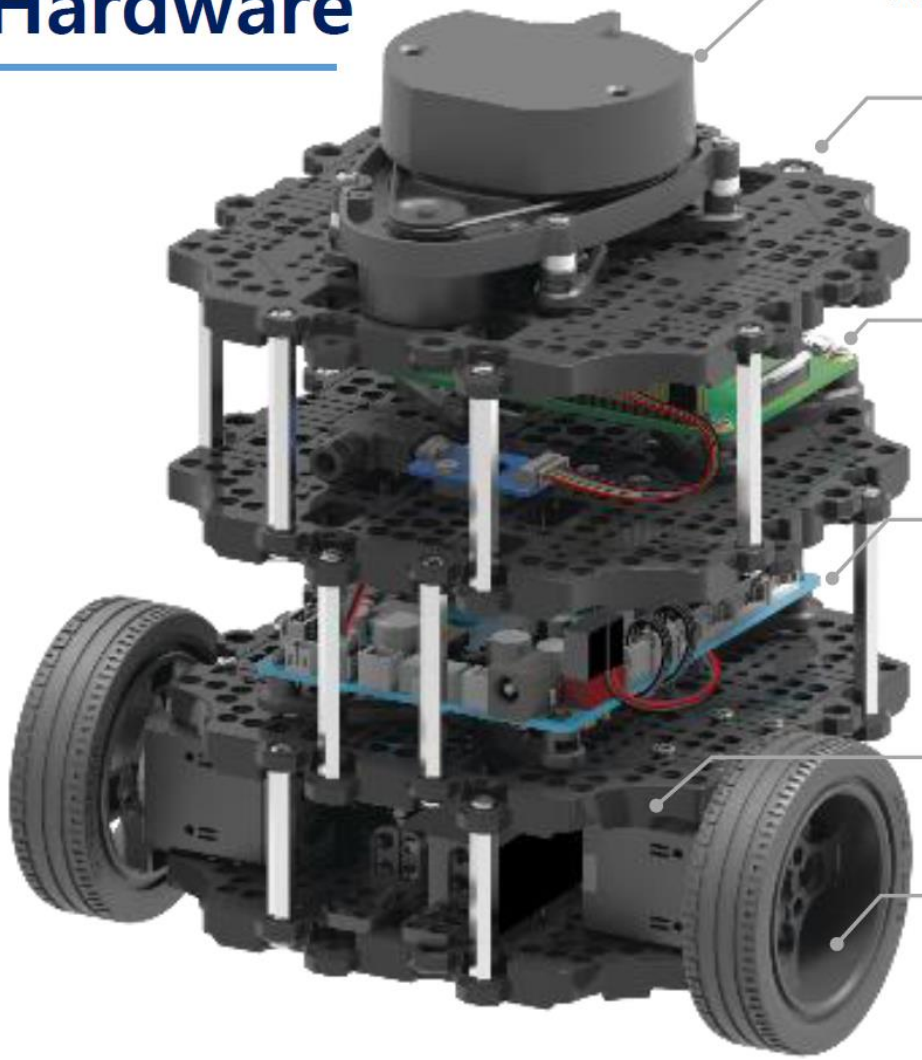
TurtleBot3 Hardware



360° LiDAR for SLAM & Navigation

Scalable Structure

Single Board Computer
(Raspberry Pi)

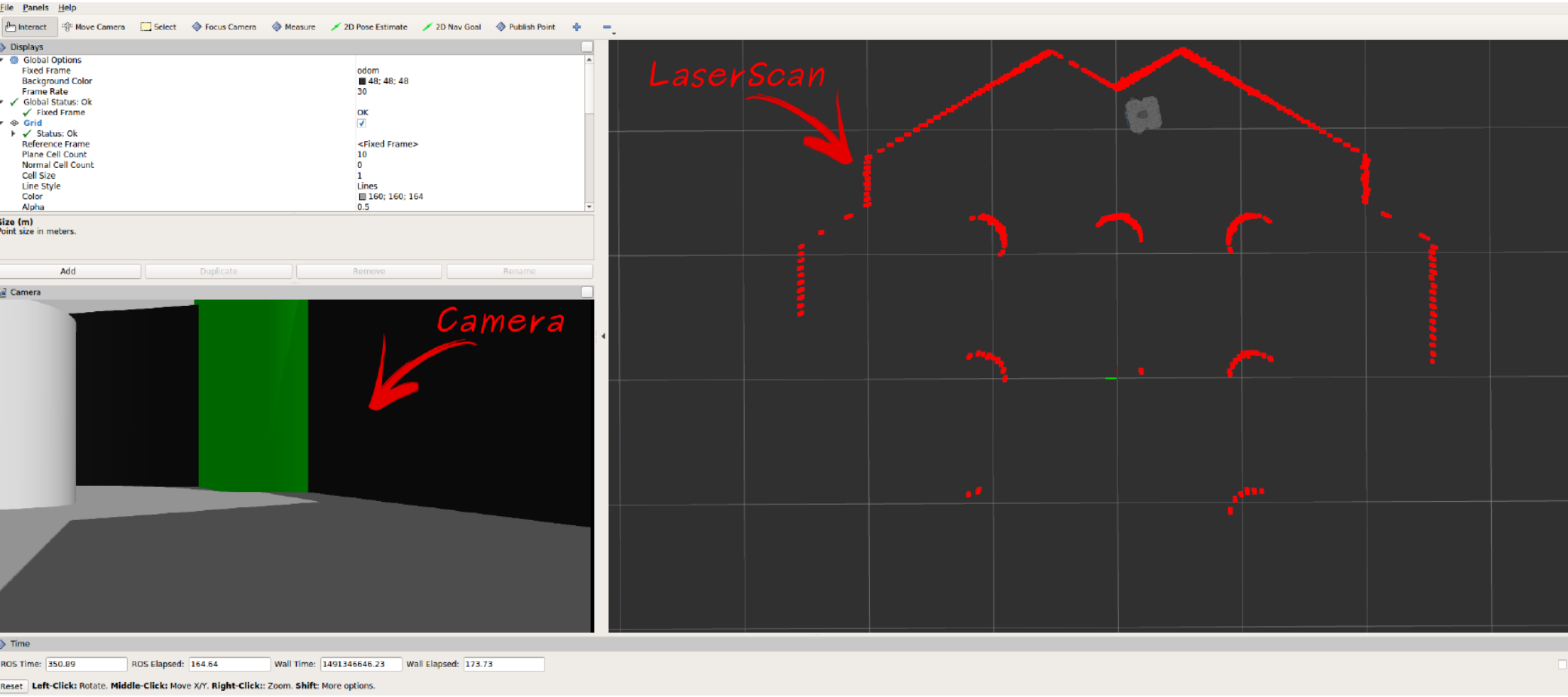
OpenCR
(32-bit ARM Cortex®-M7)

DYNAMIXEL x 2 for
Wheels

Sprocket Wheels for
Tire and Caterpillar

Li-Po Battery

Run Virtual robot with Gazebo



Simultaneous Localization And Mapping
&
Navigation

Path…

Path「Noun」

1. A way beaten, formed, or trodden by the feet persons or animals
2. A narrow walk or way
3. A route, course, or track along which something moves

Path Finding...

Path「Noun」

1. A way beaten, formed, or trodden by the feet persons or animals
2. A narrow walk or way
3. A route, course, or track along which something moves

Long companion of travel

compass & map

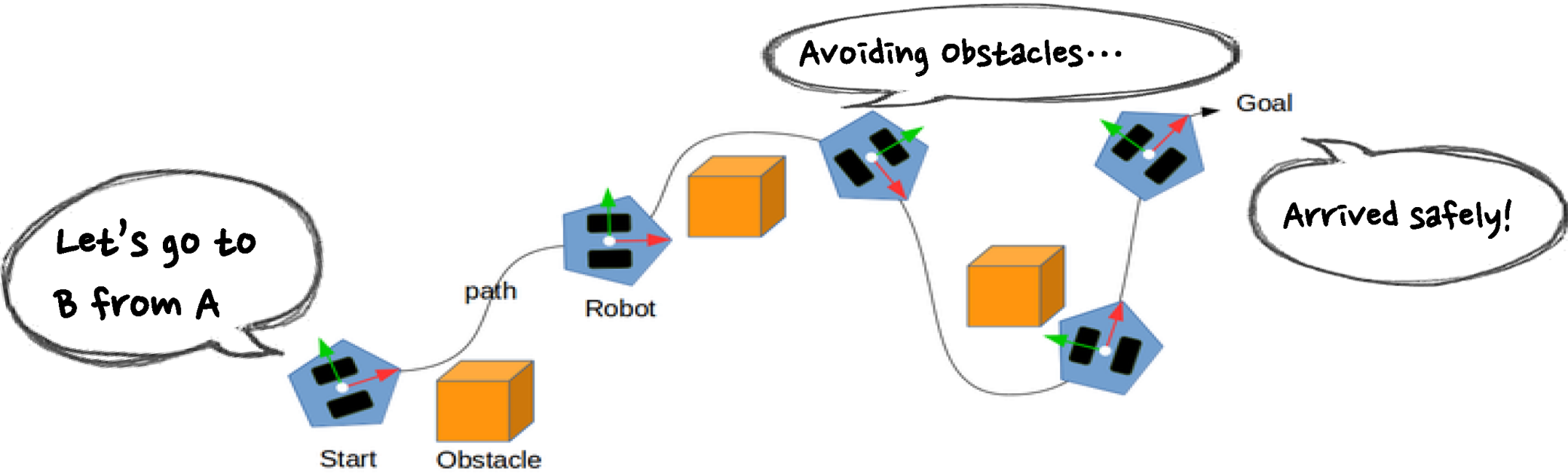
I'm here.

- Travelers with the Sun, the Moon, and stars
- Compass, one of Four great Chinese inventions
- Development of Compass
  - Magnetic compass
  - Front compass
  - GPS
- Map

Big Dipper, by Magnus Manske, Public Domain

pixabay.com, CC0

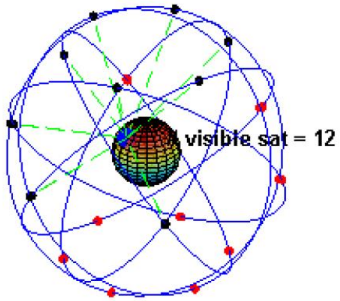
What you need for path finding!



① **Position**: Measuring/estimating the robot's position
② **Sensing**: Measuring obstacles such as walls and objects
③ **Map**: Maps with road and obstacle information
④ **Path**: Calculate optimal path to the destination and follow the path

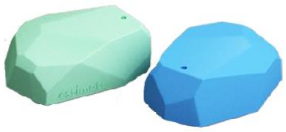
① Position: Measuring/estimating the robot's position

- GPS (Global Positioning System)



  - Error
  - Weather
  - Outdoor

- Indoor Positioning Sensor
  - Landmark (Color, IR Camera)
  - Indoor GPS
  - WiFi SLAM
  - Beacon



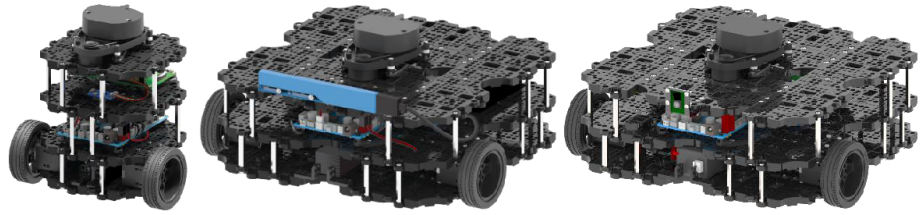
Estimote (Beacon)

StarGazer

Vicon MX

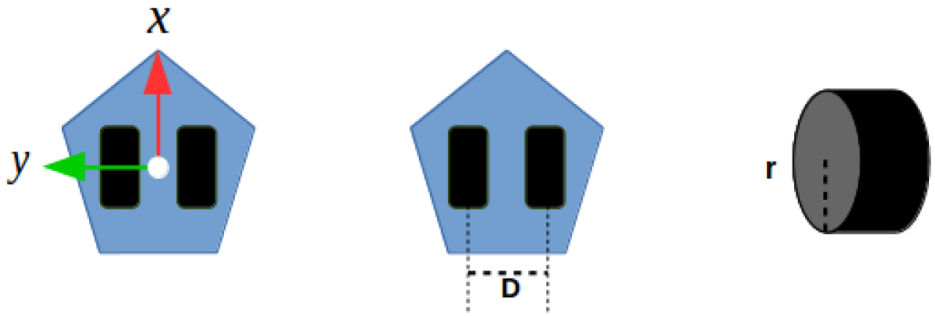
① Position: Measuring/estimating the robot's position

- Dead Reckoning
  - Using the encoder value of both wheel axes
  - Calculate moving distance and rotation angle, and then estimate position
  - Floor slip, mechanical, cumulative error
  - Position compensation with inertial sensor, filter such as IMU
  - Kalman filter...



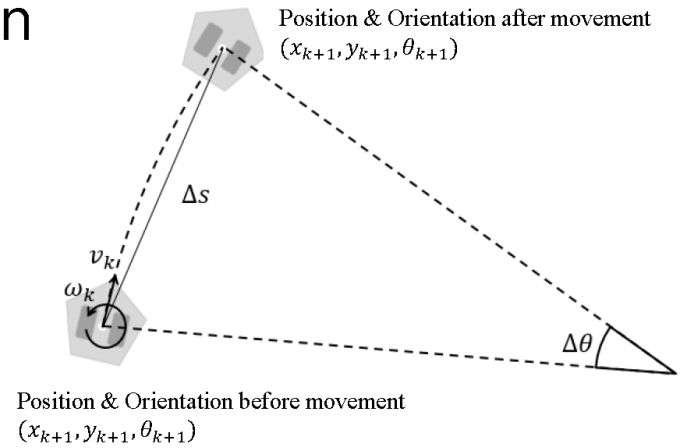
TurtleBot 3

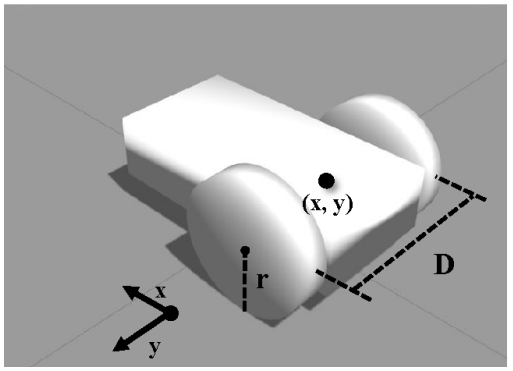
- Required Information
  - Encoder value E on both wheel axes (Recalculated as gear ratio for motor shaft)
  - Distance between wheels D
  - Wheel radius r



① Position: Measuring/estimating the robot's position

- Dead Reckoning Calculation
  - Linear velocity: v
  - Angular velocity: w
- Use Runge-Kutta Formula
  - Approximate value of moved position x, y
  - Rotation angle θ





$$v_l = \frac{(E_l c - E_l p)}{T_e} \cdot \frac{\pi}{180} \text{ (radian/sec)}$$

$$v_r = \frac{(E_r c - E_r p)}{T_e} \cdot \frac{\pi}{180} \text{ (radian/sec)}$$

$$V_l = v_l \cdot r \text{ (meter/sec)}$$

$$V_r = v_r \cdot r \text{ (meter/sec)}$$

$$v_k = \frac{(V_r + V_l)}{2} \text{ (meter/sec)}$$

$$\omega_k = \frac{(V_r - V_l)}{D} \text{ (radian/sec)}$$

$$\Delta s = v_k T_e \qquad \Delta\theta = \omega_k T_e$$

$$x_{(k+1)} = x_k + \Delta s \cos\left(\theta_k + \frac{\Delta\theta}{2}\right)$$

$$y_{(k+1)} = y_k + \Delta s \sin\left(\theta_k + \frac{\Delta\theta}{2}\right)$$

$$\theta_{(k+1)} = \theta_k + \Delta\theta$$

② Sensing: Measuring obstacles such as walls and objects

- Distance Sensor
  - LRF, ultrasonic sensor, infrared distance sensor

- Vision Sensor
  - Stereo camera, mono camera, omni-directional camera

- Depth camera
  - SwissRanger, Kinect-2
  - RealSense, Kinect, Xtion, Carmine(PrimeSense), Astra

③ Map: Maps with road and obstacle information

- Robots need a map to find a path!
- Map
  - Digital maps for infrastructure such as roads!
  - Maps of hospitals, cafes, companies, homes?
  - Maps of unknown, collapsed hazardous areas?

- **Map? Let's make it!**

- **SLAM**

  (Simultaneous Localization And Mapping)

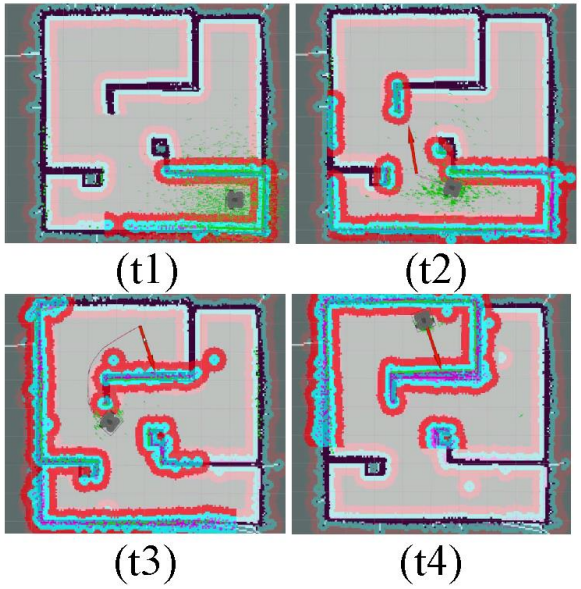
  Together Where am I? Build a map

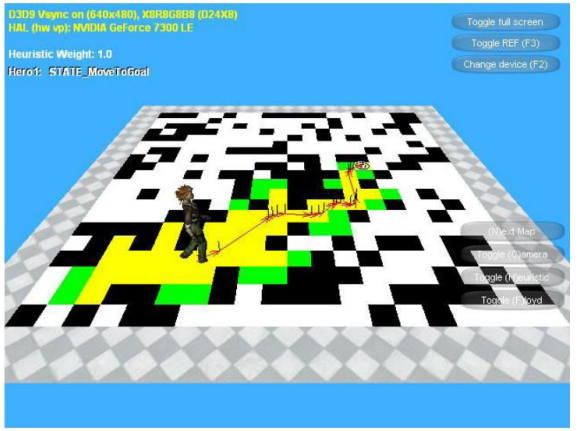
pixabay.com, CC0

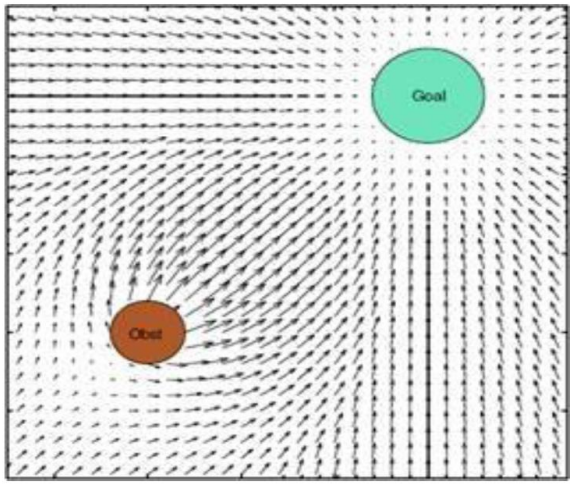
④ Path: Function to calculate optimal path to destination and travel

- Navigation
- Localization / Pose estimation
- Path search and planning

- Dynamic Window Approach (DWA)
- A* algorithm (A Star)
- Potential Field
- Particle Filter
- Graph





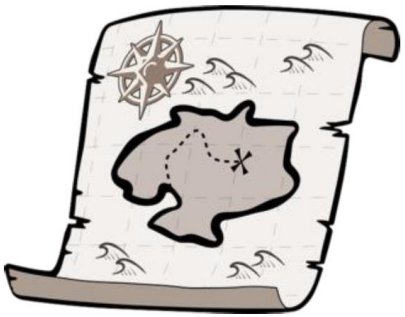


https://students.cs.byu.edu/~cs470ta, http://vimeo.com/3423169

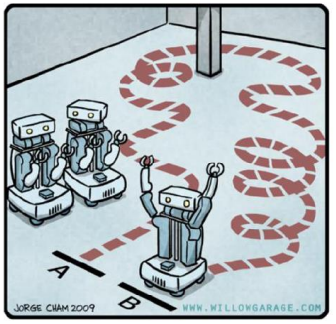
① **Position**

② **Sensing**

③ **Map**



④ **Path**



Position+Sensing → **Map**

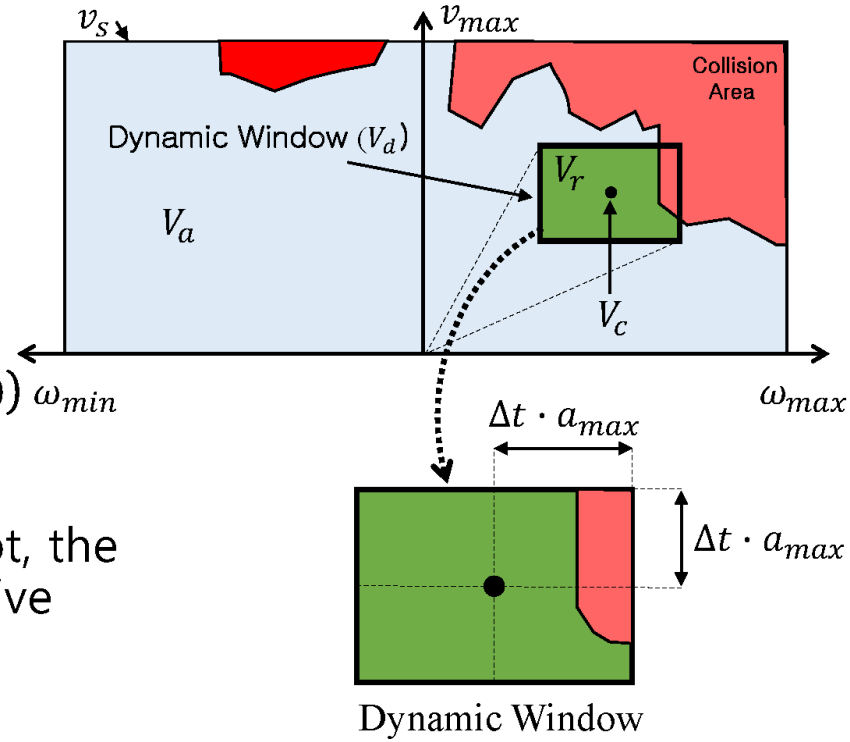
SLAM

Position+Sensing+Map → **Path**

Navigation

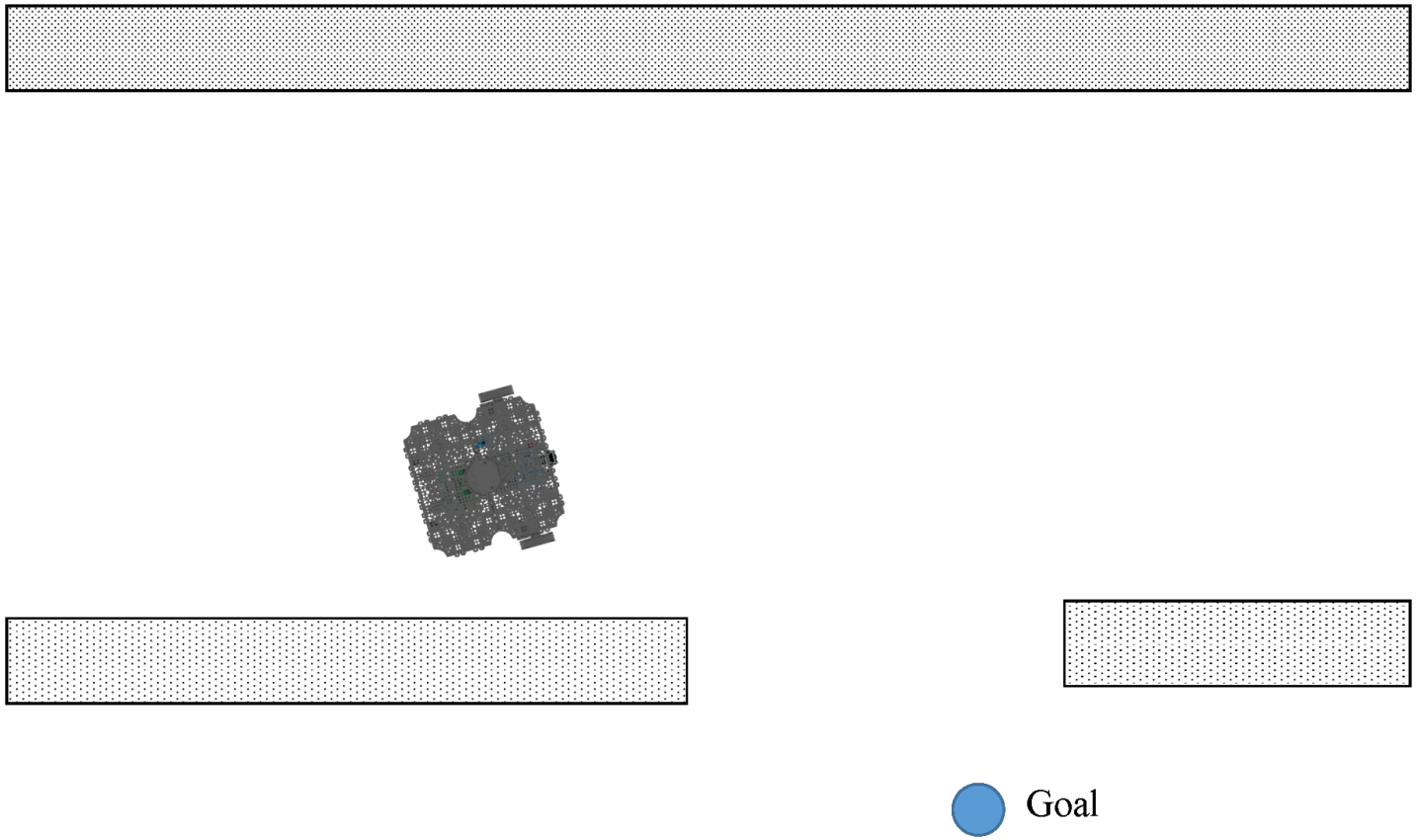
Navigation

- **Dynamic Window Approach** (Mainly used in local plan)
- How to choose the speed at which you can quickly reach the target point while avoiding the obstacles that can collide with the robot in the 'velocity search space' of the robot
  - v (linear velocity), ω (angular velocity)
  - V_s: Available speed range
  - V_a: Permissible speed range
  - V_r: Speed range in dynamic window
  - $G_{(v,\omega)} = \sigma\big(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)\big)$
- Given the direction, speed, and impact of the robot, the objective function G finds v, ω at which the objective function is maximized

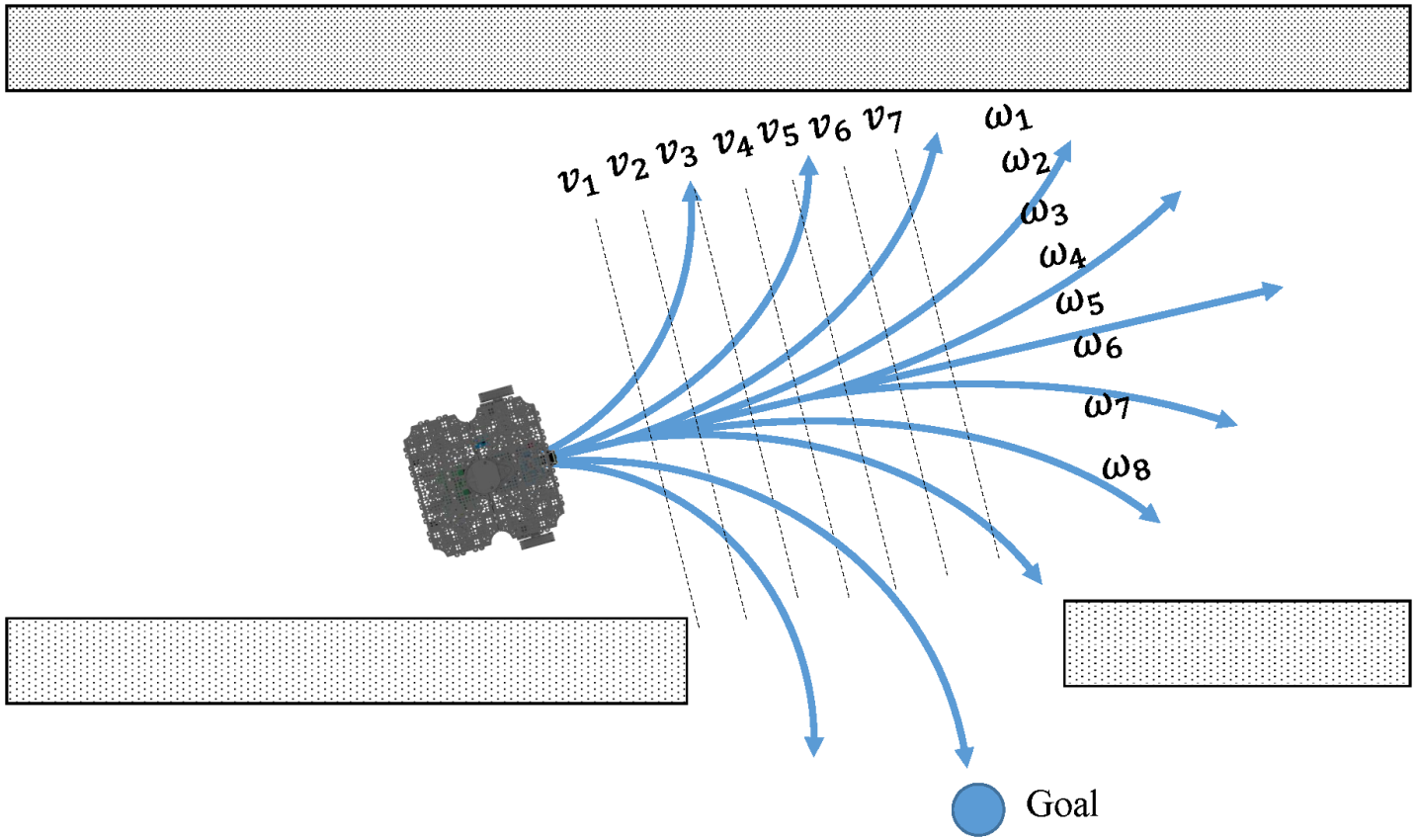


D. Fox, W. Burgard and S. Thrun, The dynamic window approach to collision avoidance, IEEE Robotics & Automation Magazine

Dynamic Window Approach (DWA)



Dynamic Window Approach (DWA)



Dynamic Window Approach (DWA)