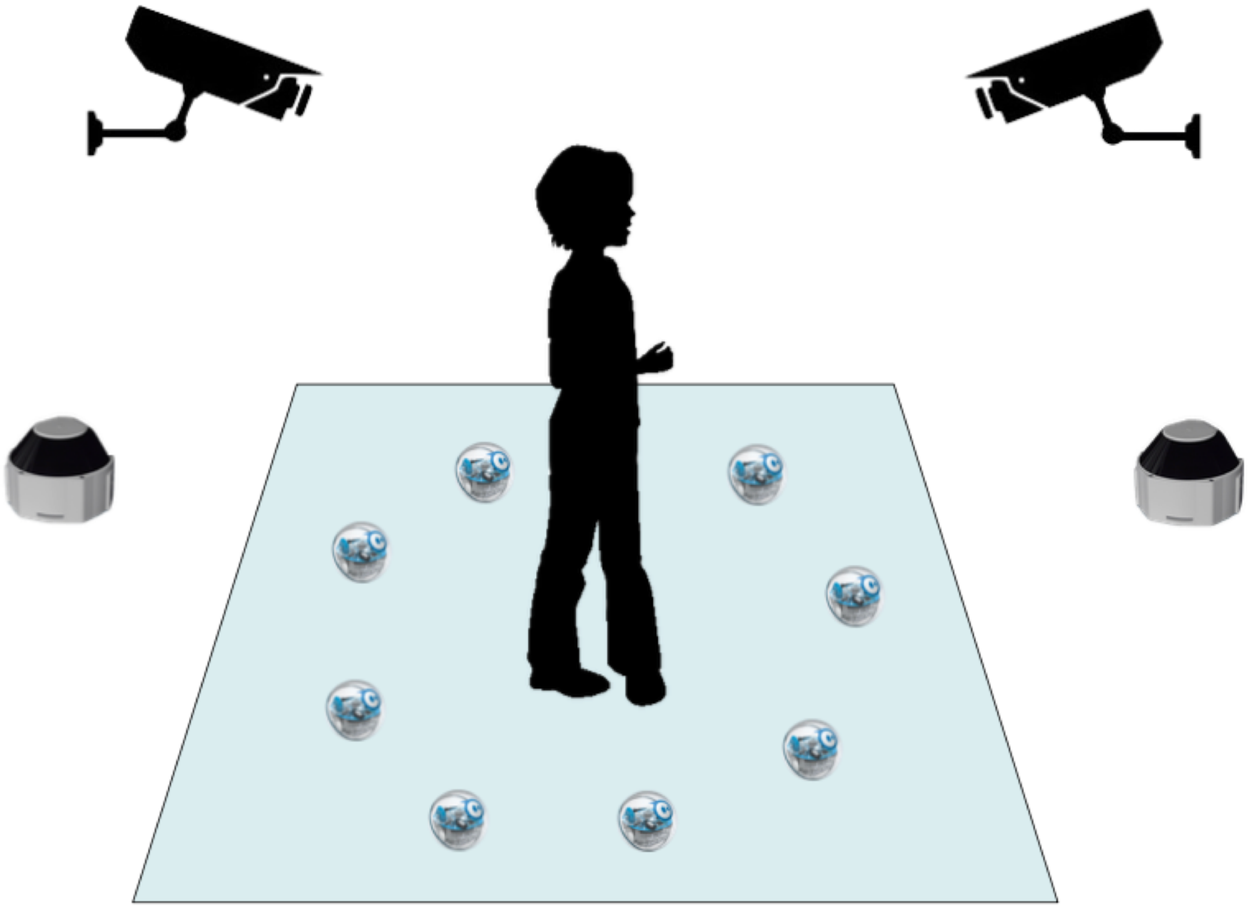
NYU TANDON SCHOOL OF ENGINEERING

Outline

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

NYU TANDON SCHOOL OF ENGINEERING

Introduction



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

NYU TANDON SCHOOL OF ENGINEERING

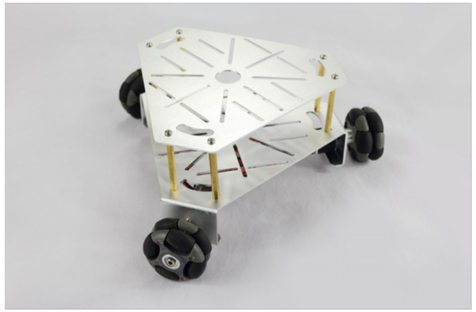
Gameplay

- 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

NYU TANDON SCHOOL OF ENGINEERING

Hardware

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



NYU TANDON SCHOOL OF ENGINEERING

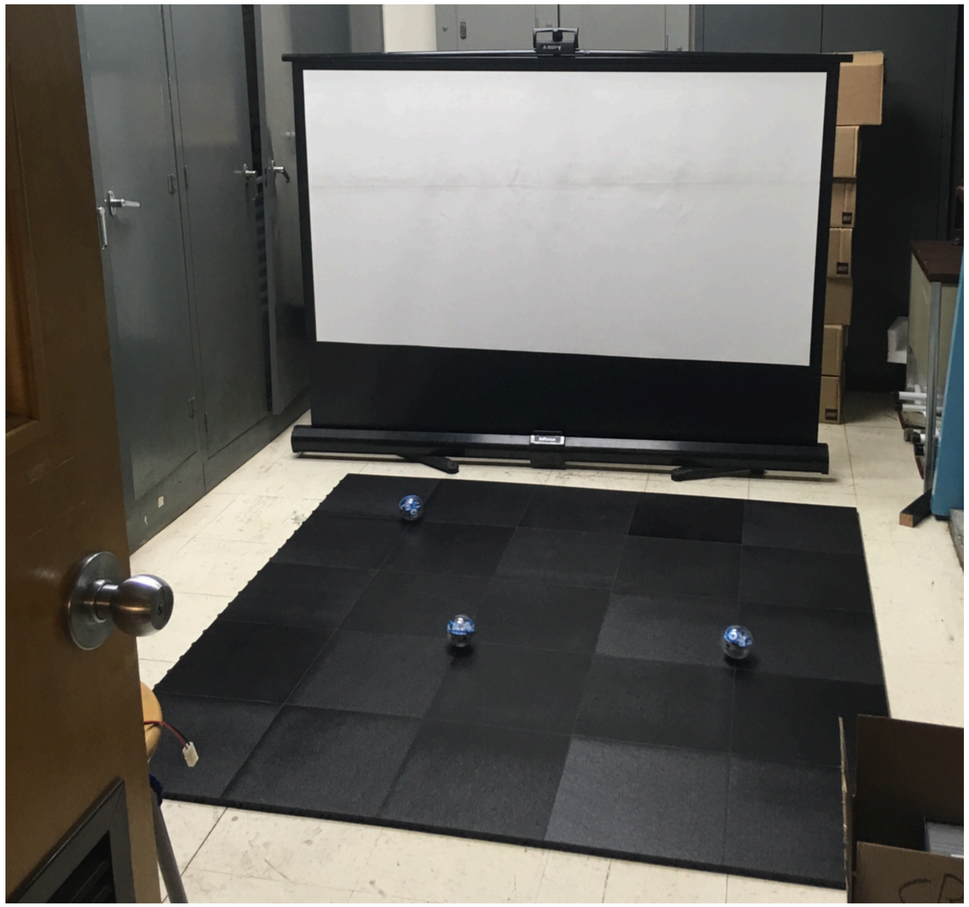
Hardware

	Cost	Build/Purchase	Time to build	Symmetry	Customizable
Rectangle	$700	Build	2 months	Cuboid 2-axis symmetry	Yes
	$1500	Purchase	10 days		
Triangle	$500	Build	2 months	Delta 2-axis symmetry	Yes
	$1400	Purchase	10 days		
Sphero Sprk+	$100	Purchase	1 day	Sphere 3-axis symmetry	No

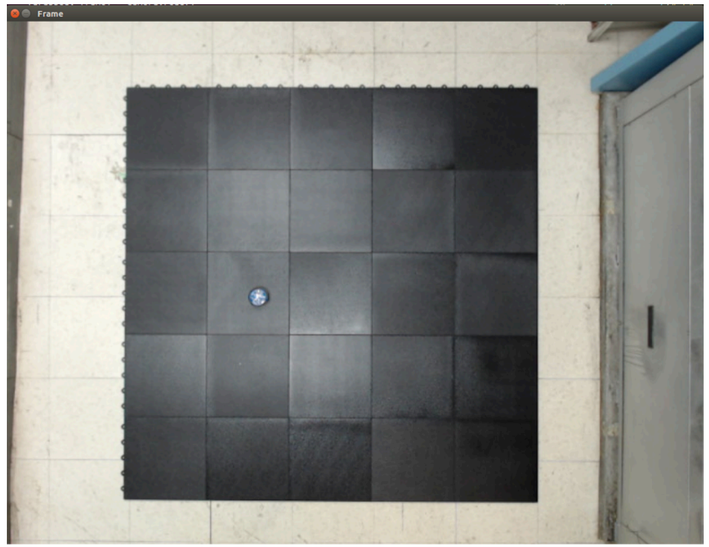
With a higher priority to building the system over building the robots Sphere Sprk+ is selected as the robot platform for this project

NYU TANDON SCHOOL OF ENGINEERING

Experimental Setup



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



Camera feed

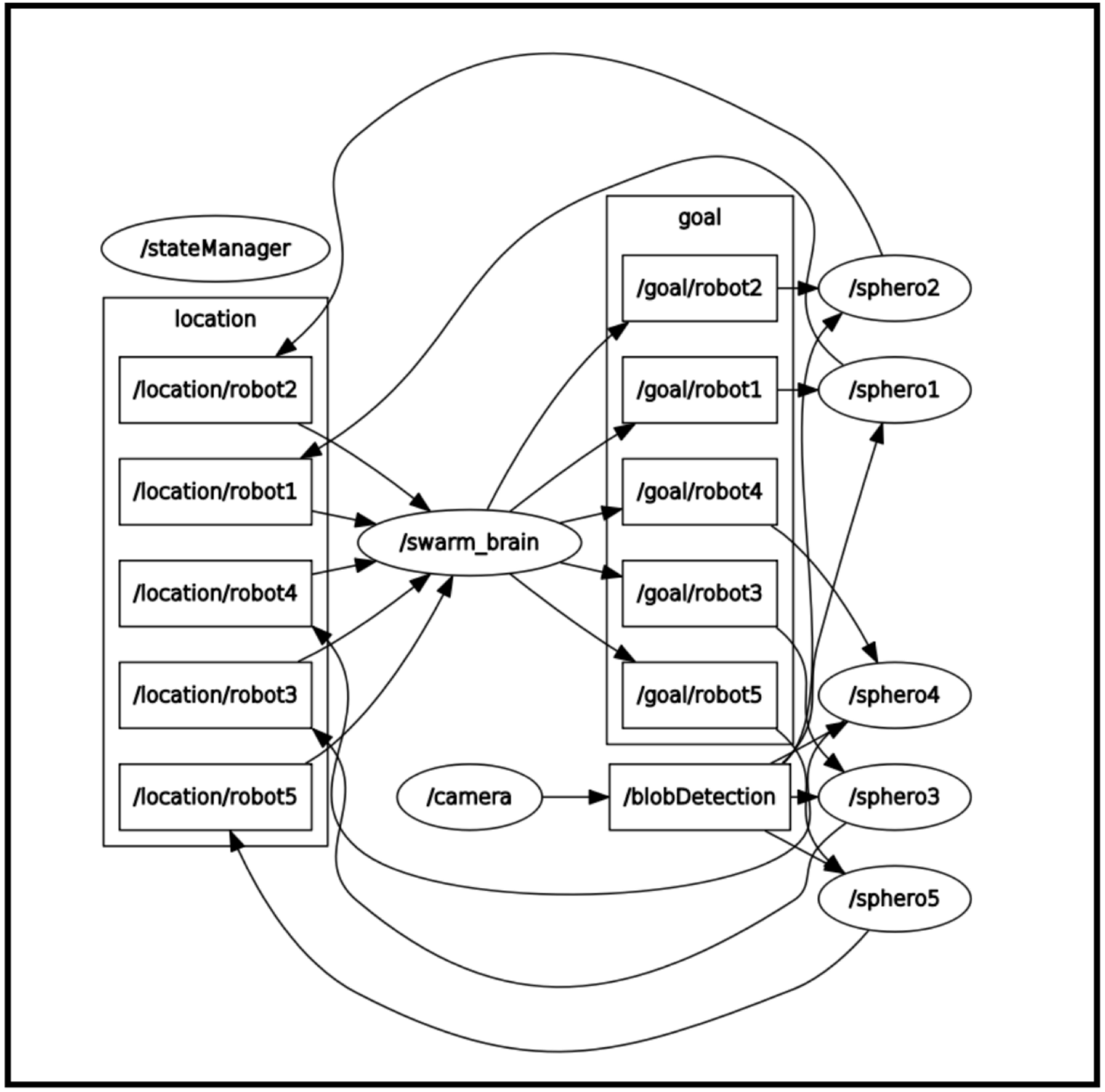
NYU TANDON SCHOOL OF ENGINEERING

Software

- 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

NYU TANDON SCHOOL OF ENGINEERING

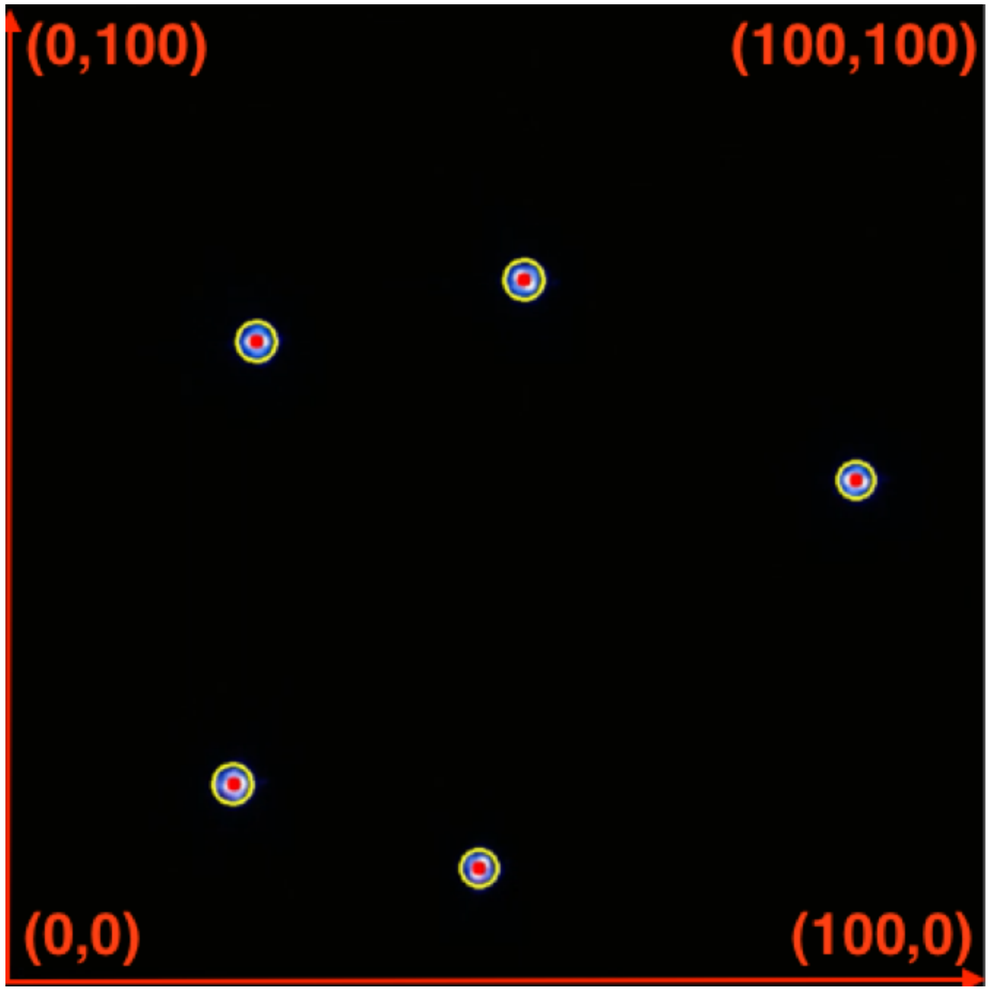
Architecture



NYU TANDON SCHOOL OF ENGINEERING

Blob Detection

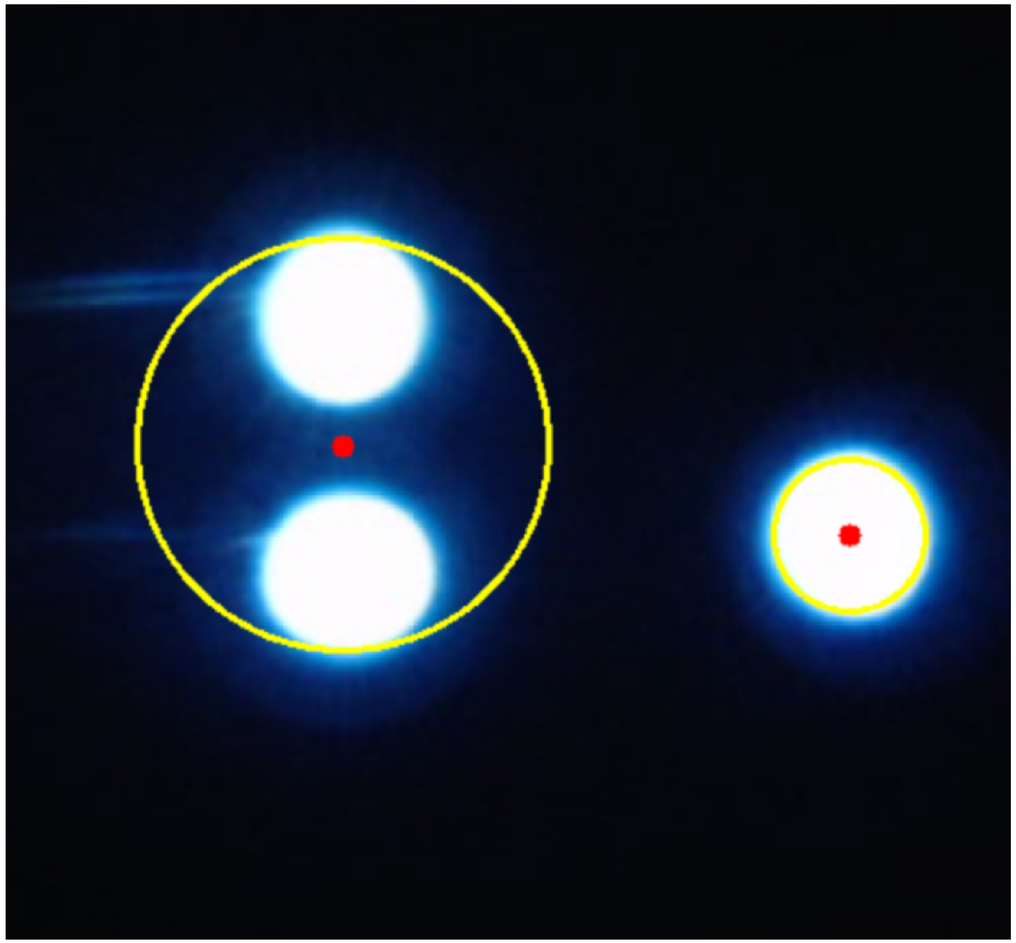
- Blob Detection is done using OpenCV
- Maps output form pixels to global frame of reference in 100x100 units
- Output type :
  - Polygon message type
  - geometry_msgs/Point32[]



NYU TANDON SCHOOL OF ENGINEERING

Blob Detection

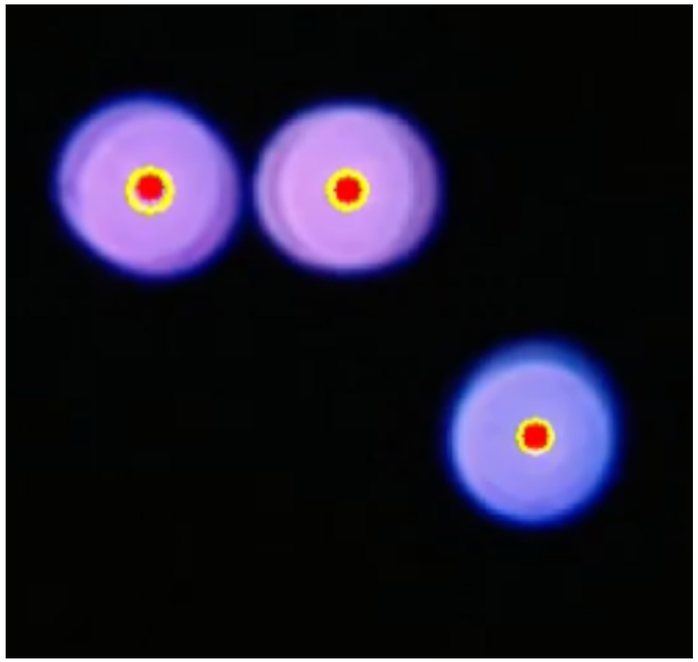
- Problem :
  - Robots cannot get closer



NYU TANDON SCHOOL OF ENGINEERING

Blob Detection

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



NYU TANDON SCHOOL OF ENGINEERING

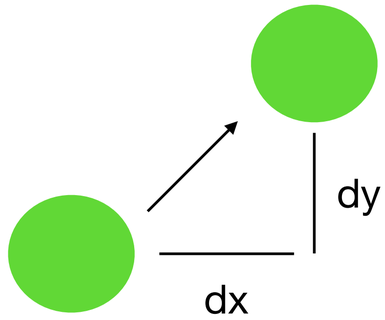
Initialization - Localization

- 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

NYU TANDON SCHOOL OF ENGINEERING

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 out 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



NYU TANDON SCHOOL OF ENGINEERING

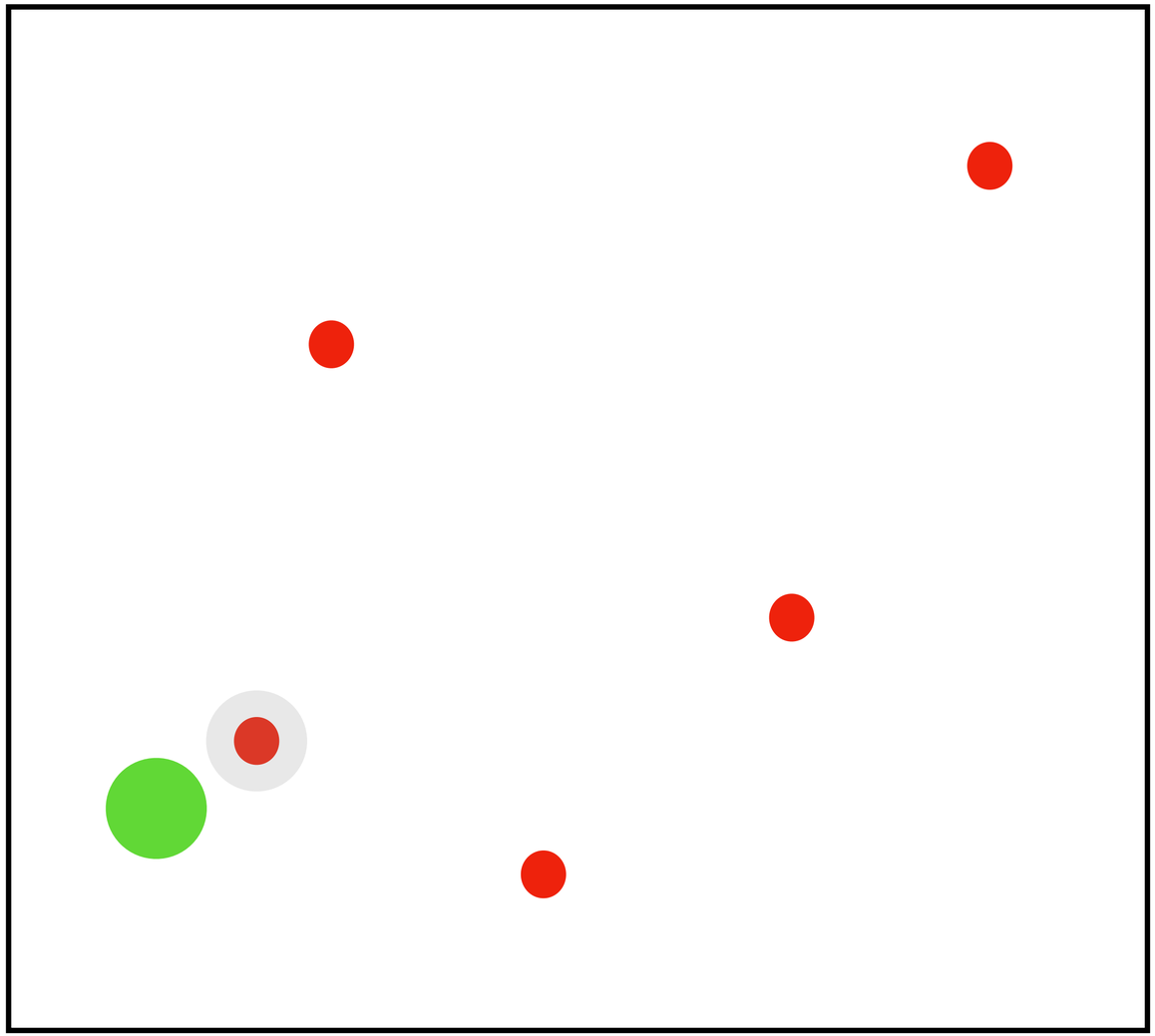
Tracking

Kalman Filter

$$\begin{bmatrix} X\_new \\ Y\_new \\ x\_velocity\_new \\ y\_velocity\_new \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ x\_velocity \\ y\_velocity \end{bmatrix}$$

NYU TANDON SCHOOL OF ENGINEERING

Distance based tracking



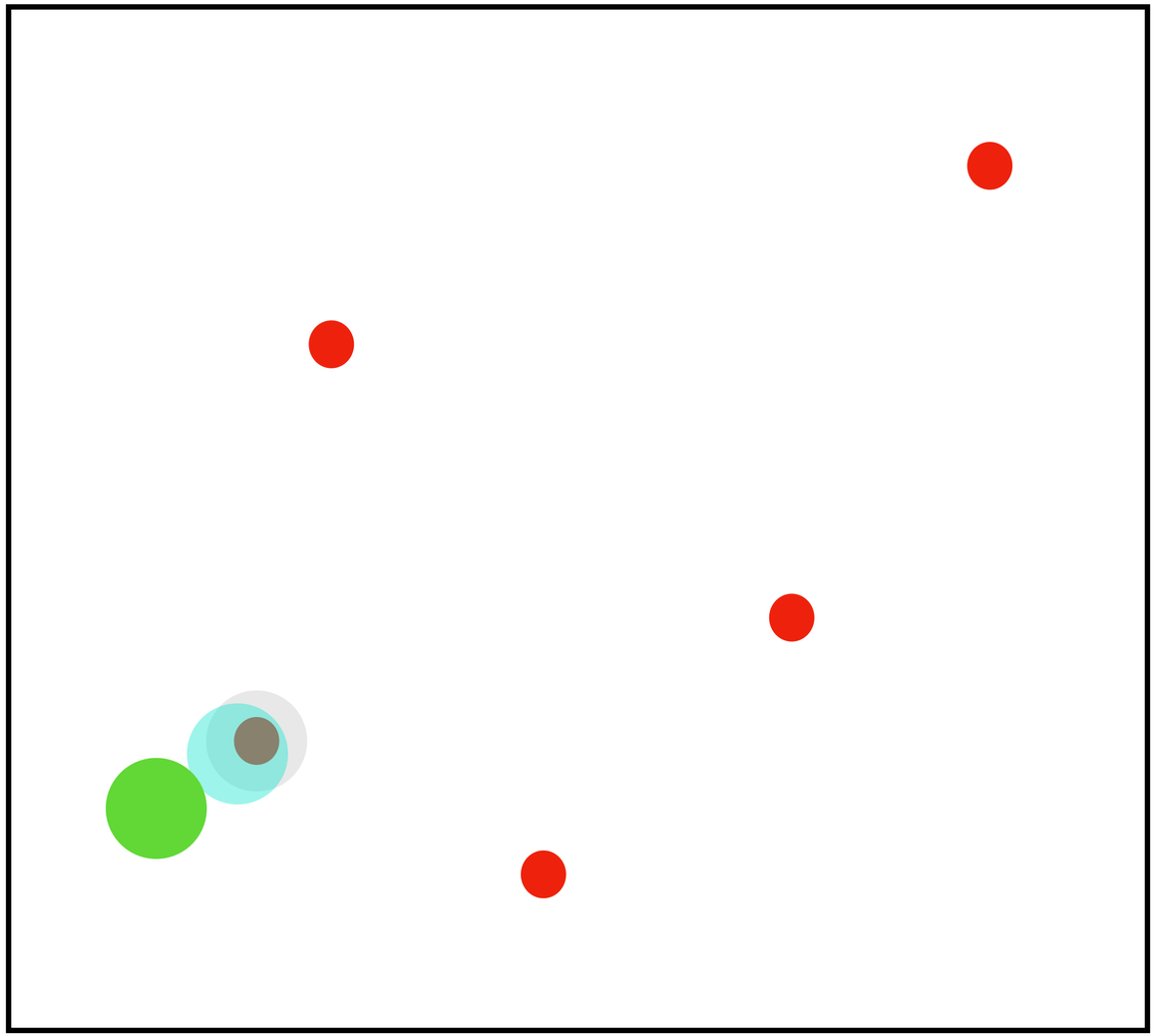
Robot(t-1)

Blobs

Best possible location at time t

NYU TANDON SCHOOL OF ENGINEERING

KF based tracking



Robot(t-1)

Blobs

Best possible location at time t

Robot position estimate(t)

NYU TANDON SCHOOL OF ENGINEERING

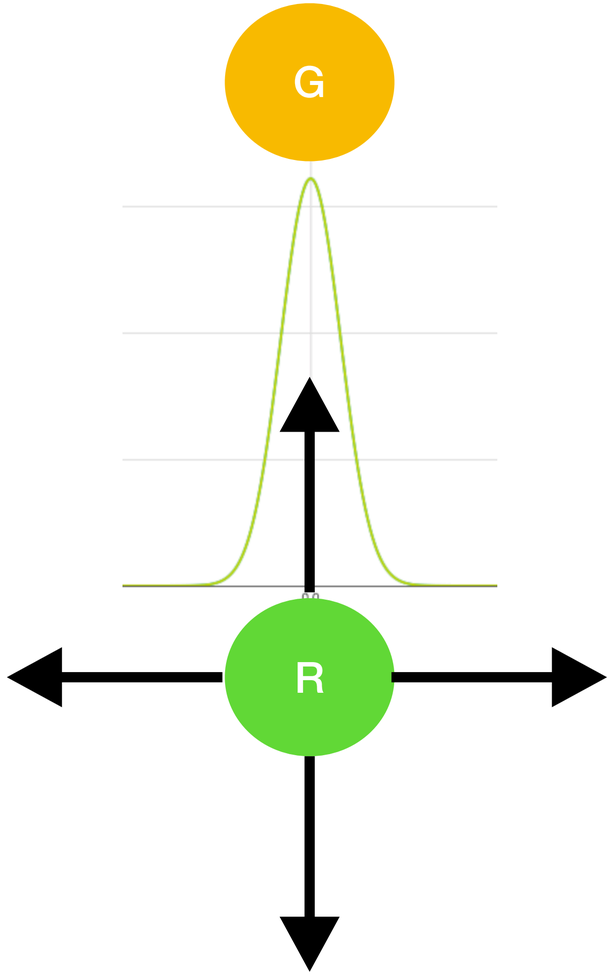
PID Control

- PD control is preferred for similar control situation.
- The reason a PID controller is used is because we might have uncertainty in out initial heading estimate resulting in a constant drift. The Integral term accounts for this drift.

NYU TANDON SCHOOL OF ENGINEERING

PID Control

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




```
dist = np.linalg.norm(np.array(position)-np.array(goal))
old_dist = np.linalg.norm(np.array(position_old)-np.array(goal))
dist_sum += dist
speed = int(guass(0.0,9.0,(angle_to_goal-steering+heading_error)/100.0) * (15*dist + 190*(old_dist - dist) + 0.00000001*dist_sum))
```

NYU TANDON SCHOOL OF ENGINEERING

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

NYU TANDON SCHOOL OF ENGINEERING

Swarming

- Swarming node also is programmed with the following functions:

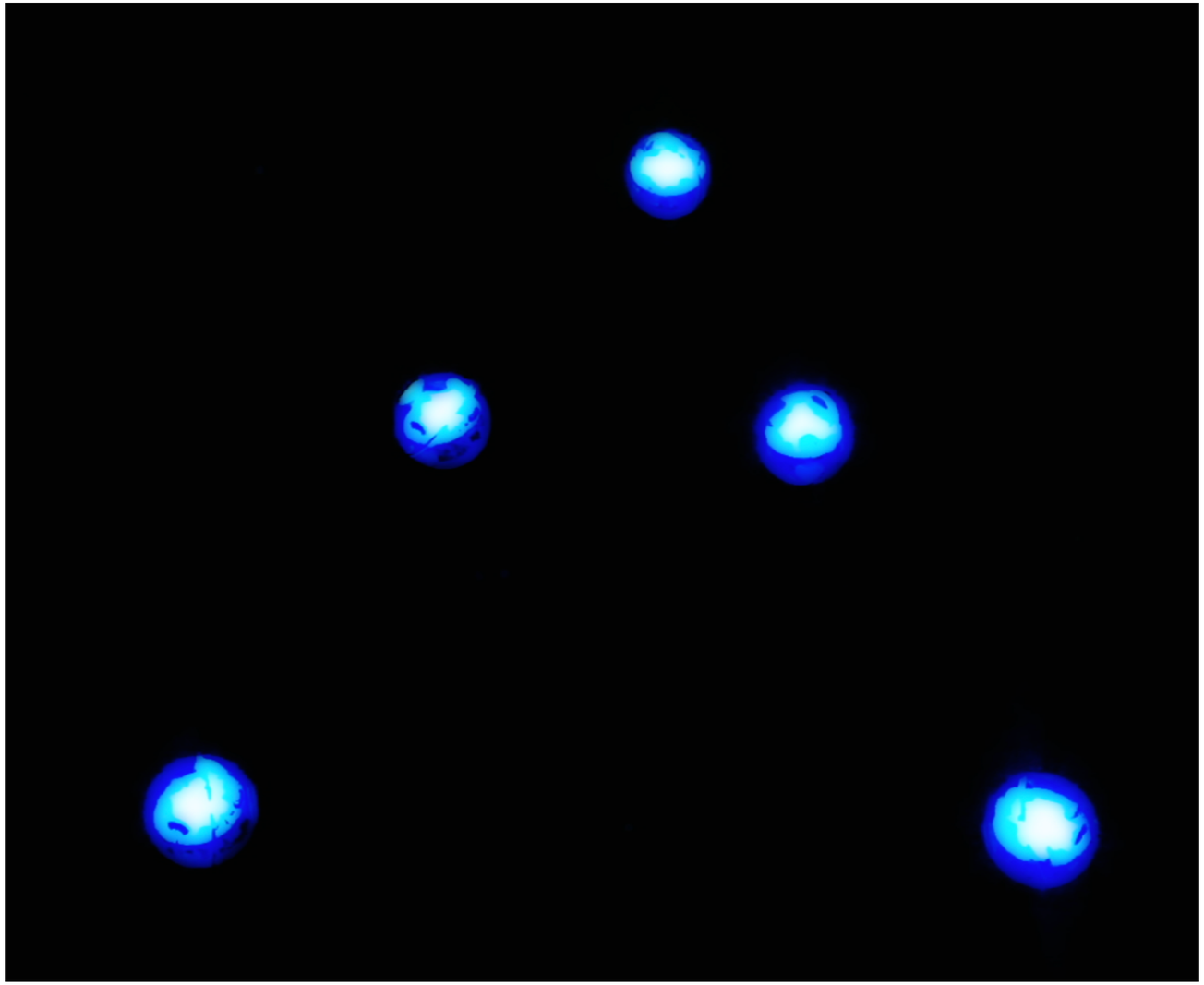
1. Hold a list of formations to cycle through
2. Check of formation consensus is achieved
3. Auto switch formations
4. Scale formations
5. Rotate formation
6. Move formation about any point
7. Camera angle compensation
8. Take user input in the command terminal to switch to any formation

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

NYU TANDON SCHOOL OF ENGINEERING

Swarming

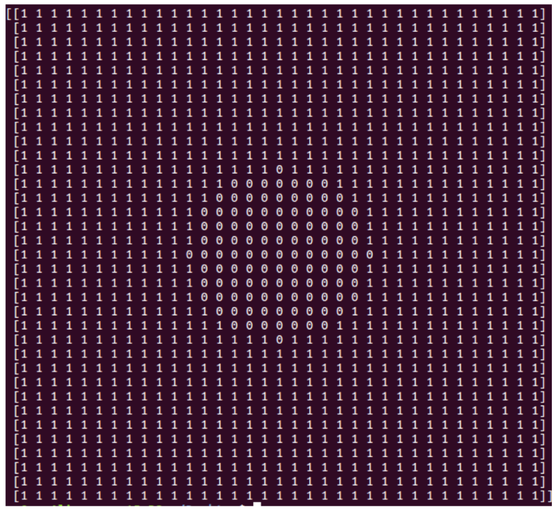


Formation of alphabet A with 5 robots

NYU TANDON SCHOOL OF ENGINEERING

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.



NYU TANDON SCHOOL OF ENGINEERING

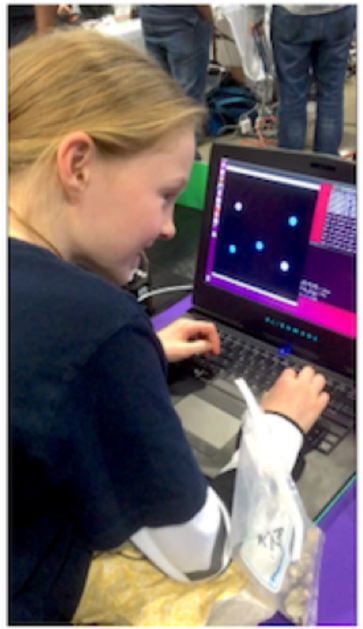
1 vs 1 escape

- 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

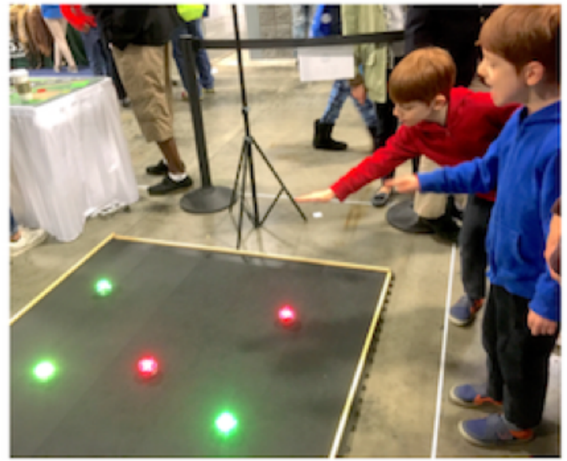
NYU TANDON SCHOOL OF ENGINEERING

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



NYU TANDON SCHOOL OF ENGINEERING

VIDEOS

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

Scripts, Report and All videos are uploaded in the drive