上海科技大学
ShanghaiTech University

CS283: **Robotics Fall 2019: Robot Arms**

Sören Schwertfeger / 师泽仁

ShanghaiTech University

Outline

- RANSAC & Hough Transform
- Arm links & joints
- Kinematics
- Admin
- Arm Planning
- MoveIt & Grasping

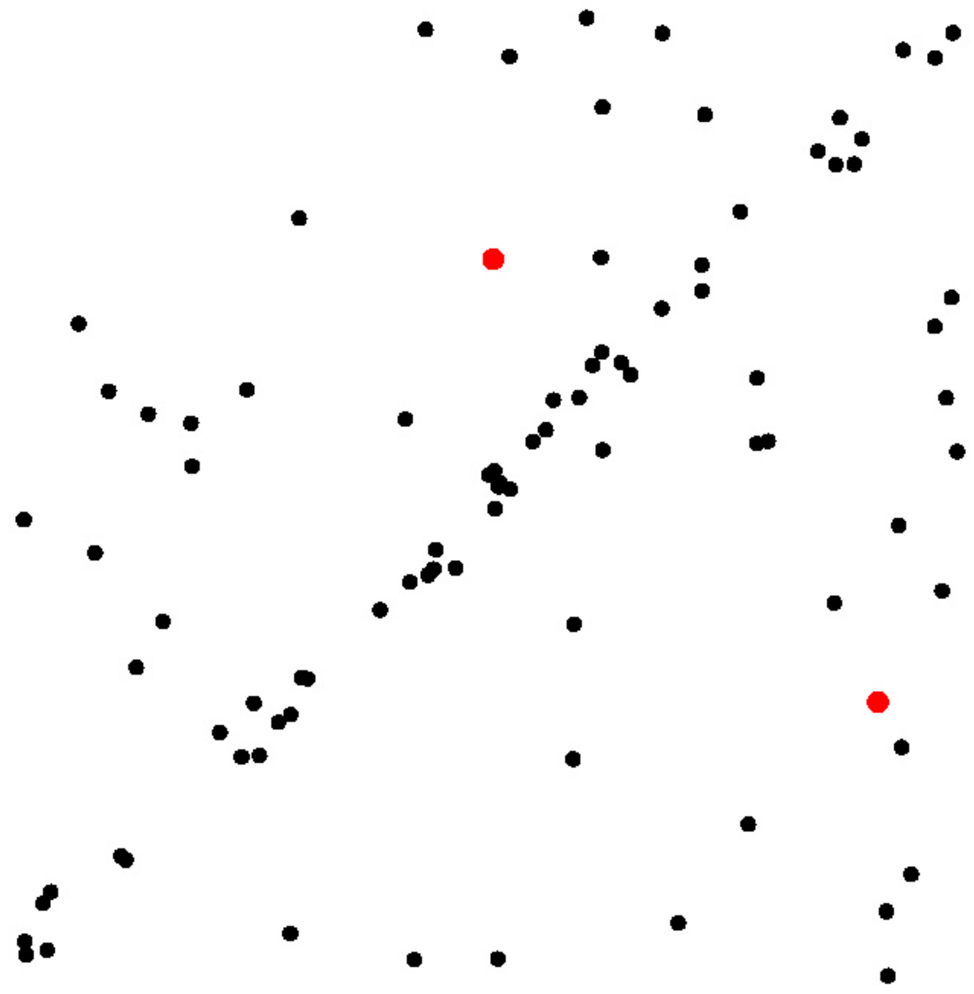
RANSAC & HOUGH TRANSFORM

Algorithm 3: RANSAC

- Acronym: **Ran**dom **Sa**mple **C**onsensus.
- Generic & robust fitting algorithm of models with outliers
  - Outliers: points which do not satisfy a model
- RANSAC: apply to any problem where:
  - identify the inliers
  - which satisfy a predefined mathematical model.
- Typical robotics applications:
  - line extraction from 2D range data (sonar or laser);
  - plane extraction from 3D range data
  - structure from motion
- RANSAC:
  - iterative method & non-deterministic
- Drawback: A nondeterministic method, results are different between runs.

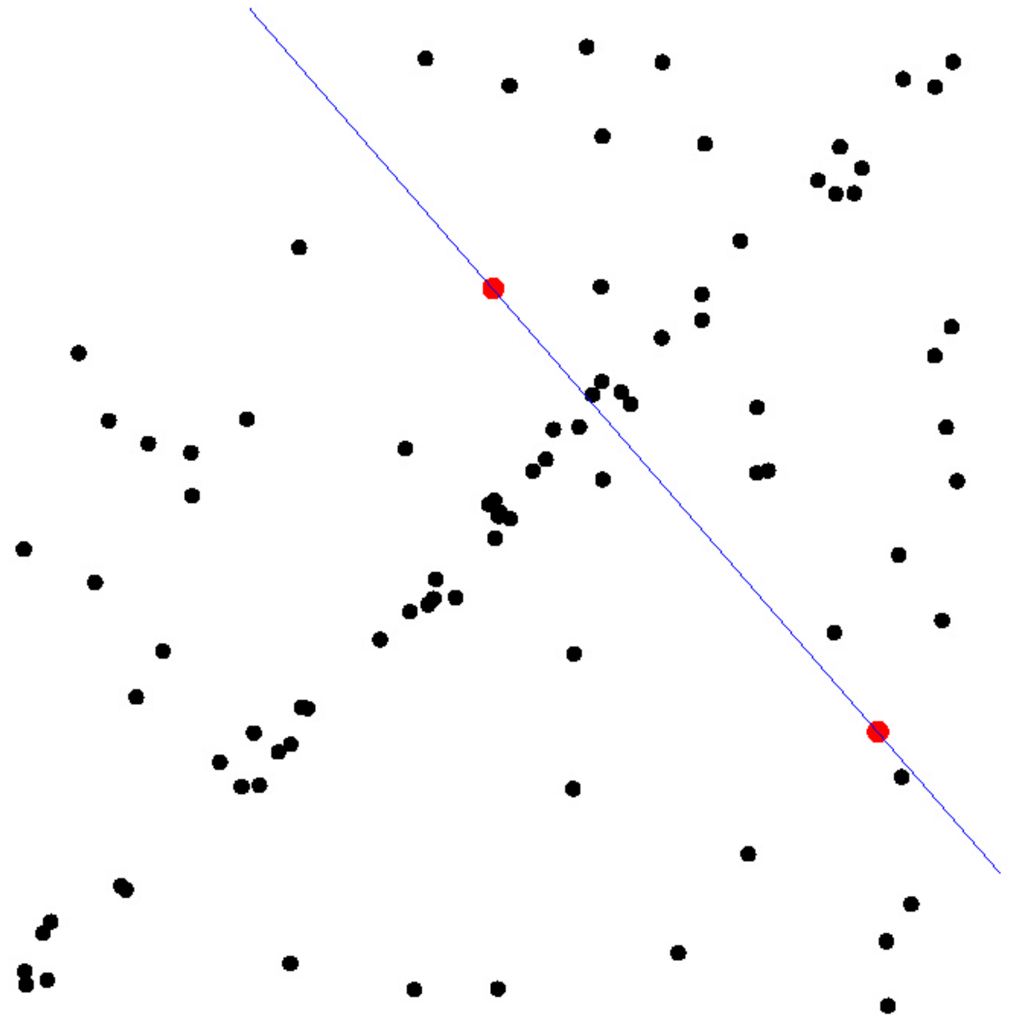
Algorithm 3: RANSAC

Algorithm 3: RANSAC



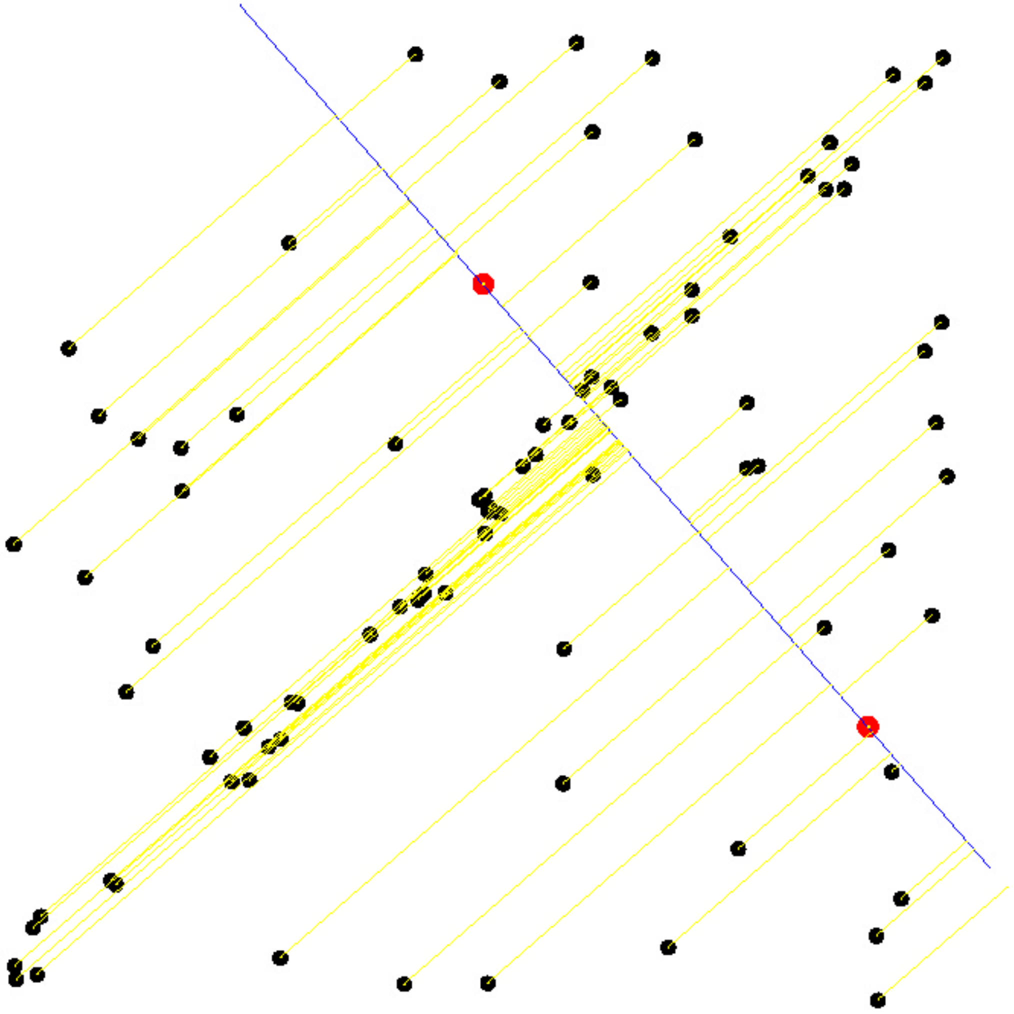
- **Select sample of 2 points at random**

Algorithm 3: RANSAC



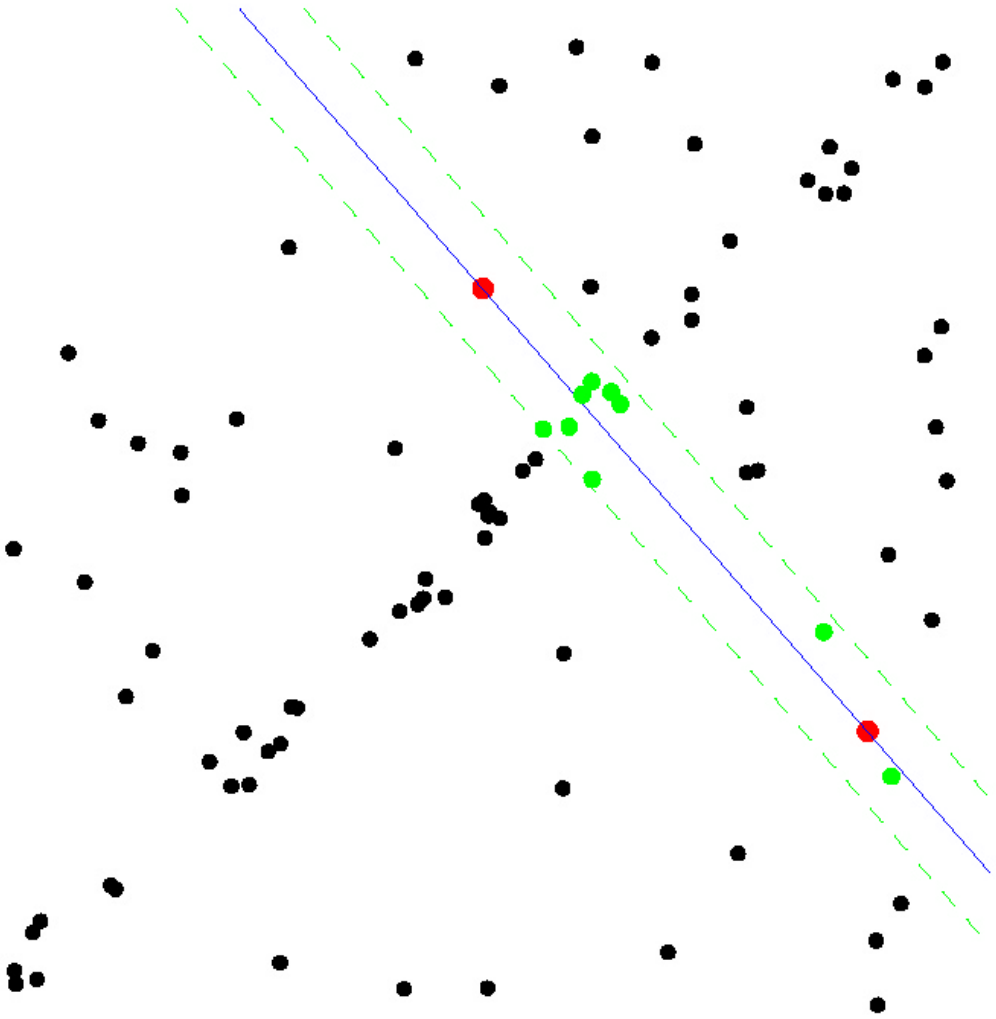
- Select sample of 2 points at random
- **Calculate model parameters that fit the data in the sample**

RANSAC



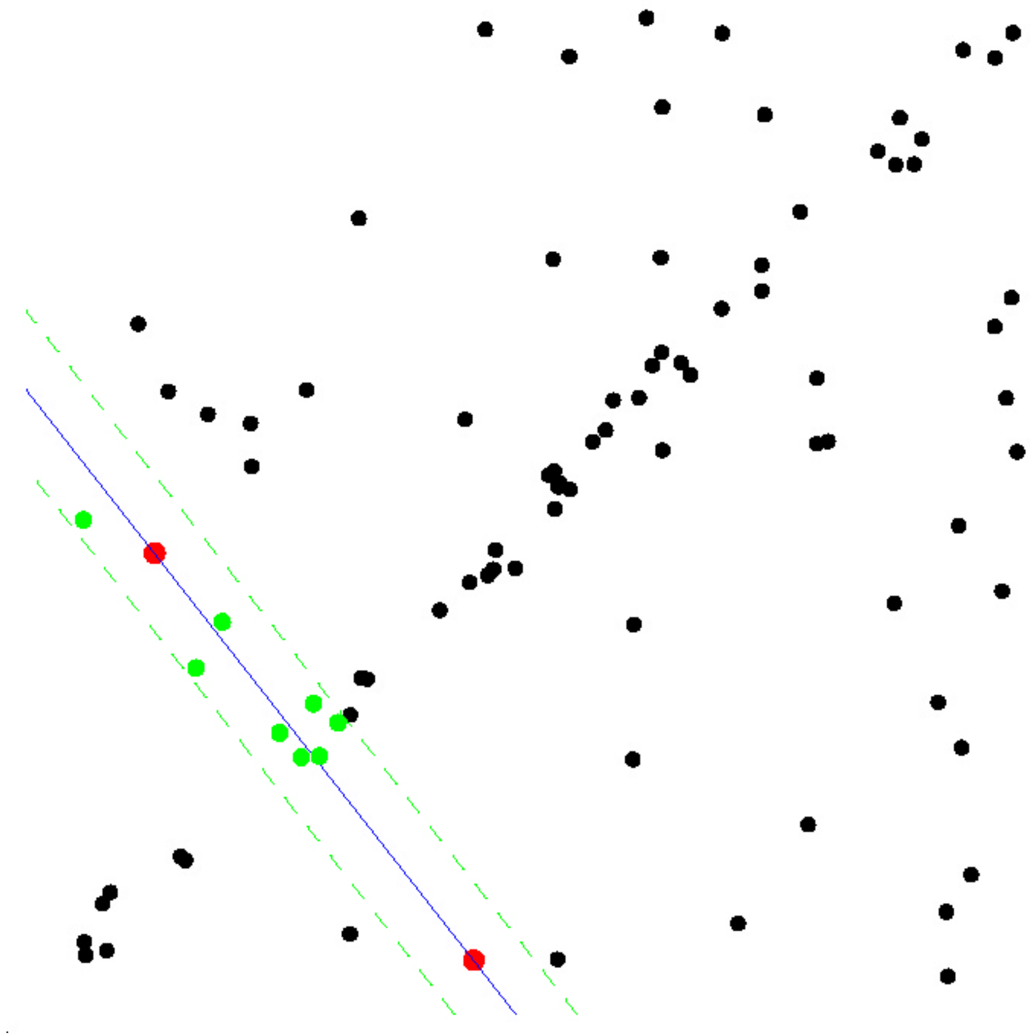
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- **Calculate error function for each data point**

Algorithm 3: RANSAC



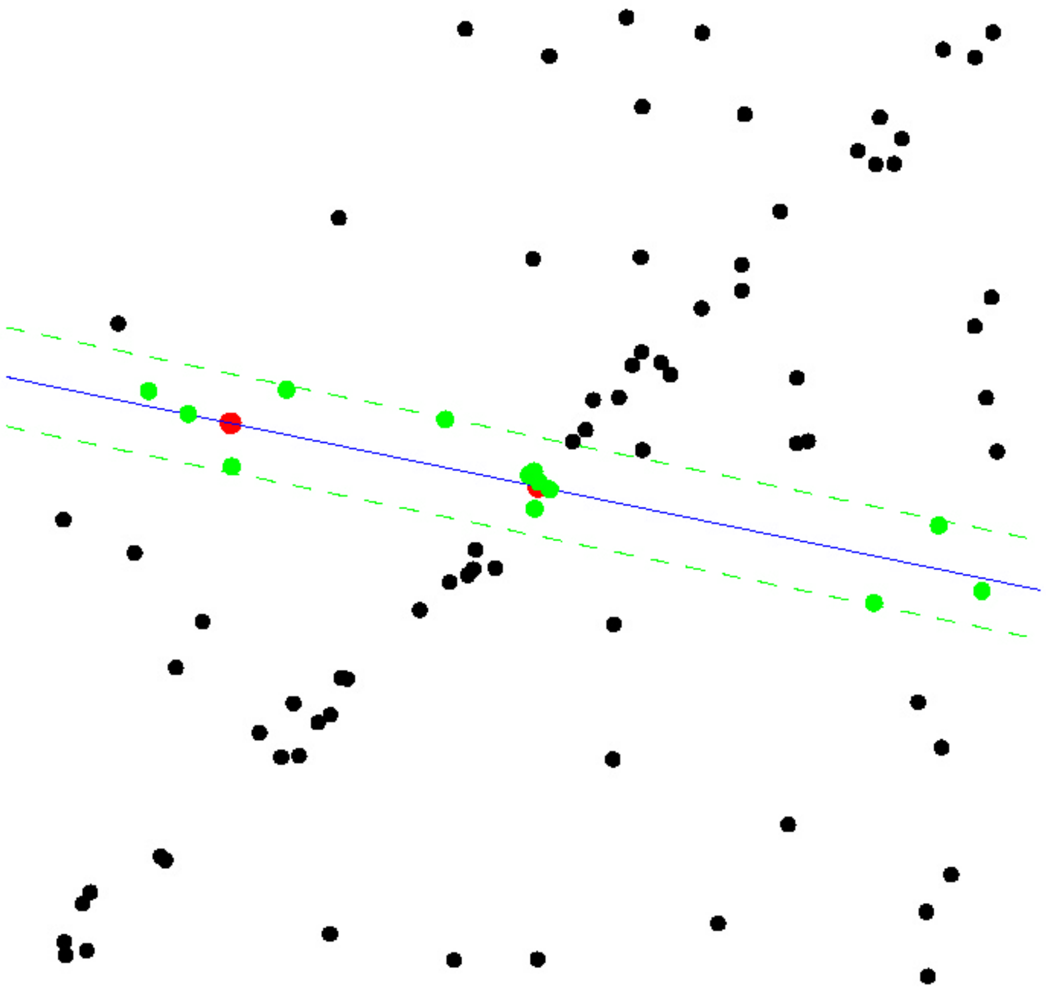
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- **Select data that support current hypothesis**

Algorithm 3: RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

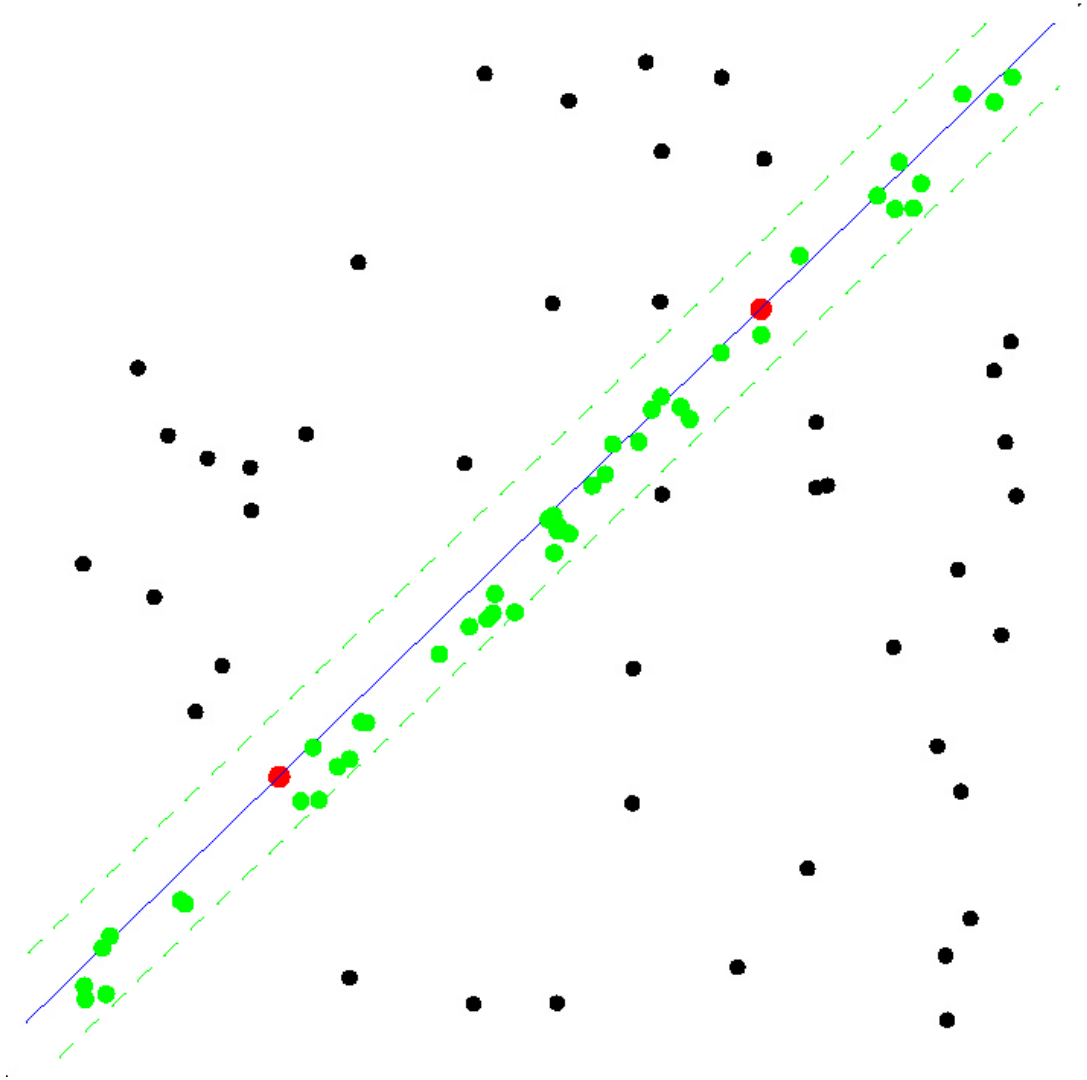
Algorithm 3: RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

Algorithm 3: RANSAC

ALL-INLIER SAMPLE



Algorithm 3: RANSAC

Algorithm 4: *RANSAC*

1. Initial: let A be a set of N points
2. **repeat**
3. Randomly select a sample of 2 points from A
4. Fit a line through the 2 points
5. Compute the distances of all other points to this line
6. Construct the inlier set (i.e. count the number of points with distance to the line $< d$)
7. Store these inliers
8. **until** Maximum number of iterations k reached
9. The set with the maximum number of inliers is chosen as a solution to the problem

Algorithm 3: RANSAC

How many iterations does RANSAC need?

- Because we cannot know in advance if the observed set contains the maximum number of inliers, the ideal would be to check all possible combinations of 2 points in a dataset of N points.

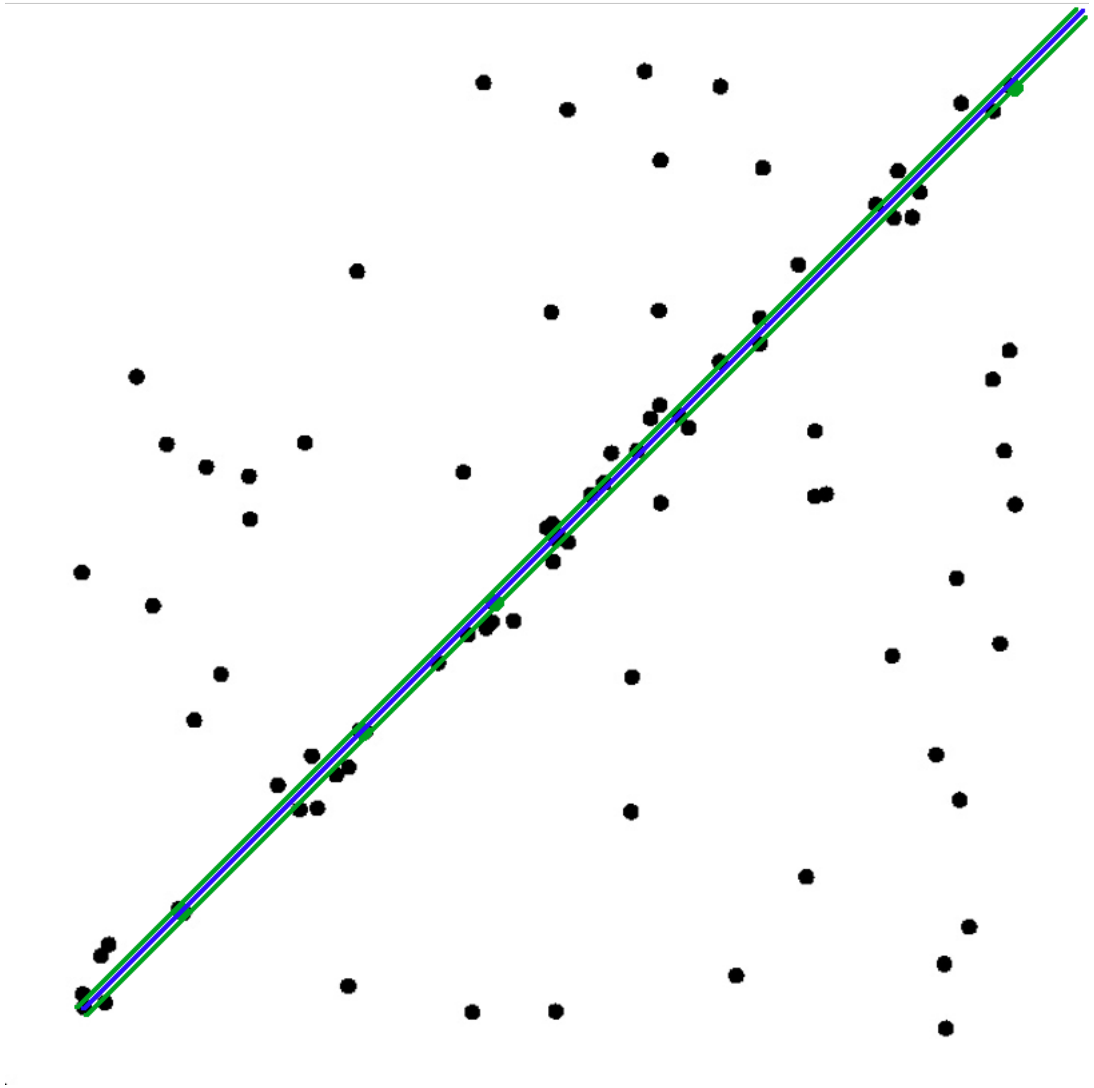
- The number of combinations is given by N(N-1)/2, which makes it computationally unfeasible if N is too large. For example, in a laser scan of 360 points we would need to check all 360*359/2= 64,620 possibilities!

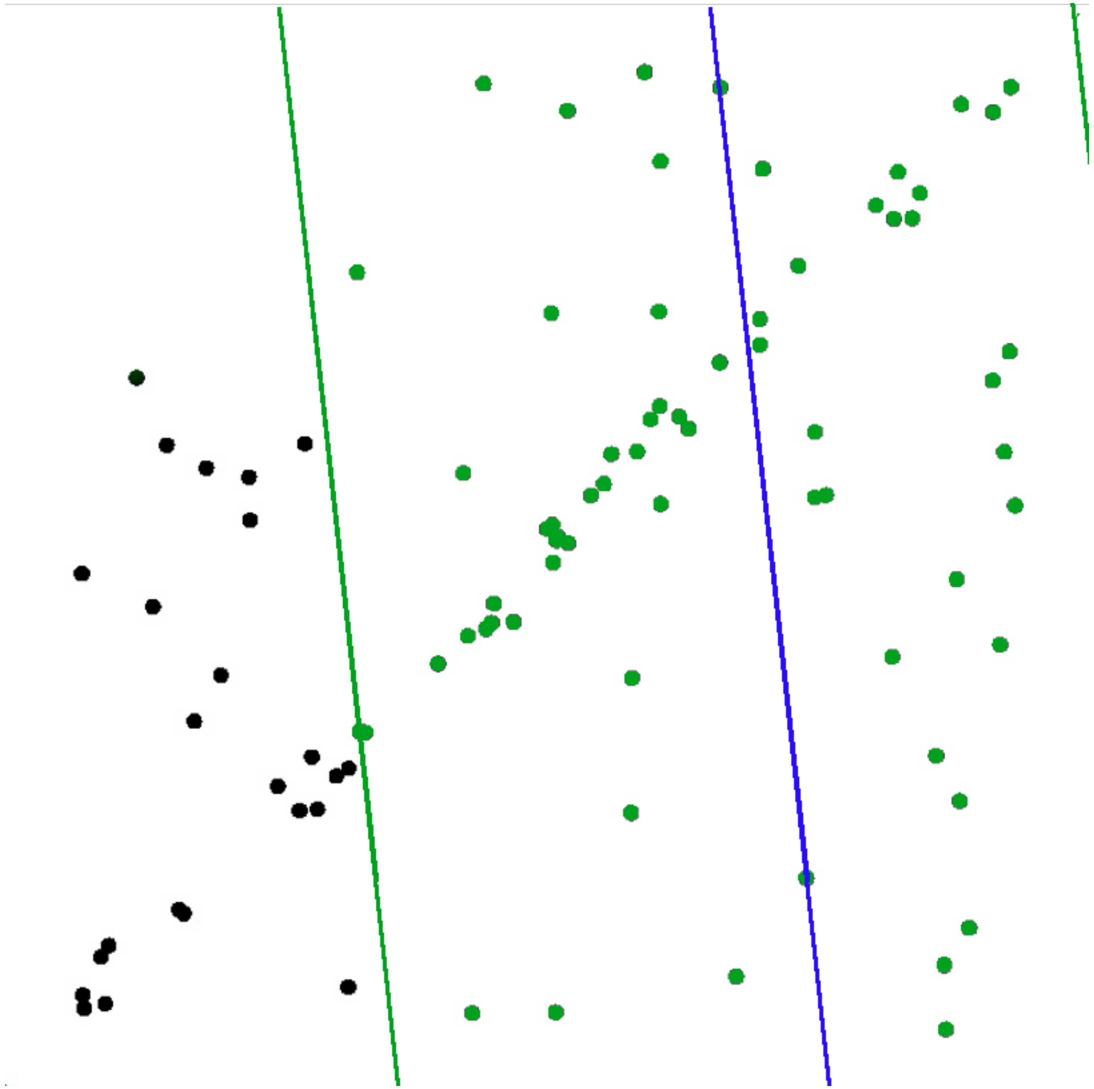
- Do we really need to check all possibilities or can we stop RANSAC after iterations? The answer is that indeed we do not need to check all combinations but just a subset of them if we have a rough estimate of the percentage of inliers in our dataset

- This can be done in a probabilistic way

RANSAC

- Need good inlier threshold





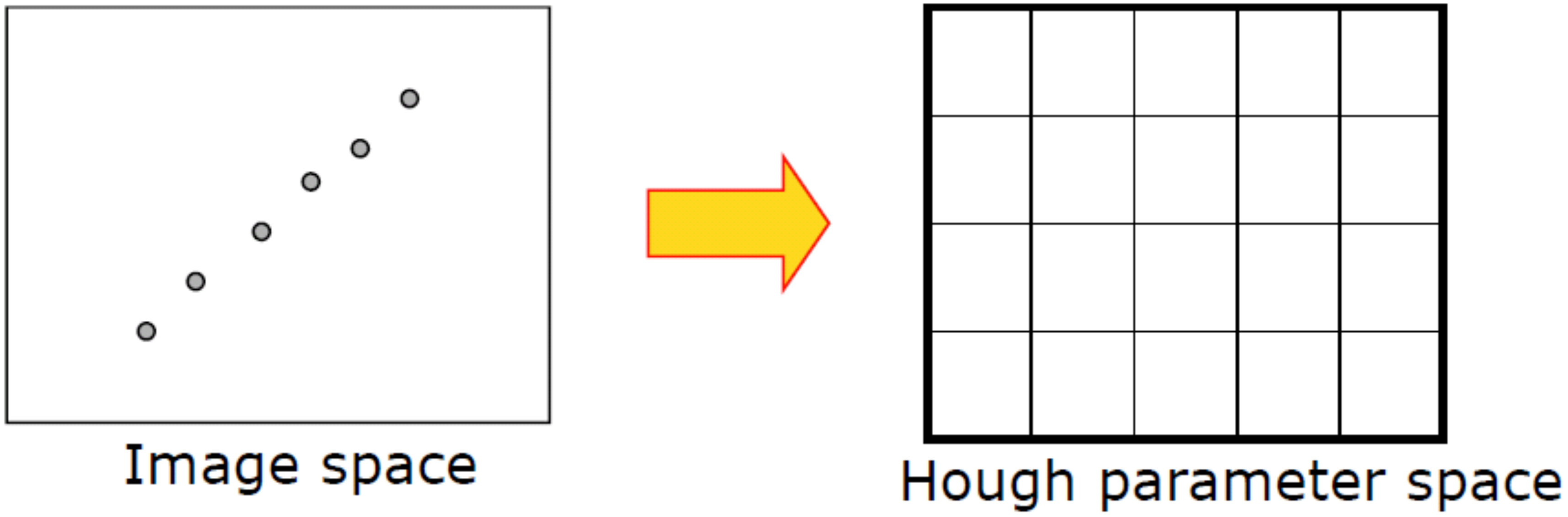
- Too narrow – very few inliers
- Too wide – too many inliers

RANSAC Applications

- Computer Vision:
  - Structure from motion
  - Estimating lines, circles, ellipses, …
  - Homography estimation
  - Video stabilization
- 3D point clouds
  - Estimating lines, planes, circles, spheres, cylinders, …
- Robotics
  - Localization (e.g. needle in ultrasound)
- Medicine/ Biology research (drug concentration prediction; phagocyte transmigration; microbial metabolomics data mining; …)
- Economy (Expert Database Retrieval; RANSAC-based method for detecting post-assembly defects in aircraft interiors; …)

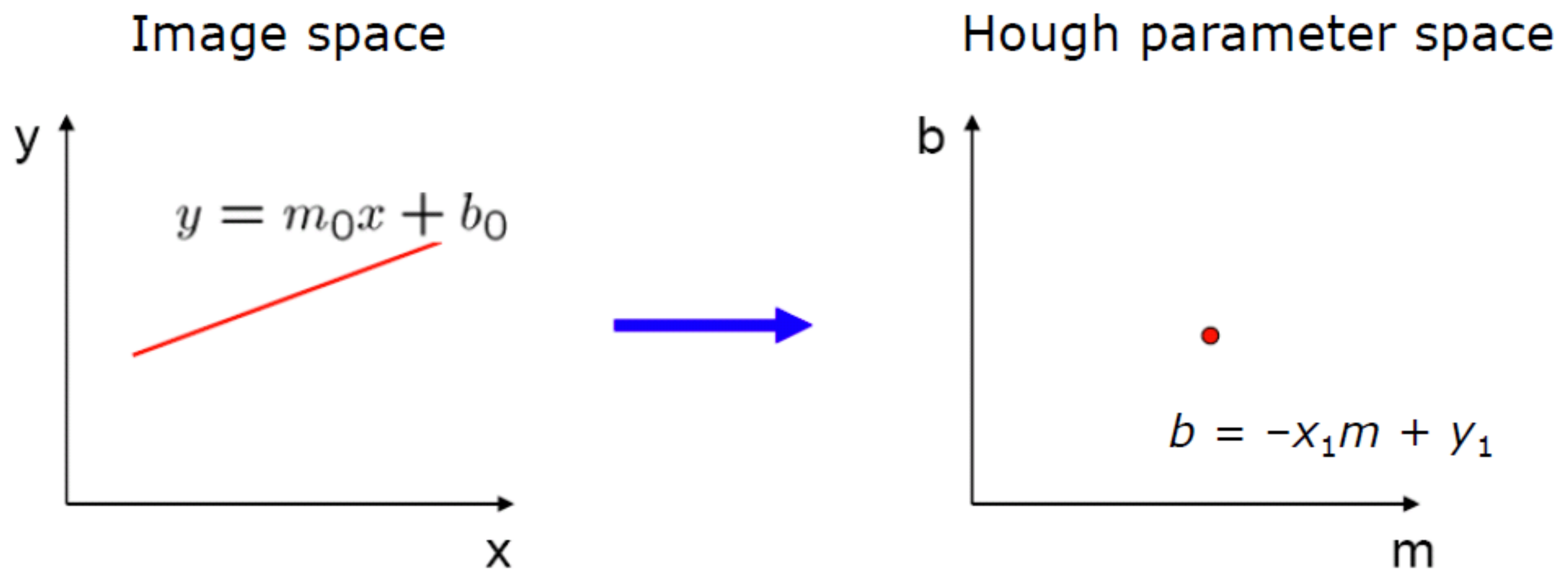
Algorithm 4: Hough-Transform

- Hough Transform uses a voting scheme



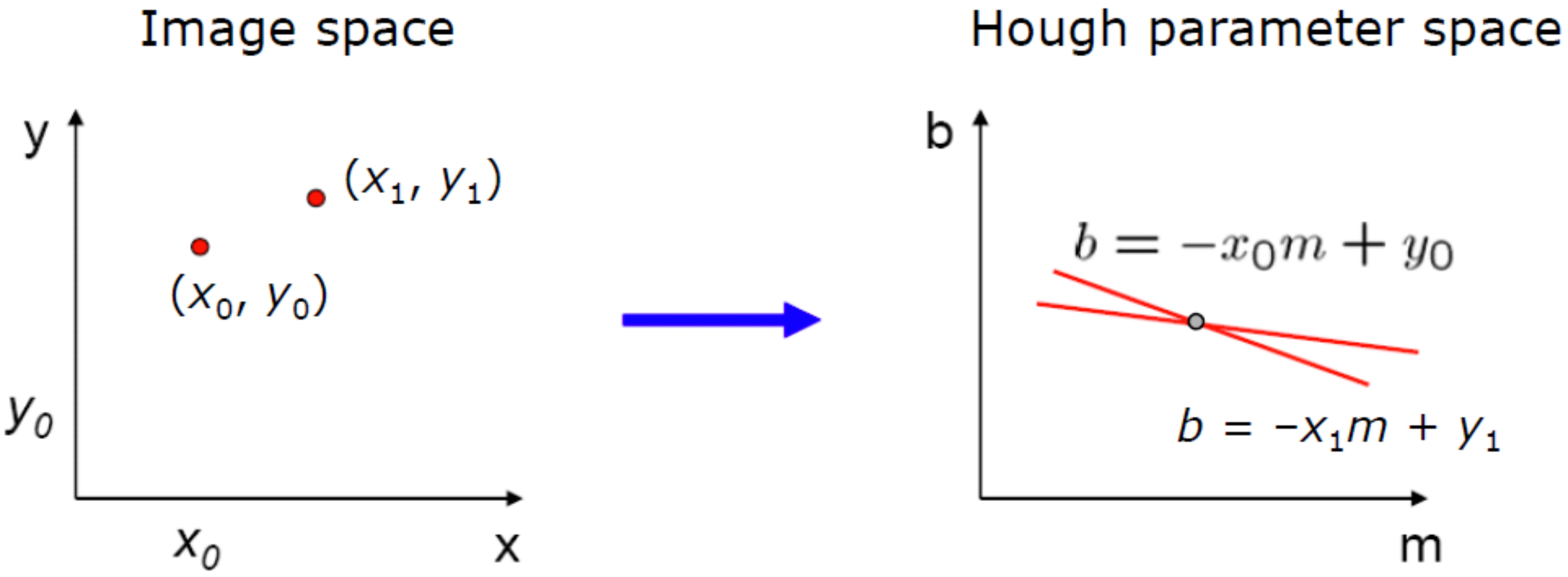
Algorithm 4: Hough-Transform

- A line in the image corresponds to a point in Hough space



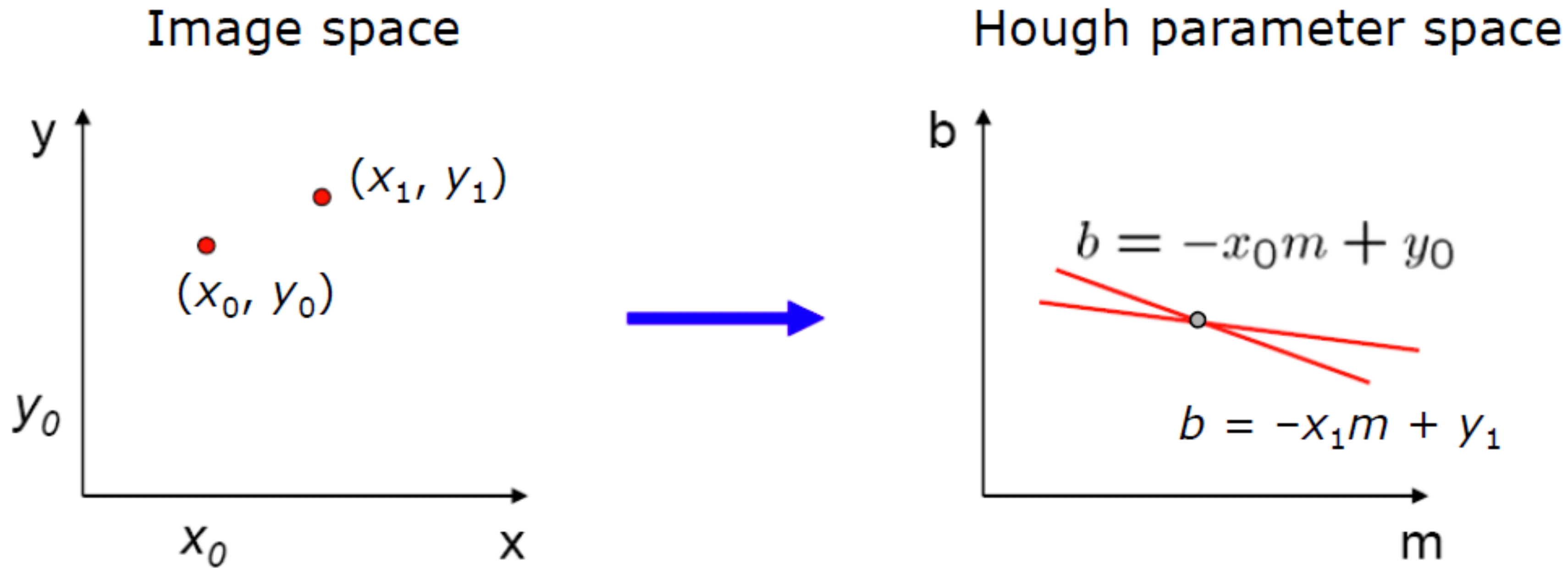
Algorithm 4: Hough-Transform

- What does a point (x_0, y_0) in the image space map to in the Hough space?



Algorithm 4: Hough-Transform

- Where is the line that contains both (x_0, y_0) and (x_1, y_1)?
  - It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$



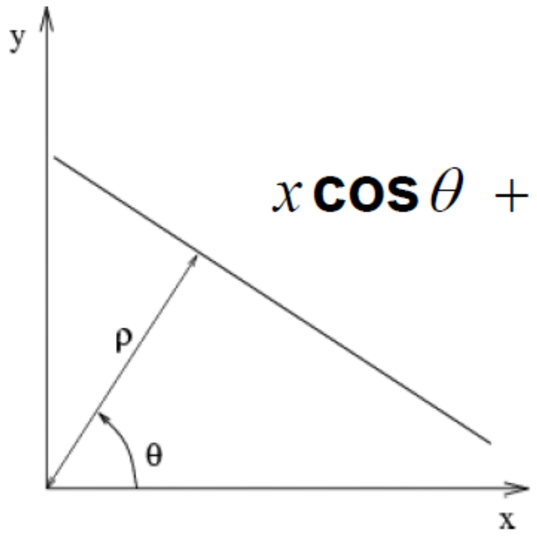
Algorithm 4: Hough-Transform

- Problems with the (m,b) space:
  - Unbounded parameter domain
  - Vertical lines require infinite m

Algorithm 4: Hough-Transform

- Problems with the (m,b) space:
  - Unbounded parameter domain
  - Vertical lines require infinite m

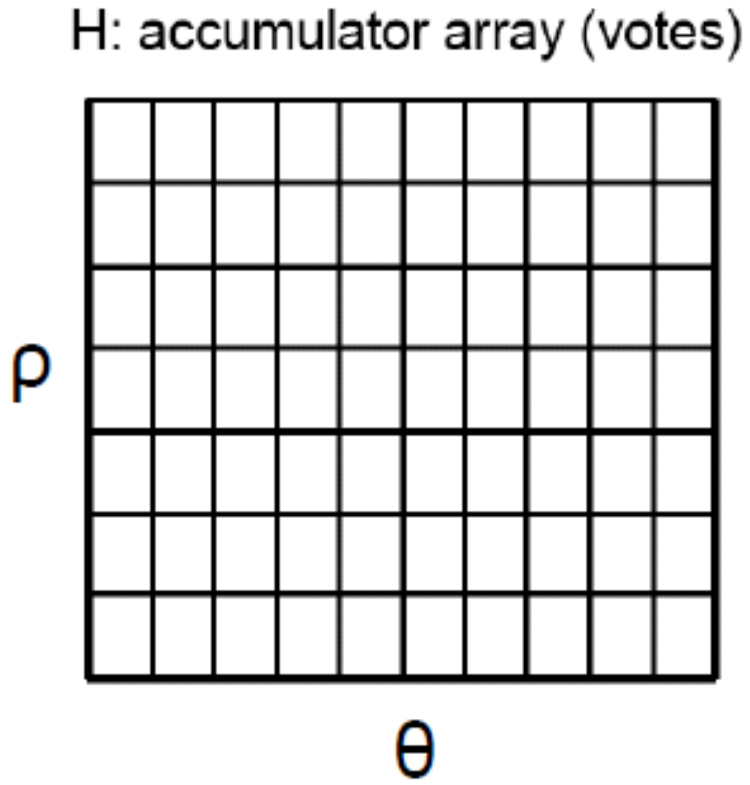
- Alternative: polar representation



$$x \cos\theta + y \sin\theta = \rho$$

Each point will add a sinusoid in the (θ,ρ) parameter space

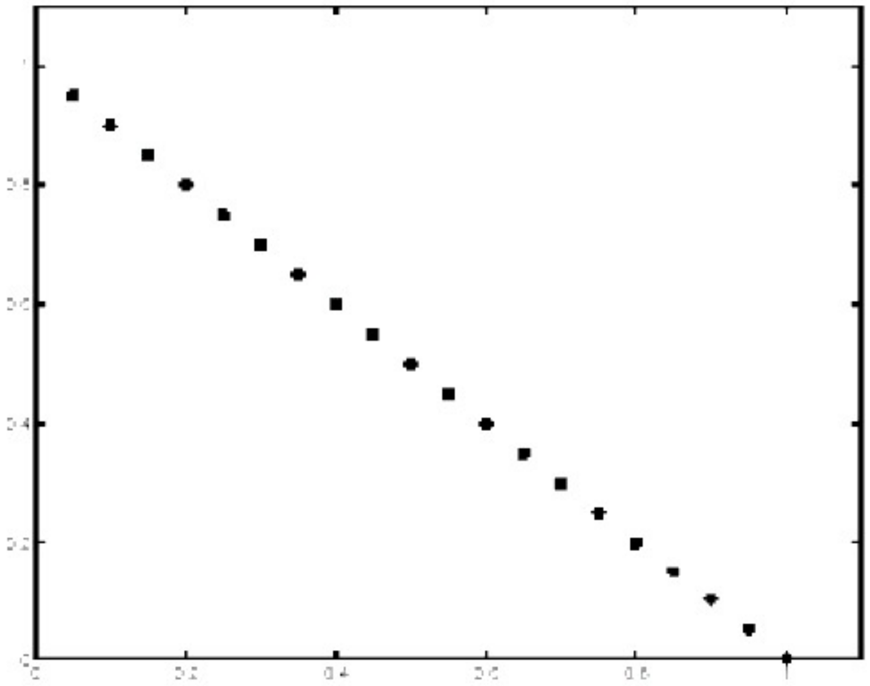
Algorithm 4: Hough-Transform



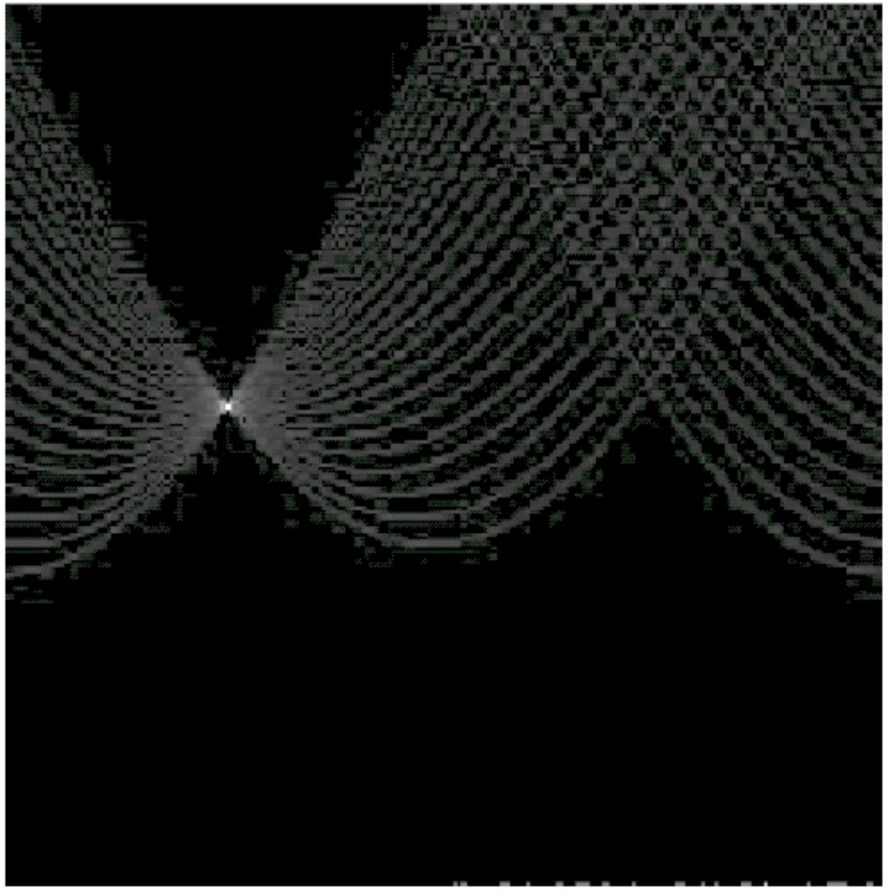
1. Initialize accumulator H to all zeros
2. For each edge point (x,y) in the image
   - For θ = 0 to 180 (with a step size of e.g. 18)
     - $\rho = x \cos \theta + y \sin \theta$
     - $H(\theta, \rho) = H(\theta, \rho) + 1$
   - end

   end
3. Find the values of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
4. The detected line in the image is given by $\rho = x \cos \theta + y \sin \theta$

Algorithm 4: Hough-Transform



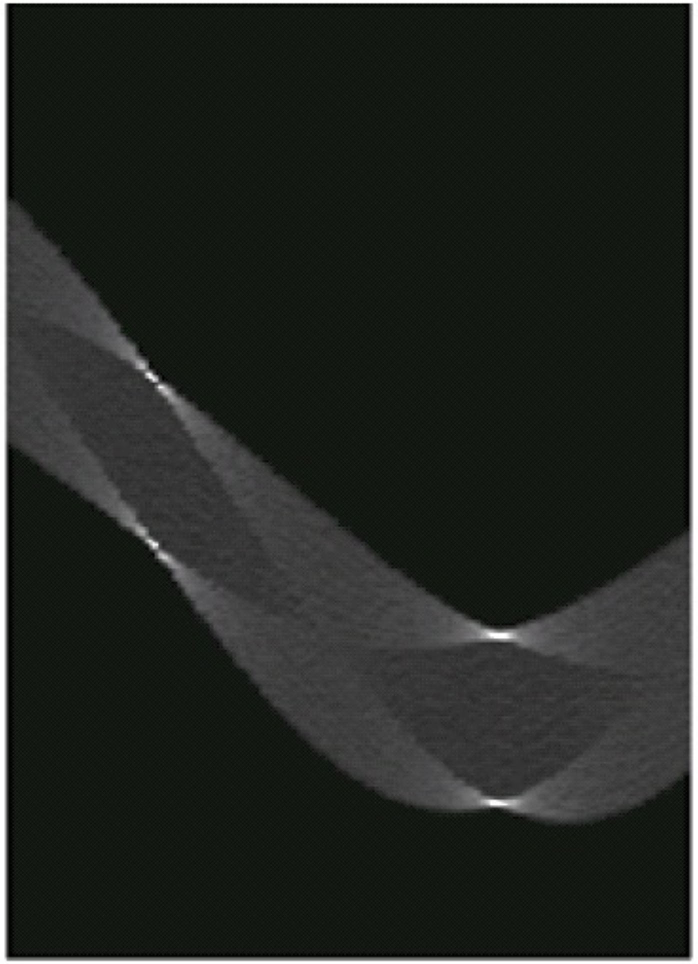
features



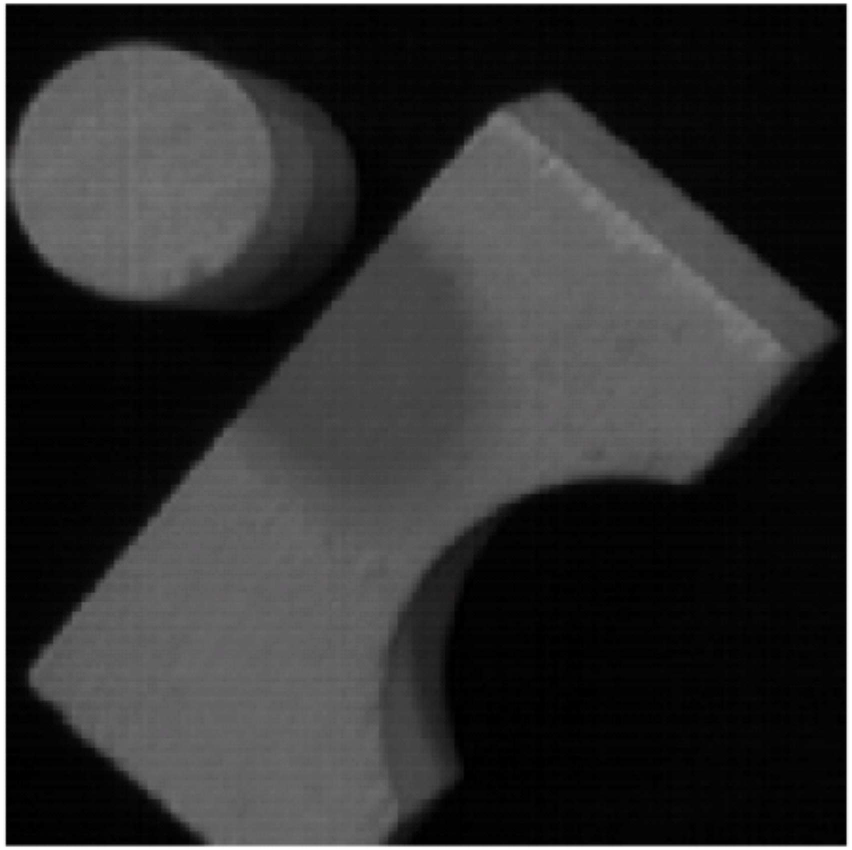
votes

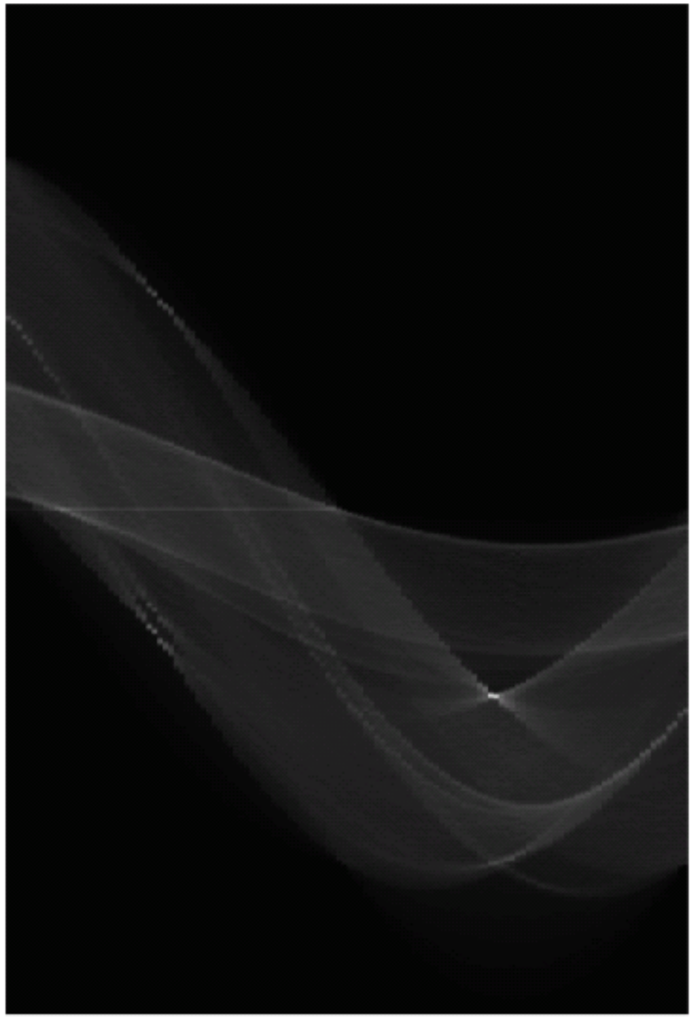
Algorithm 4: Hough-Transform

Square



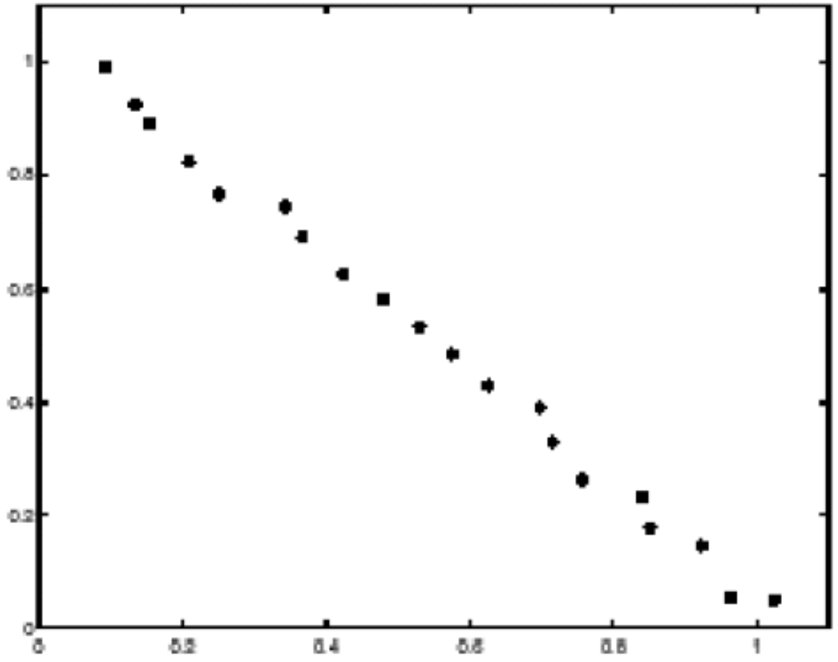
Algorithm 4: Hough-Transform



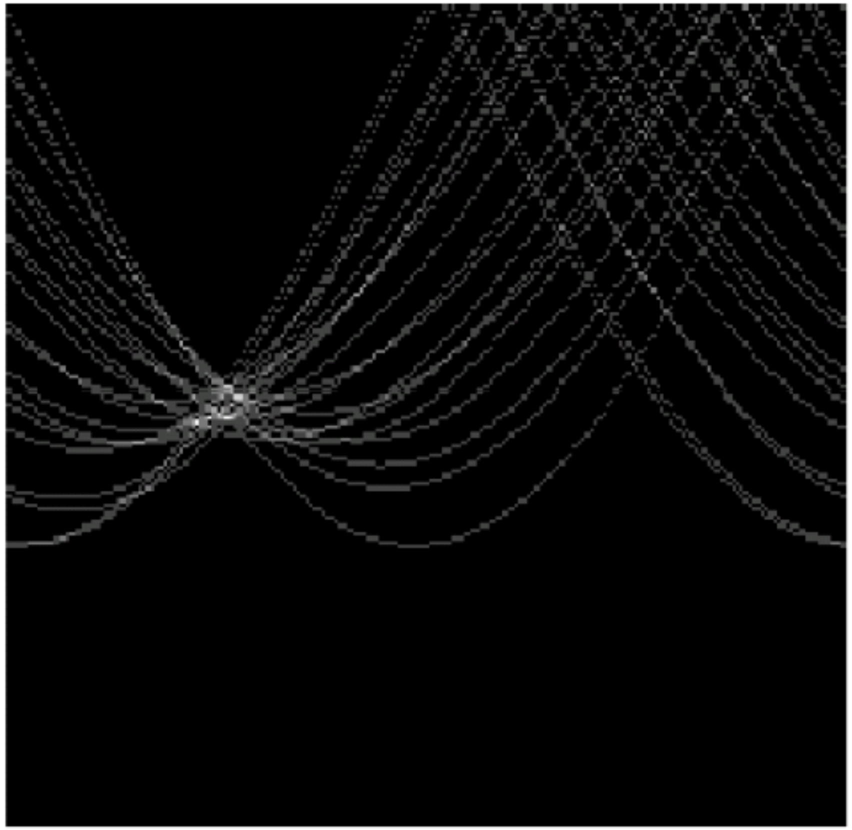


Algorithm 4: Hough-Transform

Effect of Noise



features



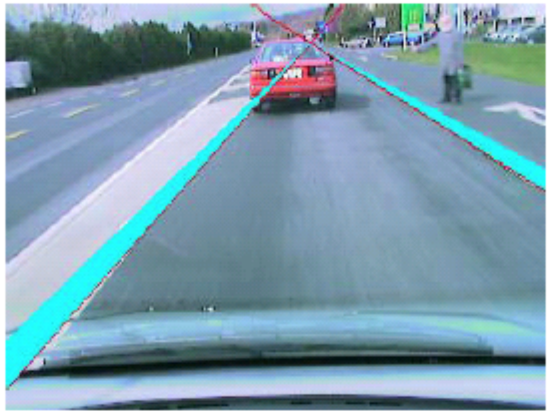
votes

- Peak gets fuzzy and hard to locate

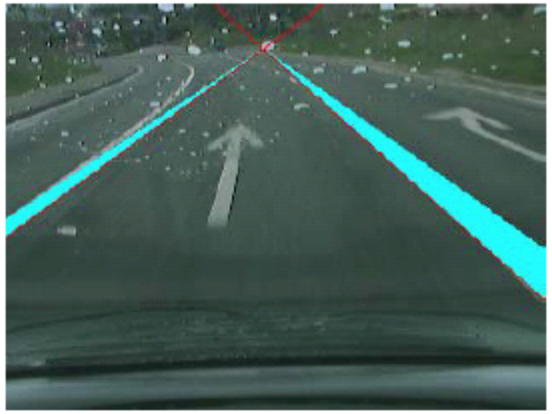
Algorithm 4: Hough-Transform

Application: Lane detection

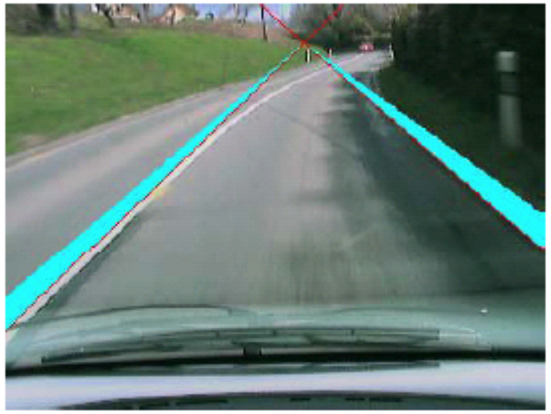
Inner city traffic



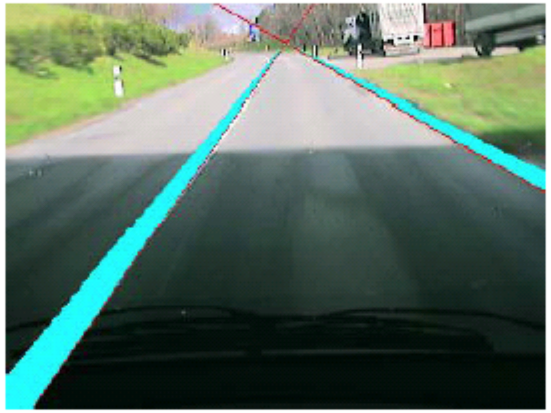
Ground signs



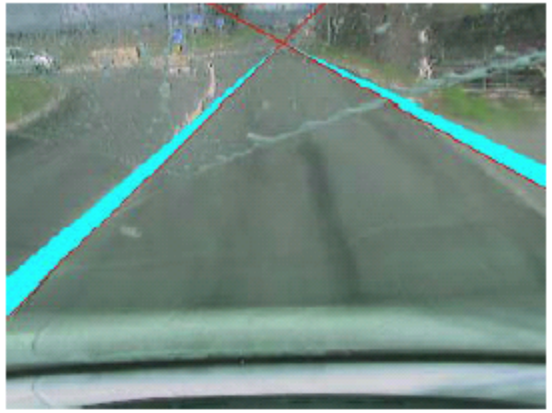
Country-side lane



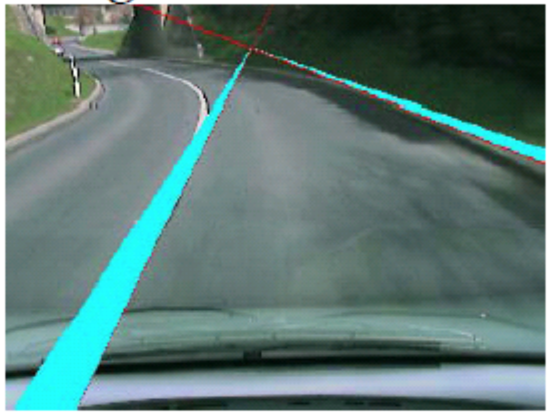
Tunnel exit



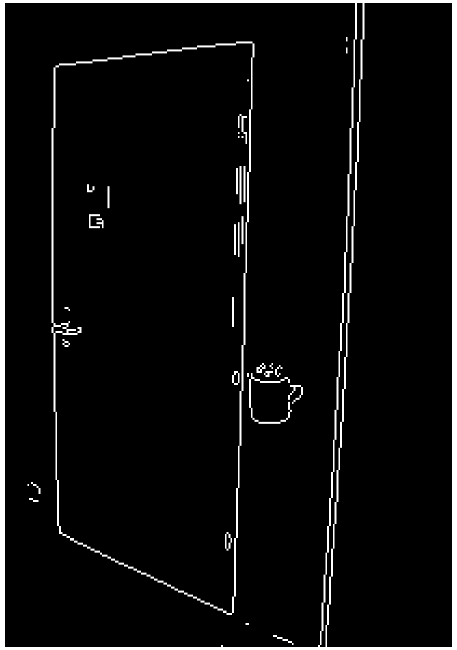
Obscured windscreen

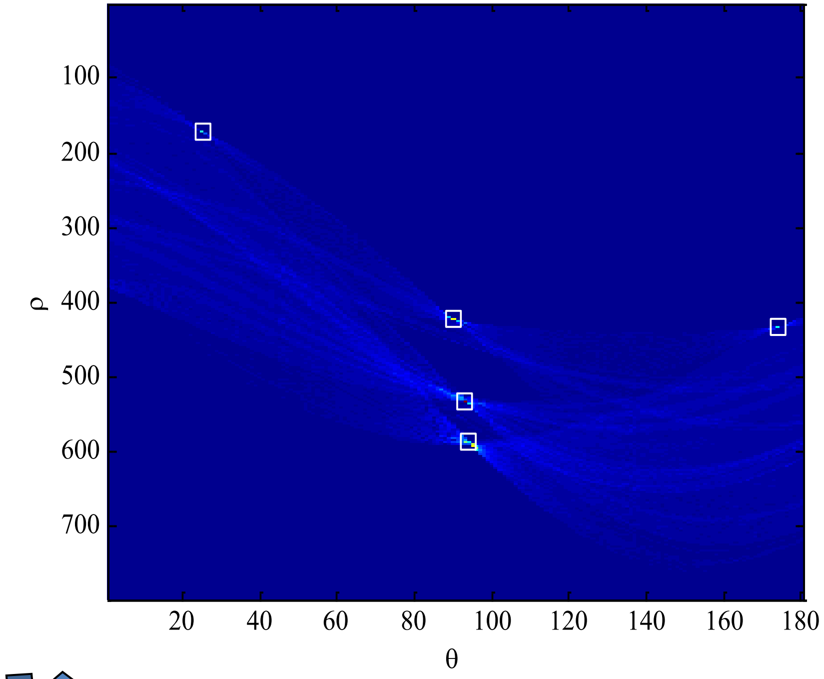


High curvature

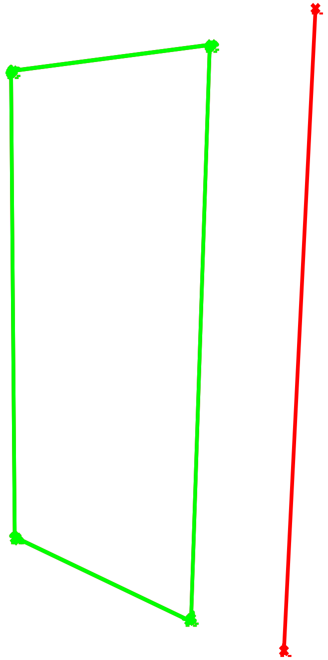


Example – Door detection using Hough Transform





Hough Transform



Hough Transform: other features

Lines:

$$p = (d, \upsilon)$$

$$g(x,y,p) := x \cdot \cos(\upsilon) + y \cdot \sin(\upsilon) - d$$

Circles:

$$p = (x_0, y_0, r)$$

$$g(x,y,p) := (x - x_0)^2 + (y - y_0)^2 - r^2$$

Ellipses:

$$p = (x_0, y_0, a, b, \psi)$$

$$g(x,y,p) := \frac{\left[(x - x_0)\cdot\cos(\psi) + (y - y_0)\cdot\sin(\psi)\right]^2}{a^2} + \frac{\left[(y - y_0)\cdot\cos(\psi) - (x - x_0)\cdot\sin(\psi)\right]^2}{b^2} - 1$$

Hough Transform

- Advantages
  - Noise and background clutter do not impair detection of local maxima
  - Partial occlusion and varying contrast are minimized
- Negatives
  - Requires time and space storage that increases exponentially with the dimensions of the parameter space

Comparison **Line Detection**

- Deterministic methods perform better with laser scans
  - Split-and-merge, Line-Regression, Hough transform
  - Make use of the sequencing property of scan points.
- Nondeterministic methods can produce high False Positives
  - RANSAC
  - Does not use the sequencing property
  - But it can cope with outliers
- Overall:
  - Split-and-merge is the fastest, best real-time application

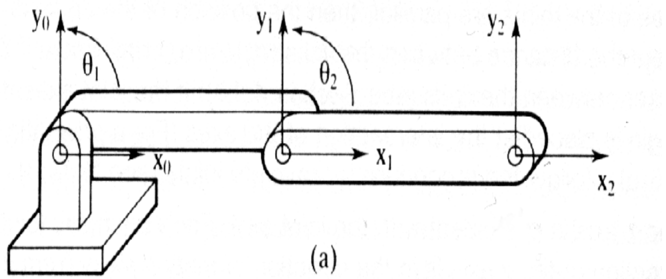
ROBOT ARMS

Robot Arm

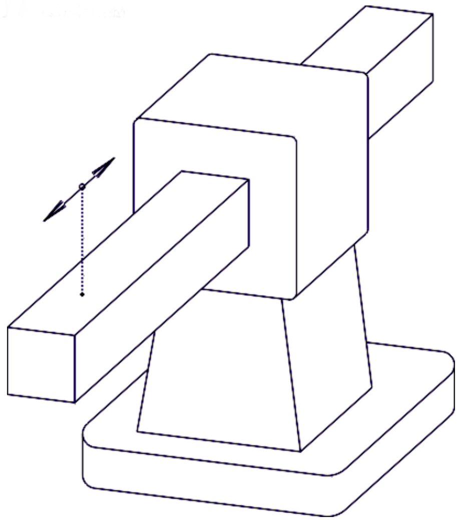
- Consists of Joints and Links …
- and a Base and a Tool (or End-Effector or Tip)

Joints

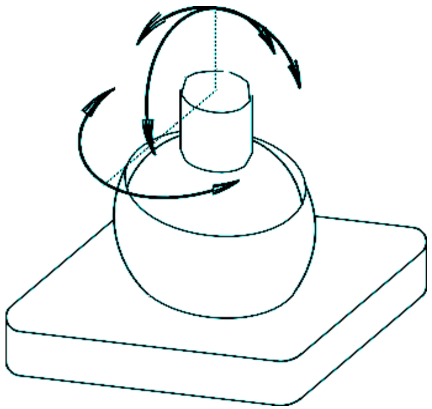
- Revolute Joint: 1DOF



- Prismatic Joint/ Linear Joint: 1DOF



- Spherical Joint: 3DOF



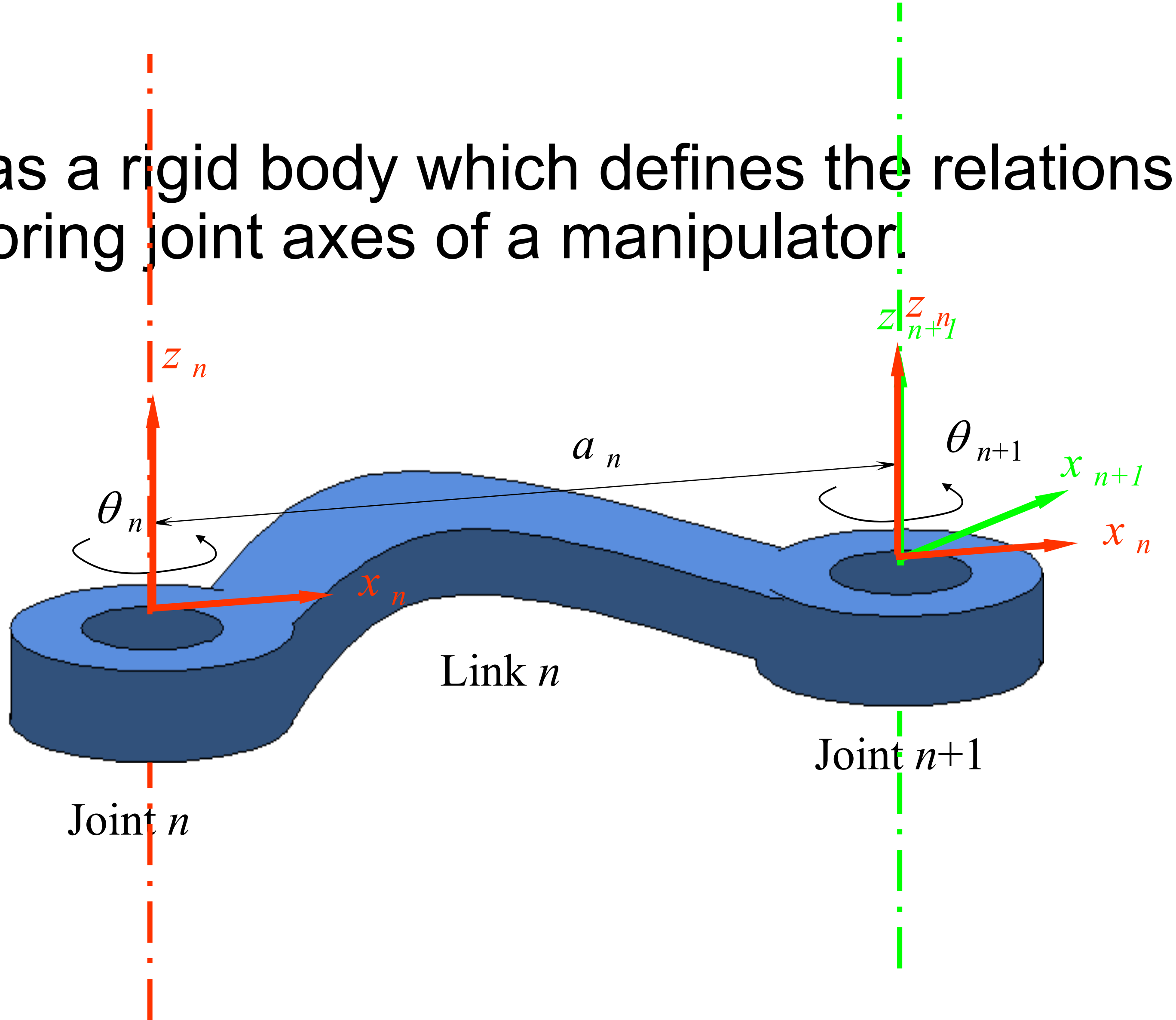
Note on Joints

- Without loss of generality, we will consider only manipulators which have joints with a single degree of freedom.

- A joint having n degrees of freedom can be modeled as n joints of one degree of freedom connected with n-1 links of zero length.

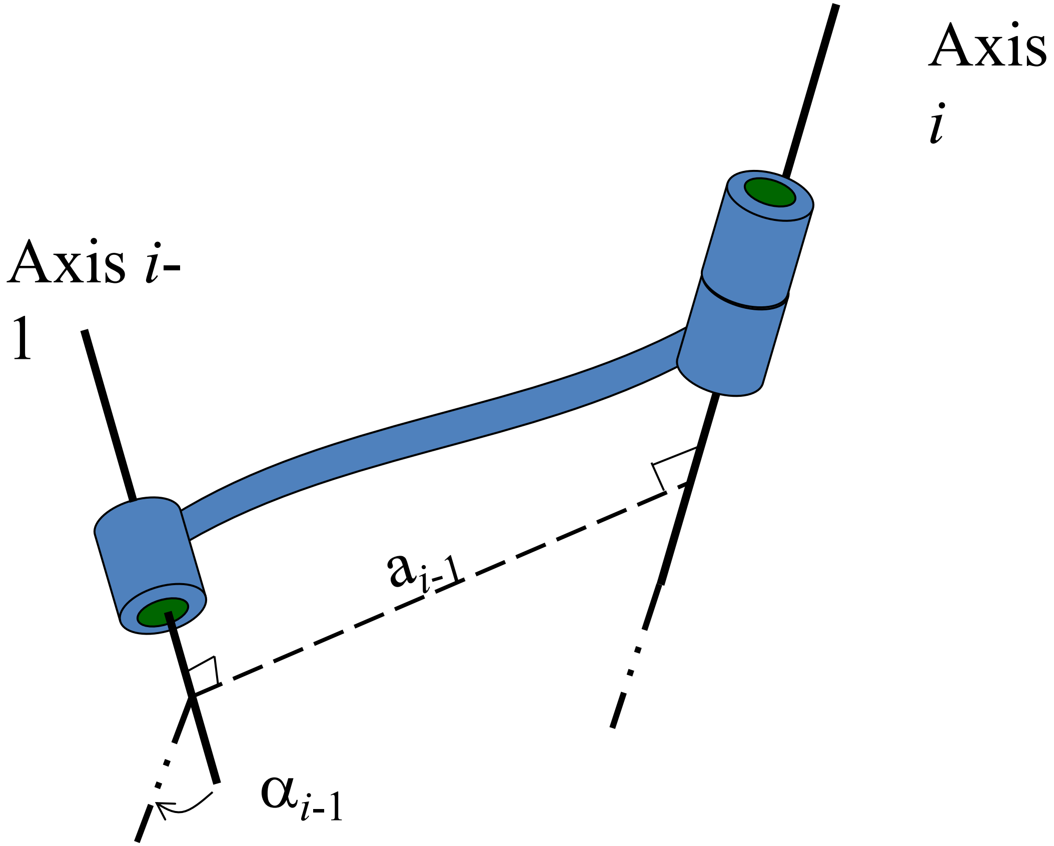
Link

- A link is considered as a rigid body which defines the relationship between two neighboring joint axes of a manipulator.



The Kinematics Function of a Link

- The kinematics function of a link is to maintain a fixed relationship between the two joint axes it supports.
- This relationship can be described with two parameters: the link length a, the link twist a



Link Length

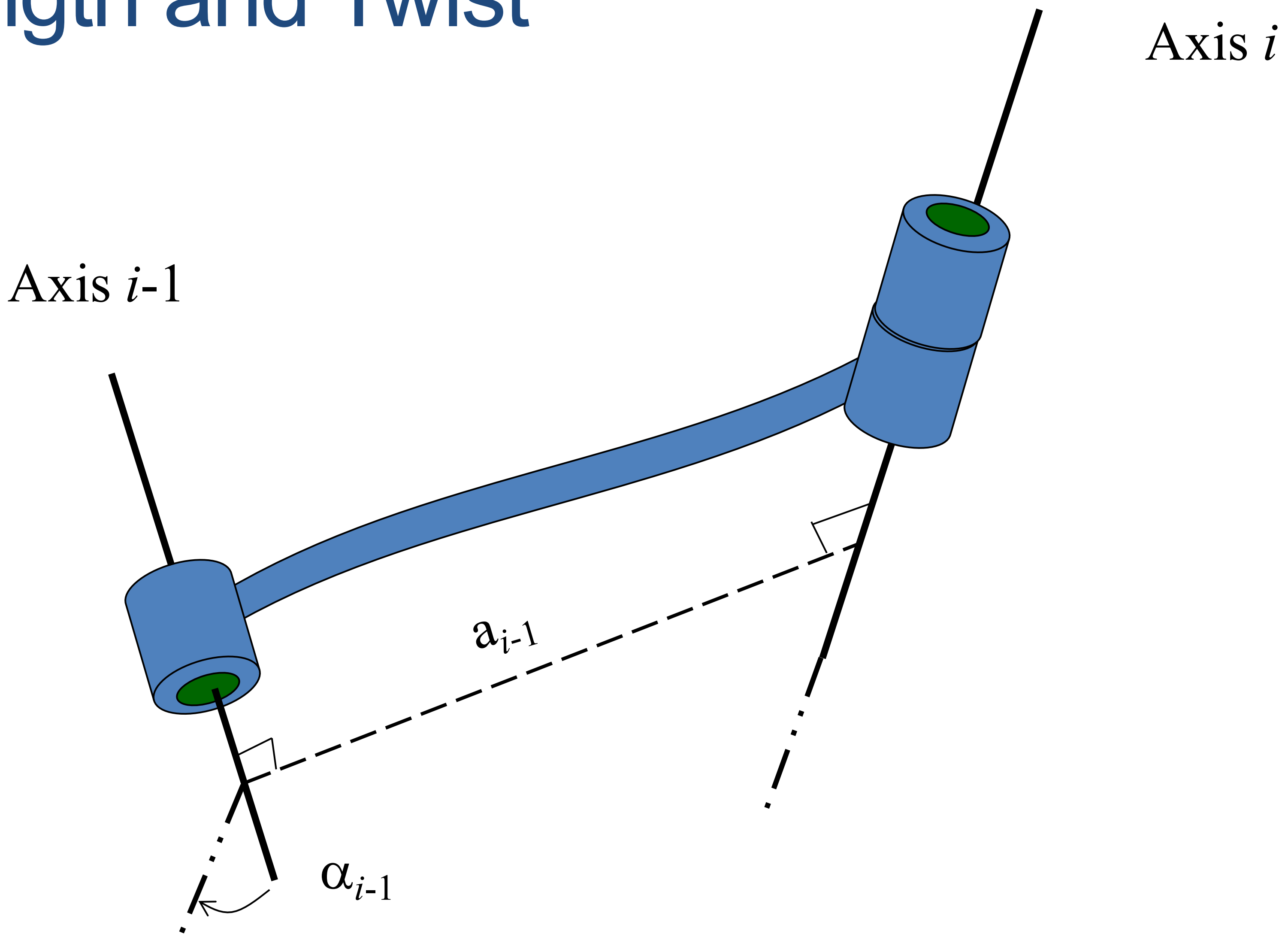
- Is measured along a line which is mutually perpendicular to both axes.
- The mutually perpendicular always exists and is unique except when both axes are parallel.

Link Twist

- Project both axes i-1 and i onto the plane whose normal is the mutually perpendicular line, and measure the angle between them

- Right-hand coordinate system

Link Length and Twist

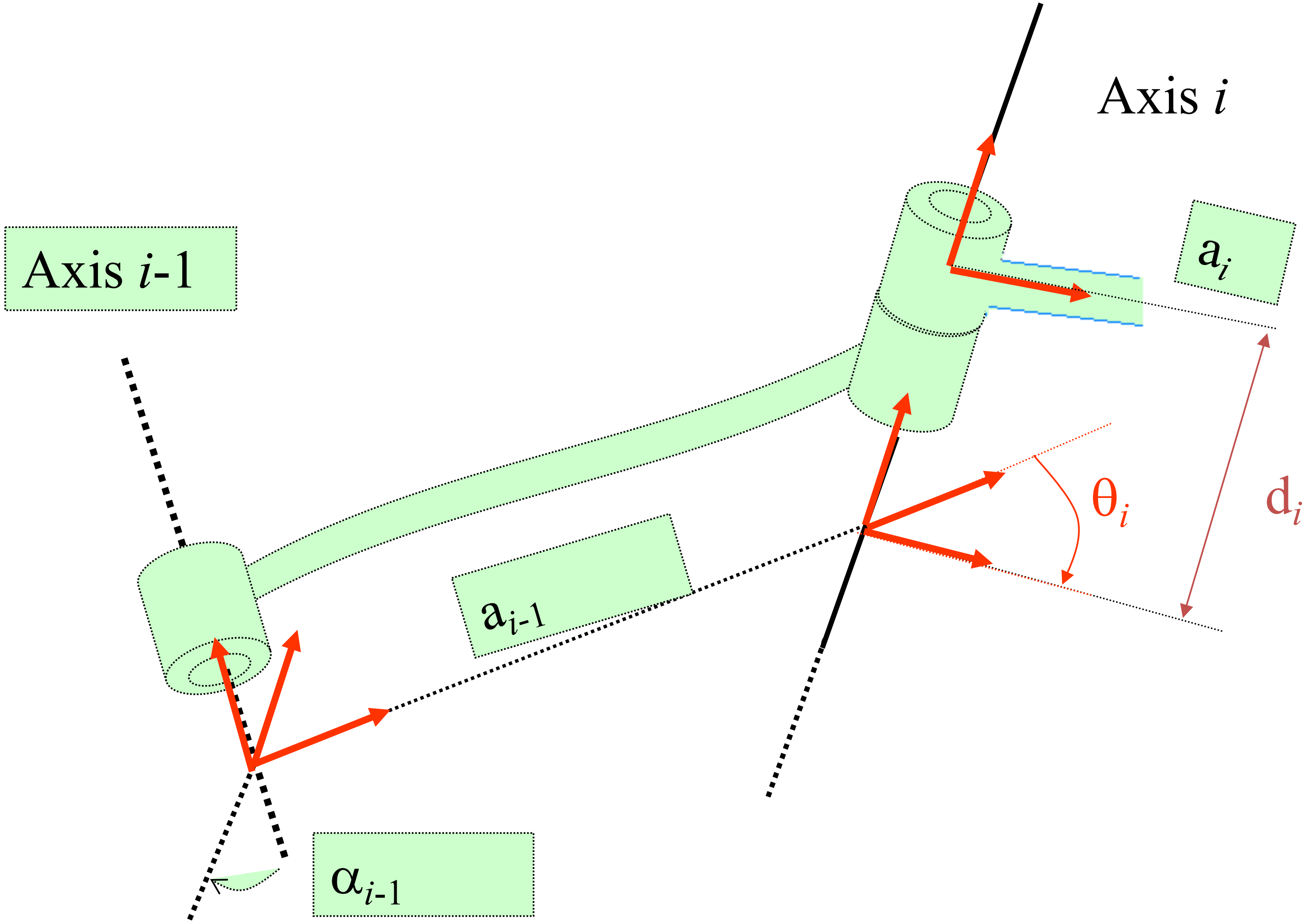


Joint Parameters (the Denavit-Hartenberg Link Parameters)

A joint axis is established at the connection of two links. This joint will have two normals connected to it one for each of the links.

- The relative position of two links is called *link offset* d_n which is the distance between the links (the displacement, along the joint axes between the links).
- The *joint angle* θ_n between the normals is measured in a plane normal to the joint axis.

Link and Joint Parameters



Link and Joint Parameters

4 parameters are associated with each link. You can align the two axis using these parameters.

- Link parameters:

a_n the length of the link.

α_n the twist angle between the joint axes.

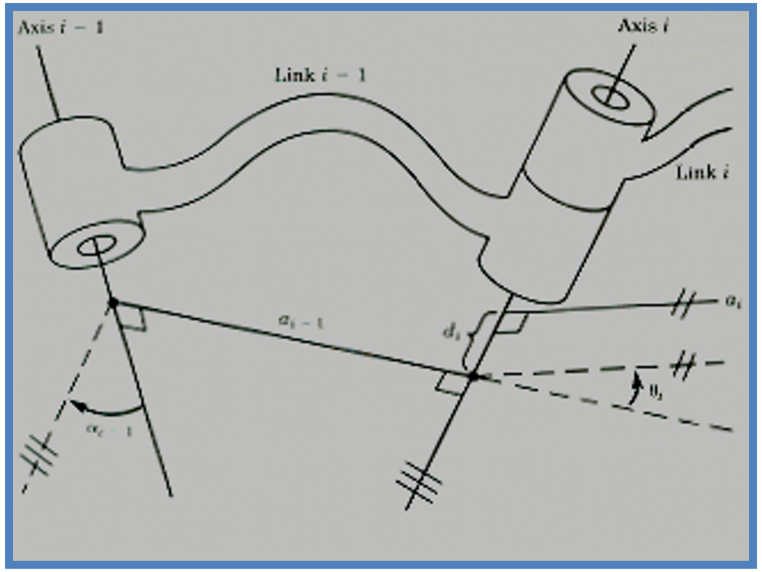
- Joint parameters:

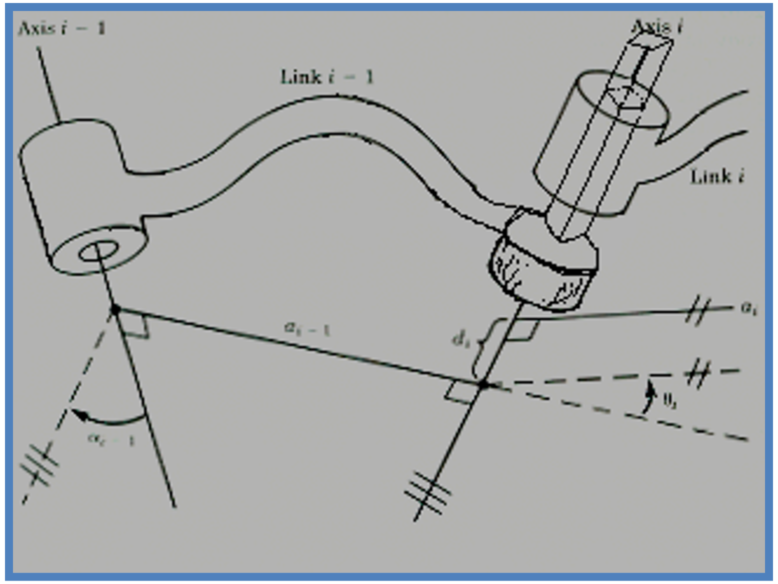
θ_n the angle between the links.

d_n the distance between the links

Link Connection Description:

For Revolute Joints: a, α, and d. are all fixed, then "θ_i" is the. Joint Variable.



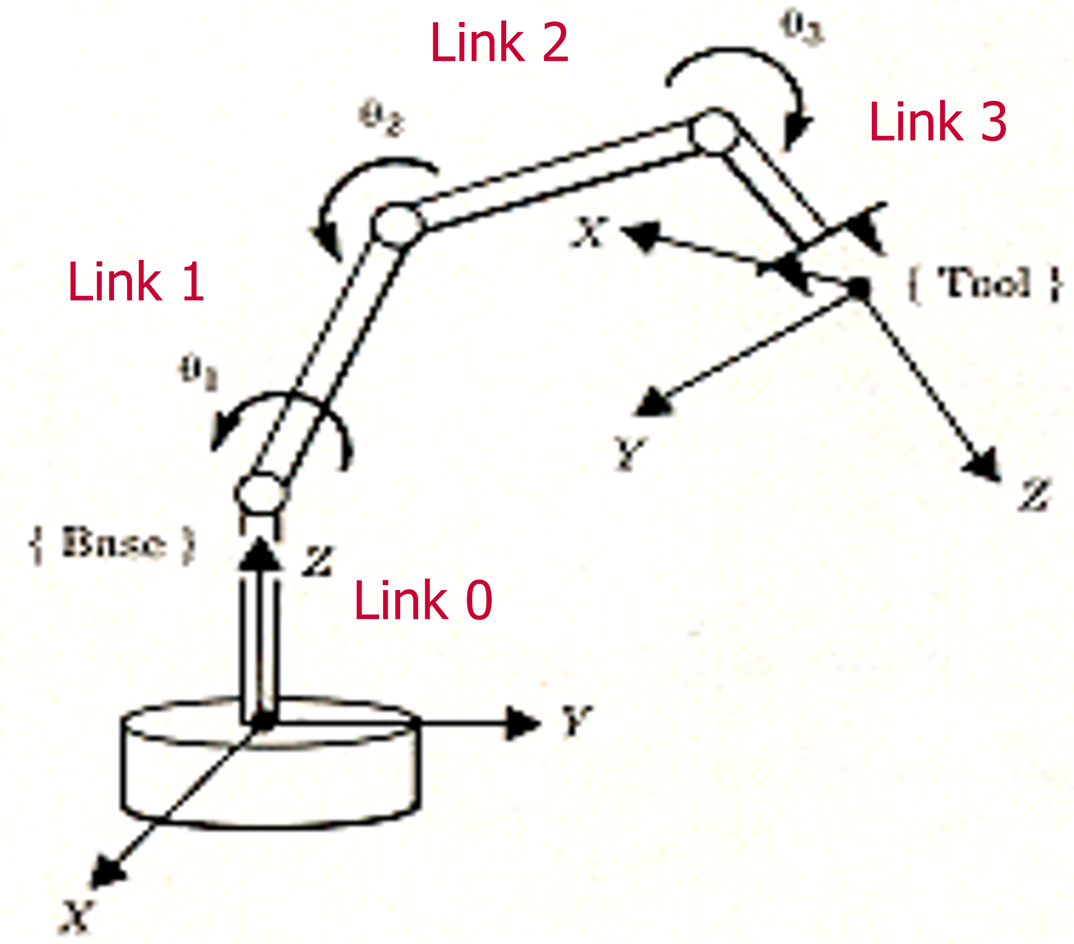


For Prismatic Joints: a, α, and θ. are all fixed, then "d_i" is the. Joint Variable.

These four parameters: (Link-Length a_{i-1}), (Link-Twist α_{i-1}), (Link-Offset d_i), (Joint-Angle θ_i) are known as the *Denavit-Hartenberg Link Parameters*.

Links Numbering Convention

Base of the arm: **Link-0**
1st moving link: **Link-1**
. .
. .
. .
Last moving link: **Link-n**



A 3-DOF Manipulator Arm

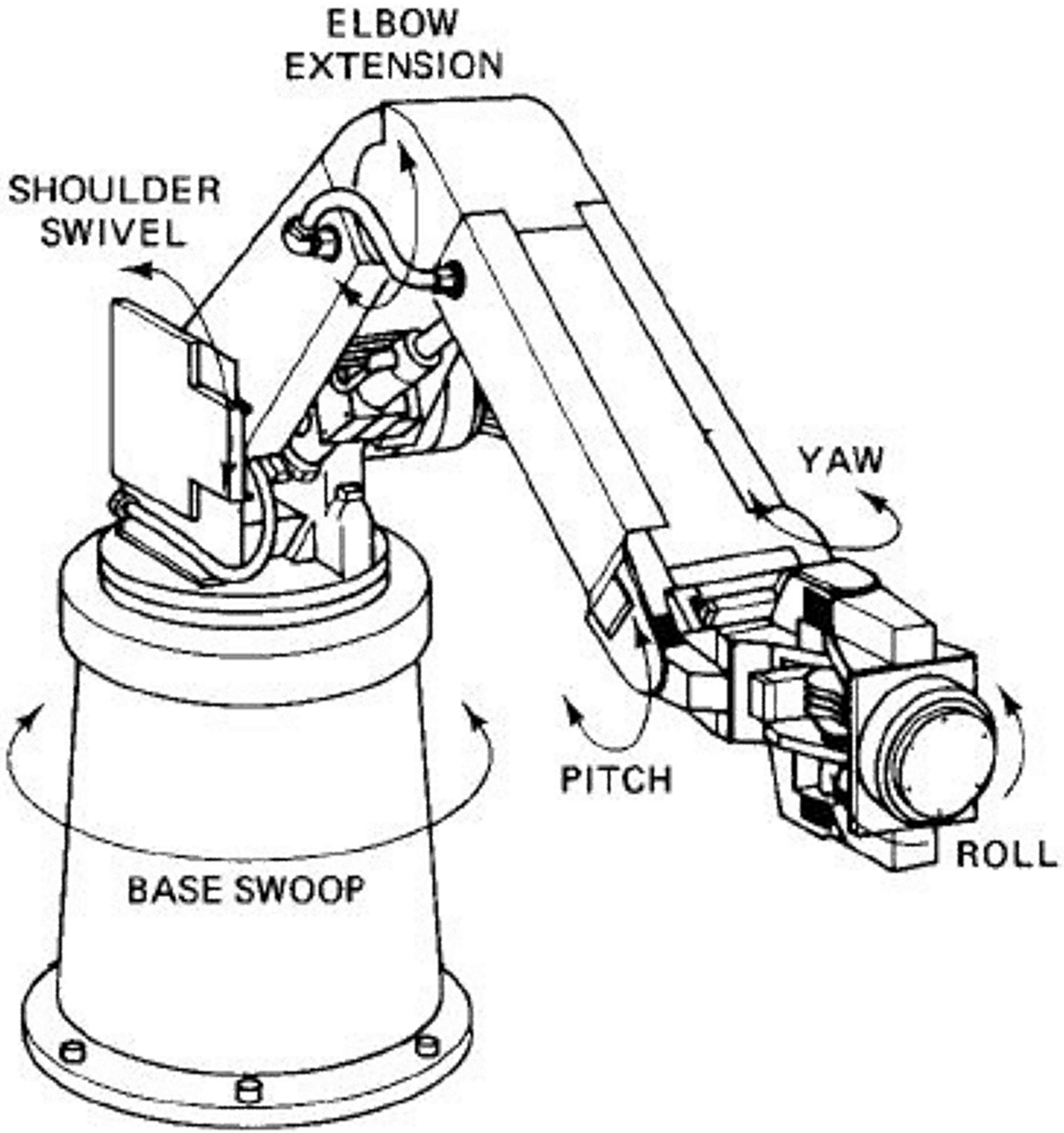
First and Last Links in the Chain

- $a_0 = \alpha_{n.} = 0$
- $\alpha_0 = \alpha_{n.} = 0$
- *If joint 1 is revolute:* $d_0 = 0$ *and* θ_1 *is arbitrary*
- *If joint 1 is prismatic:* $d_0 =$ *arbitrary and* $\theta_1 = 0$

Robot Specifications

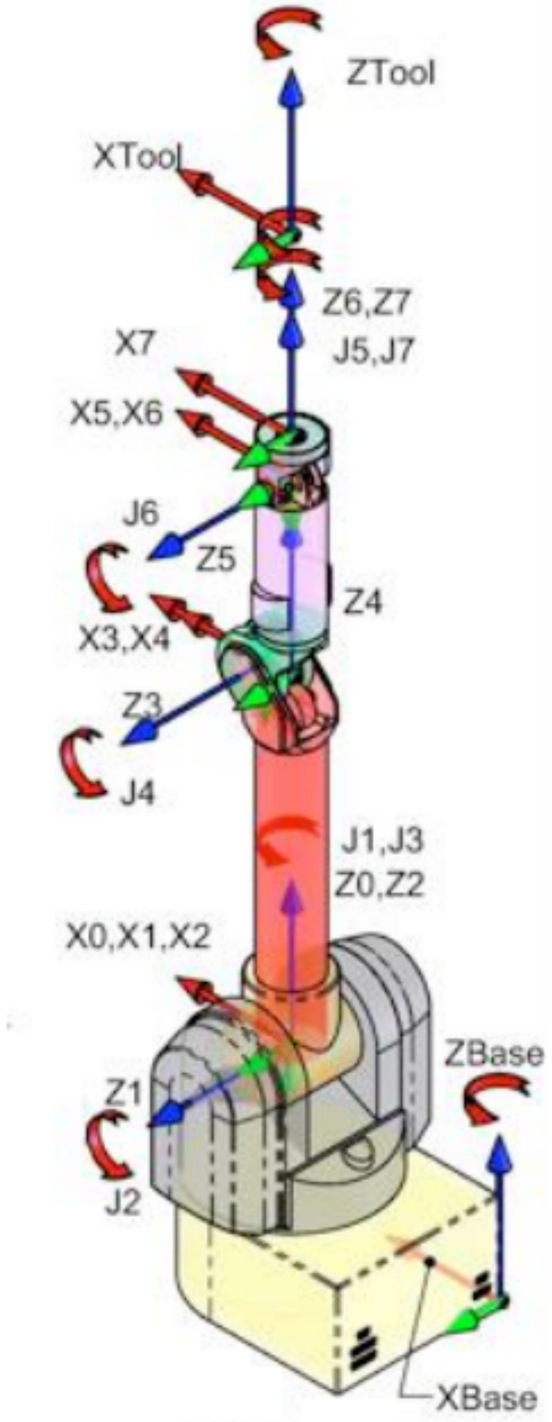
Number of axes

- Major axes, (1-3) => position the wrist
- Minor axes, (4-6) => orient the tool
- Redundant, (7-n) => reaching around obstacles, avoiding undesirable configuration



Frames

- Choose the base and tool coordinate frame
  - Make your life easy!
- Several conventions
  - Denavit Hartenberg (DH), modified DH, Hayati, etc.



KINEMATICS

Kinematics

Forward Kinematics (angles to position)

(it is straight-forward -> easy)

What you are given: The length of each link

The angle of each joint

What you can find: The position of any point (i.e. it's (x, y, z) coordinates)

Inverse Kinematics (position to angles)

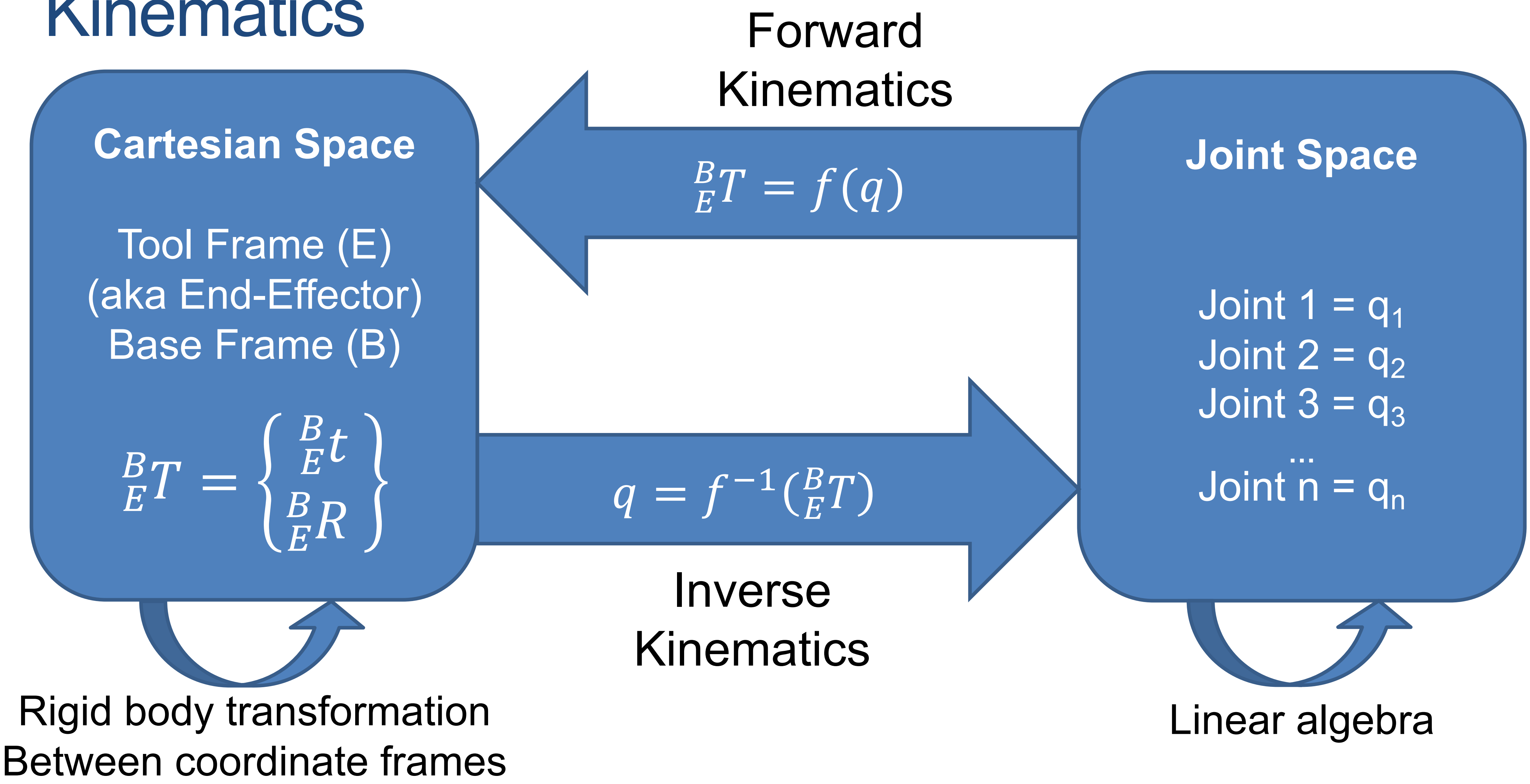
(more difficult)

What you are given: The length of each link

The position of some point on the robot

What you can find: The angles of each joint needed to obtain that position

Kinematics


Forward Kinematics
$^B_ET = f(q)$
Cartesian Space
Tool Frame (E)
(aka End-Effector)
Base Frame (B)
$^B_ET = \left\{ \begin{matrix} ^B_Et \\ ^B_ER \end{matrix} \right\}$
$q = f^{-1}(^B_ET)$
Inverse Kinematics
Joint Space
Joint 1 = q_1
Joint 2 = q_2
Joint 3 = q_3
...
Joint n = q_n
Rigid body transformation
Between coordinate frames
Linear algebra

Kinematics: Velocities

Cartesian Space
Tool Frame (E)
(aka End-Effector)
Base Frame (B)

$$ {}^{B}_{E}V = \begin{Bmatrix} {}^{B}_{E}v \\ {}^{B}_{E}w \end{Bmatrix} $$

v: linear velocity
w: angular velocity

Jacobian

$$ {}^{B}_{E}V = J(q)\dot{q} $$

$$ \dot{q} = J^{-1}(q)\ {}^{B}_{E}V $$

Inverse Jacobian

Joint Space

Joint 1 = $\dot{q}_1$
Joint 2 = $\dot{q}_2$
Joint 3 = $\dot{q}_3$
...
Joint n = $\dot{q}_n$

Rigid body transformation
Between coordinate frames

Linear algebra

INVERSE KINEMATICS (IK)

Inverse Kinematics (IK)

- Given end effector position, compute required joint angles
- In simple case, analytic solution exists
  - Use trig, geometry, and algebra to solve

- Possible Problems of Inverse Kinematics
  - Multiple solutions
  - Infinitely many solutions
  - No solutions
  - No closed-form (analytical solution)

- Generally (more DOF) difficult
  - Use Newton's method

- Analytic solution of 2-link inverse kinematics

$$x^2 + y^2 = a_1^2 + a_2^2 - 2a_1a_2\cos(\pi - \theta_2)$$

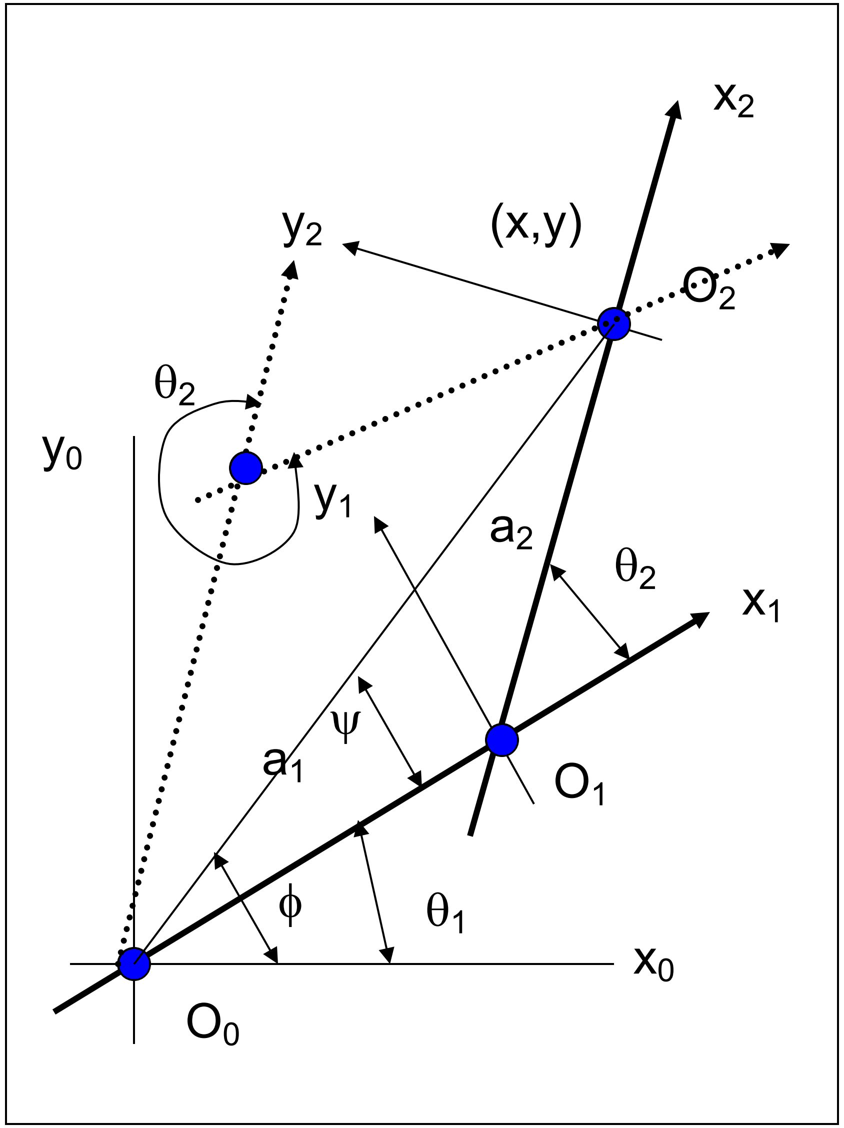
$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

for greater accuracy

$$\tan^2\frac{\theta_2}{2} = \frac{1-\cos\theta}{1+\cos\theta} = \frac{2a_1a_2 - x^2 - y^2 + a_1^2 + a_2^2}{2a_1a_2 + x^2 + y^2 - a_1^2 - a_2^2}$$

$$= \frac{(a_1^2 + a_2^2)^2 - (x^2 + y^2)}{(x^2 + y^2) - (a_1^2 - a_2^2)^2}$$

$$\theta_2 = \pm 2\tan^{-1}\sqrt{\frac{(a_1^2 + a_2^2)^2 - (x^2 + y^2)}{(x^2 + y^2) - (a_1^2 - a_2^2)^2}}$$



- Two solutions: elbow up & elbow down

Iterative IK Solutions

- Often analytic solution is infeasible
- Use Jacobian
- Derivative of function output relative to each of its inputs
- If y is function of three inputs and one output

$$y = f(x_1, x_2, x_3)$$

$$\delta y = \frac{\delta f}{\partial x_1} \cdot \delta x_1 + \frac{\delta f}{\partial x_2} \cdot \delta x_2 + \frac{\delta f}{\partial x_3} \cdot \delta x_3$$

- Represent Jacobian J(X) as a 1x3 matrix of partial derivatives

Jacobian

- In another situation,
  end effector has 6 DOFs and
  robotic arm has 6 DOFs

- $f(x_1, \ldots, x_6) = (x, y, z, r, p, y)$

- Therefore J(X) = 6x6 matrix

$$\begin{bmatrix} \frac{\partial f_x}{\partial x_1} & \frac{\partial f_y}{\partial x_1} & \frac{\partial f_z}{\partial x_1} & \frac{\partial f_r}{\partial x_1} & \frac{\partial f_p}{\partial x_1} & \frac{\partial f_y}{\partial x_1} \\ \frac{\partial f_x}{\partial x_2} & & & & & \\ \frac{\partial f_x}{\partial x_3} & & & & & \\ \frac{\partial f_x}{\partial x_4} & & & & & \\ \frac{\partial f_x}{\partial x_5} & & & & & \\ \frac{\partial f_x}{\partial x_6} & & & & & \end{bmatrix}$$

Jacobian Transpose Method

- Relates velocities in parameter space to velocities of outputs

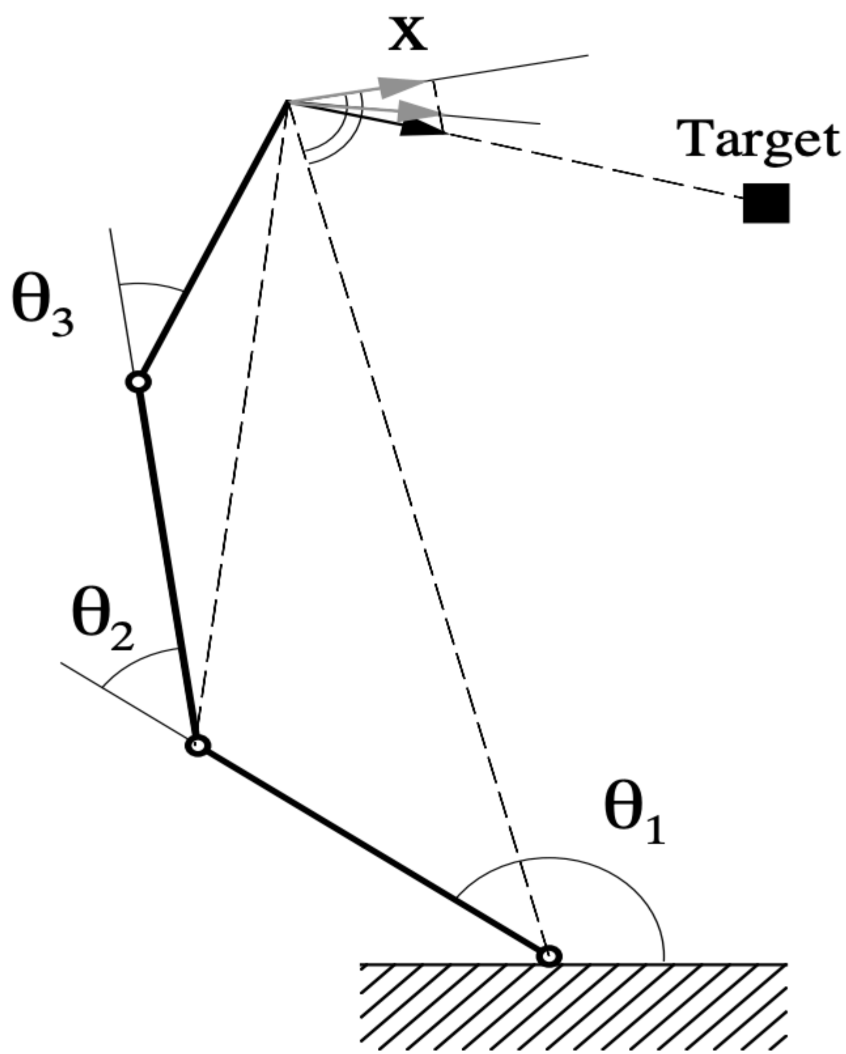
$$\dot{Y} = J(X) \cdot \dot{X}$$

- If we know $Y_{current}$ and $Y_{desired}$, then we subtract to compute $\dot{Y}$

- Invert Jacobian and solve for $\dot{X}$:

$$\dot{X} = \alpha J^T(X) \cdot \Delta Y$$

- Projects difference vector ΔY to those dimensions which can reduce it the most
- Disadvantages:
  - Needs many iterations until convergence in certain configurations (e.g., Jacobian has very small coefficients)
  - Unpredictable joint configurations



Stefan Schaal; USC, Max Planck

Iterative Solution to Inverse Kinematics

- Only holds for high sampling rates or low Cartesian velocities
- “a local solution” that may be “globally” inappropriate
- Problems with singular postures
- Can be used in two ways:
  - As an instantaneous solutions of “which way to take “
  - As an “batch” iteration method to find the correct configuration at a target

ADMIN

Project Meetings

- Make an appointment with your student advisor, Prof. and all members to meet this week!
  - Failure goes directly to “meetings” part of project grade.

- Afterwards meet every week with your advisor
  - There should be some progress every week
  - Notes will be made in a meeting.txt every week by your advisor.

Presentation

- Choose one paper from ICRA or IROS which is relevant to your project!
  - ICRA: https://ieeexplore.ieee.org/xpl/conhome/1000639/all-proceedings
    - 2020: https://ieeexplore.ieee.org/xpl/conhome/9187508/proceeding
  - IROS: https://ieeexplore.ieee.org/xpl/conhome/1000393/all-proceedings
  - Only full papers (6 or more pages) are allowed; no workshop papers

- Present the paper as if it were your own work!
- Front page: Name of the Paper; Full citation of the paper; **Your name in Pinyin; Your email address**
- Last slide: **ONE** slide about how this paper is relevant to your project.
- Your presentation has to be professional – not cute…
- Submit pdf or ppt to the paper repository till Wednesday, Oct 15 22:00 to repo! Late submissions (or if you come with the ppt/ pdf to the presentation time) will receive a flat 33% loss of points!
- Presentations in 4 slots – during lecture slots & most likely in the evenings of Oct 19-23.
- 8 minute presentation plus 1 minute project relevance plus 3 minutes questions
  - Do not rush your presentation! Better present less items more slowly!
  - 8 minute presentation => 5 – max. 10 slides
  - Maybe have a slide towards the end that you can skip if you run out of time.
  - Give a test presentation to your friends beforehand!
- Finish early for practicing – don't learn by heart.

Grading of the Presentation

- 10 %: Your basic understanding/ knowledge about the paper you present
- 20 %: Presentation timing (plus or minus one minute is ok) – no rushing – good speed!
- 10 %: Correct written English in presentation:
  - No complete sentences, no grammatical or spelling mistakes
- 10 %: Good structure of presentation:
  - Depends on the type of paper, how much time you have, how long you need to present the main achievement.
  - For example: outline, introduction/ motivation, **problem statement**, state of the art, **approach**, experiments, **results**, **conclusion**, outlook
- 20 %: Clarity of written presentation
- 10 %: Good presentation style:
  - Interact with audience: look at the whole room (not just your slides, notes, or the back of the room)
  - Present the paper – do not read (or repeat the learned) speech from a prepared text
  - Use the presentation as visual aid – not as your tele-prompter to read from
  - Move your body – do not stand frozen at one place
- 10 %: Answering the questions
  - Questions have to be asked and answered in English – Chinese can be used for clarification
- 10 %: Asking questions to other students!
- **Not** scored: Your English skill

Project Proposal (1)

- **Title:** Find a nice, catchy title for your project
- **Abstract**: A short abstract/ summary what the project is about
- **Introduction**: general description & Motivation
- **State of the Art**: Literature & open-source-ROS packages
- <u>Per team member</u>:
  - present and cite three papers with just three or four sentences
  - present in more detail one further paper relevant to your project. Describe it with at least 1/3rd of a page.
  - present in detail one open source ROS package relevant to your project. At least 1/3rd of a page
  - => about one page per team member – => 3 pages for 3 person team

Project Proposal (2)

- **System Description**
- **System Evaluation**: Describe how you want to test your system.
  - Experiments & how to measure their success
- **Work Plan**: Define some mile stones.
  - Possible phases: Algorithm design, implementation, testing, evaluation, documentation – some of those things can also happen in a loop (iteration).
  - Deliverables of Project:
    - Proposal (this document)
    - Mid-term report
    - Final demo
    - Final Report
    - Website
- **Conclusions**: Short summary and conclusions

Project Proposal (3)

- Important dates:
- **Oct 1st, 22:00: due date for the proposal**
- **December 30th – Jan 08th (tbd): due date for the final report.**

- Parts of proposal go into the final project report.
- Please don't forget to take **pictures and videos** when testing your system!

- In English! Using LaTeX!
- Put sources and PDF in git.

- **Additional task:** In glit/ gitlab: "Readme.txt" with:
  - Team Name and Members; email addresses
  - Documentation and how to's regarding your project.

HW2 is published

- Due Sep 29
- Gitlab repos will be created soon

PLANNING

Kinematic Problems for Manipulation

- Reliably position the tip - go from one position to another position

- Don't hit anything, avoid obstacles

- Make smooth motions
  - at reasonable speeds and
  - at reasonable accelerations

- Adjust to changing conditions -
  - i.e. when something is picked up *respond to the change in weight*

Planning Problem

- (Arm) Pose: Set of join values
- (Arm) Trajectory:
  - Given a start pose and an end pose
  - A list of intermediate poses
  - That should be reached one after the other
  - With associated (desired) velocities and accelerations (maxima)
    - Without time (without velocity and acceleration): path! So:
    - Path: poses; Trajectory: poses with speeds (and maybe accelerations)
- Constrains:
  - Don’t collide with yourself
  - Don’t collide with anything else (except: fingers with the object to manipulate!!!)
  - Additional possible constrains:
    - Maximum joint velocities or accelerations
    - Keep global orientation of a joint (often end-effector) within certain boundaries

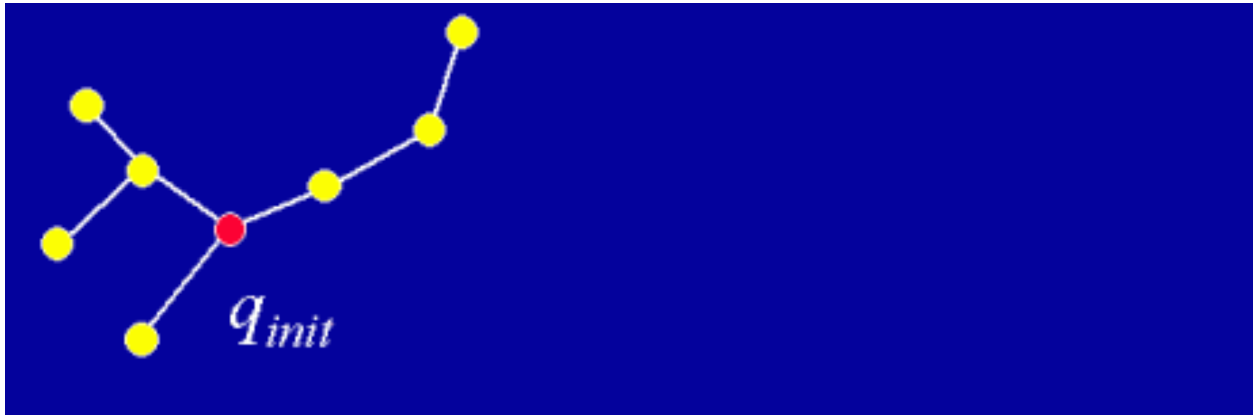
Planning Problem cont.

- Often the goal specified in Cartesian space (not joint space)
- => use IK to get joint space
- => often multiple (even infinitely many) solutions
  - Which one select for planning?
  - Plan for several solutions and select best!?

RRT

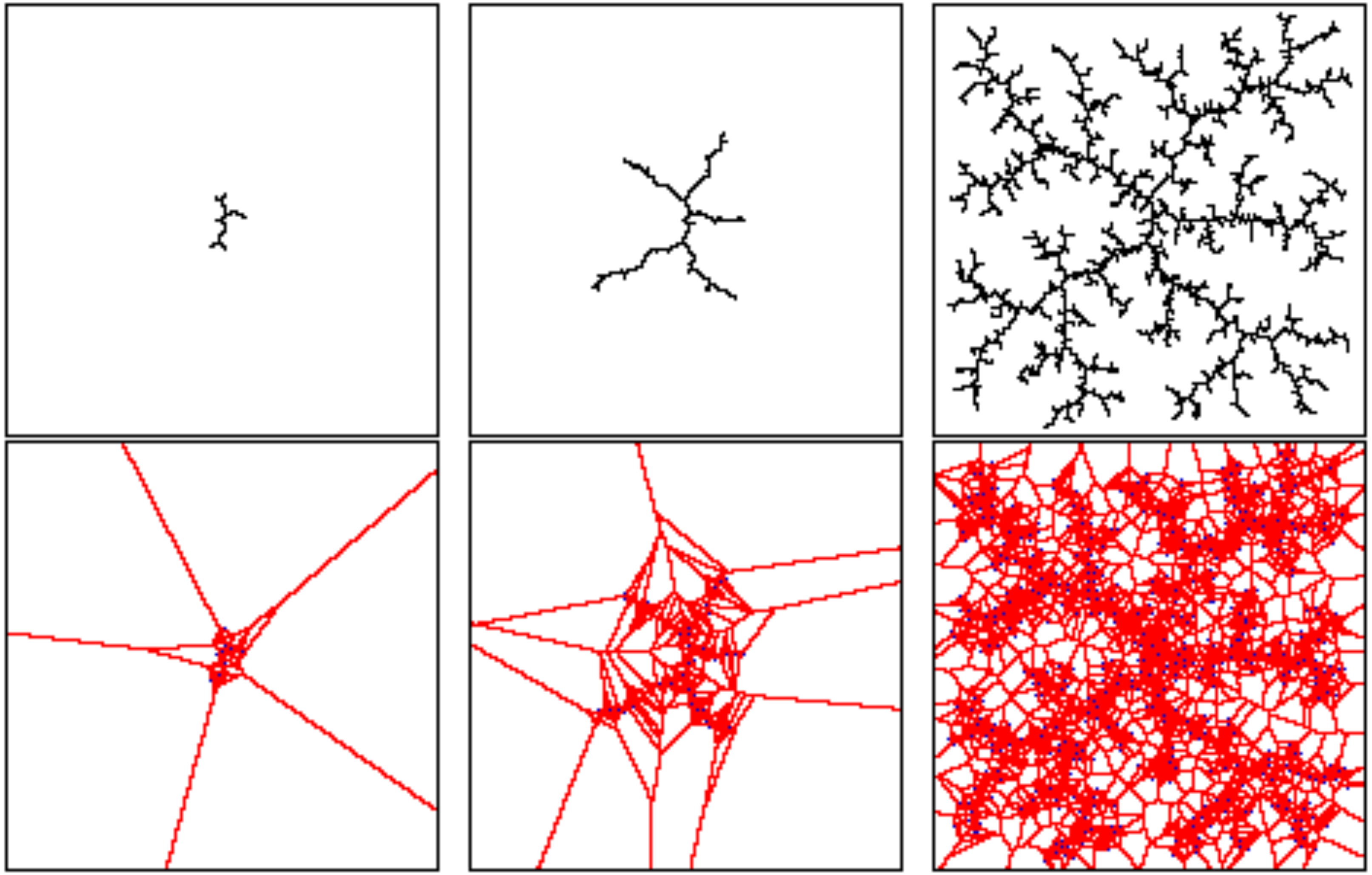
BUILD_RRT(q_{init})
$\mathcal{T}$.init(q_{init});
for $k = 1$ **to** K **do**
  $q_{rand} \leftarrow$ RANDOM_CONFIG();
  EXTEND($\mathcal{T}$, q_{rand});
Return $\mathcal{T}$

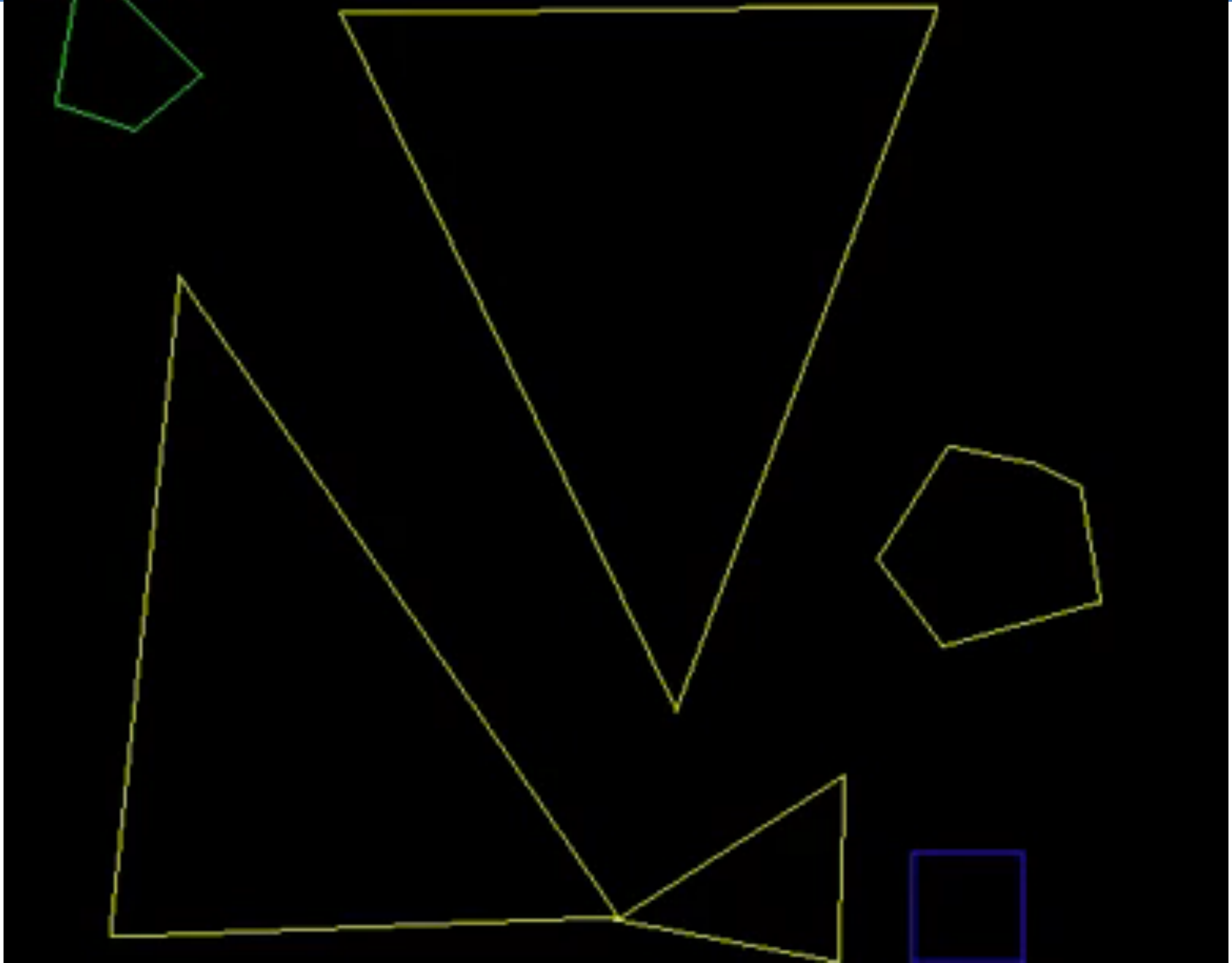
EXTEND($\mathcal{T}$, q)
$q_{near} \leftarrow$ NEAREST_NEIGHBOR(q, $\mathcal{T}$);
if NEW_CONFIG(q, q_{near}, q_{new}) **then**
  $\mathcal{T}$.add_vertex(q_{new});
  $\mathcal{T}$.add_edge(q_{near}, q_{new});
  if $q_{new} = q$ **then**
    Return *Reached*;
  else
    Return *Advanced*;
Return *Trapped*;

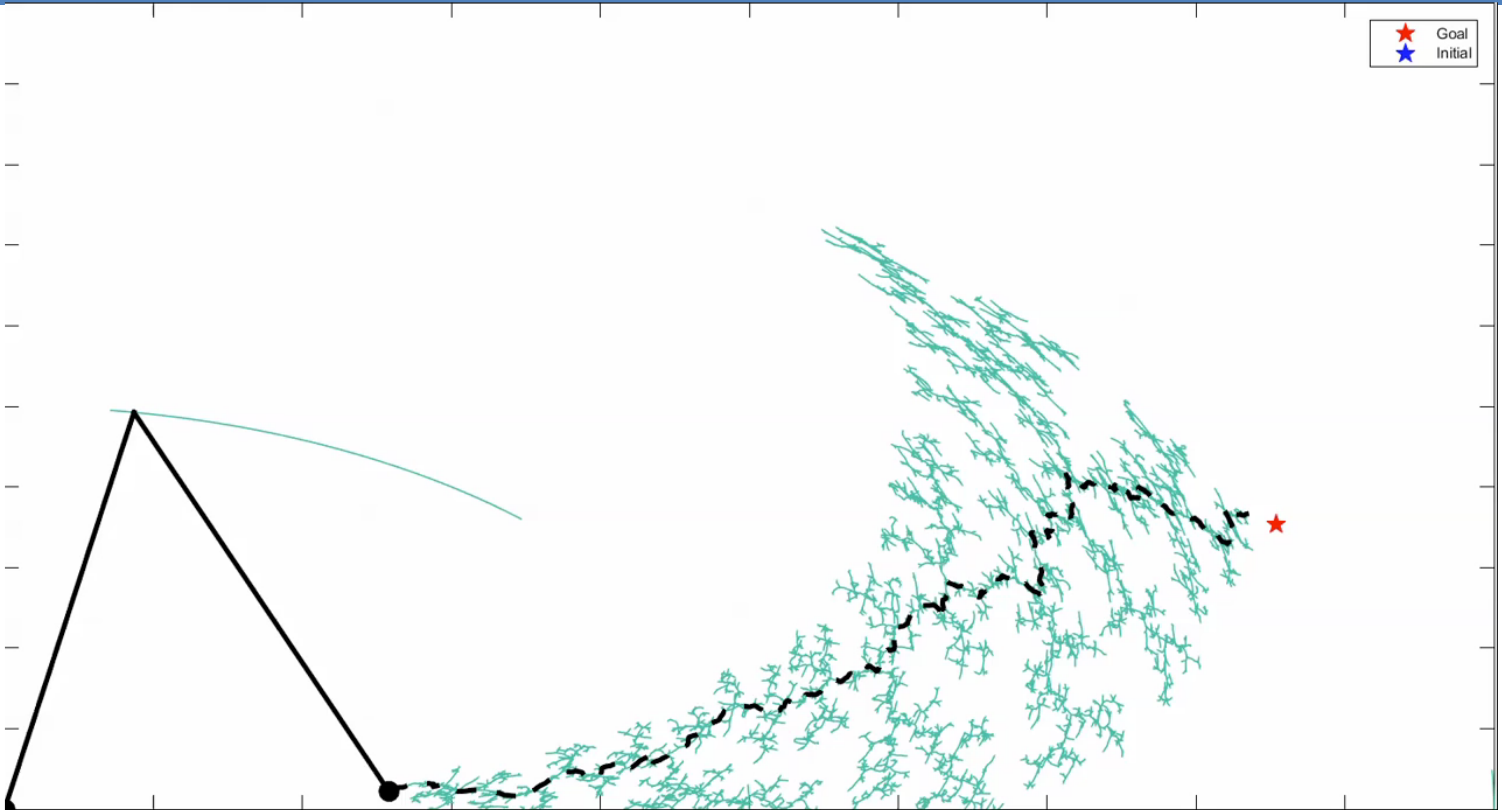


Why are RRT's rapidly exploring?

The probability of a node to be selected for expansion is proportional to the area of its Voronoi region






Goal
Initial