Graph-Based Formation with Real E-pucks

Udaranga Wickramasinghe
Arun Balajee
Objective and Our Approach

• Objective:
  – Follow a moving leader while maintaining a square formation

• Our Approach:
  – We implemented the algorithm proposed by Falconi et al.
  – We did the implementation in 3 steps then validated it in webots and finally tested on real epucks.
Three Steps in Implementation

• Step 1: Implementation of the basic algorithm with constant weight matrix
• Step 2: Incorporation of dynamic weight matrix update depending on the followers distance to the leader
• Step 3: Introduction of the Integral controller that helps to catch up the moving leader
Range and Bearing Information in e-pucks

• We used Librcom library for the measurements.
• The distance measure is the distance between the centers of 2 robots and the range is about 20cm.
• The angle is measured counter clockwise and the values are discrete.

*Values are rounded to the closest integer
Metric for Performance Evaluation

- To compare different implementations and to tune the parameters it is important to have an error measurement metric.
- Falconi et al. has used SwissTrack and Jim et al. has used an overhead camera to track the robots and have used those results to measure the performance.
- Since we did not have those capabilities we decided to come up with our own metric.
Proposed Metric

\[ MSE = \left( e_1 \sin(\alpha_2 - \alpha_1) - L \sin \left( \frac{\pi}{4} \right) \right)^2 \]
\[ + \left( e_1 \cos(\alpha_2 - \alpha_1) - L \cos \left( \frac{\pi}{4} \right) \right)^2 \]
\[ + (e_2 - \sqrt{2} L)^2 \]
\[ + \left( e_3 \sin(\alpha_3 - \alpha_1) - L \sin \left( \frac{\pi}{4} \right) \right)^2 \]
\[ + \left( e_3 \cos(\alpha_3 - \alpha_1) - L \cos \left( \frac{\pi}{4} \right) \right)^2 \]

\( L \): Length of a side of the square formation
Experiments in Webots

• Comparison between three stages of the algorithm

\[ W = I \text{ and } K_i = 0 \]  
\[ \text{Dynamic } W \text{ and } K_i = 0 \]  
\[ \text{Dynamic } W \text{ and } K_i = 0.01 \]
RMSE variation over time
Effect of Integral Controller

\[
K_i = 0, \quad K_i = 0.001, \quad K_i = 0.01, \quad K_i = 0.05, \quad K_i = 1
\]

RMSE vs Time (min)
Experiments with E-pucks

• First we put the complete implementation (with dynamic weights and integral controller) of the algorithm and tried to tune the controller.
• We set $K_i = 0$ and tried to tune $K_u$ and $K_w$. The results were unpredictable.
• Anyhow $K_w = 1$ and $K_u = 0.1$ gave little bit better results.
• Then we started increasing $K_i$ and the results become very unpredictable and tuning them was difficult.
• So decided to move to the basic version of the algorithm where we had $W = I$ and $K_i = 0$. We were able to find little bit better results with $K_u = 1.5$ and $K_u = 0.8$. But still the performance is very unpredictable.
Reasons Behind Poor Performance

• Low range of communication for epucks. As a result epucks easily gets out of the range of others.

• Even when epucks are within range, we have loss of communication.

• Discrete angles values returned by LibIrcom library.

• Differences in wheel speeds between two wheels in epucks.
Comparison between epucks and webots

- We kept all the parameters in the algorithm same in both epucks and webots.
- But we failed to do it for $K_u, K_w$ and $K_i$ because the values that give good results on webots do not work in epucks and vice versa.
- So we tuned them separately for two cases.
- We also set the range of IR emitter in webots to 20cm to mimic the real world scenario and the wheel speeds of the leader to same value (30).
RMSE Values

• We repeated experiment 9 times with epucks. Four times we got very bad results. So we removed them.

• We did the same experiment 5 times on webots as well with slight variations in initial position.
Suggestions

• Spend more time in tuning parameters
• Implement the algorithm proposed by Jim et al. for angle measurement in epucks.
Thank you!