Odometry based on wheel encoders

- Proprioceptive sensors (encoders)
- Angle difference between timesteps for each wheel is used to calculate speed \((v)\) and heading \((\theta)\) in robot’s reference frame
- Ideally no wheel slip
- Integrate noise over time (Webots : Gaussian 0.01 rad)
- Must give initial pose

![Coordinate system](image)
Formation: Crossing

Formation is achieved using Graph-Based Consensus

\[ x[k+1] = x[k] - L*(x[k] - \text{bias}) \]

Non-holonomic:

=> Transform \( x[k+1] \) into actual wheel speed using a P controller (2 params: Ku & Kw (Optimised with PSO))

\[
\begin{align*}
\mathbf{u} &= K_u \cdot \sqrt{x^2 + y^2} \cdot \cos(\arctan(y, x)) \\
\mathbf{\omega} &= K_\omega \cdot \arctan2(y, x)
\end{align*}
\]

How we made it work:

- Migration vector to move the formation
- Added Braitenberg to handle obstacles
- EKF for self absolute localisation
- Estimate other localisation from IR

Graph not fully connected

\[
L = \begin{bmatrix}
4 & -1 & -1 & -1 & -1 \\
-1 & 3 & -1 & -1 & 0 \\
-1 & -1 & 3 & 0 & -1 \\
-1 & -1 & 0 & 3 & -1 \\
-1 & 0 & -1 & -1 & 3
\end{bmatrix}
\]

Bias for the left formation flock:

\[
\text{Bias} = \begin{bmatrix}
0 & 0.15 & 0.15 & -0.15 & -0.15 \\
0 & 0.15 & -0.15 & 0.15 & -0.15
\end{bmatrix}
\]
Formation: Crossing results

- PSO optimised Ku & Kw gains
- Testing world has a bigger run-up
- Increasing the speed would result in robot not being able to “catch-up”
Odometry based on accelerometer

- Compute the speed and position based on physics
- Acceleration bias and deterministic error removed
- Angle based on wheel encoders
- Error (double) integrates

\[ s_i = s_{i-1} + \gamma \times \tilde{a}_i \times \Delta t \]
\[ x_{i+1} = x_i - s_i \times \Delta t \times \sin(\theta) \]
\[ y_{i+1} = y_i - s_i \times \Delta t \times \cos(\theta) \]
### Spatial Coordination: Flocking

- **General loop:**
  - Reevaluate self-position (Extended Kalman Filter)
  - Estimate speed and position of the robots
  - Apply Reynold’s rule
  - Check obstacle → Braitenberg obstacle avoidance
  - Rescale speed
  - Apply speed to the robot (K controller)

\[
\vec{v}_j = w_c \vec{v}_c + w_s \vec{v}_s + w_m \vec{v}_m
\]

\[
\vec{v}_c = \frac{1}{N-1} \sum_{i=1; i \neq j}^N \vec{x}_i
\]

\[
\vec{v}_s = \sum_{i=1; i \neq j}^N \begin{cases} 
0 & \text{if } \|\vec{x}_i - \vec{x}_j\| \geq 4R \\
-\frac{1}{\vec{x}_i} & \text{else}
\end{cases}
\]

\[
\vec{v}_m = \vec{x}_{obj} - \vec{x}_j
\]

- **No alignment**
- **No priority between the rules**
Particle Swarm Optimisation

- Public group homogeneous policy solution
- Lbest neighborhood
- Module independent from application (library)
- Noise resistance with PBEST
- Only trained on the obstacle world

### Manual Solution

<table>
<thead>
<tr>
<th>Rule</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohesion</td>
<td>1.1</td>
</tr>
<tr>
<td>Separation</td>
<td>0.02</td>
</tr>
<tr>
<td>Migration</td>
<td>0.015</td>
</tr>
</tbody>
</table>

### PSO Solution

<table>
<thead>
<tr>
<th>Rule</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohesion</td>
<td>0.3</td>
</tr>
<tr>
<td>Separation</td>
<td>0</td>
</tr>
<tr>
<td>Migration</td>
<td>0.009</td>
</tr>
</tbody>
</table>

---

**Approach 2: PSO pbest**

1: Initialize particles
2: for $N_i$ iterations do
3: for $N_p$ particles do
4: Update particle position
5: Evaluate particle
6: Re-evaluate personal best
7: Aggregate with previous best
8: Share personal best
9: end for
10: end for

---

**Test obstacle world with PSO solution**

10x
Formation

- Graph-based formation

\[ L = \begin{bmatrix}
4 & -1 & -1 & -1 & -1 \\
-1 & 3 & -1 & -1 & 0 \\
-1 & -1 & 3 & 0 & -1 \\
-1 & -1 & 0 & 3 & -1 \\
-1 & 0 & -1 & -1 & 3
\end{bmatrix} \]

\[ v_k = -\sum_{j=1}^{N} L_{ij} \left( dx_k^i - b^i_k \right) + v_{migr,k} \text{ for } k = 1, 2 \]

- IR communication \( \rightarrow \) relative position estimates \( dx \)
- Proportional speed control
  \[
  \begin{cases}
  u = K_u ||v||\cos(\alpha) \\
  \omega = K_\omega \alpha
  \end{cases}
  \]

  with \( \alpha = \text{atan2}(\dot{x}_2, \dot{x}_1) - \theta \)

- Braitenberg obstacle avoidance
PSO

- Optimize for proportional gains $K_u$ and $K_\omega$
- p-best PSO
- Performance metric
  \[ M_{f_0} = d_{f_0} \frac{||\dot{x}_{k+1} - \dot{x}_k||}{T} \]
- Randomness in the migration urge
  \[ v = [\pm 0.5, \varepsilon] \text{ with } \varepsilon \in U([-0.1, 0.1]) \]

Remark:
- 1 parameter set ($K_u$ and $K_\omega$) per world
Result

speed x3
Extended Kalman Filter

- EKF is used due to the non-linearity relation between states.
- 5 states: $\hat{x} = [x, z, \theta, v, \omega]$.
- Prediction model based on the velocity commands.
- Measurement models based on odometry and GPS.
- GPS provides absolute heading at certain conditions to prevent drifting.
PSO

- Public group homogeneous policy solution.

- Lbest neighborhood.

- Noise resistance with PBEST.

- Module independent from application (library).
PSO on flocking

- The parameters to tune are the Reynlod rules weights.
- Overall performance improved.
- Dispersion weight vanishes due to obstacle avoidance.
- Hard to train in the crossing world.