Wheeled mobile robots

Description, Kinematics, Modeling, Planning

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA
ROMANIA

December 14, 2023

Agenda

- Types of wheels and wheeled robots
- Moving around
- Kinematics, modeling
- Navigation, planning

Why wheeled robots?

Why are wheeled robots useful?
Provide some examples of applications

Wheels

Types of wheels

Wheels

Types of wheels

- Fixed wheel

Wheels

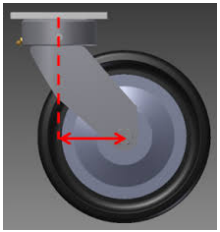
Types of wheels

- Fixed wheel
- Centered wheel

Wheels

Types of wheels

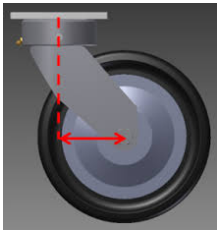
- Fixed wheel
- Centered wheel
- Off-centered wheel



Wheels

Types of wheels

- Fixed wheel
- Centered wheel
- Off-centered wheel
- Omni wheel

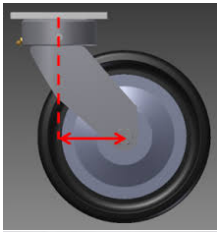


Wheels

Types of wheels

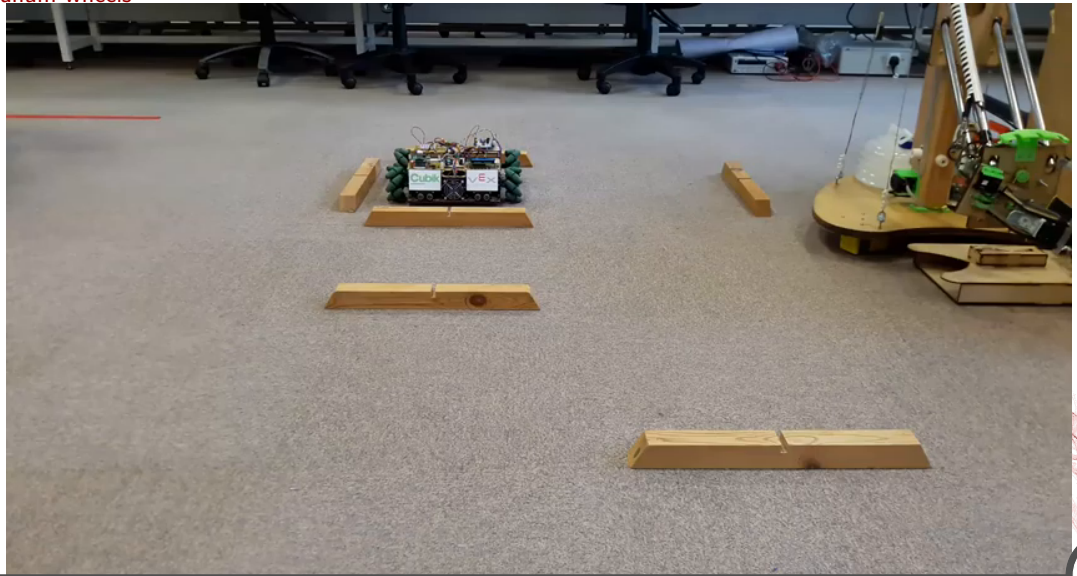
- Fixed wheel
- Centered wheel
- Off-centered wheel
- Omni wheel
- Mecanum wheel





Wheels

Mecanum wheels



Wheeled robots are categorized based on the type of wheels and configurations that they use.

- By-wheel
- Tricycle
- Four wheel
- Omnidirectional
- etc. etc.

Kinematics

Kinematics

Description of robot pose in a inertial frame

Kinematics

Description of robot pose in a inertial frame

Pose

Kinematics

Description of robot pose in a inertial frame

Pose

Position

Kinematics

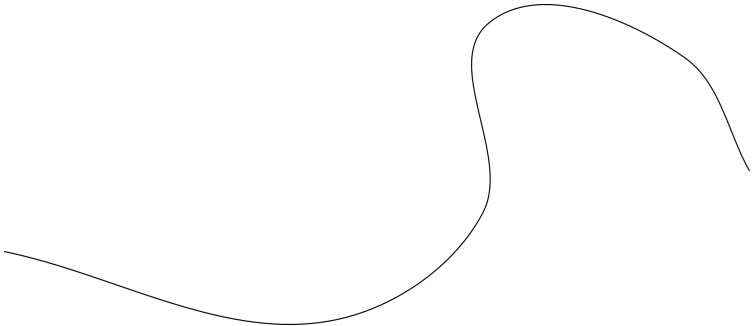
Description of robot pose in a inertial frame

Pose

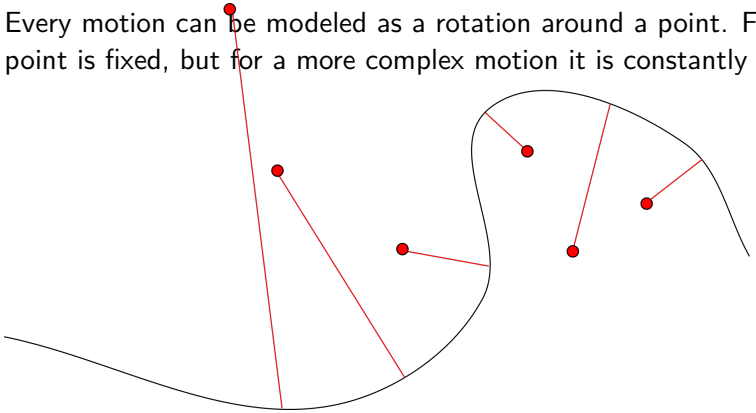
Position
Orientation

Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.

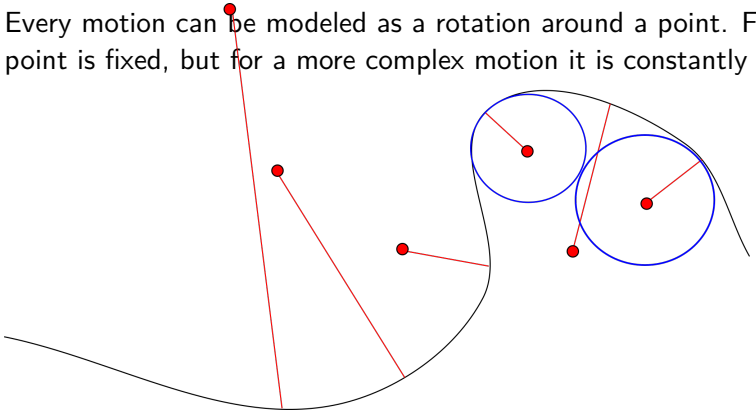
Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.



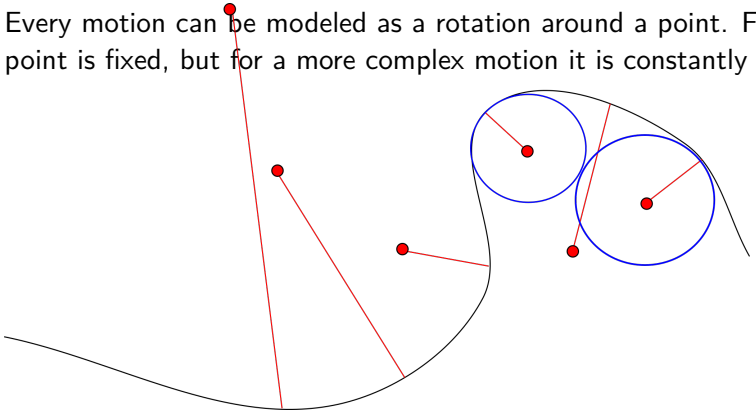
Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.



Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.



Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.



Where is the ICR for straight motion?

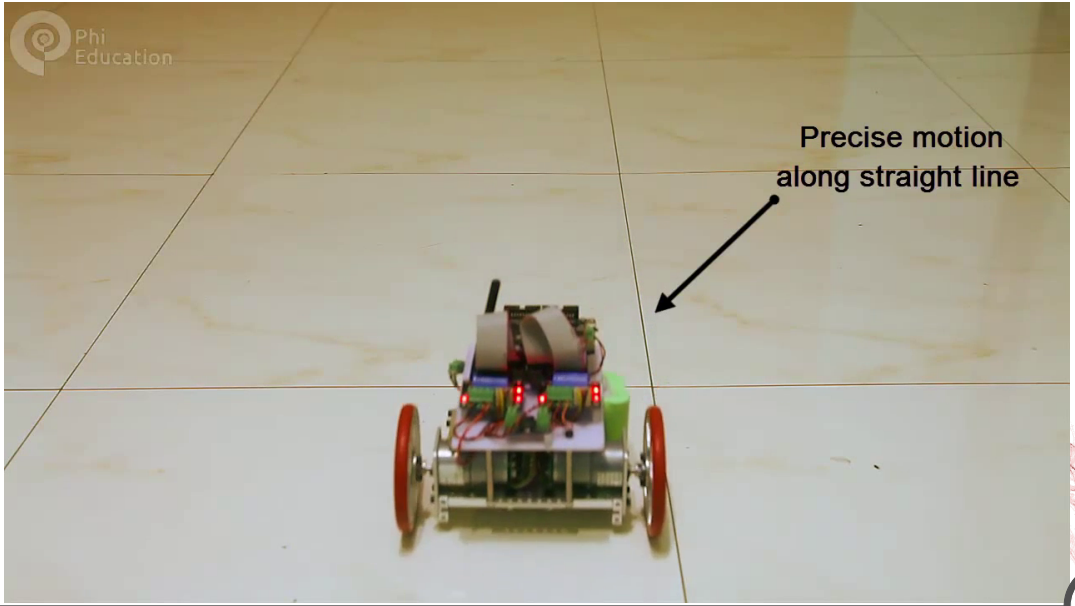
Wheeled robots

Differential drive

The differential drive is implemented:

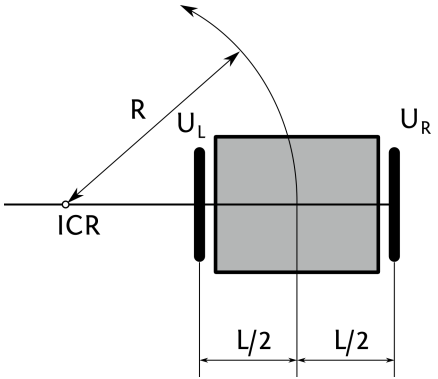
- Two driving wheels
- Each can rotate independently
- Need for a third balancing point (usually a roller-ball)
- Sensitive to relative velocity of the two wheels
- No sliding!

Differential drive



Differential drive

Kinematics modeling

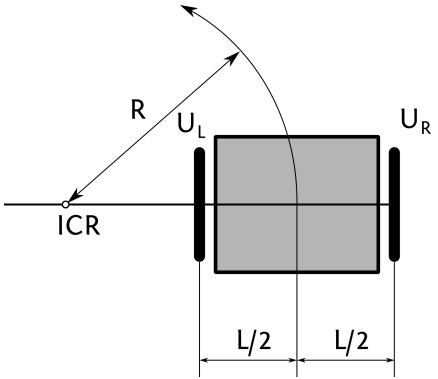


Definitions:

Ω: Angular velocity of the robot
ω_i: Angular velocity of wheel i
U: Linear velocity of the robot
u_i: Linear velocity of wheel i
r: nominal radius of each wheel
R: Instantaneous Curvature Radius

Differential drive

Kinematics modeling



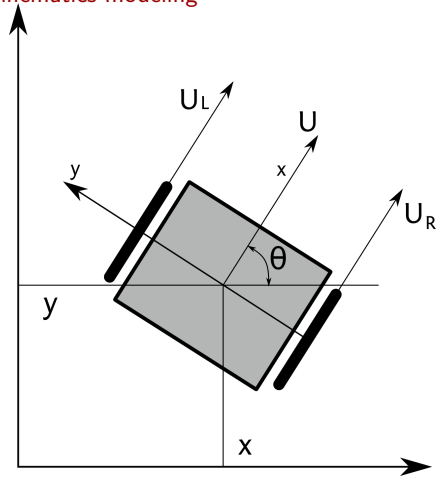
Definitions:

Ω: Angular velocity of the robot
ω_i: Angular velocity of wheel i
U: Linear velocity of the robot
u_i: Linear velocity of wheel i
r: nominal radius of each wheel
R: Instantaneous Curvature Radius

How many degrees of freedom?

Differential drive

Kinematics modeling



Pose of the robot

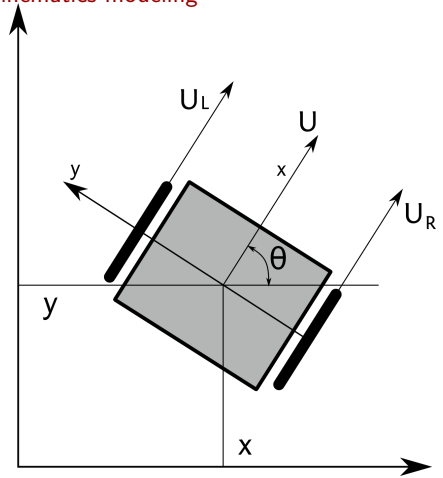
$$P = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Control input

$$U = \begin{bmatrix} U \\ \Omega \end{bmatrix}$$

Differential drive

Kinematics modeling



Pose of the robot

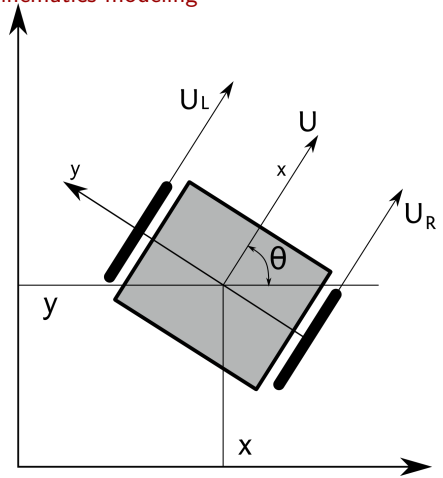
$$P = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Control input

$$U = \begin{bmatrix} U \\ \Omega \end{bmatrix} \text{ or } U = \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}$$

Differential drive

Kinematics modeling



Pose of the robot

$$P = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

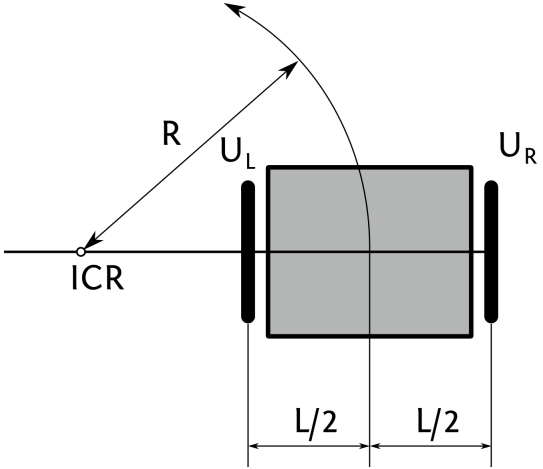
Control input

$$U = \begin{bmatrix} U \\ \Omega \end{bmatrix} \text{ or } U = \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}$$

If we want to follow a specific trajectory (i.e. a specific R), the wheels must move in such rate so they rotate around the ICR with the same angular velocity

Differential drive

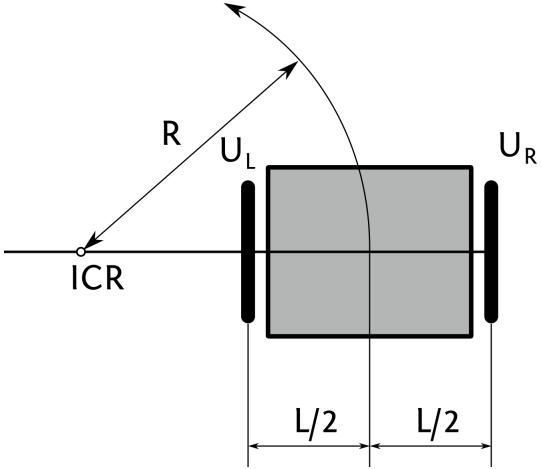
Forward kinematics modeling



$$\Omega = \frac{u_R}{R + \frac{L}{2}} = \frac{u_L}{R - \frac{L}{2}}$$

Differential drive

Forward kinematics modeling

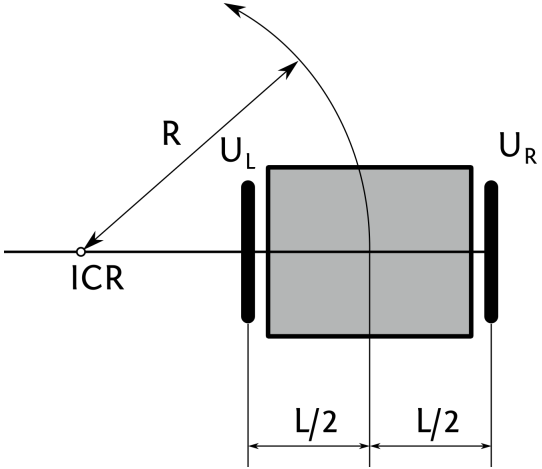


$$\Omega = \frac{u_R}{R + \frac{L}{2}} = \frac{u_L}{R - \frac{L}{2}}$$

$$R = \frac{L}{2} \frac{u_R + u_L}{u_R - u_L}$$

Differential drive

Forward kinematics modeling



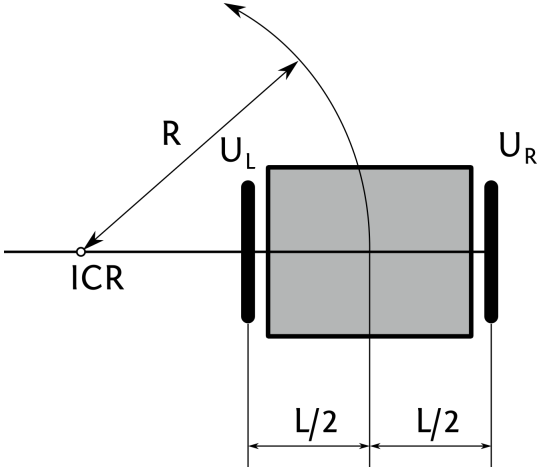
$$\Omega = \frac{u_R}{R + \frac{L}{2}} = \frac{u_L}{R - \frac{L}{2}}$$

$$R = \frac{L}{2} \frac{u_R + u_L}{u_R - u_L}$$

System of two equations with two unknowns

Differential drive

Forward kinematics modeling



$$\Omega = \frac{u_R}{R + \frac{L}{2}} = \frac{u_L}{R - \frac{L}{2}}$$

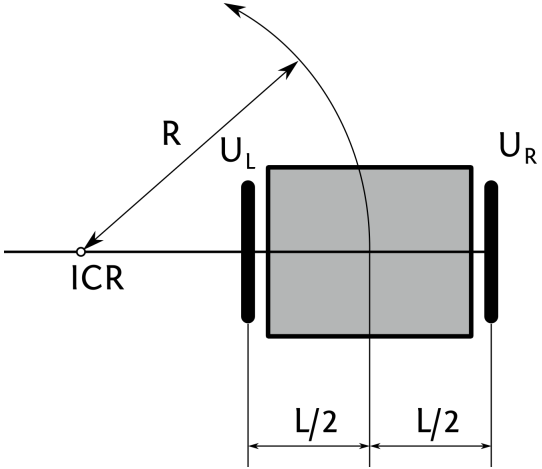
$$R = \frac{L}{2} \frac{u_R + u_L}{u_R - u_L}$$

System of two equations with two unknowns

We determine either ω_L and ω_R, or Ω and U

Differential drive

Forward kinematics modeling

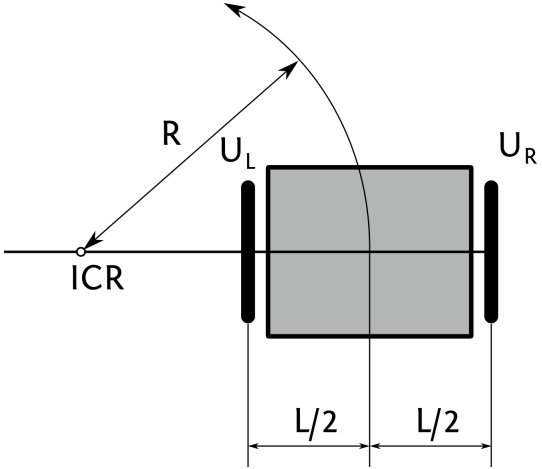


We can observe:

$$U = \frac{u_R + u_L}{2} = \frac{(\omega_R + \omega_L)r}{2}$$

Differential drive

Forward kinematics modeling



We can observe:

$$U = \frac{u_R + u_L}{2} = \frac{(\omega_R + \omega_L)r}{2}$$

Knowing that:

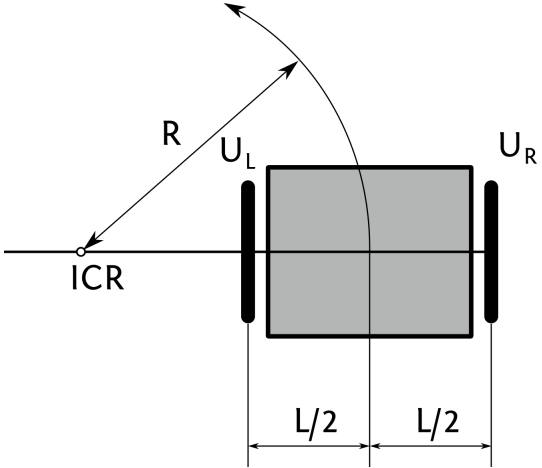
$$R = \frac{L}{2}\frac{u_R + u_L}{u_R - u_L}$$

and:

$$U = \Omega R$$

Differential drive

Forward kinematics modeling



We can observe:

$$U = \frac{u_R + u_L}{2} = \frac{(\omega_R + \omega_L)r}{2}$$

Knowing that:

$$R = \frac{L}{2}\frac{u_R + u_L}{u_R - u_L}$$

and:

$$U = \Omega R$$

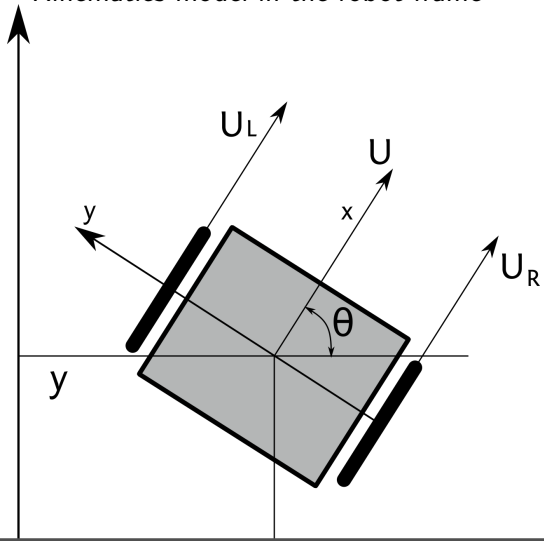
$$\Omega = \frac{U_R - U_L}{L} = \frac{(\omega_R - \omega_L)r}{L}$$

What happens when $u_R = u_L$ or $u_R = -u_L$?

Differential drive

Forward kinematics modeling

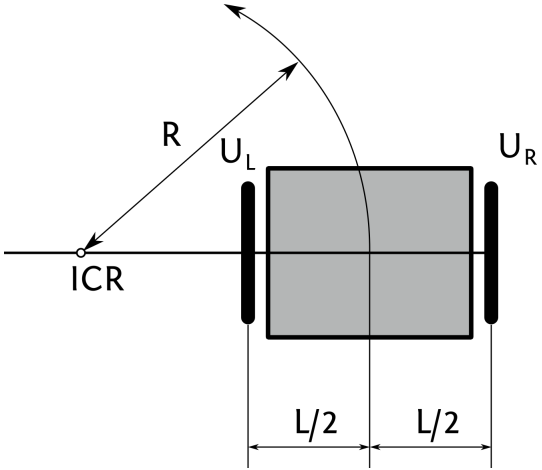
Kinematics model in the robot frame



$$\begin{bmatrix} U \\ \Omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

Differential drive

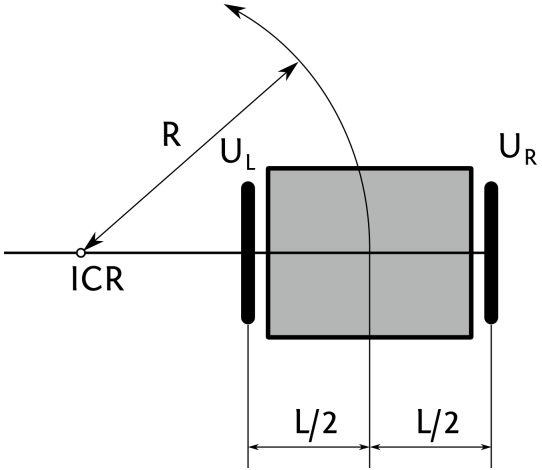
Inverse kinematics modeling



How do we define the inverse kinematics?

Differential drive

Inverse kinematics modeling

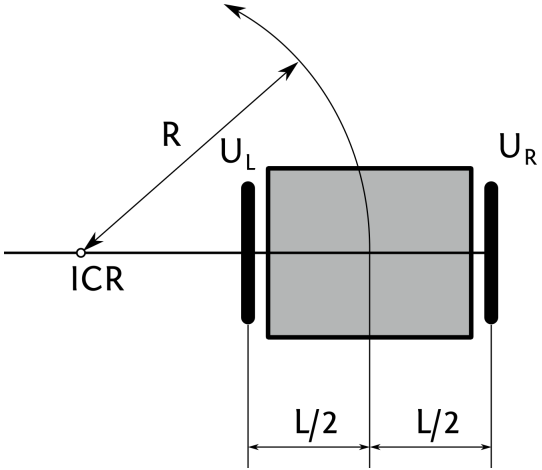


How do we define the inverse kinematics?

$$u_R = \Omega(R + \frac{L}{2}) = U(1 + \frac{L}{2R})$$
$$u_L = \Omega(R - \frac{L}{2}) = U(1 - \frac{L}{2R})$$

Differential drive

Inverse kinematics modeling



How do we define the inverse kinematics?

$$u_R = \Omega(R + \frac{L}{2}) = U(1 + \frac{L}{2R})$$
$$u_L = \Omega(R - \frac{L}{2}) = U(1 - \frac{L}{2R})$$

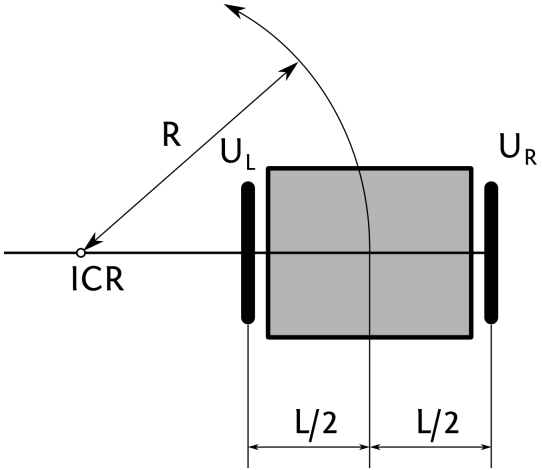
Where:

$u_R = r\omega_R$

$u_L = r\omega_L$

Differential drive

Inverse kinematics modeling



How do we define the inverse kinematics?

$$u_R = \Omega(R + \frac{L}{2}) = U(1 + \frac{L}{2R})$$
$$u_L = \Omega(R - \frac{L}{2}) = U(1 - \frac{L}{2R})$$

Where:

$$u_R = r\omega_R$$
$$u_L = r\omega_L$$

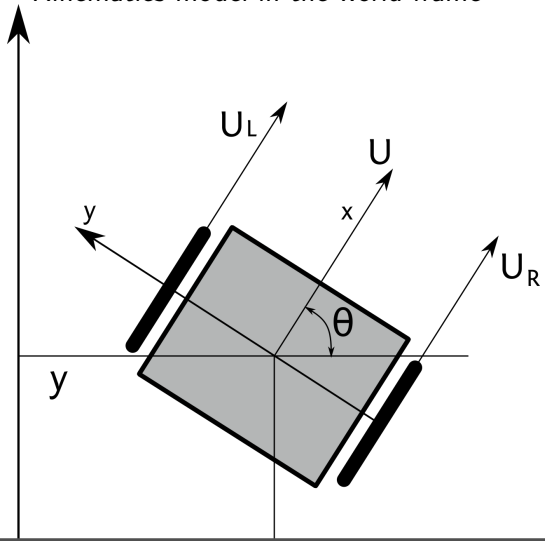
Therefore:

$$\omega_R = \Omega\frac{R + \frac{L}{2}}{r} = U\frac{1 + \frac{L}{2R}}{r}$$
$$\omega_L = \Omega\frac{R - \frac{L}{2}}{r} = U\frac{1 - \frac{L}{2R}}{r}$$

Differential drive

Kinematics modeling

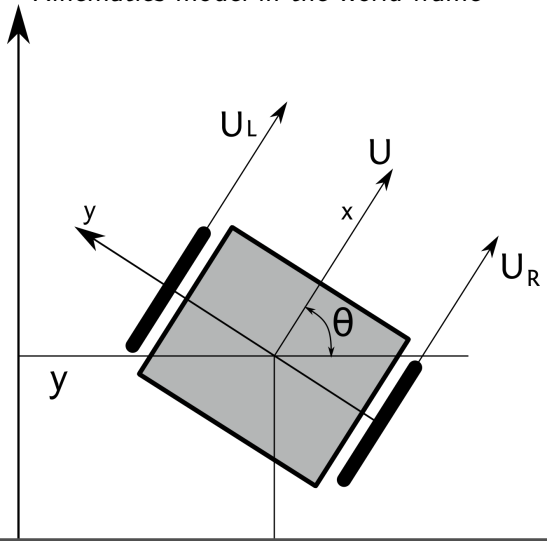
Kinematics model in the world frame



Differential drive

Kinematics modeling

Kinematics model in the world frame

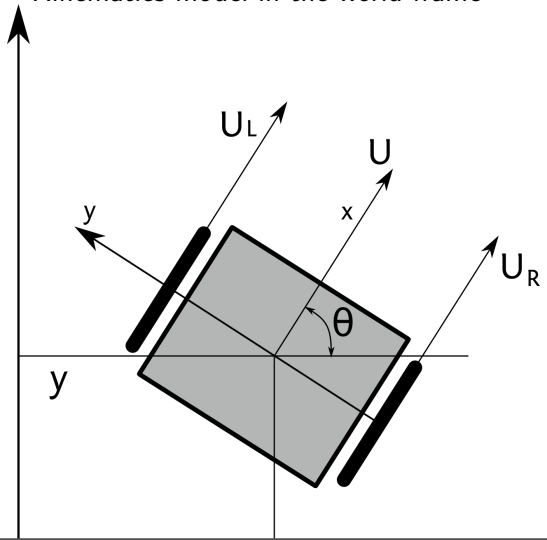


$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ \Omega \end{bmatrix}$$

Differential drive

Kinematics modeling

Kinematics model in the world frame



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ \Omega \end{bmatrix}$$

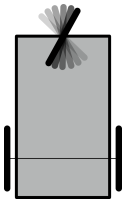
How do we calculate velocity in world frame with respect to ω_i?

Tricycle

Description

A wheeled robot with three wheels:

- Two fixed wheels with the same axis
- The two wheels can move independently
- One wheel that steers and pushes the robot
- The third wheel is usually between the other two with an offset

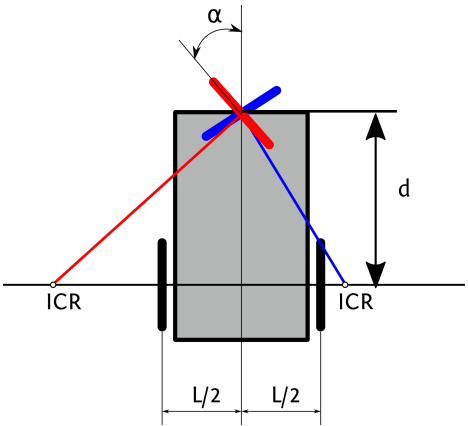


Tricycle

Tricycle

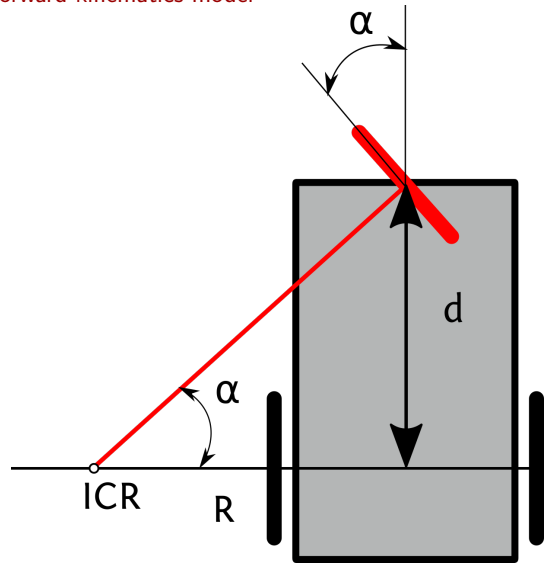
Kinematics

We control the location of the ICR by changing the steering angle α, and the velocity, by changing the wheel velocity ω_w



Tricycle

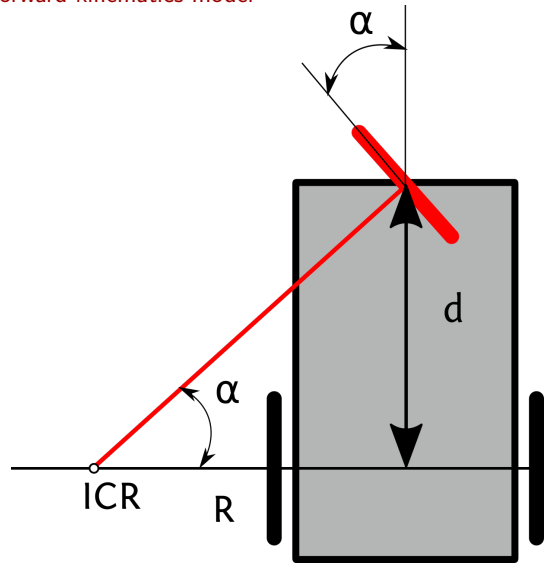
Forward kinematics model



If r is the steering wheel radius, then:

Tricycle

Forward kinematics model



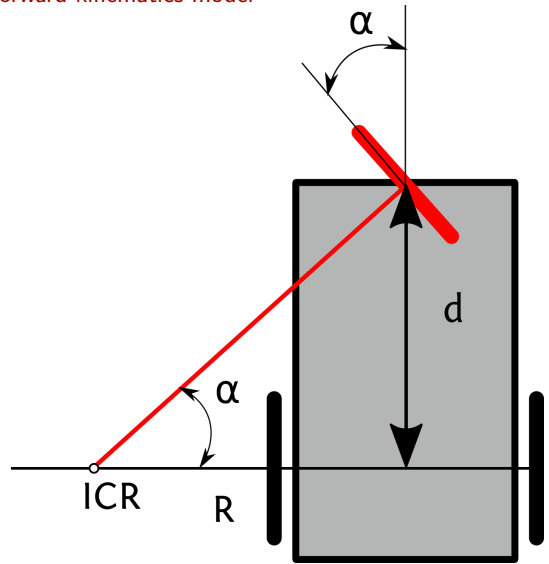
If r is the steering wheel radius, then:

$u_w = \omega_w r$

$R = d * tan(\frac{\pi}{2} - \alpha)$

Tricycle

Forward kinematics model



If r is the steering wheel radius, then:

$u_w = \omega_w r$

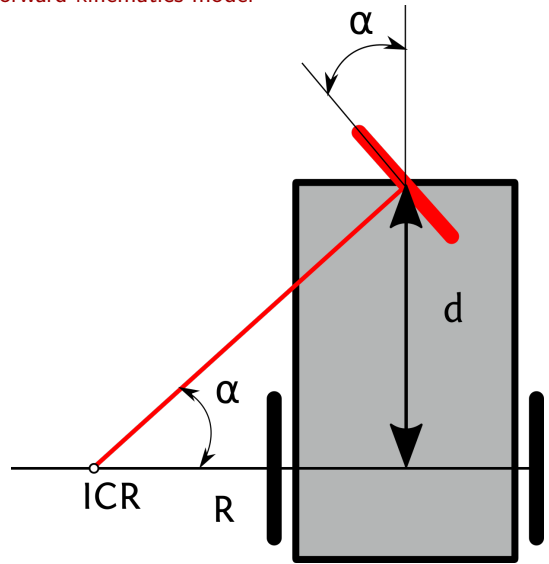
$R = d * tan(\frac{\pi}{2} - \alpha)$

The angular velocity of the robot relative to the base frame:

$$\Omega = \frac{u_w}{\sqrt{d^2 + R^2}}$$

Tricycle

Forward kinematics model



If r is the steering wheel radius, then:

$u_w = \omega_w r$

$R = d * tan(\frac{\pi}{2} - \alpha)$

The angular velocity of the robot relative to the base frame:

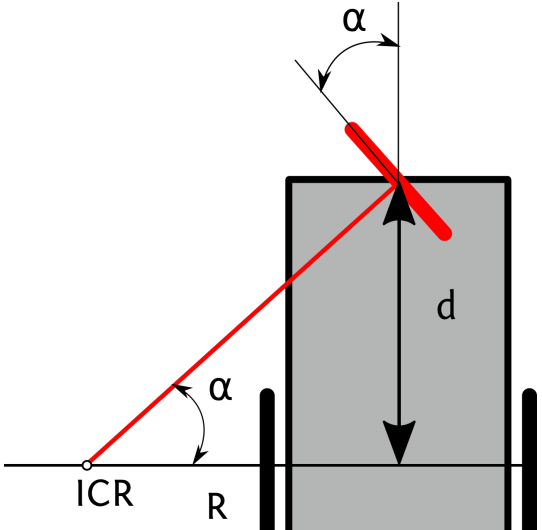
$$\Omega = \frac{u_w}{\sqrt{d^2 + R^2}}$$

$$\Omega = \frac{u_w}{d} sin(\alpha)$$

Tricycle

Forward kinematics model

Kinematics model in the robot body frame:



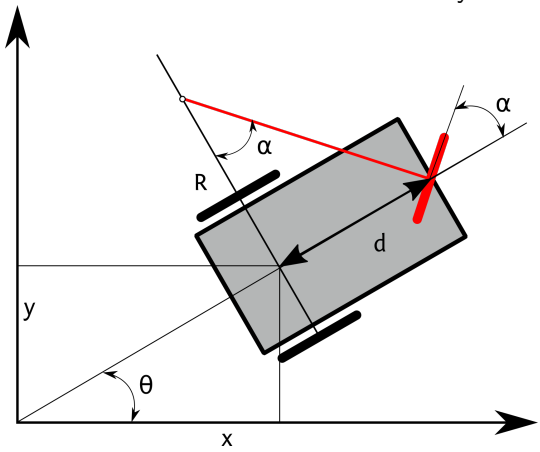
$$U = u_w cos(\alpha)$$

$$\Omega = \frac{u_w}{d} sin(\alpha)$$

Tricycle

Forward kinematics model

Kinematics model in the world body frame:



$$\dot{x} = u_w cos(\alpha)\cos(\theta)$$

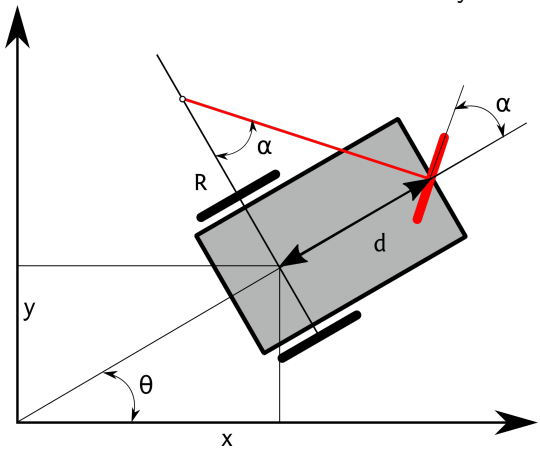
$$\dot{y} = u_w cos(\alpha)\sin(\theta)$$

$$\dot{\theta} = \omega = \frac{u_w}{d} sin(\alpha)$$

Tricycle

Forward kinematics model

Kinematics model in the world body frame:



$$\dot{x} = u_w cos(\alpha)\cos(\theta)$$

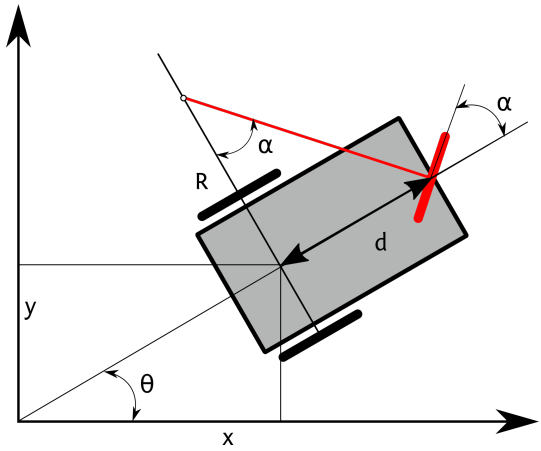
$$\dot{y} = u_w cos(\alpha)\sin(\theta)$$

$$\dot{\theta} = \omega = \frac{u_w}{d} sin(\alpha)$$

What about the inverse?

Tricycle

Inverse kinematics model



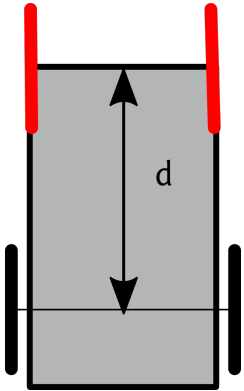
$$\alpha = atan(\frac{\dot{\theta} d sin\theta}{\dot{y}})$$

$$u_w = \frac{\dot{y}}{cos(\alpha)sin(\theta)}$$

Four wheels

Description

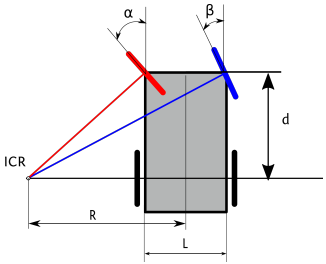
Another type of wheeled robot, is with four wheels. The two front are transmitting the power and are steered, while the back ones are fixed wheels



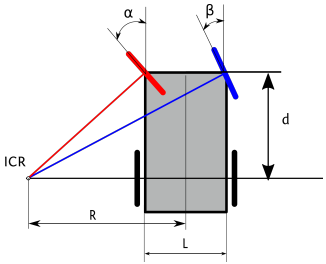
Four wheels

Ackerman drive

For this to work, the steering of the two wheels must be coordinated:



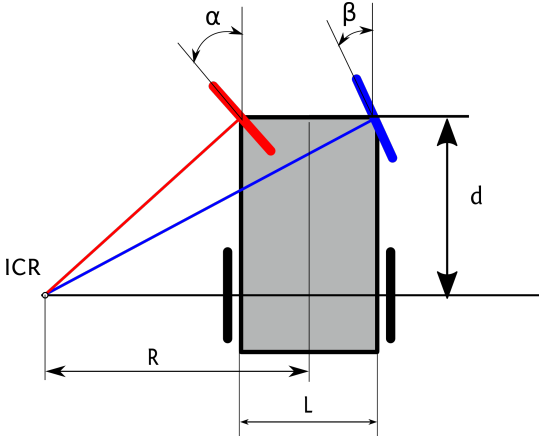
For this to work, the steering of the two wheels must be coordinated:



$\alpha > \beta$: when turning left
$\beta > \alpha$: when turning right

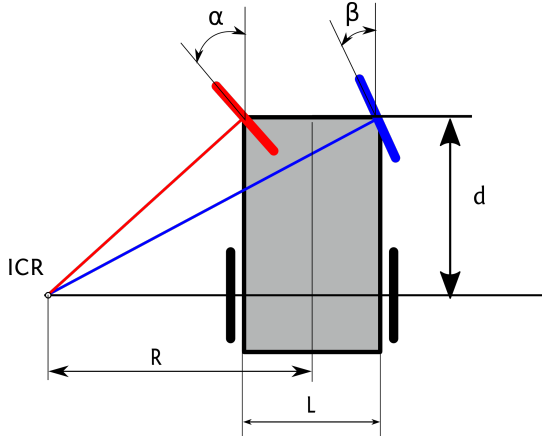
Four wheels

Ackerman drive



Four wheels

Ackerman drive

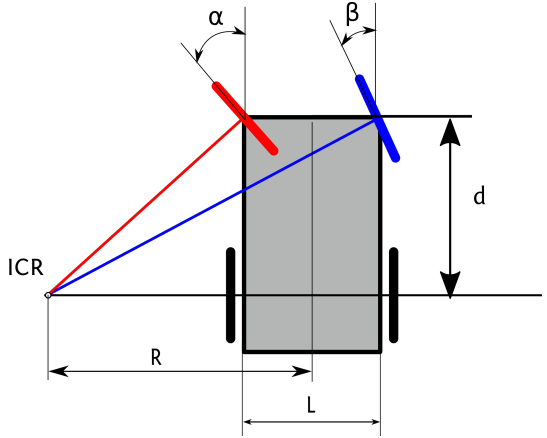


$$cot(\alpha) = \frac{R - \frac{L}{2}}{d}$$

$$cot(\beta) = \frac{R + \frac{L}{2}}{d}$$

Four wheels

Ackerman drive



$$cot(\alpha) = \frac{R - \frac{L}{2}}{d}$$

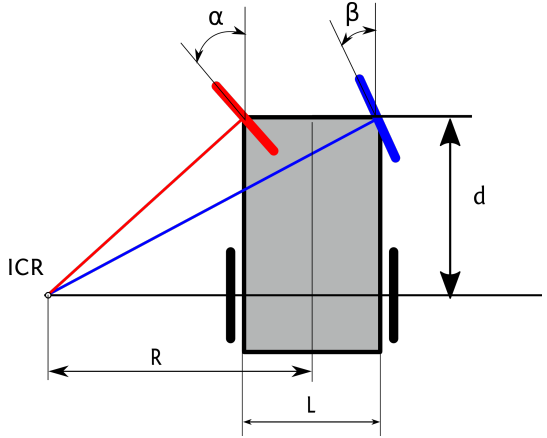
$$cot(\beta) = \frac{R + \frac{L}{2}}{d}$$

Therefore:

$$cot(\beta) - cot(\alpha) = \frac{L}{d}$$

Four wheels

Ackerman drive



$$cot(\alpha) = \frac{R - \frac{L}{2}}{d}$$

$$cot(\beta) = \frac{R + \frac{L}{2}}{d}$$

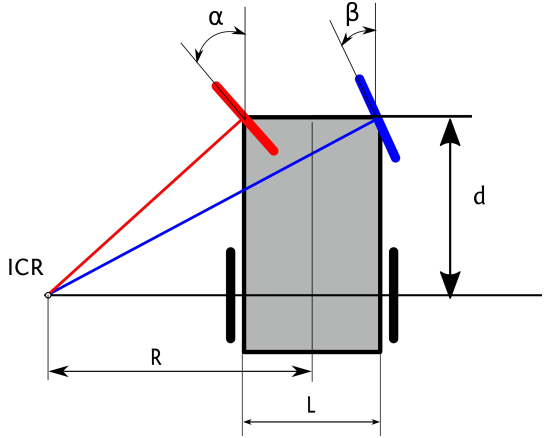
Therefore:

$$cot(\beta) - cot(\alpha) = \frac{L}{d}$$

What happens when $\alpha = \beta = 0$?

Four wheels

Ackerman drive



$$cot(\alpha) = \frac{R - \frac{L}{2}}{d}$$

$$cot(\beta) = \frac{R + \frac{L}{2}}{d}$$

Therefore:

$$cot(\beta) - cot(\alpha) = \frac{L}{d}$$

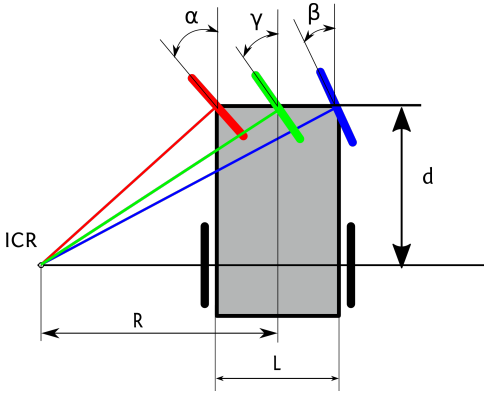
What happens when $\alpha = \beta = 0$?

What is the relationship between angular velocities ω_L and ω_R?

Four wheels

Ackerman and tricycle

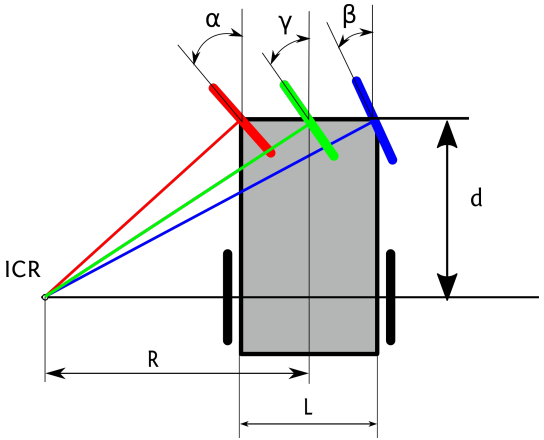
We can describe the ackerman drive kinematics, the same way as for the tricycle, if we consider a virtual fifth wheel between the two front ones



Four wheels

Ackerman drive

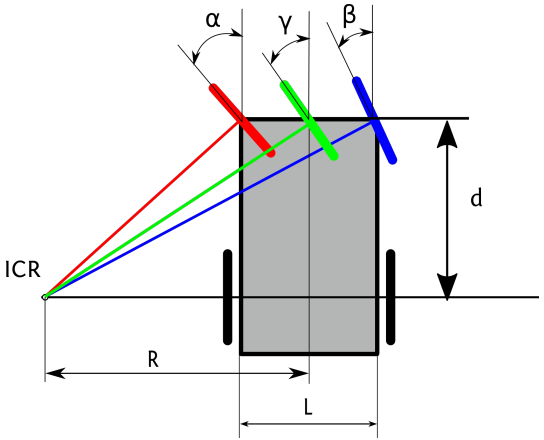
We can easily calculate the equivalent virtual angle γ



Four wheels

Ackerman drive

We can easily calculate the equivalent virtual angle γ



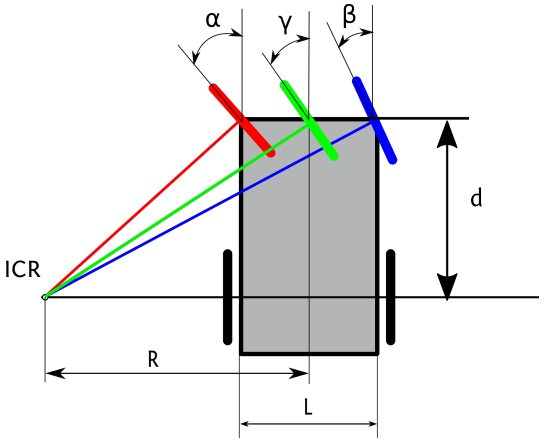
$$cot(\gamma) = cot(\alpha) + \frac{L}{2d}$$

$$cot(\gamma) = cot(\beta) - \frac{L}{2d}$$

Four wheels

Ackerman drive

We can easily calculate the equivalent virtual angle γ



$$cot(\gamma) = cot(\alpha) + \frac{L}{2d}$$

$$cot(\gamma) = cot(\beta) - \frac{L}{2d}$$

The kinematics models then are the same as for a tricycle with steering angle γ

Four wheels

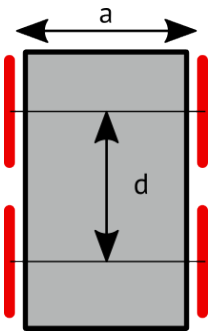
Skid steer drive

The skid steer drive consists of four individually driven wheels, all with a fixed direction

Four wheels

Skid steer drive

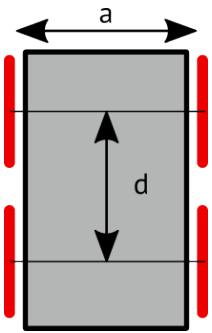
The skid steer drive consists of four individually driven wheels, all with a fixed direction



Four wheels

Skid steer drive

The skid steer drive consists of four individually driven wheels, all with a fixed direction

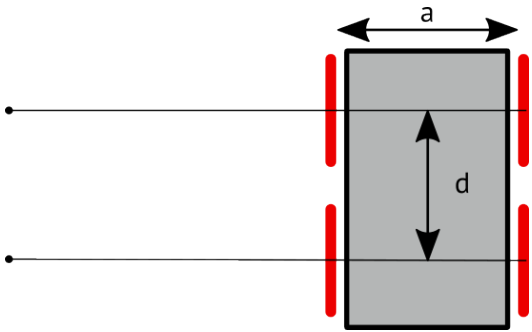


What is the issue with this design?

Four wheels

Skid steer drive

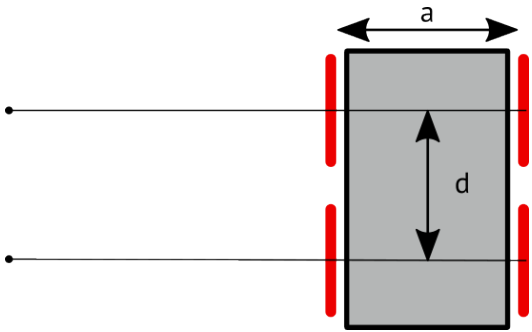
The skid steer drive consists of four individually driven wheels, all with a fixed direction



Four wheels

Skid steer drive

The skid steer drive consists of four individually driven wheels, all with a fixed direction



Two different ICR for the robot!

Skid steer drive

Issues

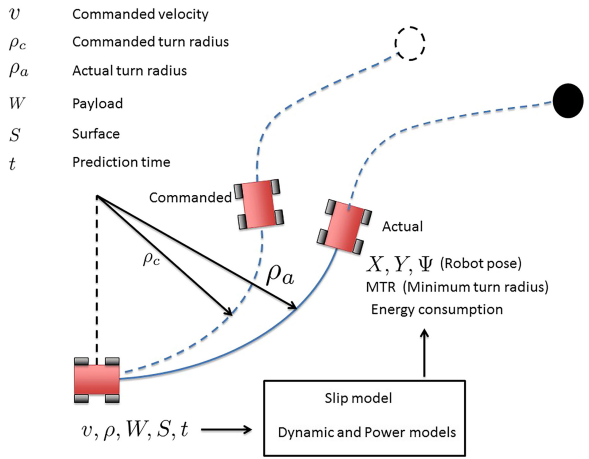


Figure: From: Learning of skid-steered kinematic and dynamic models for motion planning, Ordonez et. al.

Skid steer drive

Modelling

We assume a differential drive model:

$$\begin{bmatrix} U \\ \Omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

Skid steer drive

Modelling

We assume a differential drive model:

$$\begin{bmatrix} U \\ \Omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

We add one more dimension for lateral movement (slip, u_l)

Skid steer drive

Modelling

We assume a differential drive model:

$$\begin{bmatrix} U \\ \Omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

We add one more dimension for lateral movement (slip, u_l)

$$\begin{bmatrix} U_f \\ U_l \\ \Omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

Skid steer drive

Modelling

We assume a differential drive model:

$$\begin{bmatrix} U \\ \Omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

We add one more dimension for lateral movement (slip, u_l)
And we consider some disturbances on each direction

$$\begin{bmatrix} U_f \\ U_l \\ \Omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} + \begin{bmatrix} \delta u_f \\ \delta u_l \\ \delta \omega \end{bmatrix}$$

Skid steer drive

Modelling slipage

There are different ways to model these *disturbances*:

Skid steer drive

Modelling slipage

There are different ways to model these *disturbances*:

$$\delta u_f = \alpha_{11} u_c + \alpha_{12} \omega_c + \alpha_{13} u_c \omega_c$$
$$\delta u_l = \alpha_{21} u_c + \alpha_{22} \omega_c + \alpha_{23} u_c \omega_c$$
$$\delta \omega = \alpha_{31} u_c + \alpha_{32} \omega_c + \alpha_{33} u_c \omega_c$$

Skid steer drive

Modelling slipage

There are different ways to model these *disturbances*:

$$\delta u_f = \alpha_{11} u_c + \alpha_{12} \omega_c + \alpha_{13} u_c \omega_c$$
$$\delta u_l = \alpha_{21} u_c + \alpha_{22} \omega_c + \alpha_{23} u_c \omega_c$$
$$\delta \omega = \alpha_{31} u_c + \alpha_{32} \omega_c + \alpha_{33} u_c \omega_c$$

or:

$$\begin{bmatrix} \delta u_f \\ \delta u_l \\ \delta \omega \end{bmatrix} = A \begin{bmatrix} u_c \\ \omega_c \\ u_c \omega_c \end{bmatrix}$$

Skid steer drive

Modelling slipage

There are different ways to model these *disturbances*:

$$\delta u_f = \alpha_{11} u_c + \alpha_{12} \omega_c + \alpha_{13} u_c \omega_c$$
$$\delta u_l = \alpha_{21} u_c + \alpha_{22} \omega_c + \alpha_{23} u_c \omega_c$$
$$\delta \omega = \alpha_{31} u_c + \alpha_{32} \omega_c + \alpha_{33} u_c \omega_c$$

or:

$$\begin{bmatrix} \delta u_f \\ \delta u_l \\ \delta \omega \end{bmatrix} = A \begin{bmatrix} u_c \\ \omega_c \\ u_c \omega_c \end{bmatrix}$$

What does the matrix A depend on?

There are different ways to model these *disturbances*:

$$\delta u_f = \alpha_{11} u_c + \alpha_{12} \omega_c + \alpha_{13} u_c \omega_c$$
$$\delta u_l = \alpha_{21} u_c + \alpha_{22} \omega_c + \alpha_{23} u_c \omega_c$$
$$\delta \omega = \alpha_{31} u_c + \alpha_{32} \omega_c + \alpha_{33} u_c \omega_c$$

or:

$$\begin{bmatrix} \delta u_f \\ \delta u_l \\ \delta \omega \end{bmatrix} = A \begin{bmatrix} u_c \\ \omega_c \\ u_c \omega_c \end{bmatrix}$$

What does the matrix A depend on?
How do we calculate it?

There are different ways to model these *disturbances*:

$$\delta u_f = \alpha_{11} u_c + \alpha_{12} \omega_c + \alpha_{13} u_c \omega_c$$
$$\delta u_l = \alpha_{21} u_c + \alpha_{22} \omega_c + \alpha_{23} u_c \omega_c$$
$$\delta \omega = \alpha_{31} u_c + \alpha_{32} \omega_c + \alpha_{33} u_c \omega_c$$

or:

$$\begin{bmatrix} \delta u_f \\ \delta u_l \\ \delta \omega \end{bmatrix} = A \begin{bmatrix} u_c \\ \omega_c \\ u_c \omega_c \end{bmatrix}$$

What does the matrix A depend on?
How do we calculate it?
What do we do when we cannot rely on it?

Inertial Measurement Unit

IMU

When we cannot rely on characterising matrix A, we need external sensors for feedback.

Inertial Measurement Unit

IMU

When we cannot rely on characterising matrix A, we need external sensors for feedback.

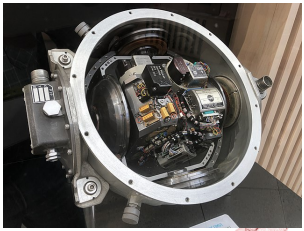


Figure: Inertial Measurement Unit of the apollo missions

Inertial Measurement Unit

IMU

When we cannot rely on characterising matrix A, we need external sensors for feedback.

IMUs provide information on acceleration, rotational rate, and sometimes magnetic direction

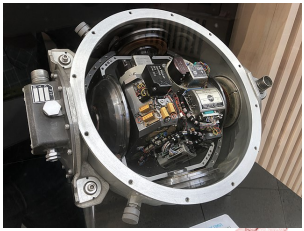


Figure: Inertial Measurement Unit of the apollo missions

Inertial Measurement Unit

IMU

When we cannot rely on characterising matrix A, we need external sensors for feedback.

IMUs provide information on acceleration, rotational rate, and sometimes magnetic direction

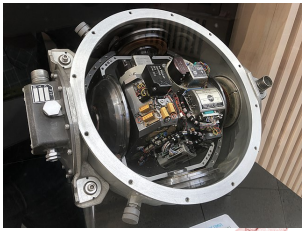
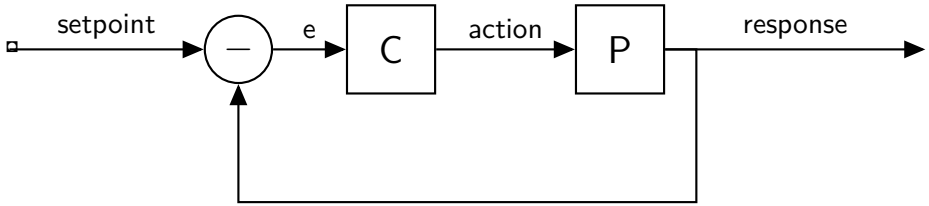
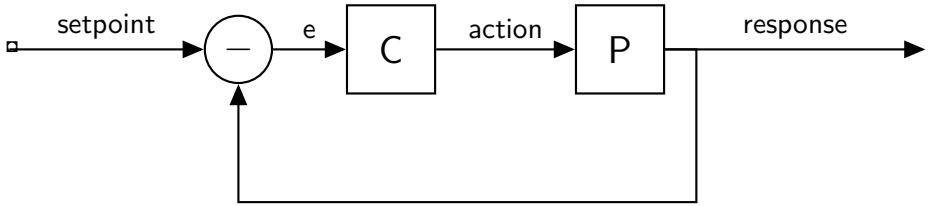


Figure: Inertial Measurement Unit of the apollo missions

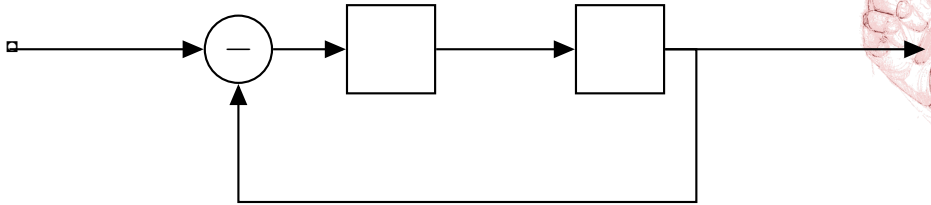
Basic control



Basic control

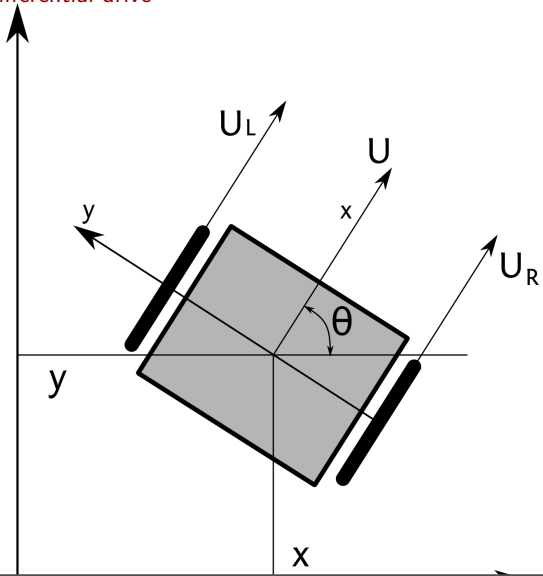


Let's fill in the blanks



Dynamic modelling

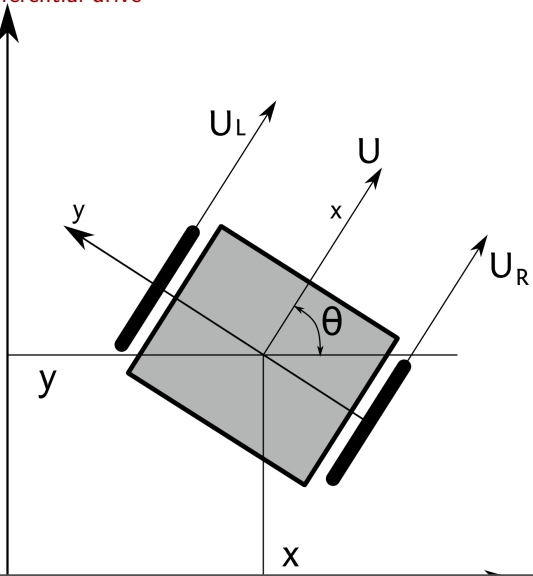
Differential drive



Generalised coordinates?

Dynamic modelling

Differential drive



Generalised coordinates?
Euler-Lagrange?

Why do we need planning?

Why do we need planning?

- The world is full of obstacles

Mobile robots

Motion planning

Why do we need planning?

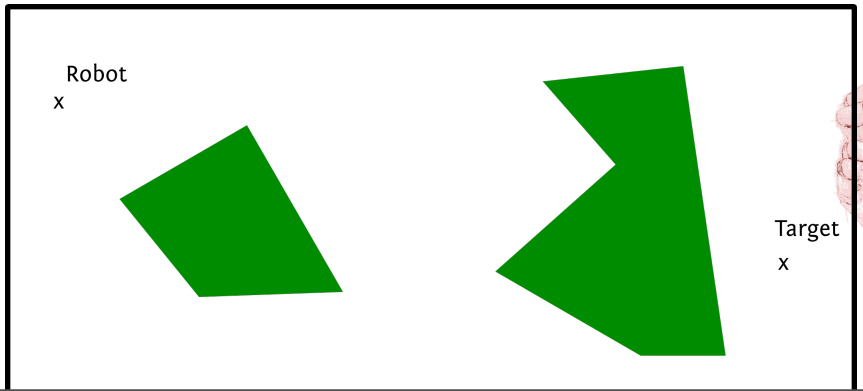
- The world is full of obstacles
- We want to find the most efficient way

Mobile robots

Motion planning

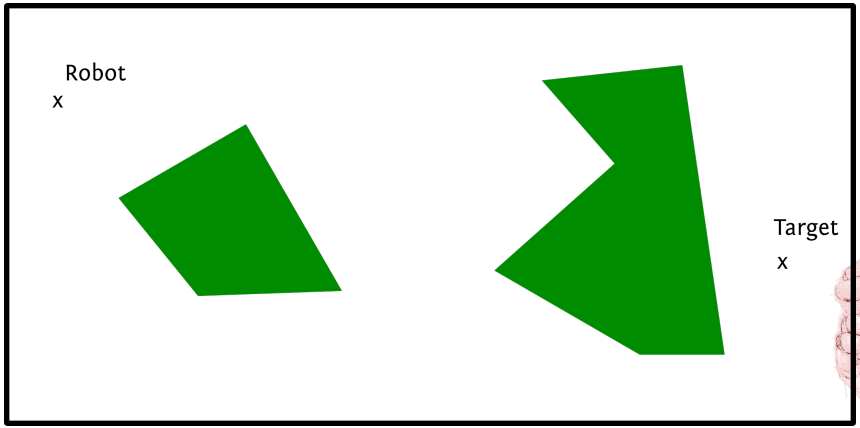
Why do we need planning?

- The world is full of obstacles
- We want to find the most efficient way



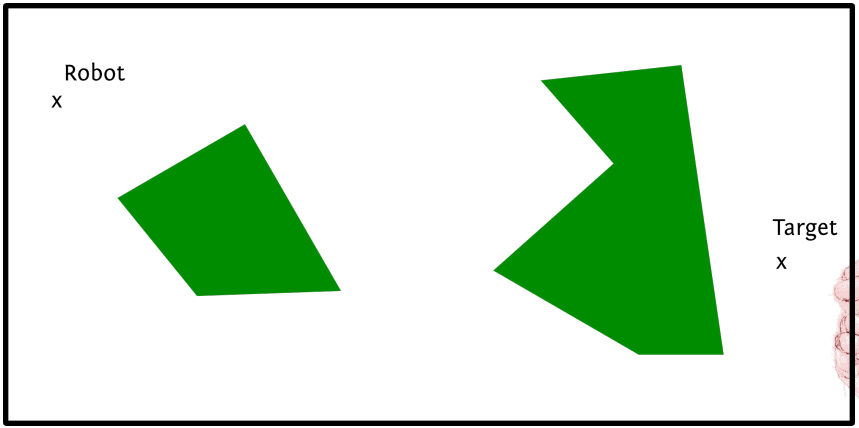
Mobile robots

Motion planning



Mobile robots

Motion planning

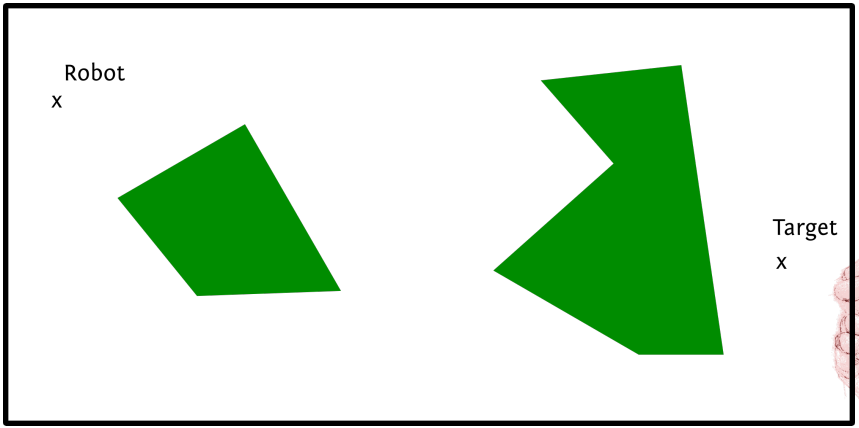


Input

- Geometric description of robot
- Geometric description of the environment

Mobile robots

Motion planning



Input

- Geometric description of robot
- Geometric description of the environment

Output

A path from the initial position until the goal

Mobile robots

Motion planning methods

Roadmap approaches:
Reduce all the possible paths to a subset of them

Roadmap approaches:
Reduce all the possible paths to a subset of them

Cell decomposition:
Account for all of the free space

Roadmap approaches:
Reduce all the possible paths to a subset of them

Cell decomposition:
Account for all of the free space

Potential fields:
Local control strategies, optimality

Roadmap approaches:
Reduce all the possible paths to a subset of them

Cell decomposition:
Account for all of the free space

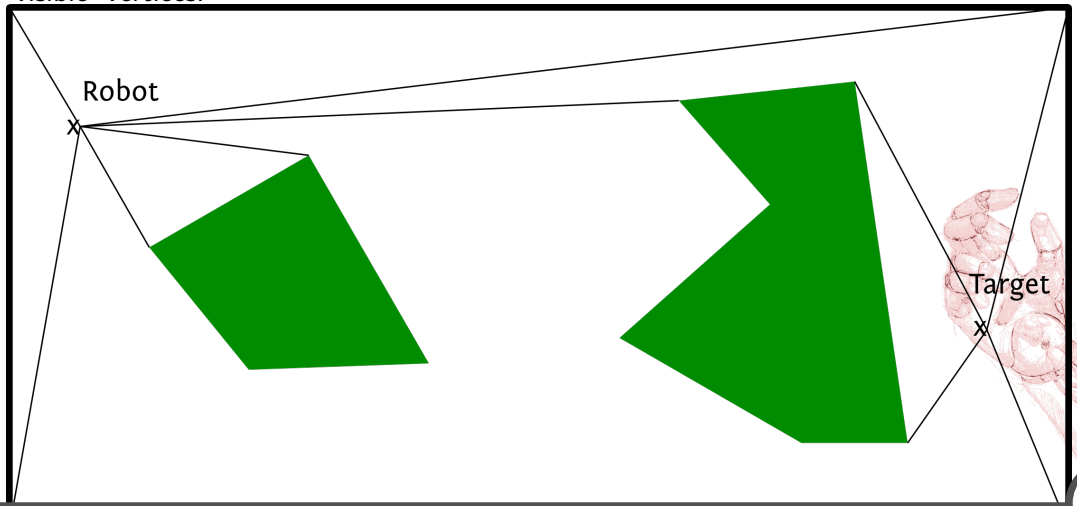
Potential fields:
Local control strategies, optimality

Bug algorithms:
Limited knowledge of environment

Motion planning

Roadmap approaches

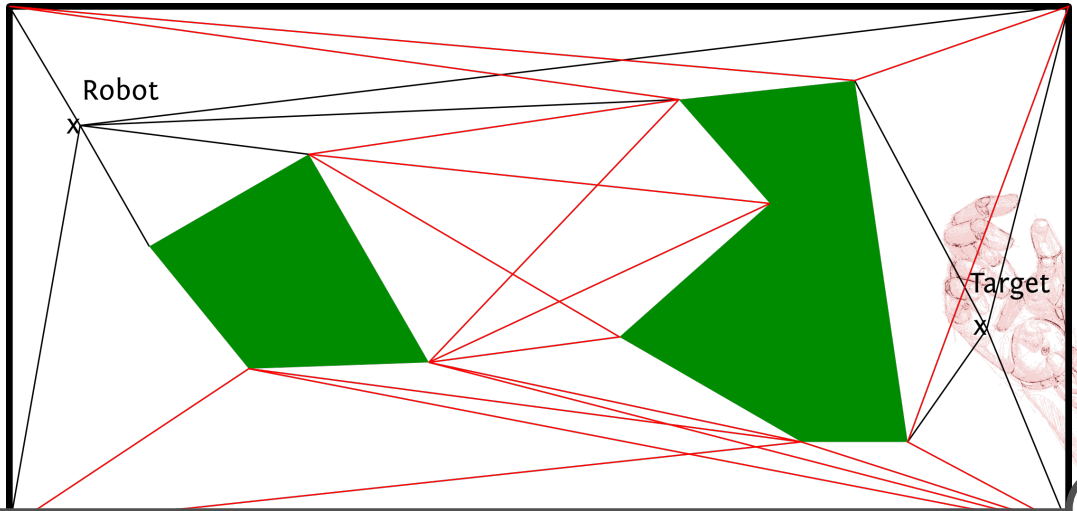
We construct by drawing lines of 'sight' from the initial position and target to all their 'visible' vertices.



Motion planning

Roadmap approaches

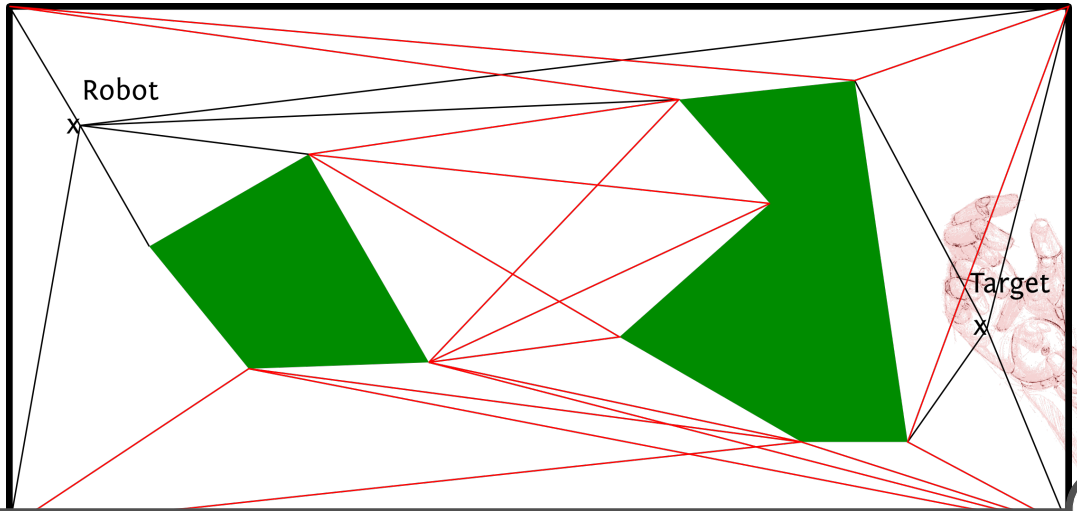
We then do the same for all vertices



Motion planning

Roadmap approaches

We then do the same for all vertices



Motion planning

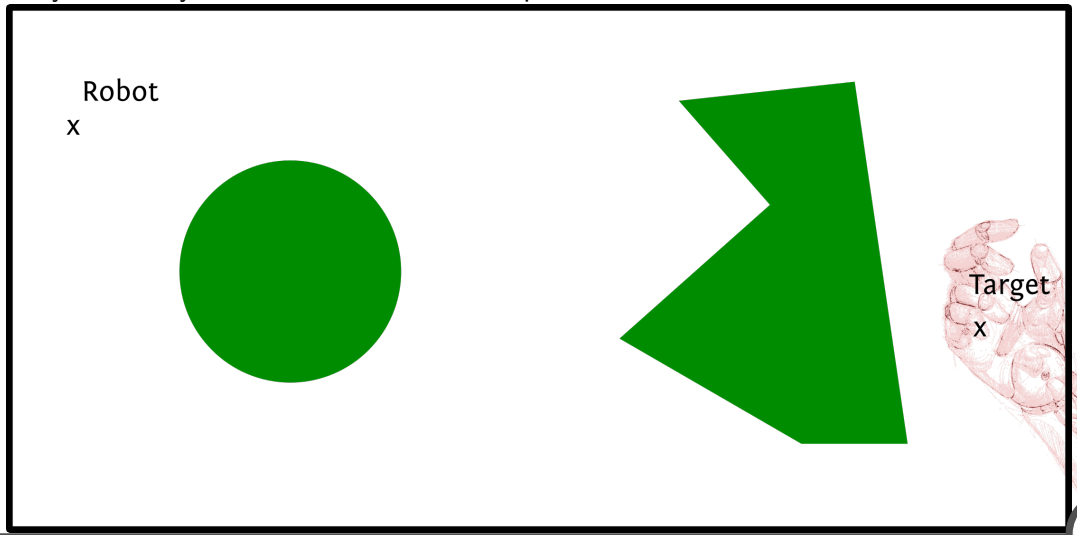
Roadmap approaches

Do you see any drawback with this technique?

Motion planning

Roadmap approaches

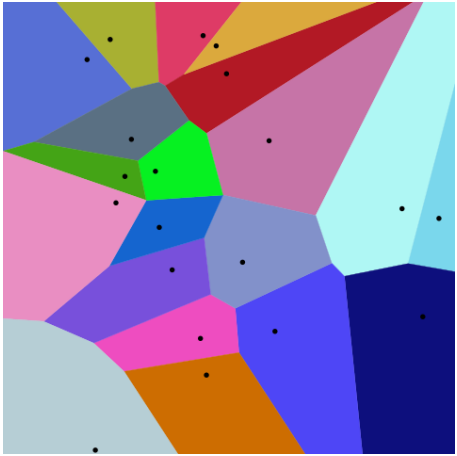
Do you see any drawback with this technique?



Motion planning

Voronoi diagrams

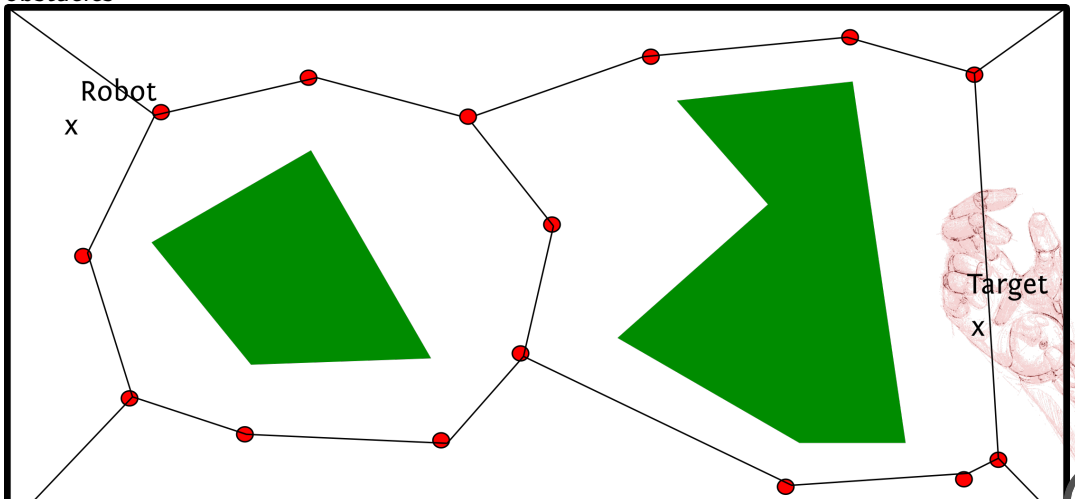
A Voronoi diagram is a partitioning of a plane so that different areas are the closest to a specific point.



Motion planning

Voronoi diagrams

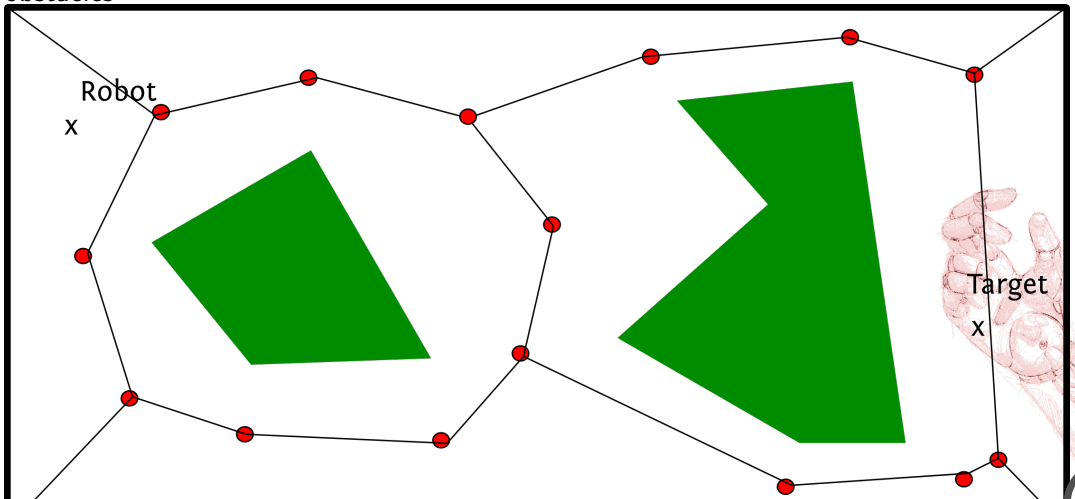
We use a very similar approach for constructing lines that are equally appart from obstacles



Motion planning

Voronoi diagrams

We use a very similar approach for constructing lines that are equally appart from obstacles

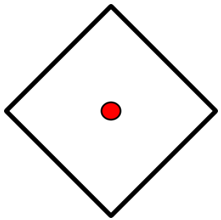


There are different metrics for defining the distance:

There are different metrics for defining the distance:

L1 metric:

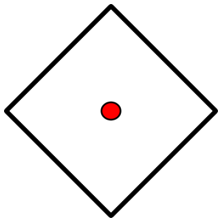
$$(x, y) : |x| + |y| = const$$



There are different metrics for defining the distance:

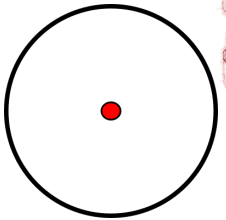
L1 metric:

$(x, y) : |x| + |y| = const$



L2 metric:

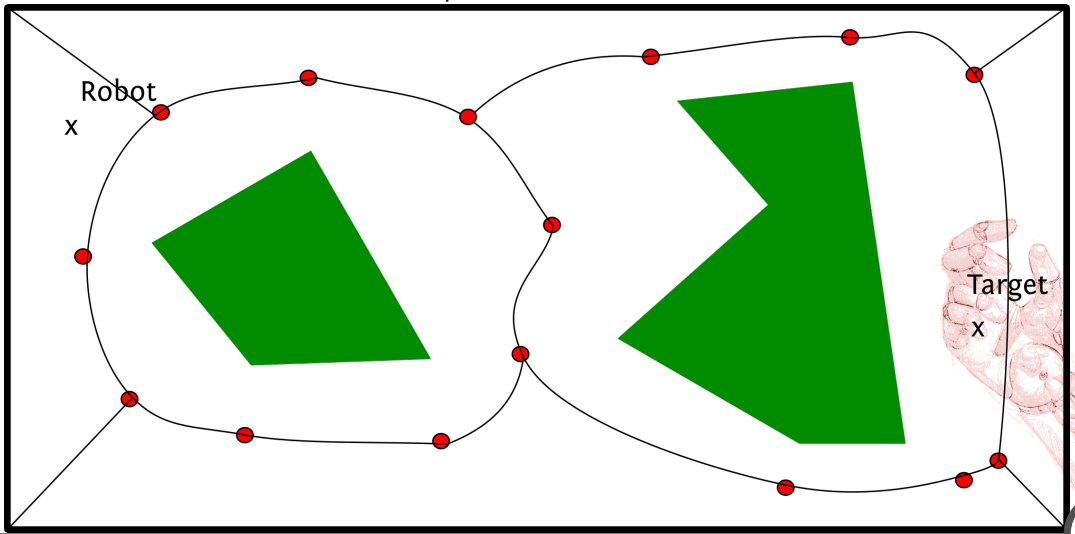
$(x, y) : x^2 + y^2 = const$



Motion planning

Voronoi diagrams

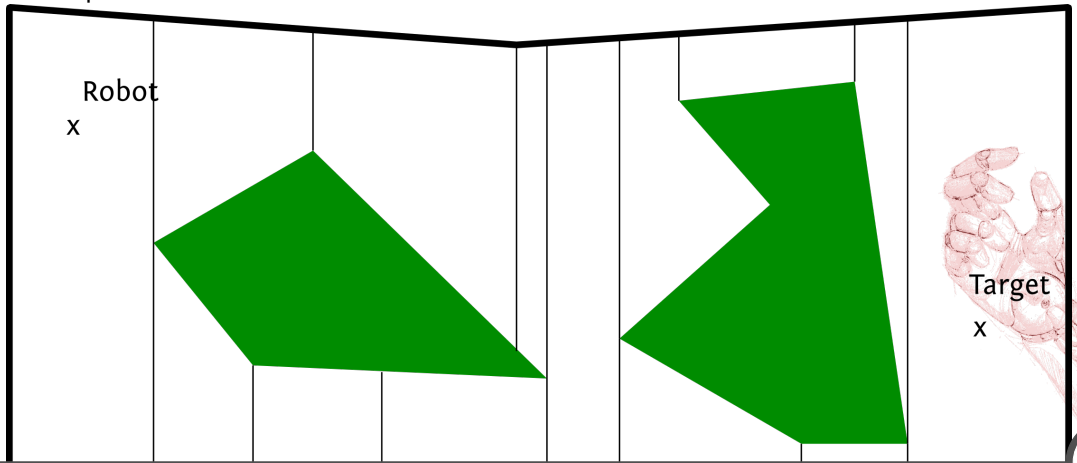
Different metrics, result in different paths



Motion planning

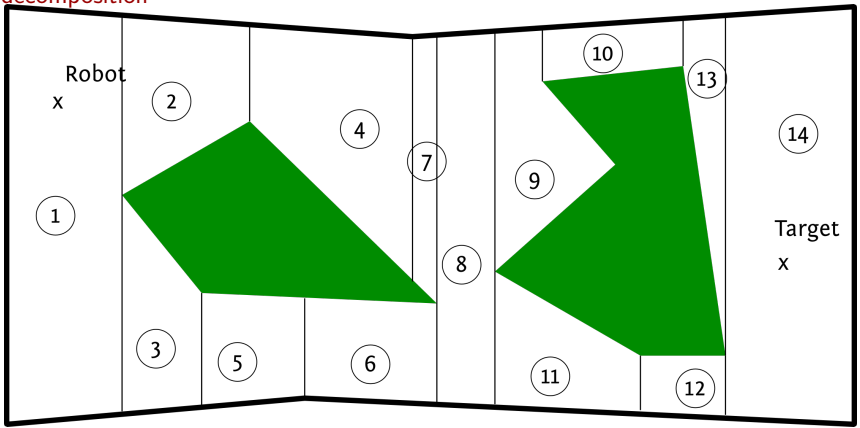
Cell decomposition

We decompose the available space into cells, and we create a connectivity graph, which helps us identify possible paths. There are different ways of performing the decomposition.



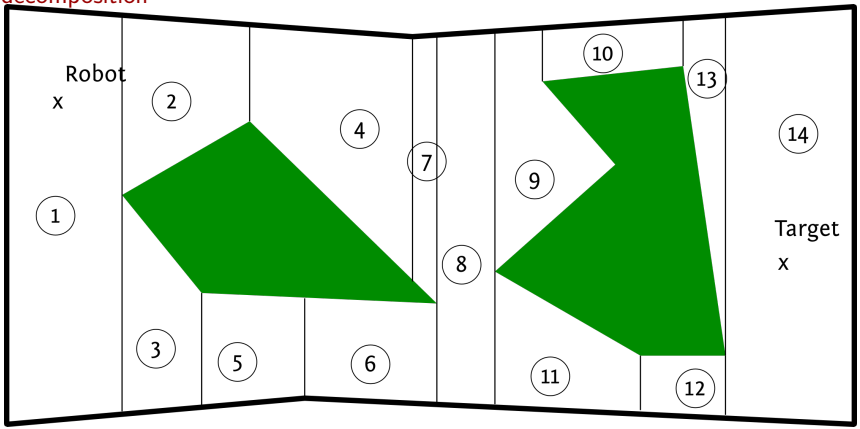
Motion planning

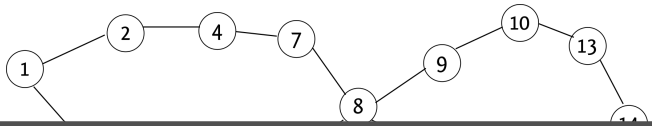
Cell decomposition



Motion planning

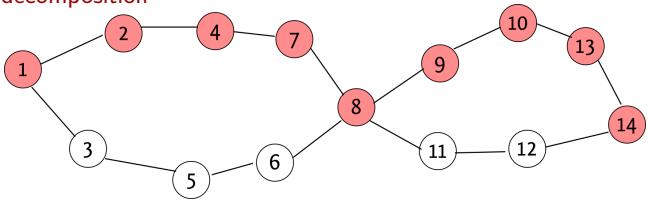
Cell decomposition





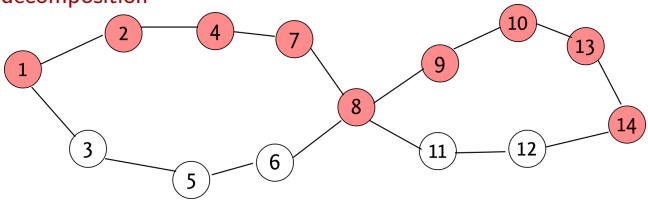
Motion planning

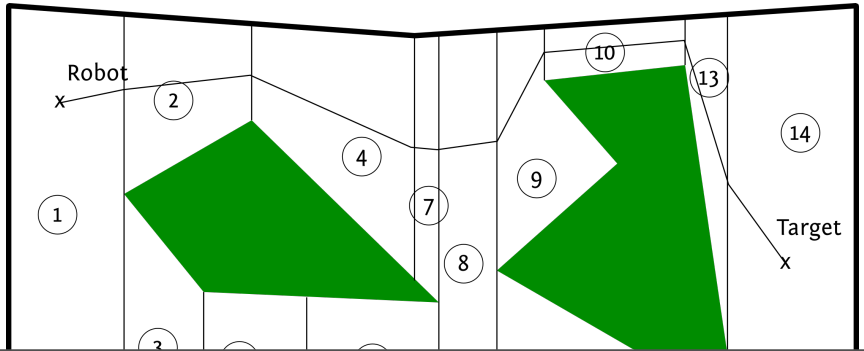
Cell decomposition



Motion planning

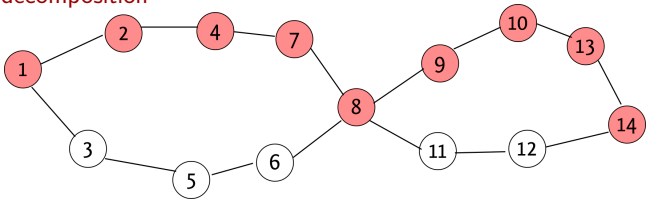
Cell decomposition

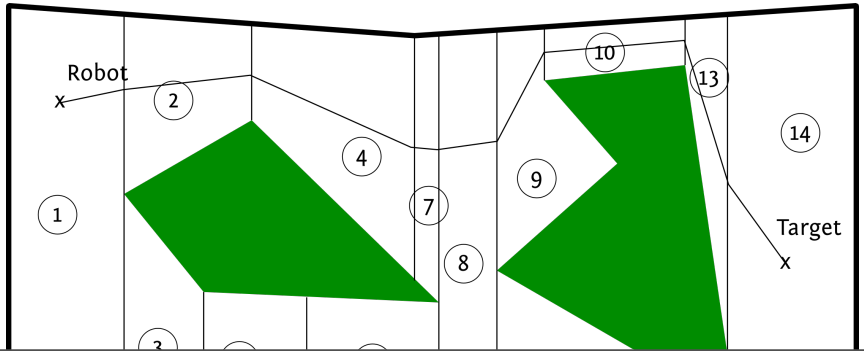




Motion planning

Cell decomposition





We construct a potential function that 'pulls' our robot towards the goal and is being 'pushed' by the obstacles. To do this, we need to:

We construct a potential function that 'pulls' our robot towards the goal and is being 'pushed' by the obstacles. To do this, we need to:

- Generate an attractive potential function centered at the goal

We construct a potential function that 'pulls' our robot towards the goal and is being 'pushed' by the obstacles. To do this, we need to:

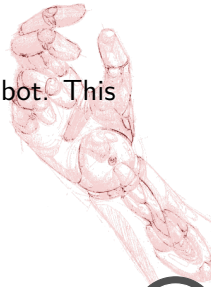
- Generate an attractive potential function centered at the goal
- Generate repulsive potential functions at the edges of the obstacles

We construct a potential function that 'pulls' our robot towards the goal and is being 'pushed' by the obstacles. To do this, we need to:

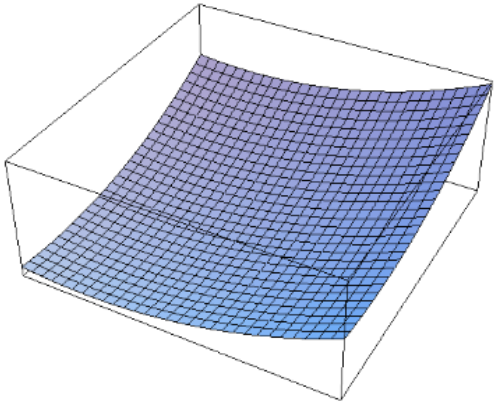
- Generate an attractive potential function centered at the goal
- Generate repulsive potential functions at the edges of the obstacles
- Add the two together to come up with a complex potential function

We construct a potential function that 'pulls' our robot towards the goal and is being 'pushed' by the obstacles. To do this, we need to:

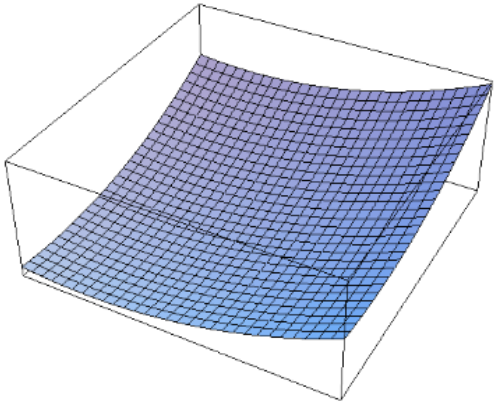
- Generate an attractive potential function centered at the goal
- Generate repulsive potential functions at the edges of the obstacles
- Add the two together to come up with a complex potential function
- The gradient of the total potential is an artificial force that drives the robot. This ensures optimal path



Attractive field

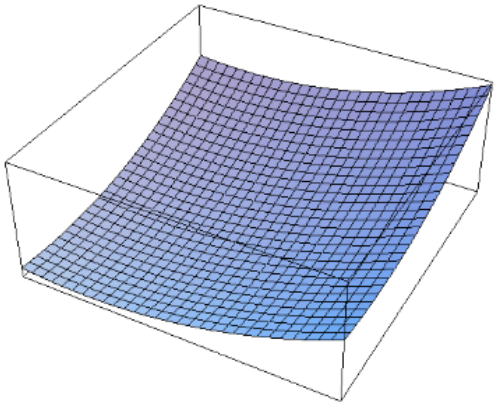


Attractive field



$$U_{at} = \frac{1}{2}\xi \|q - q_{goal}\|^2$$

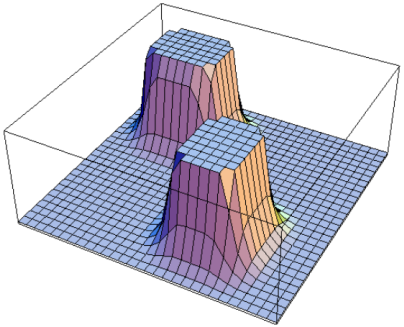
Attractive field



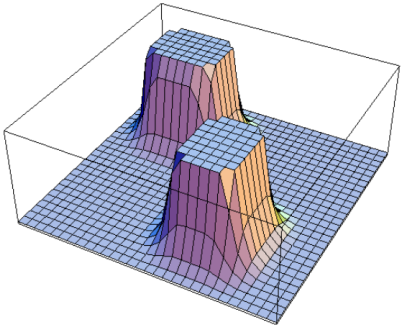
$$U_{att} = \frac{1}{2}\xi \|q - q_{goal}\|^2$$

The potential is parabolic and centered at position q_{goal}

Repulsive field



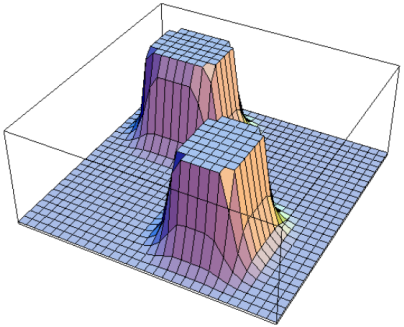
Repulsive field



$$U_{rep} = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases}$$

The potential is reciprocal with distance ρ, which is the distance from the edge of the obstacle.

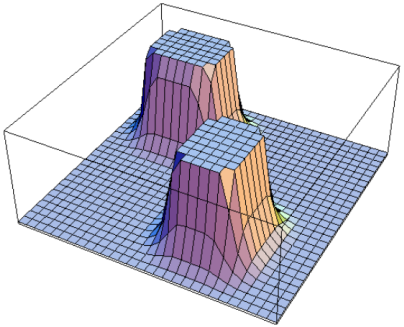
Repulsive field



$$U_{rep} = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases}$$

The potential is reciprocal with distance ρ, which is the distance from the edge of the obstacle.
We want the effect of the repulsion to wear off after distance ρ_0

Repulsive field

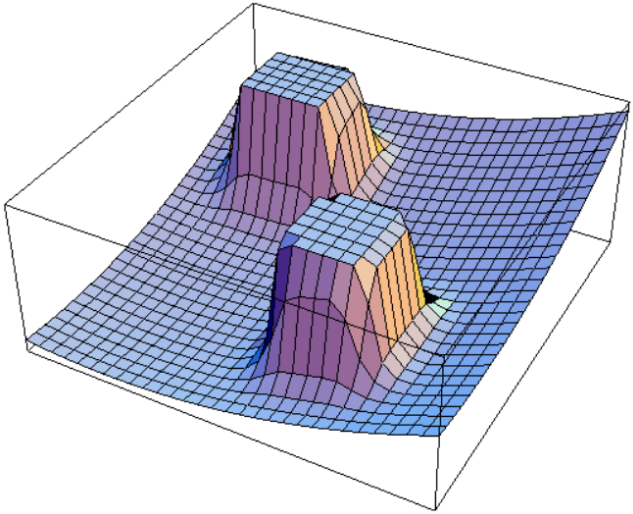


$$U_{rep} = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases}$$

Motion planning

Potential field method

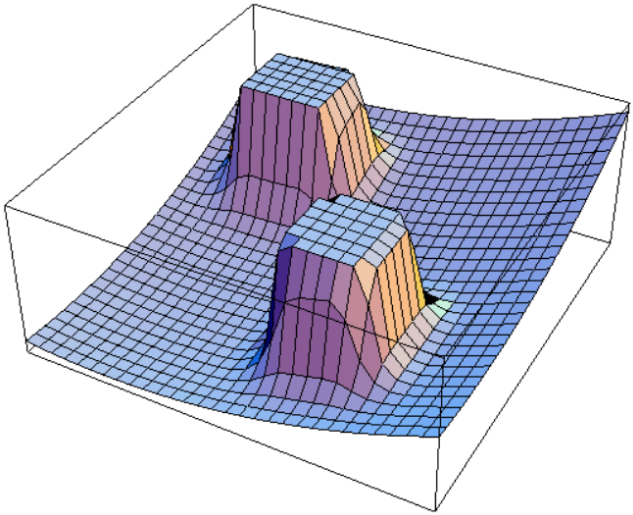
When adding the two potentials, we get a complex potential that can guide our robot



Motion planning

Potential field method

When adding the two potentials, we get a complex potential that can guide our robot

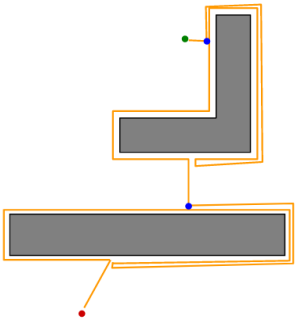


Of course, this isn't a perfect solution. What do you think are its limitations?

Motion planning

Bug 0 algorithm

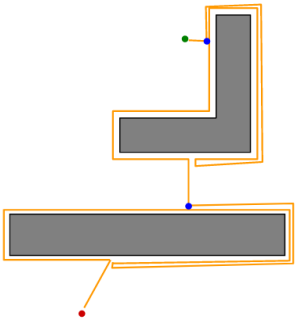
- Known initial and goal state



Motion planning

Bug 0 algorithm

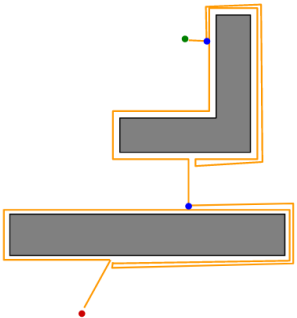
- Known initial and goal state
- Unknown obstacles



Motion planning

Bug 0 algorithm

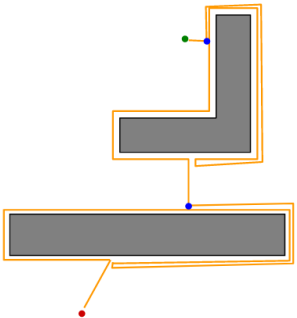
- Known initial and goal state
- Unknown obstacles
- Follow direction towards goal, stop when encounter an obstacle



Motion planning

Bug 0 algorithm

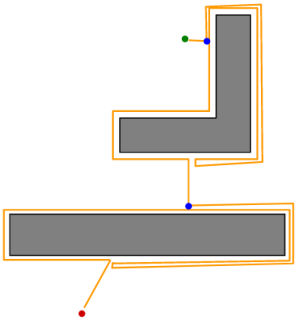
- Known initial and goal state
- Unknown obstacles
- Follow direction towards goal, stop when encounter an obstacle
- Encircle the obstacle until the point of encounter



Motion planning

Bug 0 algorithm

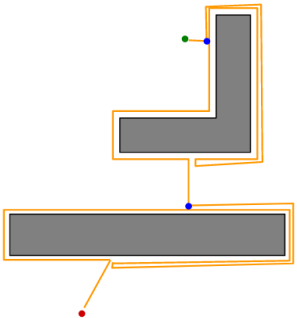
- Known initial and goal state
- Unknown obstacles
- Follow direction towards goal, stop when encounter an obstacle
- Encircle the obstacle until the point of encounter
- Go to the point closest to the goal



Motion planning

Bug 0 algorithm

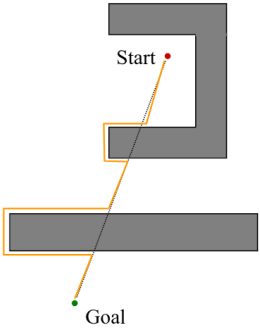
- Known initial and goal state
- Unknown obstacles
- Follow direction towards goal, stop when encounter an obstacle
- Encircle the obstacle until the point of encounter
- Go to the point closest to the goal
- Repeat until reaching the goal



Motion planning

Bug 1 algorithm

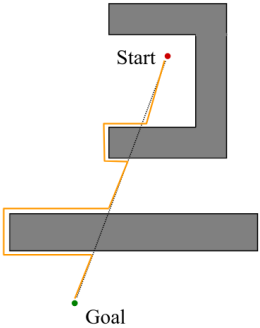
- Known initial and goal state



Motion planning

Bug 1 algorithm

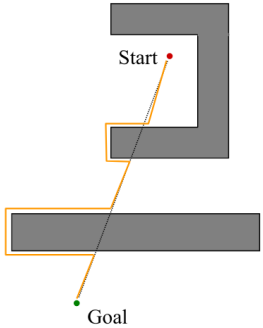
- Known initial and goal state
- Unknown obstacles



Motion planning

Bug 1 algorithm

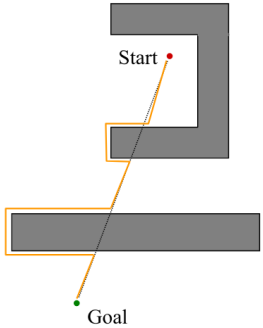
- Known initial and goal state
- Unknown obstacles
- Follow direction towards goal, stop when encounter an obstacle



Motion planning

Bug 1 algorithm

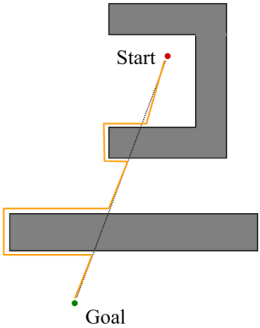
- Known initial and goal state
- Unknown obstacles
- Follow direction towards goal, stop when encounter an obstacle
- Encircle the obstacle until we reach again the line of sight with the goal



Motion planning

Bug 1 algorithm

- Known initial and goal state
- Unknown obstacles
- Follow direction towards goal, stop when encounter an obstacle
- Encircle the obstacle until we reach again the line of sight with the goal
- Repeat until reaching the goal



Questions?