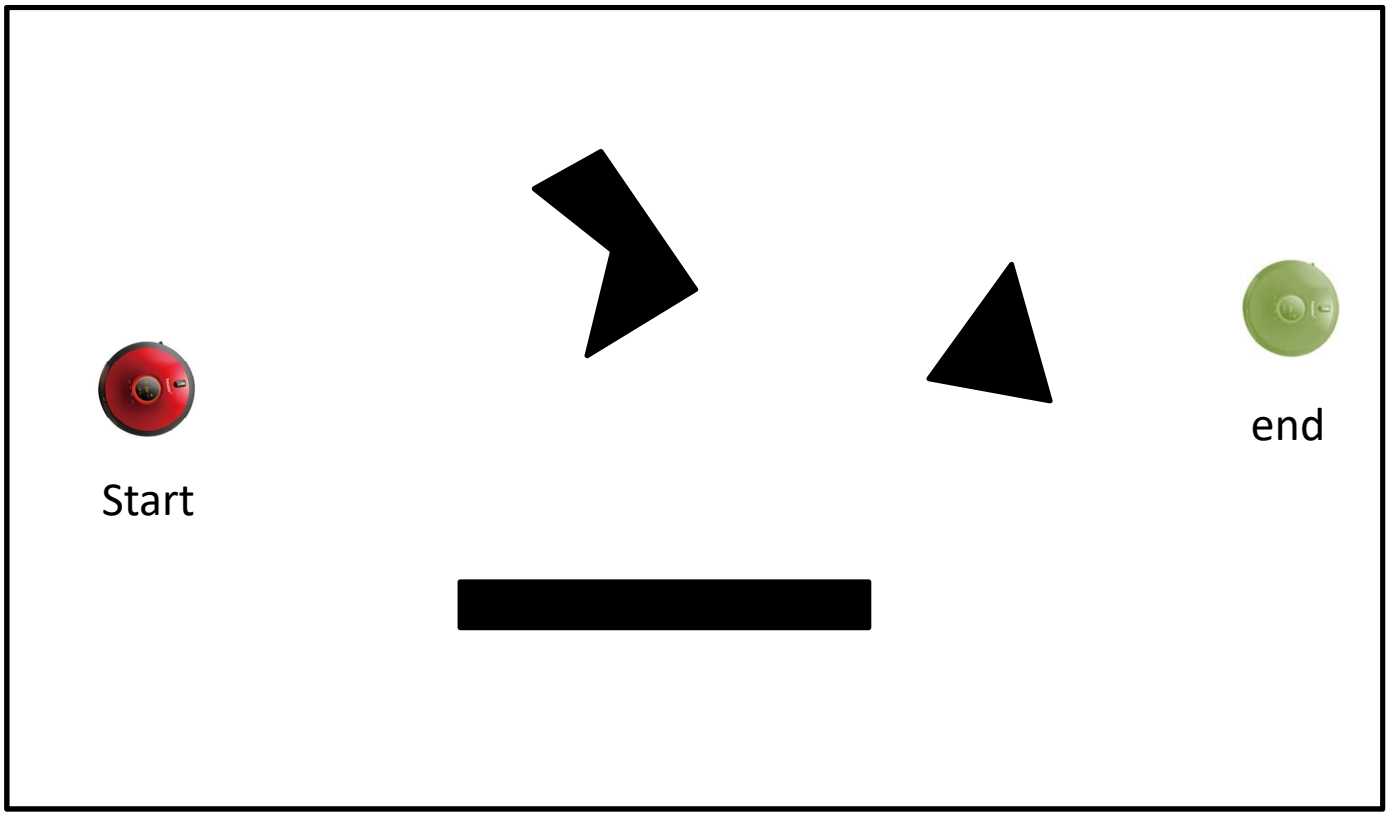
Rapid exploration Random Trees (RRT)

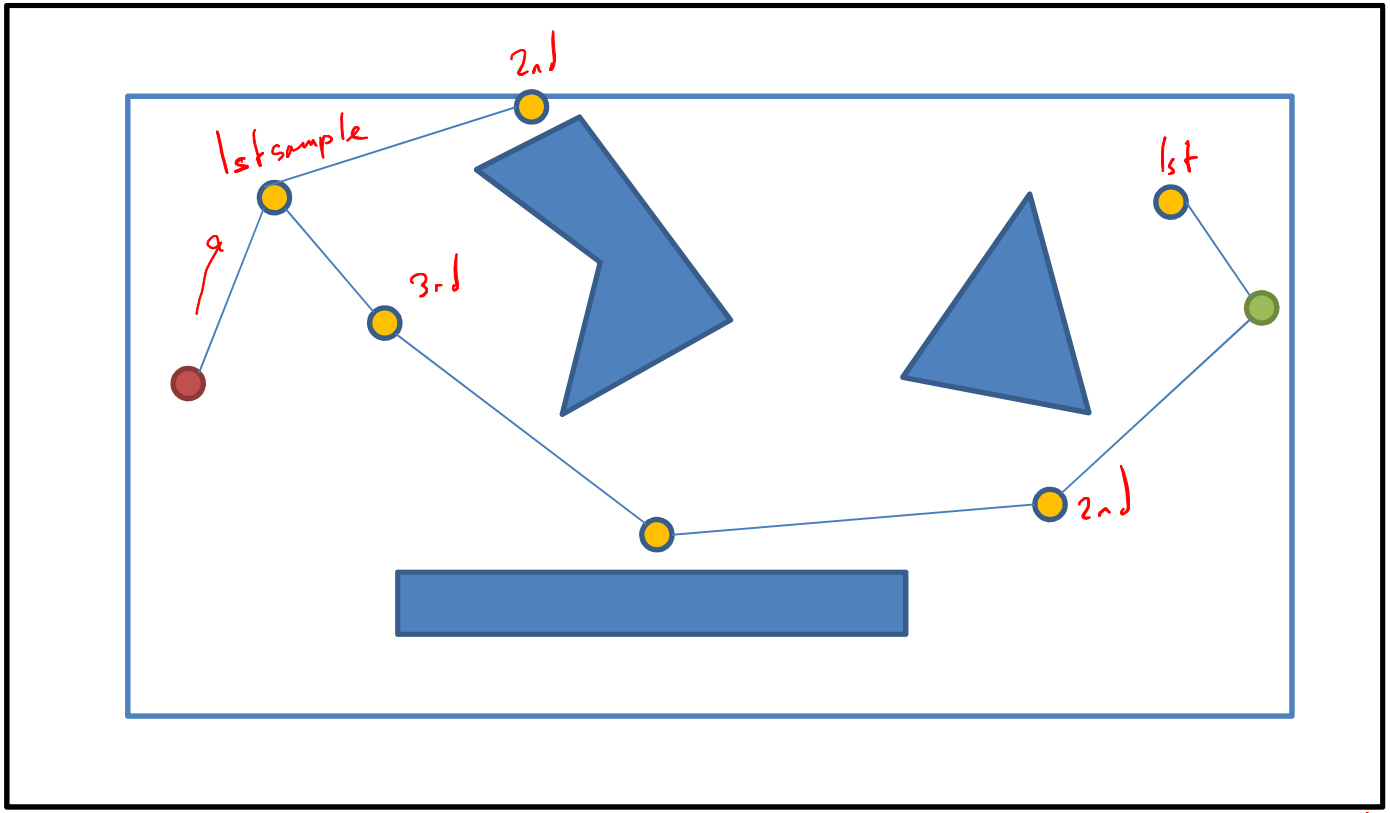
- Build a tree through generating “next states” in the tree by executing random controls.
- Can be combined with A* for efficiency (RRT*).
- Can incorporate kinematic constraints more easily.

Sampling-based planning and RRTs



Sampling-based planning and RRTs

Sample point in vicinity of start/end locations, and connect them to closest point in tree

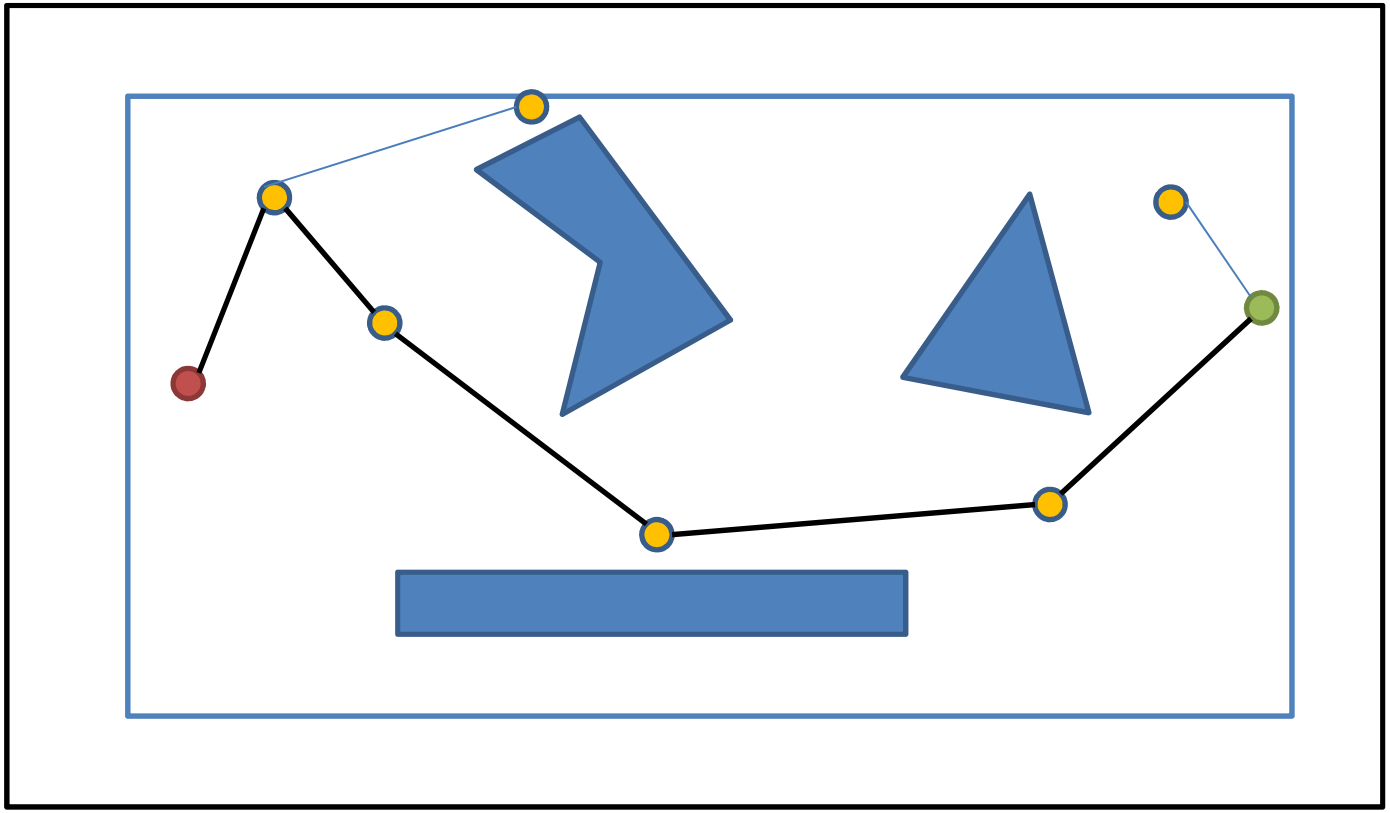


Use Configuration Space!

Generate 'trees' from both start and goal.

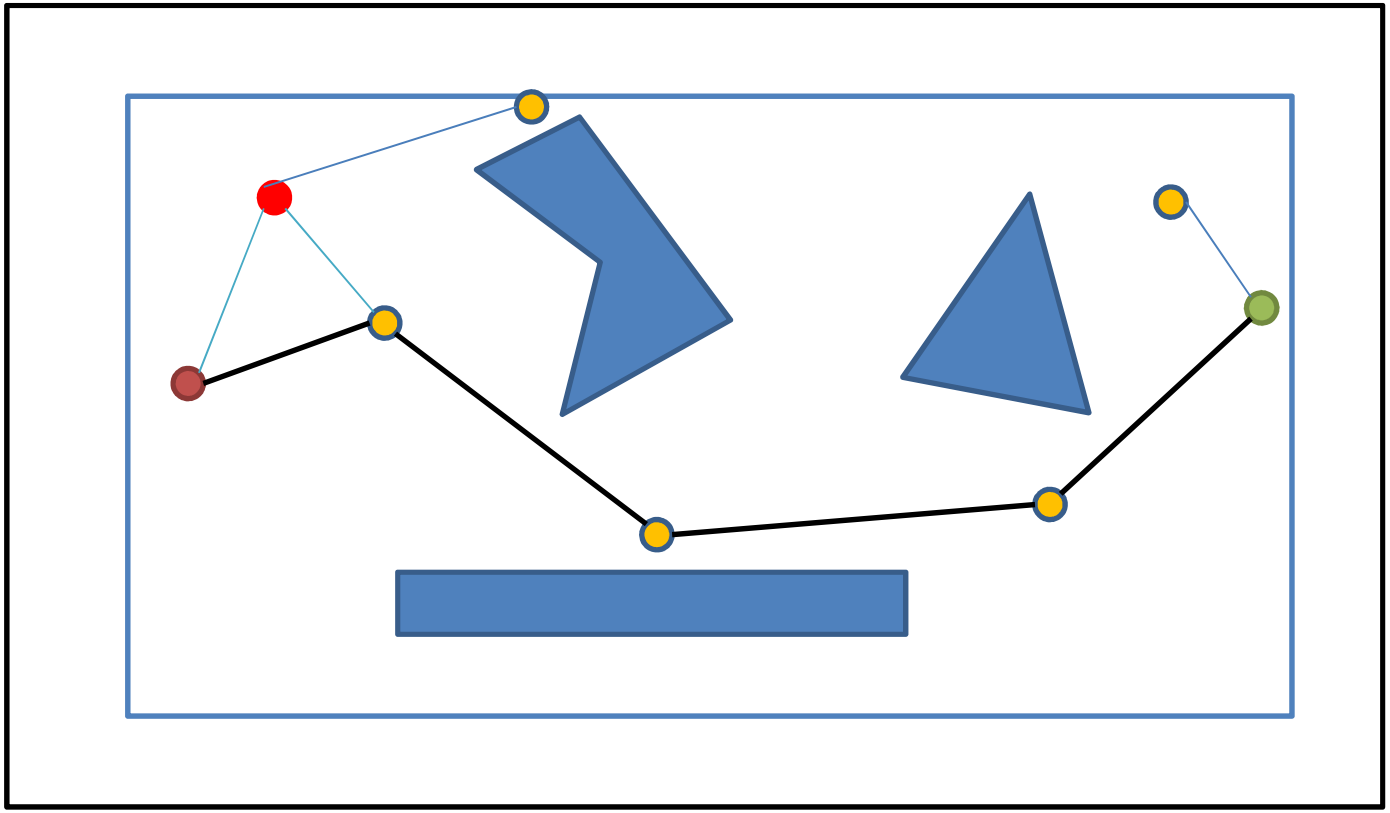
Sampling-based planning and RRTs

Find a non-collision path from start to goal via random but achievable next steps



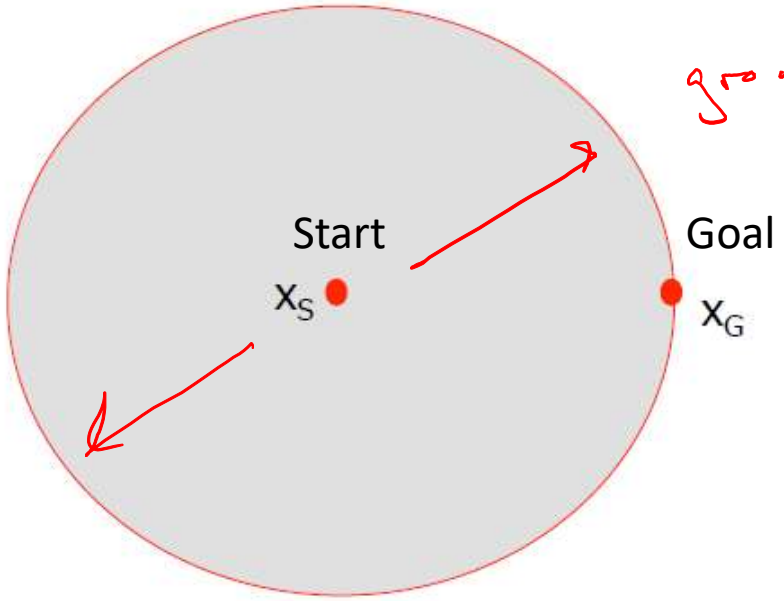
Sampling-based planning and RRTs

May be possible to apply some smoothing eg by looking at a point to see if it can be removed



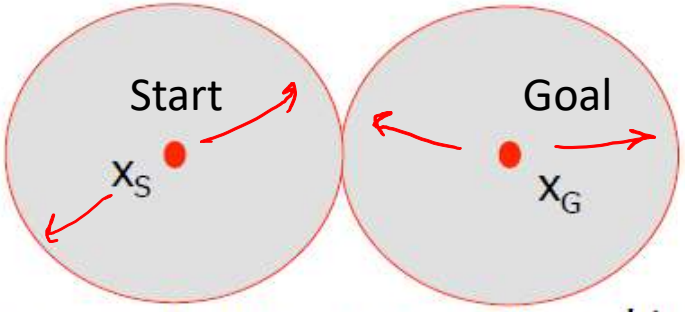
Uni vs Bi-directional exploration

- Volume swept out by unidirectional RRT:



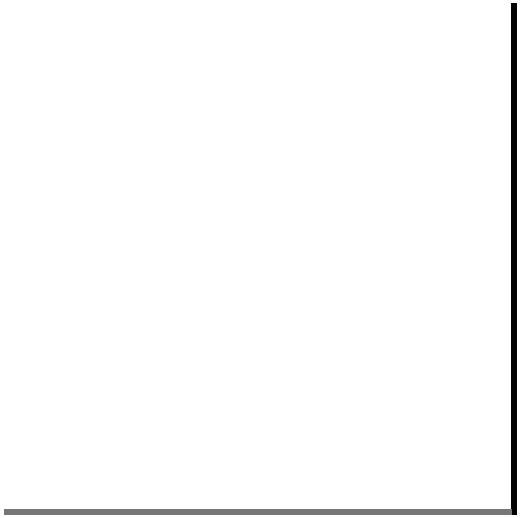
grow in the spaces.

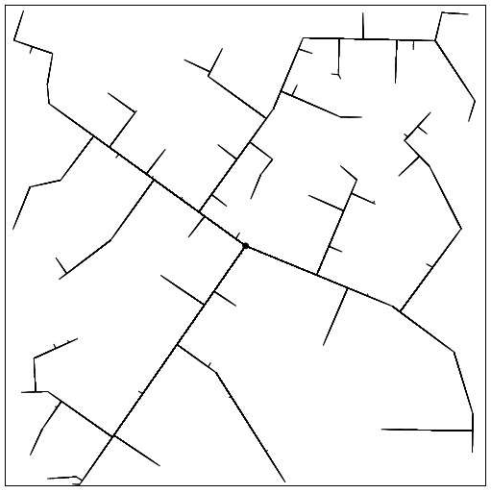
- Volume swept out by bi-directional RRT:



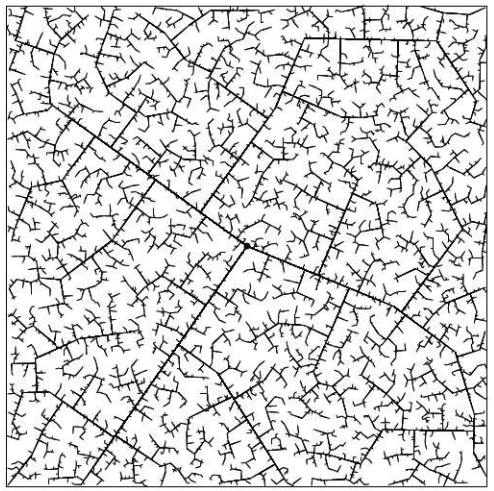
- Difference becomes even more pronounced in higher dimensions

There can be dense-covering strategies





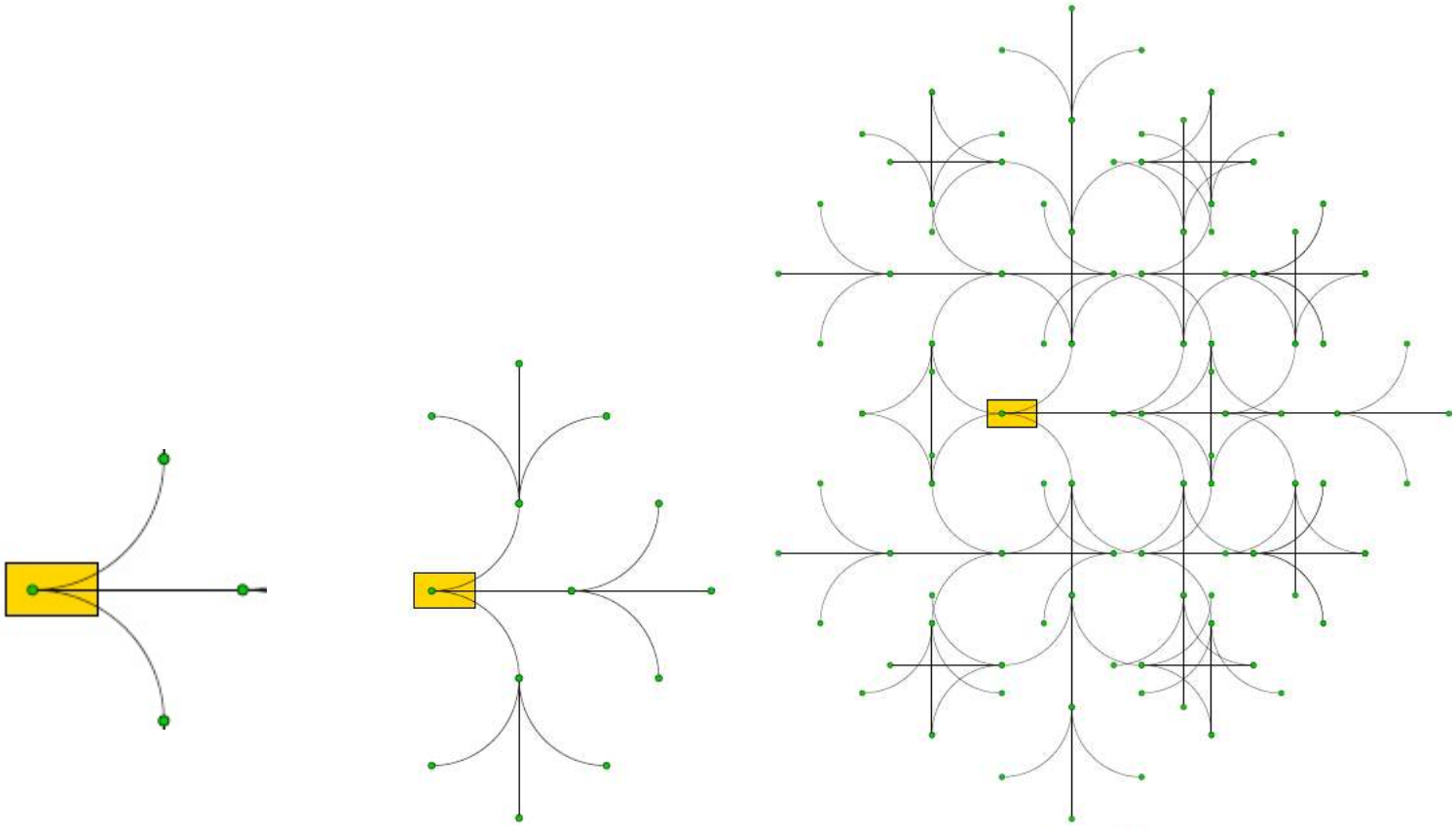
45 iterations



2345 iterations

From Lavalle's book

And incorporation of kinematics



One stage

Two stages

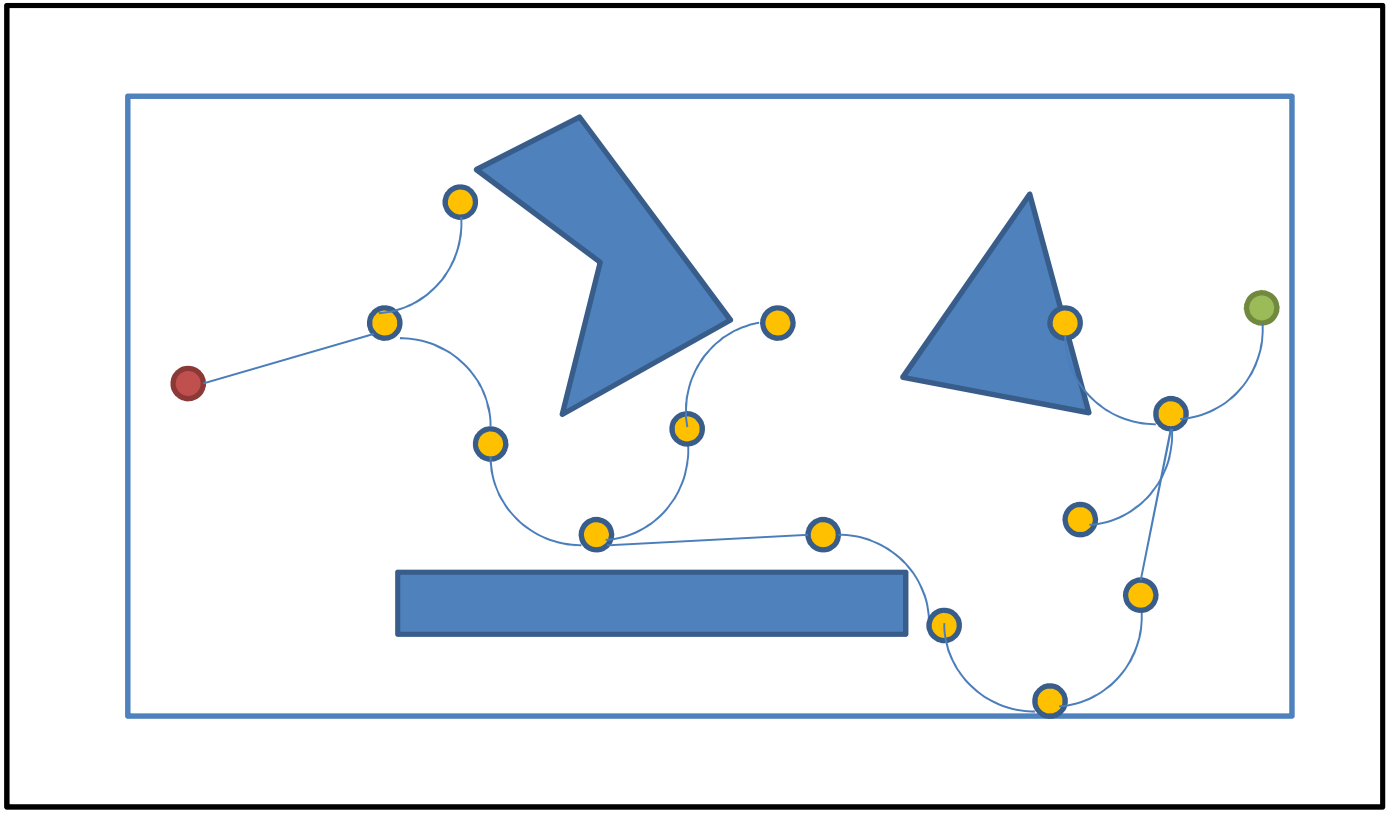
Four stages

Figure 14.6: A reachability tree for the Dubins car with three actions. The kth stage produces 3^k new vertices.

From Lavalle's book

Sampling-based planning and RRTs

Connections to the tree can use kinematic constraints



Holonomicity

- Degrees of freedom (DOF): number of independent parameters that define the displacement and deformation (configuration) of the body in a robot.

- Holonomic: if the robot can move instantaneously in all the degrees of freedom where it operates. E.g. a train has 1 motor and moves in a 1D space.

- Non-holonomic: An arbitrary configuration is not achieved immediately but through time. E.g. a twin motorized wheeled robot moving on a flat surface (which is a 3D space: x,y,theta).

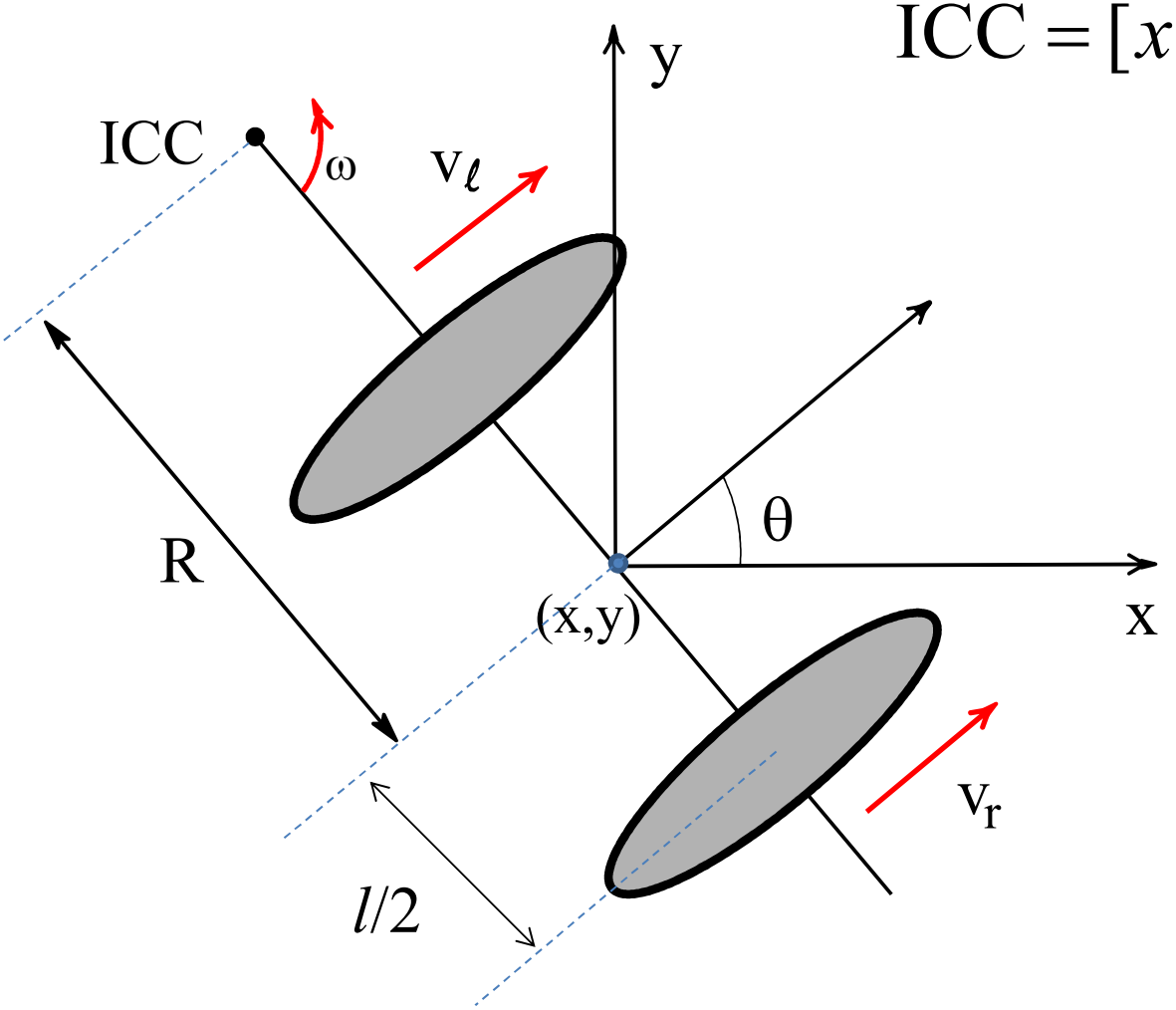
Holonomic platforms do exist

Segway OMNI

$$v_y = (v_0 + v_1 + v_2 + v_3)/4$$
$$v_x = (v_0 - v_1 + v_2 - v_3)/4$$
$$v_\theta = (v_0 + v_1 - v_2 - v_3)/4$$
$$v_{error} = (v_0 - v_1 - v_2 + v_3)/4$$

Complex hardware and relies on certain assumptions

Differential Drive



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

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

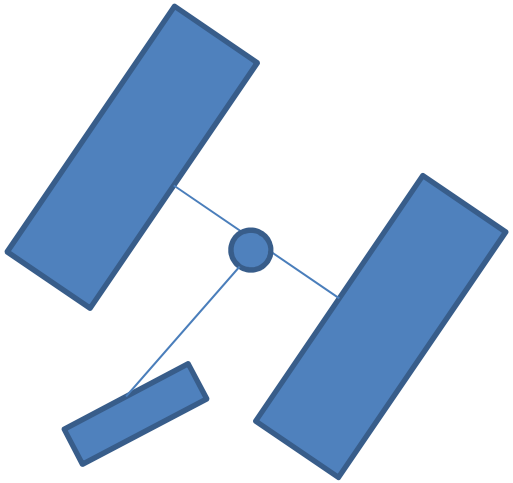
$$\omega(R - l/2) = v_l$$

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

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

ICC=Instantaneous centre of curvature

Differential Drive



Castor wheel

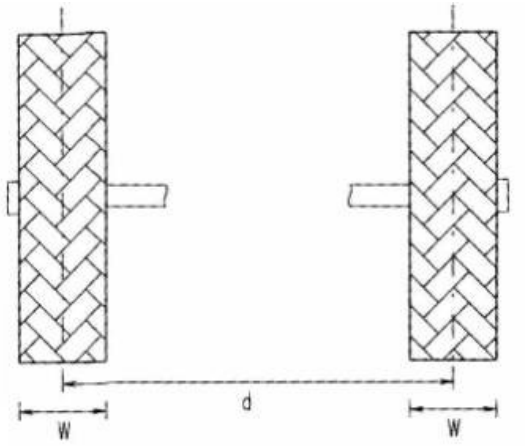
Most lightweight differential drive configurations use extra contact points or non-actuated wheels. These can often be ignored from the calculations.

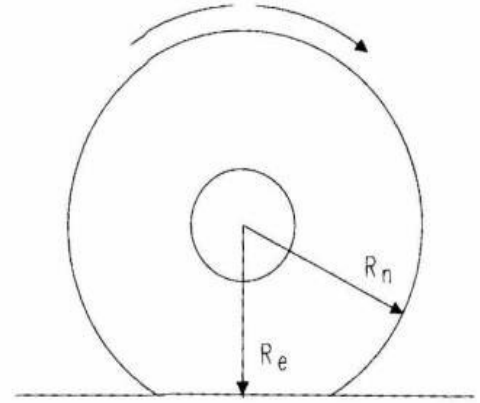
Castor wheel

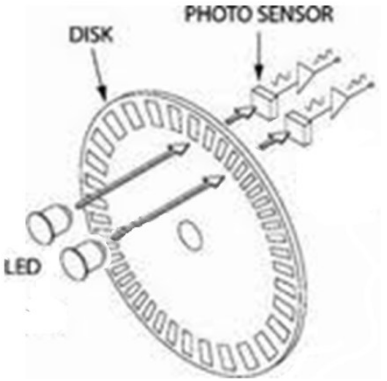
Most popular configuration for cheap and small robots.
Advantages: Easy to build, Simple design.

Disadvantages: Difficult straight line motion.

Problems with Differential Drive



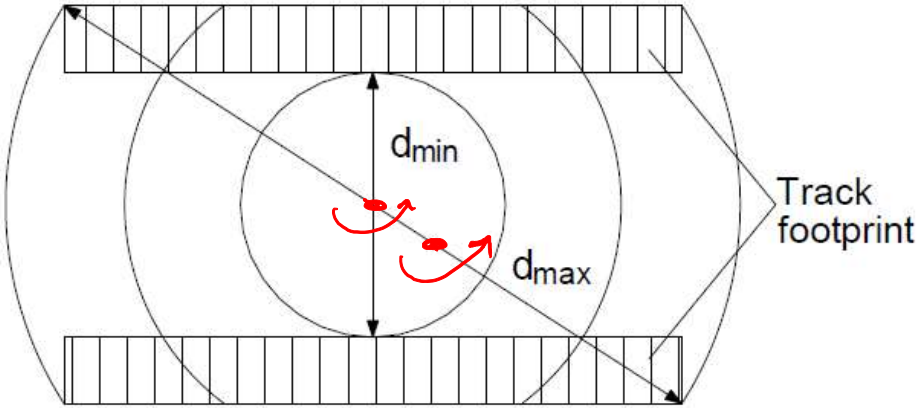




- Changing diameter makes for uncertainty in dead-reckoning
- If serious, odometry as sensed at shaft, wont be reliable in the long run

Some Pictures from “Navigating Mobile Robots: Systems and Techniques” Borenstein, J.

Skid Steering



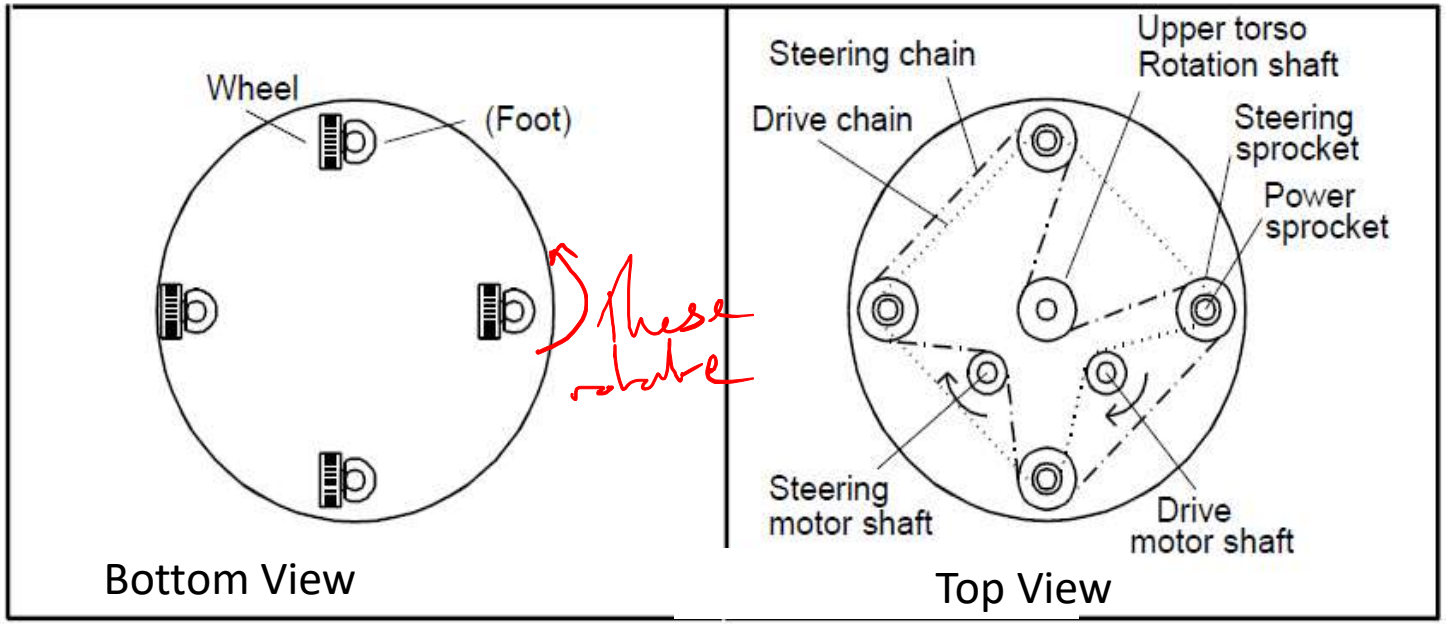
The effective point of contact for a skid-steer vehicle is roughly constrained on either side by a rectangular zone of ambiguity corresponding to the track footprint. As is implied by the concentric circles, considerable slippage must occur in order for the vehicle to turn [Everett, 1995]

In laymans: centre of rotation could be anywhere in the above circles

Uses larger amount of energy to turn than differential drive.

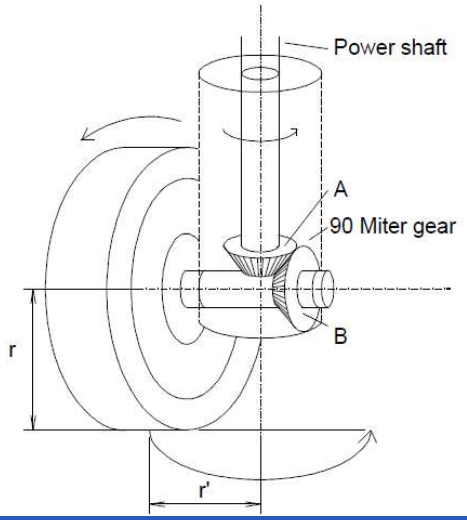
Some Pictures from "Where am I" Borenstein, Everett and Feng

Synchro Drive



- Improved odometry as all wheels produce more equal and parallel force vectors
- Better straight line motion
- More difficult to build though
- Note, its still a non-holonomic system

Belts are problematic → change length due to heat etc.



Pictures from "Where am I" .Borenstein, Everett and Feng

Drive configurations

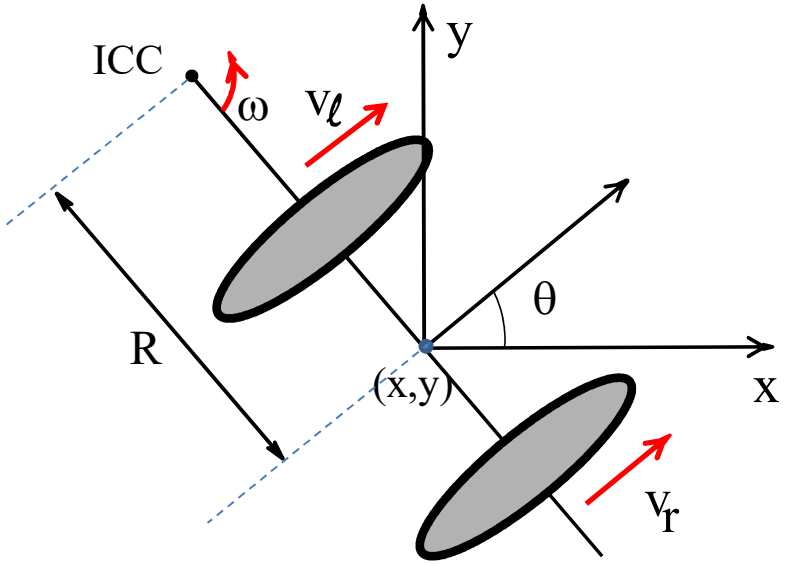
- Synchro drive is more accurate (on flat terrain) but complex to build.
- Tracks are good in some respects (tanks work in tough terrains!) but less good for odeometry.
- Differential drive is easier to build and assuming wheels and alignments are not so different should provide usable odometry over short periods of time.
- Recall: odometry should only be trusted on *short* periods of time before a more reliable outward-looking sensor resets position.

Other options

- Legs => keep centre of mass under control!
- Snake-like motion => manage multiple DOF.
- Flying vehicles => manage drag and lift, ground effects .
- Etc.

Differential Drive Forward Kinematics

If the robot starts form state (x,y,θ) and has ground velocities Vℓ and Vr during the period of time δt then:



$$\mathrm{ICC} = (x - \mathrm{R}\sin(\theta), y + \mathrm{R}\cos(\theta))$$

Its new state at t+δt is defined by:

$$\begin{pmatrix} x^{new} \\ y^{new} \\ \Theta^{new} \end{pmatrix} = \begin{pmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x - \mathrm{ICC}_x \\ y - \mathrm{ICC}_y \\ \Theta \end{pmatrix} + \begin{pmatrix} \mathrm{ICC}_x \\ \mathrm{ICC}_y \\ \omega\delta t \end{pmatrix}$$

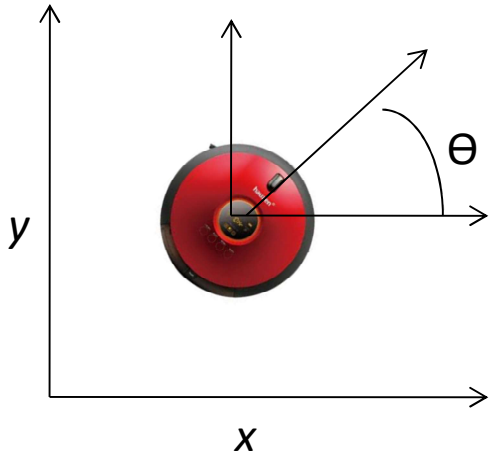
Recall previous slide for expressions of R and ω

Use either method -

If a rotate-translate-rotate strategy

simplifies control at the price of motion efficiency

Robot's *state* vector => $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$



After a pure translation t:

$$\begin{pmatrix} x^{new} \\ y^{new} \\ \Theta^{new} \end{pmatrix} = \begin{pmatrix} x + t\cos\Theta \\ y + t\sin\Theta \\ \Theta \end{pmatrix}$$

After a rotation on the spot by α :

$$\begin{pmatrix} x^{new} \\ y^{new} \\ \Theta^{new} \end{pmatrix} = \begin{pmatrix} x \\ y \\ \Theta + \alpha \end{pmatrix}$$

This approach can also be used to simplify the *inverse* kinematics problem for such a robot!

Summary

- There are several approaches for taking a robot from one point to another.
- There is no overall elegant solution, most people use combinations of reactive (bug-like methods) and planned (exhaustive/sampled search).
- Its possible to have a basic motion planning strategy with little information of the environment, or with a lot.
- We did not consider dynamic obstacles, or multiple degrees of freedom robots e.g. Arms but these are covered elsewhere (Kinematics/3D graphics units).

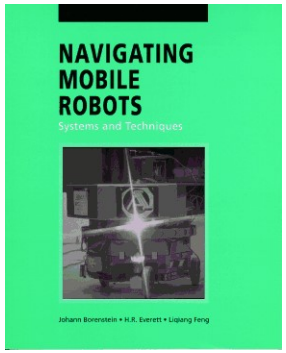
To read Further

- Online book:

Navigating Mobile Robots,

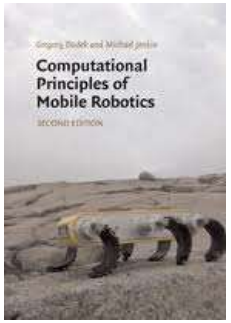
J. Borenstein, H. R. Everett, and L. Feng

http://www-personal.umich.edu/~johannb/my_book.htm



- Book:

Computational Principles of Mobile Robotics, G. Dudek and M. Jenkin, Second Edition.
(chapters 3 and 6)



- Online Book: Planning Algorithms, S. Lavalle.

http://planning.cs.uiuc.edu/booka4.pdf