Part 1: Localization techniques

Odometry based on wheel encoders, help with debugging
Part 2: Spatial coordination

Local avoidance, Help with debugging (flocking)

Neural Net form:
- Easy to implement
- PSO
- Close to the sensors
- Smooth trajectories
Part 3: Particle Swarm Optimization

Run several simulations, Help with debugging

Simulations:
- 780 run
- ~3h of computation

Bugs:
- World restart
- Robot angles
- Speed saturation

Report:
Participation in the writing
Part 1: Localisation

Controller

Supervisor

- GPS
- Accelerometer
- Wheel encoder
- Kalman filter

+ Test and debugging

Trajectory localization

Graph showing trajectory localization with different measurement techniques:
- True position
- GPS measure
- Wheel odometry measure
- Kalman estimation
- Accelerometer odometry
Part 2: Formation and Flocking

Supervisor

Formation
- Follower Controller
- Leader Controller

Local avoidance

Communication

Localisation

Flocking Controller

Supervisor

ID, position
Part 3: Optimization

Use metric defined in previous part

Supervisor PSO flocking

Flocking Controller

Supervisor PSO formation

Formation Controllers

Example of formation simulation with PSO

Metric over time (moving average)

- Test and debugging
DIS Project
Implementing a flocking / formation control

Group 1: Frank Centamori’s participation
1st Part: Localization

GPS:
- Retrieve position every second
- Set orientation when robot is still (hardcoded) or compute it when the robot is moving straight
- Store complete pose in a meaningful way for other parts

Metric:
- Send estimated position to the supervisor to compute the metric

Supervisor:
- Creation of the controller
- Retrieve estimated positions via communication
- Retrieve exact positions using supervisor built-in function (wb_supervisor)
- Compute metric using previous data
- Display result and store position data for analysis
2nd Part: Group movement

Close cooperation with Emmanuel Lehnherr who carried out Flocking

**Formation in a leader – follower case**
- Leader sends estimated location to followers
- Leader → migration goal
- Followers → position around leader
- PI controller on transformation of direction for non-holonomic robots
- Tuning by hand to ensure good formation rather than speed
- Adaptation of code for obstacle or crossing case

- Computes metric from absolute locations of robots
- Stores data for analysis
- Adaptation of code for obstacle or crossing case
3rd Part: PSO

- Merging controllers from previous part with PSO
- Allowing PSO to start from scratch or from previous solution
- Testing

- Adaptation of the controller

- Discussion on implementation
- Debugging
- Report
Distributed Intelligent Systems
Project Presentation

Group 1
Emmanuel Lehn herr

Lausanne 08-07-2021
Odometry: accelerometer

**Algorithm**

1. **Calibration (5s)**
   - \( \text{acc}_\text{mean} \)

2. **Measure acc.**
   - \( \text{acc} = \text{acc} - \text{acc}_\text{mean} \)

3. **Compute velocity**
   - \( v = \text{acc} \Delta T \)
   - \( v_x = v \sin(\text{heading}) \)
   - \( v_y = v \cos(\text{heading}) \)

4. **Compute global position**
   - \( x = x + v_x \Delta T \)
   - \( y = y + v_x \Delta T \)
Flocking

Reynolds Rules

1. **Cohesion**: the robots try to move towards the center of the flock

2. **Separation**: they move away from their neighbors

3. **Alignment**: they try to keep the same average speed

• Calculation of the weighted contribution for each rule and for migration
• Controller: transformation of velocity into robot coordinates, max_speed
• Crossing: solving problem with initial orientation
• Supervisor: calculate the metric for flocking performance
• Debugging

+ Migratory urge
PSO - Flocking

With PSO

 ✓ Faster to the goal
 ✓ Better mean metric
 ✓ Slightly better results with our own solution as starting point

• Adaptation of the flocking controller
• Testing and running PSO
• Debugging PSO (problem for restarting the simulation)
• Report