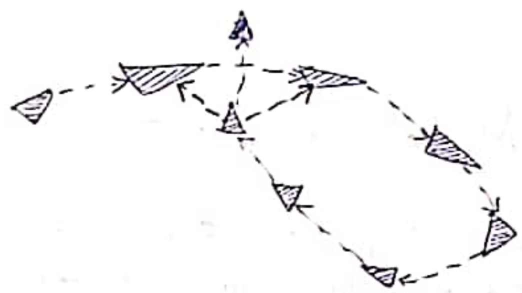
Graph-Based SLAM

{from Intro to Mobile Robotics}

⇒ Constraints connect the poses of the robot while it is moving.

⇒ Constraints are inherently uncertain.



Robot pose — Constraint (odometry) — Constraint (loop closure)

* Idea of Graph based SLAM

→ Use a graph to represent the problem.

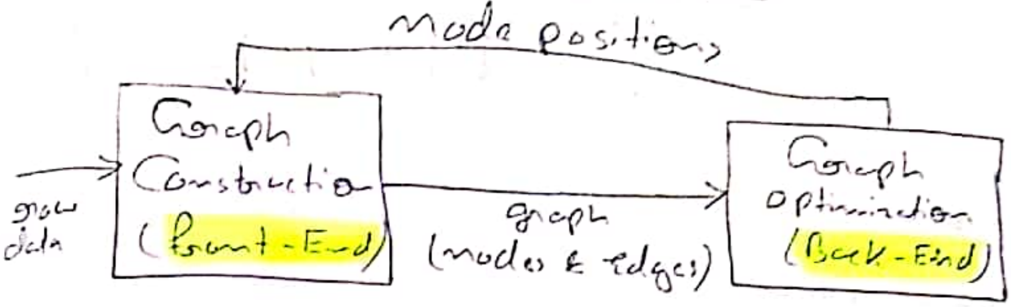
→ Every node in the graph corresponds to a pose of the robot during mapping.

→ Every edge between two nodes corresponds to a spatial constraint between them.

Graph based Slam

↳ Build the graph and find a node configuration that minimize the error introduced by the Constraints

* The Overall SLAM System



⇒ For laser scan, Iterative closest point (ICP) is the only algorithm used for data association.

* Least Squares in General

⇒ Approach for computing a solution for a Overdetermined System.

{More equations than unknowns}

⇒ Minimize the sum of the squared errors in the equation.

Wheeled Robot Locomotion

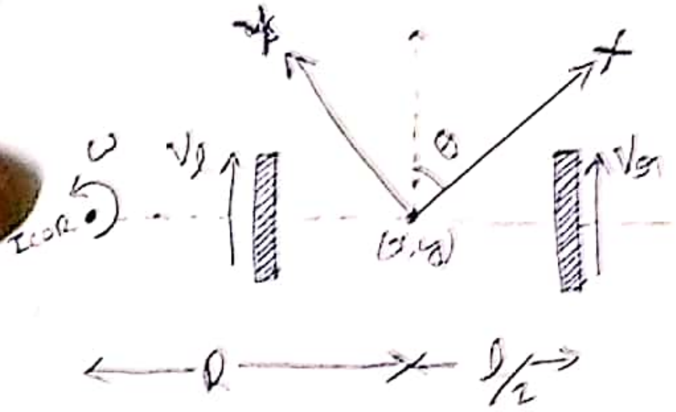
* Locomotion of Wheeled Robot

- Differential drive
- Ackerman Steering
- Synchronous drive
- XR4000
- Mecanum wheels

* Instantaneous Center of rotation (ICOR)

If wheels rotate at proper velocity then, the whole system will rotate about point O called ICOR

* Differential Drive



$$V_l = \omega\left(D - \frac{l}{2}\right) \quad \text{——} \ (1)$$
$$V_r = \omega\left(D + \frac{l}{2}\right) \quad \text{——} \ (2)$$

⇒ Using eq (1) & (2) we get:

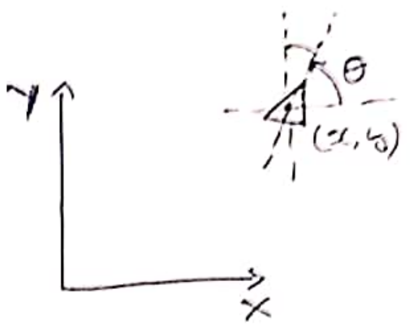
$$\boxed{R = \frac{l}{2}\,\frac{V_l + V_r}{V_r - V_l}}$$

$$\boxed{\omega = \frac{V_r - V_l}{l}}$$

$$\boxed{V = R\omega = \frac{V_r + V_l}{2}}$$

$$ICOR = [x - R\sin\theta,\ y + R\cos\theta]$$

★ **Differential drive Forward Kinematics**



⇒ Let x', y' and θ' be pose of the robot at time t

⇒ Let $V(t)$ be the linear velocity and $\omega(t)$ be the angular velocity at time t.

$$\frac{dx}{dt} = V(t)\cos\theta(t) \quad \frac{dy}{dt} = V(t)\sin\theta(t) \quad \frac{d\theta}{dt} = \omega(t)$$

$$\boxed{x' = x + \frac{1}{2}\int_0^t [V_r(t) + V_l(t)]\cos[\theta(t)]\,dt}$$

$$\boxed{y' = y + \frac{1}{2}\int_0^t [V_r(t) + V_l(t)]\sin[\theta(t)]\,dt}$$

$$\boxed{\theta' = \frac{1}{l}\int_0^t [V_r(t) - V_l(t)]\,dt + \theta}$$

★ Dead Reckoning and Odometry

use of data from motion sensors to estimate change in position over time

Path and Motion Planning

Part 1

★ Motion planning

Goals

→ Collision-free trajectories
→ Robot should reach the goal location as fast as possible.

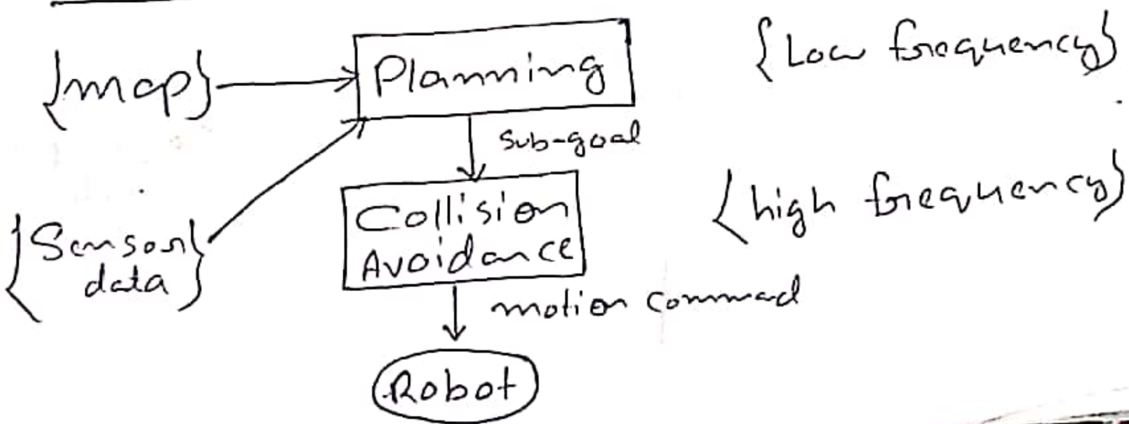
- Dynamic Window Approaches
- Grid map based planning
- Nearness Diagram Navigation
- Vector-Field-Histogram+
- A*, D*, D* Lite, ARA*, ...

★ Two Challenges

① Calculate the optimal path taking potential Uncertainities in the actions into account.

② Quickly generate actions in the case of Unforeseen objects.

★ Classic Two-layered Architecture



* Dynamic Window Approach

- Collision avoidance ⇒ Determine collision free trajectories using geometric operations.

→ Robot moves on a circular arcs.

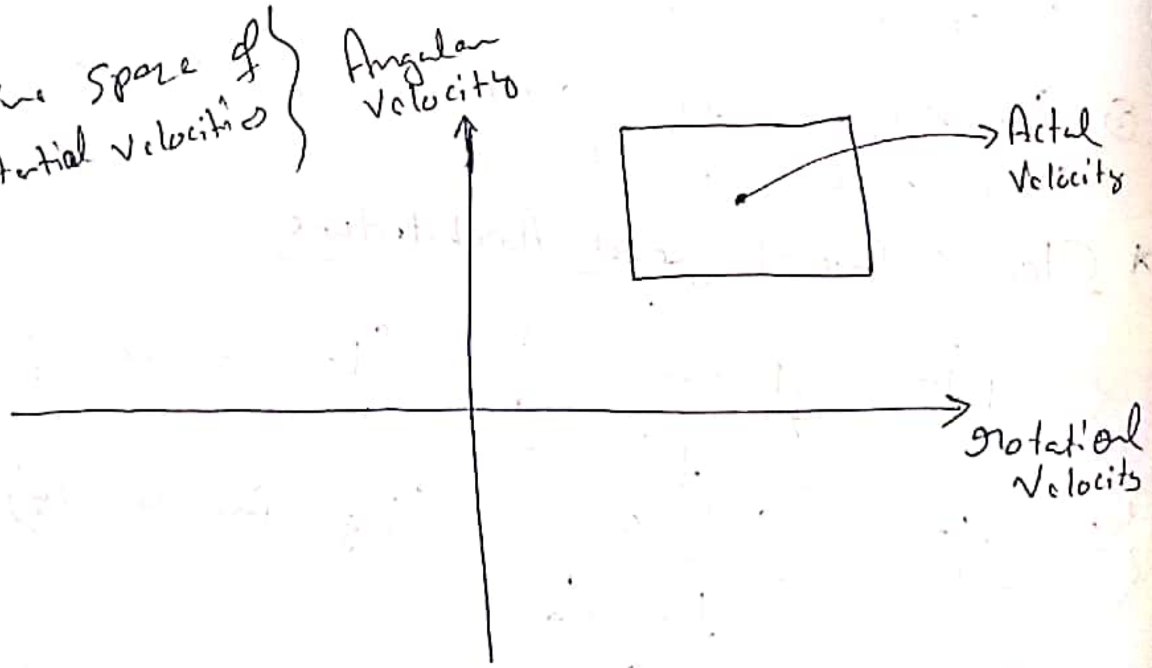
→ Motion Command (v, ω)

↳ which (v, ω) are admissible and reachable.

Speeds are admissible if the robot would be able to stop before reaching the obstacle.

$$V_a \equiv \left\{ (v, \omega) \;\middle|\; \begin{array}{l} v \leq \sqrt{2\,dist(v,\omega)\, a_{trans}} \\ \omega \leq \sqrt{2\,dist(v,\omega)\, a_{rot}} \end{array} \right\}$$

Entire space of potential velocities



$$V_d \equiv \left\{ (v,\omega) \;\middle|\; \begin{array}{l} v \in [v - a_{trans}\, t,\ v + a_{trans}\, t] \\ \omega \in [\omega - a_{rot}\, t,\ \omega + a_{rot}\, t] \end{array} \right\}$$

$V_s \equiv$ All possible speeds of the robot.

$V_a \equiv$ Obstacle free area

$V_d \equiv$ Speeds reachable within a certain time frame based on possible accelerations.

$$\boxed{V_r = V_s \cap V_a \cap V_d}$$

⇒ Steering Commands are chosen by a <u>heuristic navigation function.</u>

→ This function tries to minimize the travel time by "driving fast in the right direction"

* Navigation Function

$$NF = \alpha \cdot \underbrace{vel}_{\text{Maximize Velocity}} + \beta \cdot \underbrace{nf}_{\text{Consider cost to reach goal}} + \gamma \cdot \underbrace{\Delta nf}_{\text{Follows grid based Path computed by A*}} + \delta \cdot \underbrace{goal}_{\text{Goal nearness}}$$

→ Reacts quickly
→ Low CPU power requirements
→ Guides a robot on a collision-free path
→ Successfully used in a lot of real-world scenarios.
→ Resulting trajectories sometime suboptimal
→ Local minima might prevent the robot from reaching the goal location.

* Motion planning Formulation

⇒ The problem of motion planning can be stated as follows.

Given
→ A start pose of the robot.
→ A desired goal pose.
→ A geometric description of the robot.
→ A geometric description of the world.

To find
→ Path that moves the robot gradually from start and goal while never touching any obstacle.

* Configuration Space

⇒ A robot configuration q is a specification of the positions of all robot points relative to a fixed coordinate system.

⇒ Usually a configuration is expressed as a vector of positions and orientations.

Free Space

Obstacle region

$\Rightarrow$ With $W = \mathbb{R}^m$ being the work space, $O \in W$ the set of obstacles, $A(q)$ the robot in configuration $q \in C$.

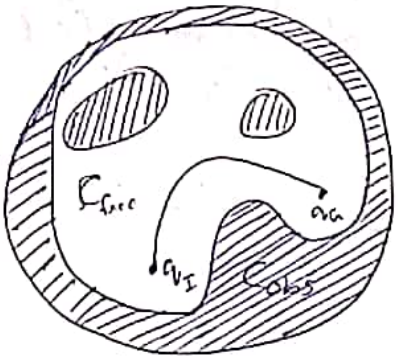
$$C_{free} = \{ q \in C \mid A(q) \cap O = \phi \}$$

$$C_{obs} = C / C_{free}$$

$\Rightarrow$ We further define

q_I : Start Configuration

q_G : Goal Configuration



$\Rightarrow$ Then, motion planning amounts to

- Finding a continuous path

  $\tau : [0,1] \rightarrow C_{free}$

  with $\tau(0) = q_I$, $\tau(1) = q_G$

★ C-Space Discretizations

$\Rightarrow$ Continuous terrain needs to be discretized for path planning.

$\Rightarrow$ There are two general approachs to descritize C-spaces:

① Combinatorial Planning

Characterizing C_{free} explicitely by Capturing the Connectivity of C_{free} into a Graph and find Solution using Search.

② Sampling-based planning

Use collision-detection to probe and ~~[illegible]~~ incrementally search the C-space for solution.

★ Search

⇒ The problem of search: finding a sequence of actions (a path) that leads to desirable state (goal).

Example Algorithm: breadth-first, Uniform cost, depth-first, bidirectional etc.

Uninformed Search

{ besides the problem definition, no further information about the domain }

Informed Search

{ further information about the domain through heuristics }

Example algorithms:

greedy best-first search, A*, many variants of A*, D* etc...

⇒ The performance of search algorithm is measured in 4 ways.

① Completeness { Does the algorithm find the solution when there is one. }

② Optimality {Is the solution best one of all possible solutions in terms of path cost?}

③ Time Complexity

④ Space Complexity

* **Informed Search: A***

→ Finds the shortest path.

→ Requires a graph structure.

→ Limited number of edges.

→ In robotics: Planning on a 2d occupancy gridmap.

⇒ To compute the shortest path from every state to one goal state, use deterministic value iteration.

⇒ Very similar to Dijkstra's Algorithm.

⇒ Such a cost distribution is the optimal heuristic for A*.

* **Typical Assumption in Robotics for A* path Planning**

① The robot is assumed to be localized.

② The robot computes its path based on an occupancy grid.

③ The correct motion commands are executed.

★ Problems

1. What if the robot is (slightly) delocalized?
2. Moving on the shortest path often guides the robot along a trajectory close to obstacles.
3. Trajectory aligned to the grid structure.

★ Convolution of the Grid Map

⇒ Convolution blurs the Map.

⇒ Obstacles are assumed to be bigger than in reality.

⇒ Perform an A* Search in such a convolved map (Using Occupancy as traversal cost)

⇒ Robot increases distance to obstacles and moves on a short path.

Example ⇒ Gaussian blur.

★ A* in Convolved Maps

⇒ The Costs are a product of path length and occupancy probability of the cells.

⇒ Cells with higher probability are avoided by the robot.
↳ Thus it keeps distance to obstacles.

⇒ This technique is fast and quite reliable

Scanned by CamScanner

* 5D-Planning - an Alternative to the Two-layered Architecture

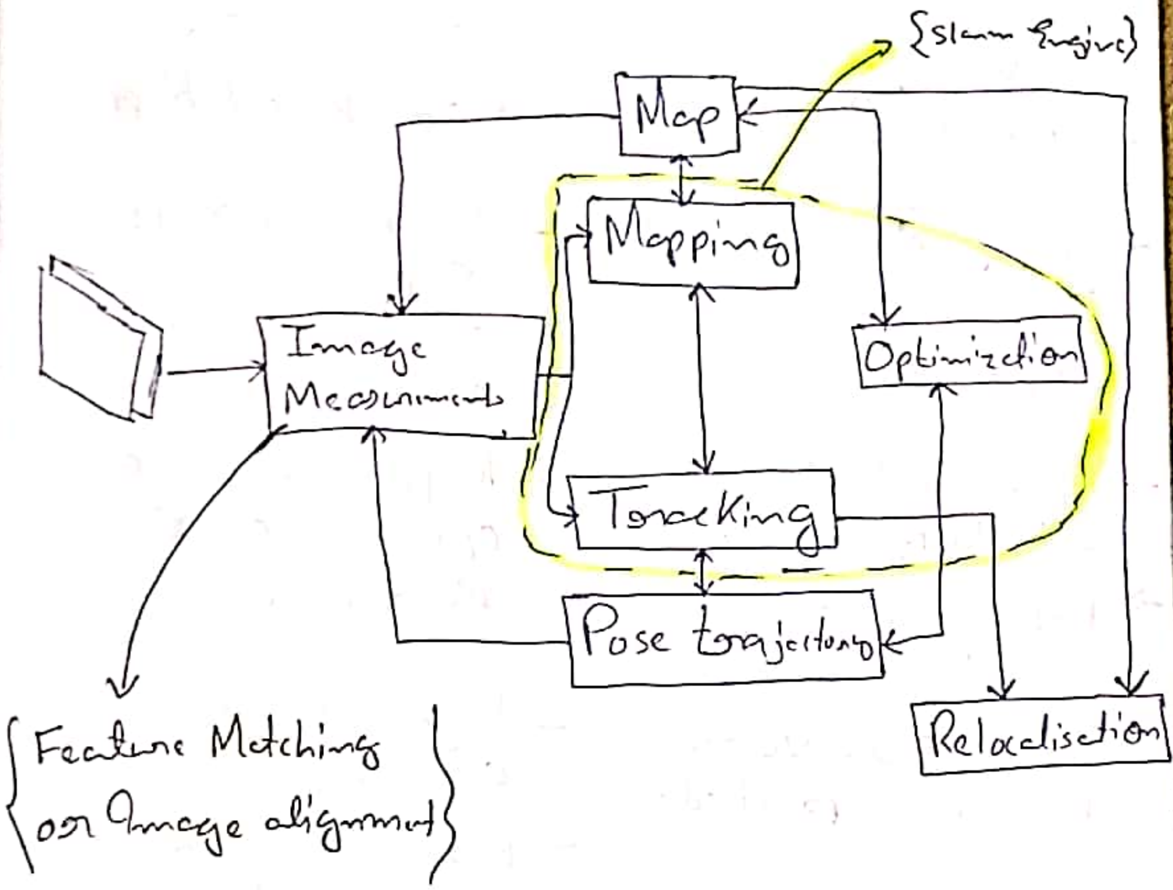
⇒ Plans in the full $\langle x, y, \theta, v, \omega \rangle$ Configuration Space using A*.

↳ Considers the robot's Kinematic Constraints.

⇒ Generates a Sequence of Steering Commands to reach the goal location.

⇒ Maximizes trade-off between driving time and distance to obstacles.

Robust and Efficient Visual SLAM



* Image measurements

① **Point features**

→ Sparse map consisting of 3-D points

② **Image alignment**

→ dense 3-D map, eg Volumetric grid.

⇒ Feature matching with descriptors

Example: SIFT, BRISK, ORB etc.

Slam Engine

Probablistic filtering

→ Tightly coupled tracking & mapping.

EKF, PF

→ Good for local mapping with loopy motion.

→ Performance degrades over large areas

→ Large state vectors become impractical.

Tracking & Mapping

→ Tracking & mapping separate.

→ Tracking efficitively visual odometry.

→ Mapping based on optimization over key frame.

→ Higher accuracy, inc. over large areas.

→ Allows dense mapping as well as feature based.

→ but computationally demanding.

(a) Mono SLAM. (EKF Slam)

(b) PTAM DTAM

(c) ORB SLAM

(d) LSD - SLAM

(e) RGBD SLAM

a → Scale predicted feature matching.