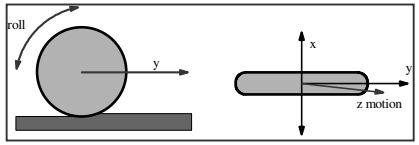
CSE-571
AI-based Mobile Robotics

Wheeled Locomotion

Reactive Collision Avoidance

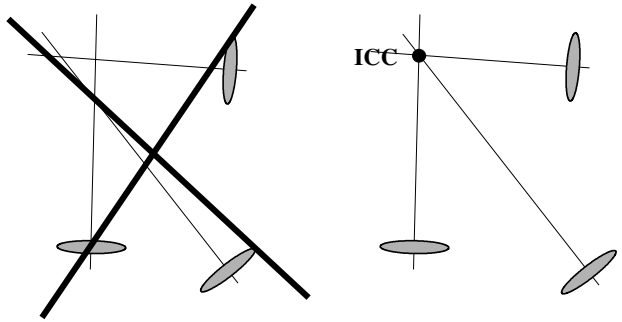
Locomotion of Wheeled Robots

Locomotion (Oxford Dict.):
Power of motion from place to place



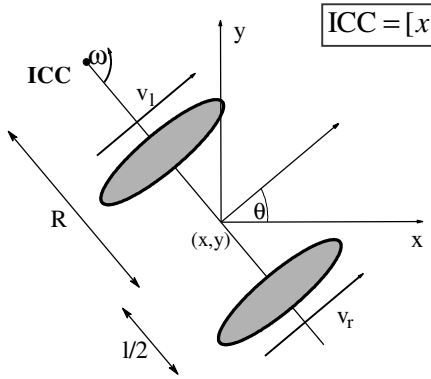
- Differential drive (AmigoBot, Pioneer 2-DX)
- Synchronous drive (B21)
- Car drive (Ackerman steering)
- Holonomic drive (XR4000)
- Tracked vehicles (Urban)

Instantaneous Center of Curvature



- For rolling motion to occur, each wheel has to move along its y-axis

Differential Drive



$$\mathrm{ICC} = [x - R\sin\theta, y - R\cos\theta]$$

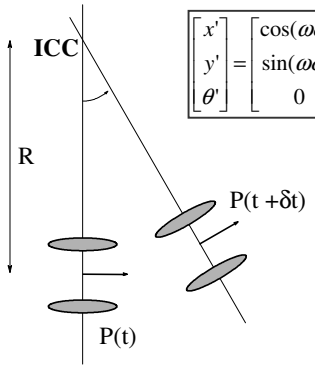
$$\omega(R + l/2) = v_r$$

$$\omega(R - l/2) = v_l$$

$$R = \frac{l}{2}\frac{(v_l + v_r)}{(v_r - v_l)}$$

$$\omega = \frac{v_r - v_l}{l}$$

Differential Drive



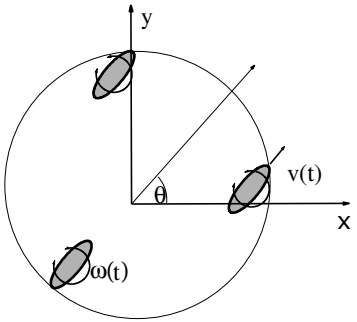
$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - \mathrm{ICC}_x \\ y - \mathrm{ICC}_y \\ \theta \end{bmatrix} + \begin{bmatrix} \mathrm{ICC}_x \\ \mathrm{ICC}_y \\ \omega\delta t \end{bmatrix}$$

$$x(t) = \frac{1}{2}\int_0^t [v_r(t') + v_l(t')]\ \cos[\theta(t')]dt'$$

$$y(t) = \frac{1}{2}\int_0^t [v_r(t') + v_l(t')]\ \sin[\theta(t')]dt'$$

$$\theta(t) = \frac{1}{l}\int_0^t [v_r(t') - v_l(t')]\ dt'$$

Synchonous Drive



$$x(t) = \int_0^t v(t')\ \cos[\theta(t')]dt'$$

$$y(t) = \int_0^t v(t')\ \sin[\theta(t')]dt'$$

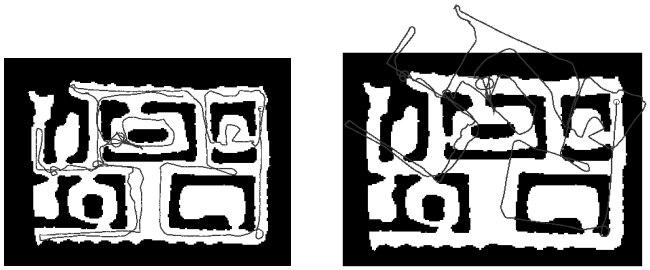
$$\theta(t) = \int_0^t \omega(t')\ dt'$$

Holonomic Drive: Priamos, XR4000

[Priamos, Karlsruhe]

Tracked Vehicle: Urban Robot

Dead Reckoning



CSE-571
AI-based Mobile Robotics

Reactive Collision Avoidance

Dynamic Window Approach

Gradient Method

Reactive Collision Avoidance

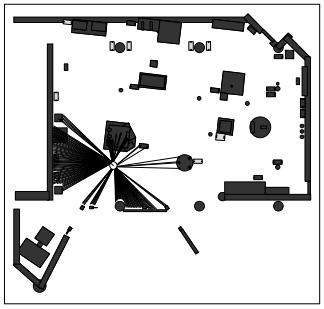
- Dynamic window approach
- Gradient method

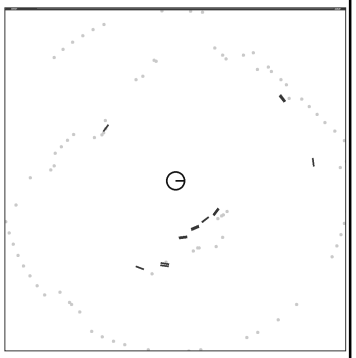
Dynamic Window Approach

- Consider combinations *(v,w)* of translational and rotational velocities.
- Each combination specifies a circular trajectory and the robot's velocity on this trajectory.
- Search for optimal velocities considering the following constraints:

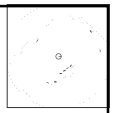
  - **Hard constraints** restrict search space to guarantee safe navigation.
  - **Soft constraints** express preferences among velocities.

Obstacle Line Field

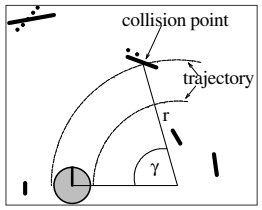


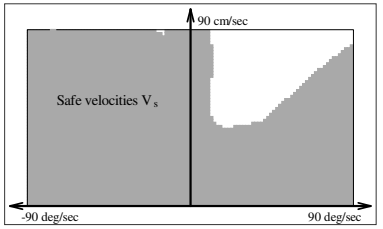


Safety



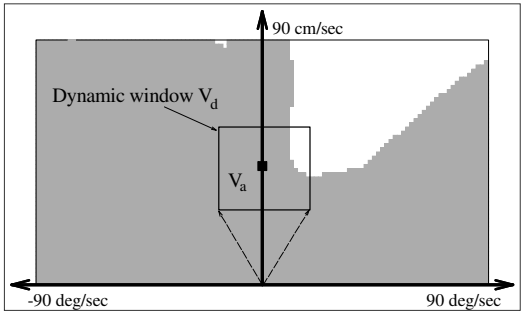
- Robot can stop safely before reaching an obstacle





Dynamics

- Velocities must obey dynamics of the robot.



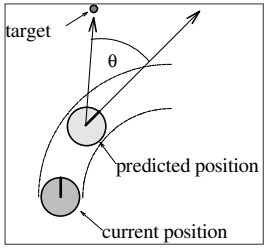
Soft Constraints

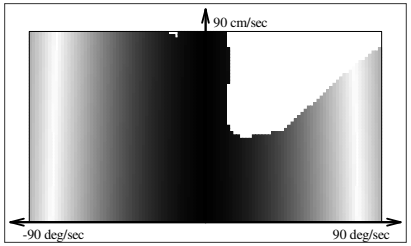
- Maximize objective function

$$G(v,\omega) = \sigma(\alpha \cdot \text{heading}(v,\omega) + \beta \cdot \text{dist}(v,\omega) + \gamma \cdot \text{vel}(v,\omega))$$

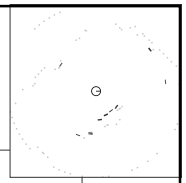
- **Target heading**: Alignment with goal.
- **Clearance**: Distance to closest obstacle on trajectory.
- **Velocity**: Forward velocity of the robot.

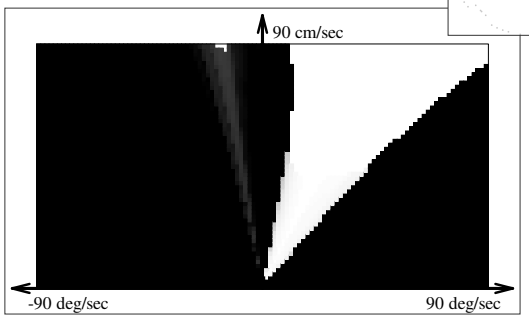
Target Heading



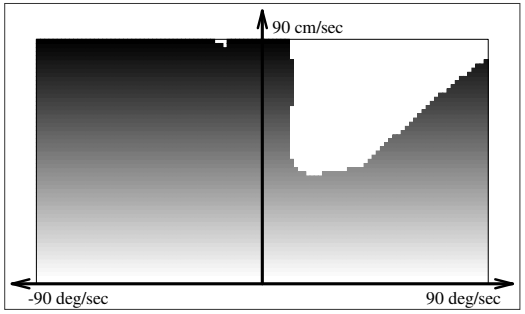


Clearance

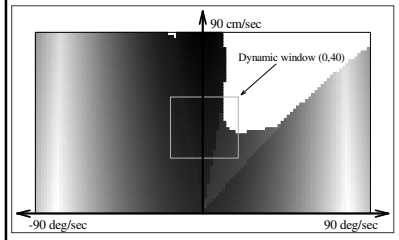


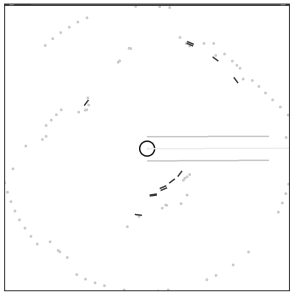


Velocity

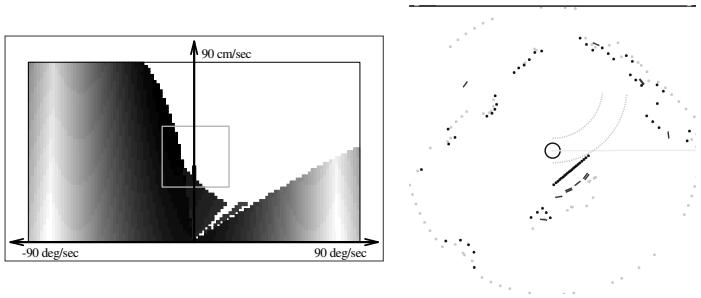


Smoothed Evaluation

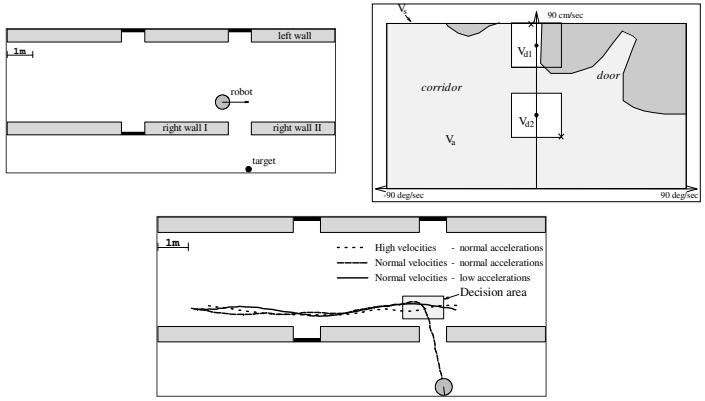




Additional Map Information



Why Dynamics?



Gradient Method

- Performs efficient, optimal path planning in the local perceptual space
- Path costs depend on length and closeness to obstacles

Paths and Costs

- Path is list of points $P=\{p_1, p_2, \ldots p_k\}$
- p_k is only point in goal set
- Cost of path is separable into intrinsic cost at each point along with adjacency cost of moving from one point to the next

$$F(P) = \sum_i I(p_i) + \sum_i A(p_i, p_{i+1})$$

  - Adjacency cost typically Euclidean distance
  - Intrinsic cost typically occupancy, distance to obstacle

Navigation Function

- Assignment of potential field value to every element in configuration space [Latombe, 91].
- Goal set is always downhill, no local minima.
- Navigation function of a point is cost of minimal cost path that starts at that point.

$$N_k = \min_{P_k} F(P_k)$$

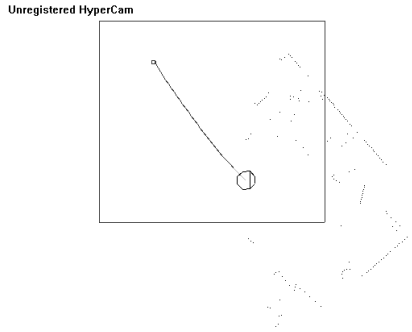
Computation of Navigation Function

- Initialization
  - Points in goal set ← 0 cost
  - All other points ← infinite cost
  - Active list ← goal set
- Repeat
  - Take point from active list and update neighbors
  - If cost changes, add the point to the active list
- Until active list is empty

Further Details

- Keep paths away from obstacles by setting intrinsic cost dependent on distance from obstacles.
- Local perceptual space depends on quality of sensors and odometry.
- Can easily achieve 10Hz update rate on low-end PC (10 x 10m LPS, 10cm resol.).
- Performs superior to human operator.

Example Run



Discussion

- Imagine you want your robot to perform navigation tasks, which approach would you choose?
- What are the benefits of the behavior based paradigm?
- Which approaches will win in the long run?