

CMP3103M/CMP9050M

AUTONOMOUS MOBILE ROBOTS



INTRODUCTION

Marc Hanheide



UNIVERSITY OF
LINCOLN

CMP3103M AUTONOMOUS MOBILE ROBOTS

► Semester B: AMR

- Marc Hanheide, Paul Baxter, Ayse Kucukyilmaz
- Demonstrators: Sergi Molina + XXX
- Assessment items:
 1. Robotics Practical Assignment (30%)
 2. Exam (70%)

SEMESTER B: AMR

- ▶ **Lectures**
- ▶ **Workshops**
 - ▶ Group A, B, C, D
 - ▶ Robot programming in ROS, partially in simulation, partially on real robots
- ▶ Attendance of all scheduled sessions is mandatory!
- ▶ Please check your timetable & blackboard for any timing & updates

WHAT'S ON?

B	Week	Lecture	Lecturer
1	22/01/2019	Intro	Marc Hanheide
2	29/01/2019	Robot Programming ROS	Marc Hanheide
3	05/02/2019	Robot Sensing	Marc Hanheide
4	12/02/2019	Motion and Control	Marc Hanheide
5	19/02/2019	Robot Behaviour	Ayse Kucukyilmaz
6	26/02/2019	Navigation	Ayse Kucukyilmaz
7	05/03/2019	Navigation	Ayse Kucukyilmaz
8	12/03/2019	Robot mapping - SLAM	Ayse Kucukyilmaz
9	19/03/2019	Control Architecture	Paul Baxter
10	26/03/2019	HRI1	Paul Baxter
11	02/04/2019	HRI2	Paul Baxter
12	09/04/2019	Application	Paul Baxter
13	30/04/2019	Mock Exam	Marc Hanheide

COURSEWORK



UNIVERSITY OF
LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

**Title: CMP3103/CMP9050M
Autonomous Mobile Robotics,
Autonomous Mobile Robotics (M),
Assessment Item One – Robotics Practical
Assignment**

**Indicative Weighting CMP3103M: 30%
Indicative Weighting CMP9050M: 20%**

Learning Outcomes:

On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

COURSEWORK



UNIVERSITY OF
LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Autonomous Mobile Robotics (M), Assessment Item One – Robotics Practical Assignment	Indicative Weighting CMP3103M: 30% Indicative Weighting CMP9050M: 20%
---	--

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements

This assessment item is split into two main tasks: “Group Robot Tasks” and “Visual Object Search”, both detailed below

Note: The “one-stop shop” for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/LCAS/teaching/wiki/CMP3103M>.

1. “Group Robot Tasks” (Criterion 1)

Your first task (relating to Criterion 1 “Group Robot Tasks” in the CRG, 30% of the assessment item one mark) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation.

The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshop sessions to program, test and demonstrate each task). Demonstrating later than the defined deadline results in a reduction of the merit mark by 50%. for this individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/LCAS/teaching/wiki/CMP3103M>.

COURSEWORK

UNIVERSITY OF LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Autonomous Mobile Robotics (M), Assessment Item One – Robotics Practical Assignment	Indicative Weighting CMP3103M: 30% Indicative Weighting CMP9050M: 20%
---	--

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements

This assessment item is split into two main tasks: "Group Robot Tasks" and "Visual Object Search", both detailed below.

Note: The "one-stop shop" for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

1. "Group Robot Tasks" (Criterion 1)

Your first task (relating to Criterion 1 "Group Robot Tasks" in the CRG, 30% of the assessment item one mark) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation. The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshop sessions to program, test and demonstrate each task). Demonstrating later than the defined deadline results in a reduction of the merit mark by 50% for this individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can who answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

2. "Visual Object Search" (Criterion 2 & 3)

Your second task (relating to Criterion 2 and 3 in the CRG, total of 70% of the mark for assessment item one) is to develop an object search behaviour, programmed in Python using ROS, that enables a robot to search for coloured objects visible in the robot's camera. This assessment is purely done in simulation, and *not* on the real robot. As part of this task, you must submit an *implementation* (criterion 2) and a *presentation* (criterion 3).

Implementation (Criterion 2)

Your task is to implement a behaviour that enables the robot in simulation to find a total of 4 objects distributed in a simulated environment. You need to utilise the robot's sensory input and its actuators to guide the robot to each item. Success in locating an item is defined as: (a) being less than 1m from the item, and (b) indication from the robot that it has found an object.

For the development and demonstration of your software component, you will be provided with a simulation environment (called "Gazebo"). The required software is installed on all machines in the Labs. The simulated environment includes four brightly coloured objects hidden in the environment at increasing difficulty. Your robot starts from a predefined position. You will be provided with a "training arena" in simulation (a simulation of an indoor environment in which 4 objects will be "hidden"). This "training arena" will resemble the "test arena" in terms of structure and complexity (same floor plan of the environment), but the positions of the objects will slightly vary to assess the generality of your approach.

You may choose any sensors available on the robot to drive your search behaviour. However, your system design should include the following elements:

1. Perception of the robot's environment using the Kinect sensor, either in RGB or Depth space, or using a combination of both RGB and Depth data in order find the object;
2. An implementation of an appropriate control law implementing a search behaviour on the robot. You may choose to realise this as a simple reactive behaviour or a more complex one, eg, utilising a previously acquired map of the environment;
3. Motor control of the (simulated) Turtlebot robot using the implemented control law.

COURSEWORK

UNIVERSITY OF LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Autonomous Mobile Robotics (M), Assessment Item One – Robotics Practical Assignment	Indicative Weighting CMP3103M: 30% Indicative Weighting CMP9050M: 20%
---	--

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements

This assessment item is split into two main tasks: "Group Robot Tasks" and "Visual Object Search", both detailed below.

Note: The "one-stop shop" for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

1. "Group Robot Tasks" (Criterion 1)

Your first task (relating to Criterion 1 "Group Robot Tasks" in the CRG, 30% of the assessment item one mark) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation.

The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshop sessions to program, test and demonstrate each task). Demonstrating later than the defined deadline results in a reduction of the merit mark by 50% for this individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

2. "Visual Object Search" (Criterion 2 & 3)

Your second task (relating to Criterion 2 and 3 in the CRG, total of 70% of the mark for assessment item one) is to develop an object search behaviour, programmed in Python using ROS, that enables a robot to search for coloured objects visible in the robot's camera. This assessment is purely done in simulation, and *not* on the real robot. As part of this task, you must submit an *implementation* (criterion 2) and a *presentation* (criterion 3).

Implementation (Criterion 2)

Your task is to implement a behaviour that enables the robot in simulation to find a total of 4 objects distributed in a simulated environment. You need to utilise the robot's sensory input and its actuators to guide the robot to each item. Success in locating an item is defined as: (a) being less than 1m from the item, and (b) indication from the robot that it has found an object.

For the development and demonstration of your software component, you will be provided with a simulation environment (called "Gazebo"). The required software is installed on all machines in the Labs. The simulated environment includes four brightly coloured objects hidden in the environment at increasing difficulty. Your robot starts from a predefined position. You will be provided with a "training arena" in simulation (a simulation of an indoor environment in which 4 objects will be "hidden"). This "training arena" will resemble the "test arena" in terms of structure and complexity (same floor plan of the environment), but the positions of the objects will slightly vary to assess the generality of your approach.

You may choose any sensors available on the robot to drive your search behaviour. However, your system design should include the following elements:

1. Perception of the robot's environment using the Kinect sensor, either in RGB or Depth space, or using a combination of both RGB and Depth data in order find the object;
2. An implementation of an appropriate control law implementing a search behaviour on the robot. You may choose to realise this as a simple reactive behaviour or a more complex one, eg, utilising a previously acquired map of the environment;
3. Motor control of the (simulated) Turtlebot robot using the implemented control law.

The minimum required functionality consists of a simple reactive behaviour, allowing in principle to find objects. For an average mark the behaviour should be able to successfully find some objects at unknown locations. Further extensions are possible to improve your mark in this assessment, also to enable you to find all objects. Possible extensions to the system may include (but are not limited to):

- An enhanced perception system – in-built colour appearance learning, use of additional visual cues (e.g. edges), combination of RGB and Depth features, etc.;
- Exploiting maps and other structural features in the environment or more clever search strategies.
- Utilising other existing ROS components that are available (like localisation, mapping, etc)

The software component must be implemented in Python and be supported by use of ROS to communicate with the robot. The code should be well-commented and clearly structured into functional blocks. The program must run on computers in Labs B and C. To obtain credit for this assignment you will need to demonstrate the various components of your software to the module instructors and be ready to answer questions related to the development of the solution – please follow carefully the instructions given in the lectures on the requirements for the demonstration and see below for presentation requirements.

Your implementation needs to be submitted via blackboard, with the source code containing good documentation (functionality and code quality account for 40% of assessment item one).

COURSEWORK

UNIVERSITY OF LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Autonomous Mobile Robotics (M), Assessment Item One – Robotics Practical Assignment	Indicative Weighting CMP3103M: 30% Indicative Weighting CMP9050M: 20%
--	--

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements

This assessment item is split into two main tasks: "Group Robot Tasks" and "Visual Object Search", both detailed below.

Note: The "one-stop shop" for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

1. "Group Robot Tasks" (Criterion 1)

Your first task (relating to Criterion 1 "Group Robot Tasks" in the CRG, 30% of the assessment item one mark) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation. The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshop sessions to program, test and demonstrate each task). Demonstrating later than the defined deadline results in a reduction of the merit mark by 50%, for this individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

2. "Visual Object Search" (Criterion 2 & 3)

Your second task (relating to Criterion 2 and 3 in the CRG, total of 70% of the mark for assessment item one) is to develop an object search behaviour, programmed in Python using ROS, that enables a robot to search for coloured objects visible in the robot's camera. This assessment is purely done in simulation, and *not* on the real robot. As part of this task, you must submit an *implementation* (criterion 2) and a *presentation* (criterion 3).

Implementation (Criterion 2)

Your task is to implement a behaviour that enables the robot in simulation to find a total of 4 objects distributed in a simulated environment. You need to utilise the robot's sensory input and its actuators to guide the robot to each item. Success in locating an item is defined as: (a) being less than 1m from the item, and (b) indication from the robot that it has found an object.

For the development and demonstration of your software component, you will be provided with a simulation environment (called "Gazebo"). The required software is installed on all machines in the Labs. The simulated environment includes four brightly coloured objects hidden in the environment at increasing difficulty. Your robot starts from a predefined position. You will be provided with a "training arena" in simulation (a simulation of an indoor environment in which 4 objects will be "hidden"). This "training arena" will resemble the "test arena" in terms of structure and complexity (same floor plan of the environment), but the positions of the objects will slightly vary to assess the generality of your approach.

You may choose any sensors available on the robot to drive your search behaviour. However, your system design should include the following elements:

1. Perception of the robot's environment using the Kinect sensor, either in RGB or Depth space, or using a combination of both RGB and Depth data in order find the object;
2. An implementation of an appropriate control law implementing a search behaviour on the robot. You may choose to realise this as a simple reactive behaviour or a more complex one, eg, utilising a previously acquired map of the environment;
3. Motor control of the (simulated) Turtlebot robot using the implemented control law.

The minimum required functionality consists of a simple reactive behaviour, allowing in principle to find objects. For an average mark the behaviour should be able to successfully find some objects at unknown locations. Further extensions are possible to improve your mark in this assessment, also to enable you to find all objects. Possible extensions to the system may include (but are not limited to):

- An enhanced perception system – in-built colour appearance learning, use of additional visual cues (e.g. edges), combination of RGB and Depth features, etc;
- Exploiting maps and other structural features in the environment or more clever search strategies.

• Utilising other existing ROS components that are available (like localisation, mapping, etc) The software component must be implemented in Python and be supported by use of ROS to communicate with the robot. The code should be well-commented and clearly structured into functional blocks. The program must run on computers in Labs B and C. To obtain credit for this assignment you will need to demonstrate the various components of your software to the module instructors and be ready to answer questions related to the development of the solution – please follow carefully the instructions given in the lectures on the requirements for the demonstration and see below for presentation requirements.

Your implementation needs to be submitted via blackboard, with the source code containing good documentation (functionality and code quality account for 40% of assessment item one).

COURSEWORK

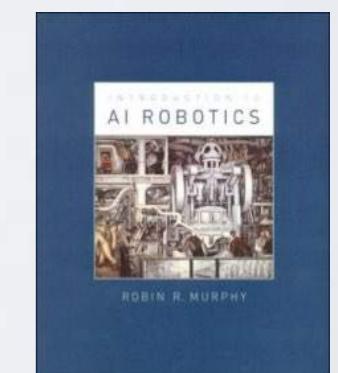
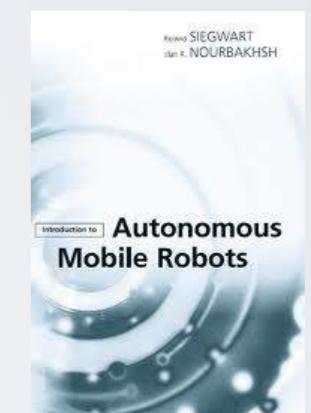
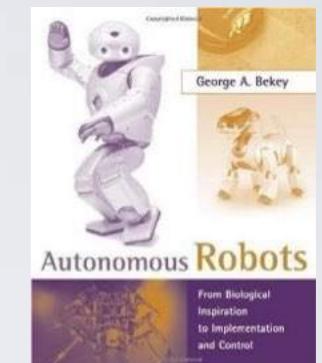
Learning Outcome	Criterion	Pass	2:2	2:1	1st
[LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments	Criterion 1: Group Robot Tasks (30%)	You and your group have demonstrated basic functionality in simulation only.	You and your group have demonstrated the full functionality in simulation only, or the basic functionality in simulation and on the real robot.	You and your group have demonstrated the full functionality in simulation and basic functionality on the real robot.	You and your group have demonstrated the full functionality in both, simulation and on the real robot.
	Criterion 2: Individual Visual Search Task(in Simulation only, 40%), Implementation	A working software component with basic functionality. Fair program structure and some code comments. The working implementation is demonstrated, accomplishing the search task partially.	A working software component with good functionality. Clear program structure and appropriate comments. The implementation is demonstrated successfully, accomplishing most of the search task with a good performance.	A good implementation with some extra functionality or originality. The program code is well structured and commented. Good demonstration of basic and additional features, accomplishing the search task with a very good performance.	An excellent implementation featuring original functionality and elements beyond the original specification. The program code is efficient, well structured and commented. The solution is demonstrated very well, highlighting the additional functionalities, accomplishing the full search task with an excellent performance.
	Criterion 3: Individual Visual Search Task (in Sim, 30%), Presentation	A basic presentation of the system design and its performance.	A good presentation of the system design and reflections on its performance.	A very good presentation of the system design and reflections on its performance, including evidence of testing and evaluation of the important system features.	An excellent presentation of the system design and reflections on its performance, including evidence of thorough testing and evaluation of the important system features.
Weighting	The criteria for this assessment are weighted as indicated.				

WHAT'S ON?

B week	Date	Workshop
1	25/01/2019	Minitask 1: setup, running simulation
2	01/02/2019	Minitask 2: ROS intro, topics, services, example to make robot move cmd_vel
3	08/02/2019	Minitask 3: Rviz, point clouds, tf, more on event-driven programming, open_cv, etc.
4	15/02/2019	Minitask 4: cmd_vel, Braitenberg
5	22/02/2019	assignment support
6	01/03/2019	assignment support
7	08/03/2019	assignment support
8	15/03/2019	assignment support
9	22/03/2019	assignment support
10	29/03/2019	eval
11	05/04/2019	eval
12	12/04/2019	Q&A
13		

READING MATERIAL

- ▶ Bekey, G.A.: *Autonomous robots*. MIT Press, 2005
- ▶ Siegwart, R. and Nourbakhsh, I.R.: *Introduction to autonomous mobile robots*. MIT Press, 2004
- ▶ Murphy, R.R.: *An Introduction to AI Robotics*. MIT Press, 2000





LiveSlides web content

To view

Download the add-in.

liveslides.com/download

Start the presentation.

TURTLEBOT 2 – OUR WORKSHOP BUDDY

This year's
assignment:
design and
implement a
robotic visual
search
(more soon...)



ROBOT

- ▶ What is it?
- ▶ “I can't define a robot, but I know one when I see one.”, J. Engelberger
- ▶ What is it for?
- ▶ help/replace humans in monotonous, boring, repetitive, dangerous, difficult tasks
- ▶ companion? helper? servant?

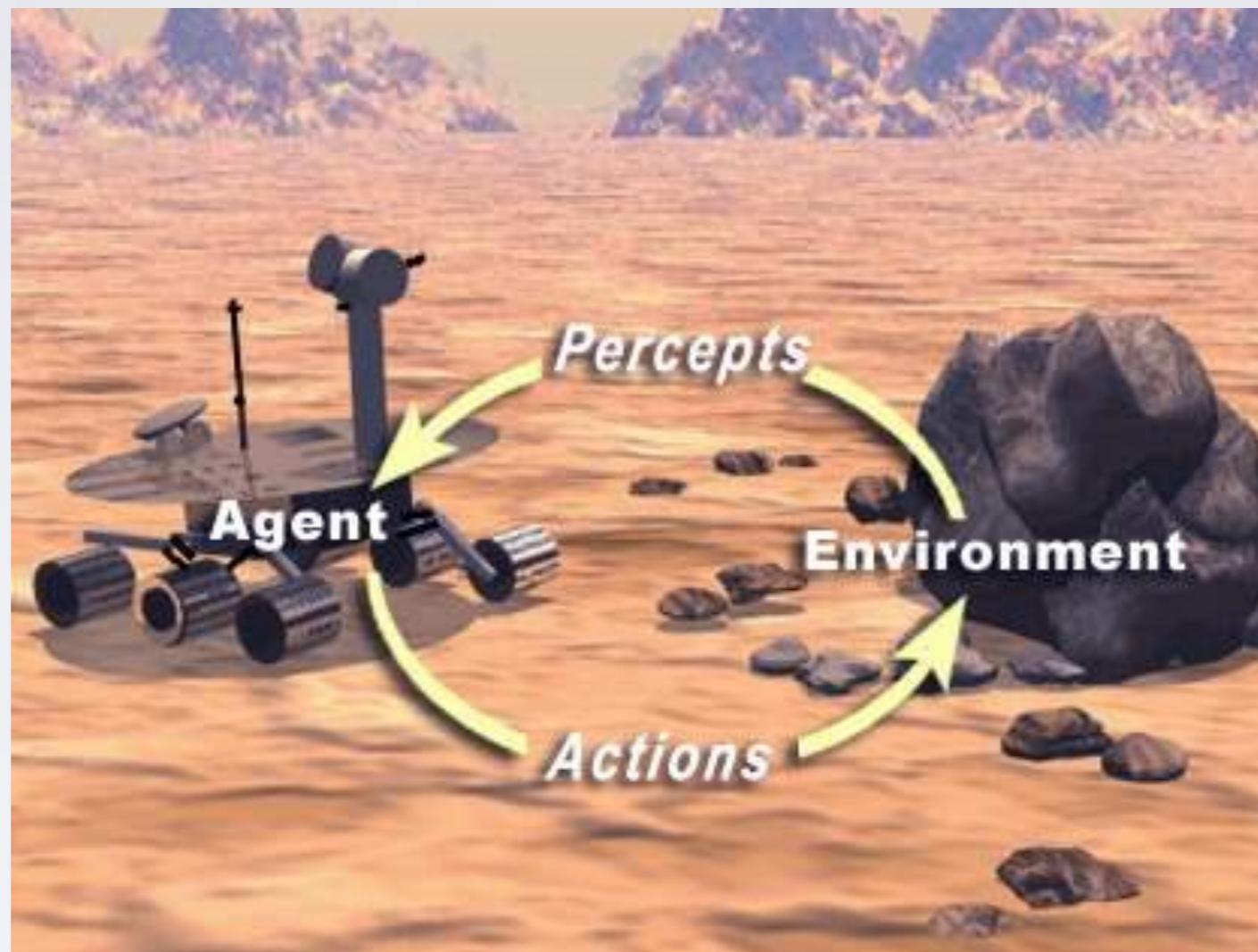


[HTTPS://POLLEV.COM/MHANHEIDE](https://pollev.com/mhanheide)

What do you associate with the word robot?

fury
future :-p
perception
doom

ROBOTICS



- Robotics = "the intelligent connection of perception and action" (Brady 1985)

Which of the following features are essential for a robot

Hardware Controllers

Actuators

Locomotion (e.g. wheels
or legs)

Embodiment (physical
interaction with the world)

Internal Sensors

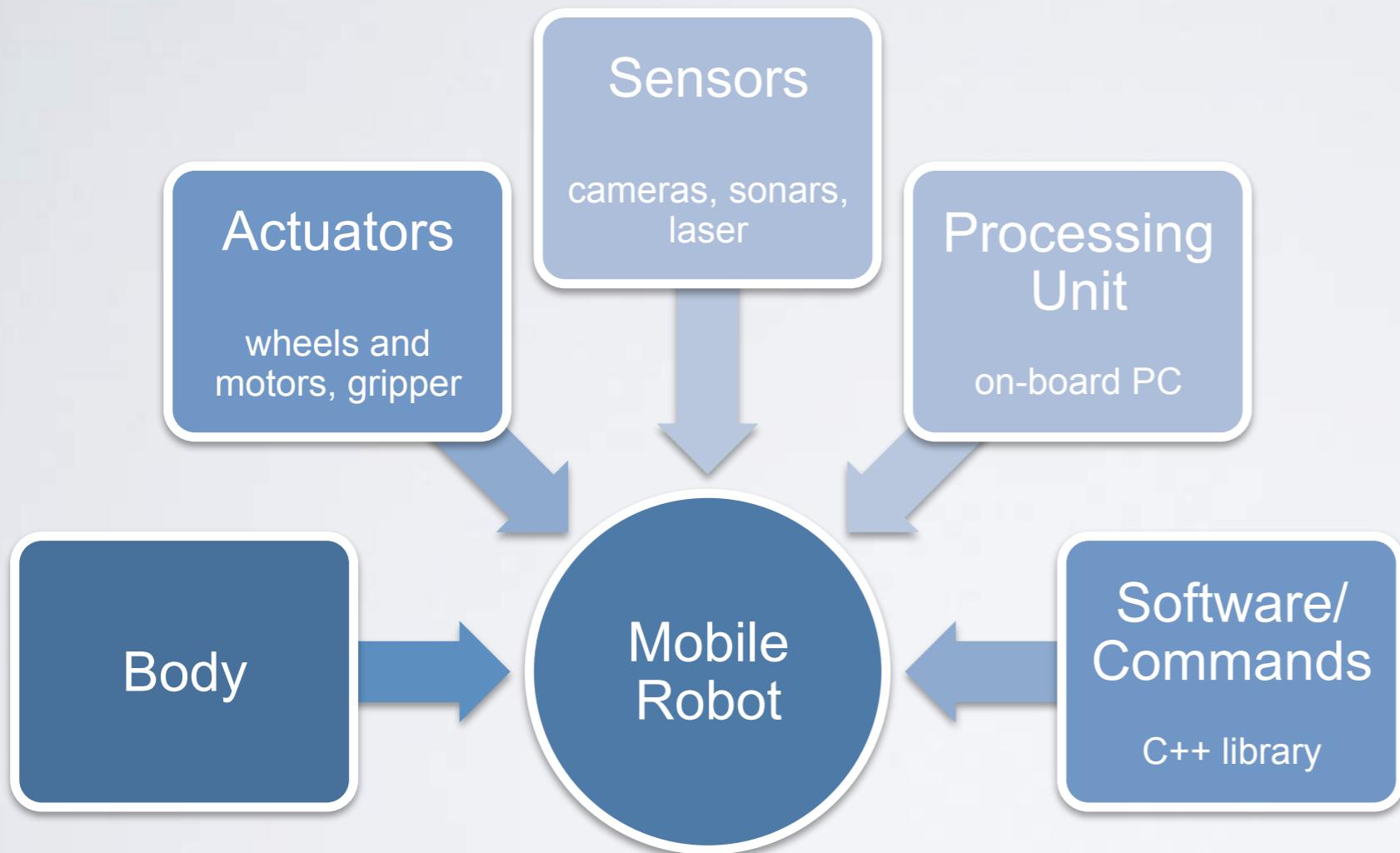
External Sensors

MOBILE ROBOTS

- ▶ Mobility
 - ▶ opens possibilities for new tasks: transportation, surveillance, cleaning etc.
 - ▶ unstructured environments
 - ▶ main challenge: navigation
- ▶ Autonomy
 - ▶ reasoning: making decisions, plan
 - ▶ learning from experience
 - ▶ building representation of the (dynamic) environment

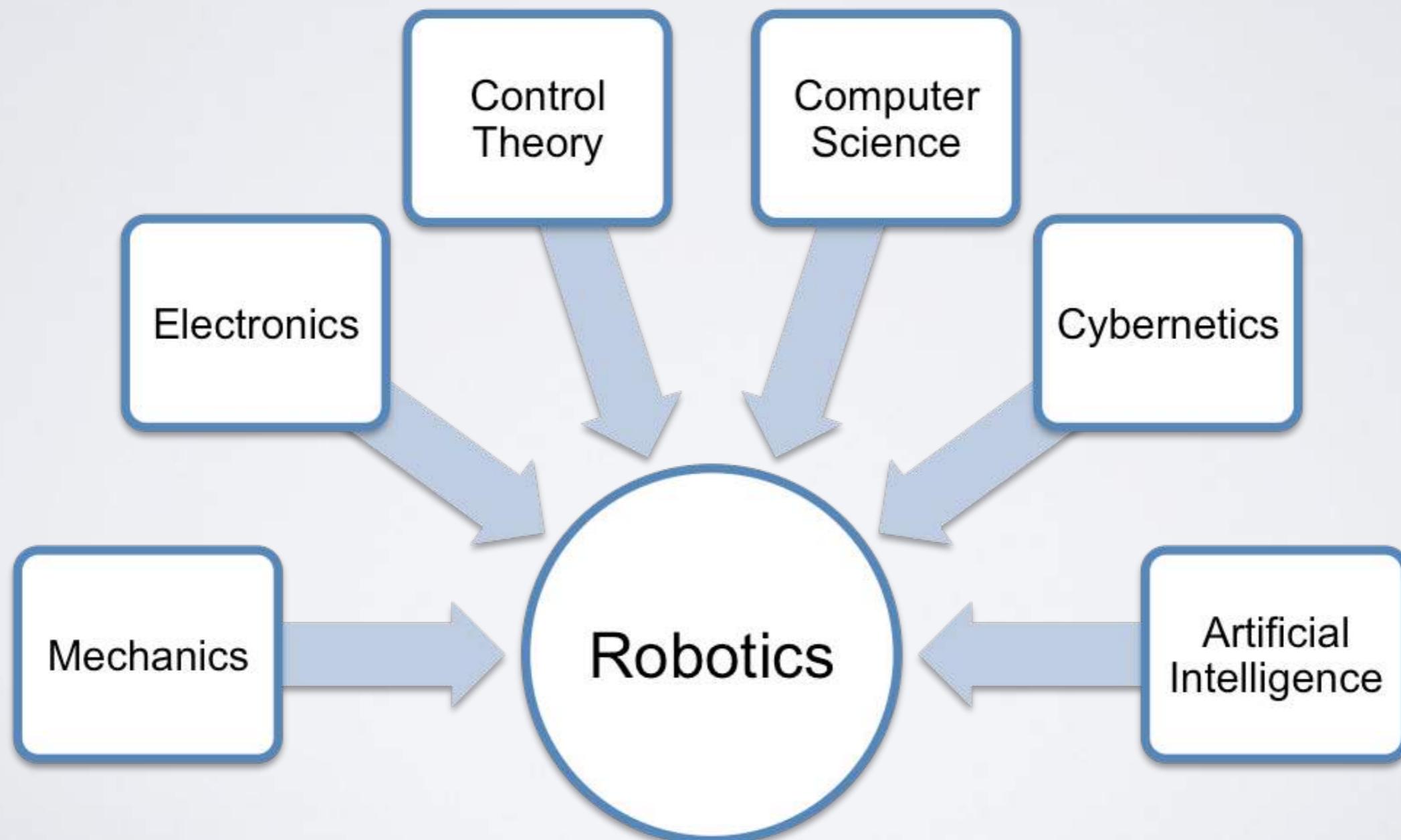


ANATOMY OF A MOBILE ROBOT



ROBOTICS

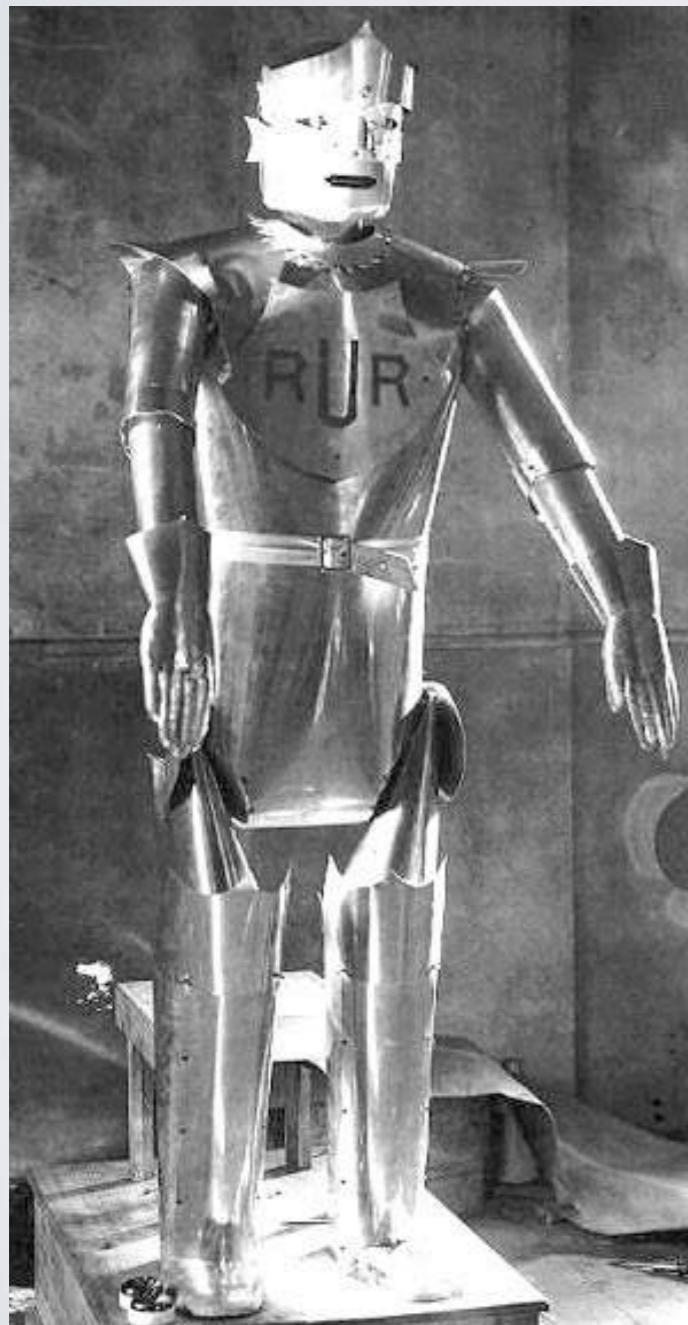
Science and technology of robots





PAST AND PRESENT

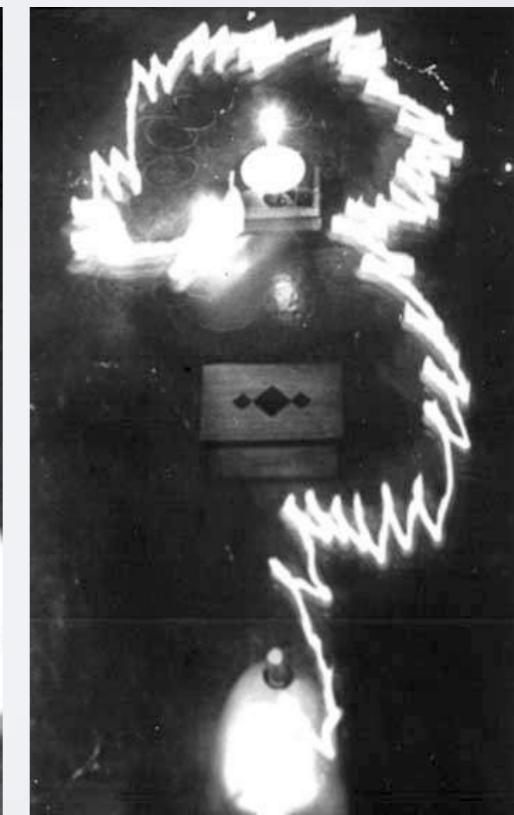
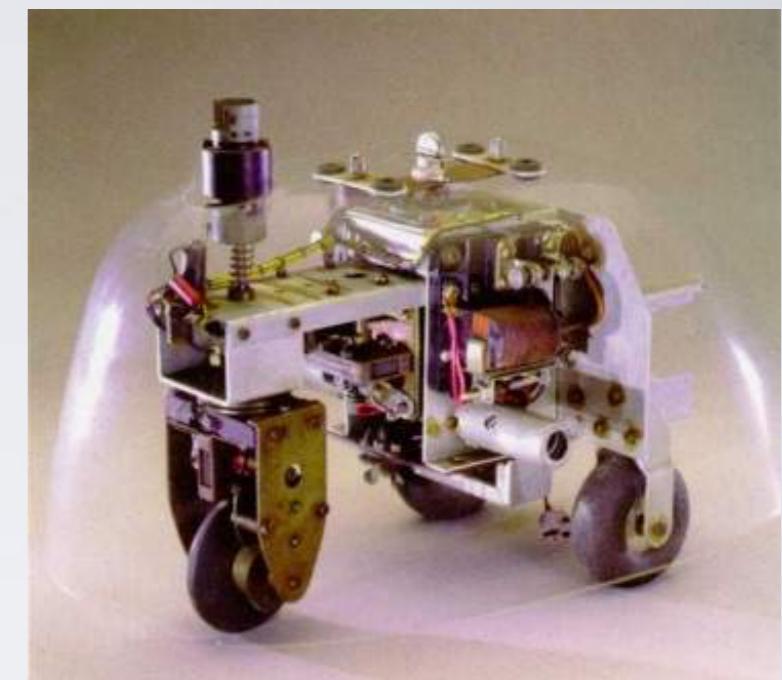
R.U.R.



- ▶ Karel Čapek, 1921
- ▶ R.U.R. - Rossum's Universal Robots, a stage play
- ▶ the word 'robot' appears for the first time
- ▶ 'roboťa' – forced/hard labour in Czech
- ▶ robots – artificial men that can think but seem to be happy to serve
- ▶ exploited (?) by humans
- ▶ finally rebel against their creators

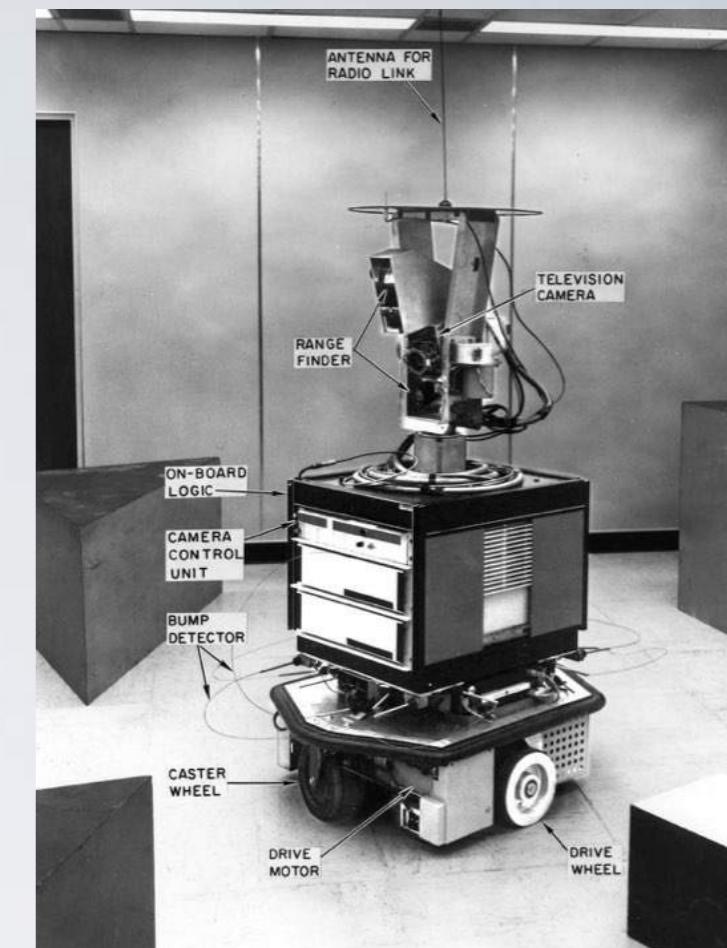
GREY WALTER'S TURTLES

- ▶ “Machina Speculatrix”, 1948
- ▶ experiments in reflex behaviour
- ▶ built of electronic valves and photo-cells
- ▶ approach or escape a light source



SHAKEY

- ▶ Stanford Artificial Intelligence Centre, 1966
- ▶ first mobile robot that could be programmed for various tasks
- ▶ on-board I/O logic
- ▶ radio-link
- ▶ external computer (PDP-10)

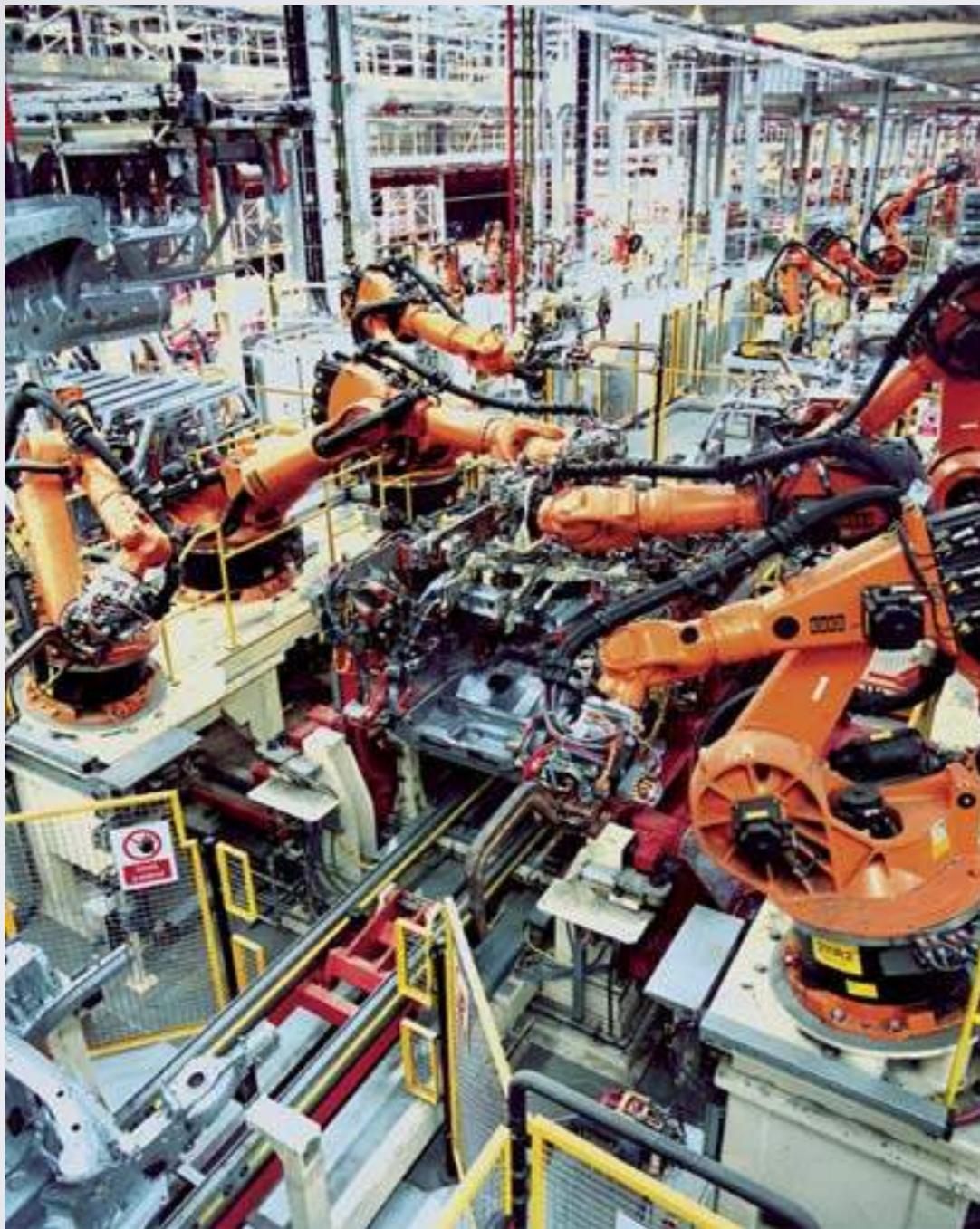


INDUSTRIAL ROBOTS

- ▶ Manipulators
 - ▶ “big arms”
 - ▶ precise, strong and fast
 - ▶ well studied
 - ▶ many manufactured
 - ▶ operate in controlled environments
 - ▶ limited sensory abilities
 - ▶ pre-programmed



INDUSTRIAL ROBOTS



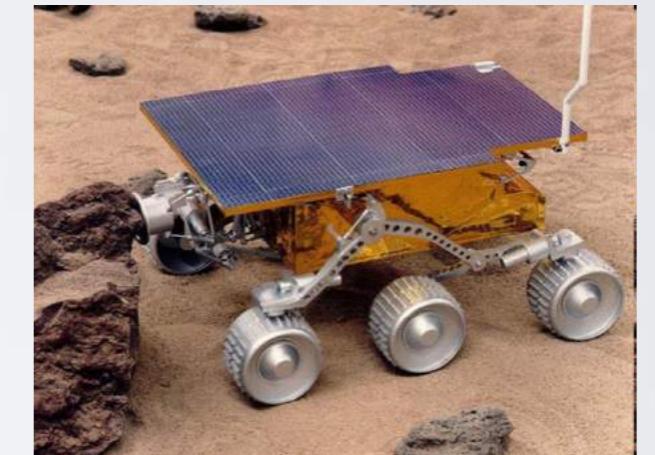
“Big arms”



AGVs

APPLICATIONS

- ▶ Exploration
- ▶ Surveillance
- ▶ Security
- ▶ Care assistants
- ▶ Agricultural robots
- ▶ Intelligent vehicles
- ▶ many others...



Which jobs are most likely to be replaced by Robots/AI? Put the most likely one to the top.

Bin man

Investment Banker

Law Associate

Dentist

Programmer

Brick layer

Fruit picker

Lecturer

Accountant

Which jobs are at risk?

Researchers at Oxford University published a widely referenced study in 2013 on the likelihood of computerisation for different occupations.

Out of around 700 occupations, 12 were found to have a 99 per cent chance of being automated in the future:

- Data Entry Keyers
- Library Technicians
- New Accounts Clerks
- Photographic Process Workers and Processing Machine Operators
- Tax Preparers
- Cargo and Freight Agents
- Watch Repairers
- Insurance Underwriters
- Mathematical Technicians
- Sewers, Hand
- Title Examiners, Abstractors, and Searchers
- Telemarketers

► <http://www.telegraph.co.uk/news/2017/09/27/jobs-risk-automation-according-oxford-university-one/>



BREAK?!

SENSORS AND ACTUATORS

EFFECTORS AND ACTUATORS

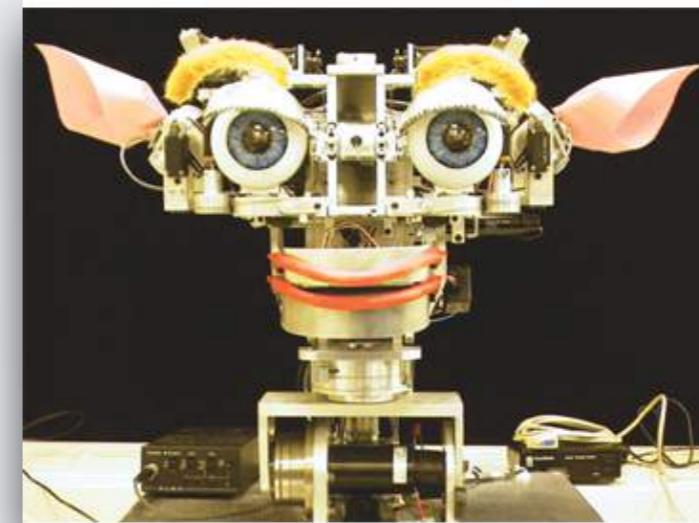
- ▶ Effectors:
 - ▶ hand, arm, gripper – manipulators
 - ▶ wheels, legs, tracks, rotors – mobile robots
- ▶ Actuators:
 - ▶ electric motors, pneumatic and hydraulic systems



EFFECTORS – EXAMPLES



pan-tilt unit



robotic head



parallel arm



gripper

SENSORS: CLASSIFICATION

- ▶ **Proprioceptive**

- ▶ internal state of the robot, self-awareness

- ▶ **Exteroceptive**

- ▶ state of the environment

Which of the following are exteroceptive sensors?

- Laser scanner
- Sonar
- CPU Thermometer
- Wheel encoder
- Radar
- Microphone
- Camera
- Speaker
- Gyroscope
- GPS

SENSORS: CLASSIFICATION

▶ **Proprioceptive**

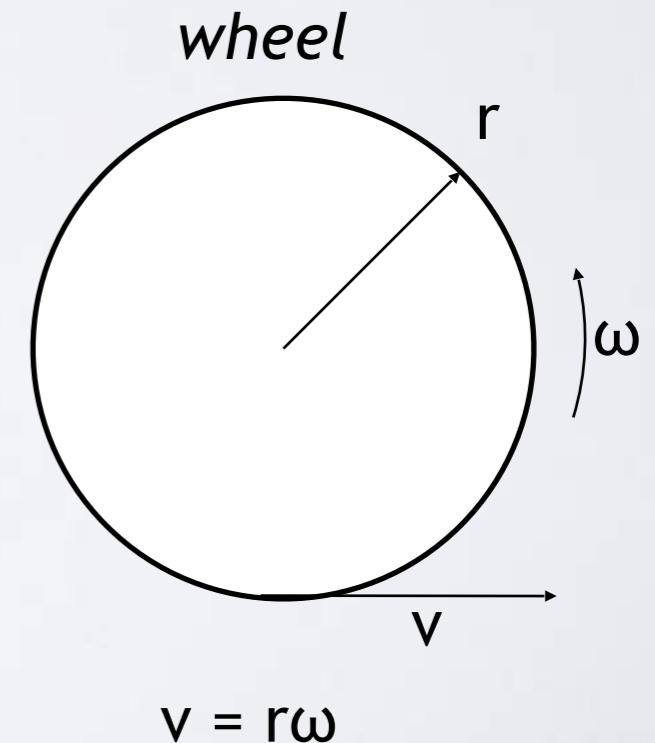
- ▶ internal state of the robot, self-awareness
- ▶ e.g. odometry, battery level, temperature

▶ **Exteroceptive**

- ▶ state of the environment, light intensity, distance measurements
- ▶ e.g. sonars, video cameras
- ▶ Passive vs. Active
- ▶ measuring phenomena directly or indirectly (e.g. reflected light/sound)

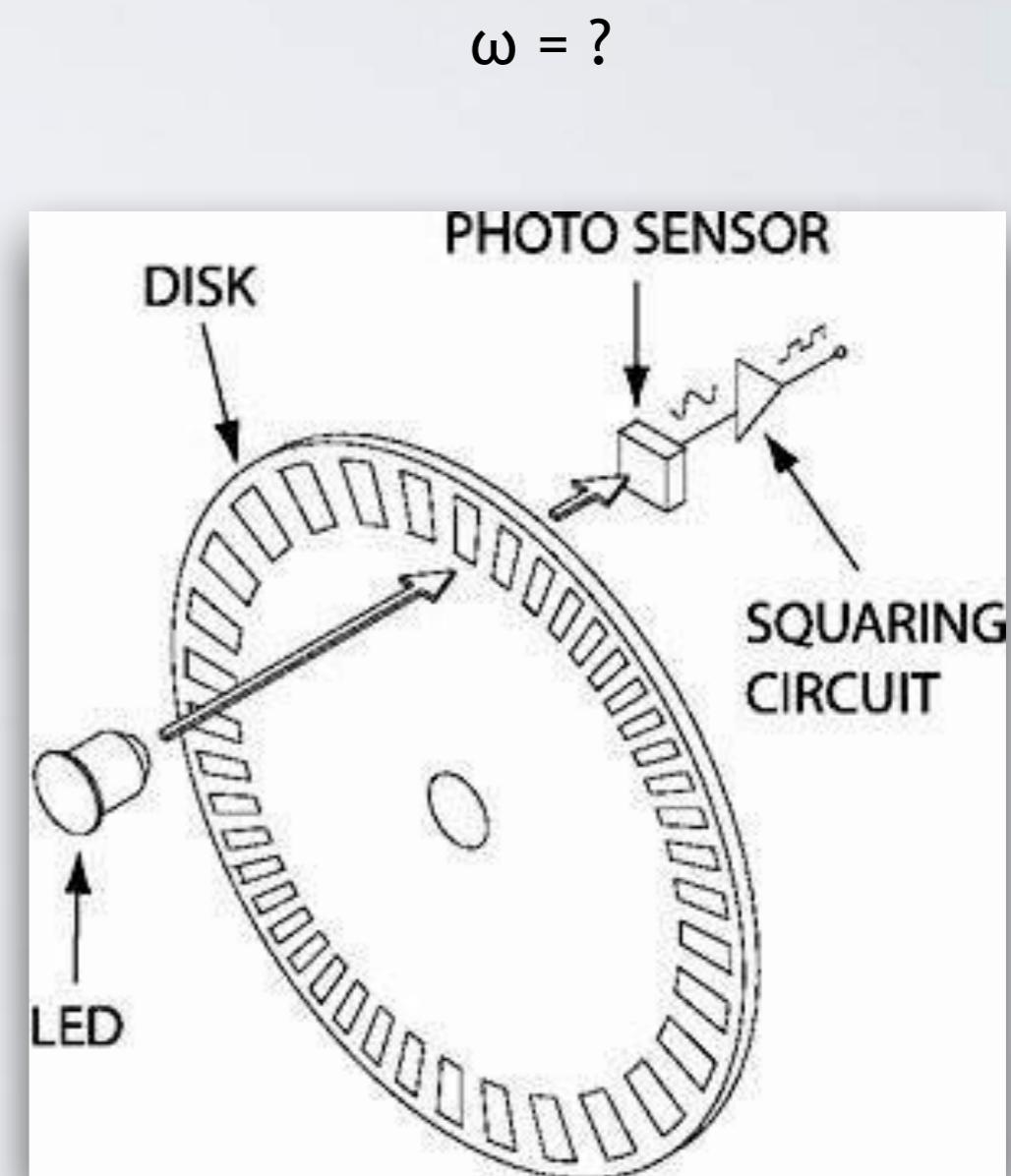
ODOMETRY – WHERE AM I?

- ▶ Dead reckoning
 - ▶ “deduced reckoning” – marine navigation
 - ▶ position estimation based on the previous known position
 - ▶ used by animals: pigeons, ants
- ▶ Mobile robots – odometry sensor
 - ▶ measure the speed of each wheel
 - ▶ use wheel geometry to calculate the velocity of a robot



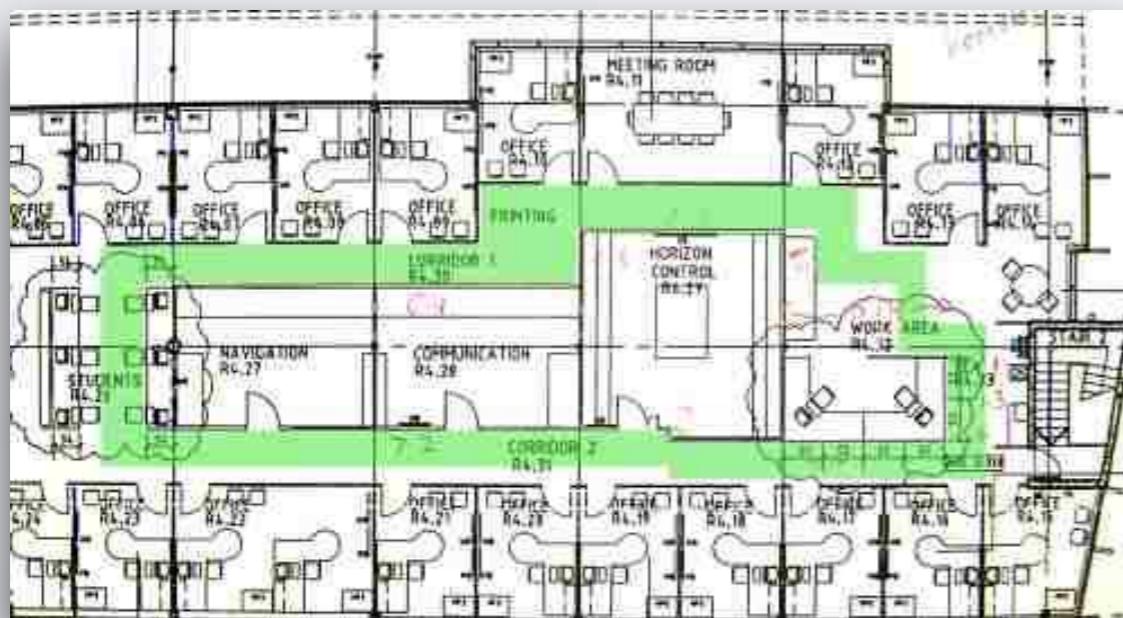
WHEEL SPEED ESTIMATION

- Nominal motor speed + gear ratio
 - provided by the motor manufacturer
 - may change under different loads
- Motor Encoder
 - sensor mounted on the wheel shaft
 - counts motor revolutions
 - similar to a computer mouse

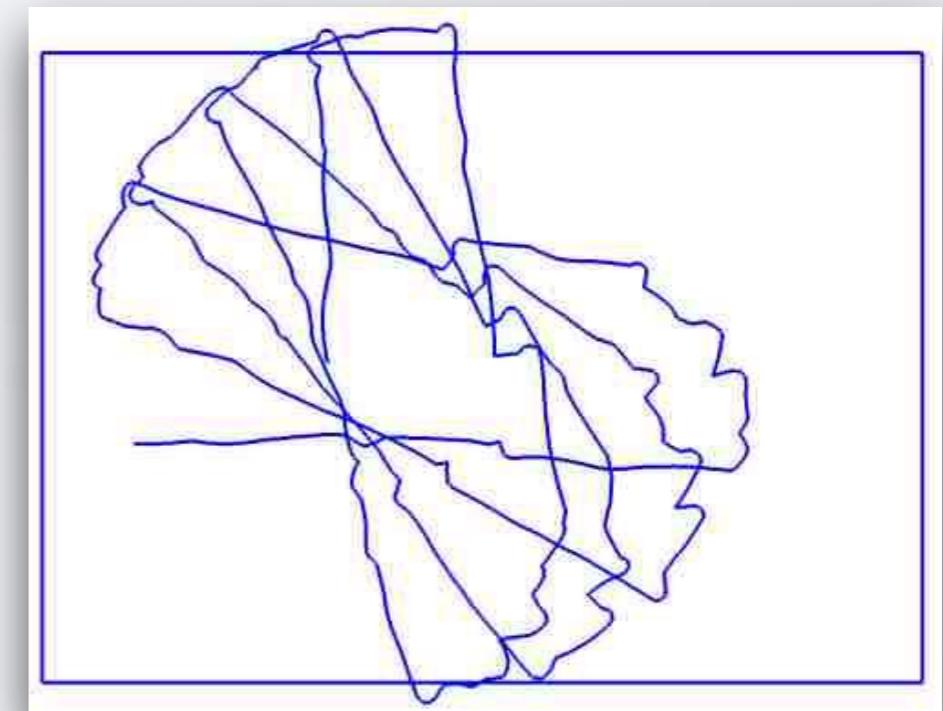


ODOMETRY – LIMITATIONS

- ▶ Wheel slippage, uneven friction and wheel size, etc. can cause errors that accumulate with time



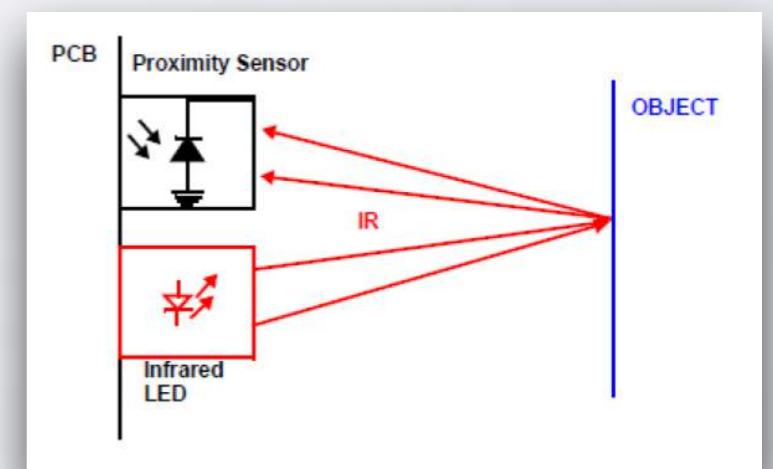
Floor plan and robot route



Robot's perceived trajectory using odometry

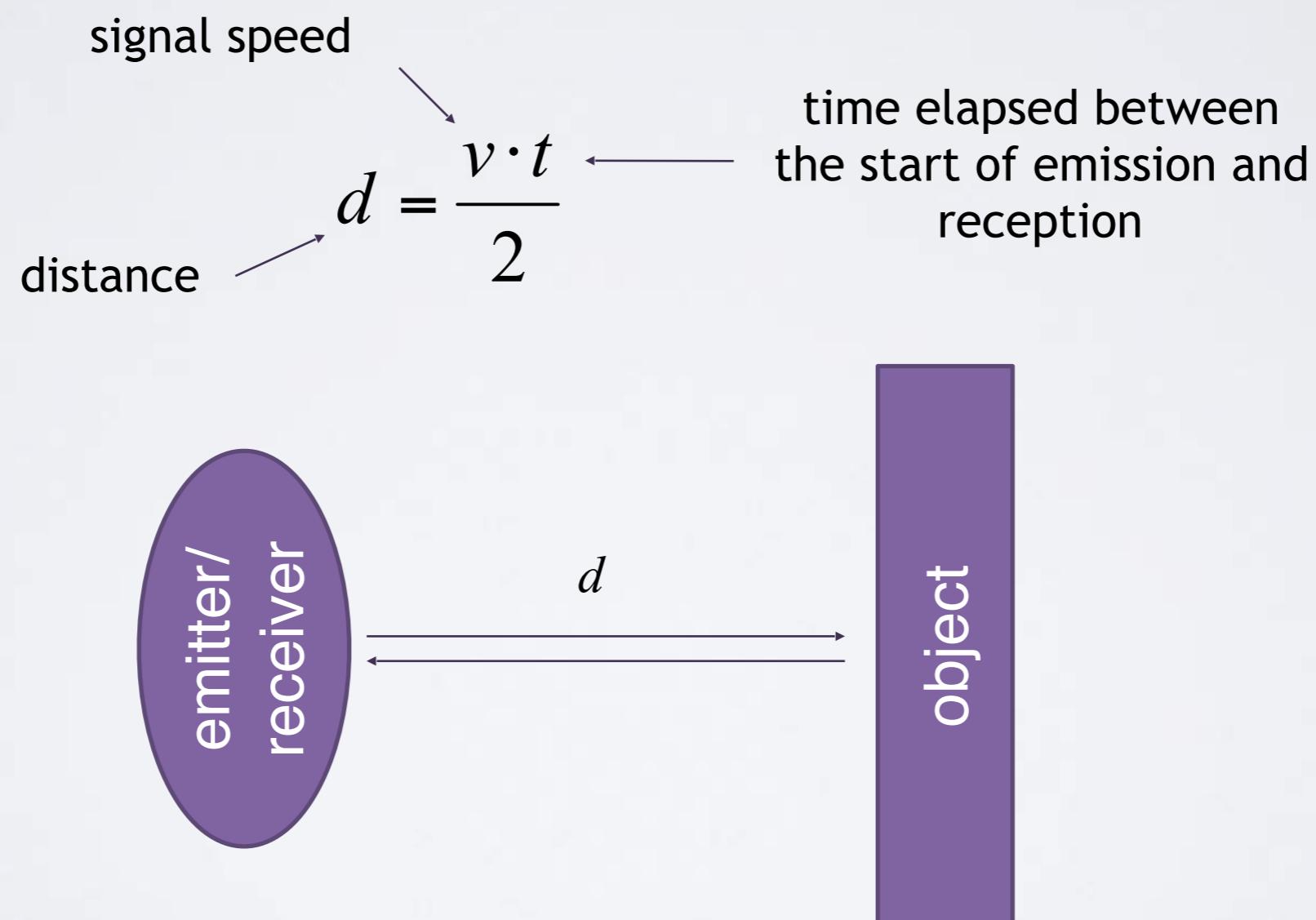
LIGHT SENSOR

- ▶ Passive – measuring light intensity directly (e.g. photo-resistor)
- ▶ light can be used to mark important places e.g. recharging station, exit from the room, etc.
- ▶ Active – measuring reflected light
 - ▶ e.g. IR sensor
 - ▶ inexpensive but short range



TIME OF FLIGHT SENSORS

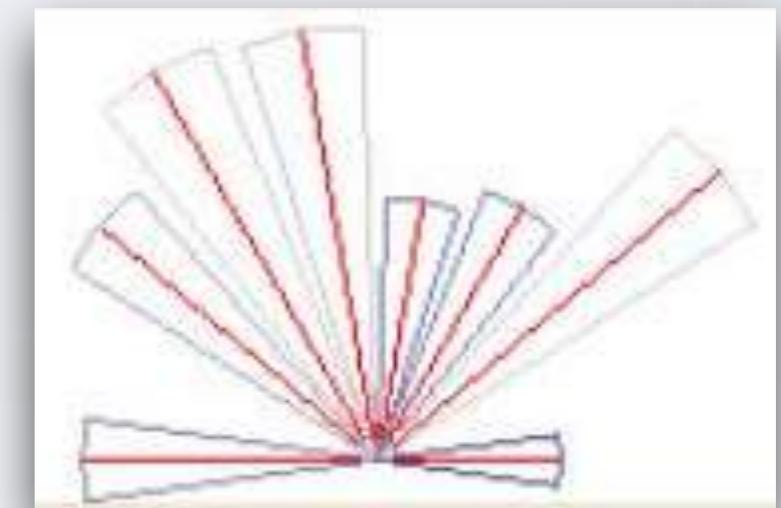
- ▶ Active distance measurements – reflected signal



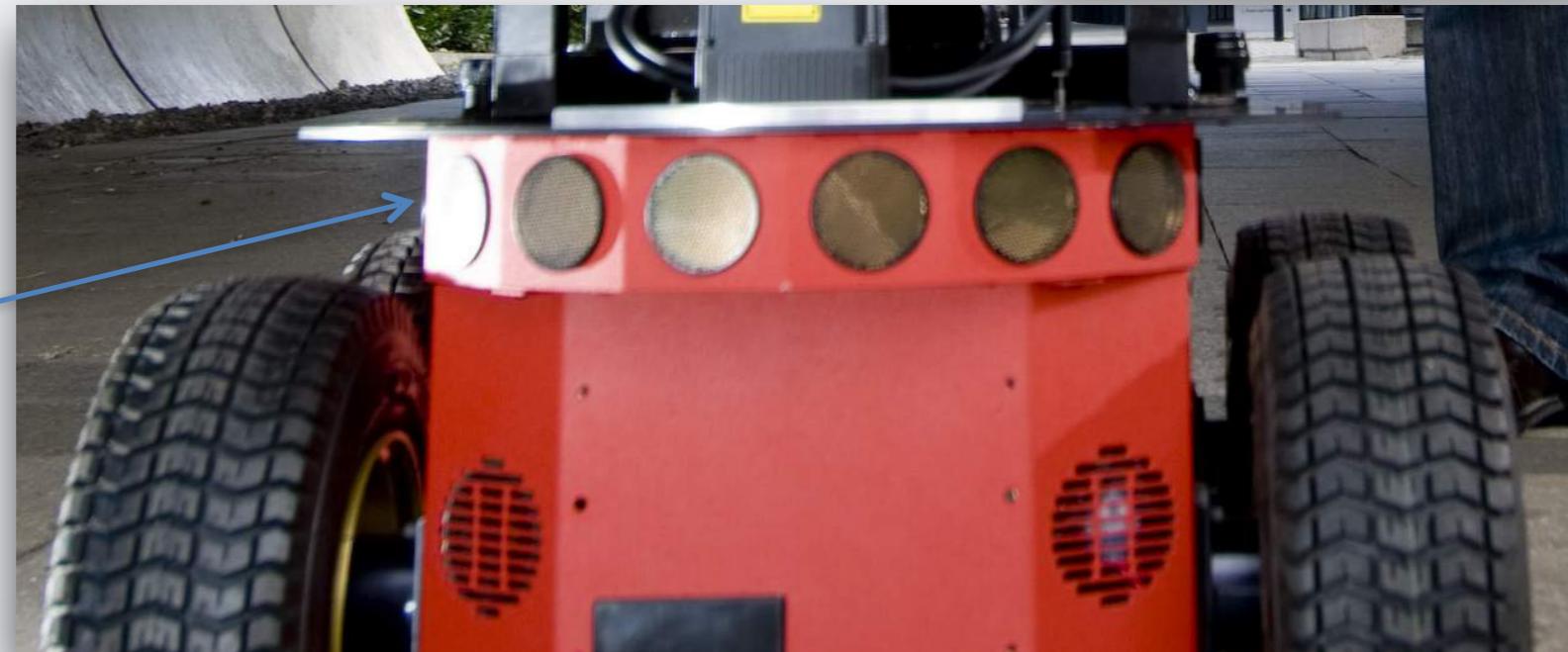
SONAR SENSOR

- ▶ Sonar
- ▶ ultrasonic signal
- ▶ v is speed of sound = 343 m/s
- ▶ processing is 'slow'

sonar reading

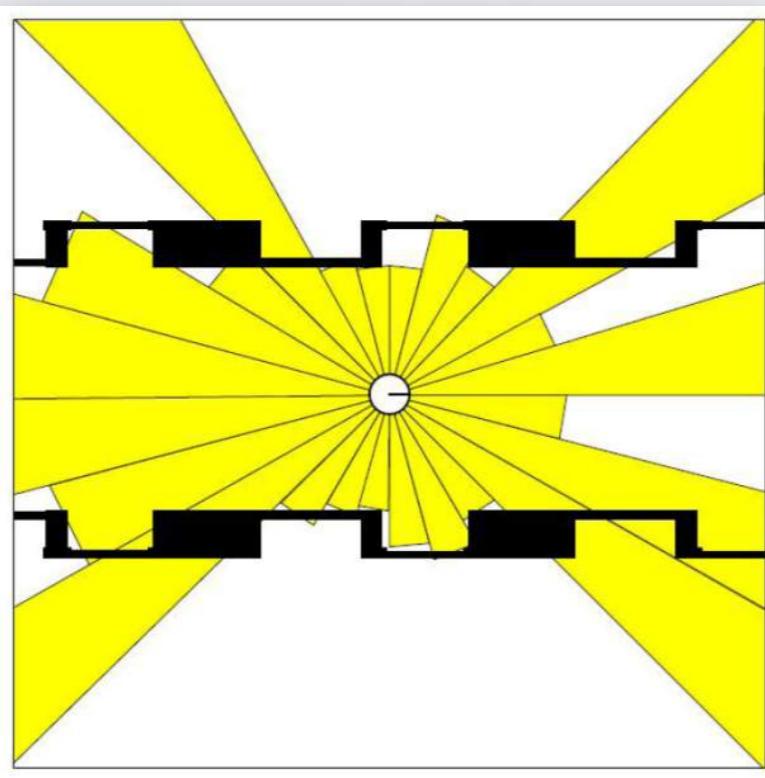


an array of
sonar sensors

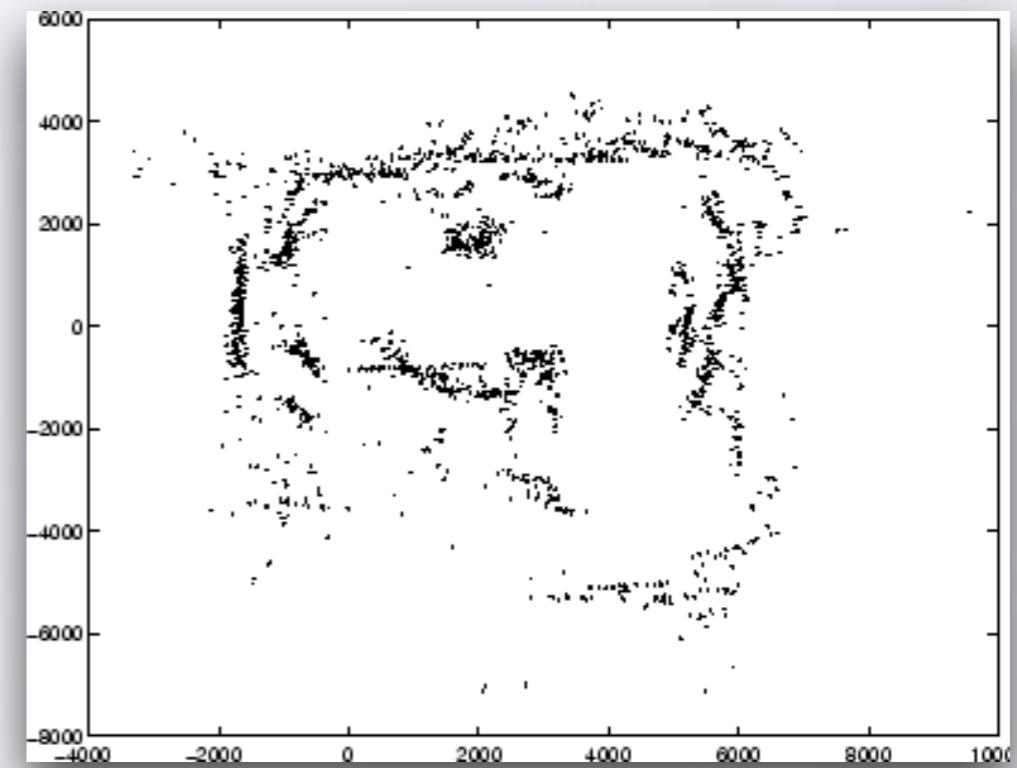


SONAR – APPLICATIONS

- ▶ Pros: cheap and good for obstacle avoidance
- ▶ Cons: slow and noisy



sonar reading in a
corridor



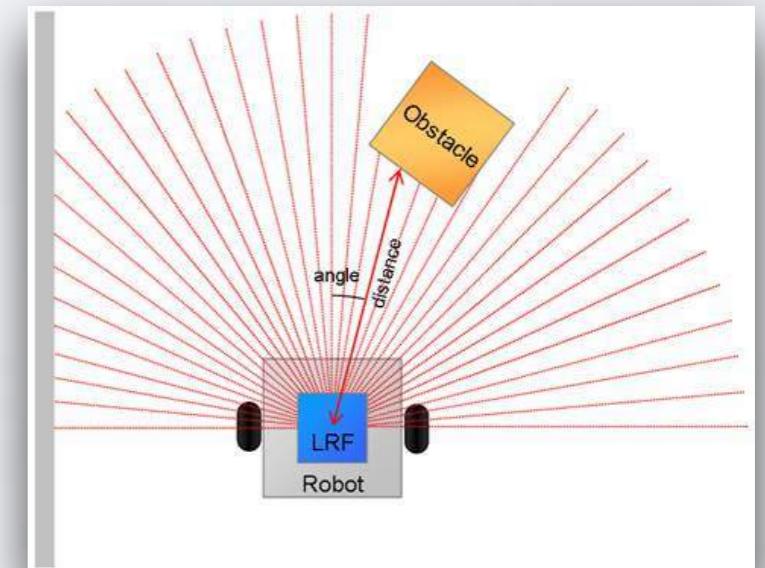
sonar map

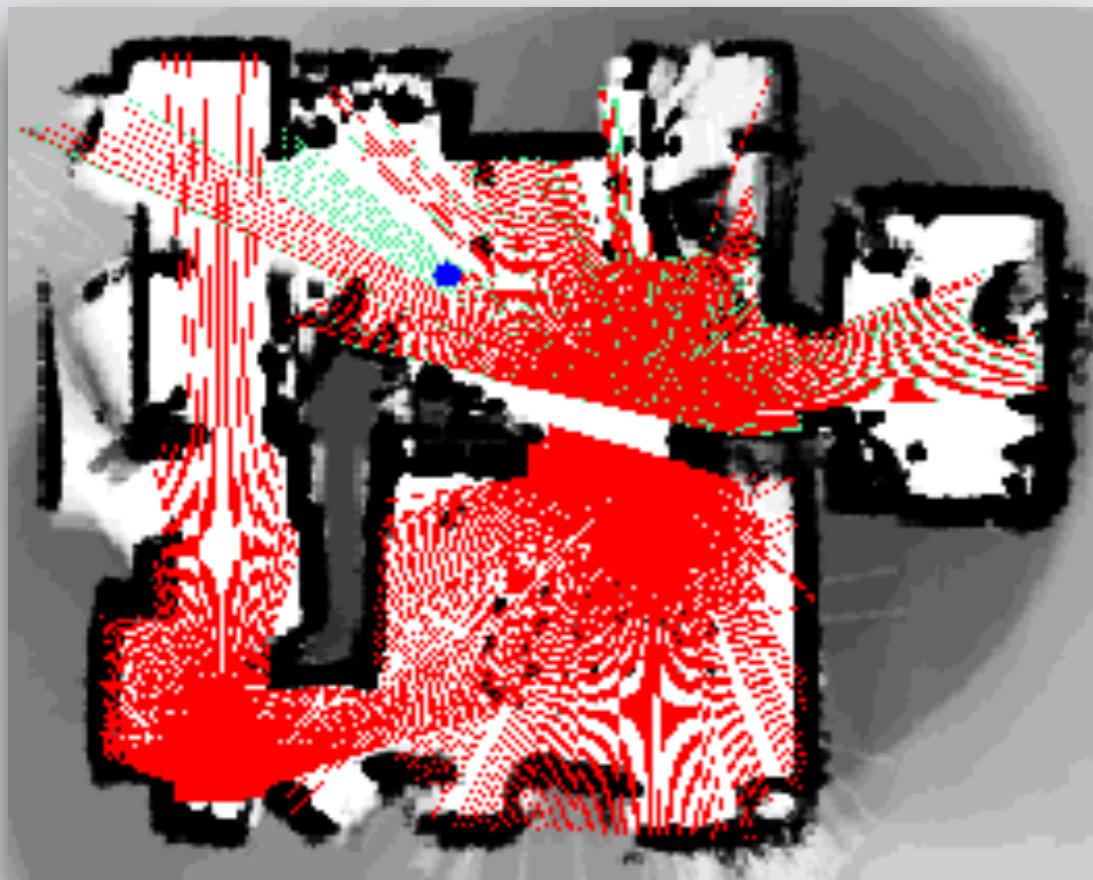
LASER SENSORS

- ▶ Principle
 - ▶ time of flight sensor (light)
 - ▶ pulsed laser and rotating mirror



- ▶ Characteristics
 - ▶ high precision (mm)
 - ▶ long range (tens of meters)
 - ▶ wide field of view
 - ▶ fast (~30 scans/s)





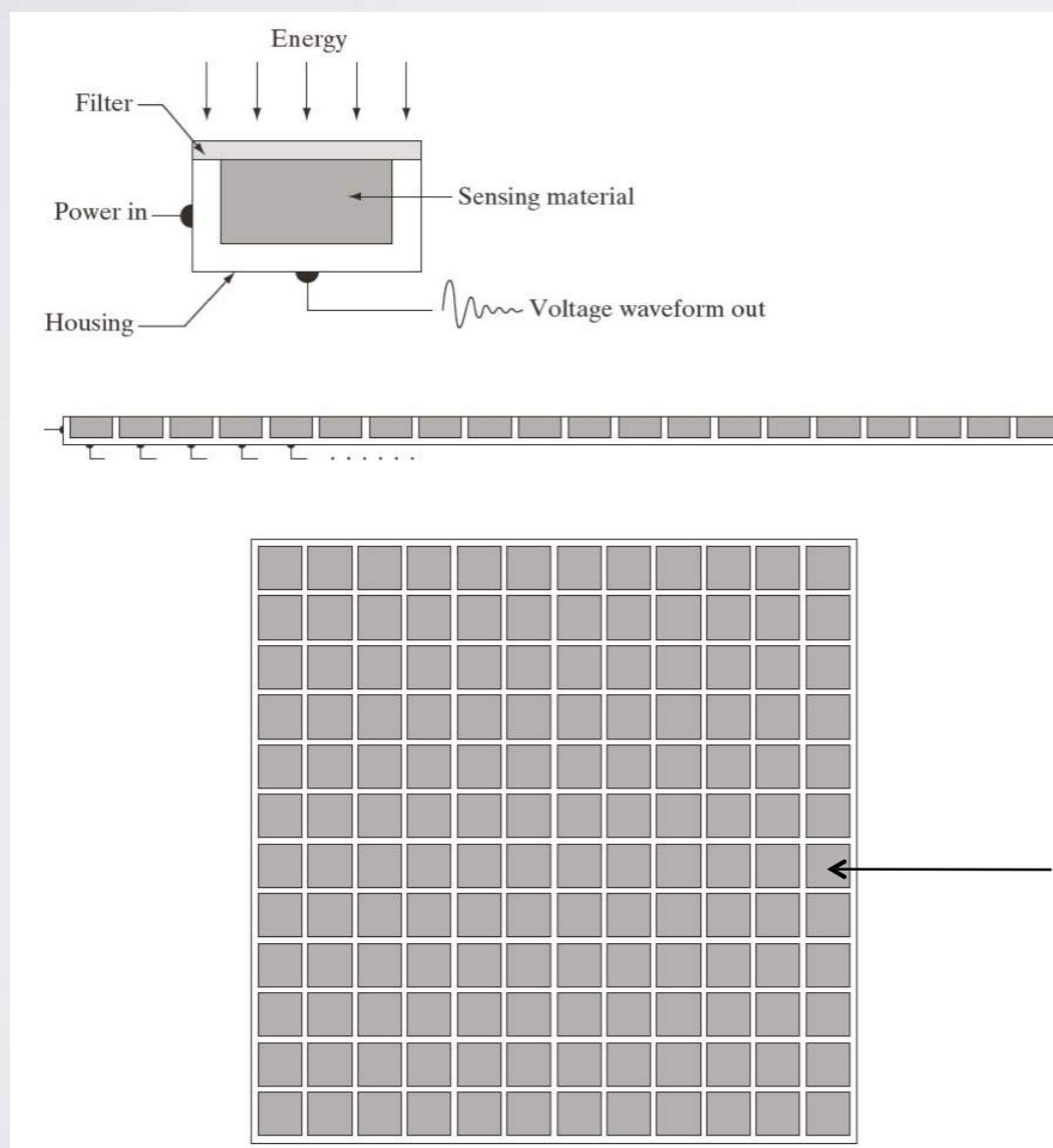
2d maps and people
detection



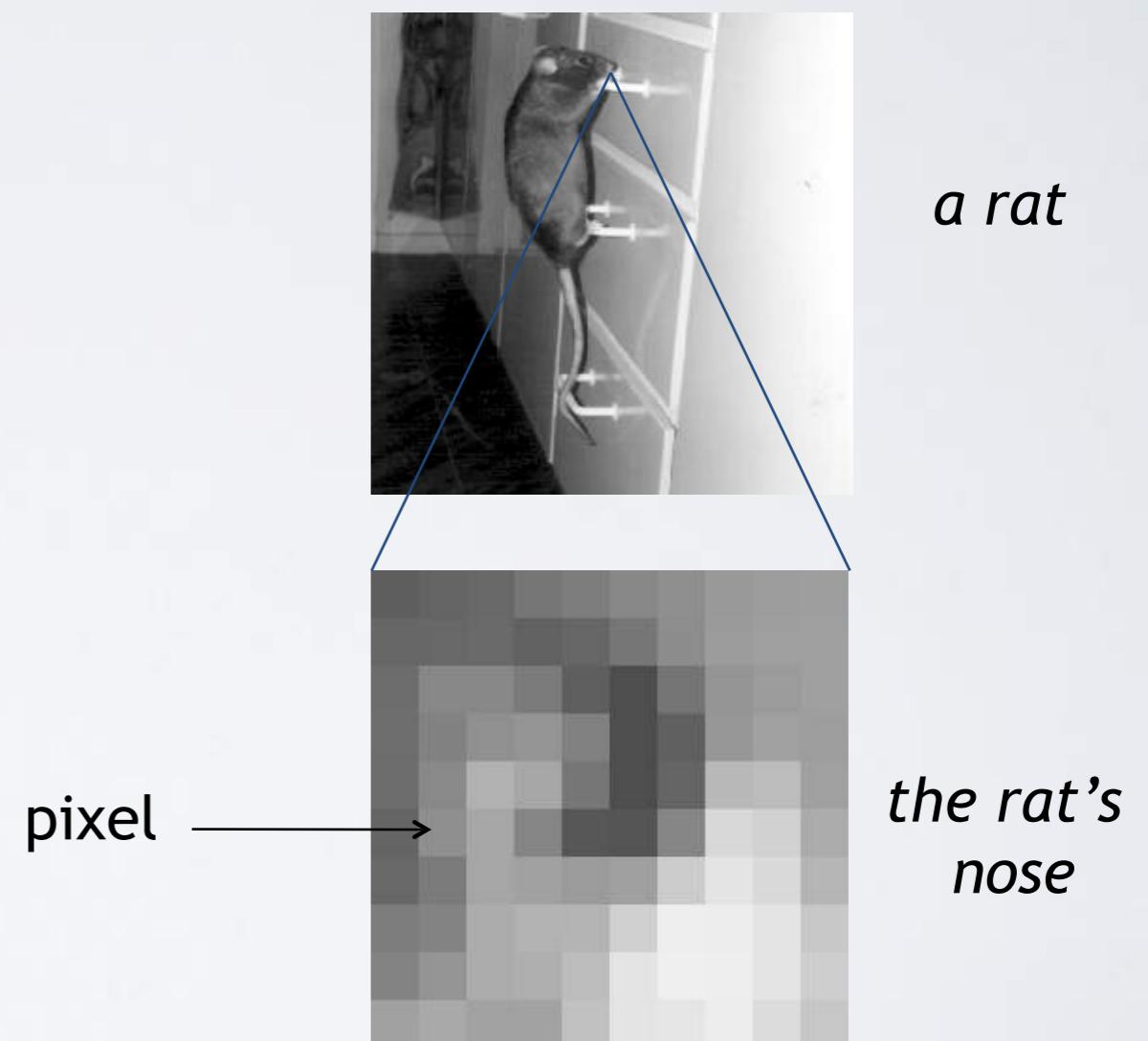
3d maps

VISION SENSOR

Vision Sensor



Digital Image



DIFFERENT TYPE OF VISION SENSORS

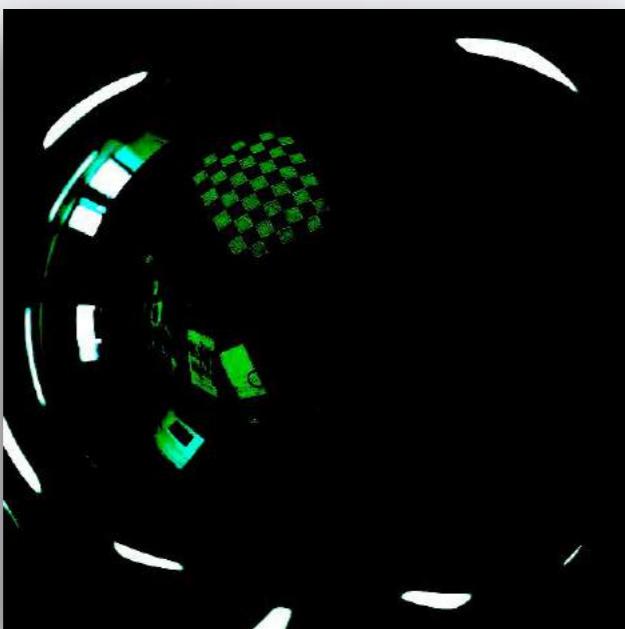
colour



thermal



omni-directional



stereo

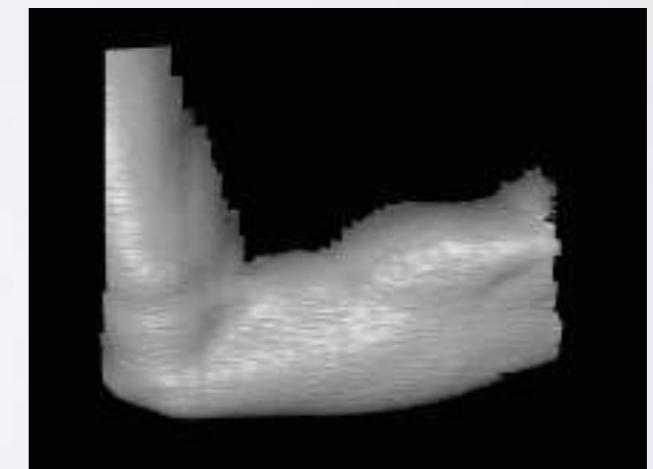
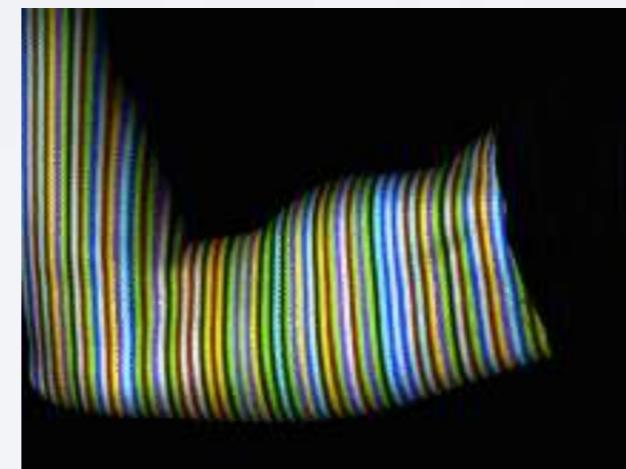
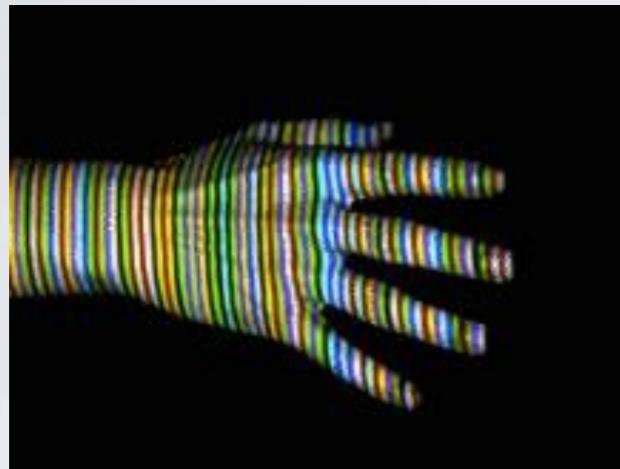


Kinect



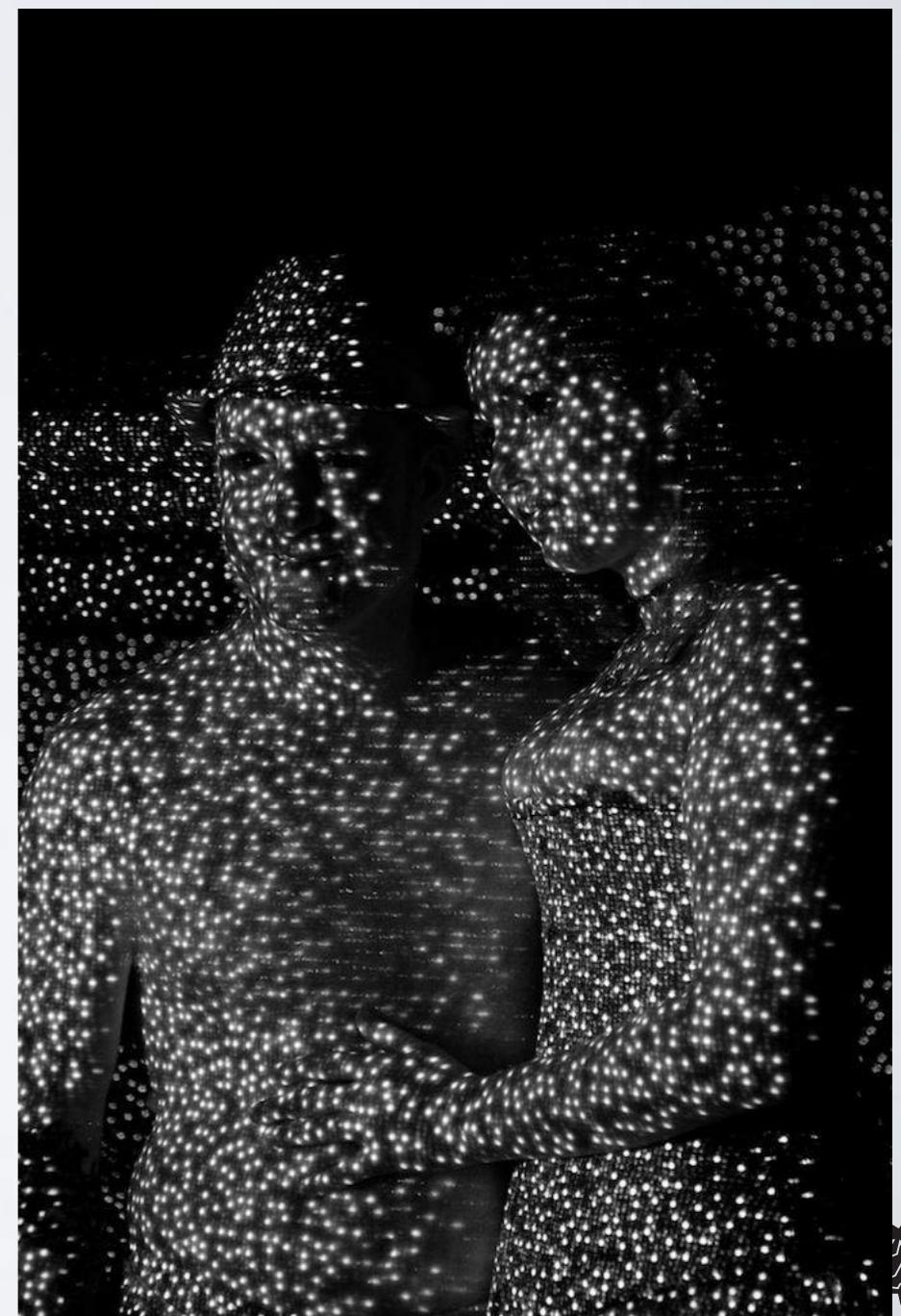
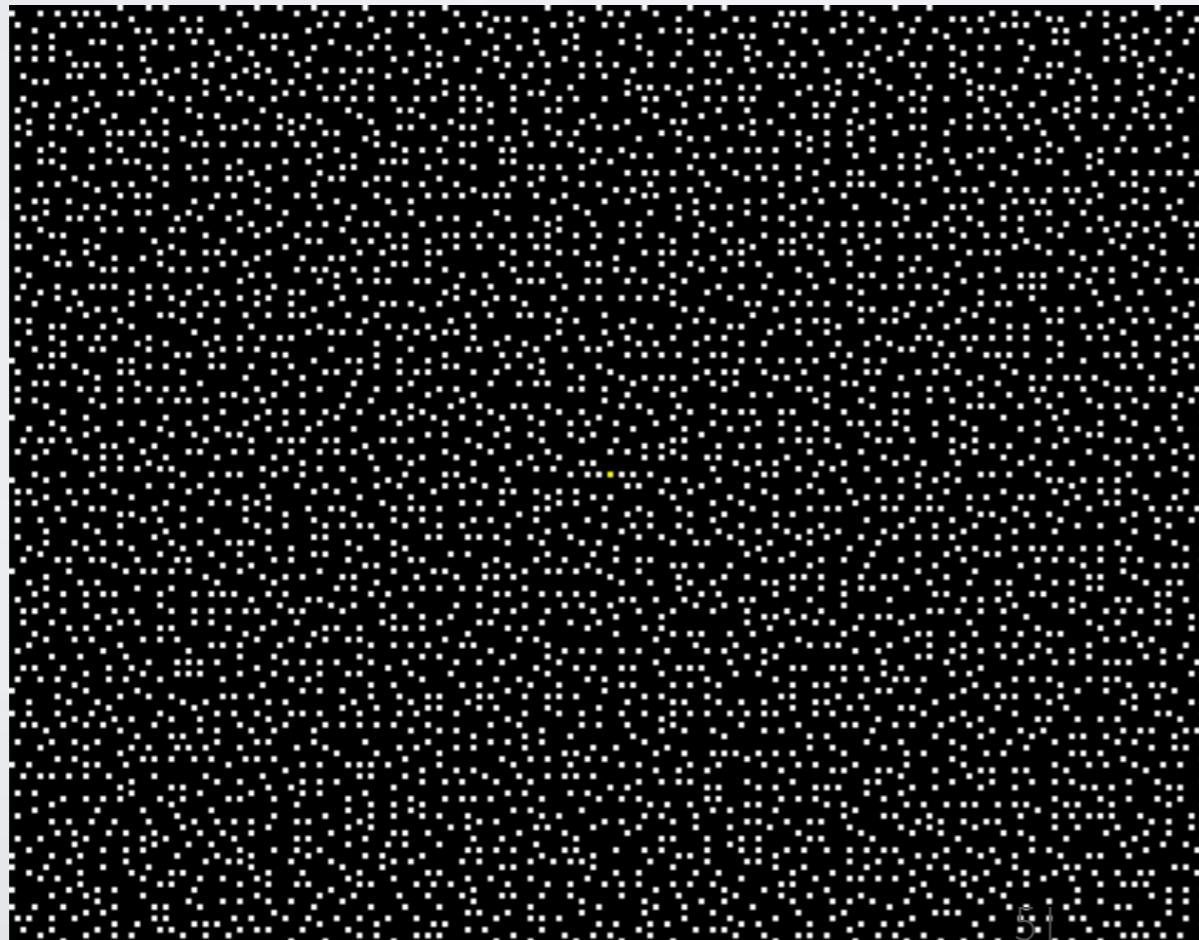
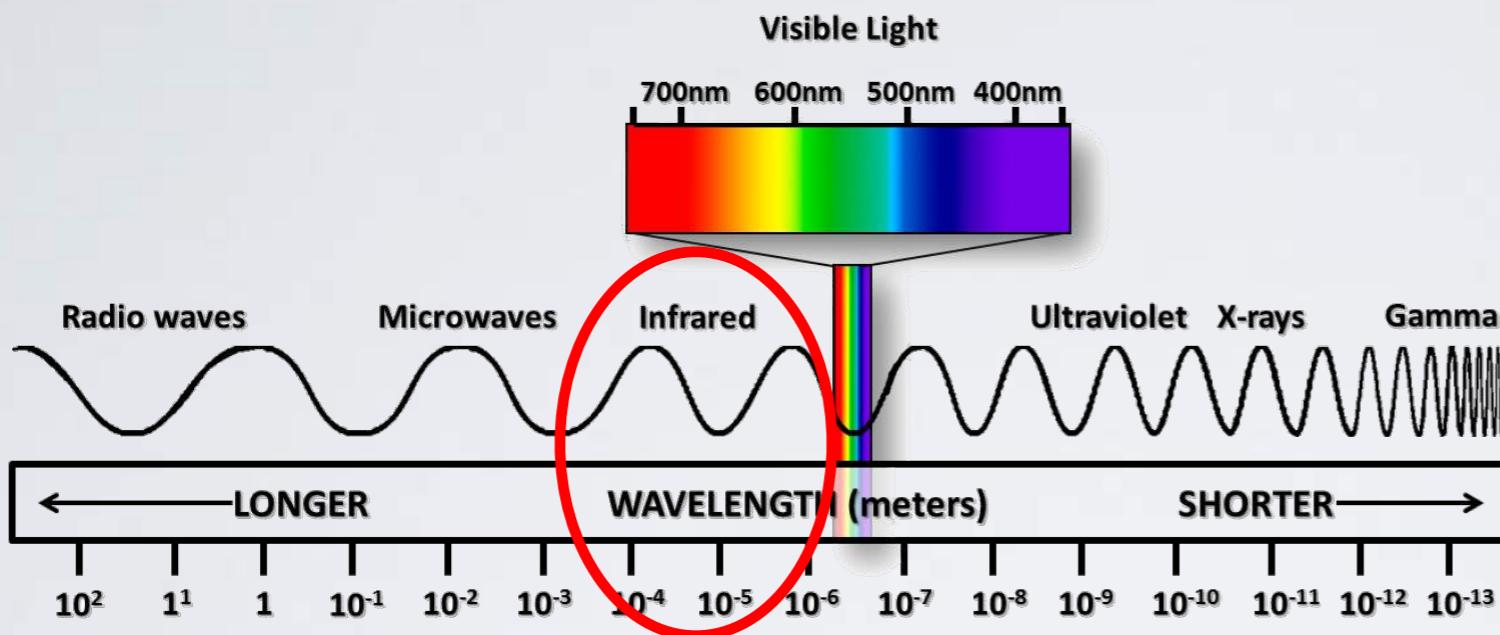
PrimeSense™
Natural Interaction

STRUCTURED LIGHT

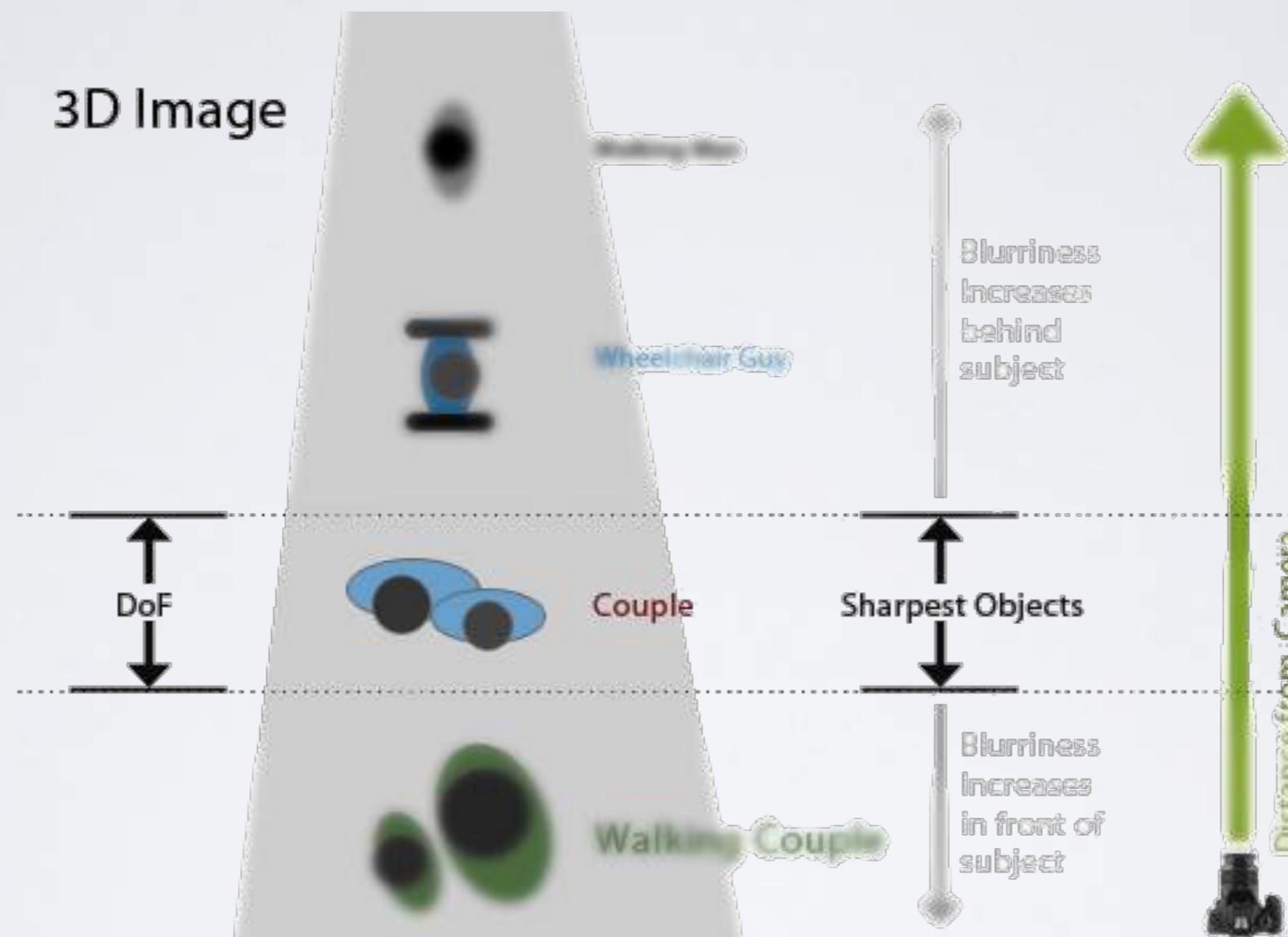


Zahng et al., 2002
<http://grail.cs.washington.edu/projects/moscan/>

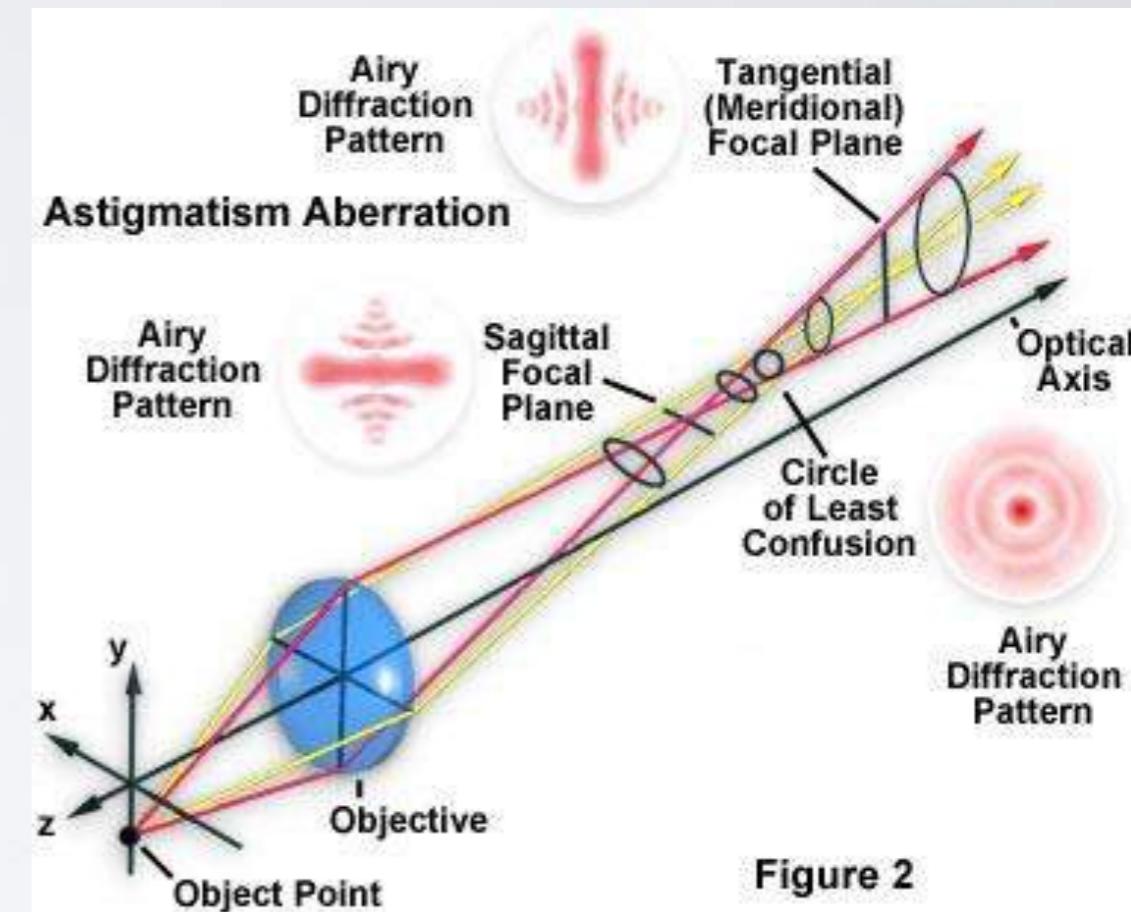
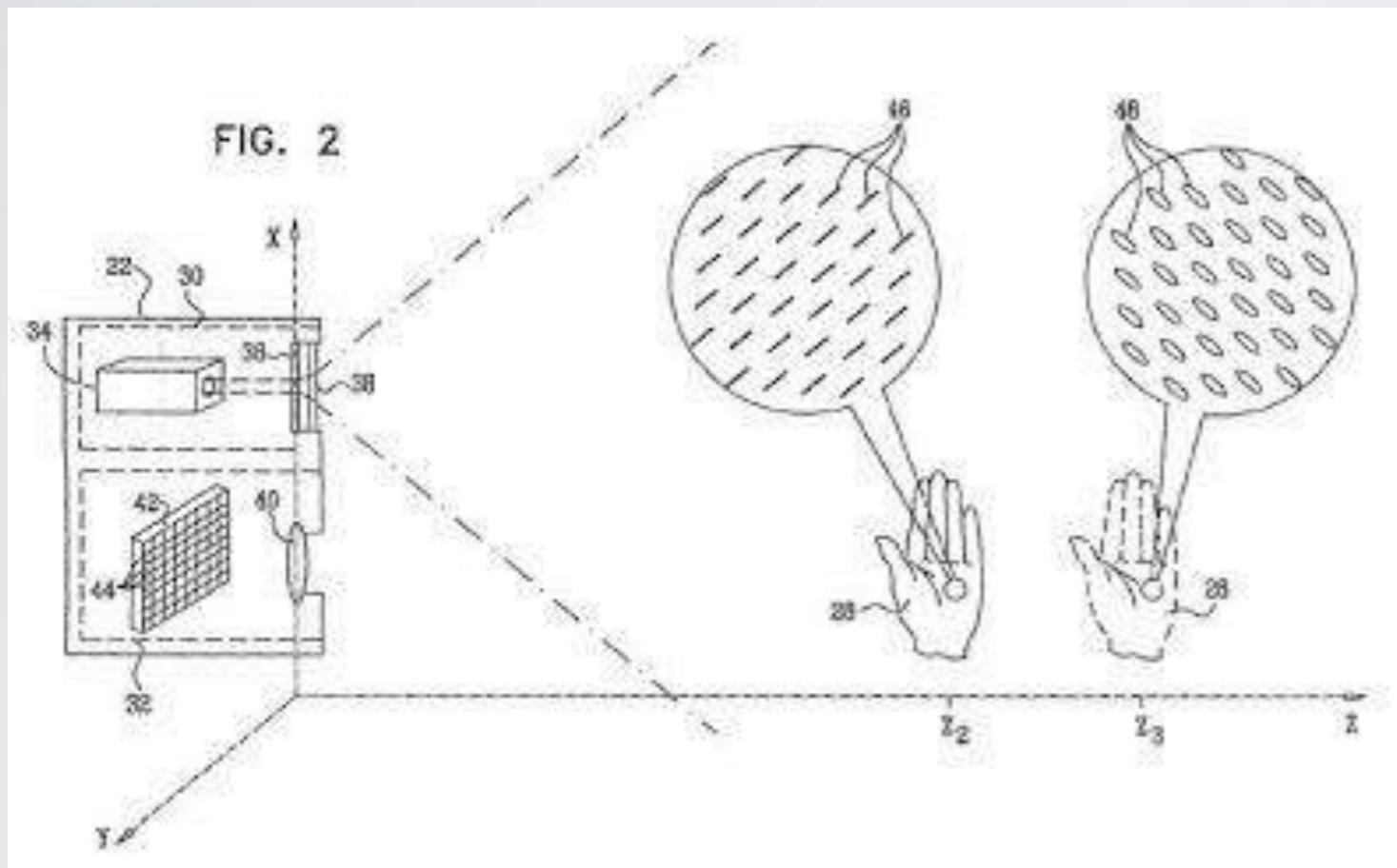
STRUCTURED LIGHT - KINECT



DEPTH FROM FOCUS

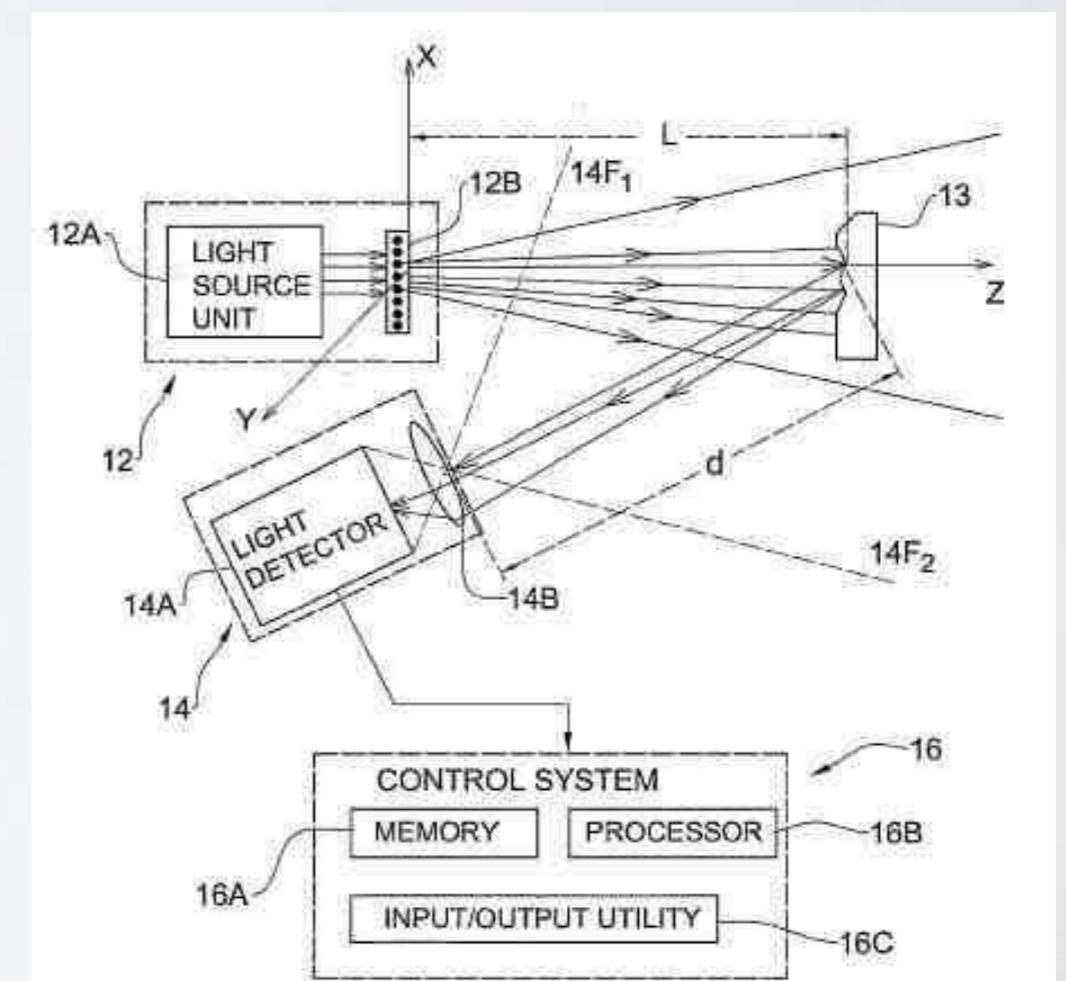
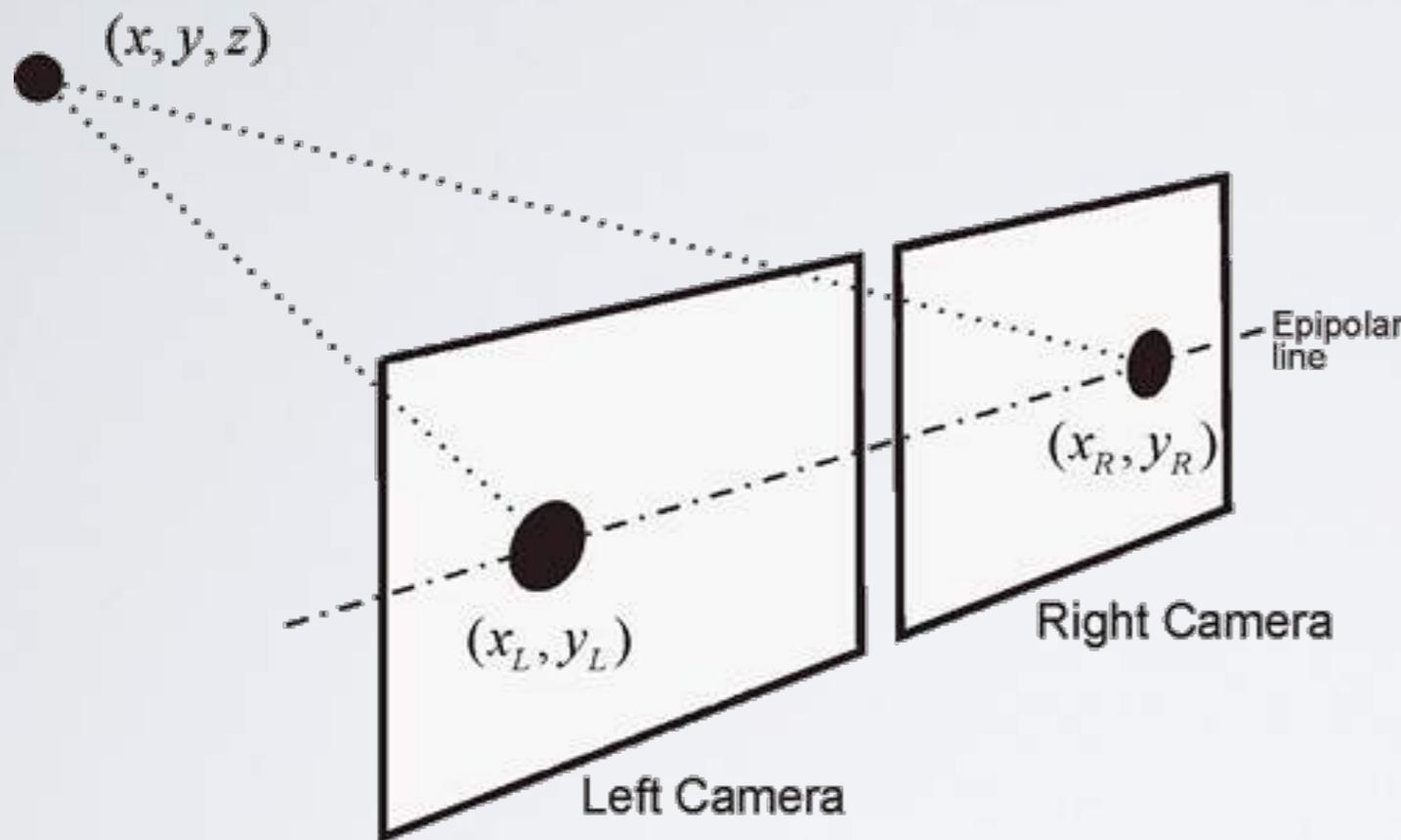


DEPTH FROM FOCUS - KINECT



“astigmatic” lens with different focal length in x and y direction

DEPTH FROM STEREO



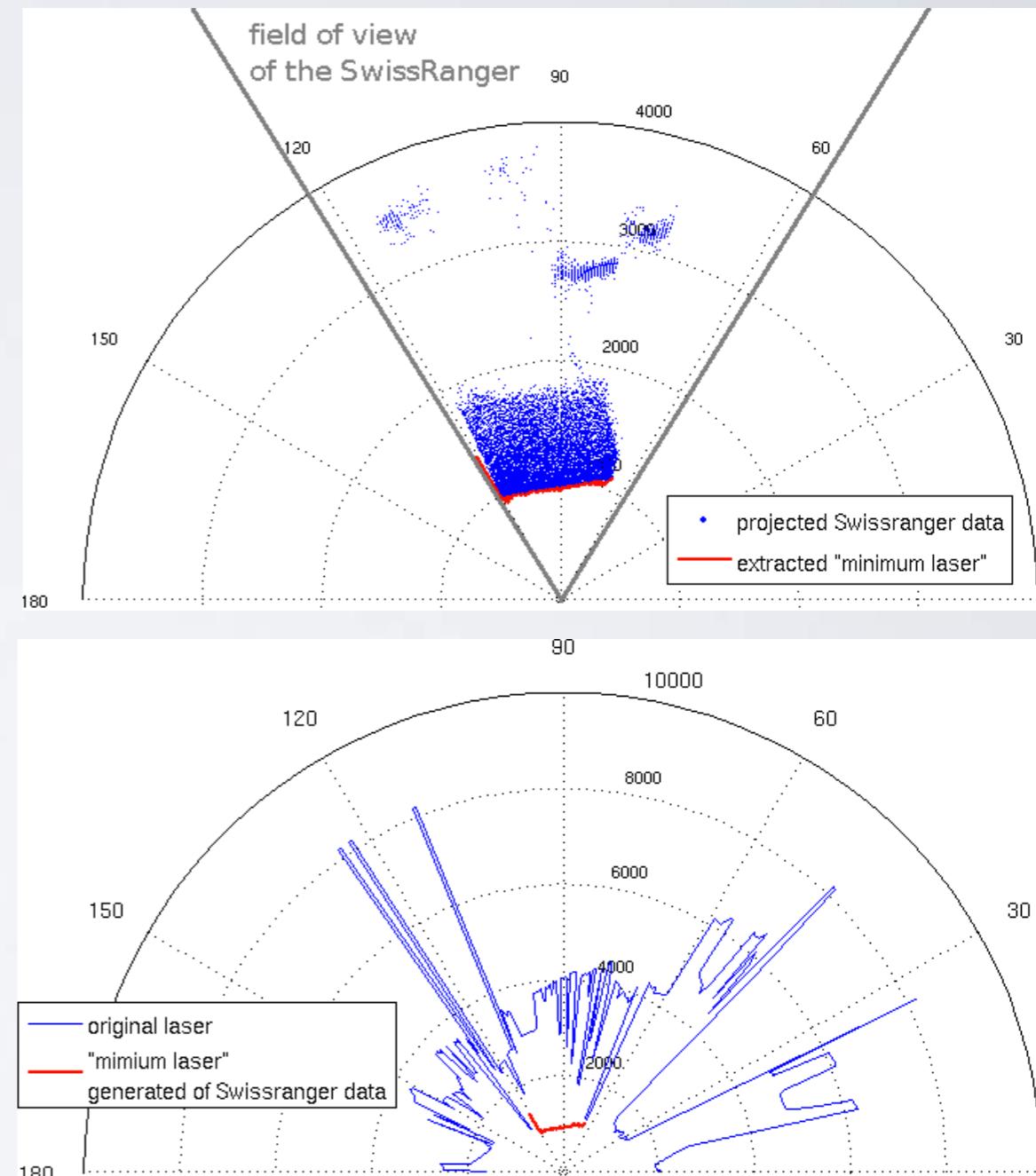
KINECT 2

- ▶ a “real” 512x424 pixel time-of-flight camera
- ▶ higher resolution
- ▶ higher frame rate
- ▶ increased field of view

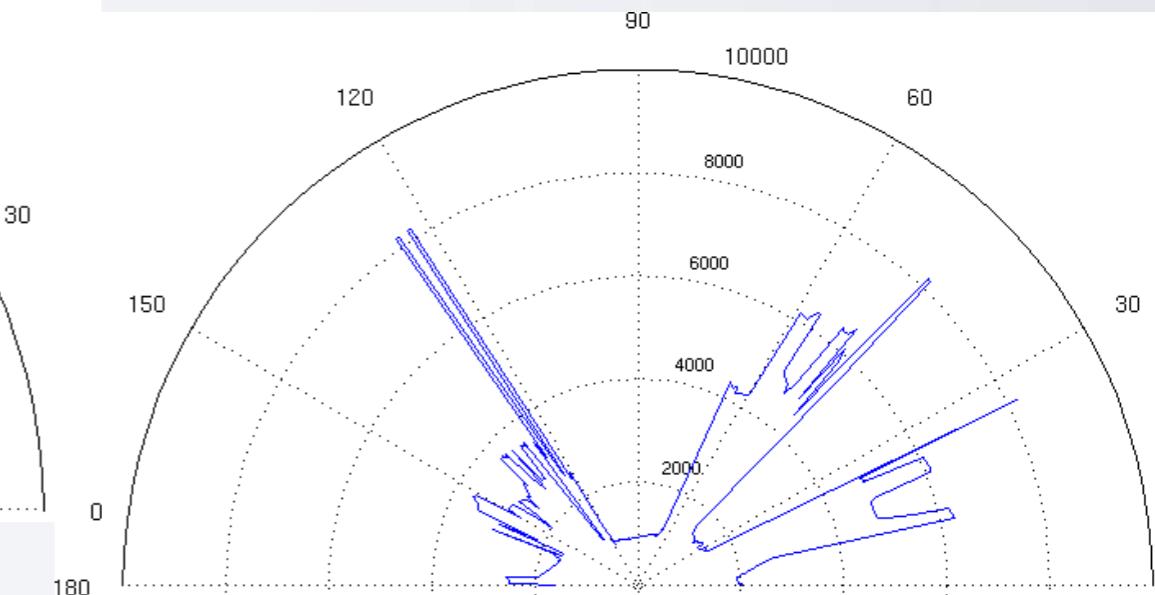
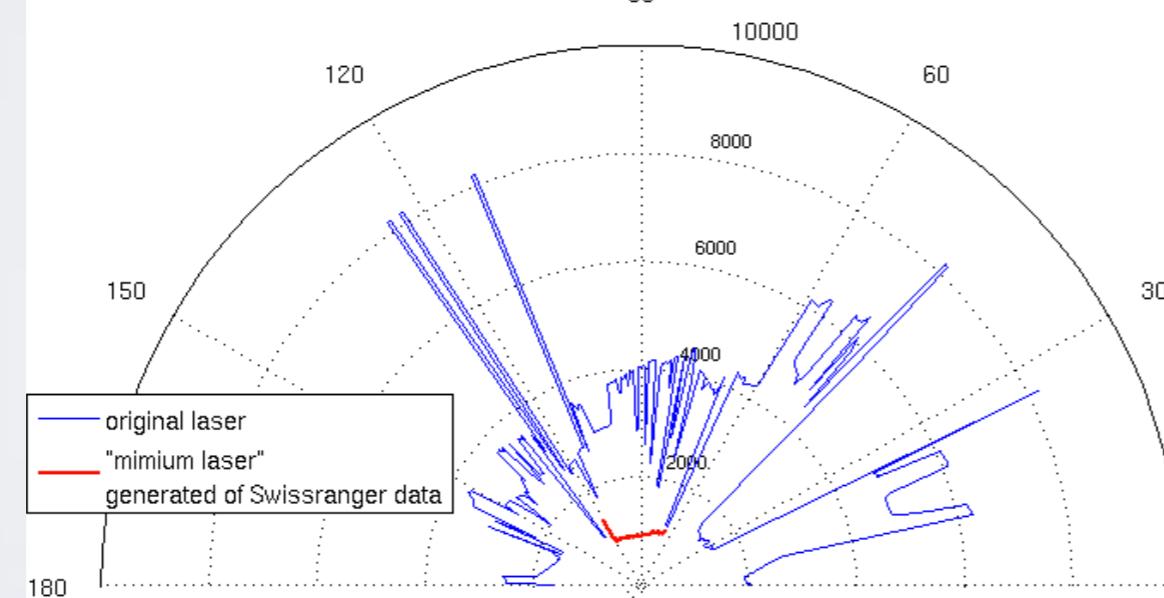
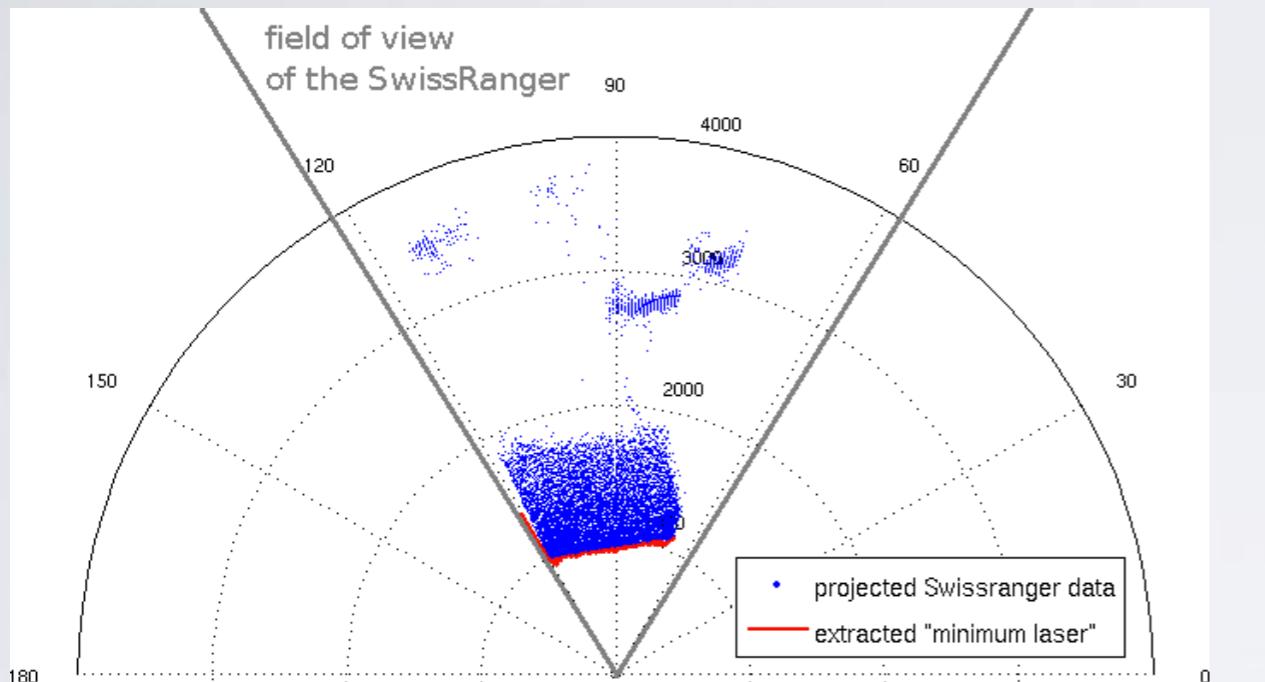


EMULATING A LASER SCANNER AND MERGING READINGS

- ▶ requires “calibration” of the two sensors (measuring where one is in relation to the other)
=> Rotation and Translation
- ▶ in 3D given by 6 degrees of freedom
- ▶ often represented as a 3×3 rotation matrix and a 3×1 translation vector

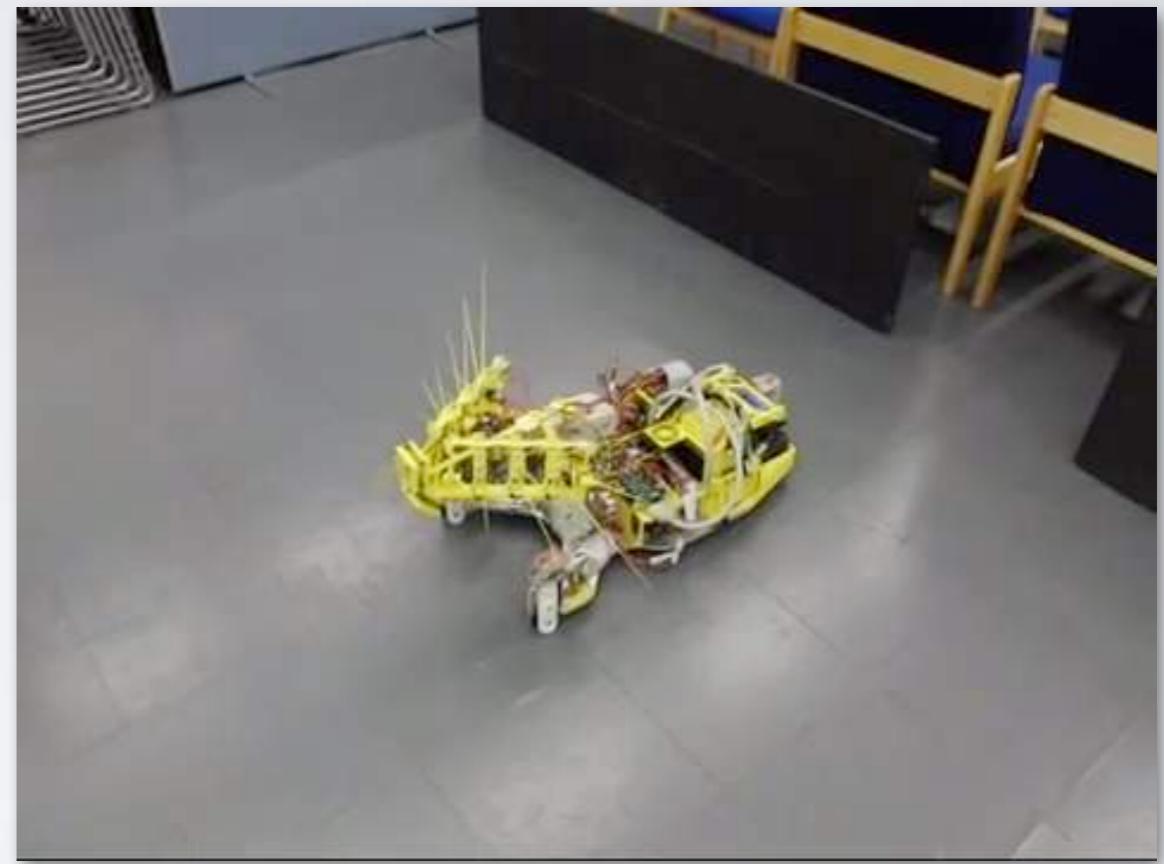


AVOIDING THE TABLE

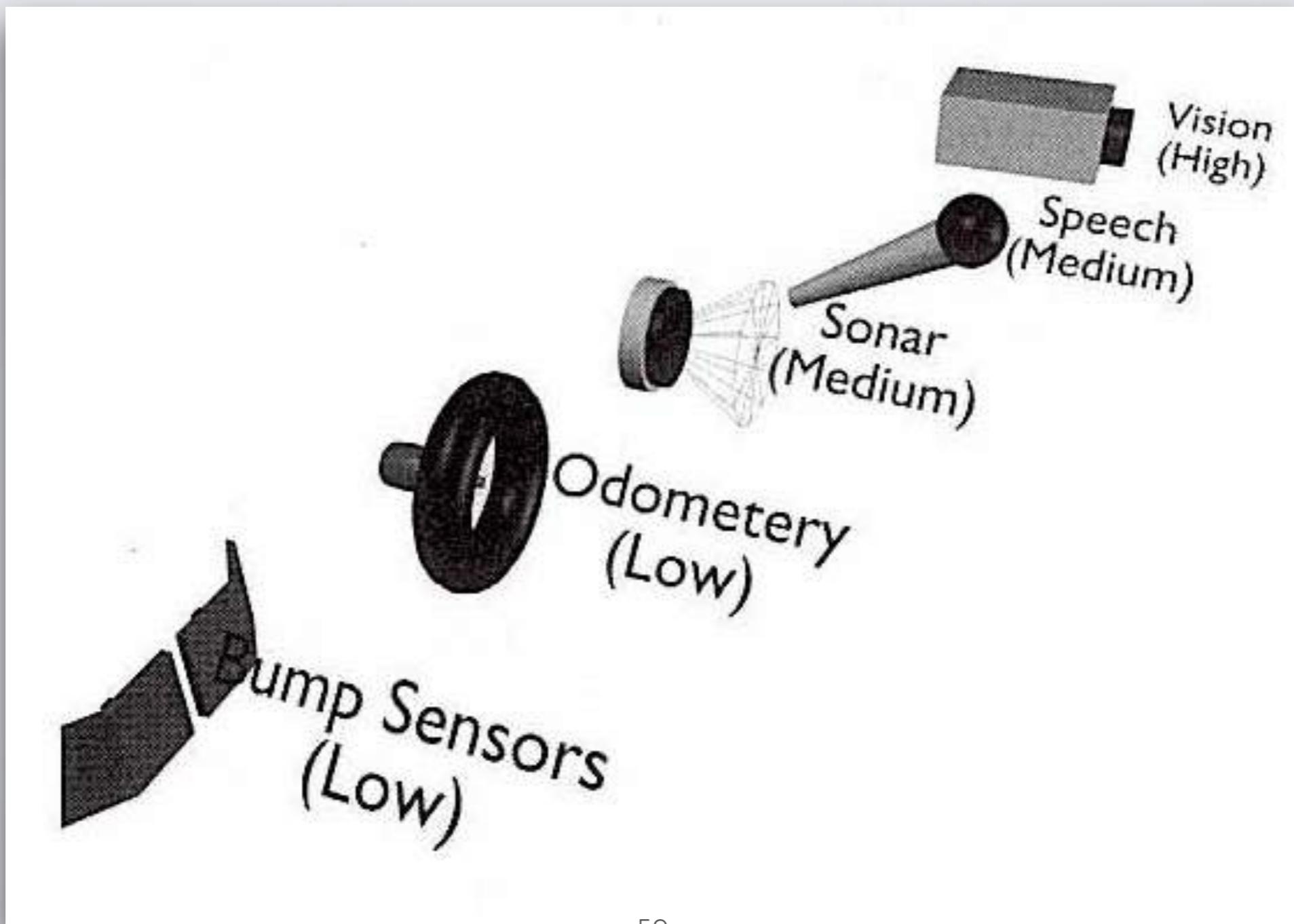


OTHER SENSORS

- ▶ Tactile
 - ▶ bumpers, whiskers, buttons
- ▶ Direction and orientation
 - ▶ compass, gyroscope, accelerometers
- ▶ Global position
 - ▶ GPS
- ▶ Motion
 - ▶ Gyroscope
- ▶ Temperature, sound, even smell!



HIERARCHY OF PROCESSING REQUIREMENTS

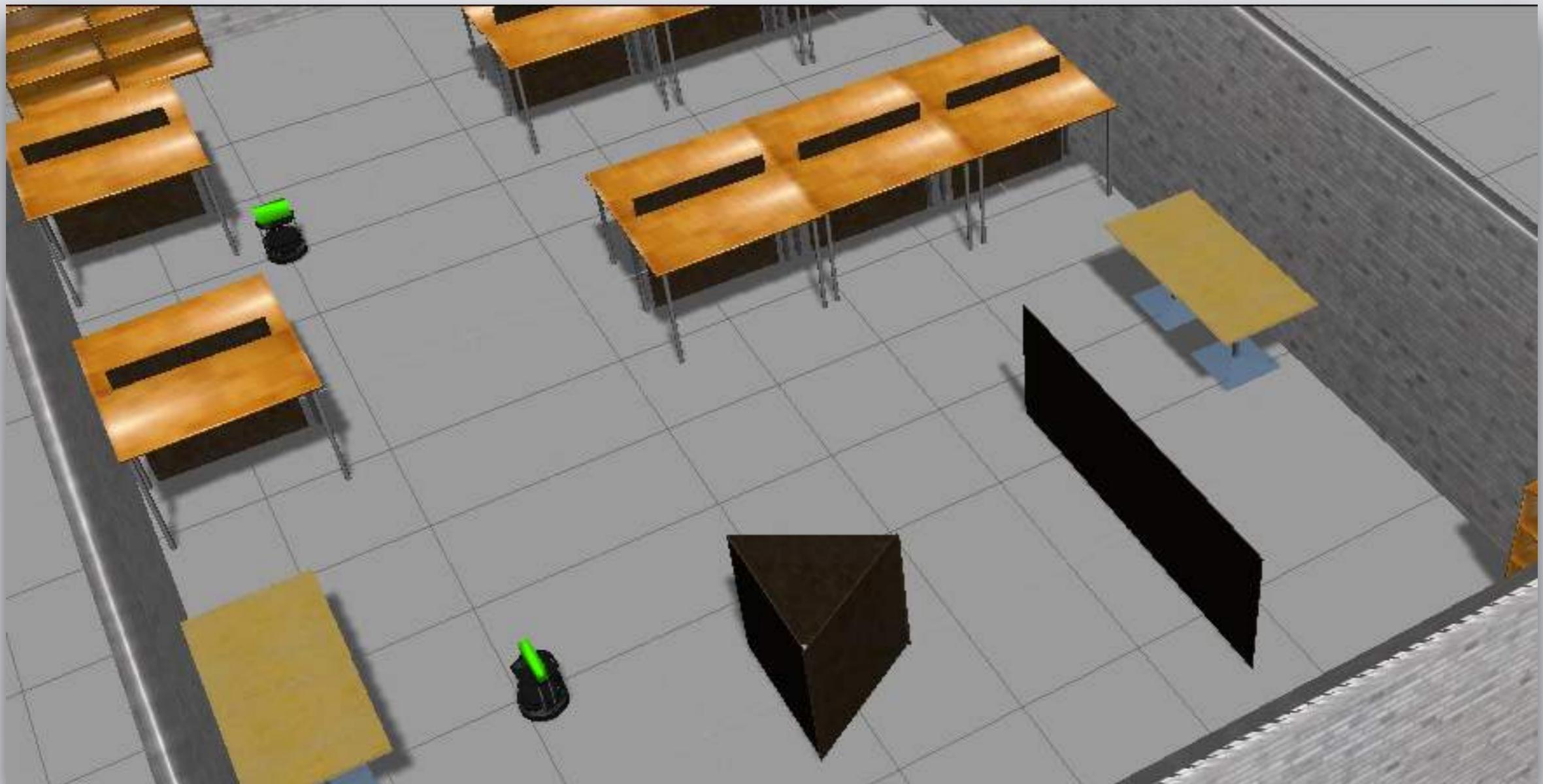


PERCEPTION

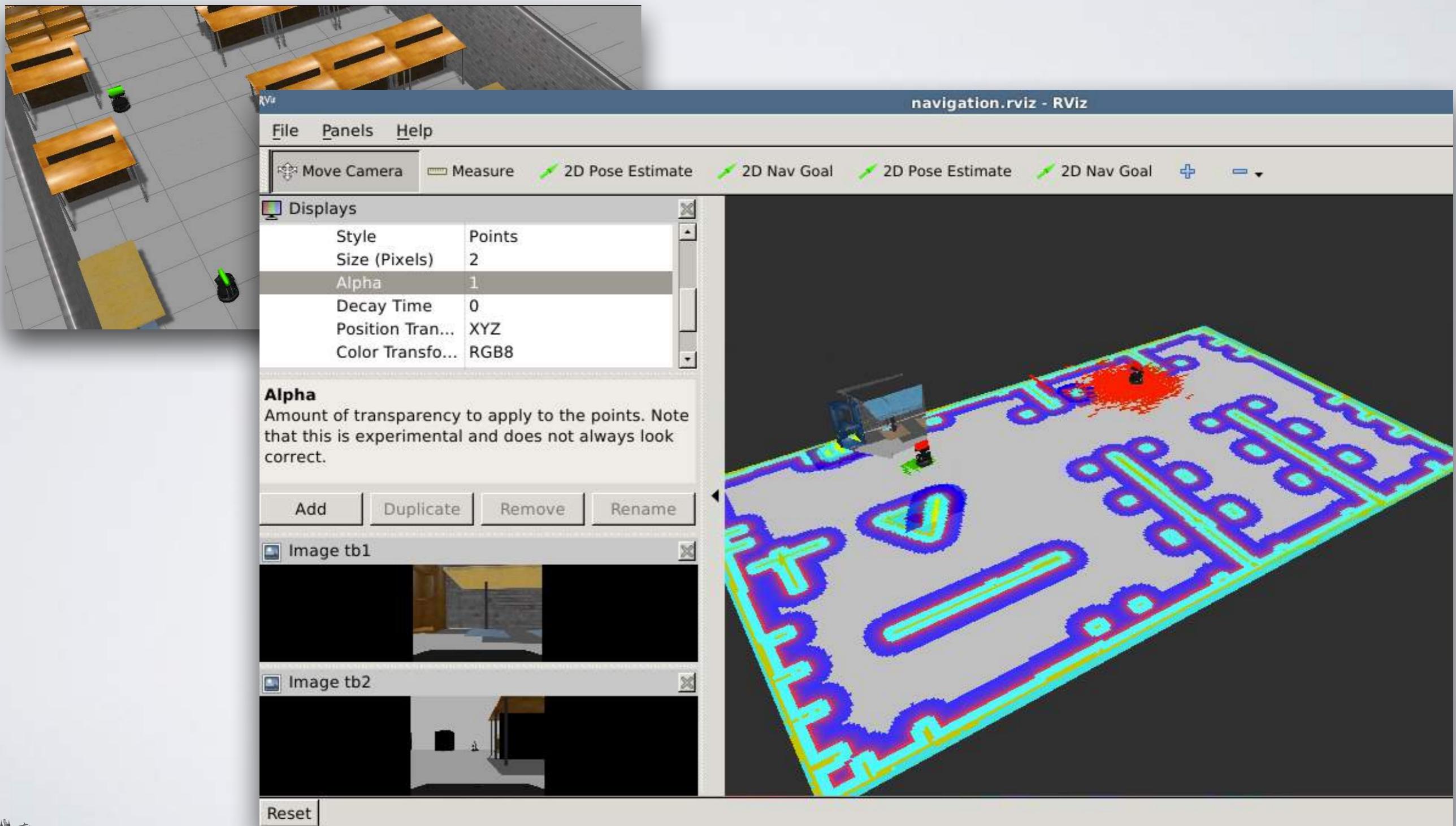
- ▶ Data Interpretation and Processing
- ▶ sensors often do not provide direct measurements nor provide the exact values
- ▶ some sensors provide very rich information (e.g. vision) that needs to be reduced
- ▶ some sensors provide very sparse information that needs to be interpolated
- ▶ The **data bandwidth** of sensors differs significantly!



SIMULATED TURTLEBOTS



SIMULATED TURTLEBOTS





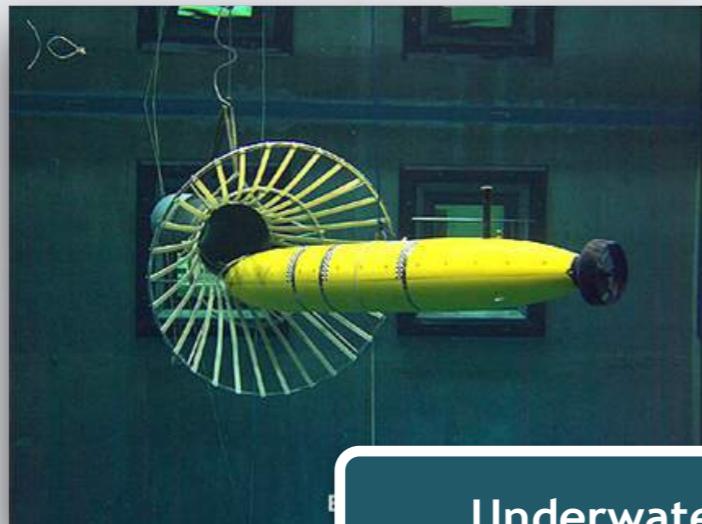
APPLICATIONS

Which robot is more autonomous?

Which one of these robots is more autonomous?



DIFFERENT ROBOT – DIFFERENT APPLICATION



Underwater

autonomous exploration



Tracked

inspection of radiated areas
(Pioneer)



Airborne

surveillance (MQ-1 Predator)



Tracked

gutter cleaner (Jool)

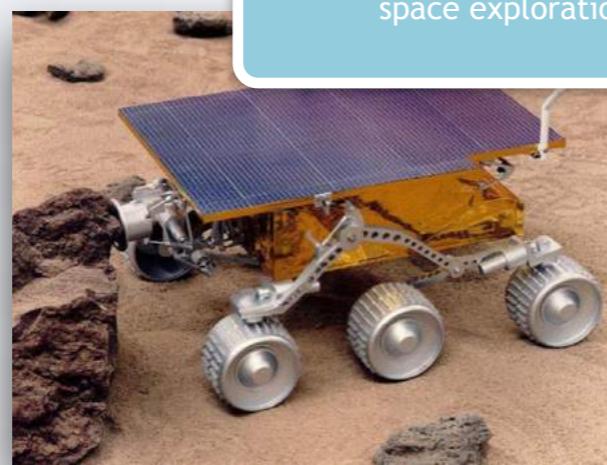
DIFFERENT ROBOT – DIFFERENT APPLICATION



Legged
exploration of volcanoes
(Sojourner)



Legged
transportation (BigDog)



Wheeled
space exploration



Wheeled
autonomous car (DARPA Grand
Challenge)

A ROBOT IN EVERY HOME

- ▶ Service Robots
 - ▶ cohabitant with humans in their everyday environments
- ▶ Bill Gates predicts (in 2006):
 - ▶ “in 20 years (2026) there will be a robot in every home”
 - ▶ read his [article](#) in Scientific American

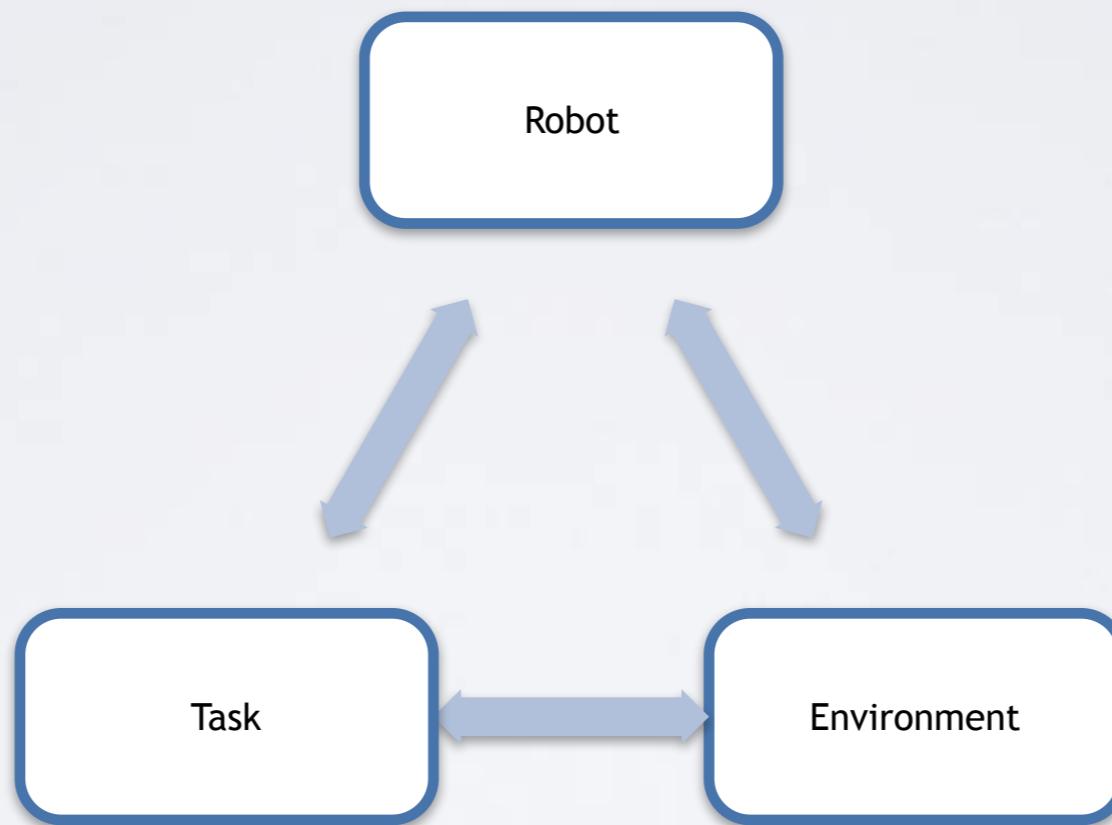


Overview/*The Robotic Future*

- The robotics industry faces many of the same challenges that the personal computer business faced 30 years ago. Because of a lack of common standards and platforms, designers usually have to start from scratch when building their machines.
- Another challenge is enabling robots to quickly sense and react to their environments. Recent decreases in the cost of processing power and sensors are allowing researchers to tackle these problems.
- Robot builders can also take advantage of new software tools that make it easier to write programs that work with different kinds of hardware. Networks of wireless robots can tap into the power of desktop PCs to handle tasks such as visual recognition and navigation.

ROBOT, TASK, ENVIRONMENT

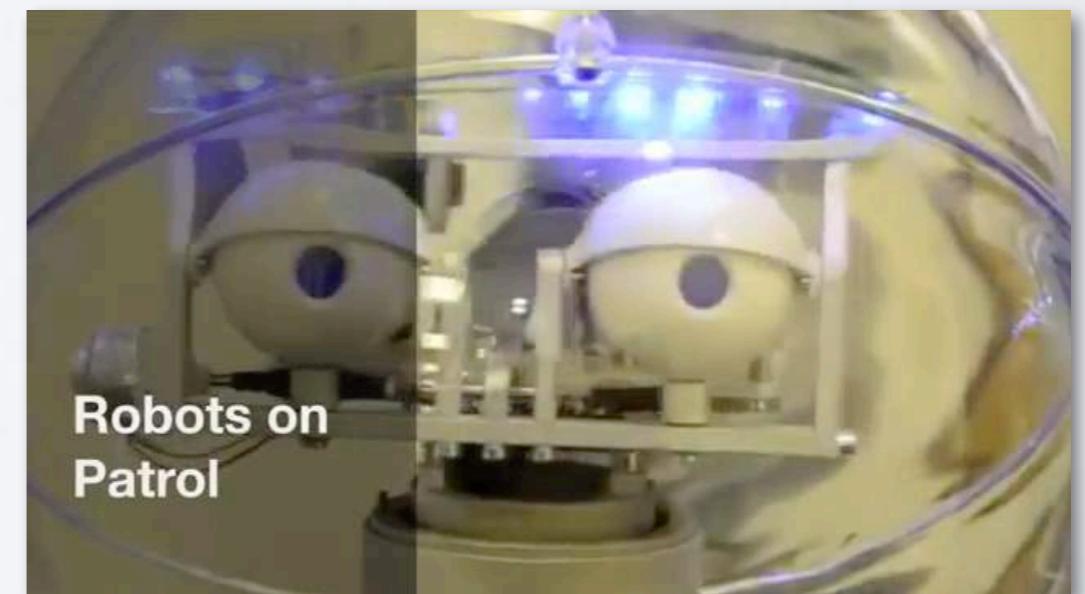
- ▶ There is no general purpose robot



- ▶ These three aspects are dependent upon and influence each other

ROBOTICS RESEARCH AT LINCOLN

- ▶ Our webpage: lcas.lincoln.ac.uk
 - ▶ Human-Centered Robotics
 - ▶ Agri-Food Technology
 - ▶ Bio-inspired Embedded Systems
 - ▶ Learning for Autonomous Systems
- ▶ Applications
 - ▶ Security
 - ▶ Assistive Care
 - ▶ Agricultural Robotics
 - ▶ Intelligent Transportation
 - ▶ Nuclear Robotics



[FP7 Project STRANDS](http://FP7-Project-STRANDS)

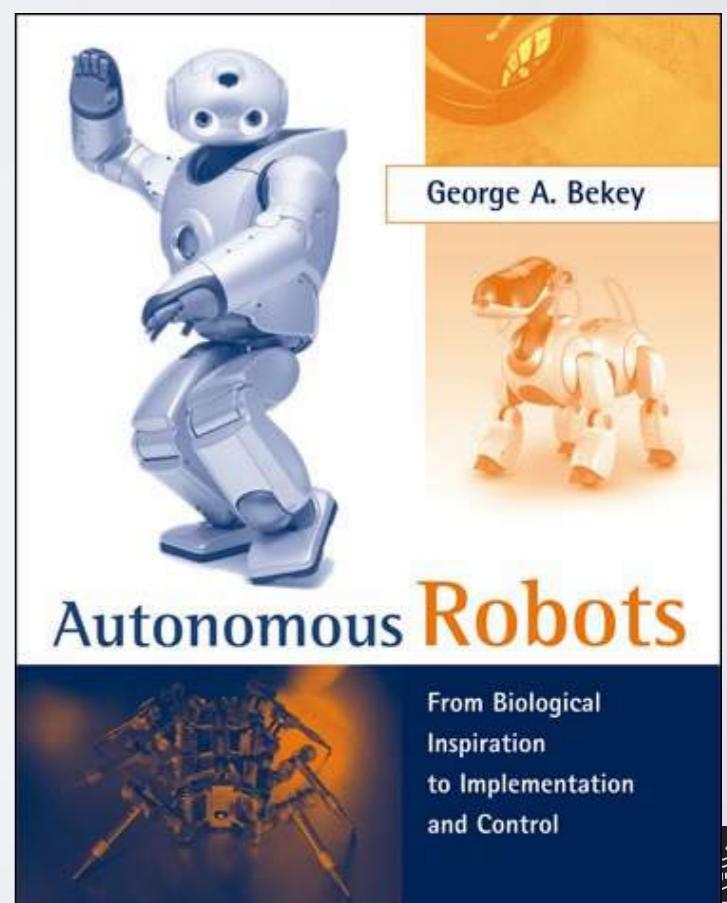
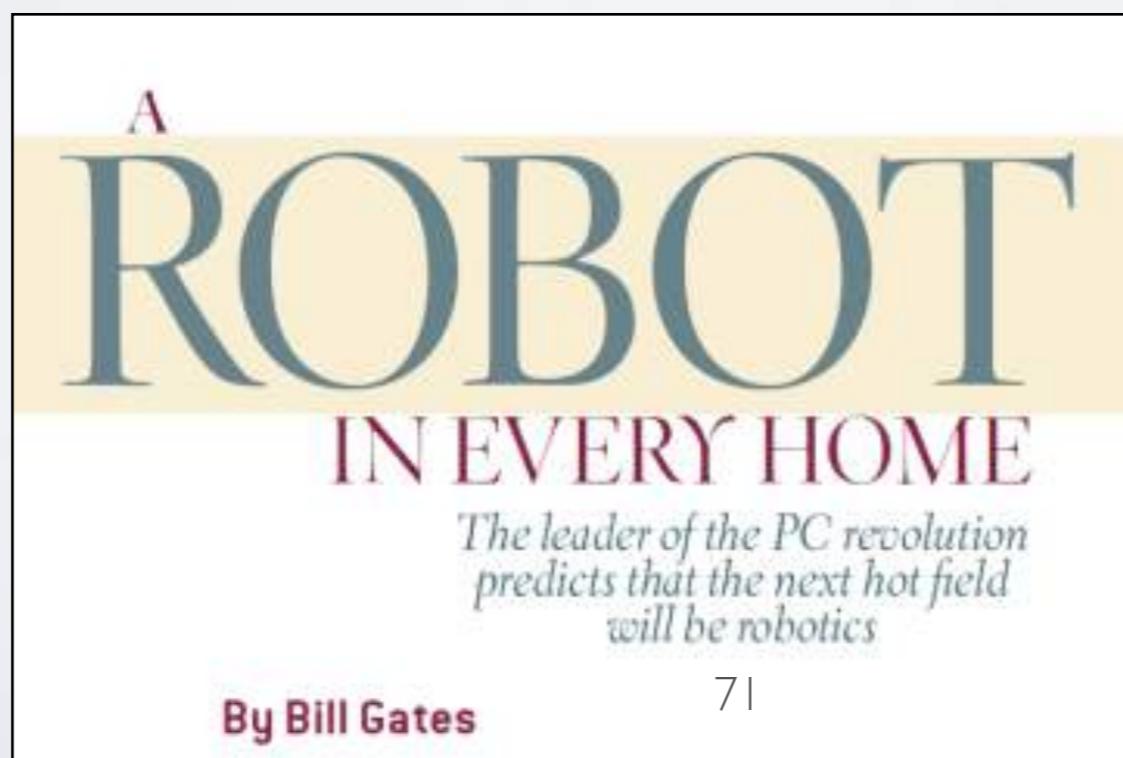
ROBOTICS RESEARCH AT LINCOLN

- ▶ Our webpage: lcas.lincoln.ac.uk
 - ▶ Human-Centered Robotics
 - ▶ Agri-Food Technology
 - ▶ Bio-inspired Embedded Systems
 - ▶ Learning for Autonomous Systems
- ▶ Applications
 - ▶ Security
 - ▶ Assistive Care
 - ▶ Agricultural Robotics
 - ▶ Intelligent Transportation
 - ▶ Nuclear Robotics

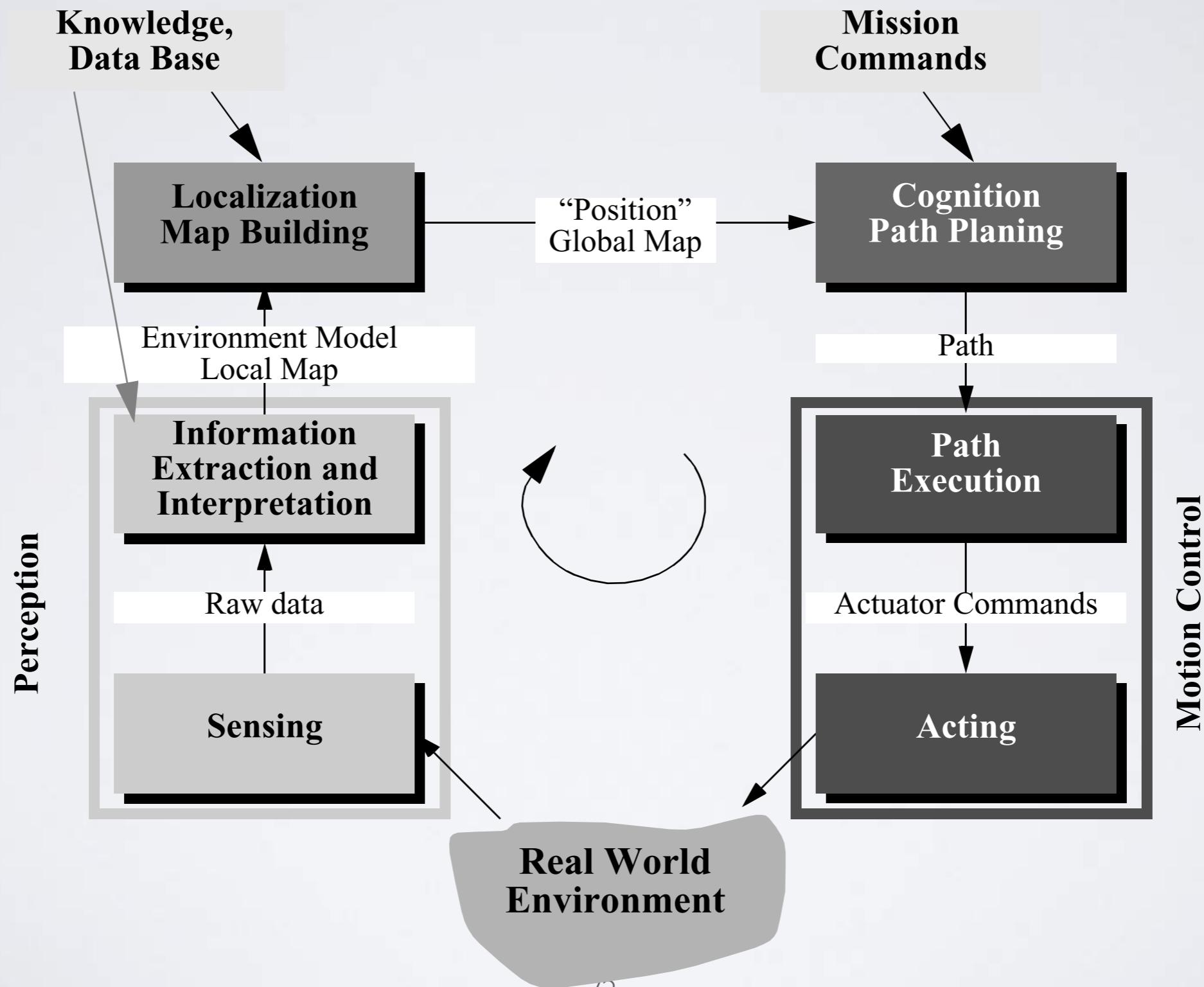


RECOMMENDED READING

- ▶ Gates, B. *A Robot in Every Home*, Scientific American, 2006
- ▶ Siegwart et al. Autonomous Mobile Robots, 2004
(chapter 1, also on blackboard)
- ▶ Bekey, G.A. Autonomous robots –
Chapter 1, also Section 4.1



SUMMARY



Thank you for listening!
Any questions ?



End of Lecture Feedback

When survey is active, respond at PollEv.com/mhanheide

MH

0 surveys done

↻ 1 survey underway

Start the presentation to see live content. Still no live content? Install the app or get help at PollEv.com/app

CMP3103M AMR

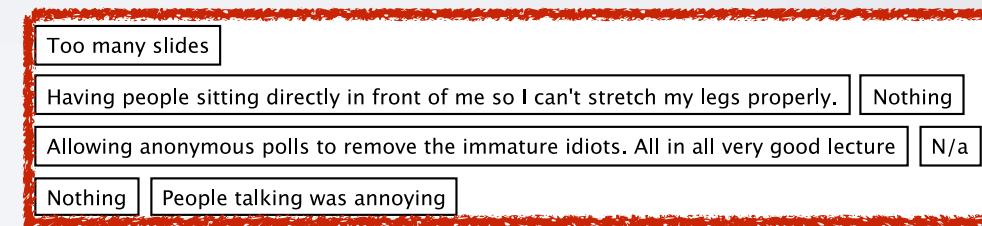
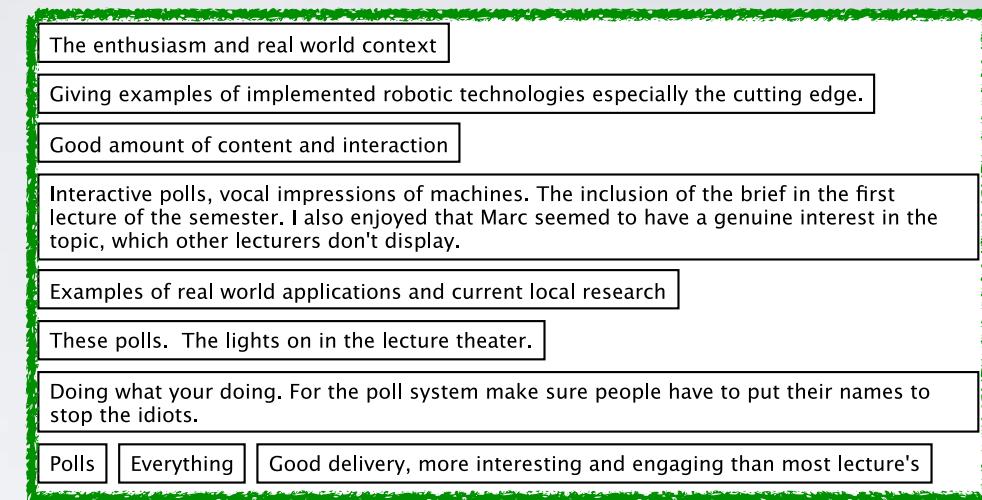
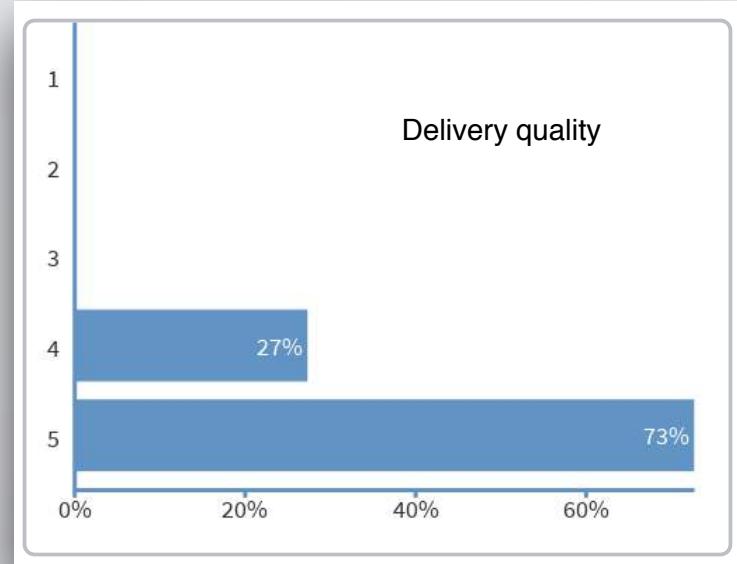
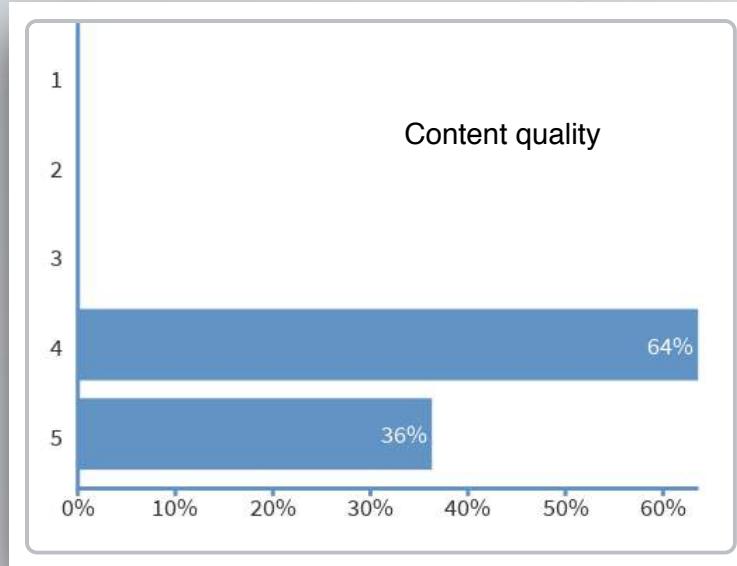
Prof. Marc Hanheide



UNIVERSITY OF
LINCOLN



LINCOLN
ROBOTICS



SYLLABUS

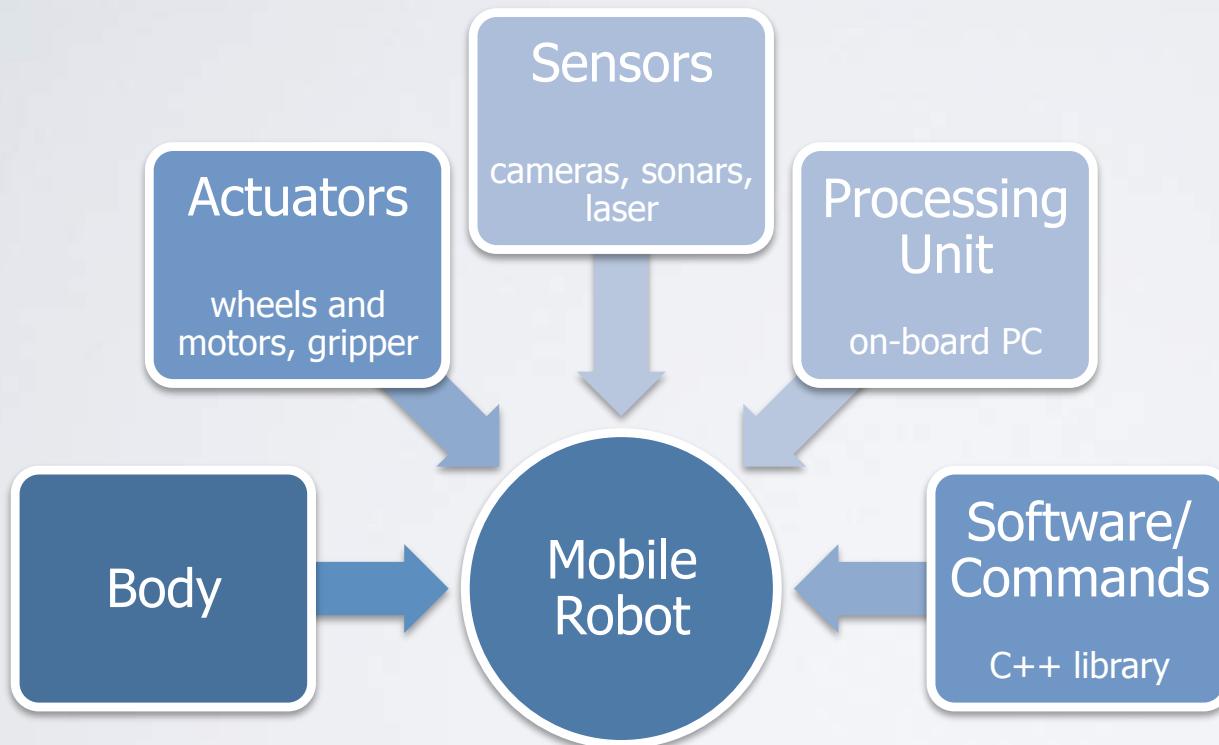
- ▶ Introduction to Robotics
- ▶ **Robot Programming**
- ▶ Robot Vision
- ▶ Robot Control
- ▶ Robot Behaviours
- ▶ Control Architectures
- ▶ Navigation Strategies
- ▶ Map Building

Disclaimer:
These slides are not self-contained, this is going to be an interactive lecture

Click on all exteroceptive sensors!

microphone
barometer
GPS
speaker
sonar
inertia-measuring-unit
radar camera
odometry
gyroscope
kinect
LIDAR
CPU-thermometer

MOBILE ROBOT COMPONENTS



PROCESSING UNIT

- ▶ On-board
 - ▶ fast responses
 - ▶ embodied, processing power limited by the physical size of a robot
- ▶ Off-board
 - ▶ remote computer(s) - significant processing power
 - ▶ problems with communication, data transfer and synchronisation
- ▶ **Hybrid architecture**
 - ▶ **on board for low-level tasks**
 - ▶ **PC for higher-level tasks**



ROBOT SOFTWARE

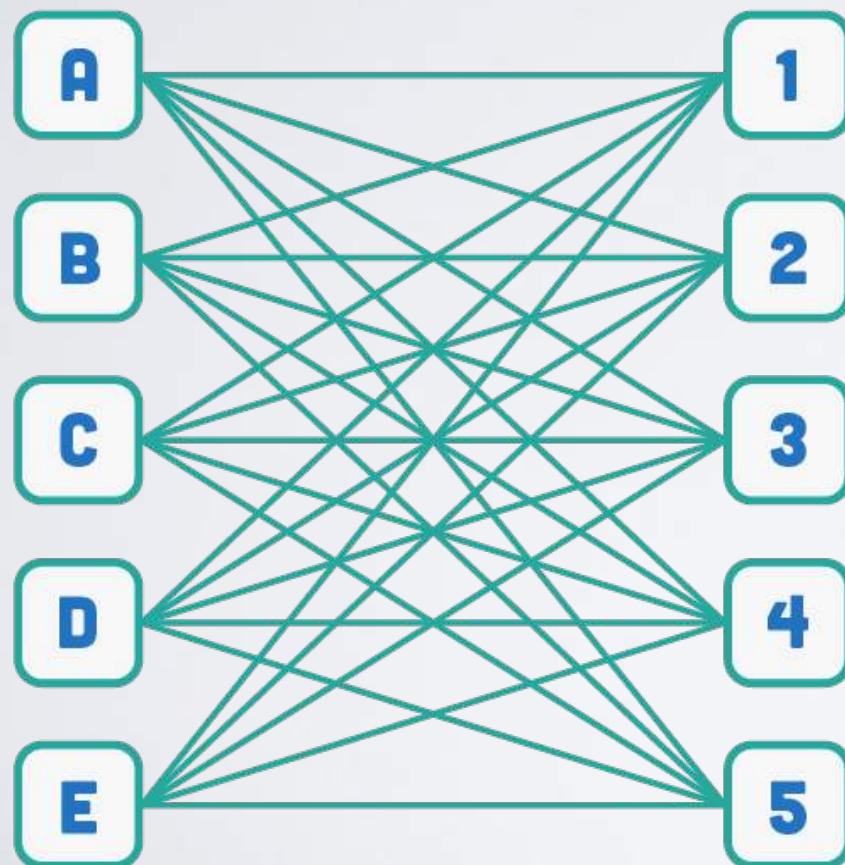
- ▶ Hardware **drivers**, (real-time) operating system
- ▶ Audio/video encoders
- ▶ Command interface, **firmware**
- ▶ Sensor/Image processing **library**
- ▶ Software **components** implementing AI, navigation, decision making
- ▶ **Simulator**

HOW TO TALK TO A ROBOT

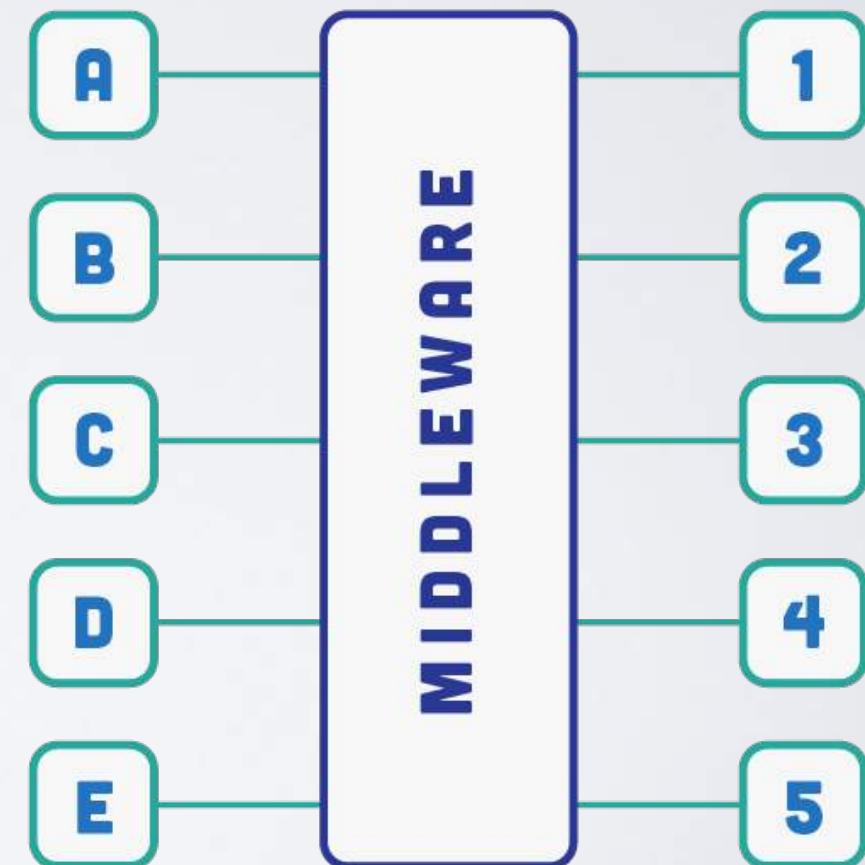
- ▶ We need to be able to:
 - ▶ send motor/actuator commands
 - ▶ read and process data from sensors
- ▶ in some robots there is an abstraction layer featuring a shared **domain-specific command language** to access all sensors and actuators (highly integrated)
 - ▶ implemented inside the robot
 - ▶ can be messages sent through serial port, Ethernet (Wifi)
- ▶ Our Turtlebots are more modular; different sensors talk via USB
 - ▶ they have their dedicated ROS driver nodes to talk to us
- ▶ HENCE, we need to be able to **communicate!**

MIDDLEWARE?

WITHOUT MIDDLEWARE

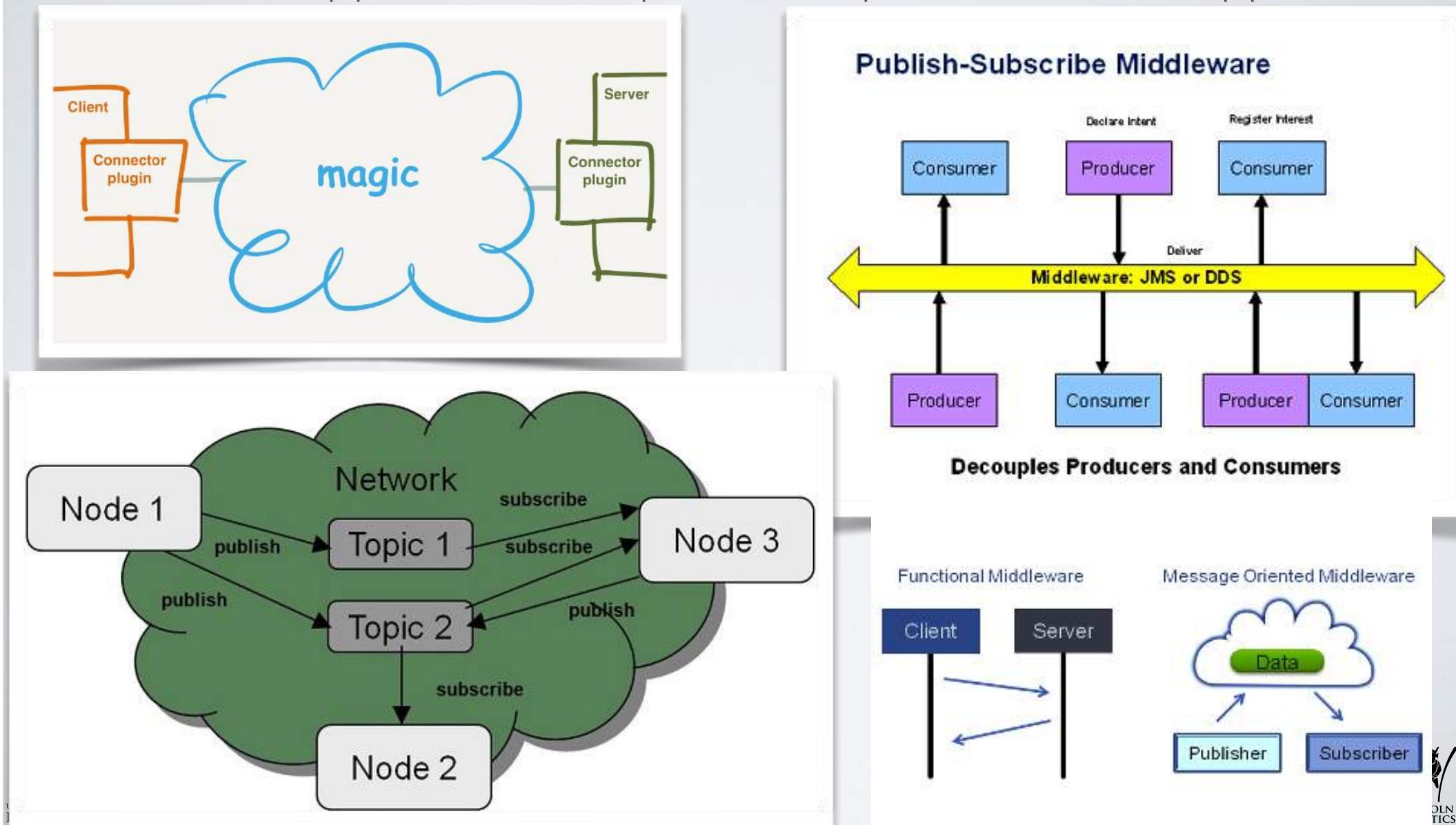


WITH MIDDLEWARE



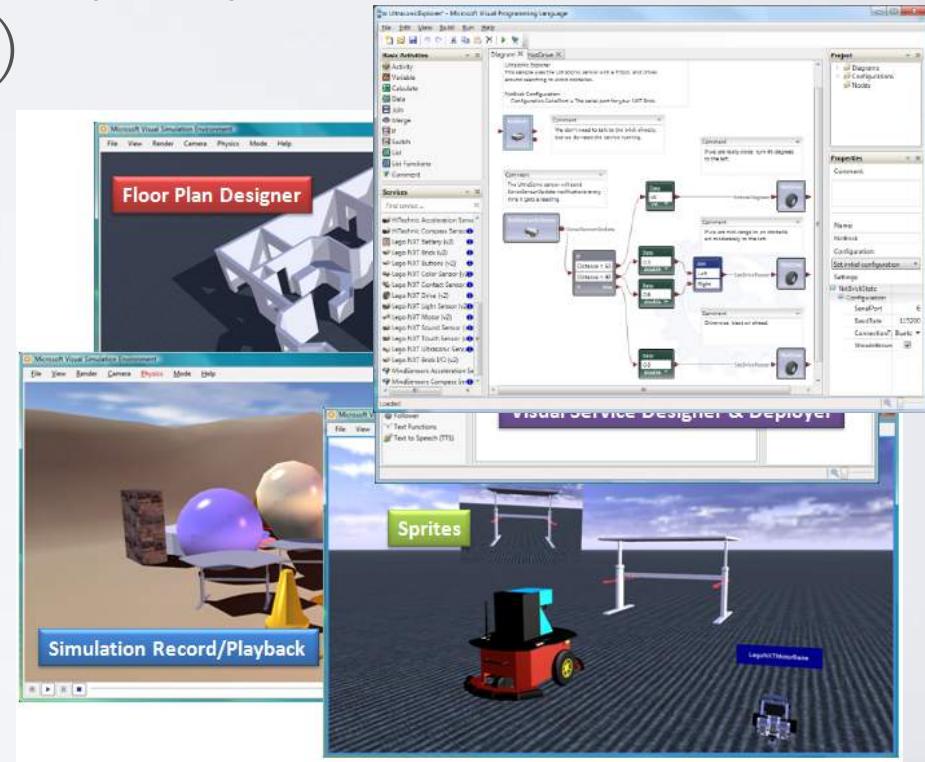
MIDDLEWARE?

Middleware supports and simplifies complex distributed applications.



ROBOTIC MIDDLEWARES

- ▶ Support for different robots, platforms, unified interface, plug-ins/modules (e.g. navigation, object recognition), simulator, etc.
 - ▶ **Robot Operating System (ROS)**
(<http://www.ros.org/wiki/rovio>)
 - ▶ **Microsoft Robotics Developer Studio**
for Windows
 - ▶ OROCOS
 - ▶ YARP
 - ▶ RSB
 - ▶ ...

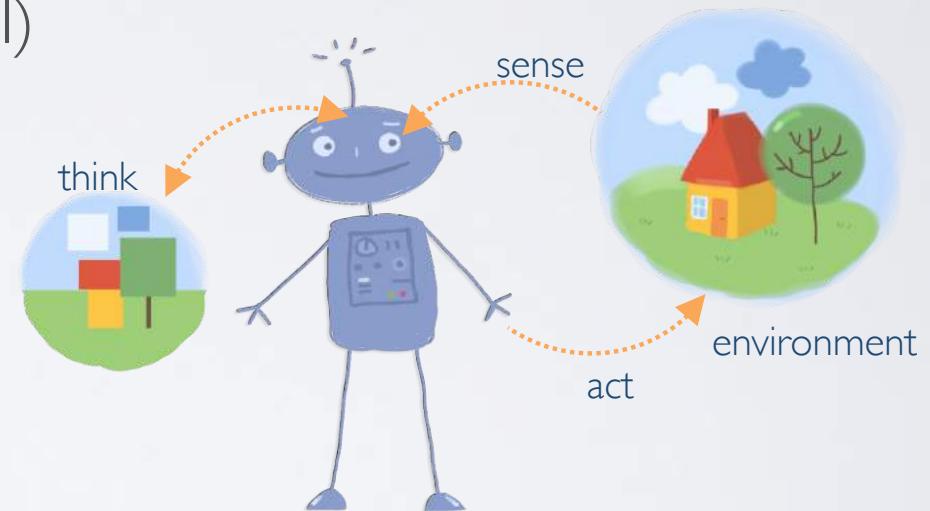


ROS

- ▶ ROS is a communication middleware with a huge library of state of the art algorithms being freely available
- ▶ ROS encapsulates functionality into individual nodes
- ▶ Nodes communicate via data streams
 - ▶ nodes implement callback (data push)
- ▶ C++, java, Python (and others more) supported

IMPLEMENTING TASKS/ BEHAVIOURS

- ▶ Scripts
 - ▶ A pre-programmed sequence of commands
- ▶ Continuous operation (robot control)
 - ▶ Sense
 - ▶ read sensor data
 - ▶ Think
 - ▶ process data and make decisions
 - ▶ Act
 - ▶ execute actions (send movement commands)



SENSE - (THINK) - ACT

- ▶ Two Options

data pull

- ▶ Synchronous:

- ▶ like a while loop:

```
while (true)
{
    robot.sense();
    robot.think();
    robot.act();
}
```

Easier

- ▶ Asynchronous:

data callbacks

- ▶ different threads with shared (and synchronised) memory access

basically how ROS works

AND NOW FOR SOME REAL
CODING IN ROS

STOLEN **ADOPTED** INTRO
ABOUT ROS

Introduction to ROS

Pierrick Koch, Séverin Lemaignan
based on slides by Thomas Moulard

LAAS robotics courses, January 2013

 ROS.org

Using ROS

Jeremiah Via

23 February 2011

How Robotics
Research Keeps...

Re-Inventing the Wheel

First, someone
Publishes...



...and they write
code that barely
works but lets
them publish...



...a paper with
a proof-of-
concept robot.



This prompts
another lab to
try to build on
this result...



But inevitably,
time runs out...



...but they can't
get any details
on the software
used to make it
work...



...and countless
sleepless nights
are spent
writing code
from scratch.



So, a grandiose
plan is formed
to write a new
software API...



...and all the
code used by
previous lab
members is a mess.

ROS.org

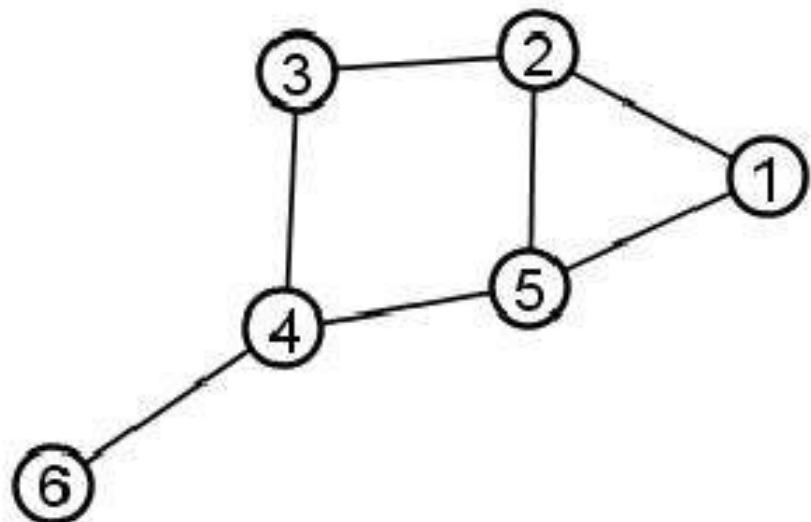
- Robot Operating System
- Meta-operating System
- Originally developed in 2007 at Stanford AI Lab
- Now developed at Willow Garage

So what is ROS?

- A component-oriented robotics framework,
- An Inter Process Communication middleware,
- A development suite,
- A (bad) package management system,
- An (active) community.

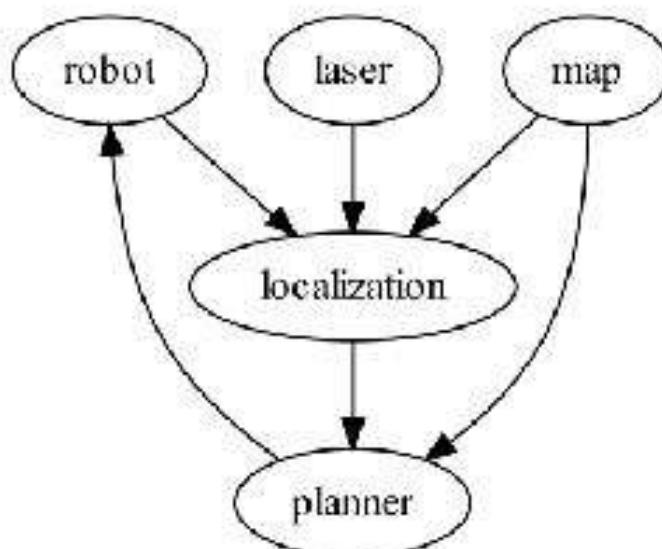
Node

- A process that performs computation



Peer-to-Peer

- Designed to have many nodes running
- Communication is managed by the master node



<http://www.ros.org/wiki/ROS/Tutorials/UnderstandingNodes>

The ROS framework is **component oriented**.

Each component is called a **node**. **Nodes** communicate using **topics** or **services**.

topics represents datastreams. For instance: camera images, robot joint configuration or position can be modelled as topics.

Topics values are typically published at regular rate to keep the whole system up-to-date.

services represents requests which are sent asynchronously, and usually at lower rate. Slower components such as a motion planning node will typically provide services.

Topic

- String which labels a stream of data, e.g., cmd_vel or scan
- Nodes can subscribe and publish to these topics



ROS

- ▶ roscore
- ▶ rosnode list
- ▶ rostopic list
- ▶ rosservice list
- ▶ rqt_graph
- ▶ (roslaunch ~/test.launch)

always use [Tab] and “-h”
when working on the
command line!

Order the processing steps when a message is to be send to a subscriber

Publisher provides socket descriptor where topic is streamed

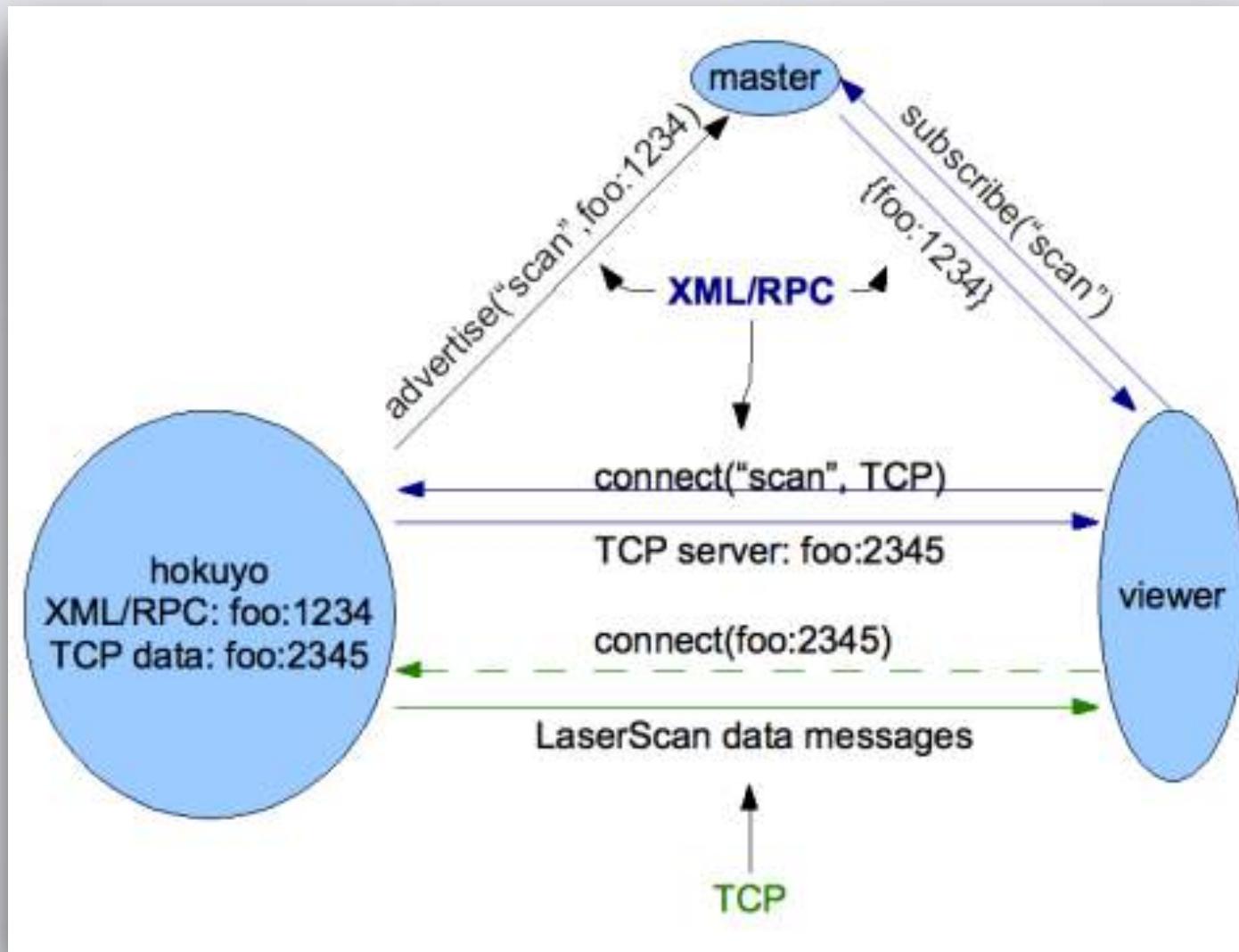
Subscriber connects to socket to receive data

Subscriber asks master for node(s) providing topics

Subscriber contacts Publisher (directly) with subscribe request

Publisher advertises topic to master

ROS PUBLISH SUBSCRIBE



Each **node** can **listen** or **publish** on a topic.

Messages types are defined using a dedicated syntax which is ROS specific:

MyMessage.msg

```
# this is a very useful comment!
float64 myDouble
string myString
float64[] myArrayOfDouble
```

DATA TYPES

- ▶ Let's have a look
 - ▶ useful tools:
 - ▶ rostopic pub
 - ▶ rostopic echo
 - ▶ rostopic info
 - ▶ rosmsg
 - ▶ rossrv
- message type define what is
being exchanged between
nodes

Each **node** can also **set** or **get** one or more parameters. Some are **public** and can be modified by other **nodes**. The other are **private** and are defined at startup only.

In the ROS documentation:

`publicParameter` is a public parameter,

`~privateParameter` is a private parameter.

Optionally, `dynamic_reconfigure` can be used to change the parameters in a GUI (or from command line) while the **node** is running. It can be used to change the camera grabbing framerate for instance.

PARAMS

- ▶ Let's have a look
- ▶ useful tools:
 - ▶ rosparam list
 - ▶ rosparam set
 - ▶ rosparam get

params are a little advanced
but important to configure
other components

Now that we have nodes with input and output, how do we make them communicate together?

If **A** publishes on *foo* and **B** listens on *foo*, **B** will receive topics data from **A**.

The services and topics names are used to match clients and servers.

A SIMPLE COMMUNICATION

- ▶ Let's implement a simple communication A->B
(without programming, just using **rostopic**)

ROS introspection tools (2)

```
# Show message type.  
rosmsg show geometry_msgs/Twist  
# Publish velocity  
rostopic pub -1 /robot/motion geometry_msgs/Twist \  
  "{linear: {x: 1.0}, angular: {z: 1.0}}"  
# See how often the robot pose is refreshed.  
rostopic hz /robot/odometry  
# Show one Odometry message  
rostopic echo -n1 /robot/odometry  
# Plot the pose.  
rxplot /robot/odometry/pose/pose/position/x,\  
       /robot/odometry/pose/pose/position/y \  
       /robot/odometry/pose/pose/orientation/z,\  
       /robot/odometry/pose/pose/orientation/w
```

Packaged Subsystems

- Some areas of research are mature enough to use standard algorithms
- No sense in reimplementing a new SLAM system for each new robot
- ROS allows multiple packages to be run together:
`roslaunch`

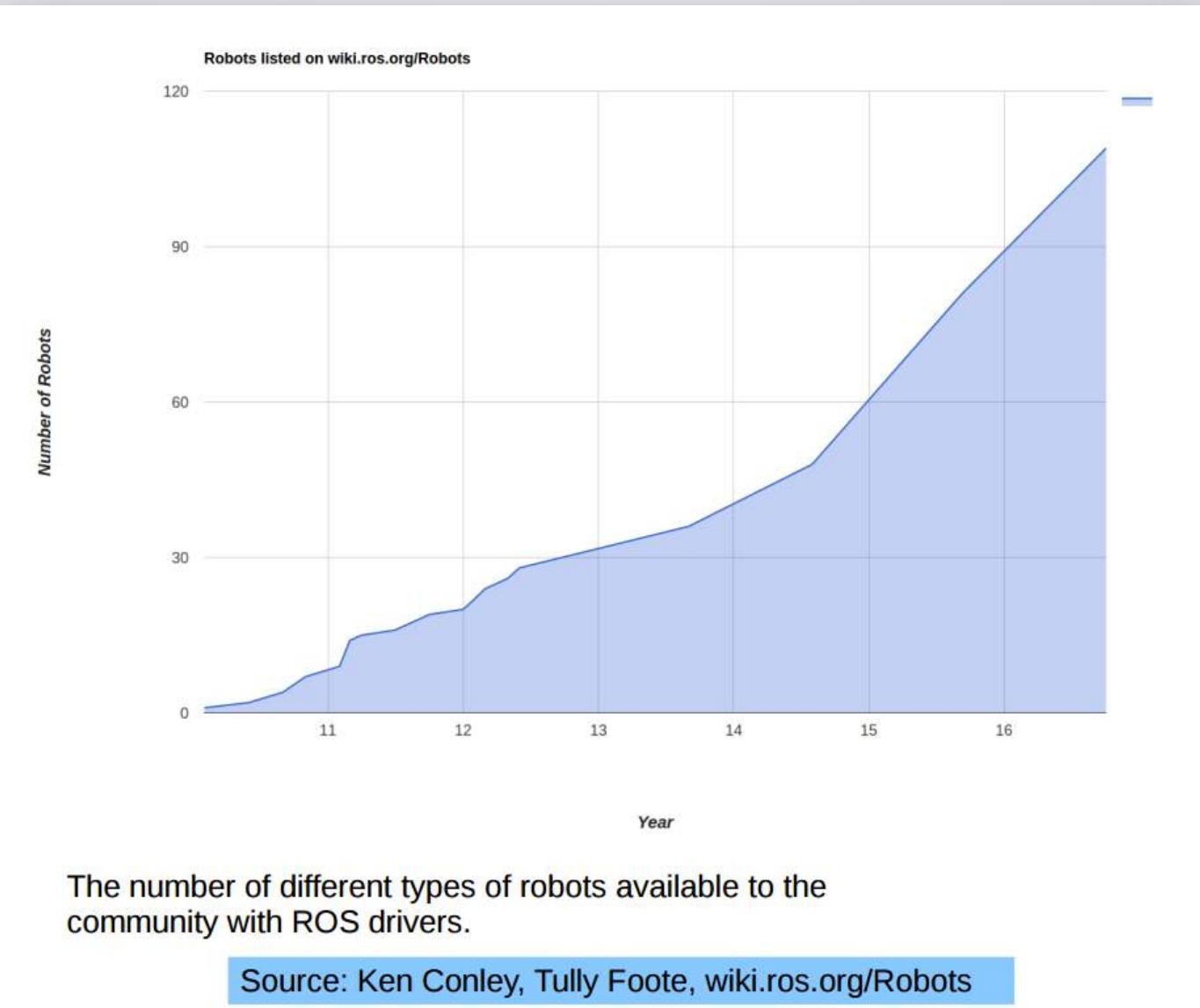
└ What is ROS?

 └ Architecture

Language Agnostic



ROS DOMINATION?



Put the steps to create a working ROS package and run it in the correct order

create a Python file in `~/catkin_ws/src/clever_robot`
and edit it (e.g. using "spyder")

`cd ~/catkin_ws`

run your Python code

`mkdir ~/catkin_ws`

`source ~/catkin_ws/devel/setup.bash`

`mkdir src`

`catkin_init_workspace .`

`catkin_make`

`catkin_create_pkg clever_robot`

TIME FOR SOME PYTHON

A FRESH START

- create your own **catkin** workspace:
 - mkdir catkin_ws
 - cd catkin_ws
 - mkdir src http://wiki.ros.org/catkin/Tutorials/create_a_workspace
 - cd src
 - catkin_init_workspace

A FRESH START

- create your own package:

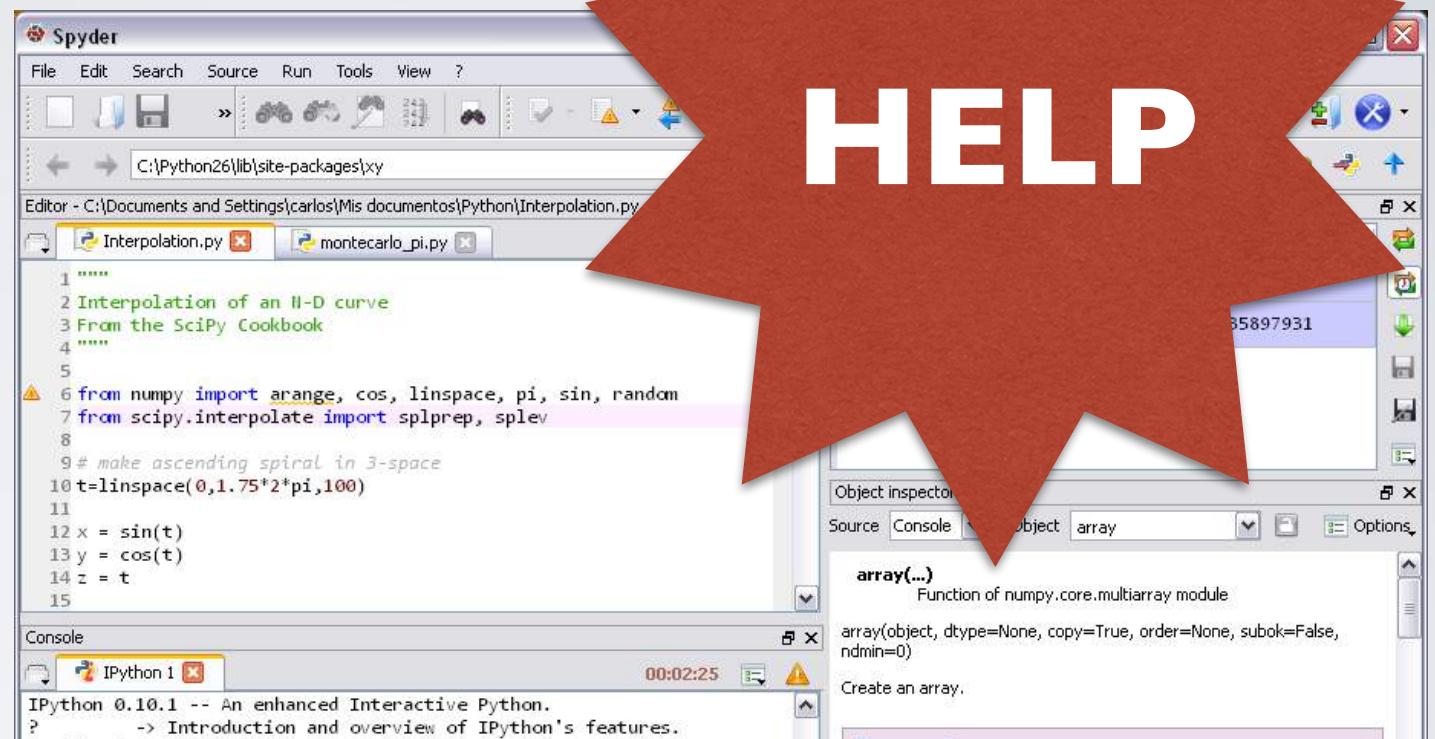
- `catkin_create_pkg [-h]`

- `cd ..`

<http://wiki.ros.org/ROS/Tutorials/CreatingPackage>

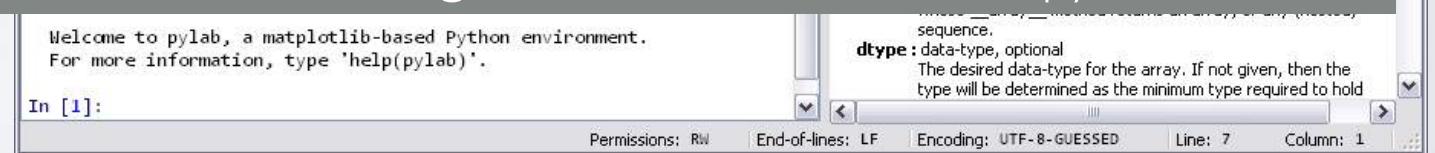
- `catkin_make`

- `source devel/setup.bash`



HELP

<http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>



COMPUTER VISION LIBRARIES

- ▶ OpenCV
 - ▶ ‘The’ Computer Vision Library – C++, Python, multiplatform
 - ▶ rich functionality, well tested, used in many systems, lots of examples

RECOMMENDED READING

- ▶ Biggs, G. and MacDonald, B. [A Survey of Robot Programming Systems](#), 2003
- ▶ Programming Robots with ROS (Morgan Quigley, Brian Gerkey & And William D. Smart.) (ebook on reading list)
- ▶ **The ROS tutorials!!!**
- ▶ Resources:
 - ▶ <http://answers.ros.org>
 - ▶ <http://wiki.ros.org/ROS/Tutorials>
 - ▶ <https://github.com/LCAS/teaching/wiki/CMP3641M>

THANK YOU FOR LISTENING!

End of Lecture Feedback

When survey is active, respond at **PollEv.com/mhanheide**

MH

0 surveys done

↻ 1 survey underway

Start the presentation to see live content. Still no live content? Install the app or get help at PollEv.com/app

CMP3103M AUTONOMOUS MOBILE ROBOTS

Prof Marc Hanheide



UNIVERSITY OF
LINCOLN



LINCOLN
ROBOTICS

SYLLABUS

- ▶ Introduction to Robotics
- ▶ Robot Programming
- ▶ **Robot Sensing & Vision**
- ▶ Robot Control
- ▶ Robot Behaviours
- ▶ Control Architectures
- ▶ Navigation Strategies
- ▶ Map Building

Let's make a robot following a line!

Which middleware paradigm do "ROS topics" implement?

- data pull
- client-server
- synchronous processing
- publish-subscribe
- broadcast

PROCESS FLOW IN ROS

```
import rospy
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Twist

class Receiver:

    def __init__(self):
        rospy.init_node('receiver')
        self.subscriber = rospy.Subscriber(
            '/scan', LaserScan, callback=self.cb)
        self.p = rospy.Publisher(
            '/mobile_base/commands/velocity', Twist, queue_size=1)

    def cb(self, incoming_data):
        # print len(incoming_data.ranges)
        if incoming_data.ranges[320] < 1.0:
            t = Twist()
            t.angular.z = 0.3
            self.p.publish(t)

rec = Receiver()

rospy.spin()
```

What is the order of processing events in this ROS code?

The robot turns

The robot's laser sensor completes a scan of the environment (with an obstacle close to the front of the robot) and publishes the data

The callback of the subscriber is called

A subscriber is created, subscribing to the topic '/scan' of type 'sensor_msgs/LaserScan'

The node is initialised with the name 'receiver'

A message is published through the publisher

A new Twist object is created

A publisher is created, announcing to publish data on '/mobile_base/commands/velocity' of type 'geometry_msgs/Twist'

PROCESS FLOW IN ROS

```
import rospy
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Twist

class Receiver:

    def __init__(self):
        rospy.init_node('receiver')
        self.subscriber = rospy.Subscriber(
            '/scan', LaserScan, callback=self.cb)
        self.p = rospy.Publisher(
            '/mobile_base/commands/velocity', Twist, queue_size=1)

    def cb(self, incoming_data):
        # print len(incoming_data.ranges)
        if incoming_data.ranges[320] < 1.0:
            t = Twist()
            t.angular.z = 0.3
            self.p.publish(t)

rec = Receiver()

rospy.spin()
```

What is wrong with this code?

```
import rospy
from std_msgs.msg import String
from sensor_msgs.msg import LaserScan

class FirstSub:
    def __init__(self):
        self.sub = rospy.Subscriber('/turtlebot_2/scan',
                                   Odometry,
                                   callback=self.callback)
        self.pub = rospy.Publisher('/warning', String)

    def callback(self, msg):
        for range in msg.ranges:
            if range < 1.0:
                print "ALERT"
                s = String()
                s.data = "ALERT"
                self.pub.publish(s)

rospy.init_node("firstsubscriber")
fp = FirstSub()
rospy.spin()
```



publisher has no topic to publish to

syntax error in for-loop

wrong type

wrong import statement

callback not correctly registered

publish called with wrong data

What is wrong with this code? (2)

```
import rospy
from std_msgs.msg import String
from sensor_msgs.msg import LaserScan

class FirstSub:
    def __init__(self):
        self.sub = rospy.Subscriber('/turtlebot_2/scan',
                                    LaserScan,
                                    callback=self.callback)
        self.pub = rospy.Publisher('/warning', String)

    def callback(self, msg):
        for range in msg.ranges:
            if range < 1.0:
                print "ALERT"
                s = String()
                s.data = "ALERT"
                self.pub.publish(s.data)

rospy.init_node("firstsubscriber")
fp = FirstSub()
rospy.spin()
```

publisher has no topic to publish to

syntax error in for-loop

wrong type

wrong import statement

callback not correctly registered

publish called with wrong data



```
import rospy
from std_msgs.msg import String
from sensor_msgs.msg import LaserScan

class FirstSub:
    def __init__(self):
        self.sub = rospy.Subscriber('/turtlebot_2/scan', LaserScan,
                                    callback=self.callback)
        self.pub = rospy.Publisher('/warning', String)

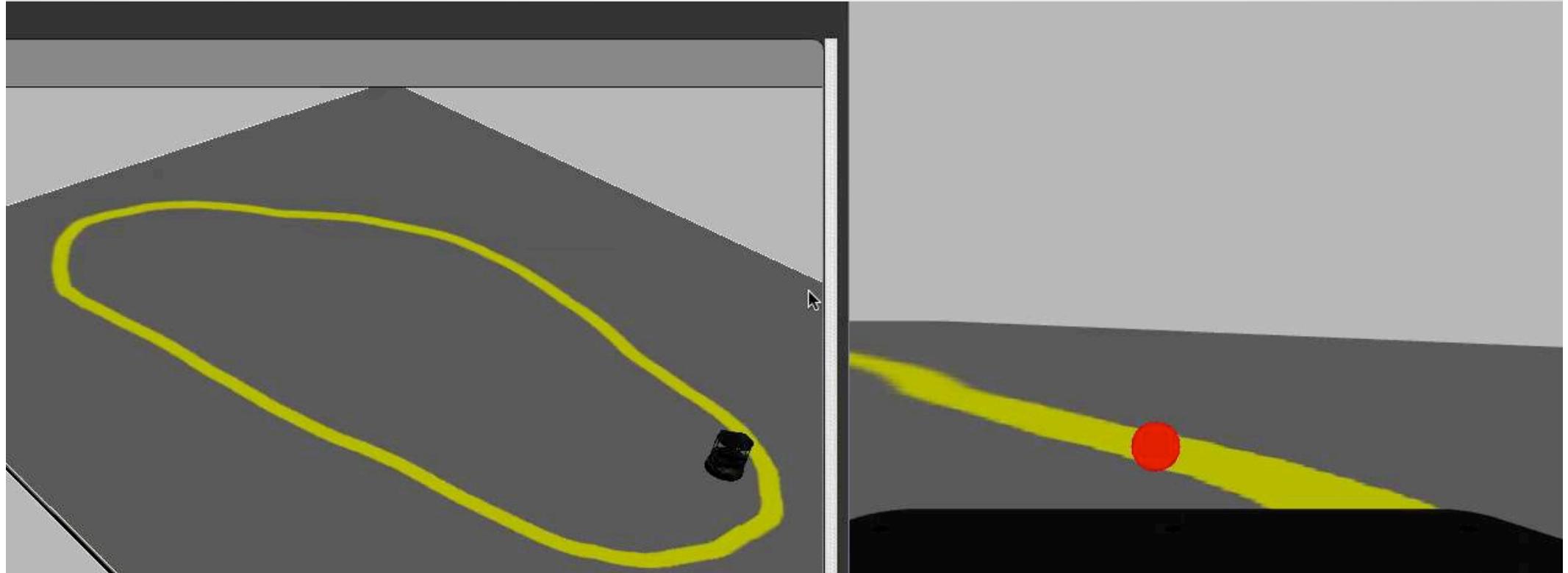
    def callback(self, msg):
        for range in msg.ranges:
            if range < 1.0:
                print "ALERT"
                s = String()
                s.data = "ALERT"
                self.pub.publish(s)

rospy.init_node("firstsubscriber")
fp = FirstSub()
rospy.spin()
```

VISION

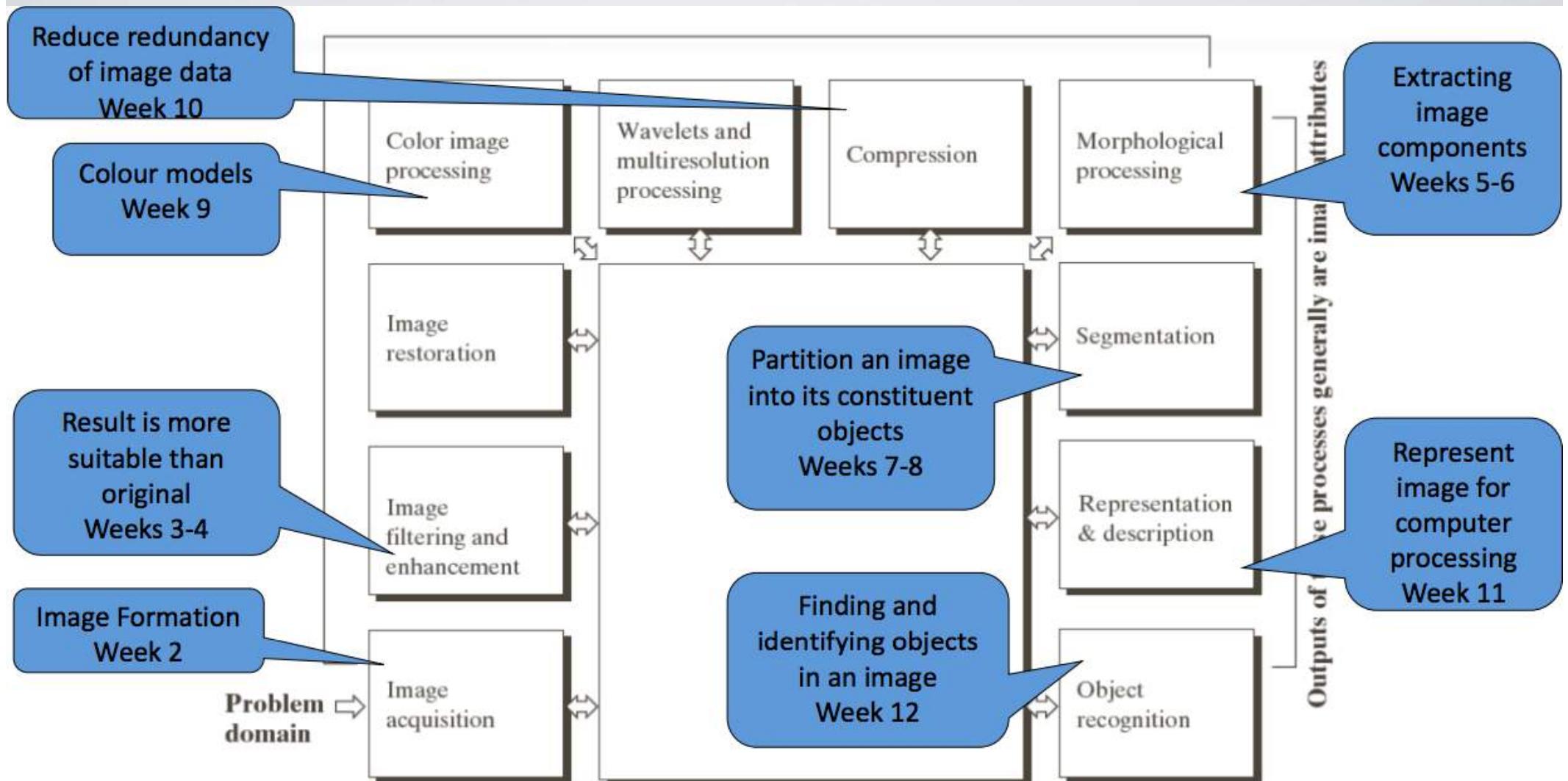
```
$> rostopic list
/camera/depth/camera_info
/camera/depth/image_raw
/camera/depth/points
/camera/parameter_descriptions
/camera/parameter_updates
/camera/rgb/camera_info
/camera/rgb/image_raw
/camera/rgb/image_raw/compressed
/camera/rgb/image_raw/compressed/parameter_descriptions
/camera/rgb/image_raw/compressed/parameter_updates
/camera/rgb/image_raw/compressedDepth
/camera/rgb/image_raw/compressedDepth/parameter_descriptions
/camera/rgb/image_raw/compressedDepth/parameter_updates
/camera/rgb/image_raw/theora
/camera/rgb/image_raw/theora/parameter_descriptions
/camera/rgb/image_raw/theora/parameter_updates
```

AIM FOR TODAY



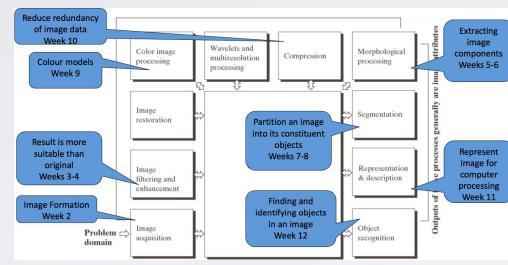
Morgan Quigley, Brian Gerkey & And William D. Smart. (2015)
Programming Robots with ROS [Chapter 12]

ROBOT VISION



ROBOT VISION

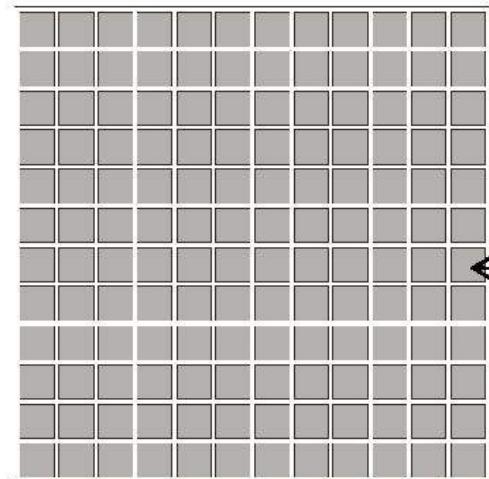
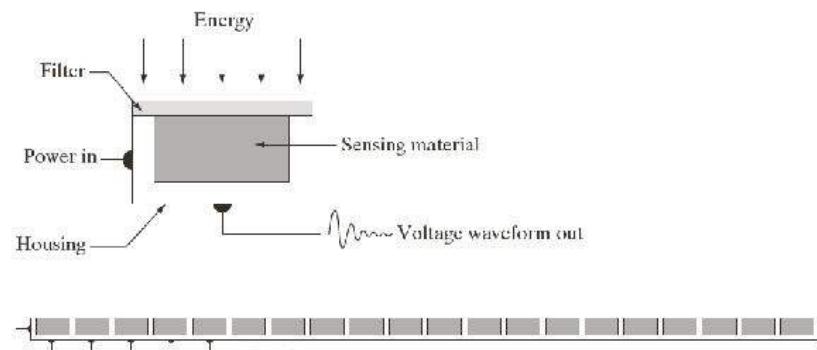
- ▶ Steps
- ▶ acquisition
- ▶ pre-processing
- ▶ segmentation
- ▶ feature extraction
- ▶ pattern recognition
- ▶ **robot control**



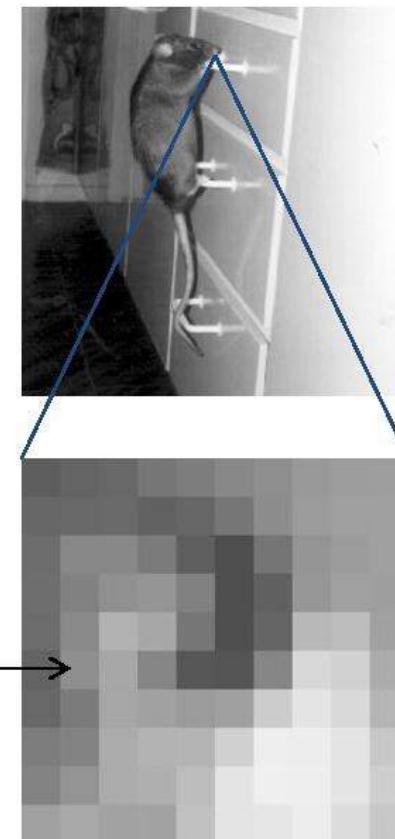
- ▶ Requirements for algorithms
- ▶ Fast (real-time)
- ▶ Robust (to noise and changes in environment)
- ▶ Non-stationary assumption (limited use of background subtraction methods)

Vision Sensor

Vision Sensor



Digital Image



a rat

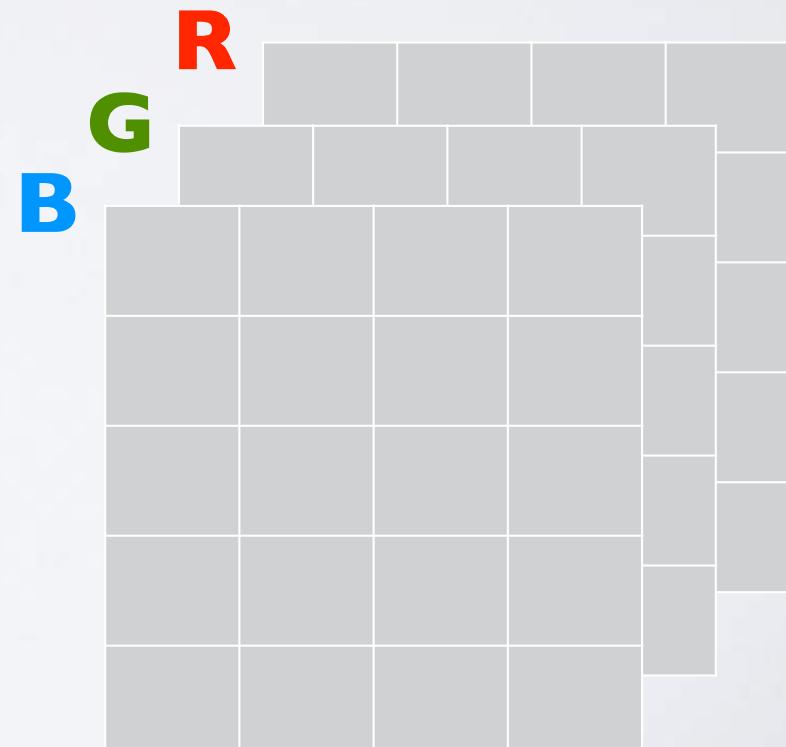
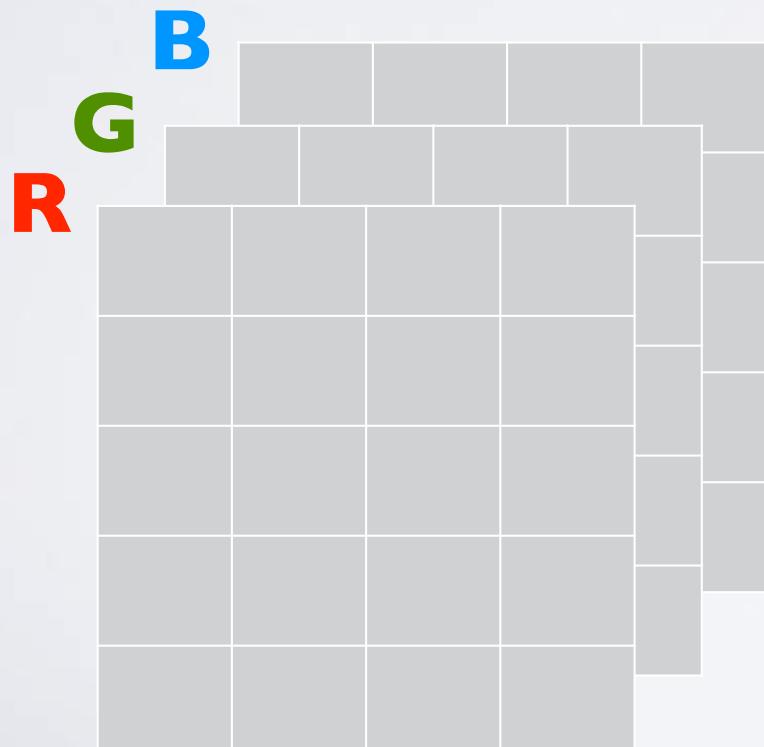
*the rat's
nose*

OPENCV

- ▶ Initially launched by Intel in 1999
- ▶ developed into the “gold standard” in computer vision processing
- ▶ cross-platform, multi-language
- ▶ Applications:
 - ▶ 2D and 3D feature toolkits
 - ▶ Egomotion estimation
 - ▶ Facial recognition system
 - ▶ Gesture recognition
 - ▶ Human–computer interaction (HCI)
- ▶ **Mobile robotics**
 - ▶ Motion understanding
 - ▶ Object identification
- ▶ **Segmentation and recognition**
 - ▶ Stereopsis stereo vision: depth perception from 2 cameras
 - ▶ Structure from motion (SFM)
- ▶ **Motion tracking**
 - ▶ Augmented reality

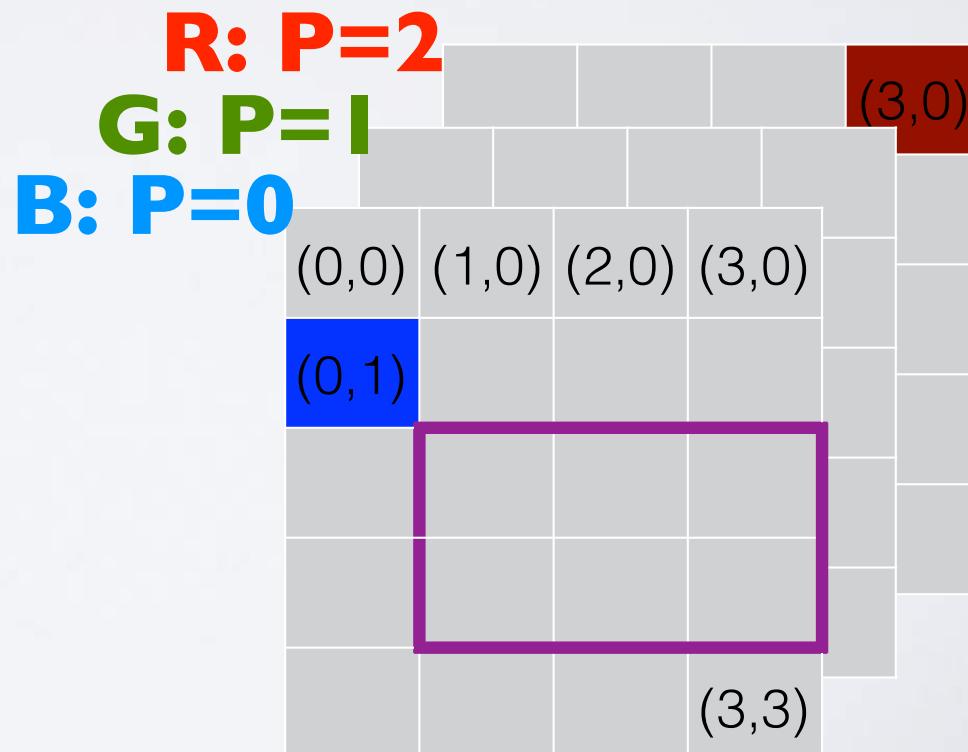
REPRESENTATION OF IMAGES IN OPENCV

- ▶ Matrices like in Matlab!
 - ▶ based on numpy
- ▶ not RGB, but BGR!?



BASIC OPERATIONS IN OPENCV

- ▶ Pixel access:
 - ▶ $\text{img}[X, Y, P]$
 - ▶ $\text{img}[3, 0, 2]$
 - ▶ $\text{img}[0, 1, 0]$
- ▶ Ranges:
 - ▶ $\text{img}[1:3, 2:3, 0]$



```

from cv2 import namedWindow, imread, imshow
from cv2 import waitKey, destroyAllWindows, startWindowThread
from cv2 import blur, Canny, circle

# declare windows you want to display
namedWindow("original")
namedWindow("blur")
namedWindow("canny")

# this is always needed to run the GUI thread
startWindowThread()

img = imread('../blofeld.jpg')
imshow("original", img)
# create a new blurred image:
img2 = blur(img, (7, 7))
# draw on the image:
circle(img2, (100, 100), 30, (255, 0, 255), 5)
# display the image:
imshow("blur", img2)
# Canny is an algorithm for edge detection
img3 = Canny(img, 10, 200)
imshow("canny", img3)
# the shape gives you the dimensions
h = img3.shape[0]
w = img3.shape[1]
# loop over the image, pixel by pixel
count = 0
# a slow way to iterate over the pixels
for y in range(0, h):
    for x in range(0, w):
        # threshold the pixel
        if img3[y, x] > 0:
            count += 1
print('count edge pixels: %d' % count)
# wait key is also always needed to sync the GUI threads
waitKey(0)
# good practice to tidy up at the end
destroyAllWindows()

```

A SIMPLE OPENCV EXAMPLE (OPENCV_INTRO.PY)

- iterate over pixels
- manipulation pixels
- read and show images

DISPLAYING AND DRAWING IN OPENCV

```
import numpy as np
import cv2

# Create a black image
img = np.zeros((512,512,3), np.uint8)

# Draw a diagonal blue line with thickness of 5 px
img = cv2.line(img,(0,0),(511,511),(255,0,0),5)
```

```
img = cv2.circle(img,(447,63), 63, (0,0,255), -1)
```

GETTING IMAGE STREAMS FROM A ROS TOPIC

```
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
```

- ▶ OpenCV and ROS play nicely
- ▶ There is a dedicated CvBridge to help you getting and processing image from the robot

http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

- subscribe on Image topic
- convert to OpenCV in callback
- process the image using OpenCV

```
def __init__(self):  
  
    self.bridge = CvBridge()  
    self.image_sub = rospy.Subscriber("/camera/image_raw",  
                                      Image, self.callback)  
    # self.image_sub = rospy.Subscriber(  
    #     "/camera/rgb/image_raw",  
    #     Image, self.callback)  
  
def callback(self, data):  
    namedWindow("Image window")  
    namedWindow("blur")  
    namedWindow("canny")  
    cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
```

```
import rospy
from cv2 import namedWindow, cvtColor, imshow
from cv2 import destroyAllWindows, startWindowThread
from cv2 import COLOR_BGR2GRAY, waitKey
from cv2 import blur, Canny
from numpy import mean
from sensor_msgs.msg import Image
from cv_bridge import CvBridge

class image_converter:

    def __init__(self):
        self.bridge = CvBridge()
        self.image_sub = rospy.Subscriber("/camera/image_raw",
                                         Image, self.callback)
        # self.image_sub = rospy.Subscriber(
        #     "/camera/rgb/image_raw",
        #     Image, self.callback)

    def callback(self, data):
        namedWindow("Image window")
        namedWindow("blur")
        namedWindow("canny")
        cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")

        gray_img = cvtColor(cv_image, COLOR_BGR2GRAY)
        print mean(gray_img)
        img2 = blur(gray_img, (3, 3))
        imshow("blur", img2)
        img3 = Canny(gray_img, 10, 200)
        imshow("canny", img3)

        imshow("Image window", cv_image)
        waitKey(1)

    startWindowThread()
    rospy.init_node('image_converter')
    ic = image_converter()
    rospy.spin()

destroyAllWindows()
```

opencv_bridge.py

FIRST STEPS TOWARDS LINE FOLLOWING

```
#!/usr/bin/env python

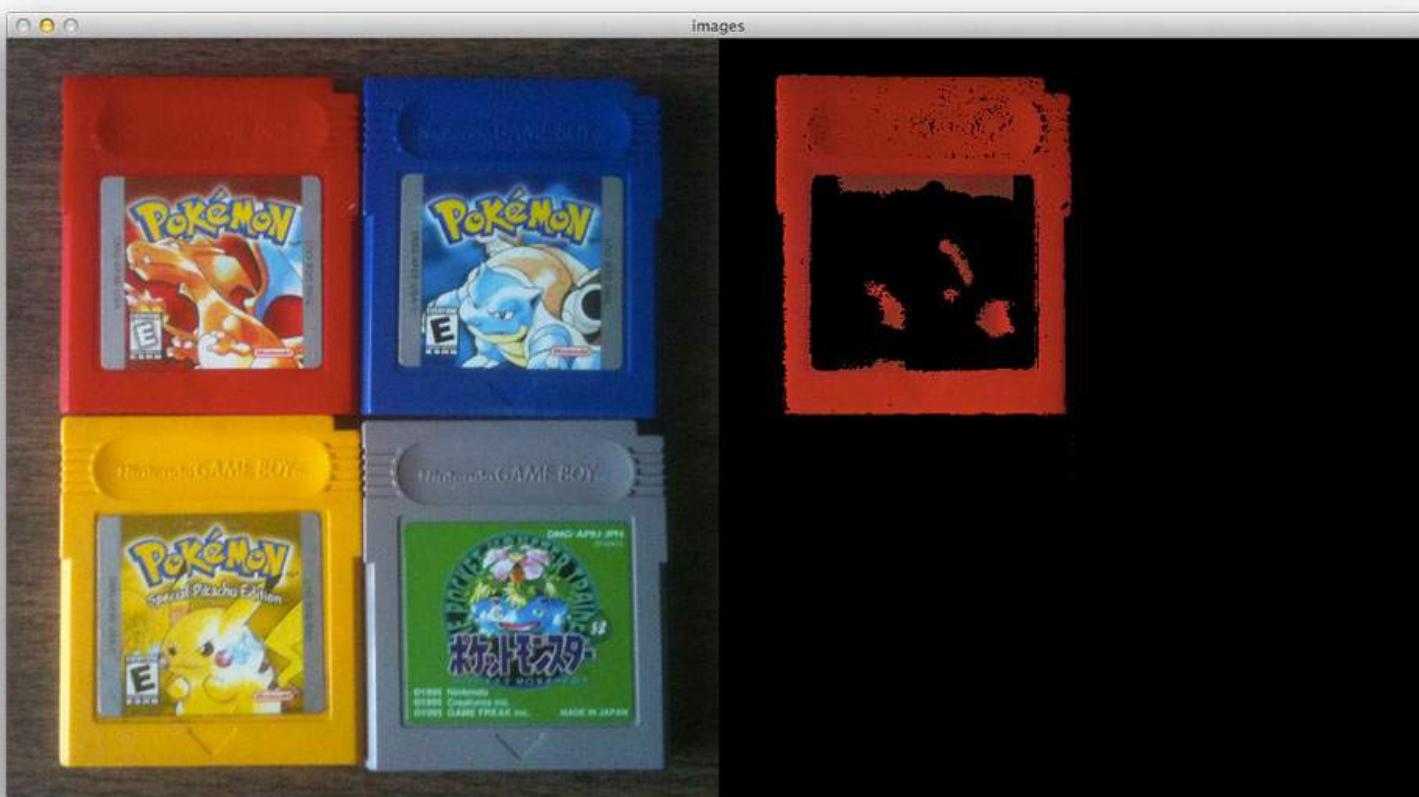
import rospy

from sensor_msgs.msg import Image

def image_callback(msg):
    pass

rospy.init_node('follower')
image_sub = rospy.Subscriber('camera/rgb/image_raw', Image,
image_callback)
rospy.spin()
```

A FIRST EXERCISE: COLOUR SLICING



<http://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection>

Colour Models

Common models and their applications

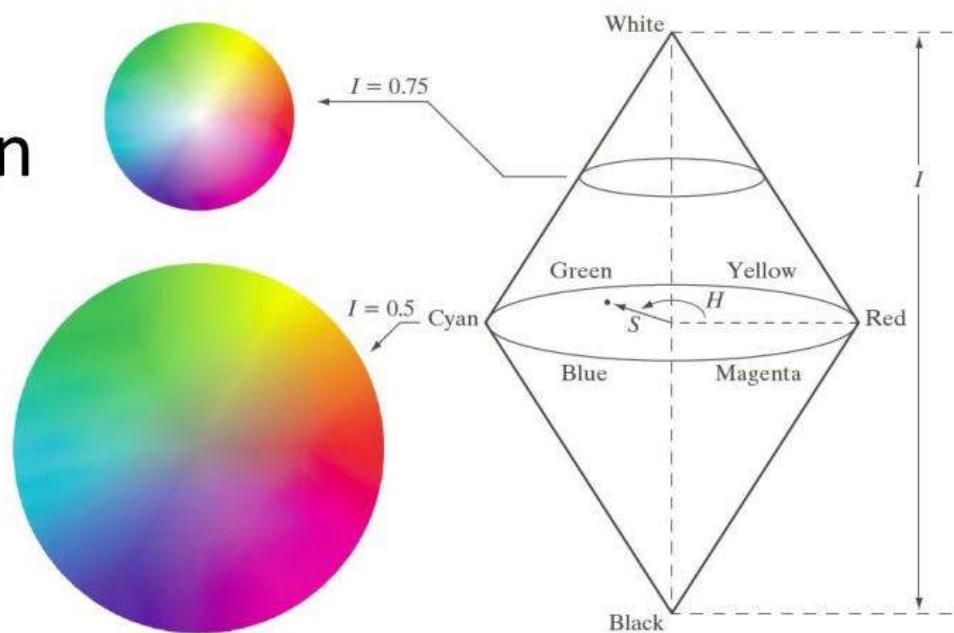
- RGB (red, green, blue)
 - colour monitors
- CMY, CMYK (cyan, magenta, yellow, black)
 - colour printers
- HSI (hue, saturation, intensity)
 - closely related to human perception of colour
 - decouples hue and intensity - very useful for real world applications where the overall intensity is often changing

Different colour image processing operations are easier or more difficult in different colour spaces

HSI Model

Hue, Saturation, Intensity

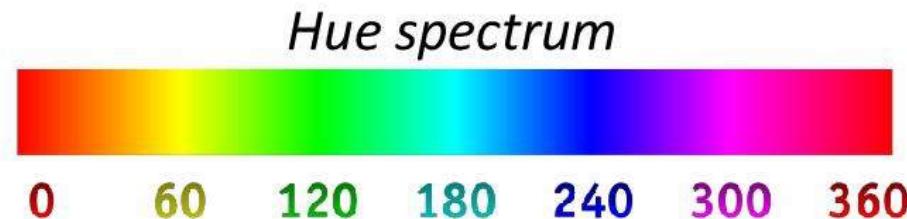
- separates intensity and hue
- resembles human vision
- difficult to display directly (transformation to RGB necessary)
- ! singularities (hue is undefined if the saturation is zero)



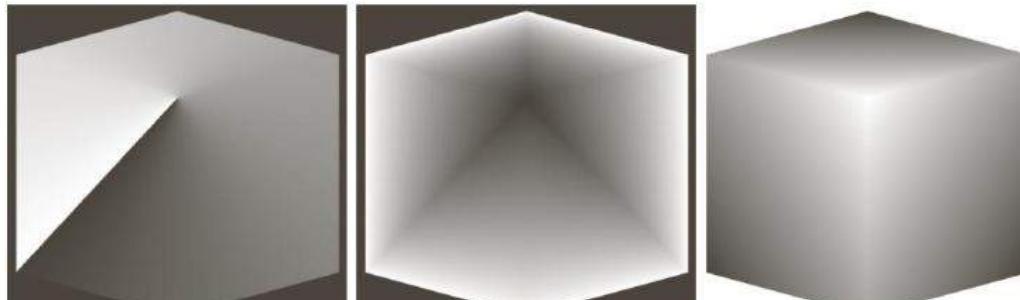
HSI Model

Hue component of HSI is expressed as an angle

- $0^\circ/360^\circ$ - Red
- 120° - Green
- 240° - Blue

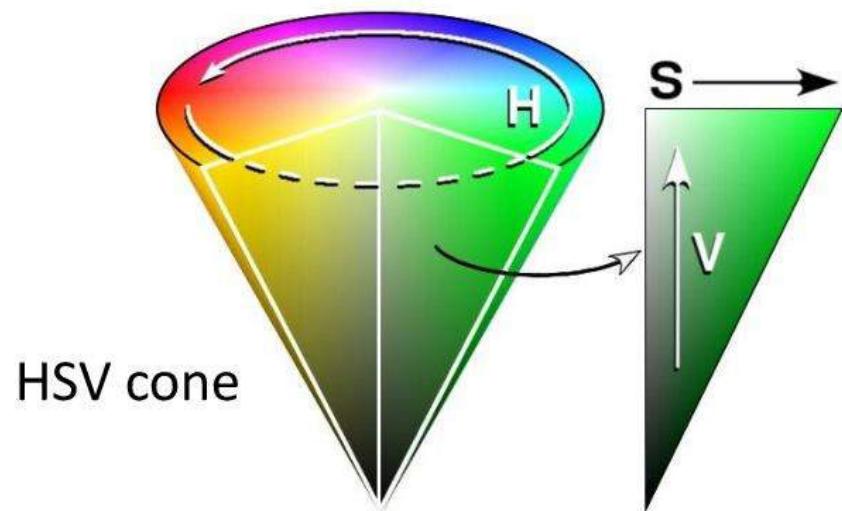


! take care to check for the discontinuity $0^\circ/360^\circ$ around “red” when processing HSI images!



a b c

FIGURE 6.15 HSI components of the image in Fig. 6.8. (a) Hue, (b) saturation, and (c) intensity images.



Colour Models Example



Full color



Red

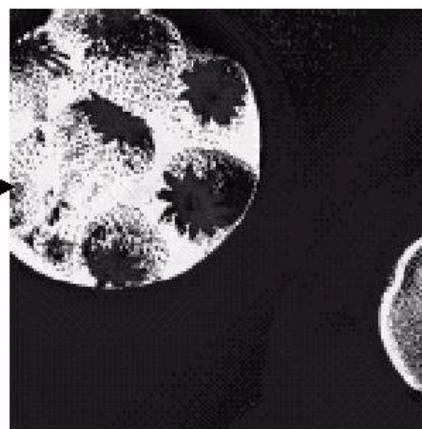


Green



Blue

Note the discontinuity (white/black) for red strawberries



Hue



Saturation



Intensity

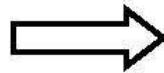
Colour Slicing

To highlight a specific range of colours

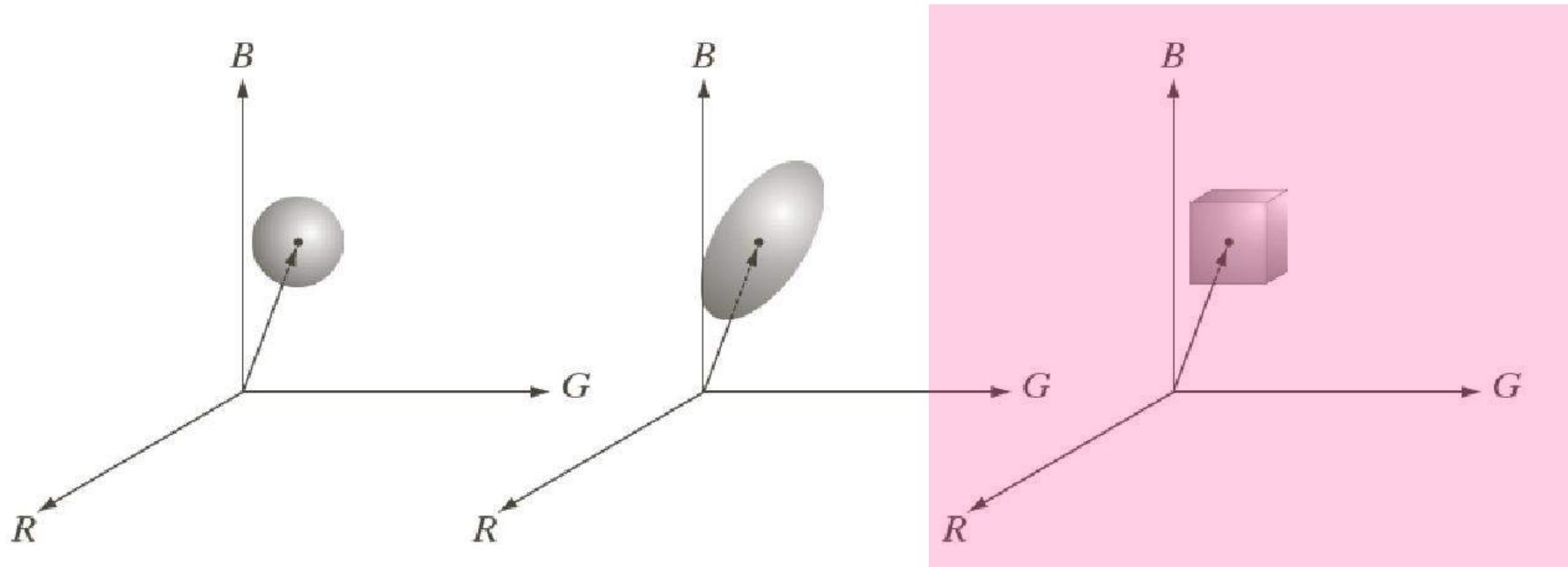
To define a mask for further processing

How to define the range of interest?

- cube/sphere/etc in colour space (centred at a prototypical colour)



Colour Slicing



Different ways to define the range of interest in RGB colour space

COLOUR SLICING IN PYTHON



```
bgr_thresh = cv2.inRange(cv_image,  
                           numpy.array((200, 230, 230)),  
                           numpy.array((255, 255, 255)))
```

- subscribe on Image topic
- convert to OpenCV in callback
- optional: convert to different colour space (from BGR)
- use `inRange` for colour slicing
- display binary image
- optional: post-process image (morphological ops)

COLOUR SLICING IN PYTHON



```
bgr_thresh = cv2.inRange(cv_image,  
                           numpy.array((200, 230, 230)),  
                           numpy.array((255, 255, 255)))
```

- ▶ What colour space to slice in?
- ▶ What values to choose?
- ▶ grab frames:
`rosrun image_view image_saver image:=/camera/rgb/image_color`
- ▶ use “gimp” and its color picker
- ▶ use <http://imagecolorpicker.com/> and upload an image

FIRST STEPS TOWARDS LINE FOLLOWING (FOLLOWER_COLOR_FILTER.PY)

```
import numpy
import cv2
import cv_bridge
import rospy
from sensor_msgs.msg import Image

class Follower:
    def __init__(self):
        self.bridge = cv_bridge.CvBridge()
        self.image_sub = rospy.Subscriber(
            'camera/rgb/image_raw', Image,
            self.image_callback)

    def image_callback(self, msg):
        cv2.namedWindow("window", 1)
        image = self.bridge.imgmsg_to_cv2(msg)
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        lower_yellow = numpy.array([10, 60, 170])
        upper_yellow = numpy.array([255, 255, 255])
        mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
        cv2.bitwise_and(image, image, mask=mask)
        cv2.imshow("window", mask)
        cv2.waitKey(1)

cv2.startWindowThread()
rospy.init_node('follower')
follower = Follower()
rospy.spin()
cv2.destroyAllWindows()
```

A SECOND EXERCISE: FINDING CONTOURS

- ▶ From colour slicing to regions!
- ▶ Work through http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html#contours-getting-started
- ▶ **see `color_contours.py`**

A THIRD EXTRA EXERCISE: FINDING CIRCLES

- ▶ Homework: Hough Transform
- ▶ What is it?





LiveSlides web content

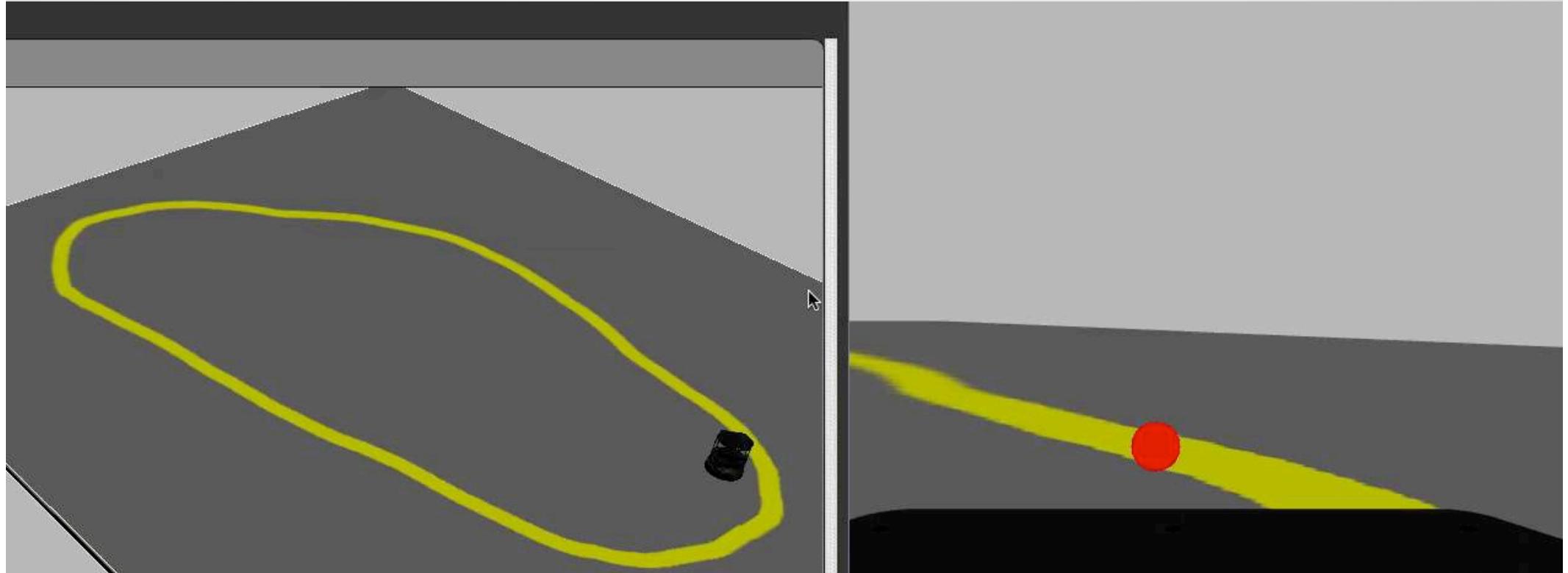
To view

Download the add-in.

liveslides.com/download

Start the presentation.

NOW FOR THE REAL STUFF



Morgan Quigley, Brian Gerkey & And William D. Smart. (2015)
Programming Robots with ROS [Chapter 12]

RESOURCES

- ▶ https://github.com/marc-hanheide/ros_book_sample_code/tree/master/chapter12
 - ▶ `catkin_init_workspace`
 - ▶ `git clone https://github.com/marc-hanheide/ros_book_sample_code.git`
 - ▶ `catkin_make (ignore errors)`
- ▶ Morgan Quigley, Brian Gerkey & And William D. Smart.
(2015) Programming Robots with ROS [Chapter 12]

FIRST STEPS TOWARDS LINE FOLLOWING (FOLLOWER_COLOR_FILTER.PY)

```
import numpy
import cv2
import cv_bridge
import rospy
from sensor_msgs.msg import Image

class Follower:
    def __init__(self):
        self.bridge = cv_bridge.CvBridge()
        self.image_sub = rospy.Subscriber(
            'camera/rgb/image_raw', Image,
            self.image_callback)

    def image_callback(self, msg):
        cv2.namedWindow("window", 1)
        image = self.bridge.imgmsg_to_cv2(msg)
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        lower_yellow = numpy.array([10, 60, 170])
        upper_yellow = numpy.array([255, 255, 255])
        mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
        cv2.bitwise_and(image, image, mask=mask)
        cv2.imshow("window", mask)
        cv2.waitKey(1)

cv2.startWindowThread()
rospy.init_node('follower')
follower = Follower()
rospy.spin()
cv2.destroyAllWindows()
```

FIRST STEPS TOWARDS LINE FOLLOWING (FOLLOWER_LINE_FINDER.PY)

[...]

```
def image_callback(self, msg):
    cv2.namedWindow("window", 1)
    image = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_yellow = numpy.array([10, 60, 170])
    upper_yellow = numpy.array([255, 255, 255])
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    h, w, d = image.shape
    search_top = 3*h/4
    search_bot = search_top + 20
    mask[0:search_top, 0:w] = 0
    mask[search_bot:h, 0:w] = 0
    M = cv2.moments(mask)
    if M['m00'] > 0:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        cv2.circle(image, (cx, cy), 20, (0, 0, 255), -1)
    cv2.imshow("window", image)
    cv2.waitKey(3)
```

[...]



LiveSlides web content

To view

Download the add-in.

liveslides.com/download

Start the presentation.

CLOSING THE LOOP (FOLLOWER_PPY)

[...]

```
def image_callback(self, msg):
    cv2.namedWindow("window", 1)
    image = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_yellow = numpy.array([10, 10, 10])
    upper_yellow = numpy.array([255, 255, 250])
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    h, w, d = image.shape
    search_top = 3*h/4
    search_bot = 3*h/4 + 20
    mask[0:search_top, 0:w] = 0
    mask[search_bot:h, 0:w] = 0
    M = cv2.moments(mask)
    if M['m00'] > 0:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        cv2.circle(image, (cx, cy), 20, (0, 0, 255), -1)
        err = cx - w/2
        self.twist.linear.x = 0.2
        self.twist.angular.z = -float(err) / 100
        self.cmd_vel_pub.publish(self.twist)
    cv2.imshow("window", image)
    cv2.waitKey(3)
```

[...]



LiveSlides web content

To view

Download the add-in.

liveslides.com/download

Start the presentation.

SOME MORE ROBOT-
RELATED VISION TO DISCUSS

APPLICATIONS

- ▶ Visual Path Following, Lagadic project, INRIA, France

Visual path following
using only monocular vision
for urban environments

- Lagadic project -

INRIA Rennes - IRISA

FEATURES

- ▶ Features
 - ▶ compact, meaningful representation of the image
- ▶ What are the good features?
 - ▶ edges, corners, blobs, colour, texture
- ▶ State of the art
 - ▶ SIFT, SURF, Haar-like, HOG, etc.
All in openCV!
- ▶ Applications - examples
 - ▶ Object detection: template matching
 - ▶ Scene reconstruction: extract depth information



VISION IN ROBOTICS

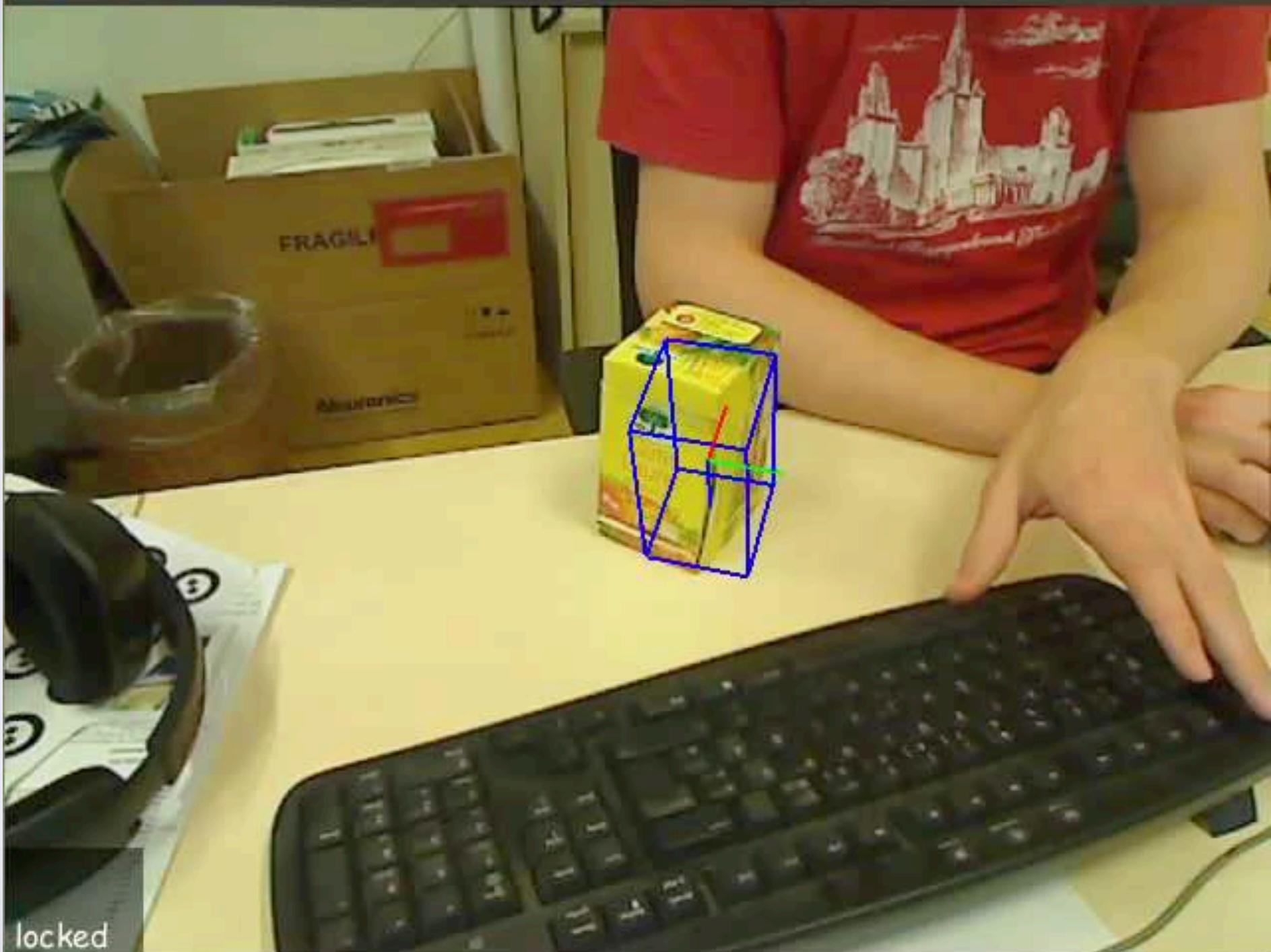
SIGGRAPH Talks 2011 **KinectFusion:** **Real-Time Dynamic 3D Surface Reconstruction and Interaction**

**Shahram Izadi 1, Richard Newcombe 2, David Kim 1,3, Otmar Hilliges 1,
David Molyneaux 1,4, Pushmeet Kohli 1, Jamie Shotton 1,
Steve Hodges 1, Dustin Freeman 5, Andrew Davison 2, Andrew Fitzgibbon 1**

1 Microsoft Research Cambridge 2 Imperial College London
3 Newcastle University 4 Lancaster University
5 University of Toronto



Tracker



VISION IN ROBOTICS



RECOMMENDED READING

- ▶ Siegwart, R. and Nourbakhsh, I.R.: Introduction to autonomous mobile robots MIT Press, 2004
- ▶ Section 4.3 – Feature Extraction
- ▶ Roth, P.M. And Winter, M., Survey of Appearance-based Method for Object Recognition, 2008

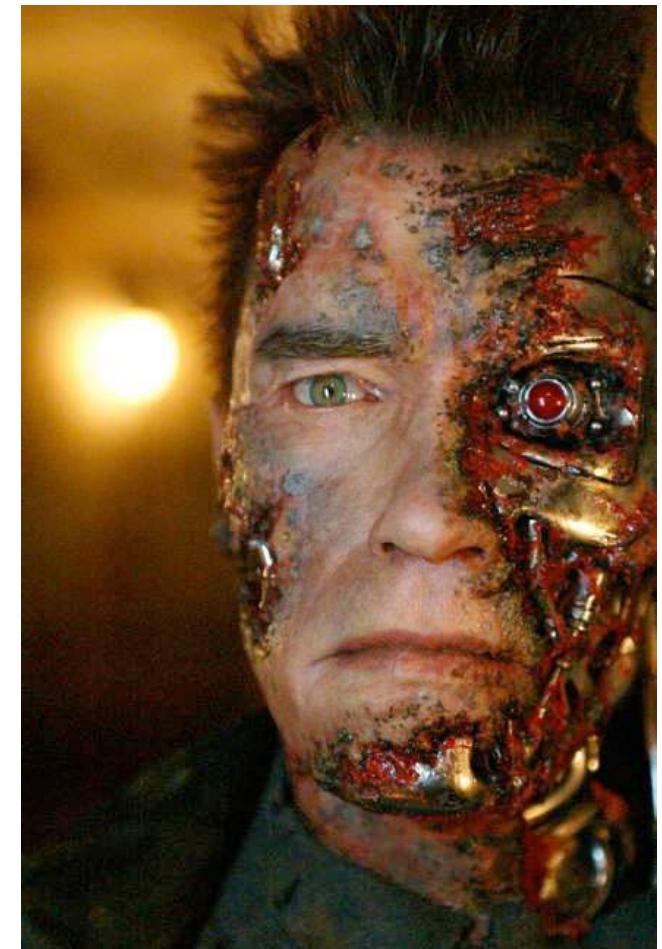


Morgan Quigley, Brian Gerkey & And William D. Smart. (2015)
Programming Robots with ROS [Chapter 12]

End of Lecture Feedback

When survey is active, respond at PollEv.com/mhanheide

THANK YOU
FOR LISTENING!



0 surveys done

⟳ 0 surveys underway

Start the presentation to see live content. Still no live content? Install the app or get help at PollEv.com/app

CMP3103M AMR

Dr. Marc Hanheide



UNIVERSITY OF
LINCOLN



LINCOLN
ROBOTICS

COURSEWORK



UNIVERSITY OF
LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M
Autonomous Mobile Robotics,
Autonomous Mobile Robotics (M),
Assessment Item One – Robotics Practical
Assignment

Indicative Weighting CMP3103M: 30%
Indicative Weighting CMP9050M: 20%

Learning Outcomes:

On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

COURSEWORK



UNIVERSITY OF
LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Autonomous Mobile Robotics (M), Assessment Item One – Robotics Practical Assignment	Indicative Weighting CMP3103M: 30% Indicative Weighting CMP9050M: 20%
---	--

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:
• [LO3] implement and empirically evaluate intelligent control strategies, by programming an autonomous mobile robot to perform complex tasks in dynamic environments

Requirements

This assessment item is split into two main tasks: “Group Robot Tasks” and “Visual Object Search”, both detailed below

Note: The “one-stop shop” for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/LCAS/teaching/wiki/CMP3103M>.

1. “Group Robot Tasks” (Criterion 1)

Your first task (relating to Criterion 1 “Group Robot Tasks” in the CRG, 30% of the assessment item one mark) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation.

The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshop sessions to program, test and demonstrate each task). Demonstrating later than the defined deadline results in a reduction of the merit mark by 50% for this individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/LCAS/teaching/wiki/CMP3103M>.

COURSEWORK

UNIVERSITY OF LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Autonomous Mobile Robotics (M), Assessment Item One – Robotics Practical Assignment	Indicative Weighting CMP3103M: 30% Indicative Weighting CMP9050M: 20%
---	--

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:
• [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements
This assessment item is split into two main tasks: "Group Robot Tasks" and "Visual Object Search", both detailed below
Note: The "one-stop shop" for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/LCAS/teaching/wiki/CMP3103M>

I. "Group Robot Tasks" (Criterion 1)
Your first task is to demonstrate "Group Robot Tasks" in the CRG. 30% of the assessment item one will consist of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation. The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshops to program, test and demonstrate the tasks). Delays to later than the defined deadline results in a deduction of 5% of the overall mark by 50% for the individual task. This is group work, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/LCAS/teaching/wiki/CMP3103M>.

2. "Visual Object Search" (Criterion 2 & 3)

Your second task (relating to Criterion 2 and 3 in the CRG, total of 70% of the mark for assessment item one) is to develop an object search behaviour, programmed in Python using ROS, that enables a robot to search for coloured objects visible in the robot's camera. This assessment is purely done in simulation, and *not* on the real robot. As part of this task, you must submit an *implementation* (criterion 2) and a *presentation* (criterion 3).

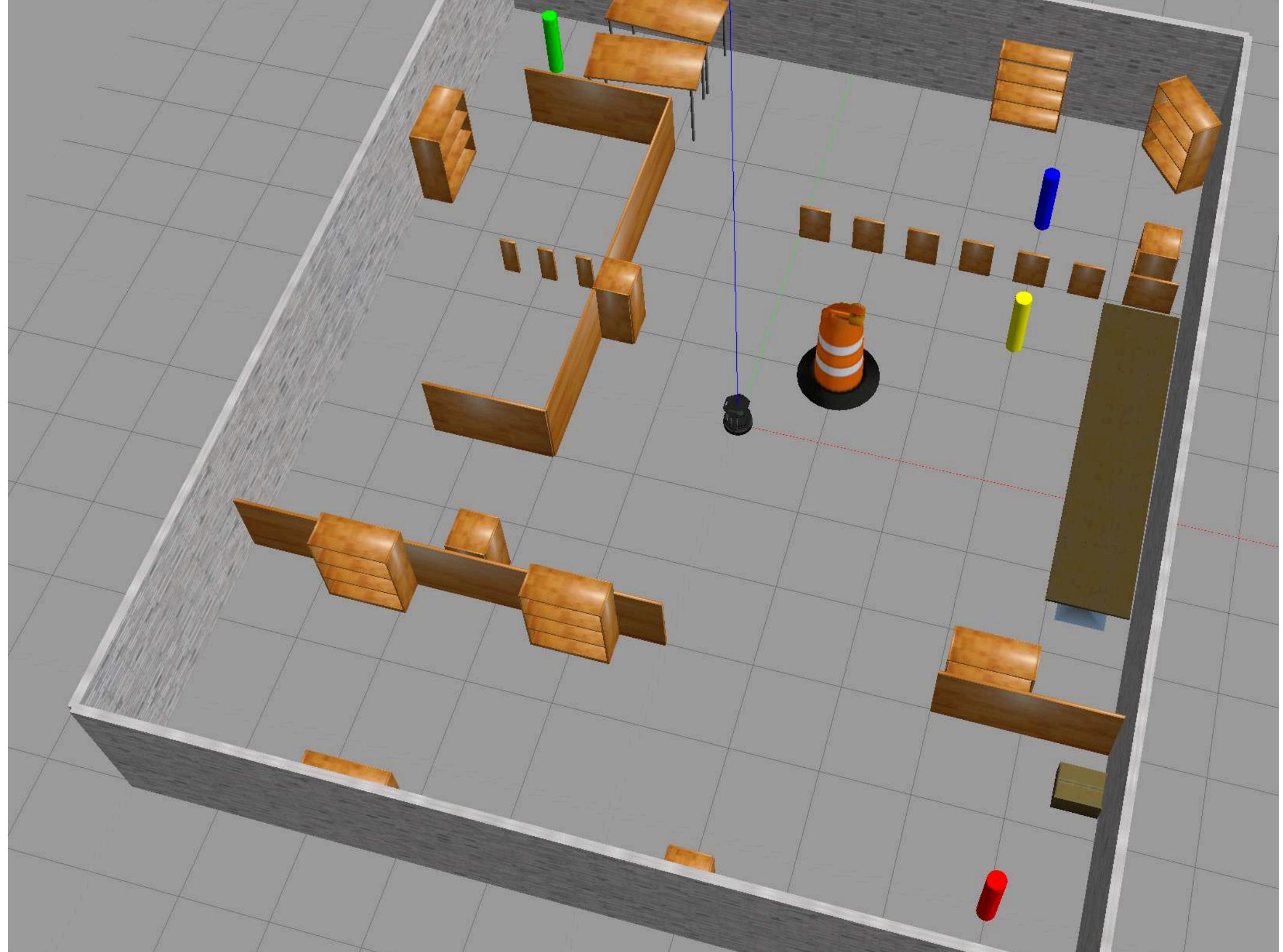
Implementation (Criterion 2)

Your task is to implement a behaviour that enables the robot in simulation to find a total of 4 objects distributed in a simulated environment. You need to utilise the robot's sensory input and its actuators to guide the robot to each item. Success in locating an item is defined as: (a) being less than 1m from the item, and (b) indication from the robot that it has found an object.

For the development and demonstration of your software component, you will be provided with a simulation environment (called "Gazebo"). The required software is installed on all machines in the Labs. The simulated environment includes four brightly coloured objects hidden in the environment at increasing difficulty. Your robot starts from a predefined position. You will be provided with a "training arena" in simulation (a simulation of an indoor environment in which 4 objects will be "hidden"). This "training arena" will resemble the "test arena" in terms of structure and complexity (same floor plan of the environment), but the positions of the objects will slightly vary to assess the generality of your approach.

You may choose any sensors available on the robot to drive your search behaviour. However, your system design should include the following elements:

1. Perception of the robot's environment using the Kinect sensor, either in RGB or Depth space, or using a combination of both RGB and Depth data in order find the object;
2. An implementation of an appropriate control law implementing a search behaviour on the robot. You may choose to realise this as a simple reactive behaviour or a more complex one, eg, utilising a previously acquired map of the environment;
3. Motor control of the (simulated) Turtlebot robot using the implemented control law.



COURSEWORK

UNIVERSITY OF LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Assessment Item One – Mobile Robotics (M), Assignment	Indicative Weighting CMP3103M: 30%
Assessment Item One – Robotics Practical	Indicative Weighting CMP9050M: 20%

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements
This assessment item is split into two main tasks: "Group Robot Tasks" and "Visual Object Search", both detailed below.
Note: The "one-stop shop" for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

1. "Group Robot Tasks" (Criterion 1)
Your first task (relating to Criterion 1 in the CRG, total of 30% of the assessment item one) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turtlebot robot and in simulation. The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshops to program, test and demonstrate the tasks). Delays later than the defined deadline results in a reduction in the overall mark by 50% for the individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

2. "Visual Object Search" (Criterion 2 & 3)
Your second task (relating to Criterion 2 and 3 in the CRG, total of 70% of the mark for assessment item one) is to develop an object search behaviour, programmed in Python using ROS, that enables a robot to search for coloured objects visible in the robot's camera. This assessment is *partly* in simulation, and *not* on the real robot. As part of this task, you must implement an *implementation* (criterion 2) and a *presentation* (criterion 3).
Implementation (Criterion 2)
Your task is to implement a behaviour that enables the robot in simulation to find a total of 4 objects distributed in a simulated environment. You need to utilise the robot's sensory input and its actuators to guide the robot to each item. Success in locating an item is defined as: (a) being less than 1m from the item, and (b) indication from the robot that it has found an object.
For the development and demonstration of your software component, you will be provided with a simulation environment (called "Gazebo"). The required software is installed on all machines in the Labs. The simulated environment includes four brightly coloured objects hidden in the environment of increasing complexity. Your robot starts from a random position. You will be required to work in "training areas" in simulation (a simulation of an indoor environment in which 4 objects will be "hidden"). This "training area" will resemble the "test area" in terms of structure and complexity (same floor plan of the environment), but the positions of the objects will slightly vary to assess generality of your approach.
You must use the sensors available on the robot to drive your search behaviour. However, your system design should include the following elements:

1. Perception of the robot's environment using the Kinect sensor, either in RGB or Depth space, taking a camera image, or using Depth in order to find the object.
2. An implementation of an appropriate control law implementing a search behaviour on the robot. You may choose to realise this as a simple reactive behaviour or a more complex one, eg, using a previously acquired map of the environment.
3. Motor control of the (simulated) Turtlebot robot using the implemented control law.

The minimum required functionality consists of a simple reactive behaviour, allowing in principle to find objects. For an average mark the behaviour should be able to successfully find some objects at unknown locations. Further extensions are possible to improve your mark in this assessment, also to enable you to find all objects. Possible extensions to the system may include (but are not limited to):

- An enhanced perception system – in-built colour appearance learning, use of additional visual cues (e.g. edges), combination of RGB and Depth features, etc.;
- Exploiting maps and other structural features in the environment or more clever search strategies.
- Utilising other existing ROS components that are available (like localisation, mapping, etc)

The software component must be implemented in Python and be supported by use of ROS to communicate with the robot. The code should be well-commented and clearly structured into functional blocks. The program must run on computers in Labs B and C. To obtain credit for this assignment you will need to demonstrate the various components of your software to the module instructors and be ready to answer questions related to the development of the solution – please follow carefully the instructions given in the lectures on the requirements for the demonstration and see below for presentation requirements.

Your implementation needs to be submitted via blackboard, with the source code containing good documentation (functionality and code quality account for 40% of assessment item one).

COURSEWORK

UNIVERSITY OF LINCOLN

Lincoln School of Computer Science

Assessment Component Briefing Document

Title: CMP3103/CMP9050M Autonomous Mobile Robotics, Assessment Item One (M), Assessment Item One – Robotics Practical Assignment	Indicative Weighting CMP3103M: 30% Indicative Weighting CMP9050M: 20%
--	---

Learning Outcomes:
On successful completion of this component a student will have demonstrated competence in the following areas:

- [LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments

Requirements
This assessment item is split into two main tasks: "Group Robot Tasks" and "Visual Object Search", both detailed below.
Note: The "one-stop shop" for resources relating to this assessment component and the workshops in CMP3103M is <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

1. "Group Robot Tasks" (Criterion 1)
Your first task (relating to Criterion 1 in the CRG, total 30% of the assessment item) consists of continuous engagement with a total of four workshop tasks you work on as a group of 3-4 students, demonstrated successfully on a real Turlebot robot and in simulation. The tasks will be made available to you at the beginning of each workshop session at the latest, and are to be demonstrated to a member of the delivery team by the week after the latest (this gives each team 2 workshop sessions to program, test and demonstrate their tasks). Delays in later than the defined times results in a deduction of 5% of the overall mark by 50% for the individual task. This is *group work*, so it is fine to work together on the task, but only students who are present at the demonstration of the group work, and can answer questions about it, will be awarded the marks. Absent group members will have to demonstrate the work individually, under the same rules as outlined above, ie, will only obtain half the marks if demonstrated later than the week after the workshop task has been officially released. The individual tasks are available from <https://github.com/L-CAS/teaching/wiki/CMP3103M>.

2. "Visual Object Search" (Criterion 2 & 3)
Your second task (relating to Criterion 2 and 3 in the CRG, total 70% of the mark for assessment item one) is to develop an object search behaviour, programmed in Python using ROS, that enables a robot to search for coloured objects visible in the robot's camera. This assessment is purely done in simulation, and not on the real robot. As part of this task, you must submit an implementation (criterion 2) and a presentation (criterion 3).
Implementation (Criterion 2)
Your task is to implement a behaviour that enables the robot in simulation to find a total of 4 objects distributed in a simulated environment. Four brightly coloured objects hidden in the environment at increasing distances from the robot's position. You will be required to work with a "training arena" in simulation (a simulation of an indoor environment in which 4 objects will be "hidden"). This "training arena" will resemble the "test arena" in terms of structure and complexity (same floor plan of the environment), but the positions of the objects will slightly vary to assess generality of your approach.
You must use the sensors available on the robot to drive your search behaviour. However, your system design should include the following elements:

1. Perception of the robot's environment using the Kinect sensor, either in RGB or Depth space,
2. An implementation of an appropriate control law implementing a search behaviour on the robot. You may choose to realise this as a simple reactive behaviour or a more complex one, eg, utilising a previously acquired map of the environment;
3. Motor control of the (simulated) Turlebot robot using the implemented control law.

The minimum required functionality consists of a simple reactive behaviour, allowing in principle to find objects. For an average mark the behaviour should be able to successfully find some objects at unknown locations. Further extensions are possible to improve your mark in this assessment, also to enable you to find all objects. Possible extensions to the system may include (but are not limited to):

- An enhanced perception system – in-built colour appearance learning, use of additional visual cues (e.g. edges), combination of RGB and Depth features, etc.;
- Exploiting maps and other structural features in the environment, or more clever search strategies;
- Utilising other existing ROS components that are available (like localisation, mapping, etc)

The software component must be implemented in Python and be supported by use of ROS to communicate with the robot. The code should be well-commented and clearly structured into functional blocks. The program must run on computers in Labs B and C. To obtain credit for this assignment you will need to demonstrate the various components of your software to the module instructors and be ready to answer questions related to the development of the solution – please follow carefully the instructions given in the lectures on the requirements for the demonstration and see below for presentation requirements.

Your implementation needs to be submitted via blackboard, with the source code containing good documentation (functionality and code quality account for 40% of assessment item one).

COURSEWORK

Learning Outcome	Criterion	Pass	2:2	2:1	1st
[LO3] implement and empirically evaluate intelligent control strategies, by programming autonomous mobile robots to perform complex tasks in dynamic environments	Criterion 1: Group Robot Tasks (30%)	You and your group have demonstrated basic functionality in simulation only.	You and your group have demonstrated the full functionality in simulation only, or the basic functionality in simulation and on the real robot.	You and your group have demonstrated the full functionality in simulation and basic functionality on the real robot.	You and your group have demonstrated the full functionality in both, simulation and on the real robot.
	Criterion 2: Individual Visual Search Task (in Simulation only, 40%), Implementation	A working software component with basic functionality. Fair program structure and some code comments. The working implementation is demonstrated, accomplishing the search task partially.	A working software component with good functionality. Clear program structure and appropriate comments. The implementation is demonstrated successfully, accomplishing most of the search task with a good performance.	A good implementation with some extra functionality or originality. The program code is well structured and commented. Good demonstration of basic and additional features, accomplishing the search task with a very good performance.	An excellent implementation featuring original functionality and elements beyond the original specification. The program code is efficient, well structured and commented. The solution is demonstrated very well, highlighting the additional functionalities, accomplishing the full search task with an excellent performance.
	Criterion 3: Individual Visual Search Task (in Sim, 30%), Presentation	A basic presentation of the system design and its performance.	A good presentation of the system design and reflections on its performance.	A very good presentation of the system design and reflections on its performance, including evidence of testing and evaluation of the important system features.	An excellent presentation of the system design and reflections on its performance, including evidence of thorough testing and evaluation of the important system features.
Weighting	The criteria for this assessment are weighted as indicated.				

What's wrong with this code?

```
import rospy
from std_msgs.msg import String
from sensor_msgs.msg import LaserScan

class FirstSub:
    def __init__(self):
        self.sub = rospy.Subscriber('/turtlebot_2/scan',
                                    LaserScan,
                                    callback=self.callback)
        self.pub = rospy.Publisher('/out', String)

    def callback(self, msg):
        for range in msg.ranges:
            if range < 1.0:
                s = String()
                s.data = "we are too close!"
                self.pub.publish(s)

fp = FirstSub()
rospy.init_node("first-subscriber")
rospy.spin()
```

node is initialised
after Subscriber and
Publisher are created

Illegal constructor for
Publisher

an import statement
is missing

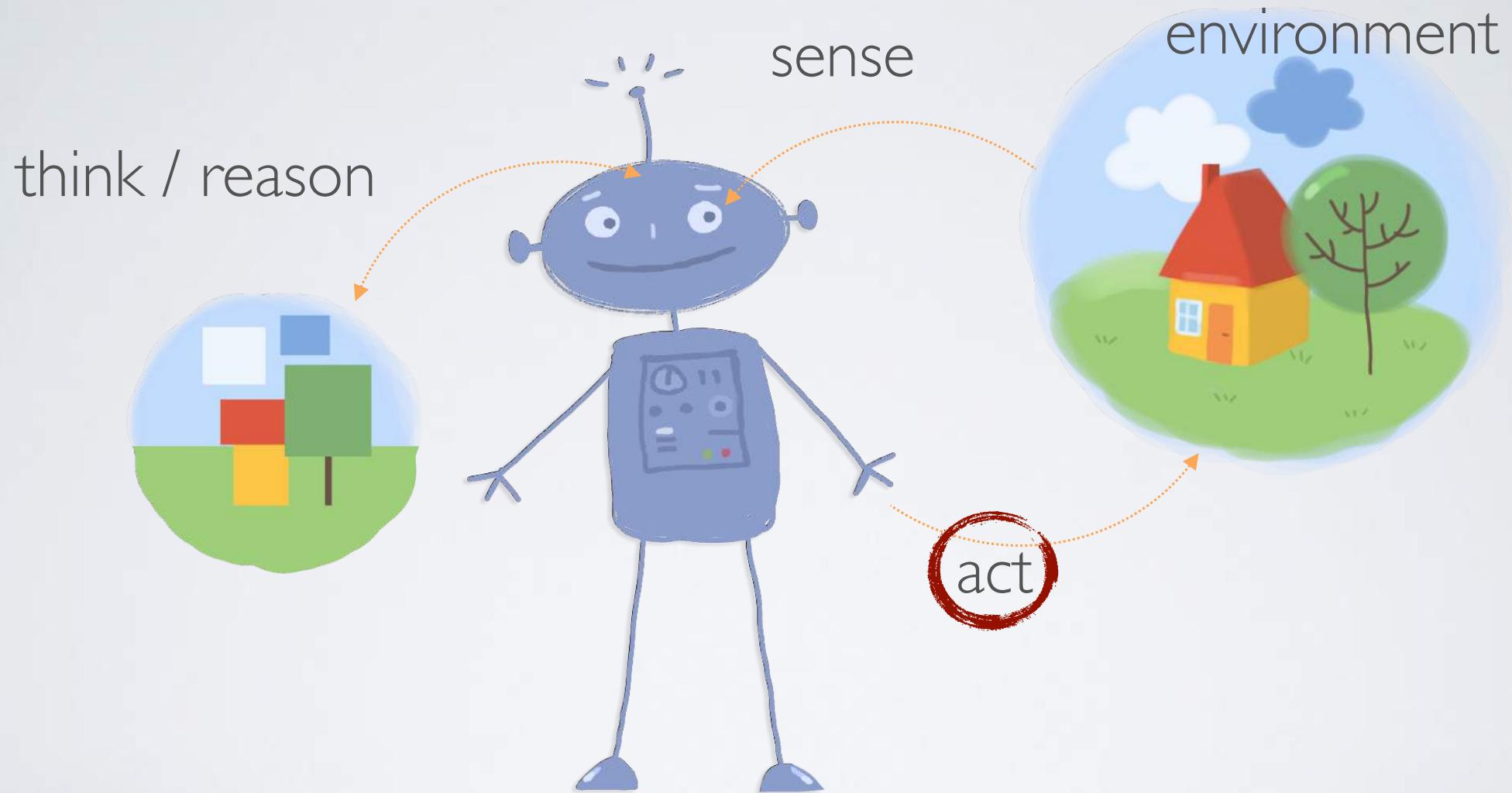
tries publishing
wrong type

construction of object
misses argument

SYLLABUS

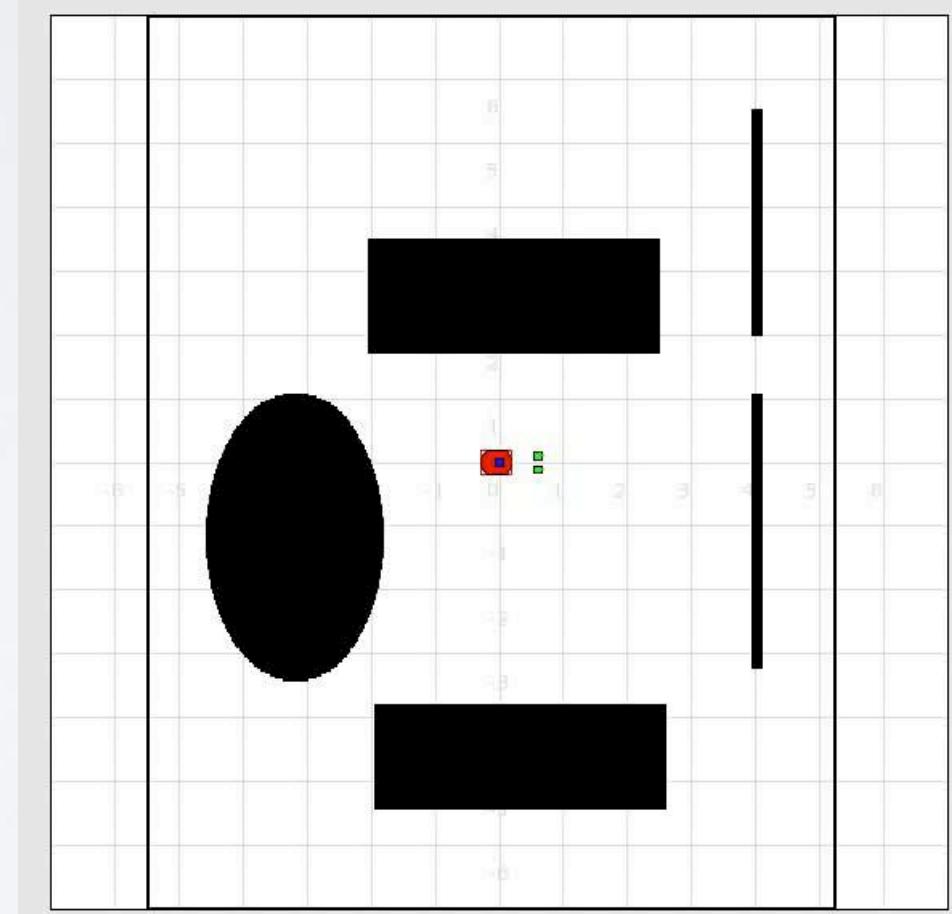
- ▶ Introduction to Robotics
- ▶ Robot Programming
- ▶ Robot Sensing & Vision
- ▶ **Robot Control**
- ▶ Robot Behaviours
- ▶ Control Architectures
- ▶ Navigation Strategies
- ▶ Map Building

ROBOTIC AGENT



MOBILE ROBOT CONTROL

- ▶ Move a robot in a desired way
 - ▶ position control
 - ▶ move to XY
 - ▶ follow a path
 - ▶ behaviours
 - ▶ follow the corridor
 - ▶ avoid obstacles
- ▶ Position control
 - ▶ open and closed loop control
- ▶ What Voltage/Current/Force to put on your motors?



THE TWIST

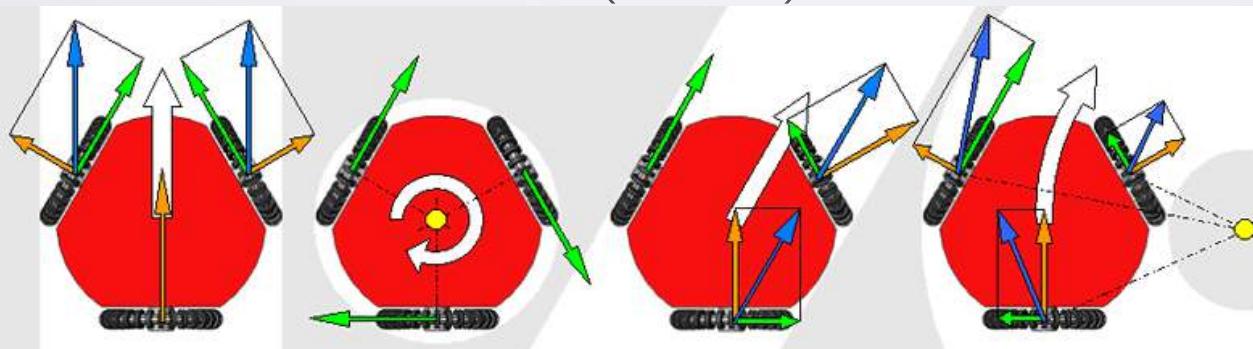
```
pub = rospy.Publisher(  
    '/cmd_vel',  
    Twist,  
    queue_size=1  
)
```

```
t = Twist()  
pub.publish(t)
```

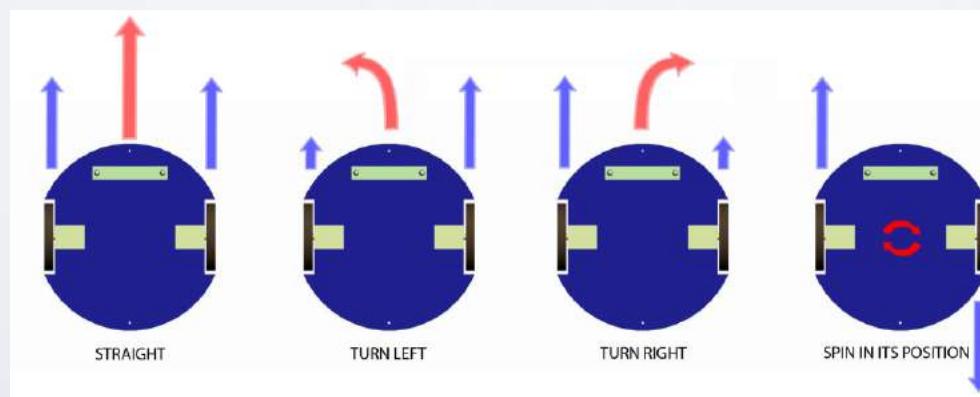


MOVEMENT COMMANDS

- ▶ Dependent on the wheel configuration
- ▶ Omni-directional Drive (Rovio)



- ▶ Differential Drive (Turtlebot/Roomba)



CONTROLLING TURTLEBOT

- ▶ Which speeds to turn each wheel to move in desired direction?
- ▶ That corresponds to the Force -> Current for every motor
- ▶ This is a control problem!

FOR MOTION CONTROL

Model

Kinematics / Dynamics

Algorithm

control-parameter

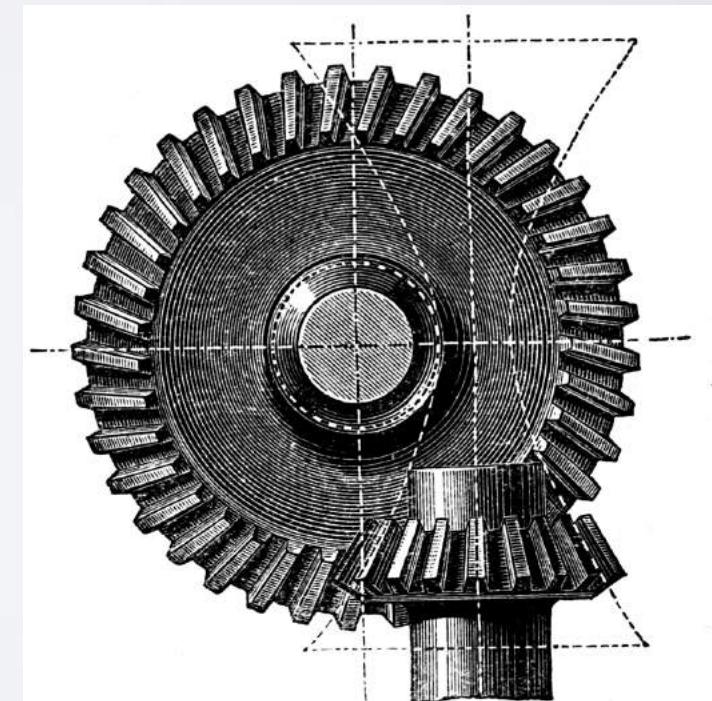
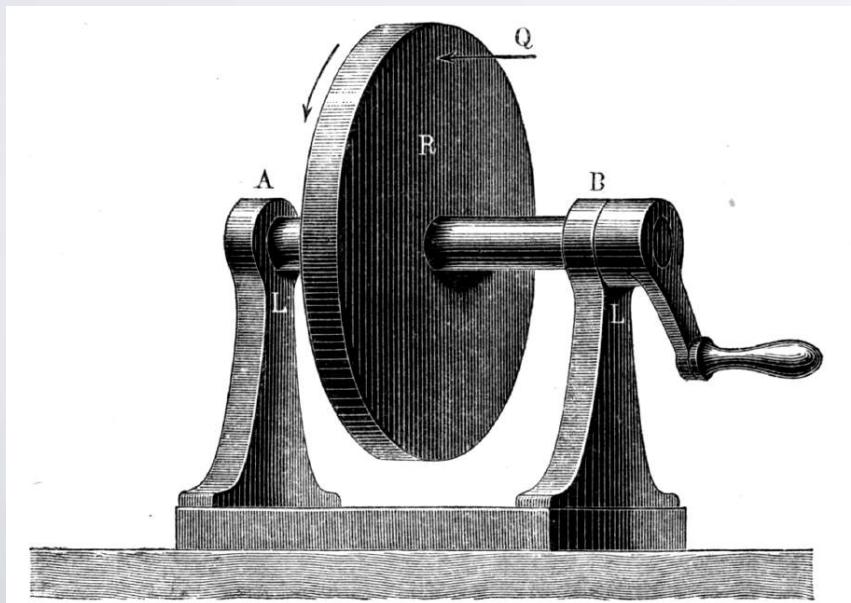
Engineering

control signal => actuators

KINEMATICS

Model

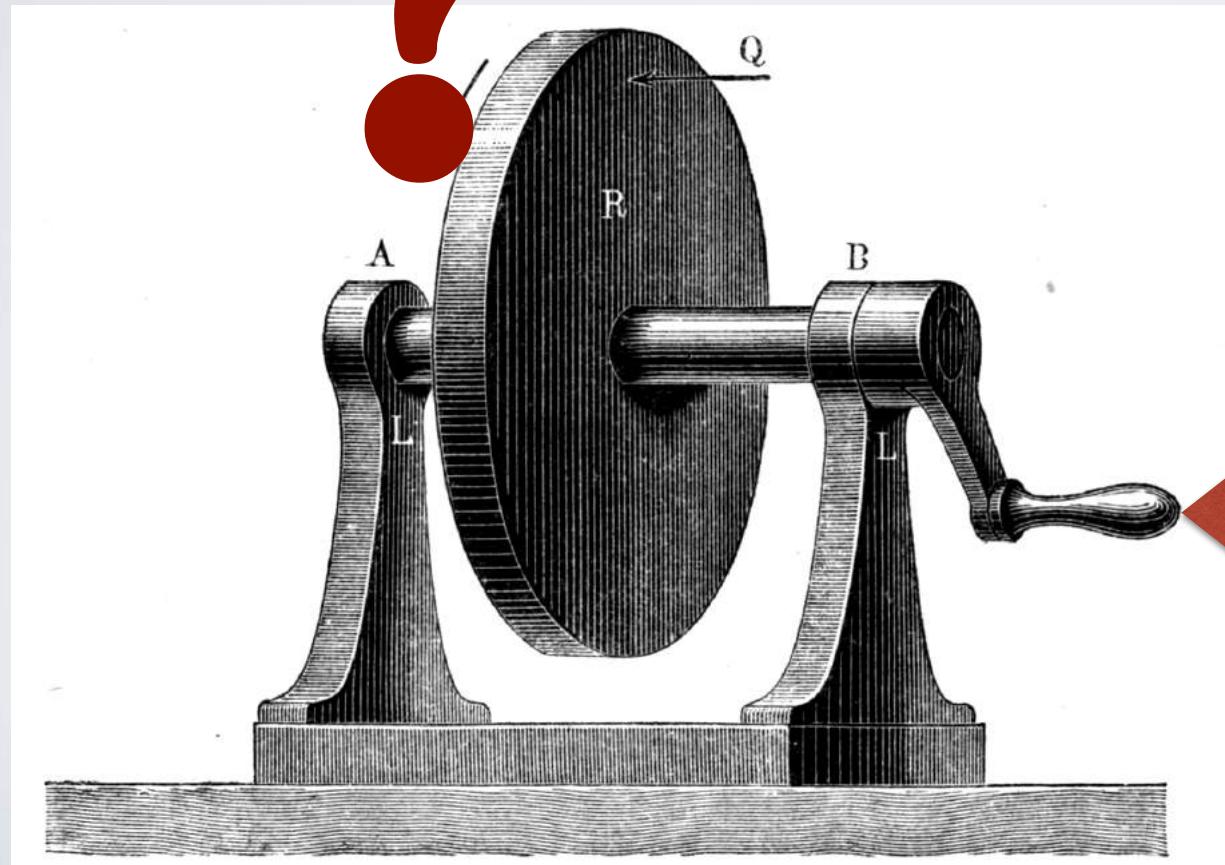
Kinematics / Dynamics



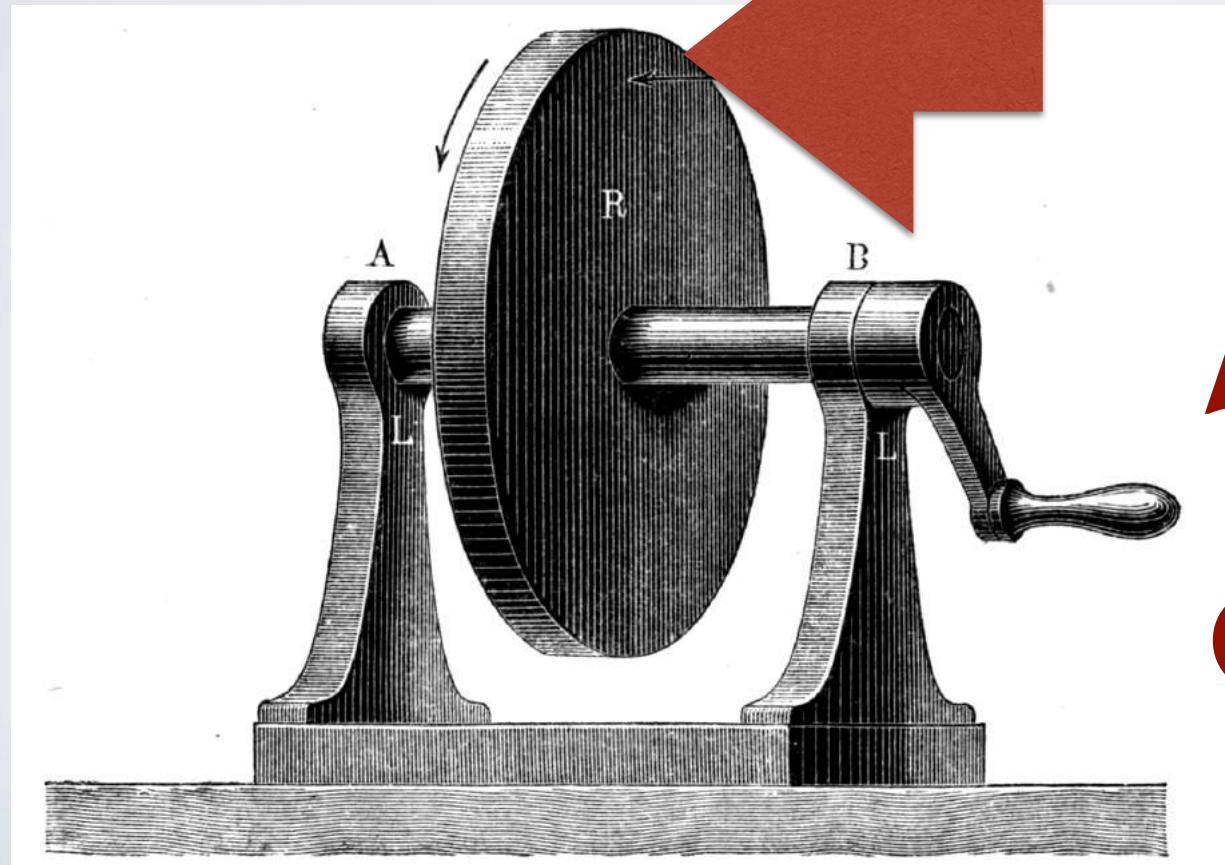
AD-HOC MODEL

- ▶ Drive in a straight line: calculate the number of steps required to perform a distance unit e.g. 1m
 - ▶ different for each speed value (non(?) - linear dependency)
 - ▶ and other factors (e.g. surface type)
- ▶ Use this value to calculate the number of steps required for a desired distance
- ▶ Similar with pure rotations/angles

FORWARD MODEL



INVERSE MODEL



AD-HOC FORWARD MODEL IS EASY?



THE WHEEL



$$c = 2\pi r$$

A SIMPLE ROBOT WITH ONE WHEEL: FORWARD MODEL



Spinning at 60 degrees per second

Degrees... Radians?

$$\begin{aligned}\omega &= (60 / 360) * 2\pi \\ &= 1.0471975511965976\end{aligned}$$

$$v = r \omega$$

c

AND WITH TWO WHEELS?

$$v = l/2 r (\omega_l + \omega_r)$$



but $\omega_l = \omega_r$ when going straight

WHAT'S THE INVERSE MODEL?



Make your turtlebot go forward
with **$v = 1\text{m/s}$** with **$r = 6\text{cm}$**

$$v = r \omega$$

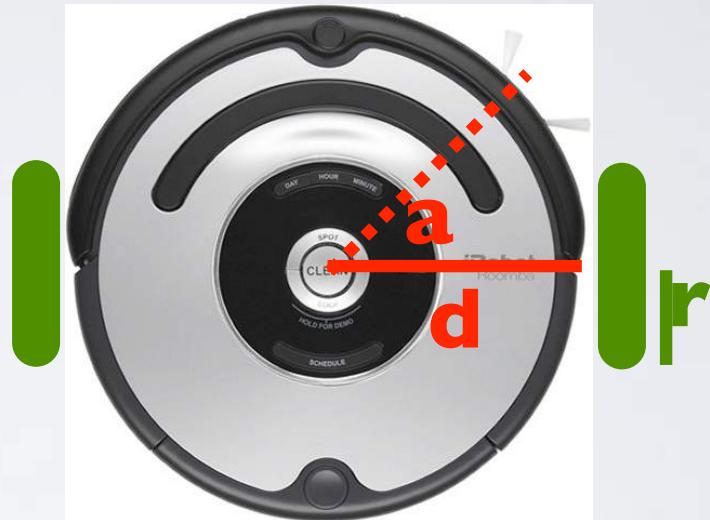
$$\omega = v/r$$

AD-HOC FORWARD MODEL IS EASY?

$$v = 0$$

$$v = 1/2 r (\omega_l + \omega_r)$$

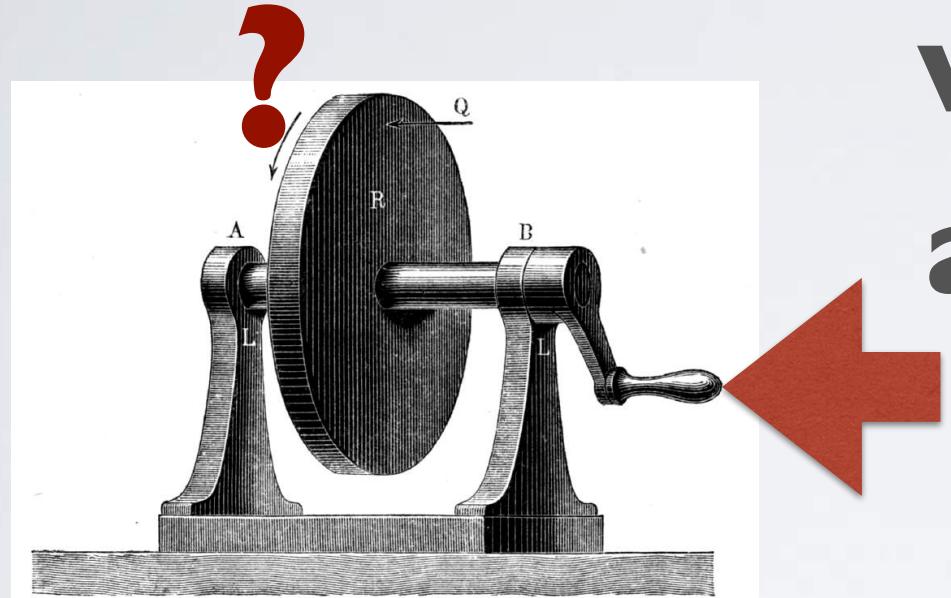
$$\Rightarrow \omega_l = -\omega_r$$



Hint: Assume both wheels moving same speed again!

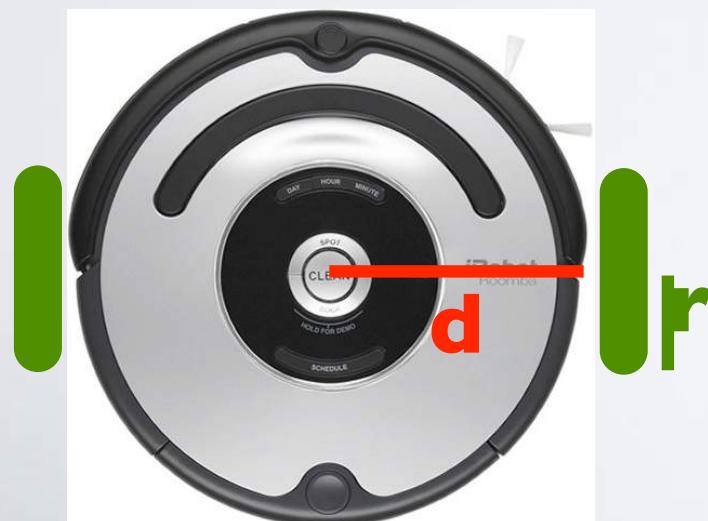
$$a = r/d (\omega_l - \omega_r)$$

FORWARD MODEL

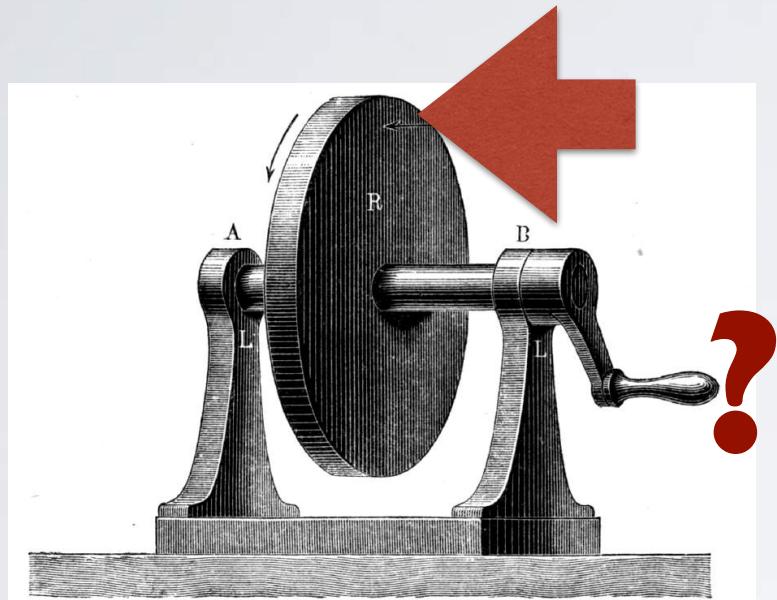


$$v = l/2 r (\omega_l + \omega_r)$$

$$a = l/d r (\omega_l - \omega_r)$$



INVERSE MODEL



$$v = l/2 r (\omega_l + \omega_r)$$
$$a = l/d r (\omega_l - \omega_r)$$



$$\omega_l = (v + (d * a)/2)/r$$
$$\omega_r = (v - (d * a)/2)/r$$



PYTHON TIME

```
import math
from geometry_msgs.msg import Twist
```





$$v = \frac{1}{2} r (\omega_l + \omega_r) \\ = (r\omega_l + r\omega_r) / 2$$

$$a = \frac{1}{d} r (\omega_l - \omega_r) \\ = (r\omega_l - r\omega_r) / d$$

PYTHON TIME

```
import math
from geometry_msgs.msg import Twist

wheel_radius = 1
robot_radius = 1

# computing the forward kinematics for a differential drive
def forward_kinematics(w_l, w_r):
    c_l = wheel_radius * w_l
    c_r = wheel_radius * w_r
    v = (c_l + c_r) / 2
    a = (c_l - c_r) / robot_radius
    return (v, a)
```





PYTHON TIME

$$\omega_l = (v + (d * a) / 2) / r$$

$$\omega_r = (v - (d * a) / 2) / r$$



```
import math
from geometry_msgs.msg import Twist

wheel_radius = 1
robot_radius = 1

# computing the forward kinematics for a differential drive
def forward_kinematics(w_l, w_r):
    c_l = wheel_radius * w_l
    c_r = wheel_radius * w_r
    v = (c_l + c_r) / 2
    a = (c_l - c_r) / robot_radius
    return (v, a)
```

```
# computing the inverse kinematics for a differential drive
def inverse_kinematics(v, a):
    c_l = v + (robot_radius * a) / 2
    c_r = v - (robot_radius * a) / 2
    w_l = c_l / wheel_radius
    w_r = c_r / wheel_radius
    return (w_l, w_r)
```



Forward and inverse Kinematics for Turtlebot

We define two functions `forward_kinematics` and `inverse_kinematics` respectively. They allow to convert from wheel speeds to robot motion and from desired robot motion to wheel speeds.

- v denotes the linear velocity of the robot
- a denotes the angular velocity of the robot
- w_l is the angular velocity of the left wheel
- w_r is the angular velocity of the right wheel

unit are m for geometry, m/s for linear velocity and rad/s for angular velocity

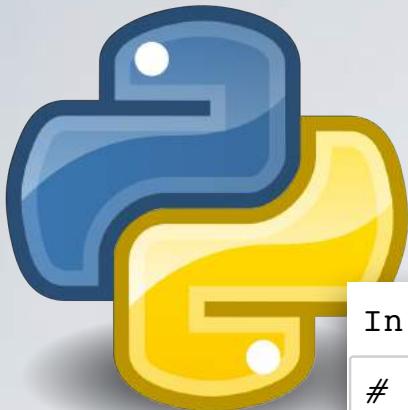
In [8]:

```
import math
#from geometry_msgs.msg import Twist

# some estimates for the robot geometry
wheel_radius = 0.05 # 5 cm radius of wheel
robot_radius = 0.25 # 25 cm radio of base

# computing the forward kinematics for a differential drive
def forward_kinematics(w_l, w_r):
    c_l = wheel_radius * w_l
    c_r = wheel_radius * w_r
    v = (c_l + c_r) / 2
    a = (c_l - c_r) / robot_radius
    return (v, a)

# computing the inverse kinematics for a differential drive
def inverse_kinematics(v, a):
    c_l = v + (robot_radius * a) / 2
    c_r = v - (robot_radius * a) / 2
    w_l = c_l / wheel_radius
    w_r = c_r / wheel_radius
    return (w_l, w_r)
```



PYTHON TIME

In [13]:

```
# try out forward kinematics, both wheels turning at `2pi rad/s` (one full turn
# per second)

(v, a) = forward_kinematics(2*math.pi, 2*math.pi)
print "v = %f,\ta = %f" % (v, a)

v = 0.314159,    a = 0.000000
```

In [15]:

```
# try out forward kinematics, one wheel turning at `2pi rad/s` (one full turn per
# second), the other `-2pi rad/s`

(v, a) = forward_kinematics(2*math.pi, -2*math.pi)
print "v = %f,\ta = %f" % (v, a)

v = 0.000000,    a = 2.513274
```

In [16]:

```
# inverse kinematics:

(w_l, w_r) = inverse_kinematics(1.0, 0.0)
print "w_l = %f,\tw_r = %f" % (w_l, w_r)

# this should give us again the desired values:
(v, a) = forward_kinematics(w_l, w_r)
print "v = %f,\ta = %f" % (v, a)

w_l = 20.000000,          w_r = 20.000000
v = 1.000000,    a = 0.000000
```



LiveSlides web content

To view

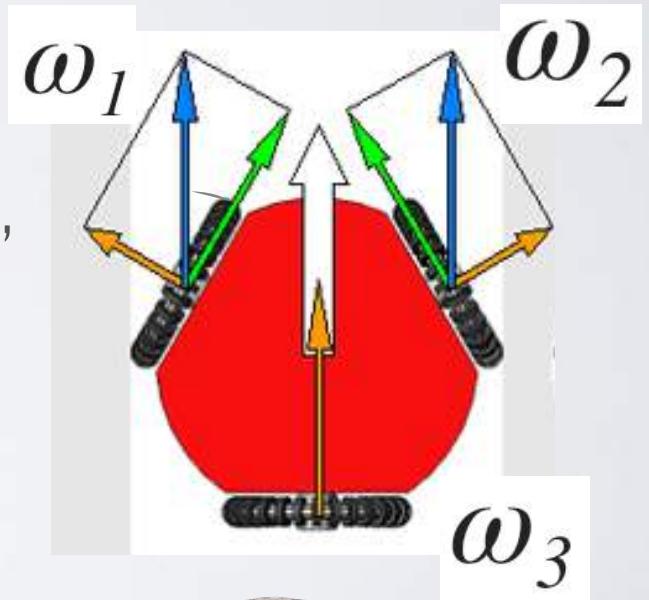
Download the add-in.

liveslides.com/download

Start the presentation.

A BIT MORE EXOTIC CONTROLLING ROVIO

- ▶ Which speeds to turn each wheel to move in desired direction?
- ▶ That corresponds to the Voltage for every motor
- ▶ This is a control problem!
- ▶ Rovio has “universal wheels” or “omni wheel”
 - ▶ simple design
 - ▶ significant slippage
 - ▶ improvement: “Mecanum wheel”



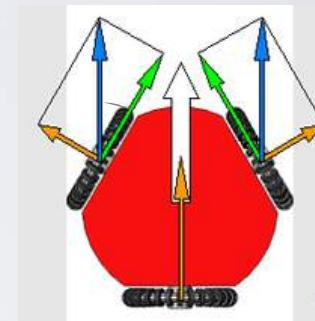
USING THE KINEMATIC MODEL

- ▶ Straight line (simplified for wheel radius=1)

$$\omega = \omega_1 = -\omega_2, \quad \omega_3 = 0$$

$$V_R = \frac{\omega r}{\cos(\alpha)}$$

forward kinematics



- ▶ Pure rotation

$$\omega = \omega_1 = \omega_2 = \omega_3$$

$$\omega_R = \frac{\omega r}{l}$$

forward kinematics



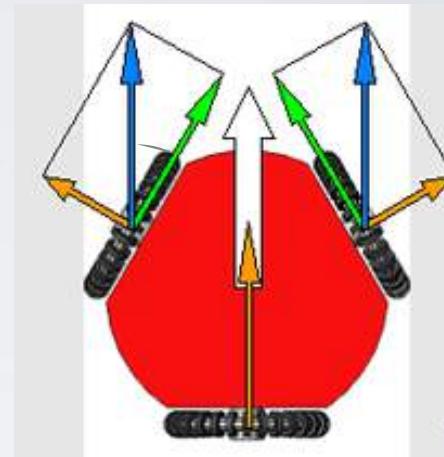
INVERSE KINEMATICS FOR ROVIO

- ▶ General (omni-drive) velocity control example

$$\omega_i = \vec{w}_i \cdot \vec{d}$$

$$\vec{w}_i = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}$$

$$\vec{d} = \begin{pmatrix} \cos(\delta) \\ \sin(\delta) \end{pmatrix} \cdot v$$

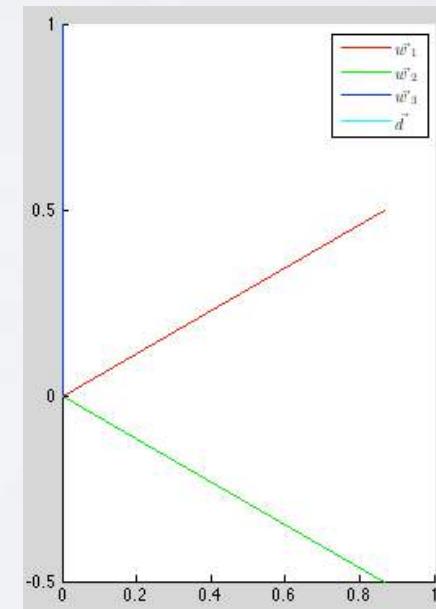
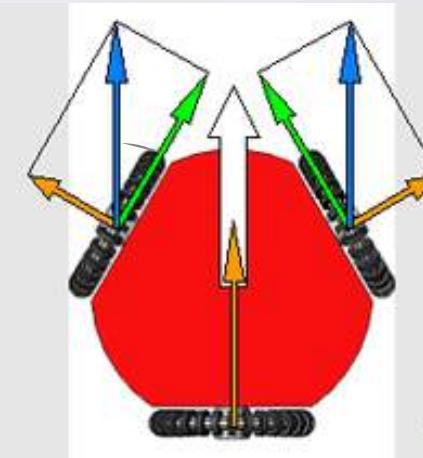


ROVIO KINEMATICS IN MATLAB

$$\vec{w}_i = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}$$

```
% Rovio Geometry
% (angles of wheels with respect to forward vector)
alpha(1)=pi/6;
alpha(2)=-pi/6;
alpha(3)=pi/2;

% create vectors w(1-3), based on wheel positions
for (i=1:3)
    w(1,i)=cos(alpha(i));
    w(2,i)=sin(alpha(i));
end;
```

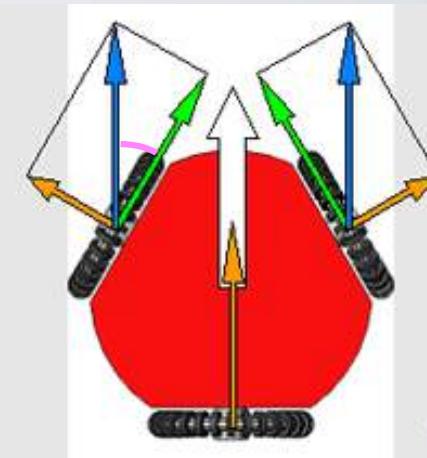


ROVIO KINEMATICS IN MATLAB

```
% Rovio Geometry
% |(angles of wheels with regard to forward vector)
alpha(1)=pi/6;
alpha(2)=-pi/6;
alpha(3)=pi/2;

% create vectors w(1-3) for wheel directions
for (i=1:3)
    w(1,i)=cos(alpha(i));
    w(2,i)=sin(alpha(i));
end;
```

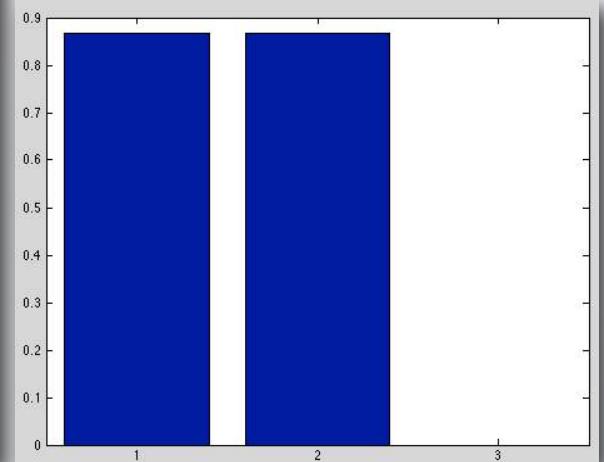
$$\begin{aligned}\omega_i &= \vec{w}_i \cdot \vec{d} \\ \vec{w}_i &= \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix} \\ \vec{d} &= \begin{pmatrix} \cos(\delta) \\ \sin(\delta) \end{pmatrix} \cdot v\end{aligned}$$



```
% now set the desired velocity and direction of the omnidrive
%desired velocity
v=1;
%desired direction
delta=0;
% -pi/3;

%resulting desired velocity vector
d=[cos(delta);sin(delta)] * v

for (i=1:3)
    omega(i)=w(:,i)'*d;
end;
```



FOR MOTION CONTROL

Model



Kinematics / Dynamics

Algorithm

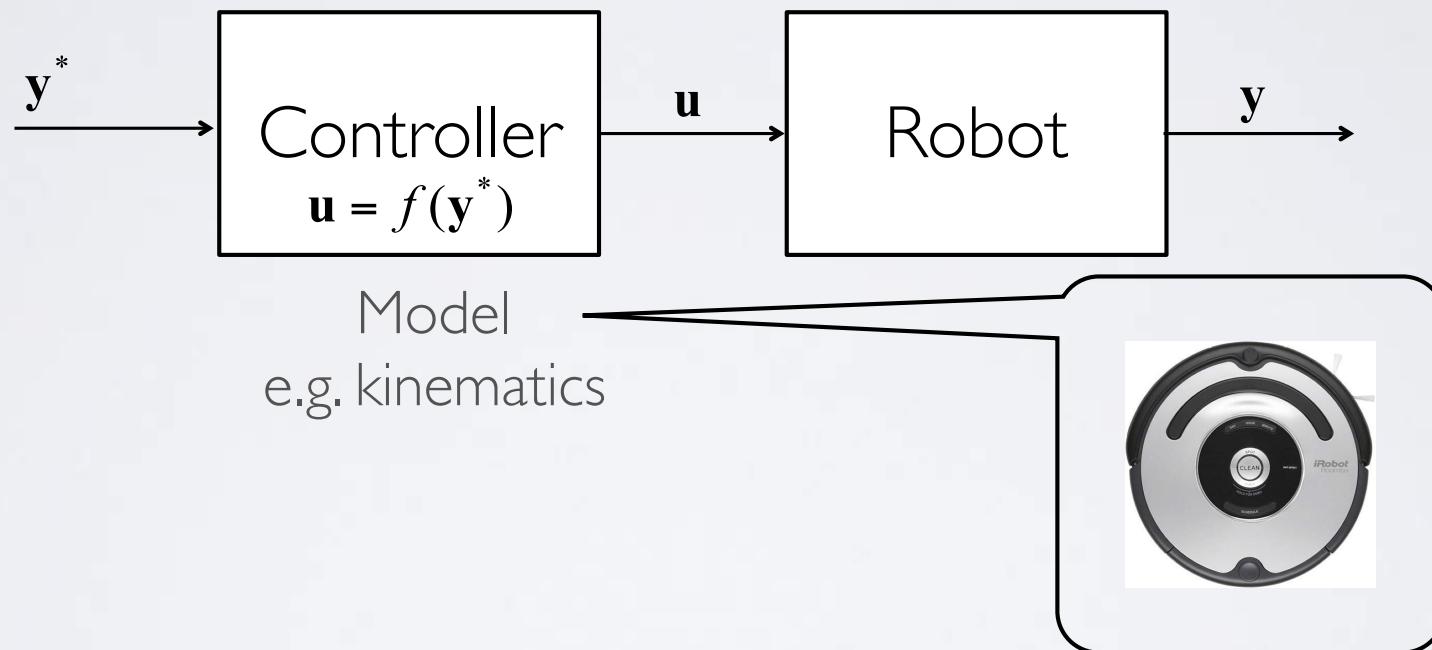
control-parameter

Engineering

control signal => actuators

OPEN LOOP CONTROL

Reference
(e.g. desired position/vel) Input
(e.g. drive command) Output
(e.g. robot's position/vel)



- ▶ Task for the controller:
 - ▶ generate control signals u so that $y = y^*$

YOU ARE A ROBOT

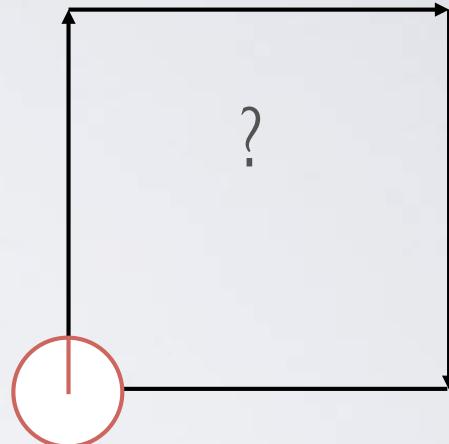
- ▶ close your eyes and hit the spot!



- ▶ ... and that's not even really open-loop control
- ▶ Proprioception: "one's own" and perception, is the sense of the relative position of neighbouring parts of the body and strength of effort being employed in movement.

OPEN LOOP CONTROL

- ▶ No measurements - no feedback
 - ▶ can rely on a model only
 - ▶ better models – smaller errors, however the errors accumulate over time
 - ▶ no possibility of correcting the errors since they are unknown
 - ▶ how to deal with unexpected events, changes, obstacles, etc.?



OPEN LOOP CONTROL, MOVE IN A SQUARE

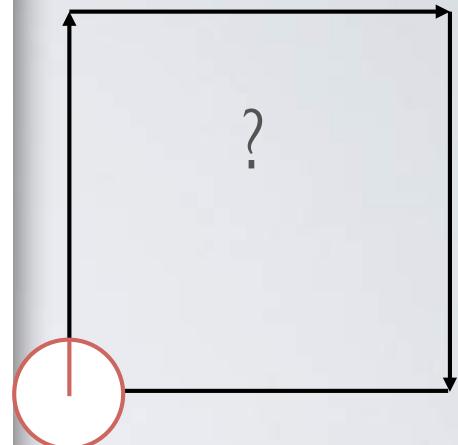
```
# 5 HZ
r = rospy.Rate(5);

# create two different Twist() variables. One for moving forward. One for turning 45 degrees.

# let's go forward at 0.2 m/s
move_cmd = Twist()
move_cmd.linear.x = 0.2
# by default angular.z is 0 so setting this isn't required

#let's turn at 45 deg/s
turn_cmd = Twist()
turn_cmd.linear.x = 0
turn_cmd.angular.z = radians(45); #45 deg/s in radians/s

#two keep drawing squares. Go forward for 2 seconds (10 x 5 HZ) then turn for 2 second
count = 0
while not rospy.is_shutdown():
    # go forward 0.4 m (2 seconds * 0.2 m / seconds)
    rospy.loginfo("Going Straight")
    for x in range(0,10):
        self.cmd_vel.publish(move_cmd)
        r.sleep()
    # turn 90 degrees
    rospy.loginfo("Turning")
    for x in range(0,10):
        self.cmd_vel.publish(turn_cmd)
        r.sleep()
    count = count + 1
    if(count == 4):
        count = 0
    if(count == 0):
        rospy.loginfo("TurtleBot should be close to the original starting position (but it's probably way off)")
```





LiveSlides web content

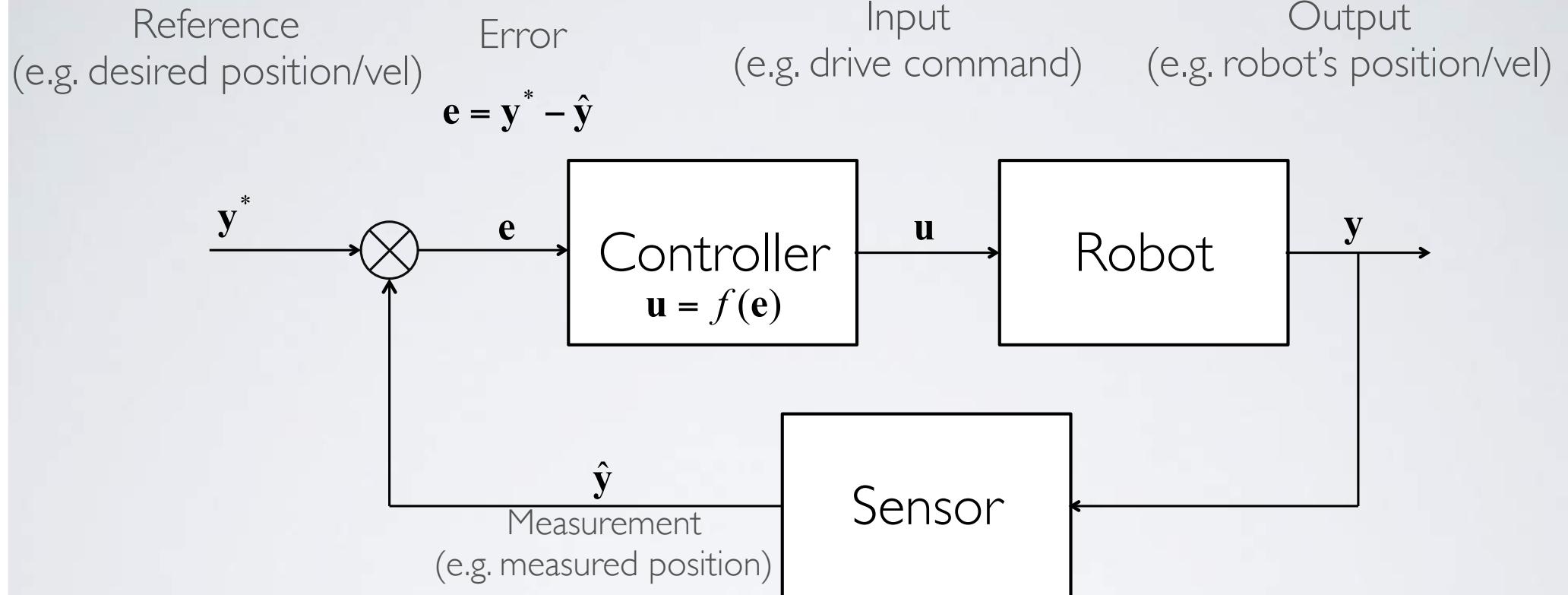
To view

Download the add-in.

liveslides.com/download

Start the presentation.

CLOSED LOOP CONTROL



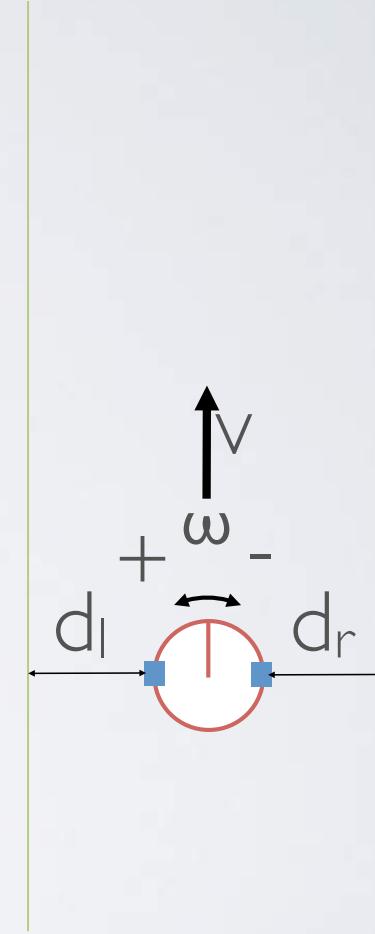
- ▶ Task for the controller:
 - ▶ generate control signals u that will keep error e as small as possible (0 would be the best)

DRIVE DISTANCE - FEEDBACK CONTROL

- ▶ Repeat
 - ▶ issue DriveForward command **u**
 - ▶ read odometry (proprioception again) and accumulate the total distance travelled **ŷ**
 - ▶ stop if the total distance is greater than the specified value $\hat{y} \geq y^*$
- ▶ Smaller time intervals – smaller errors

CORRIDOR FOLLOWING

- ▶ Scenario:
 - ▶ two sensors measuring distance to the wall d_l and d_r
 - ▶ robot moves constantly forward (v)
 - ▶ controller affects the angular speed only ($u = \omega$)
- ▶ Controller task:
 - ▶ keep $d_l = d_r$



SIMPLE CONTROLLER

- ▶ Loop:

- ▶ measure $e = d_l - d_r$
- ▶ if $e > 0$ then $\omega = +K$
- ▶ if $e < 0$ then $\omega = -K$
- ▶ $\omega = K \text{ sign}(e)$

constant

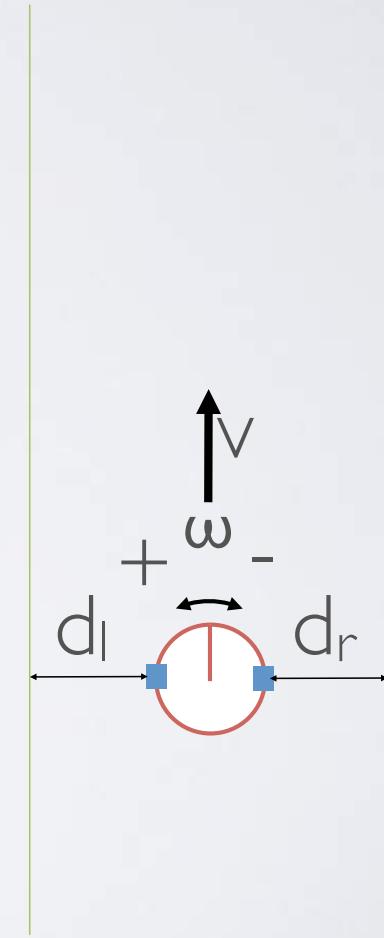
parameter

$$K = 5$$

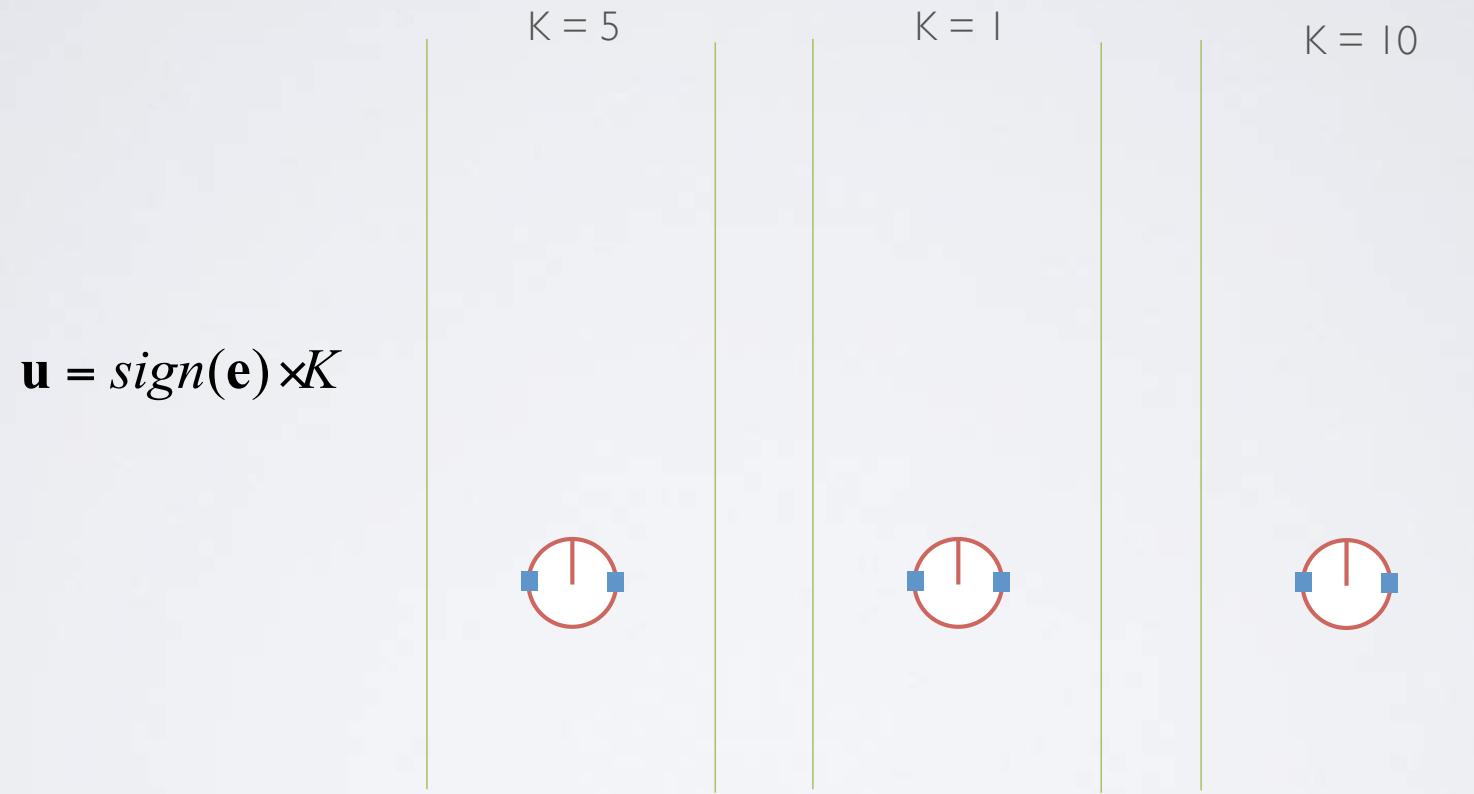
/

- ▶ In Rovio language:

- ▶ if $e > 0$ then RotateLeft(5)
- ▶ if $e < 0$ then RotateRight(5)



BANG-BANG CONTROLLER



- ▶ Control input depends only on the sign of the error



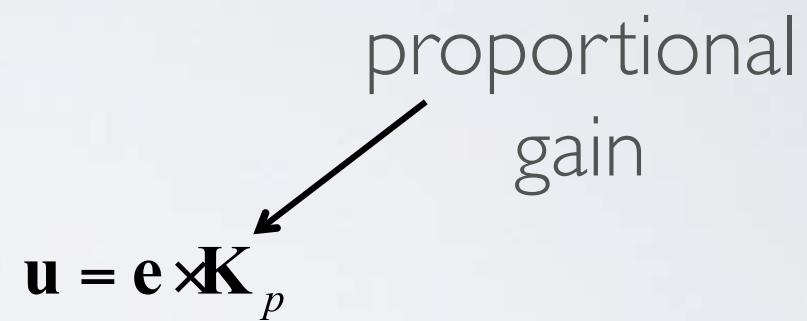
PROPORTIONAL CONTROLLER

- ▶ Change ω proportionally to the error value

- ▶ $\omega = e \times K_p$
- ▶ small e – small correction
- ▶ large e – large correction

$$u = e \times K_p$$

proportional
gain



- ▶ Result

- ▶ smoother actions and smaller errors

- ▶ K_p parameter

- ▶ large – faster reaction
- ▶ small – slower
- ▶ optimal value: smooth behaviour; robust to changes

VISION-BASED CONTROL

- ▶ Object state – in our case: x position and size

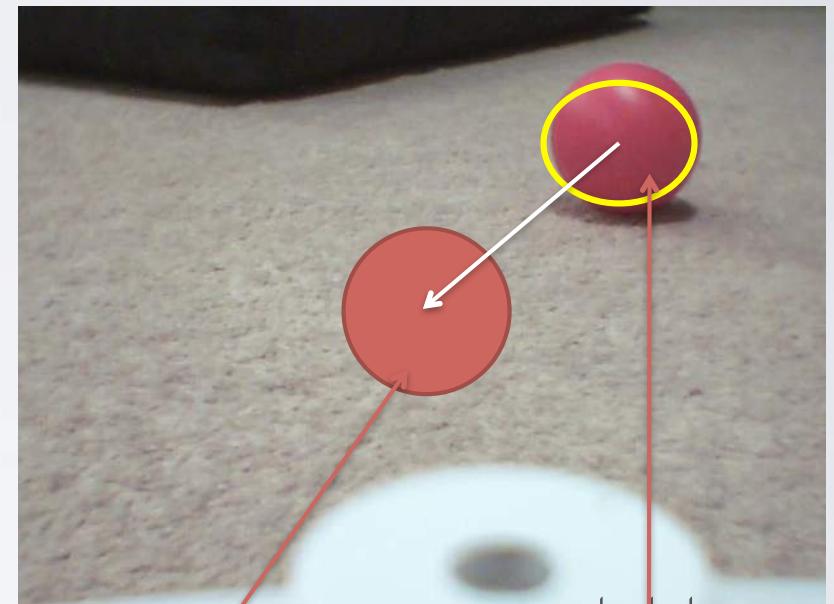
$$\mathbf{y} = \begin{bmatrix} x \\ w \times h \end{bmatrix}$$

- ▶ Error - difference between the desired and current state

$$\mathbf{e} = \mathbf{y}^* - \hat{\mathbf{y}}$$

- ▶ Control input u:

- ▶ Spin – to adjust the x position
- ▶ DriveForward - to adjust the size



desired state

current state:
information from
the object detector

FOR MOTION CONTROL

Model



Kinematics / Dynamics

Algorithm



control-parameter

Engineering

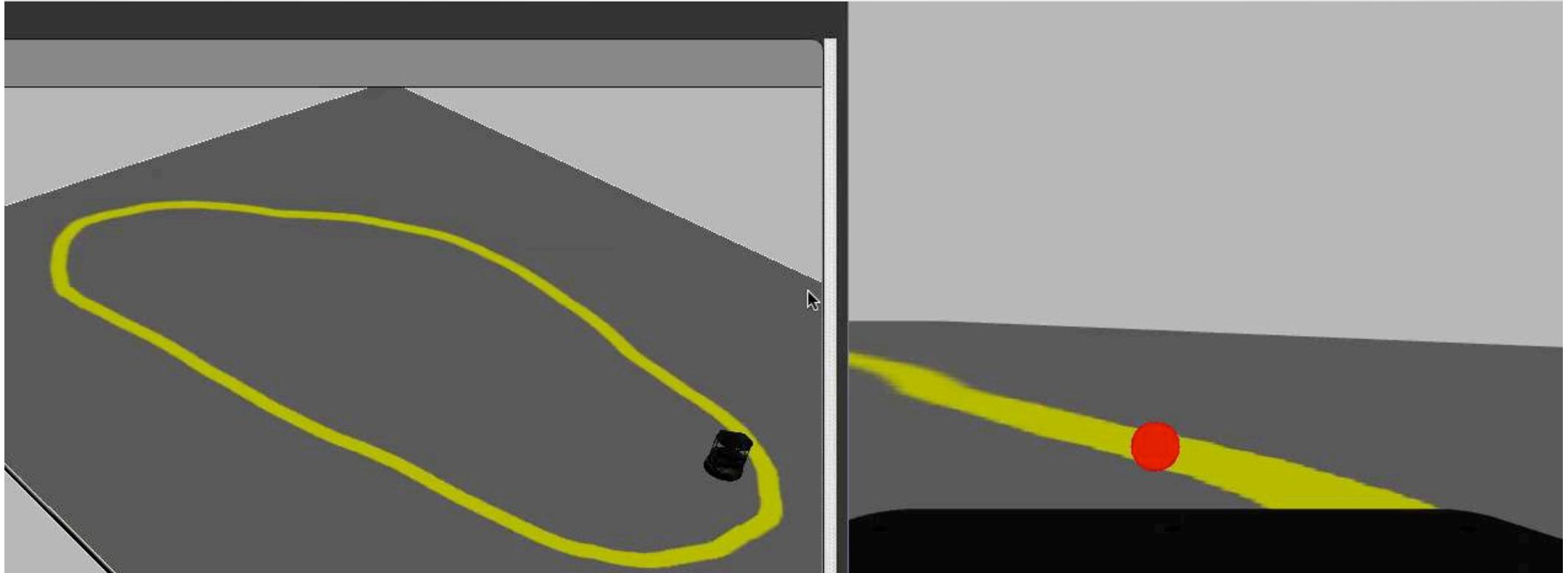
control signal => actuators



Line following project
Two light sensors.
Proportional control of the
Black level (drive forward
on White surface)
Obstacle avoidance using US
sensor

Edited by Gunnar Bolmsjö
2011

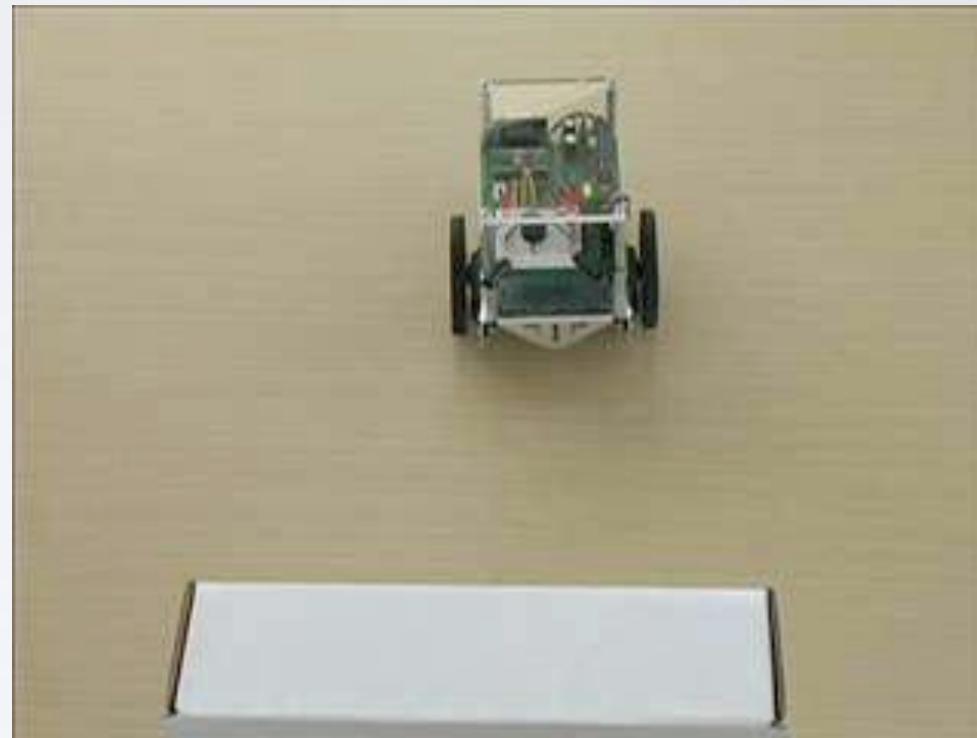
NOW FOR THE REAL STUFF



Morgan Quigley, Brian Gerkey & And William D. Smart. (2015)
Programming Robots with ROS [Chapter 12]

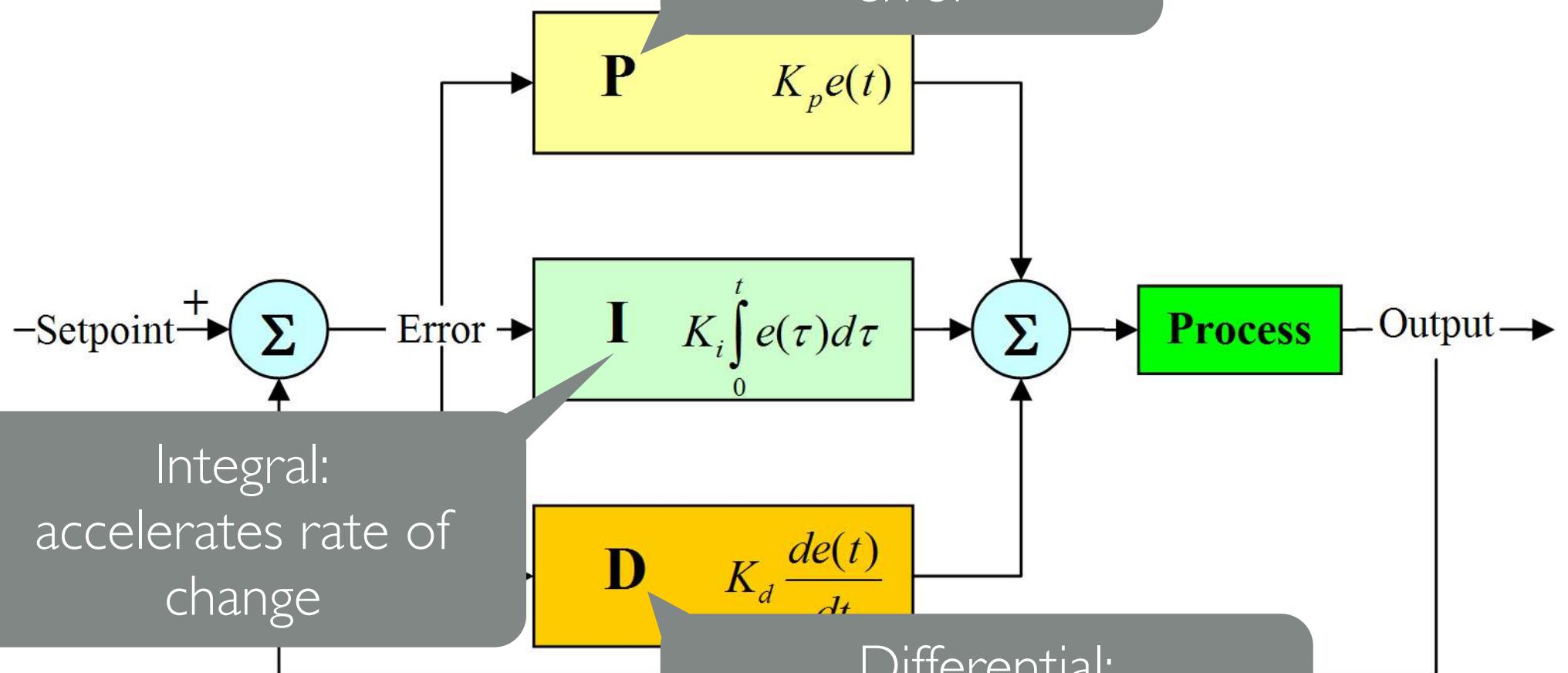
PROBLEMS WITH PROPORTIONAL CONTROLLER?

- ▶ What's the problem?



PID CONTROLLER

Proportional:
corrects the actual
error

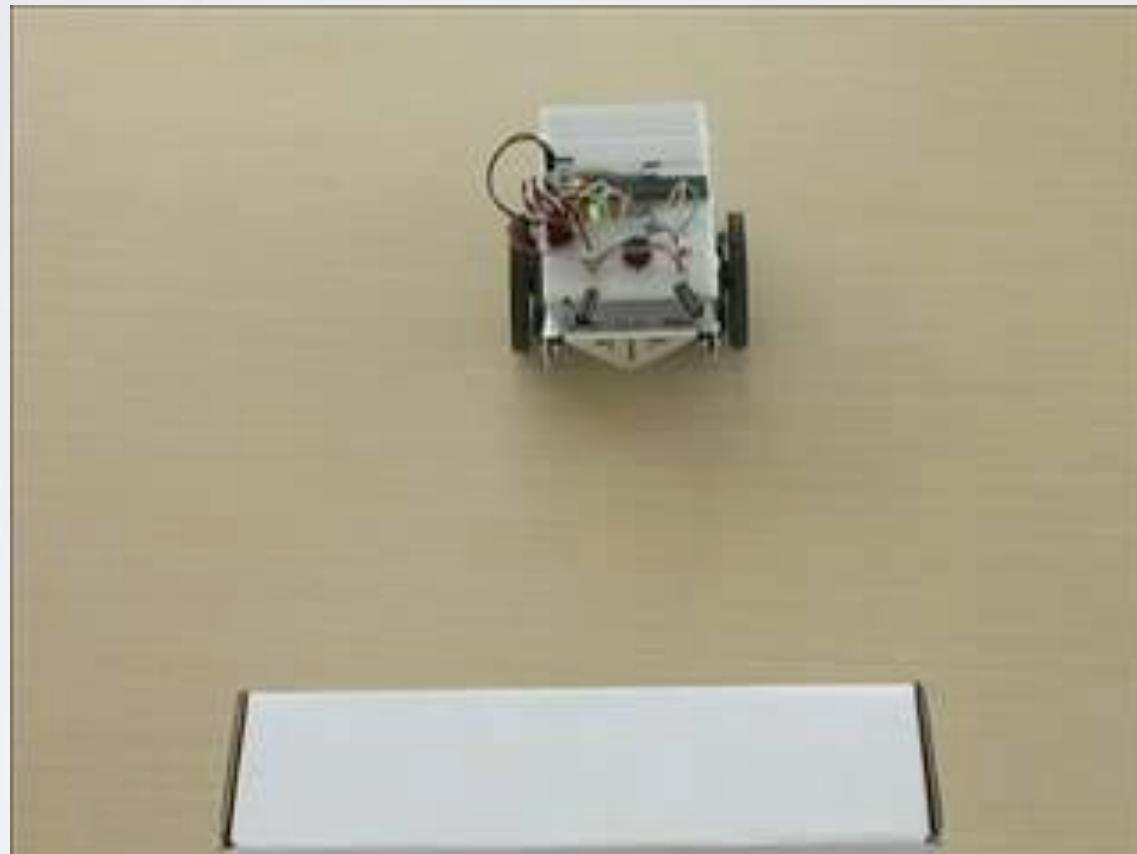


Integral:
accelerates rate of
change

Differential:
slows the rate of change
(reduces overshooting)

© SilverSTart@Wikipedia

PID CONTROLLER



APPLICATIONS – EXAMPLES

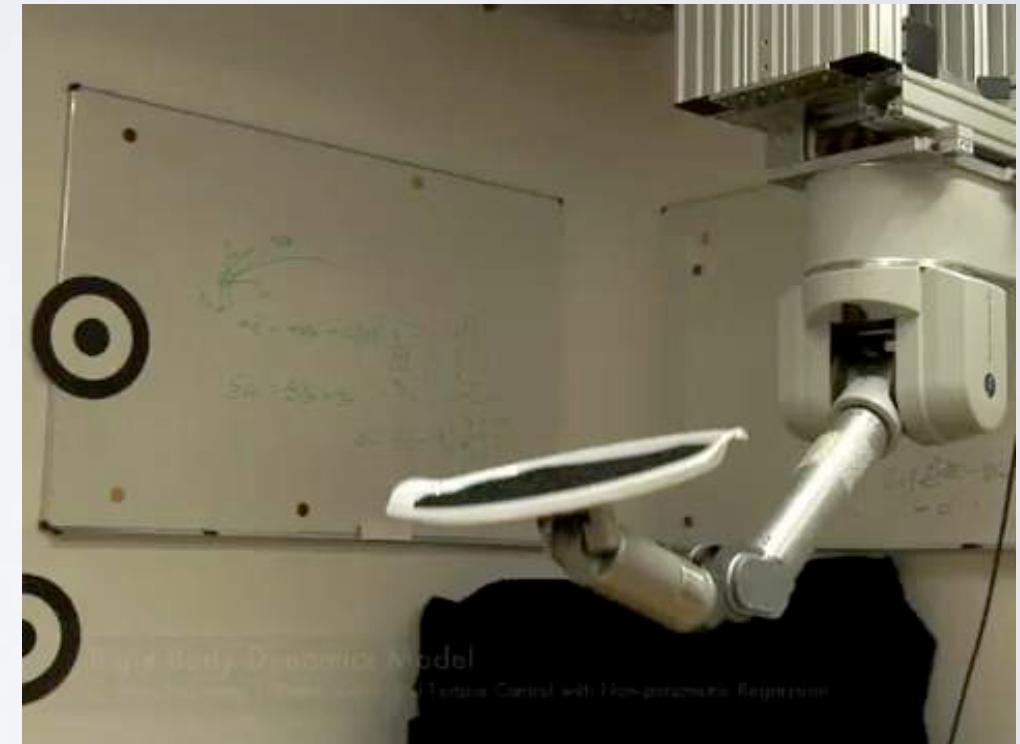
- ▶ Balancing Robot

NXTway-G
Designed, built and
programmed by
Ryo Watanabe
Waseda University
Japan

- ▶ Person Following



ANY OTHER CONTROL PROBLEMS YOU COULD THINK OF IN ROBOTICS?



Order the controller types according to their expected performance for a control problem: precision, response speed, and (energy-) efficiency

- PD Controller
- Open-Loop Controller
- bang-bang Controller
- PID Controller
- Proportional Controller

CONTROL THEORY

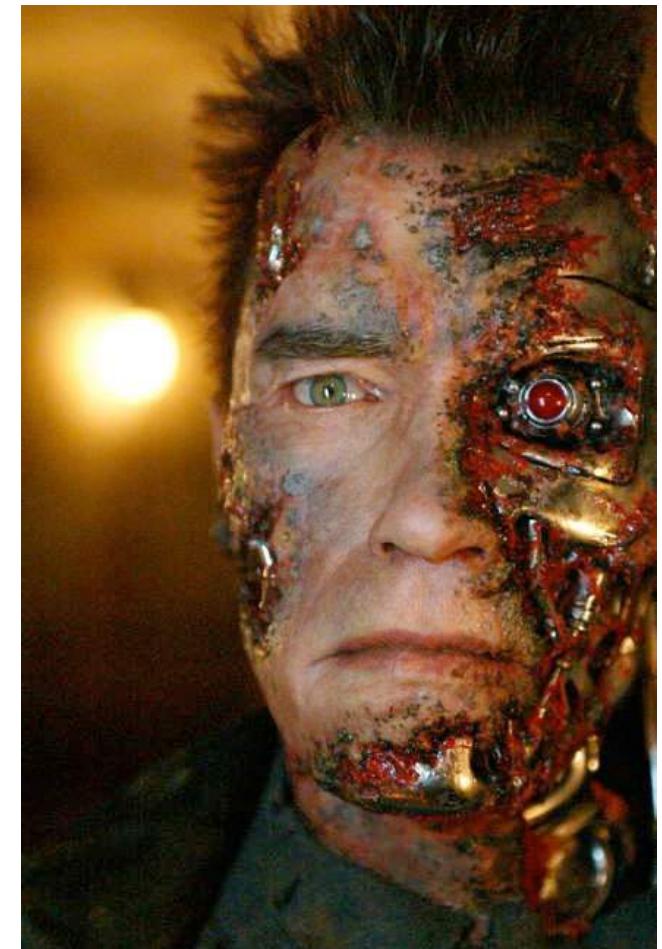
- ▶ Optimal control, minimise
 - ▶ errors
 - ▶ energy
 - ▶ response time
- ▶ Design and analysis of controllers
 - ▶ linear (P, PD, PID controllers)
 - ▶ non-linear
 - ▶ adaptive controllers
 - ▶ changing parameter values in time
 - ▶ models
 - ▶ kinematics and dynamics in robotics

read Siegwart book, chapter 4!
(on blackboard)

End of Lecture Feedback

When survey is active, respond at PollEv.com/mhanheide

THANK YOU
FOR LISTENING!



0 surveys done

⟳ 0 surveys underway

Start the presentation to see live content. Still no live content? Install the app or get help at PollEv.com/app

CMP3103M/CMP9050M

Autonomous Mobile Robotics

Dr Ayse Kucukyilmaz

University of Lincoln
Centre for Autonomous Systems

INB3201
Office hours: Wednesday 11:00-13:00

akucukyilmaz@lincoln.ac.uk
<http://webpages.lincoln.ac.uk/akucukyilmaz/>

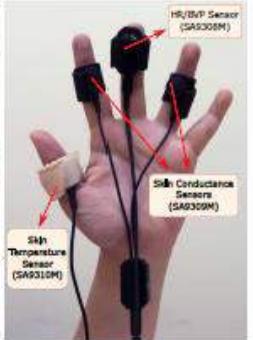
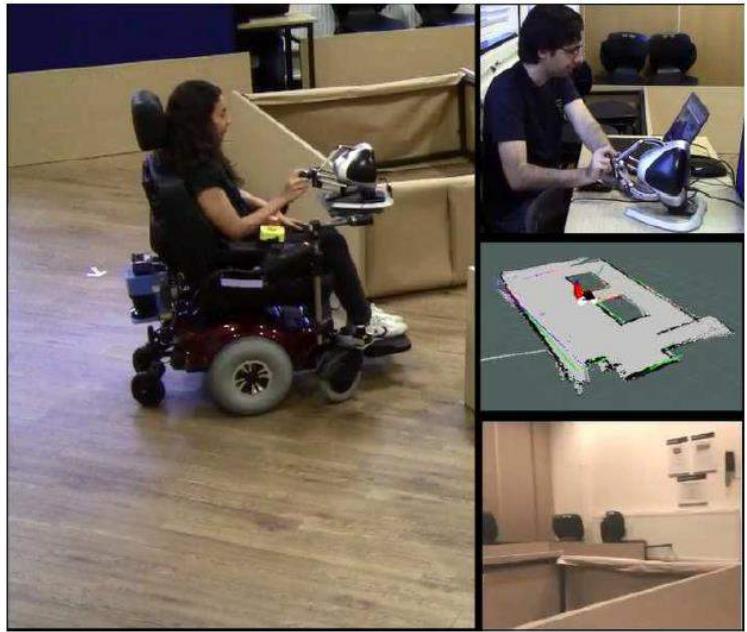


UNIVERSITY OF
LINCOLN



Research Highlights

- Adaptive haptic shared control
- Closed-loop physical human-machine interaction
- Learning by demonstration
- Behavior recognition



Syllabus

1. Introduction (MH)
2. Robot Programming (MH)
3. Robot Sensing (MH)
4. Motion and Control (MH)
5. **Robot Behaviours and Navigation (AK)**
6. Mapping (AK)
7. Localisation (AK)
8. Planning (AK)
9. Control Architectures (PB)
10. Human Robot Interaction 1 (PB)
11. Human Robot Interaction 2 (PB)
12. Applications (PB)



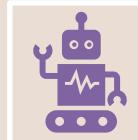
BEHAVIOUR-BASED ROBOTICS

Behaviour

NOUN

The way in which an animal or person behaves in response to a particular situation or stimulus.

Today's Lecture



Behaviour-based robotics



Braitenberg's vehicles



Learning robot behaviours

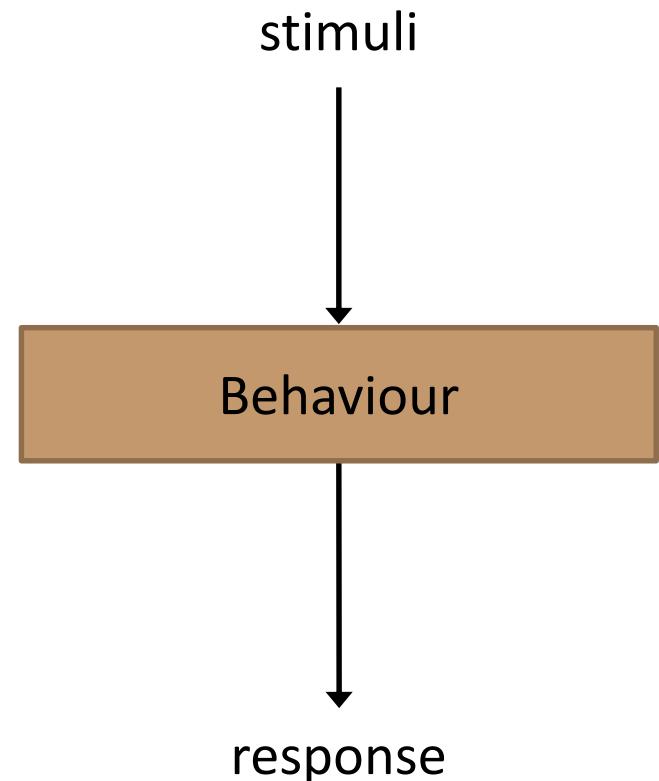


Combining robot
behaviours

Behaviour-Based Robotics

Robot Behaviour:

- Reaction to a stimulus [psychology]
- Basic building block for robot actions
- Often **reactive behaviours**: no internal data interpretation (fast reactions!)
- Reactive behaviours perform **closed loop control**
- Different behaviours can be combined into complex ones – “emergent behaviours”



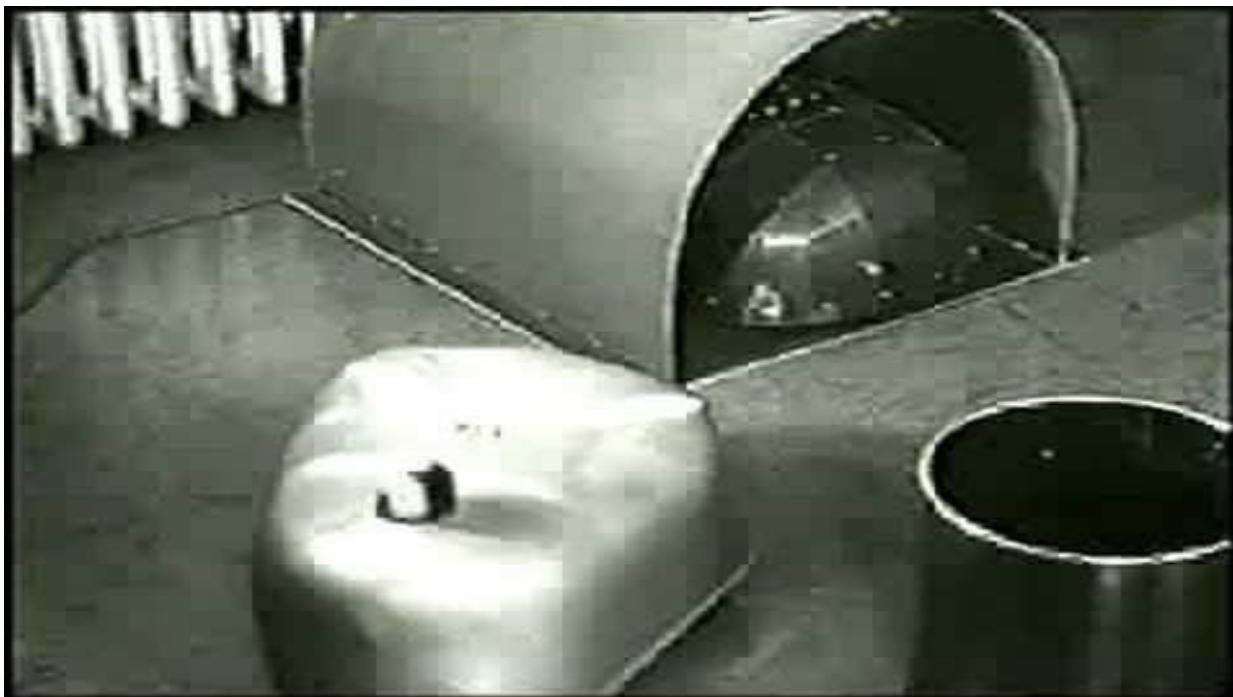
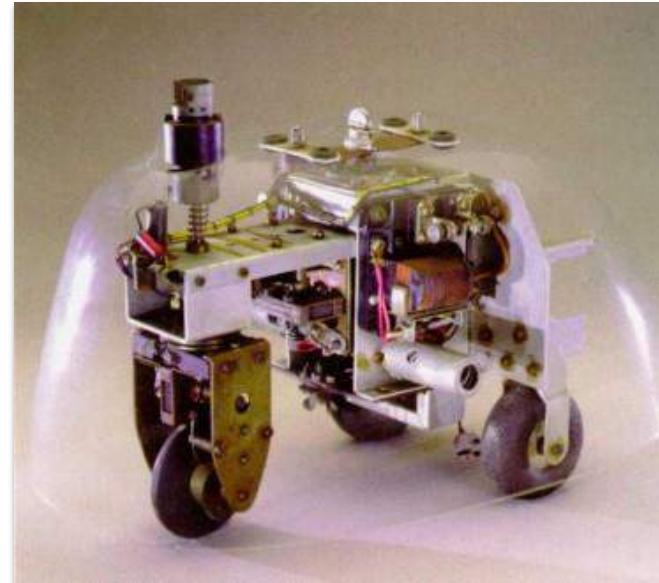
Robot Behaviours

Example	Behaviour Category
wandering	exploration/directional
goal following	goal-oriented, appetitive
obstacle avoidance	aversive/protective
road following	path following
balance	postural behaviours
flocking	social/cooperative
visual search	perceptual

Grey Walter's Turtles

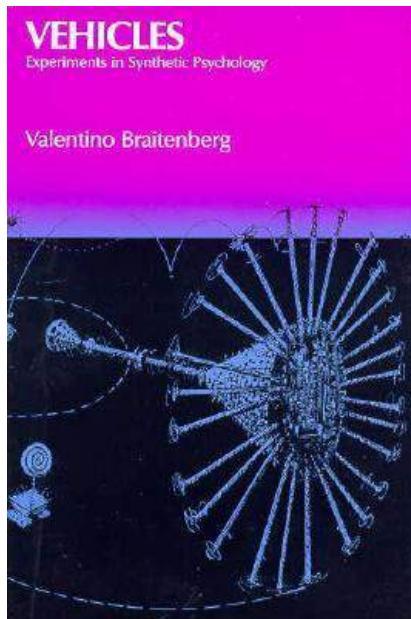
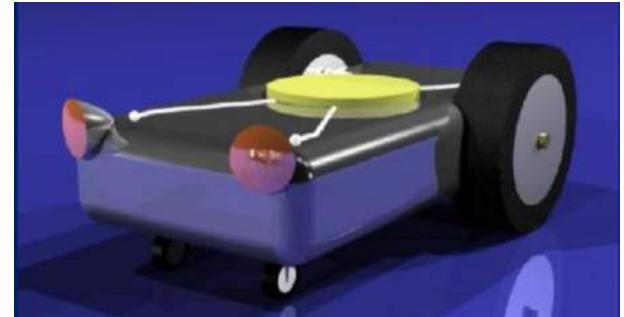
“Machina Speculatrix”, 1948

- experiments in reflex behaviour
- built of electronic valves and photo-cells
- approach or escape a light source



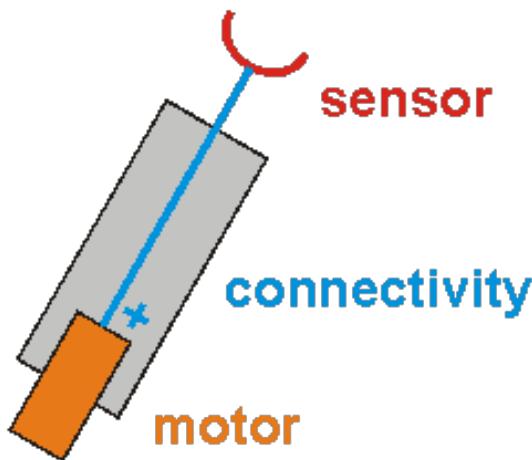
[https://www.youtube.com
/watch?v=ILULRlmXkKo](https://www.youtube.com/watch?v=ILULRlmXkKo)

Braitenberg's Vehicles



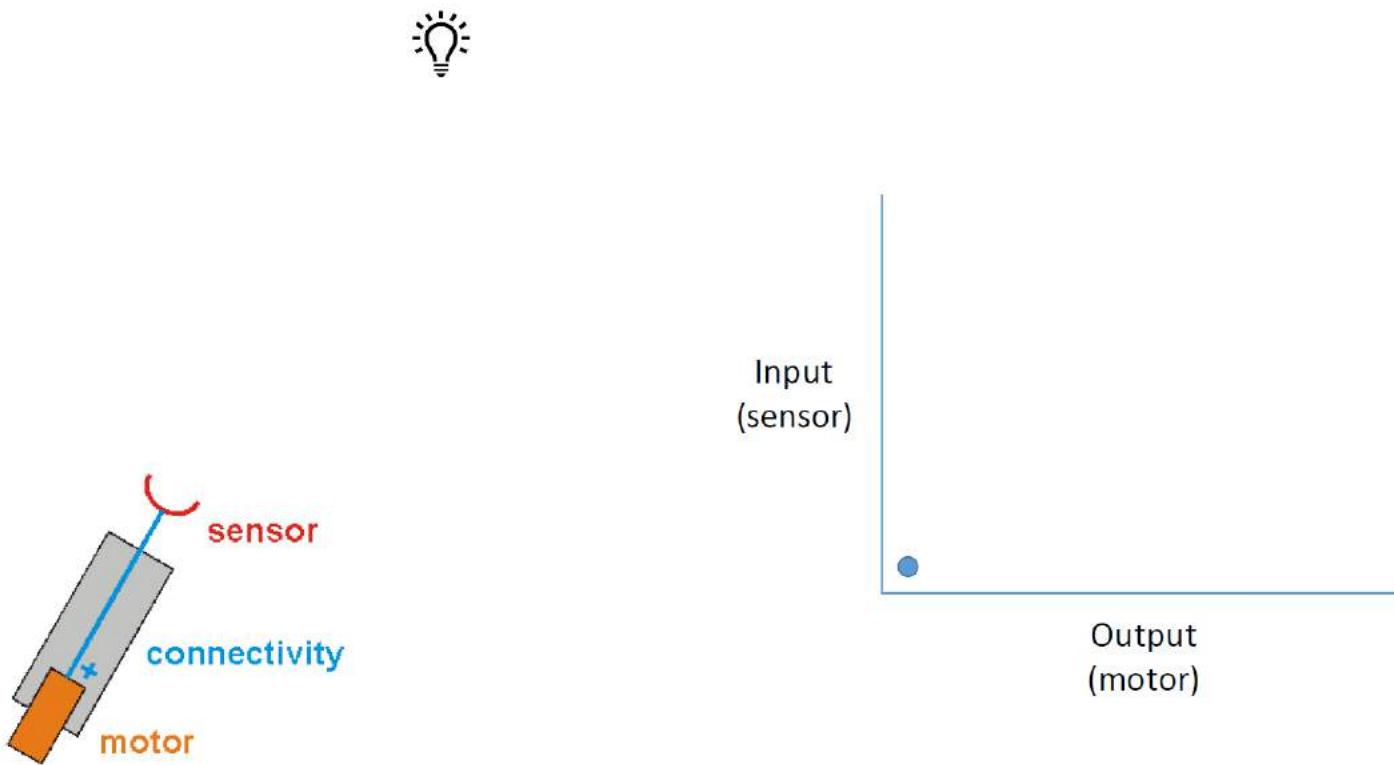
- Building seemingly complex behaviours from simple interactions.
- Valentino Braitenberg, “Vehicles: Experiments in Synthetic Psychology”, MIT Press, 1984.
- This lecture covers Vehicles 1 to 3

Vehicle 1 Getting Around

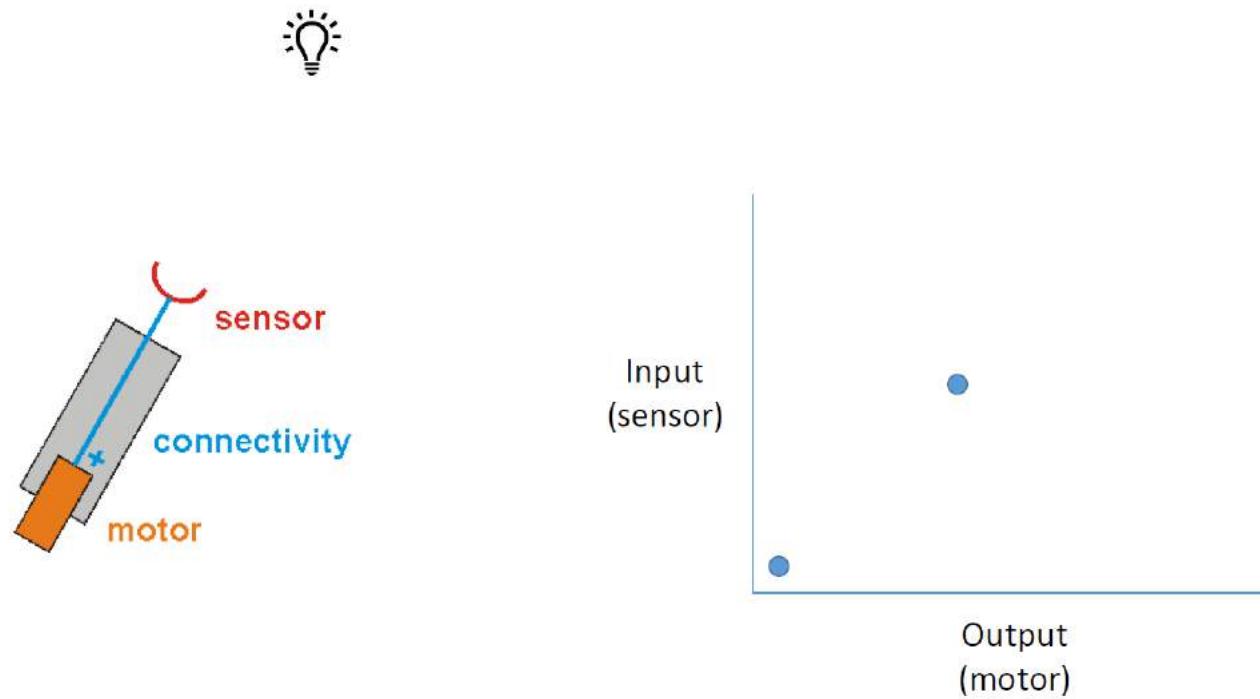


- One sensor, e.g. temperature
- One motor
- Actuation \propto sensor response
(assuming friction)

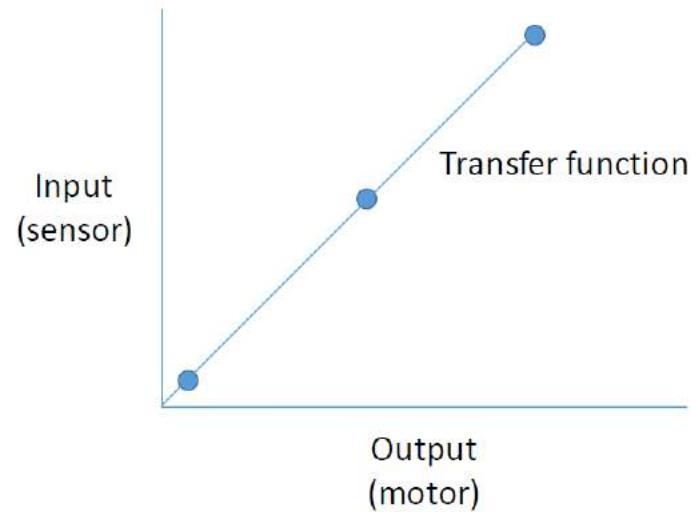
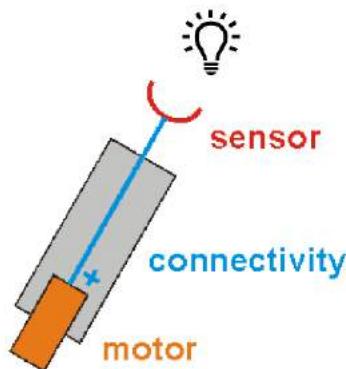
Vehicle 1 Getting Around



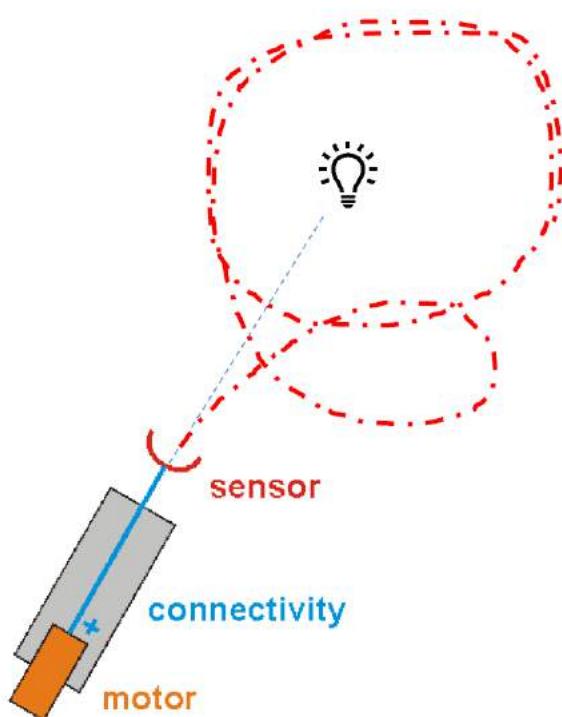
Vehicle 1 Getting Around



Vehicle 1 Getting Around



Vehicle 1 Getting Around



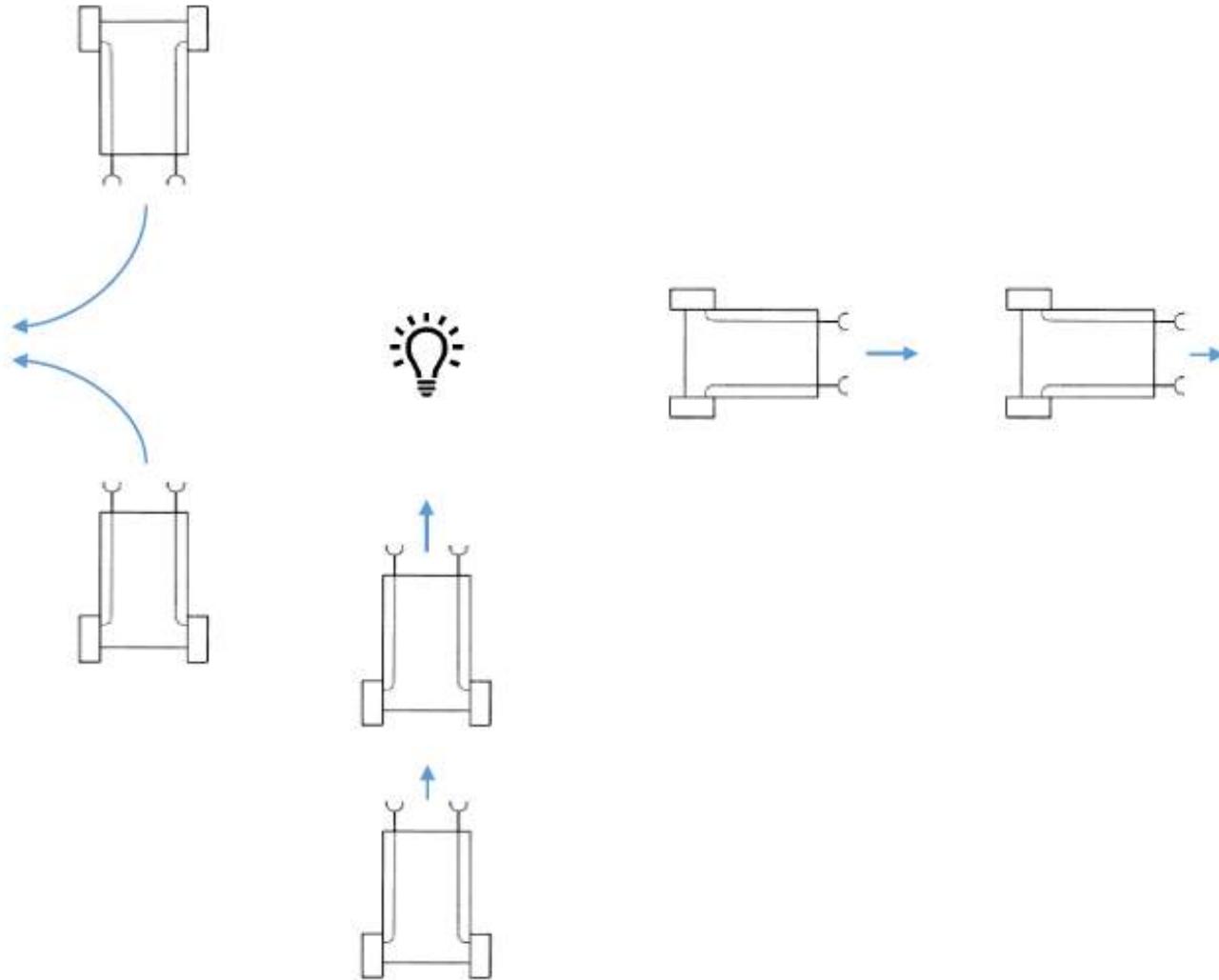
Under perfect conditions the vehicle stops on dark spots

But if more realistic physics are added e.g. non-symmetric drive or friction interesting things emerge such as:

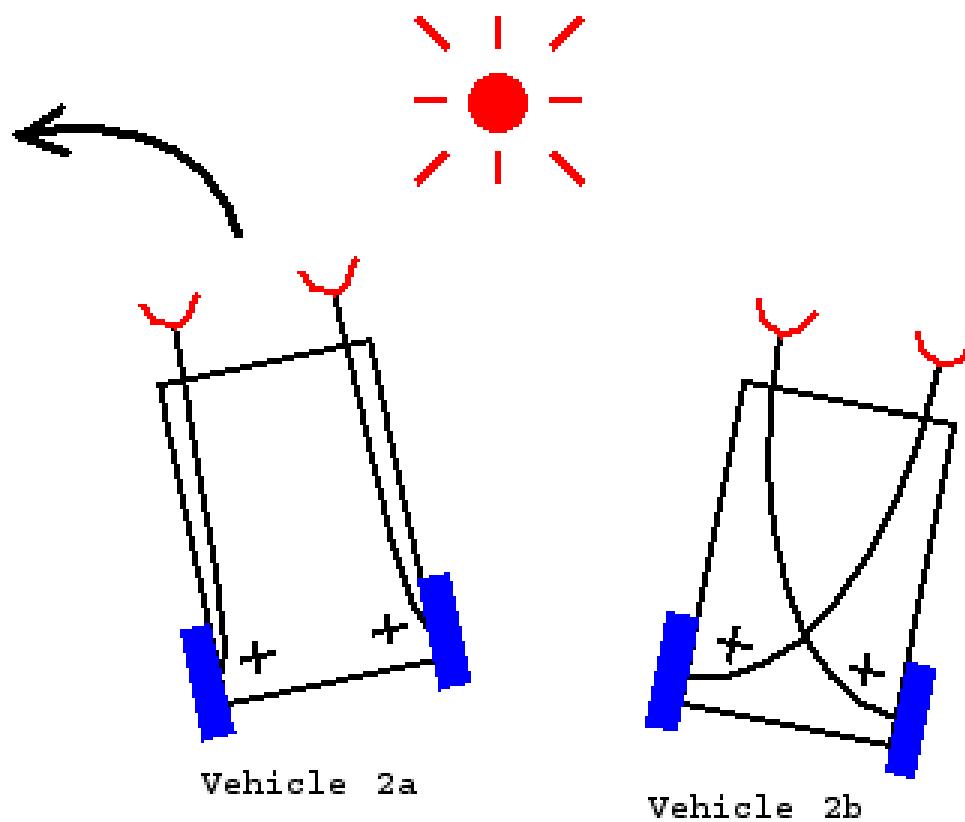
- looping continuously around sources
- Brownian motion on the group level

<http://stevebattle.github.io/braitenberg/vehicle1.html>

Vehicle 2a - What is the behaviour?

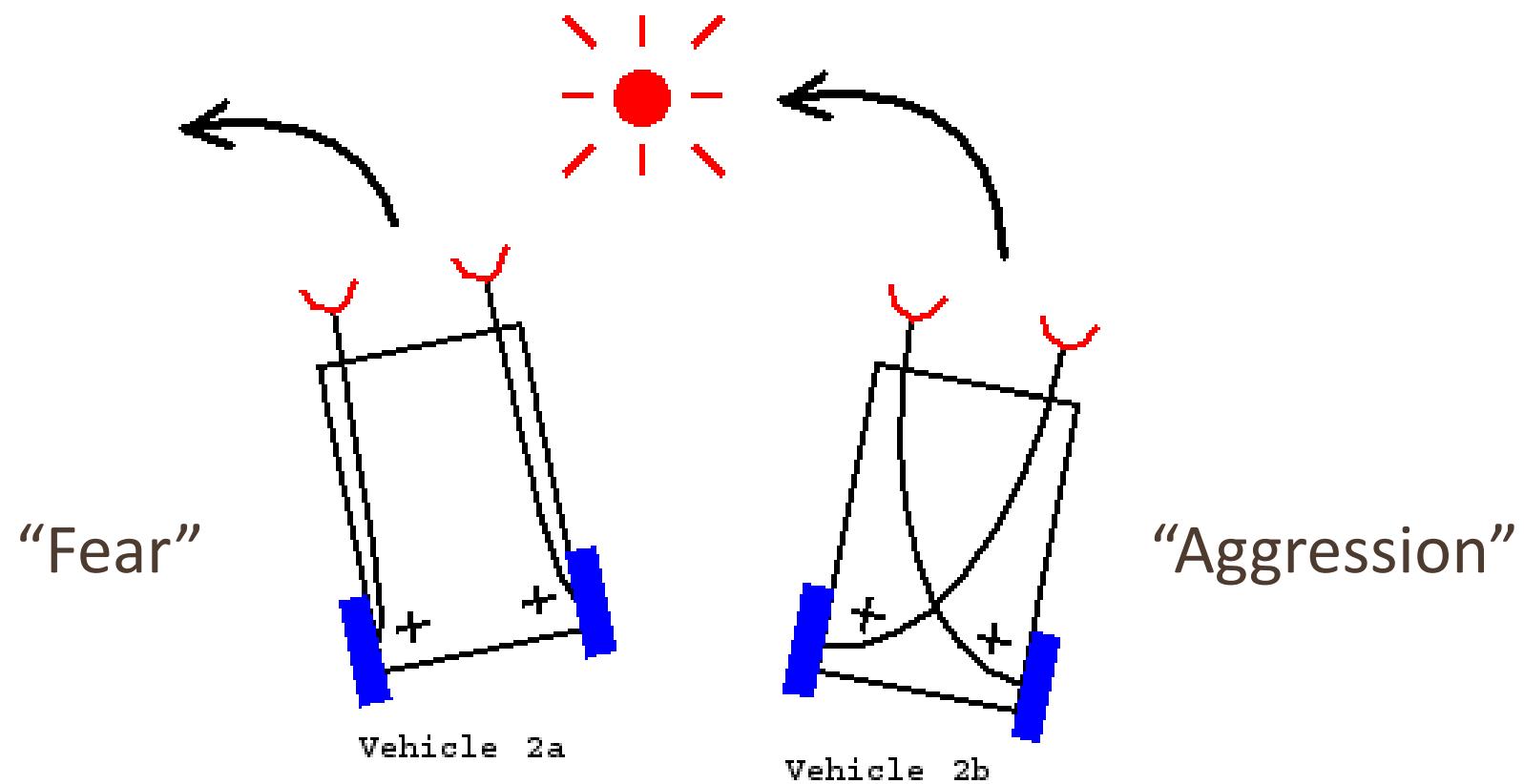


Vehicle 2b – ?



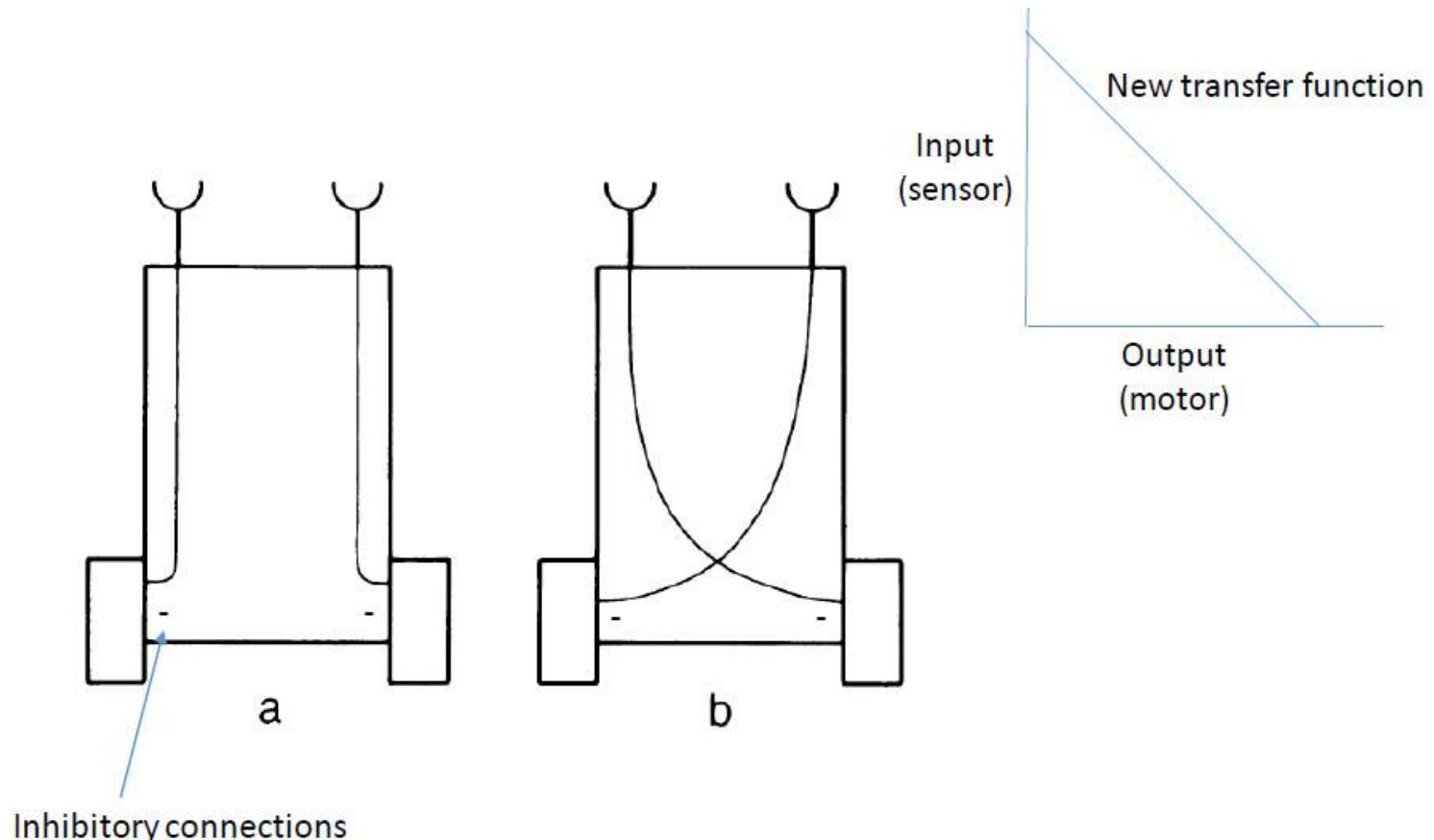
Excitatory connections (+)

Vehicle 2 – Fear and Aggression



Excitatory connections (+)

Vehicle 3 - ?



Vehicle 3 - Love

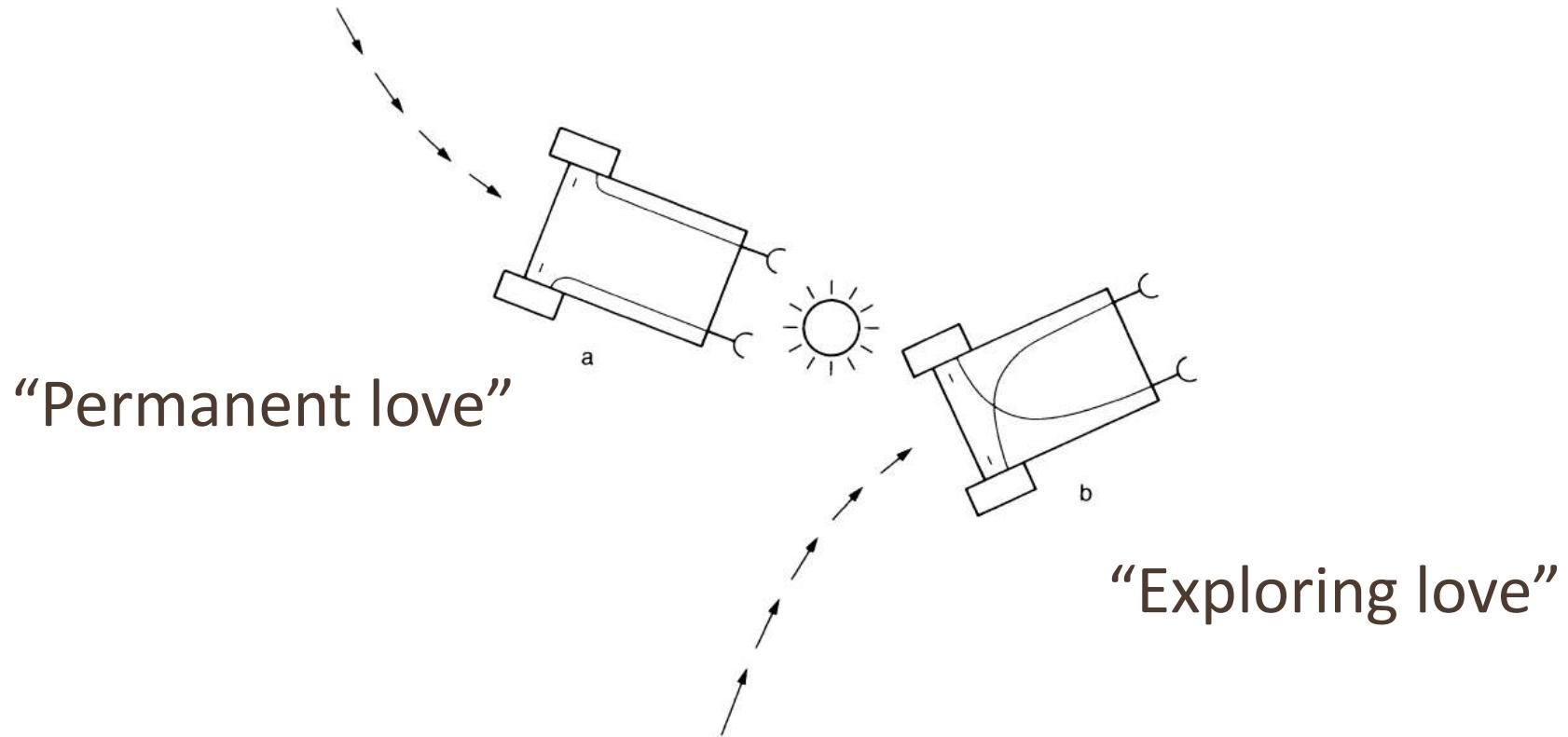


Figure 4

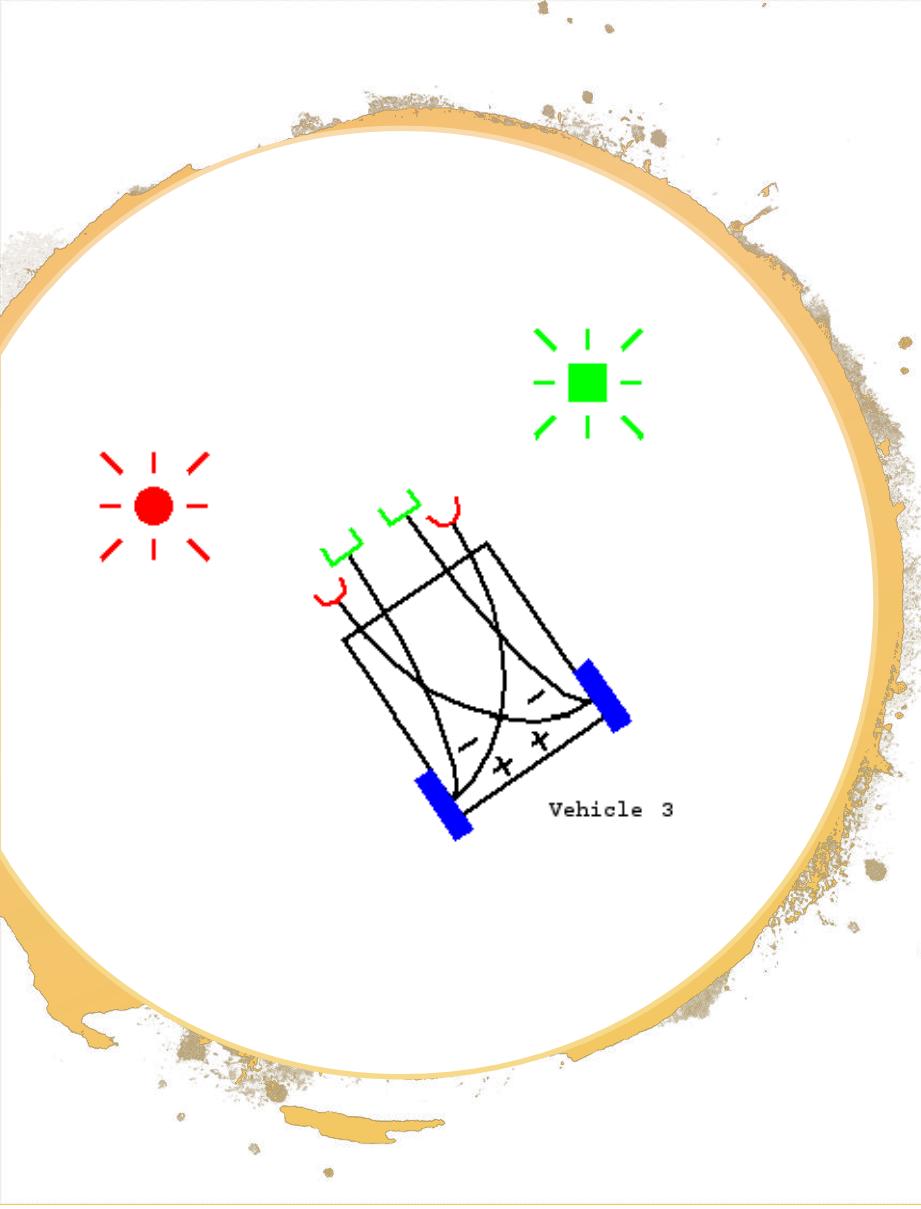
Vehicle 3, with inhibitory influence of the sensors on the motors.

Inhibitory connections (-)

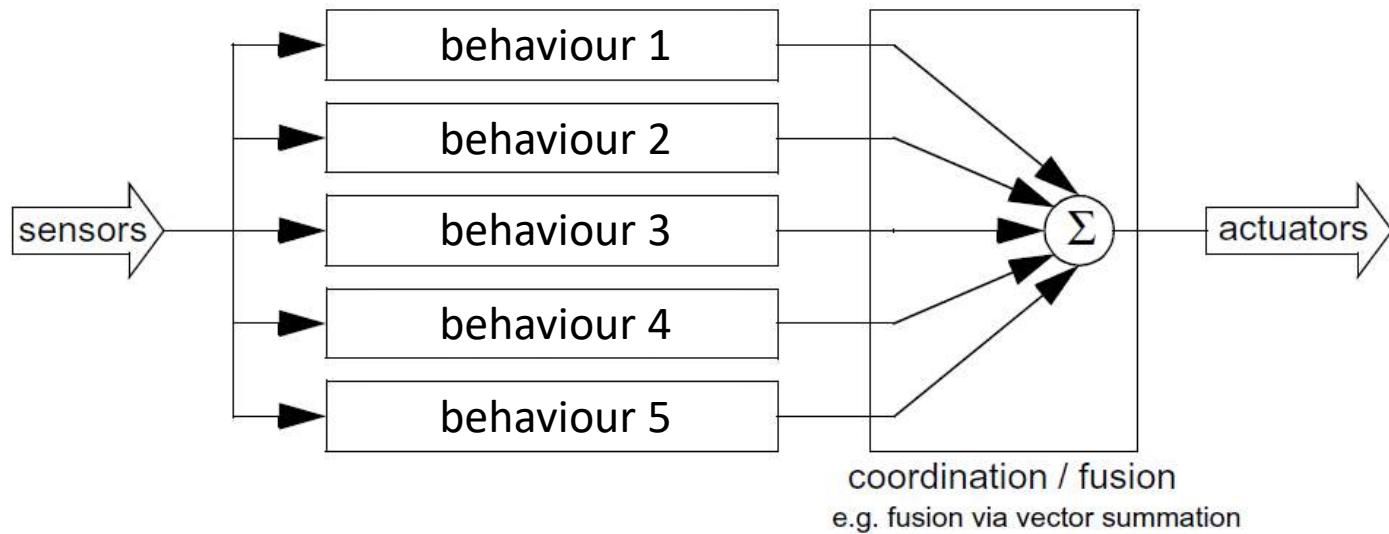
Multi-Sensory Version

Increasingly complex behaviours can be created by adding:

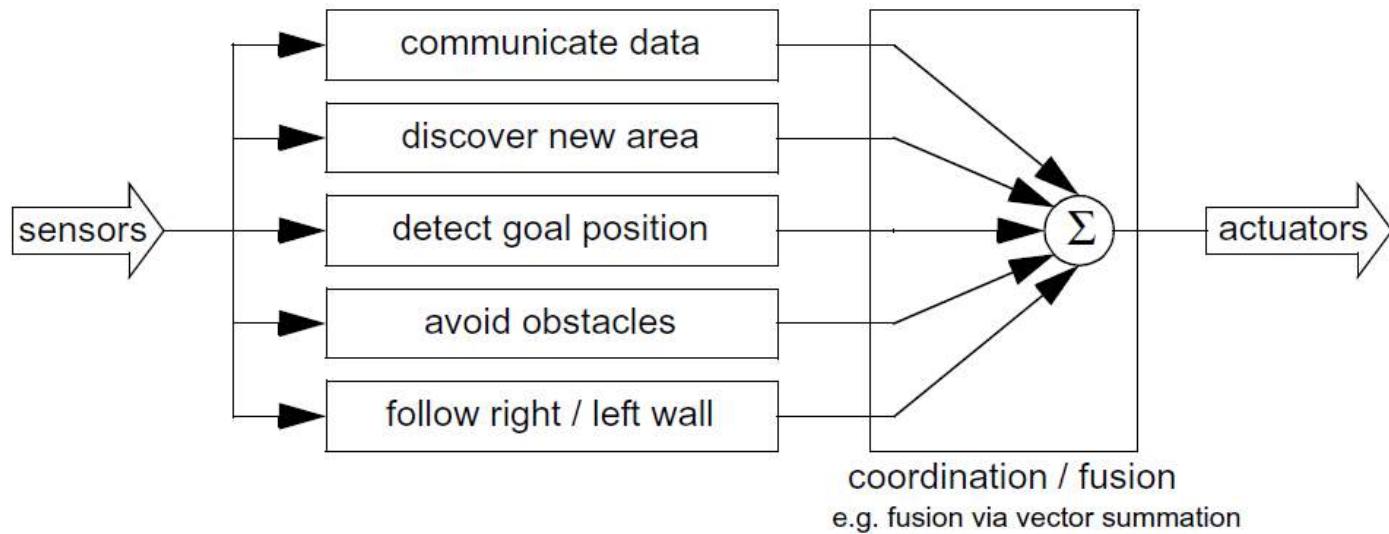
- New sensors with changing profiles
(non-linear / digital)
- New Transfer functions
- (Non-linear / weighting)
- Varying excitatory and inhibitory connectivity
- E.g. goal seeking vs obstacle avoidance



Multi-Sensory Version



Multi-Sensory Version



LEARNING ROBOT BEHAVIOURS

Neural network learning of behaviours

- Basic idea: reformulate the multi-sensory Braitenberg vehicle as a neural network!
 - Inputs: sensor responses $[x_i]$
 - Outputs: motor velocities $[y_j]$
 - Training: target outputs are provided by a human teacher with a joystick $[d_j]$
 - Delta rule:

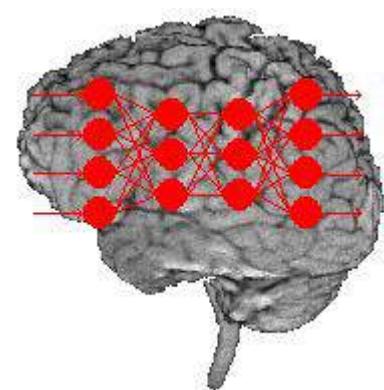
$$\Delta w_{ij} = \eta [d_j - y_j] x_i$$

w_{ij} - weight

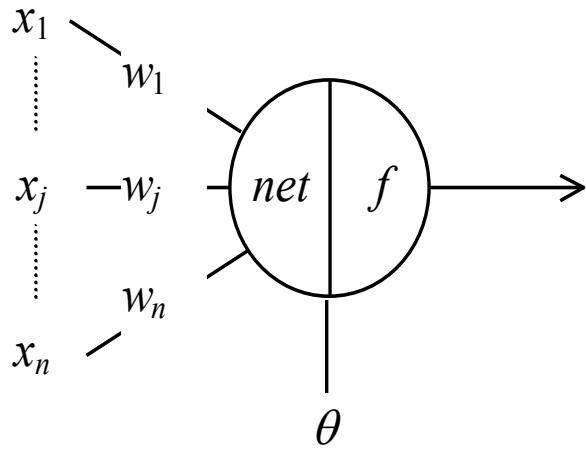
d_j - desired output,

y_j - actual output

x_i = input



Artificial neuron model



$$y = f\left(\sum_{j=1}^n w_j x_j - \theta\right) = f(\mathbf{w}^T \mathbf{x} - \theta) = f(\text{net})$$

Input vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

x_j = Input signal j .

w_j = Weight for the j^{th} input signal.

θ = Threshold/Bias of the neuron.

net = Summation function.

f = Activation function.

(e.g. **+1** is $\text{net} \geq 0$, or **-1** if $\text{net} < 0$)

Weight vector $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$

Robot Behaviour Learning

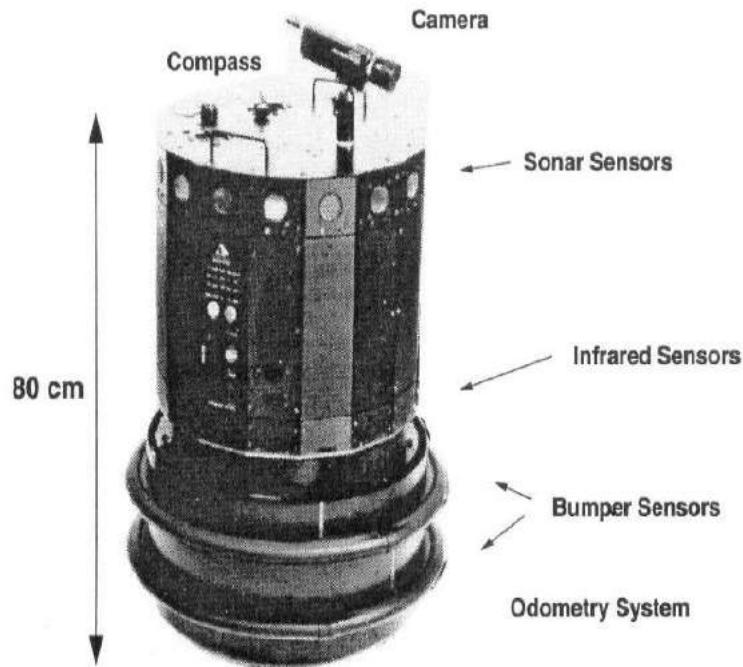


FIG. 3.17. THE NOMAD 200 MOBILE ROBOT *FortyTwo*

Learned behaviours included:

- Obstacle avoidance
- Wall following
- Box pushing

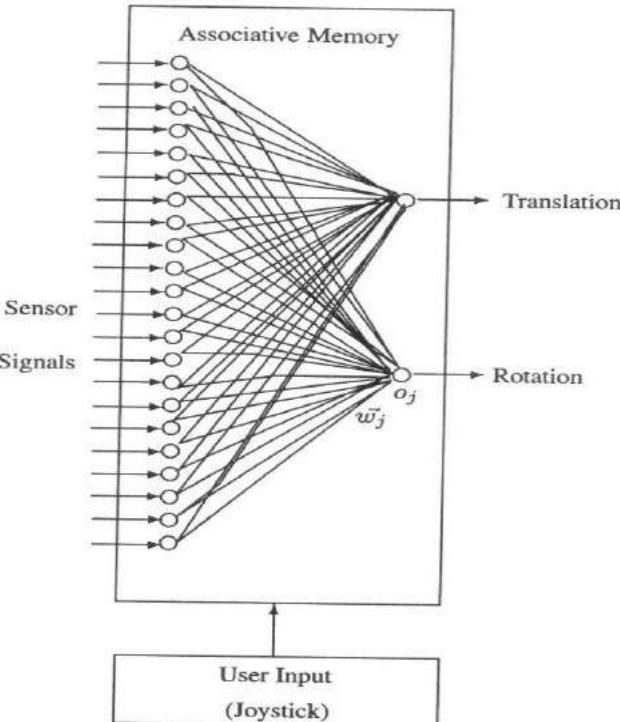


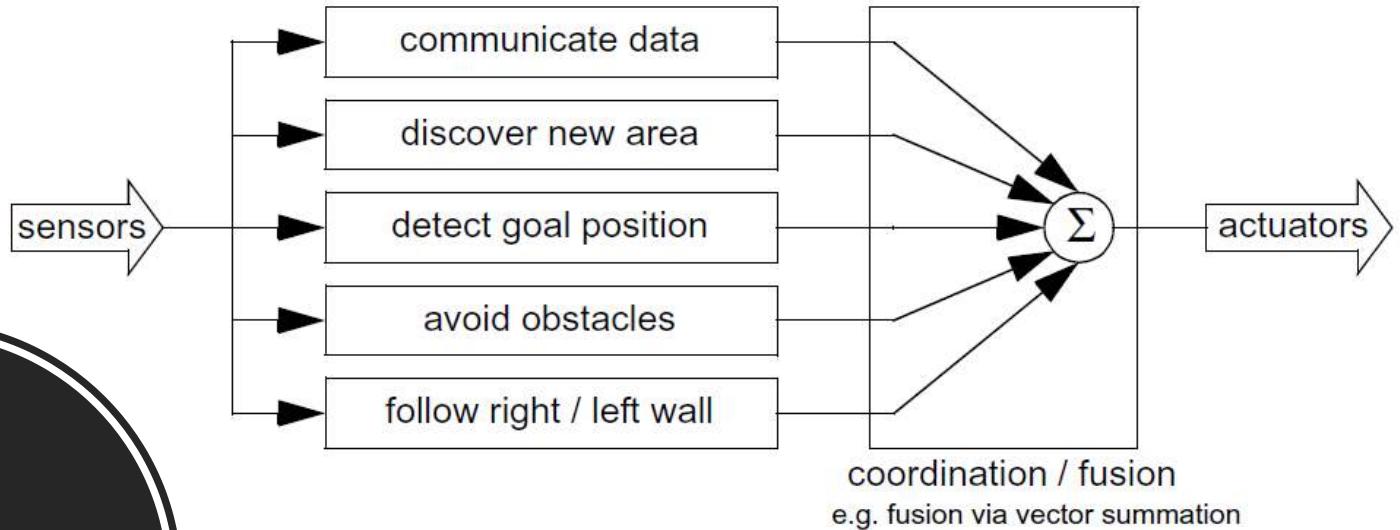
FIG. 4.15. THE CONTROLLER ARCHITECTURE

5 values	5 values	6 values	6 values
Left facing Sonars	Right facing Sonars	Left facing IRs	Right facing IRs

FIG. 4.16. THE INPUT VECTOR USED

from Nehmzow (2002)

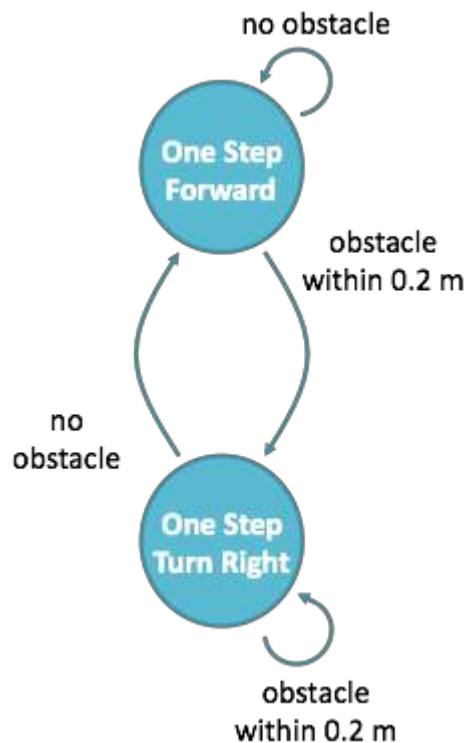
COMBINING ROBOT BEHAVIOURS



- May be implemented quickly
- Does not directly scale to other environments
- Underlying procedures must be carefully designed to produce the desired behaviour
- Sequencing - How to coordinate the order of behaviours in time?
- Concurrent behaviours - How to coordinate behaviours at the same time?

Sequencing

- Finite State Machine (FSM)
 - models behaviour of a system
 - why “finite”?
- A discrete number of states
 - behaviours in our case
- Conditional state transitions
 - based on perceptions
- Actions are associated with states
 - enter, exit, in-state



Simple obstacle avoidance using IR sensor

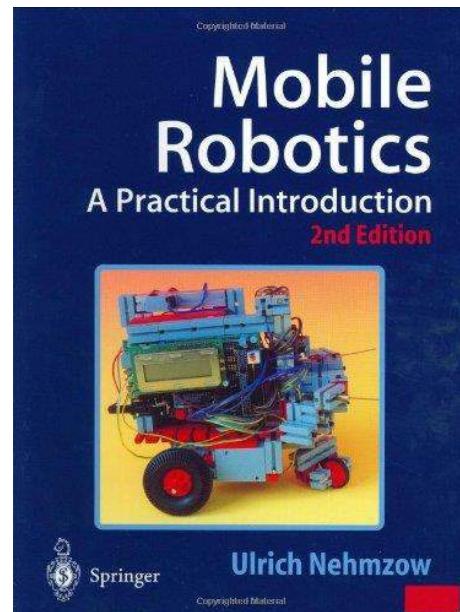
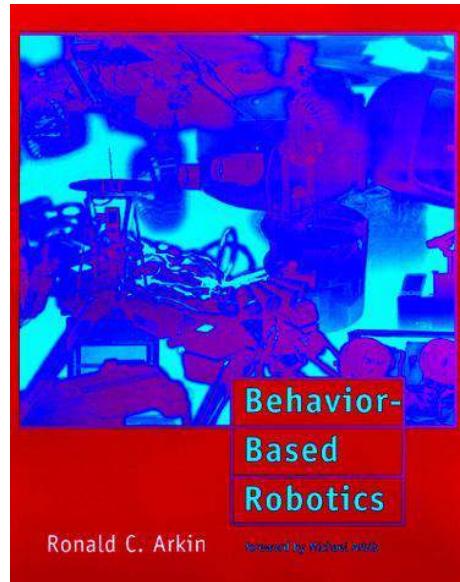
Finite State Machine

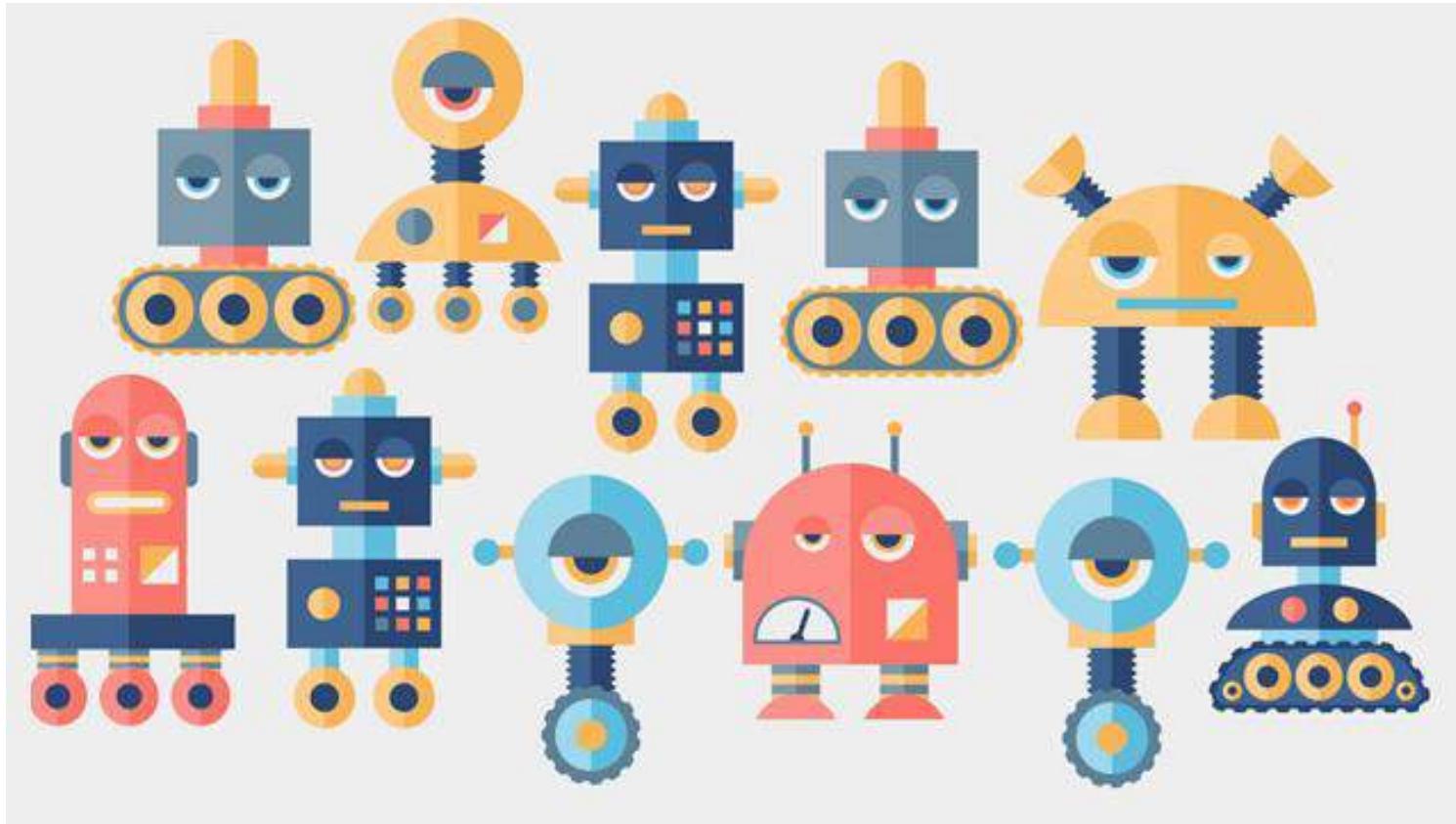
- State transition table
- Only some transitions may be valid
- Challenges:
 - avoid dead ends
 - avoid “swing states”

Current state → Input ↓	State A	State B	State C
Input X
Input Y	...	State C	...
Input Z

Recommended reading

- Ronald C. Arkin, “Behavior-based Robotics”, MIT Press, 1998.
- Ulrich Nehmzow, “Mobile Robotics: A Practical Introduction”, Springer, 2nd edition, 2002.





Thank you for listening!

Any questions ?

CMP3103M/CMP9050M

Autonomous Mobile Robotics

Dr Ayse Kucukyilmaz

University of Lincoln
Centre for Autonomous Systems

INB3201

akucukyilmaz@lincoln.ac.uk
<http://webpages.lincoln.ac.uk/akucukyilmaz/>



UNIVERSITY OF
LINCOLN



Syllabus

1. Introduction (MH)
2. Robot Programming (MH)
3. Robot Sensing (MH)
4. Motion and Control (MH)
5. Robot Behaviours and **Navigation (AK)**
6. Navigation (AK)
7. Mapping (AK)
8. Self-Localisation (AK)
9. Control Architectures (PB)
10. Human Robot Interaction 1 (PB)
11. Human Robot Interaction 2 (PB)
12. Applications (PB)



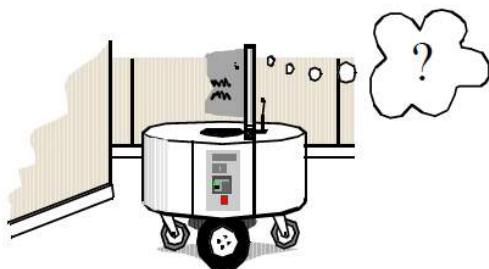
NAVIGATION

NOUN

the process of determining and maintaining a path or trajectory* to a goal destination

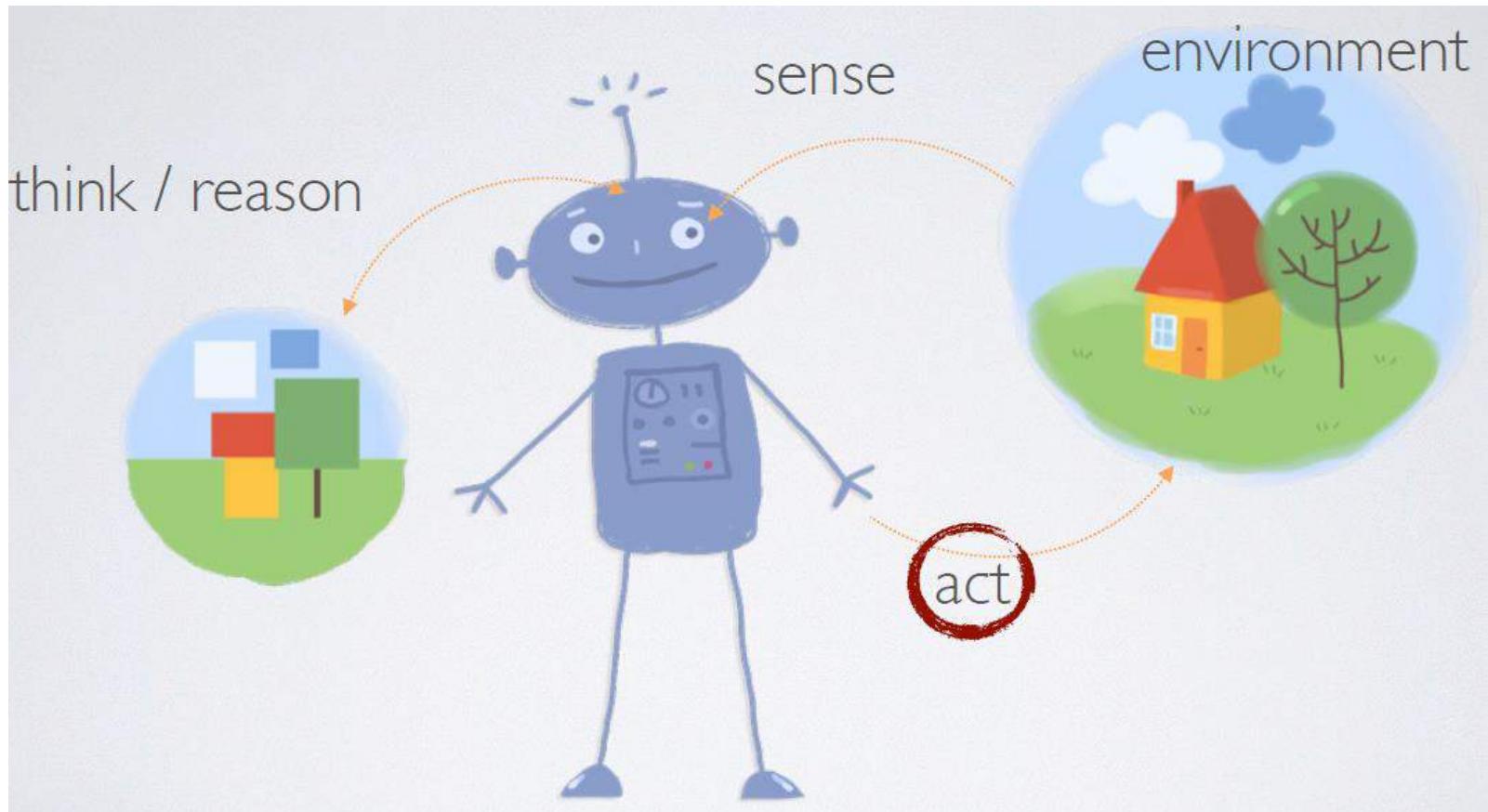
Trajectory: time history of position, velocity, and acceleration for each degree of freedom.

Key questions

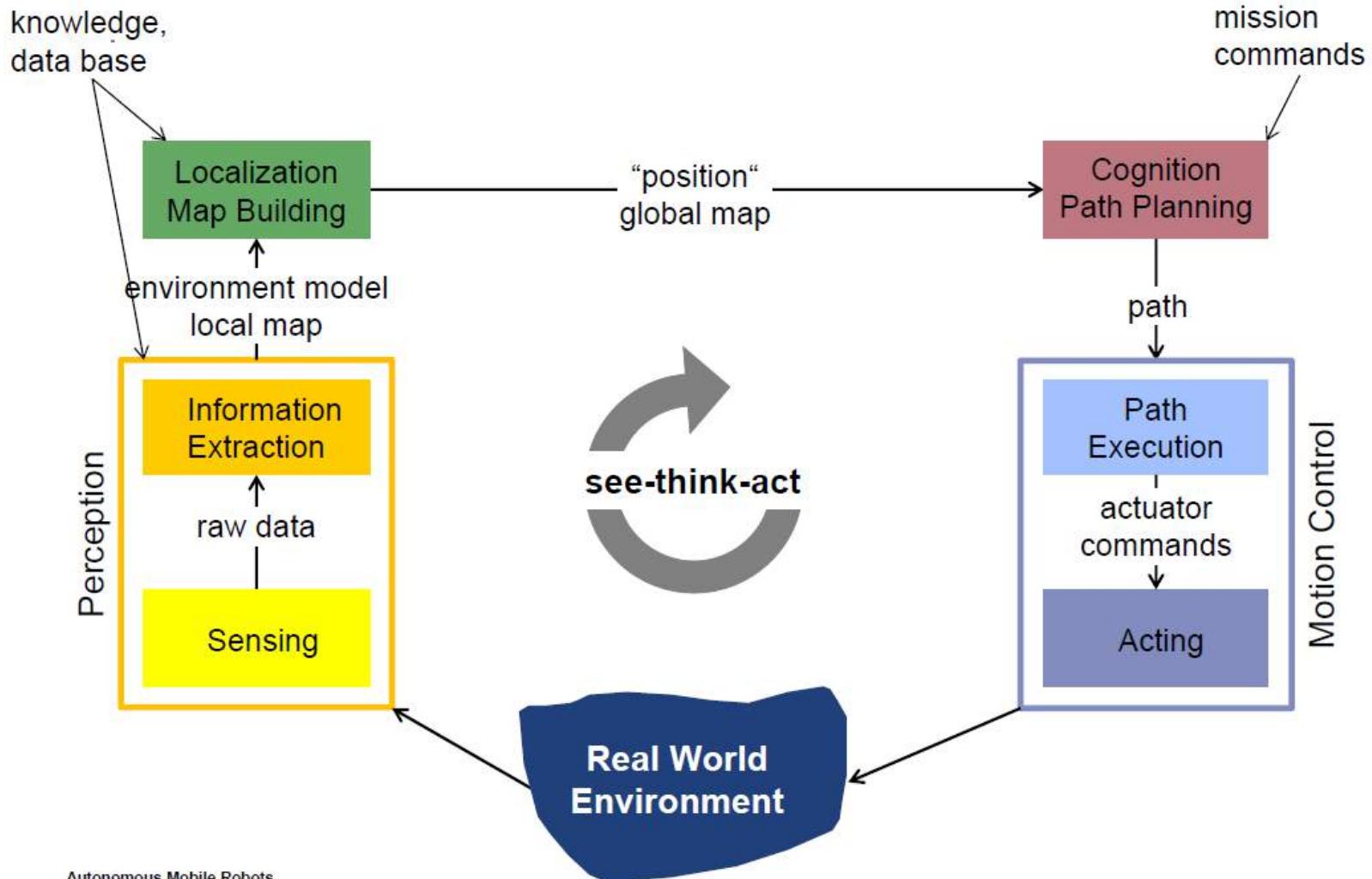


- Where am I?
- Where do I go?
- How do I go there?
- To answer these, the robot needs
 - A model of the environment (either given or autonomously built)
 - To perceive and understand the environment
 - Find its position and situation in the environment
 - Plan and execute the movement

Sense / Think / Act Cycle

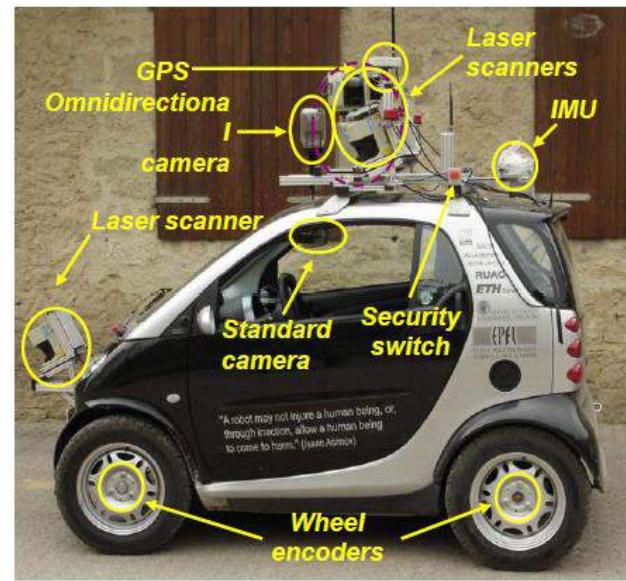
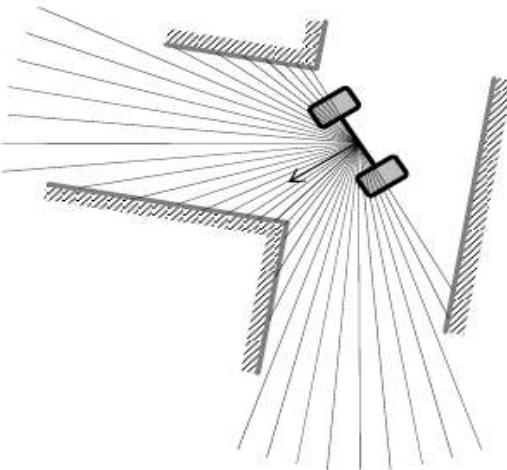
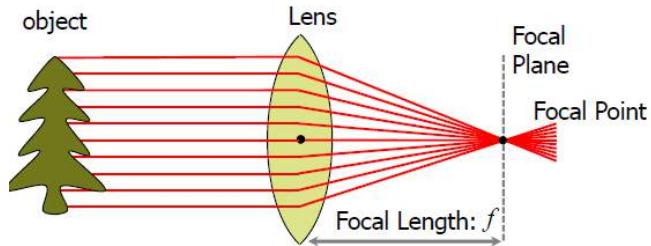


Sense / Plan / Act Cycle



Perception

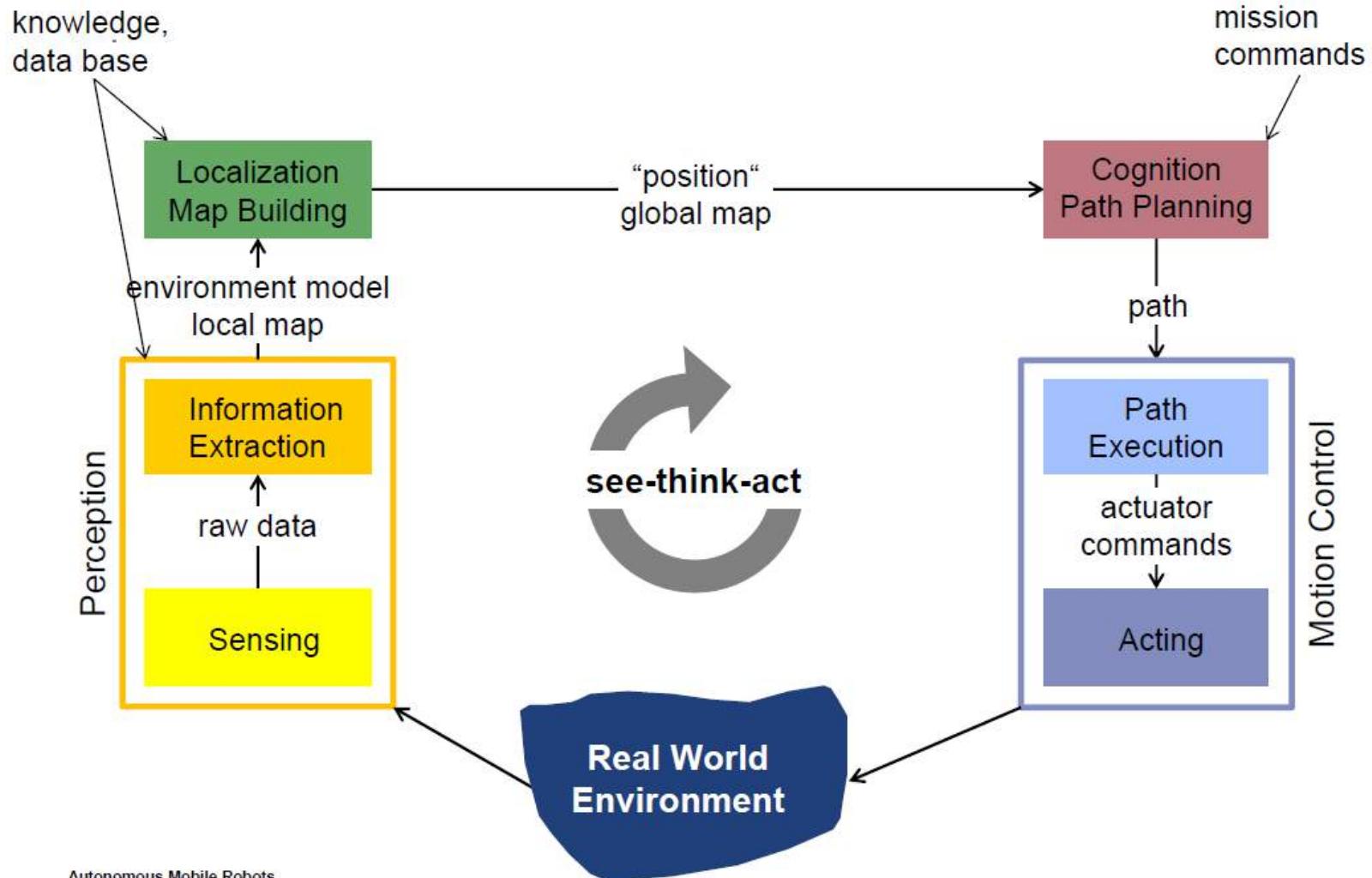
Sensing



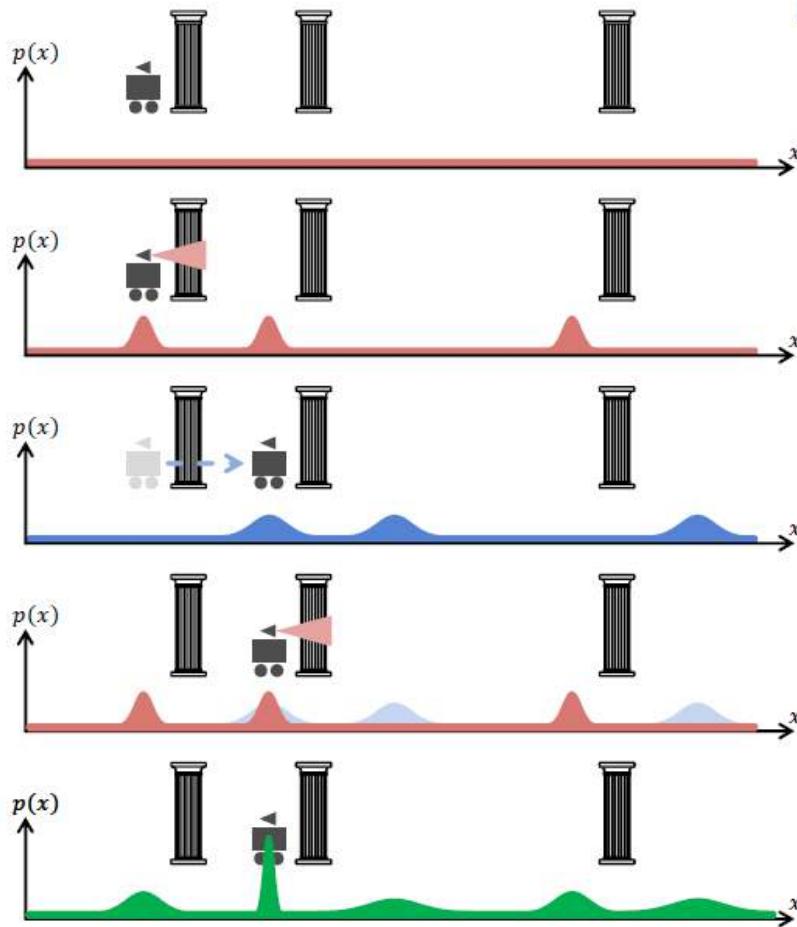
Information extraction



Sense / Plan / Act Cycle

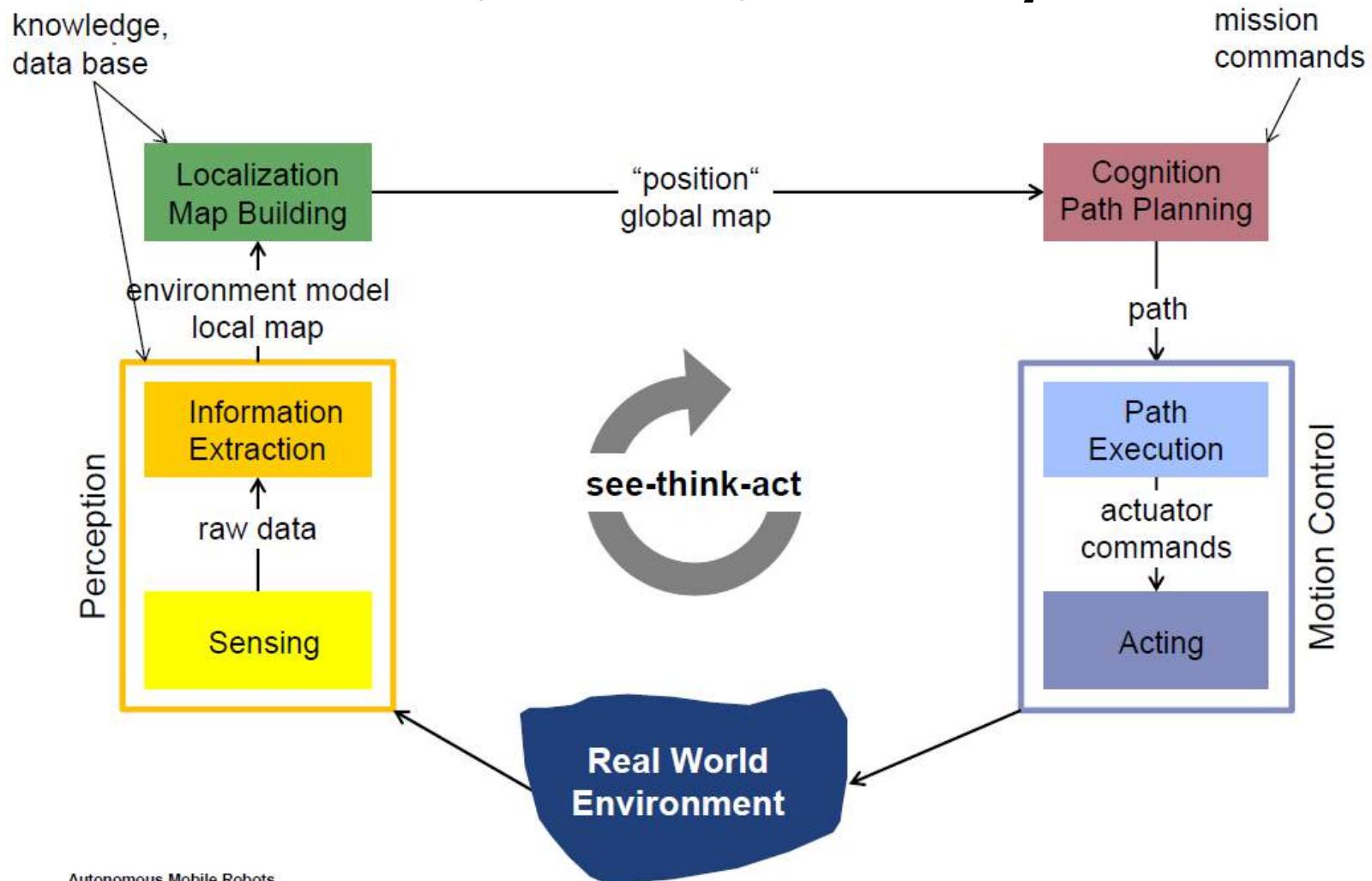


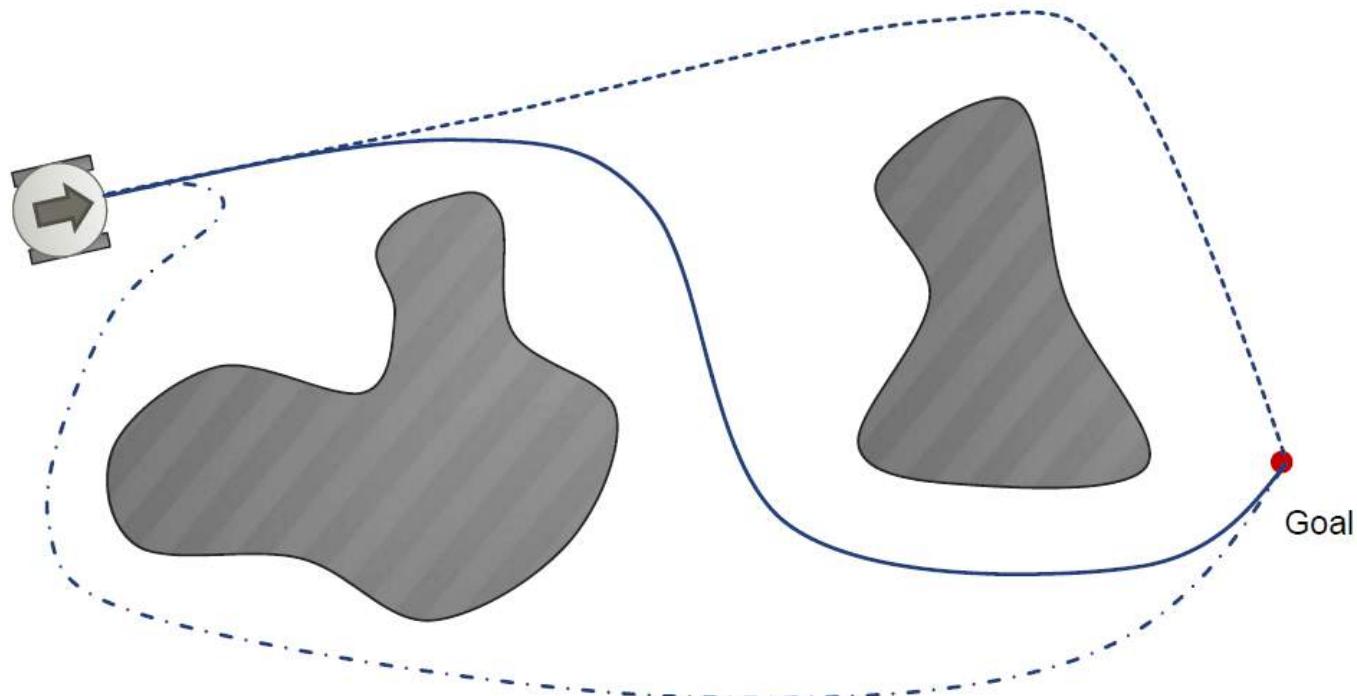
Localisation: where am I?



- SENSE: The robot queries its sensors
 - finds itself next to a pillar
- ACT: Robot moves one meter forward
 - Motion estimated by wheel encoder
 - Accumulation of uncertainty
- SENSE: The robot queries its sensors again
 - finds itself next to a pillar
- Belief update (information fusion)

Sense / Plan / Act Cycle





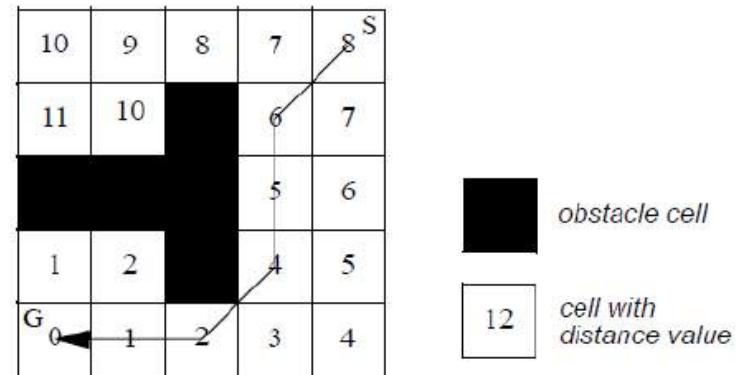
Cognition

Where do I go?
How do I go there?

Cognition

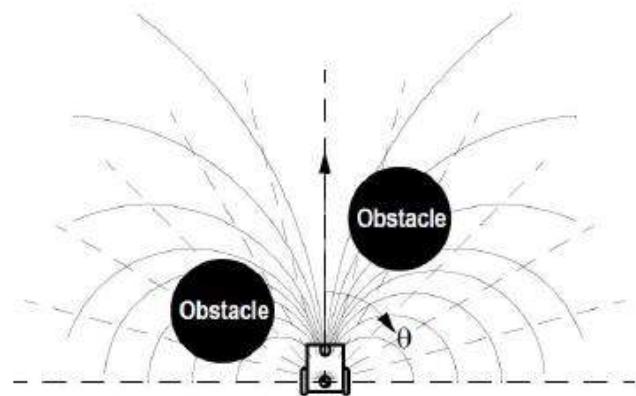
Global path planning

- Graph search

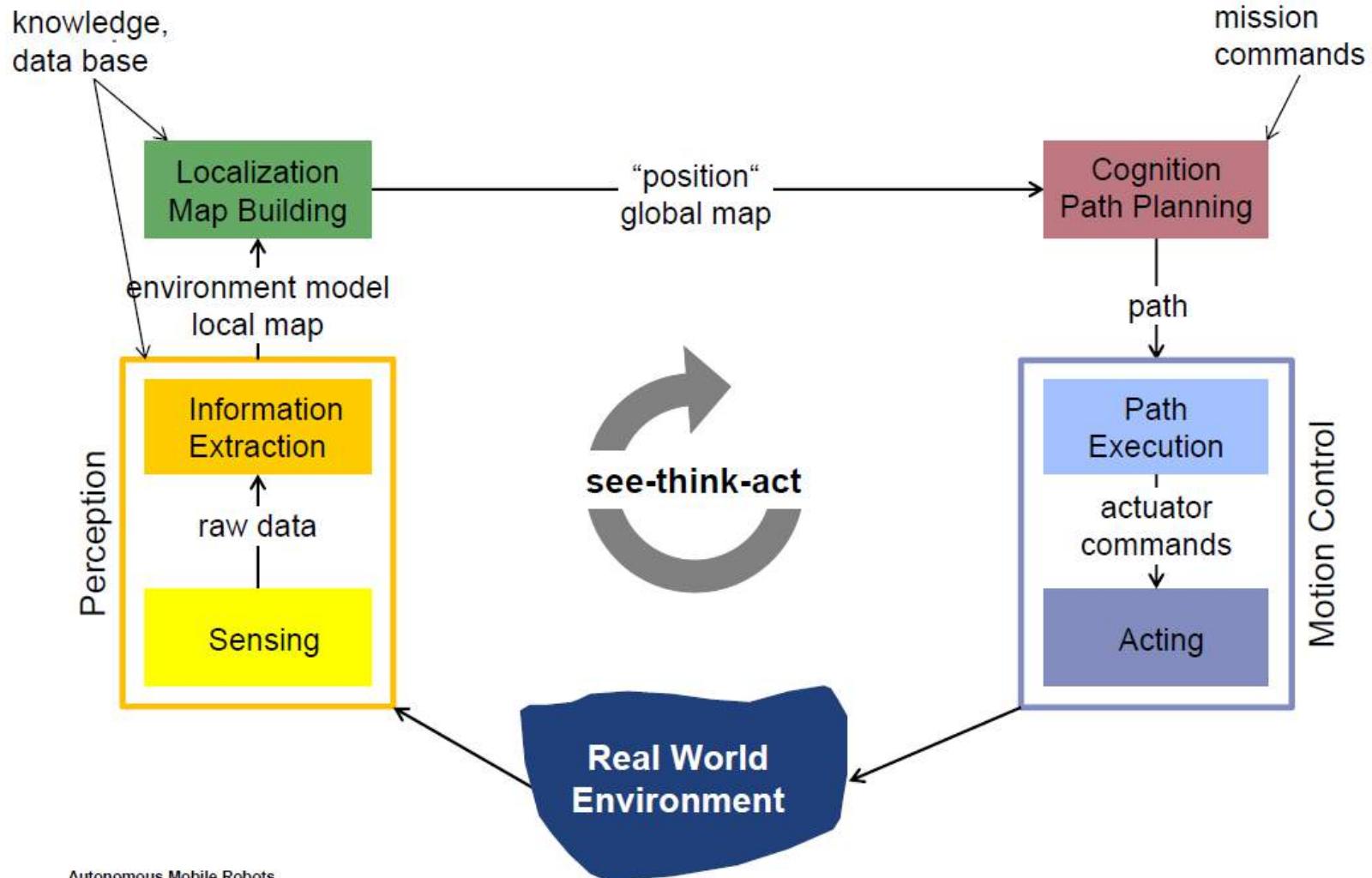


Local path planning

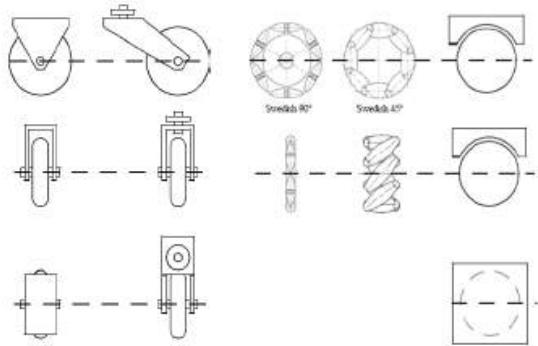
- Collision avoidance



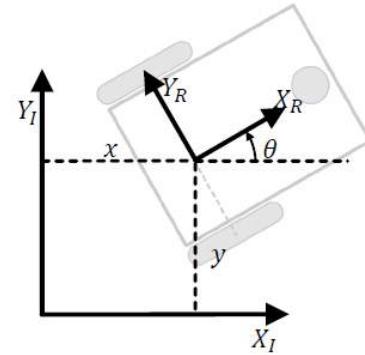
Sense / Plan / Act Cycle



Wheel types and constraints



Motion control

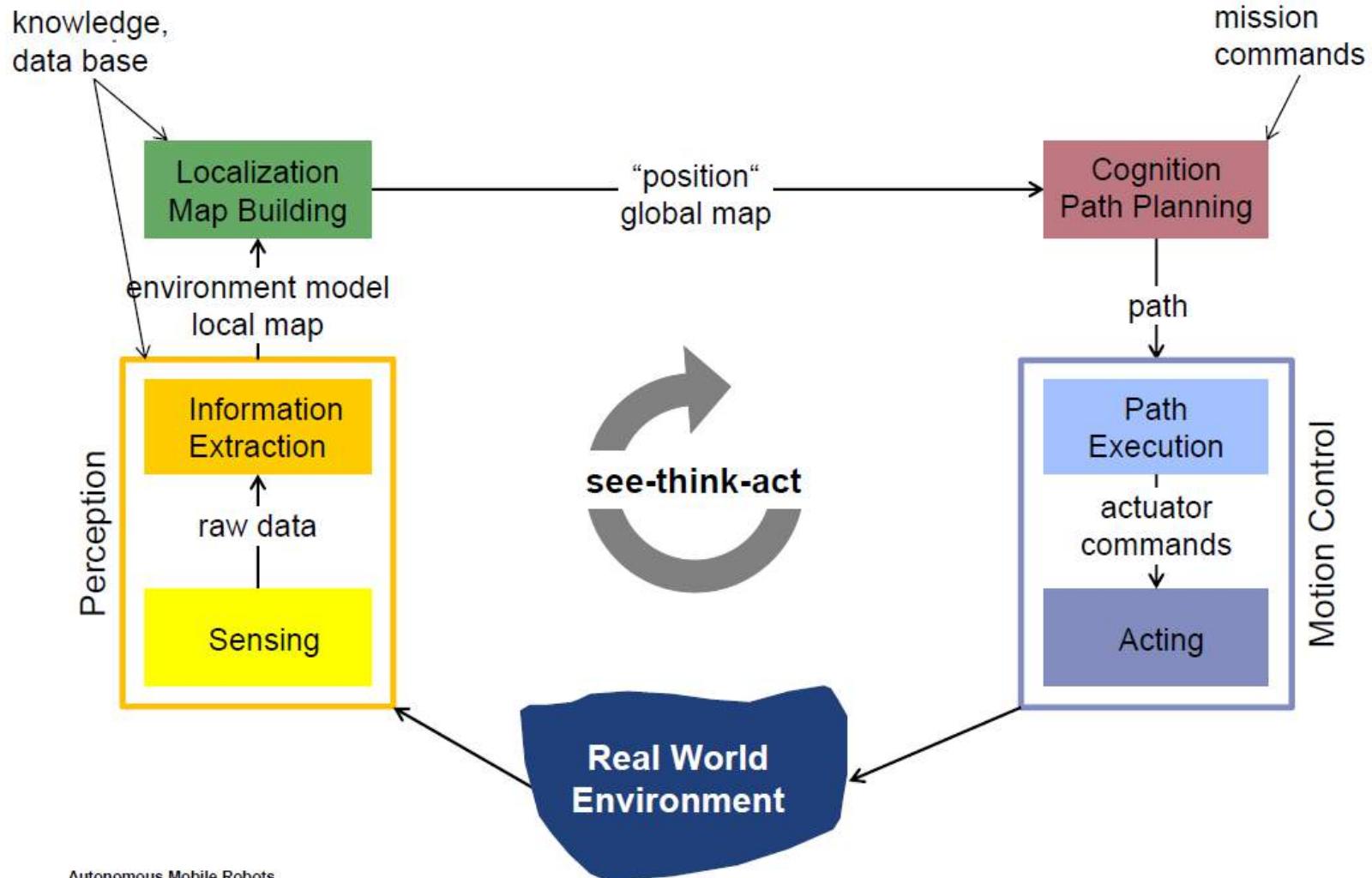


$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\dot{\phi}_1 \dots \dot{\phi}_n, \theta, \text{geometry})$$

$$\begin{bmatrix} \dot{\phi}_1 \\ \vdots \\ \dot{\phi}_n \end{bmatrix} = f(\dot{x}, \dot{y}, \dot{\theta}) \quad ?$$

Motion Control

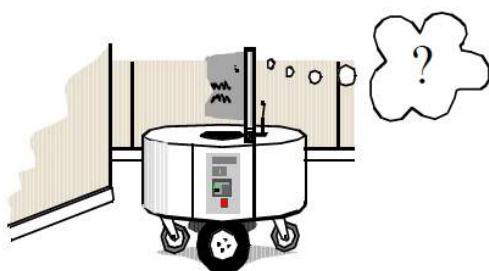
Sense / Plan / Act Cycle



NAVIGATION

by local cues

Definitions



- Global navigation
 - Robot is not told its initial position
 - Position should be estimated from scratch
- Position tracking (continuous localization)
 - Robot knows initial position
 - It has to accommodate small errors in odometry as it moves

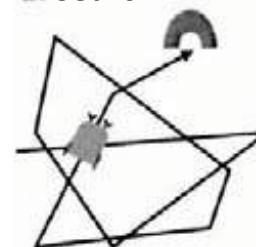
Types of Navigation

- **Global strategies** ("way-finding", requires a map!)
 - **Recognition-triggered** response: local navigation triggered by last recognized place
 - **Topological** route: based on topological maps (graphs where places are nodes and edges routes connecting them), no geometric information
 - **Survey**: all known places and spatial relations embedded in the same frame of reference, can discover new paths

Types of Navigation

- **Local strategies:**
 - **Search**: can move, and can recognise arrival at the goal
 - **Direction following**: e.g. compass direction, or trail following: find goal from one direction (e.g. Salmon swimming upstream)
 - **Aiming**: e.g. taxis to source, using landmarks: can find a salient goal from a catchment area
 - **Guidance**: information by surroundings

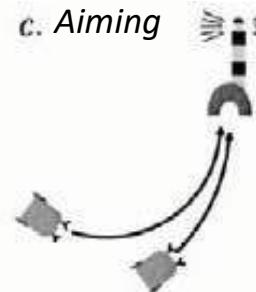
a. Search



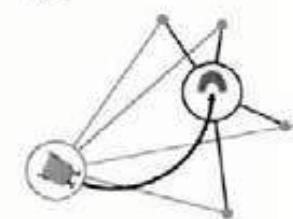
b. Direction following



c. Aiming



d. Guidance



(Franz & Mallot, 2000)

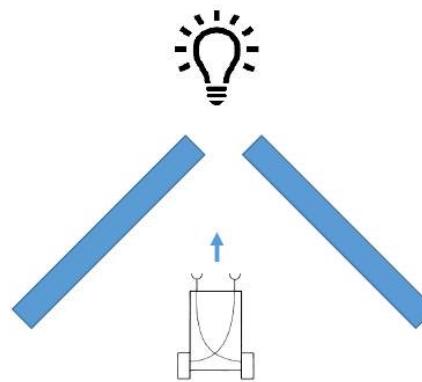
Local strategies using vision

- Can use a visual cue at the goal reactively, i.e. detect the goal from a distance and maintain a course towards it
 - = landmark navigation / beaconing
- Can use a visual cue in the simplest form of navigation
 - i.e. remember visual surroundings to recognise when you arrive at the goal
- More generally, may be able to use landmarks around the goal to determine the course towards it
 - = visual homing



Beaconing problems

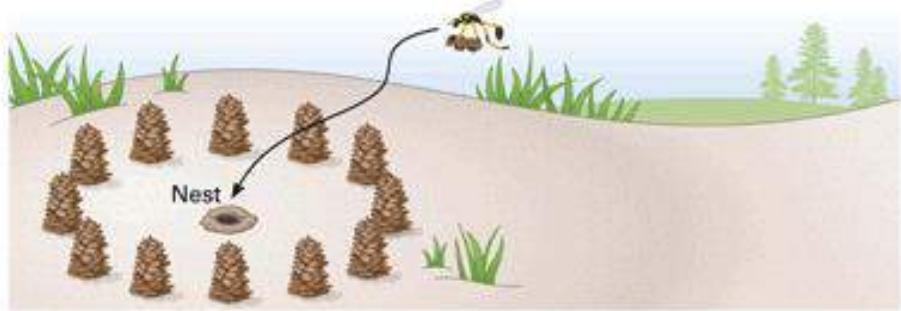
- Cues must be visible
 - Temporarily obscured
 - Short range
- Where complex planning is needed



Visual Homing in Insects

- Nico Tinbergen showed in 1920s that wasps approach their hidden nests using surrounding visual cues, i.e. the pattern of landmarks around their nests.
- Similar behaviours are found in many animals:
 - Bees
 - Ants
 - Humming birds
 - Rats

1 A digger wasp carries food (a paralyzed bee) back to her nest.



2 When the landmarks are moved, the wasp follows the landmarks rather than returning to the nest site.

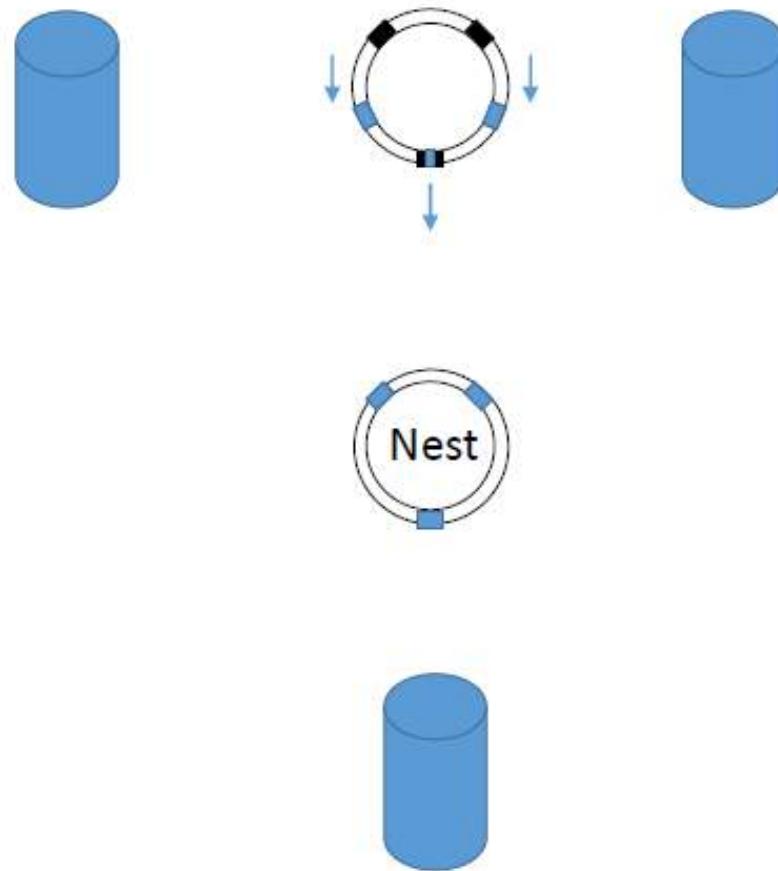


3 Changing the landmarks reveals that the wasp responds to the arrangement of the landmarks, rather than the type of landmark.



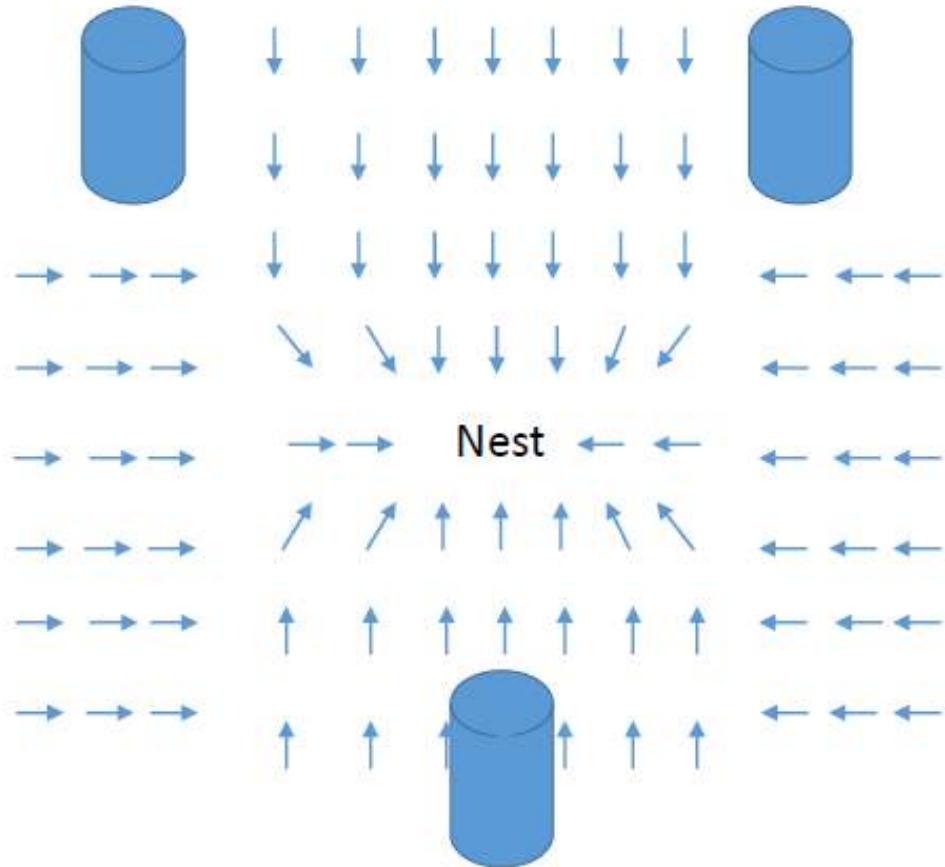
Visual Homing Snapshot Model

- Before leaving home take a visual “snapshot”
- When homing
 - Move to reduce mismatch between current view and initial snapshot
 - Repeat until home is reached



Visual Homing Snapshot Model

- Before leaving home take a visual “snapshot”
- When homing
 - Move to reduce mismatch between current view sand initial snapshot
 - Repeat until home is reached



Gradient Descent

Problems of Snapshot Model

1

How to define a
landmark?

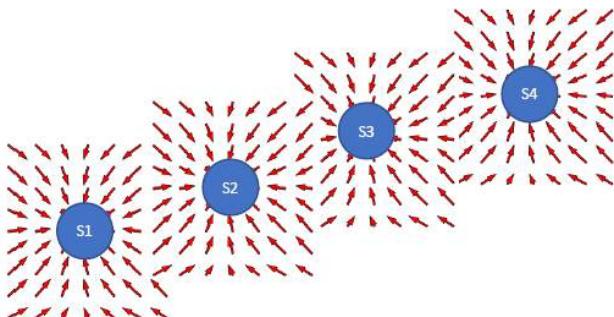
2

How to match
landmarks between
current scene and
memory?
(correspondence
problem)

3

You must be
aligned

Chaining visual snapshots into routes



Learning:

- Link visual snapshots by trajectory (e.g. local vectors)

Navigation:

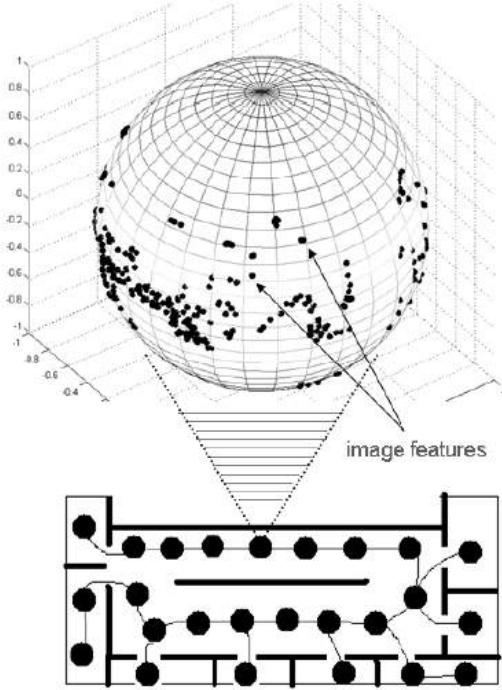
- Load appropriate movement
- Follow until new snapshot catchment area reached
- Use visual homing to pinpoint location
- Trigger new movement
- Repeat

- Sounds simple but is tricky in practice!
- **How do I know where I am on the route?**

Place Recognition on Routes

- Recognising where you are with respect to previous memories is a big problem in robotics at present.
- **Loop closure**: the ability to recognise a location even if perceived from a different pose. [In SLAM (more in next weeks lecture) this allows correction of the robot positioning using the mapping thread.]
- Commonly described in terms of the **kidnapped robot problem**



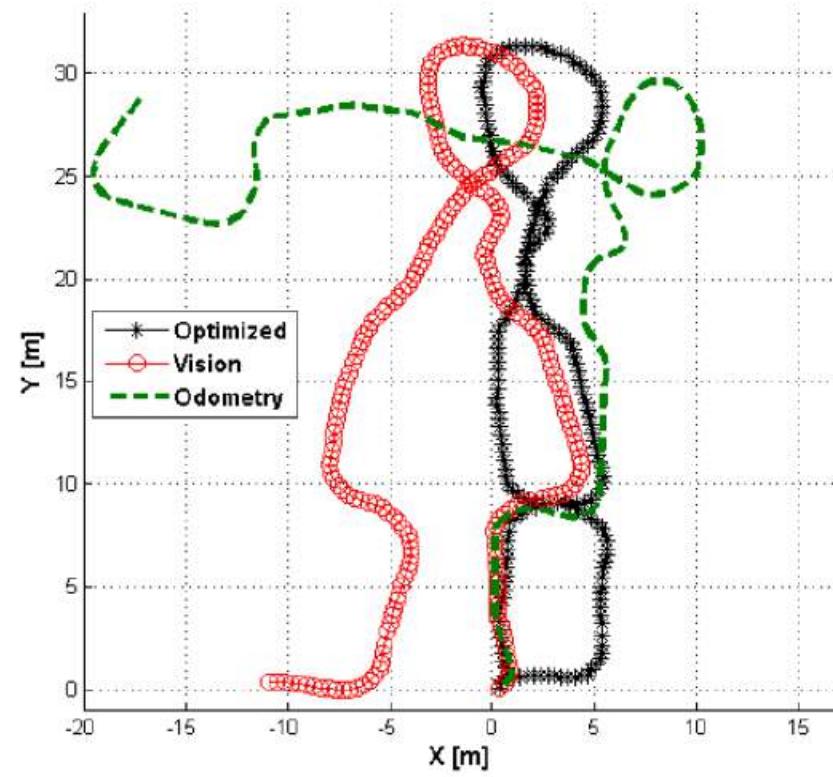
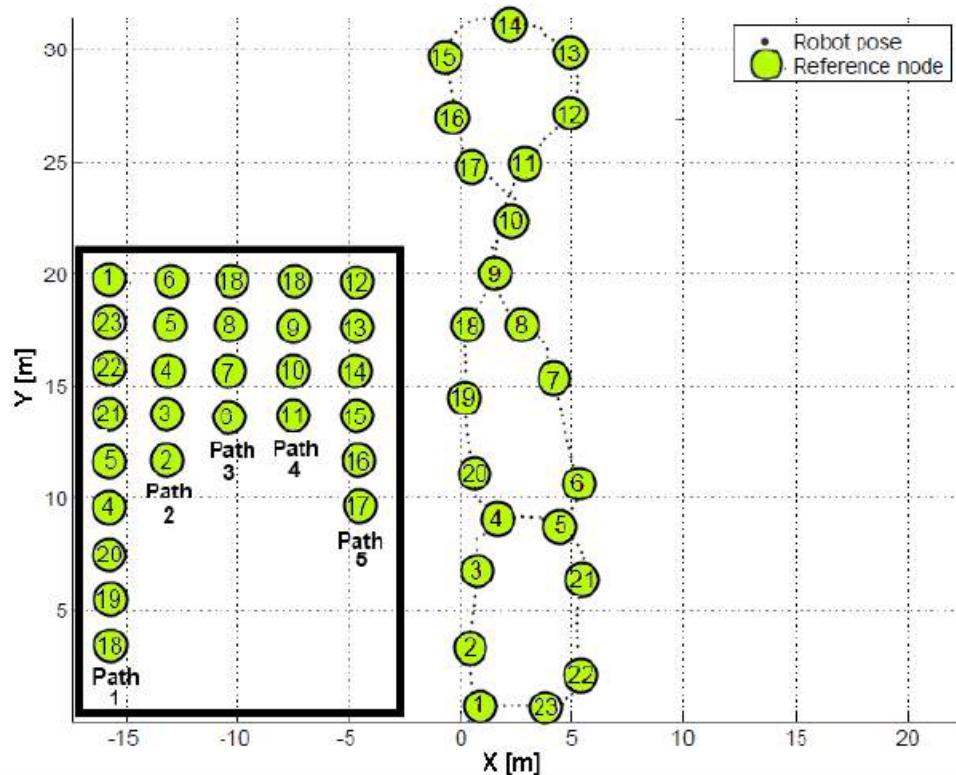


Feature Based Place Recognition

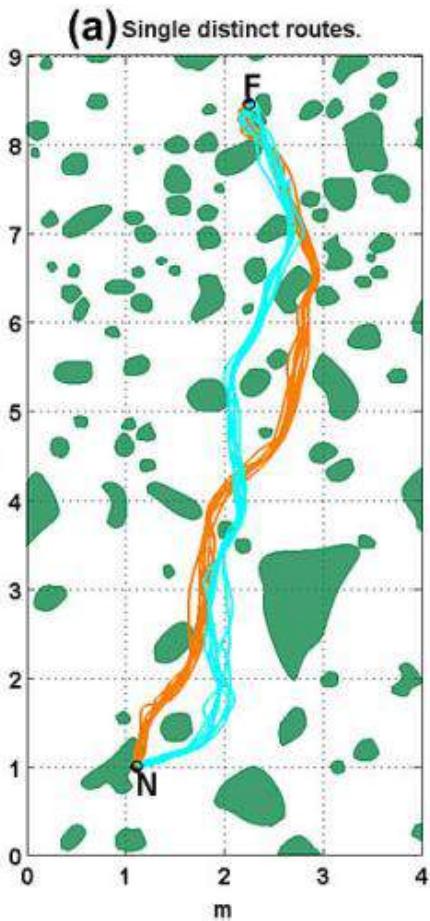
- Visual Features (e.g. SIFT, SURF) can be used describe a location.
- Matching the features and descriptors currently visible with a list of those at previously visited locations can be used for place recognition and localisation.
- Common method is called “bag of words”



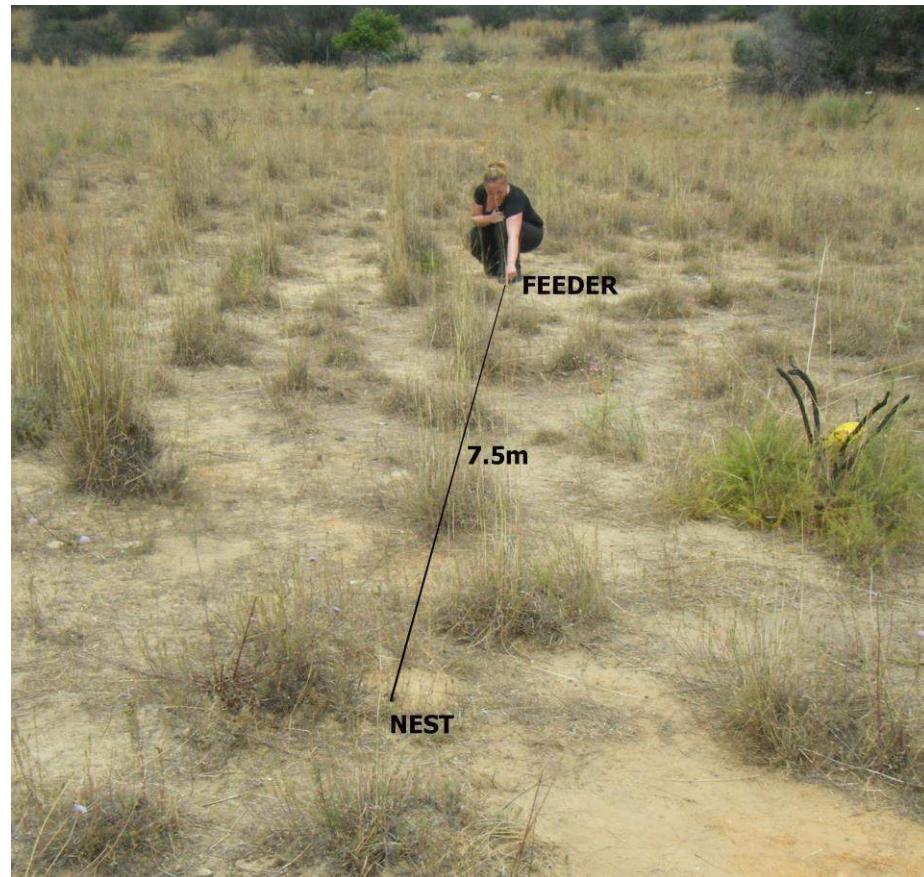
Feature Based Place Recognition



Visual Route Following in Desert Ants

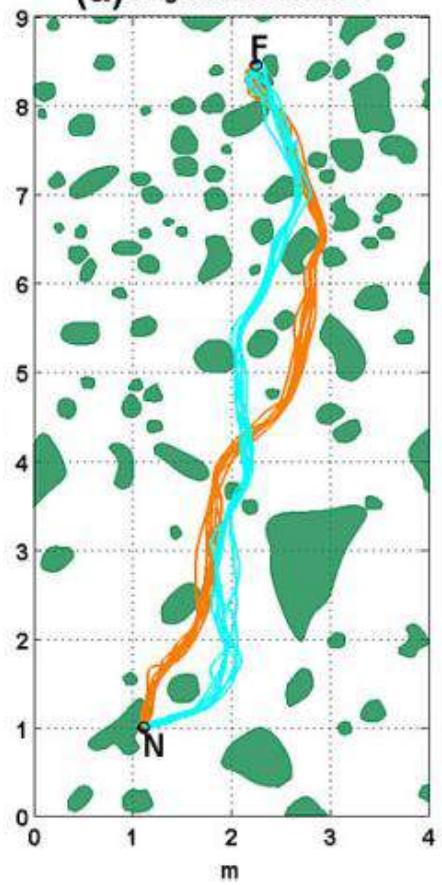


Mangan & Webb, 2012



Route Following

(a) Single distinct routes.



Mangan & Webb, 2012

FEEDER

7.5m

NEST

The Visual Compass



Route Memory

The Visual Compass



Route Memory

Current View

Error

Angle

The Visual Compass

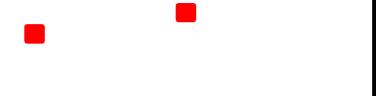


Route Memory

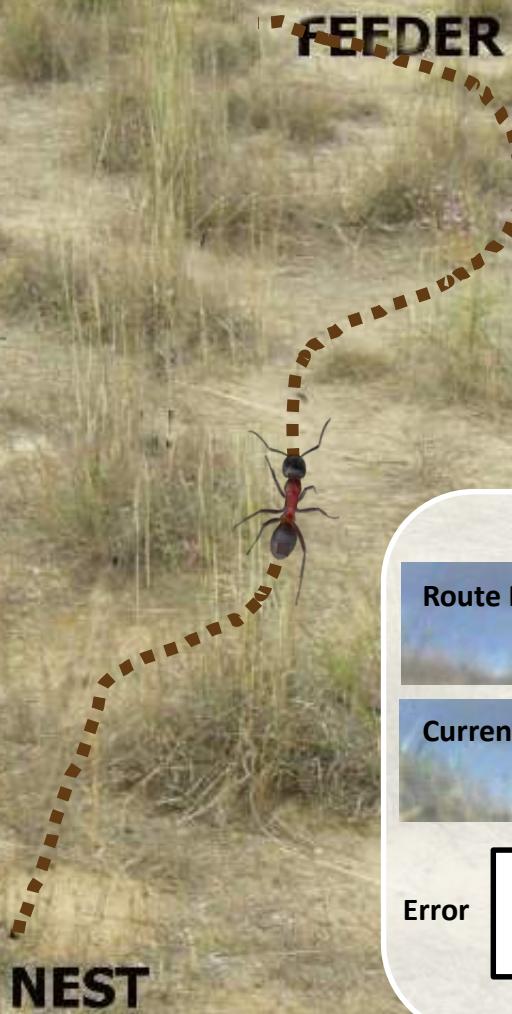
Current View

Error

Angle



The Visual Compass



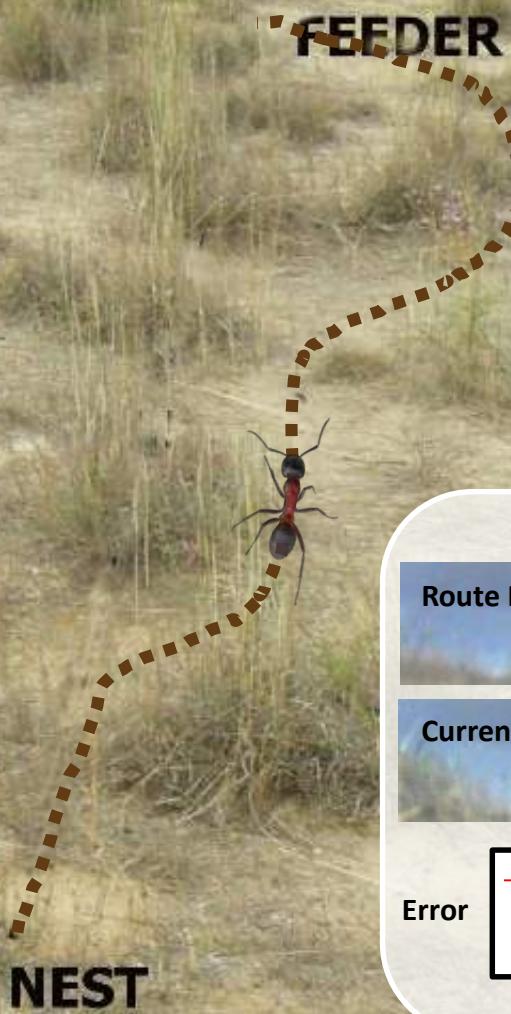
Route Memory

Current View

Error

Angle

The Visual Compass



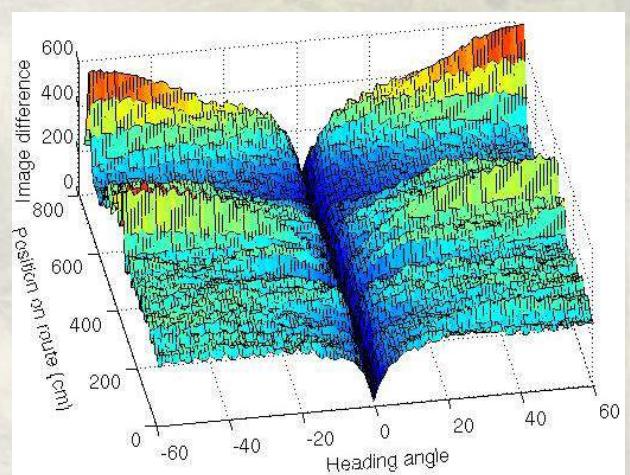
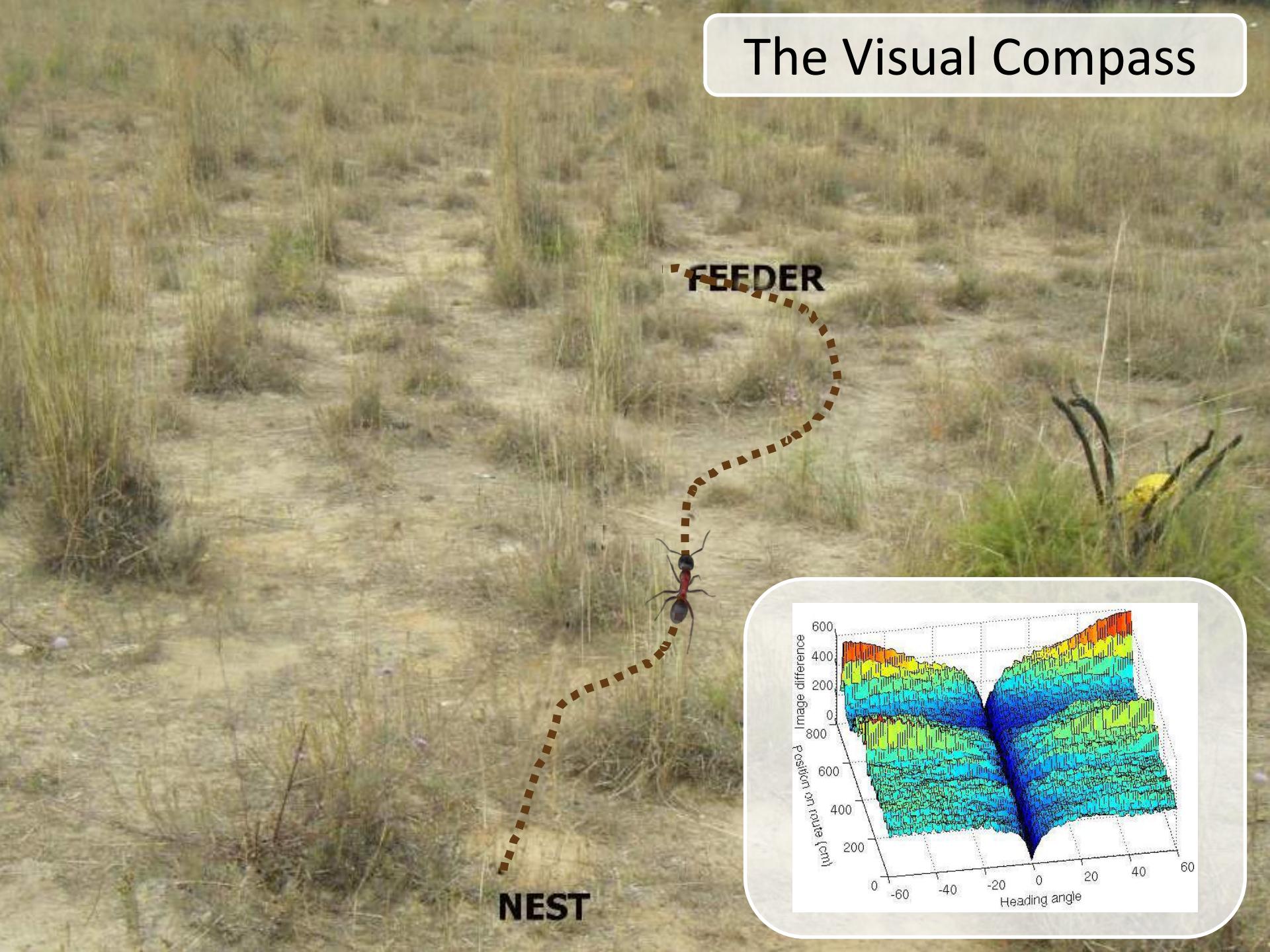
Route Memory

Current View

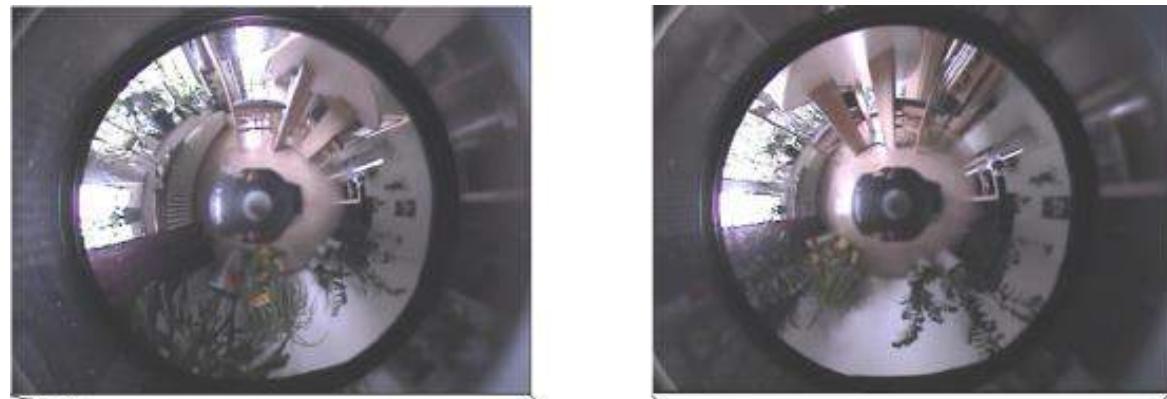
Error

Angle

The Visual Compass



Visual Homing in Robots

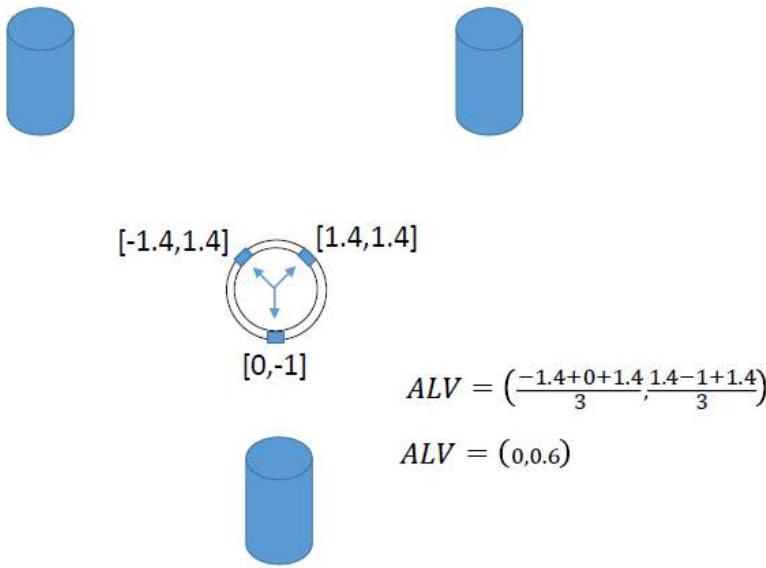


Current view (CV)

Snapshot (SS)

Compare current view to stored snapshot →
[home vector](#) pointing back to home location

Visual Homing – ALV model

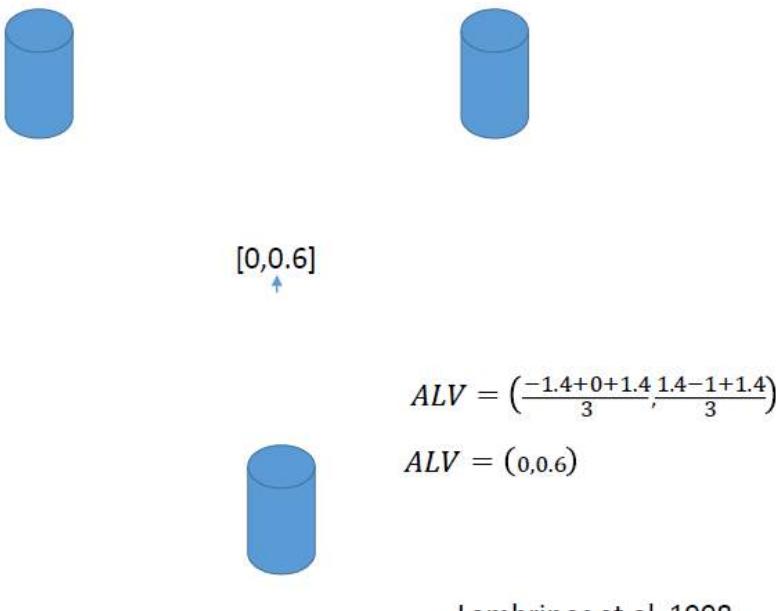


Lambrinos et al, 1998

- ALV: Average Landmark Vector
- Before leaving home:
 - Take snapshot
 - Point a unit vector to each surrounding landmark
 - Compute ALV and store
- Before leaving home:
 - Compute ALV as before
 - Use vector subtraction to generate home vector

$$\text{home vector} = \text{ALV}_{\text{current}} - \text{ALV}_{\text{home}}$$

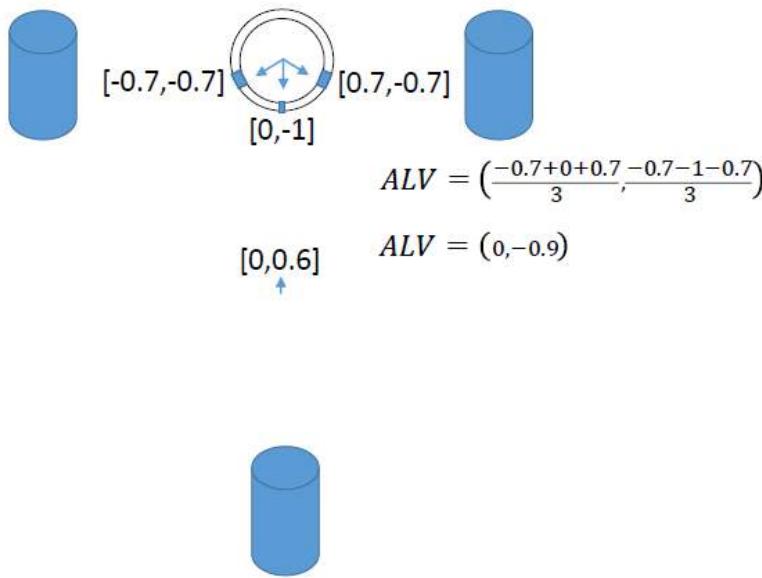
Visual Homing – ALV model



- ALV: Average Landmark Vector
- Before leaving home:
 - Take snapshot
 - Point a unit vector to each surrounding landmark
 - Compute ALV and store
- Before leaving home:
 - Compute ALV as before
 - Use vector subtraction to generate home vector

$$\text{home vector} = ALV_{\text{current}} - ALV_{\text{home}}$$

Visual Homing – ALV model

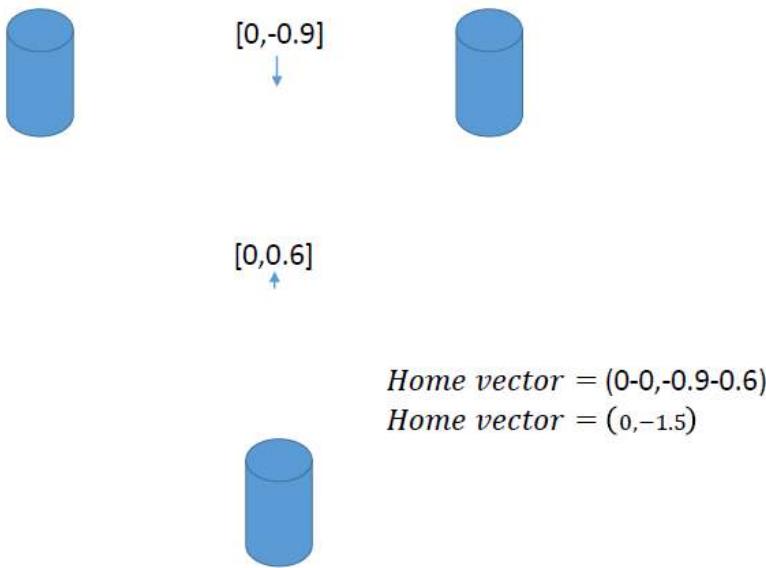


Lambrinos et al, 1998

- ALV: Average Landmark Vector
- Before leaving home:
 - Take snapshot
 - Point a unit vector to each surrounding landmark
 - Compute ALV and store
- Before leaving home:
 - Compute ALV as before
 - Use vector subtraction to generate home vector

$$\text{home vector} = ALV_{current} - ALV_{home}$$

Visual Homing – ALV model



Lambrinos et al, 1998

- ALV: Average Landmark Vector
- Before leaving home:
 - Take snapshot
 - Point a unit vector to each surrounding landmark
 - Compute ALV and store
- Before leaving home:
 - Compute ALV as before
 - Use vector subtraction to generate home vector

$$\text{home vector} = \text{ALV}_{\text{current}} - \text{ALV}_{\text{home}}$$

Problems of ALV Model

1

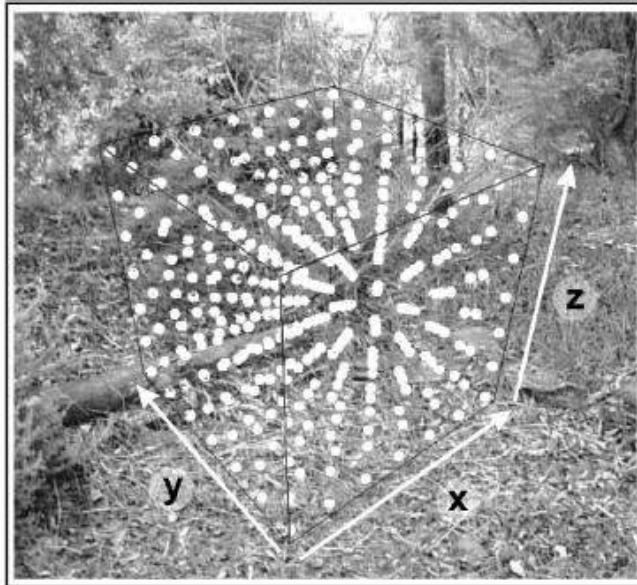
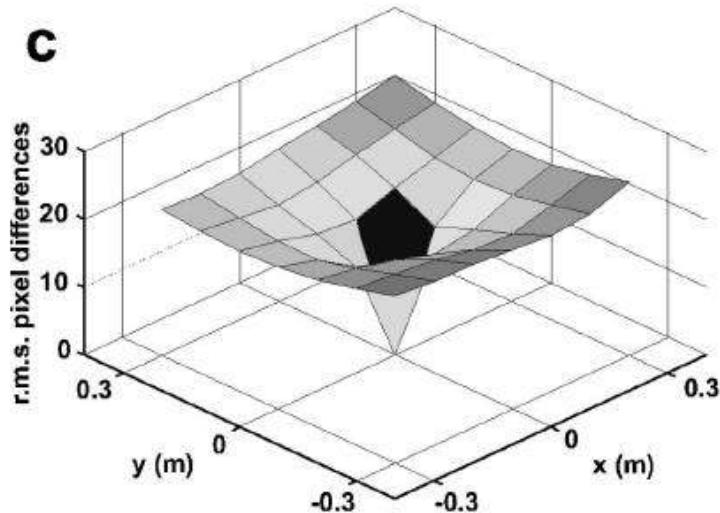
How to define a
landmark?

2

~~How to match
landmarks between
current scene and
memory?
(correspondence
problem)~~

3

You must be
aligned

a**c**

Homing by View Matching

- Zeil's pixelwise image differences is amongst the simplest algorithms proposed for insect visual homing
- Taking the RMS pixelwise difference between sampled images and a reference image, he found that the error grows monotonically with distance
→ a gradient that could be used for homing.
- The issue with this approach is that images firstly have to be rotationally aligned.

Problems of View Matching

1

~~How to define a
landmark?~~

2

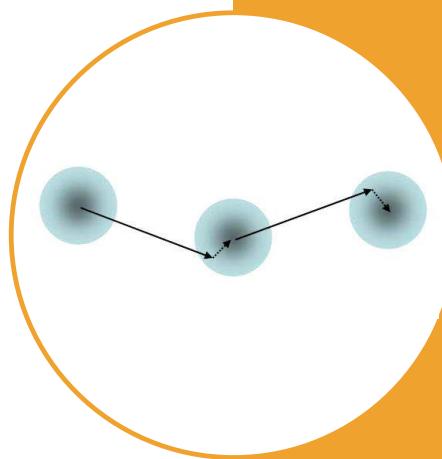
~~How to match
landmarks between
current scene and
memory?
(correspondence
problem)~~

3

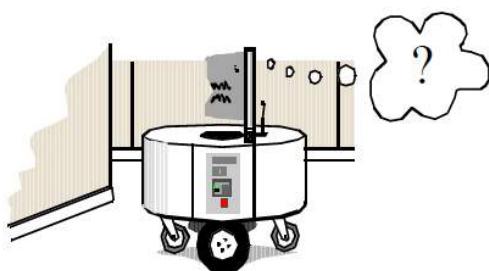
You must be
aligned

From local strategies to routes

- Local strategies
 - Target location can be found from surrounding catchment area using a memory of the target location
 - Outbound route can be used to calculate vector direction to return to starting point
- Multiple memories and/or vectors can be linked together to form route memories



Definitions



- Global navigation
 - Robot is not told its initial position
 - Position should be estimated from scratch
- Position tracking (continuous localization)
 - Robot knows initial position
 - It has to accommodate small errors in odometry as it moves

OTHER EXAMPLE NAVIGATION STRATEGIES

Bird Navigation

- Migratory birds such as geese – navigate up to 7000 km between continents
- Typical cues used by birds for navigation:
 - Earth's magnetic field,
 - orientation of the sun,
 - position of the stars,
 - landmarks such as mountains and lakes (near to the destination)



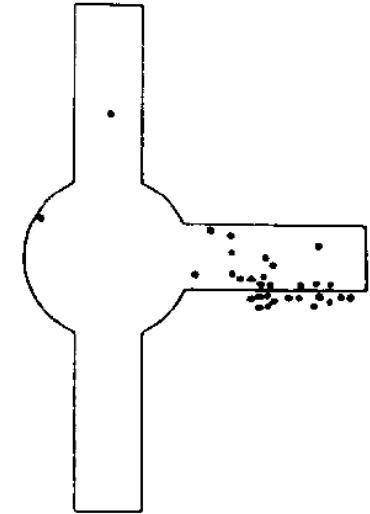
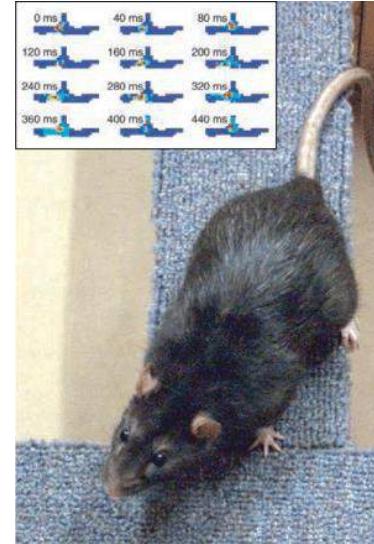
Bird Navigation - Homing Pigeons

- Use navigational map
 - Able to find their way home from large distances, even if they have been carried away in covered cages
 - Map = a memory of landmarks and also naturally occurring variations in the magnetic field near home
 - Other birds also use optic-flow information from visual system (dead reckoning)



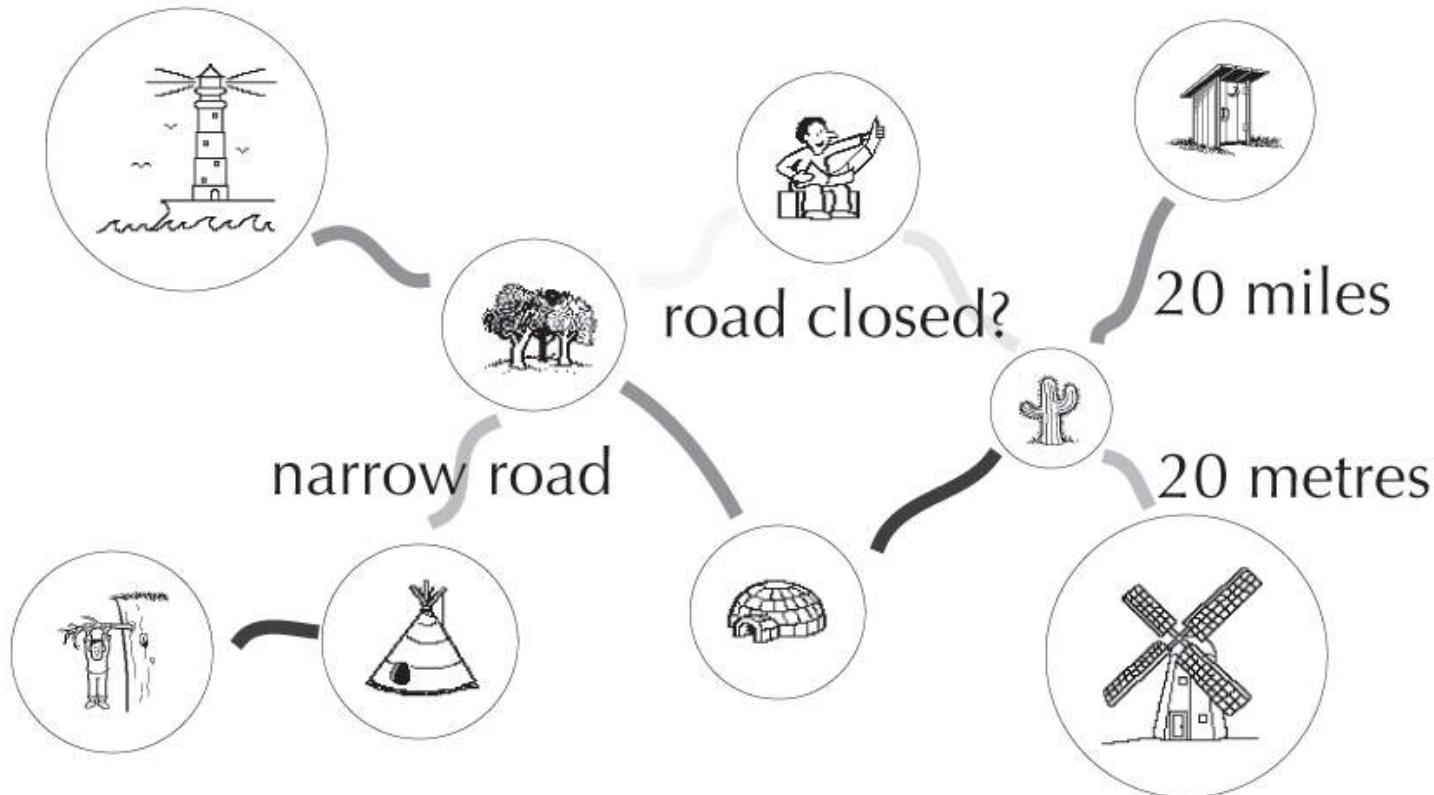
Navigation in Rats

- Place cells – neurons in rat hippocampus show increased firing rate whenever the rat is in a particular place in its environment
- In a new environment, place cell coding gradually emerges, then remains consistent
- Also head direction cells - cells that fire when the rat faces a specific direction



Human Navigation

Something like this... ?



Summary: Navigation Strategies in Nature

Local strategies

- Searching, direction following, aiming, guidance

Global strategies

- Route following, map-based navigation

Many different sensory modalities

- Exteroceptive – stimuli from outside the body
- Proprioceptive – stimuli produced within the body

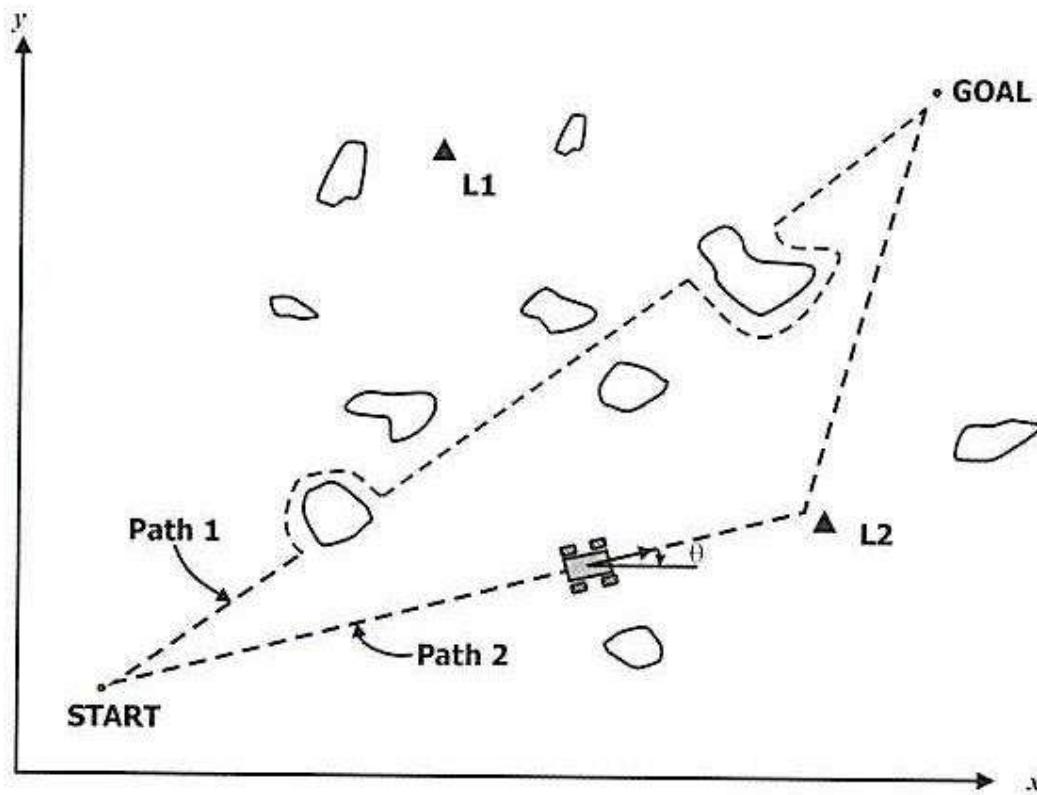
Multiple redundant mechanisms

- May switch between landmarks, route following, compass sense, etc. depending on the information currently available

NAVIGATION STRATEGIES FOR ROBOTS

Navigation Strategies for Robots

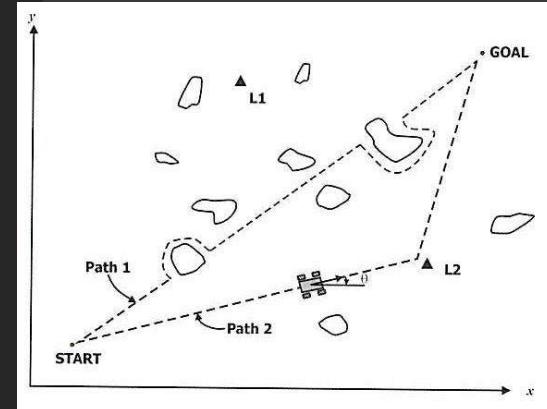
A simplified scenario:



Navigation Strategies for Robots

Some possible strategies:

1. Navigation by vision and compass (Path 1)
(assuming visible goal, unknown distance to goal)
 - Use vision to recognise goal
 - Use compass to maintain heading towards goal
 - When obstacle encountered (goal no longer visible), remember current compass reading, then travel around obstacle until the original compass direction is sensed again and goal is visible
2. Navigation using landmarks as beacons (Path 2)
(assuming goal not visible from start)
 - Aim towards landmark 2, then towards goal



Navigation Strategies for Robots

3. Navigation by position tracking (goal not visible, but known coordinates)

- Use GPS (SatNav), but...
 - ... GPS does not always work and/or not perfectly accurate
- Cannot use only odometry due to drift errors (i.e. cumulative over time)

→ use a map with positions of known landmarks to correct your odometry – problem of self-localisation (next lect.)



Navigation Strategies for Robots

- A general-purpose solution to the navigation problem in mobile robots also requires:
 - A representation of the navigable space, e.g. as a graph or grid, derived from the global map of the environment
 - A path planner – use algorithms such as A* or Dijkstra to find the best route to the goal location → a set of waypoints
 - An “auto-pilot” – control software to drive between waypoints taking into account the robot’s kinematics and dynamics
 - Also need to avoid unexpected obstacles on the way...

Suggested reading

- Nehmzow, U., *Mobile Robotics: A Practical Introduction*, (Springer, 2003). Chapter 5.
- Bekey, G.A., *Autonomous Robots: From biological inspiration to implementation and control*, (MIT Press). Chapter 14.
- Siegwart R. et al., *Autonomous Mobile Robots*, (MIT Press). Chapter 5.



CMP3103M/CMP9050M

Autonomous Mobile Robotics

Dr Ayse Kucukyilmaz

University of Lincoln
Centre for Autonomous Systems

INB3201

akucukyilmaz@lincoln.ac.uk
<http://webpages.lincoln.ac.uk/akucukyilmaz/>



UNIVERSITY OF
LINCOLN

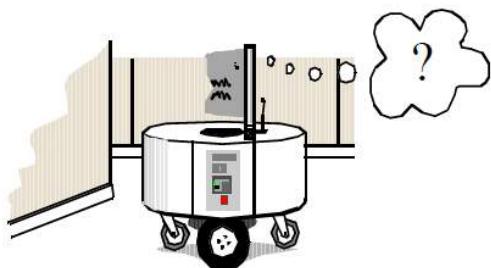


Syllabus

1. Introduction (MH)
2. Robot Programming (MH)
3. Robot Sensing (MH)
4. Motion and Control (MH)
5. Robot Behaviours and Navigation (AK)
6. **Localisation and Mapping(AK)**
7. Self-Localisation (AK)
8. Planning (AK)
9. Control Architectures (PB)
10. Human Robot Interaction 1 (PB)
11. Human Robot Interaction 2 (PB)
12. Applications (PB)



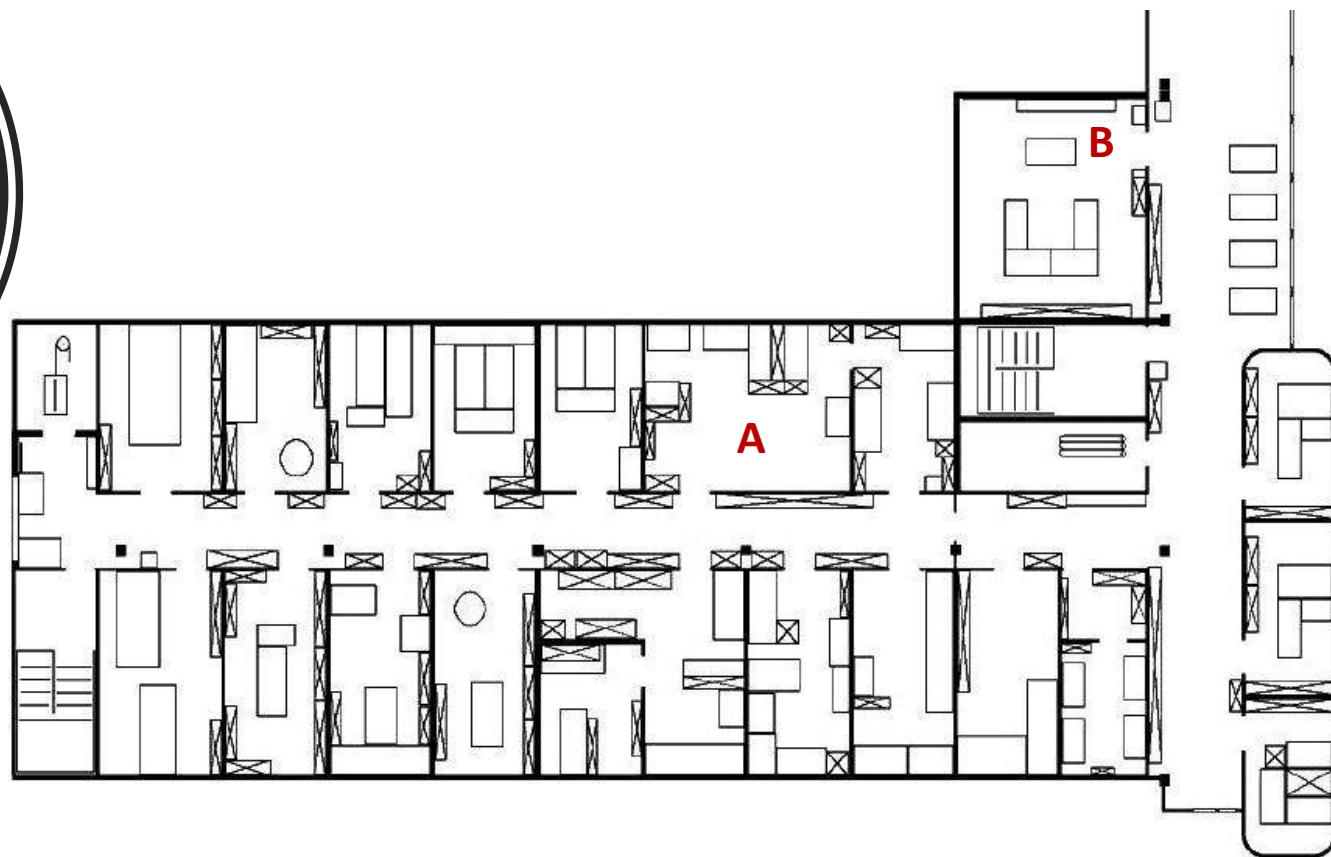
Key questions



- Where am I?
- Where do I go?
- How do I go there?
- To answer these, the robot needs
 - A model of the environment (either given or autonomously built)
 - To perceive and understand the environment
 - Find its position and situation in the environment
 - Plan and execute the movement

Does the robot need to know where it is?

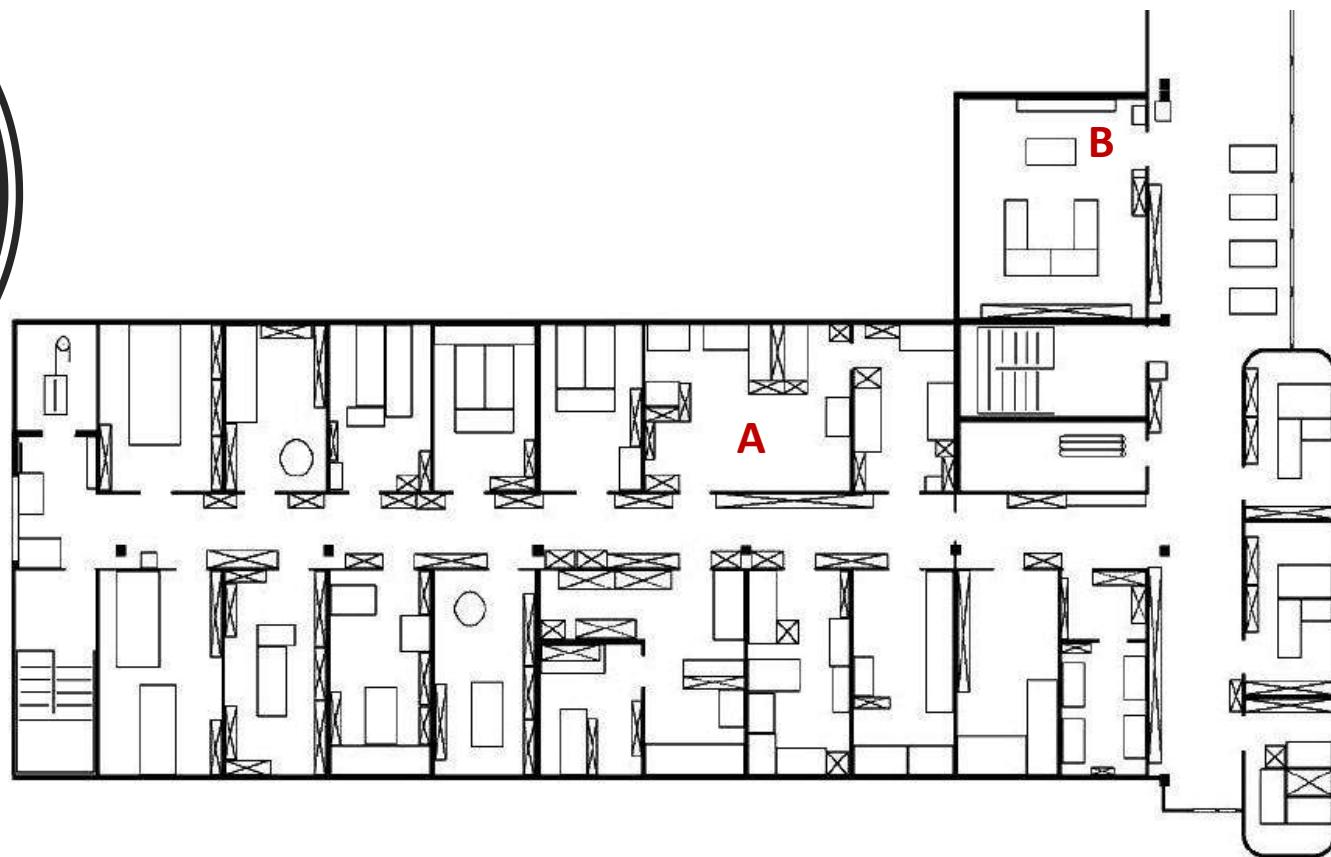
Task: Go from
A to B



Task: Go from
A to B

Navigation:

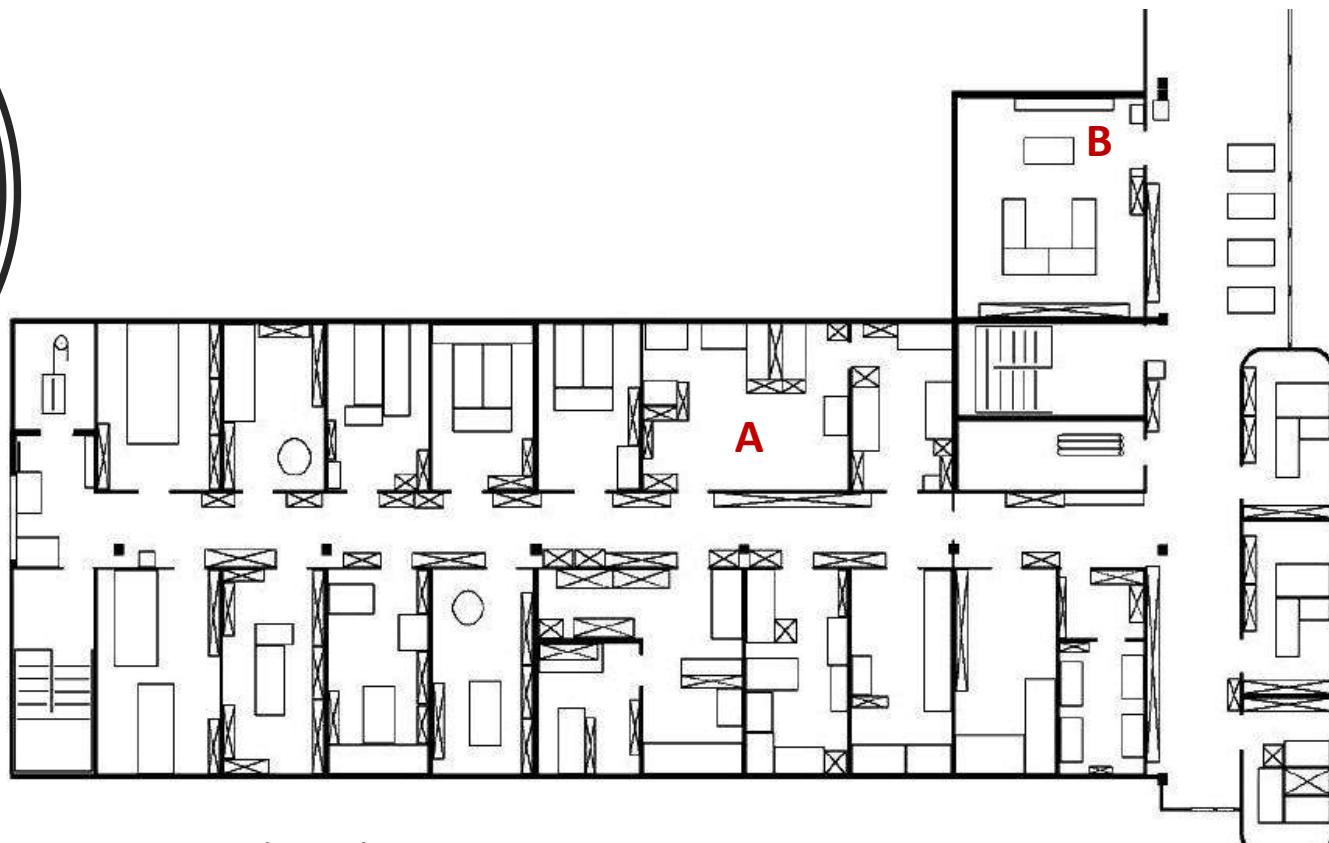
- Without hitting obstacles
- Detection of arrival at goal



Task: Go from
A to B

Behaviour based navigation:
Always following the left wall?

- But how do we know that the goal is reached?

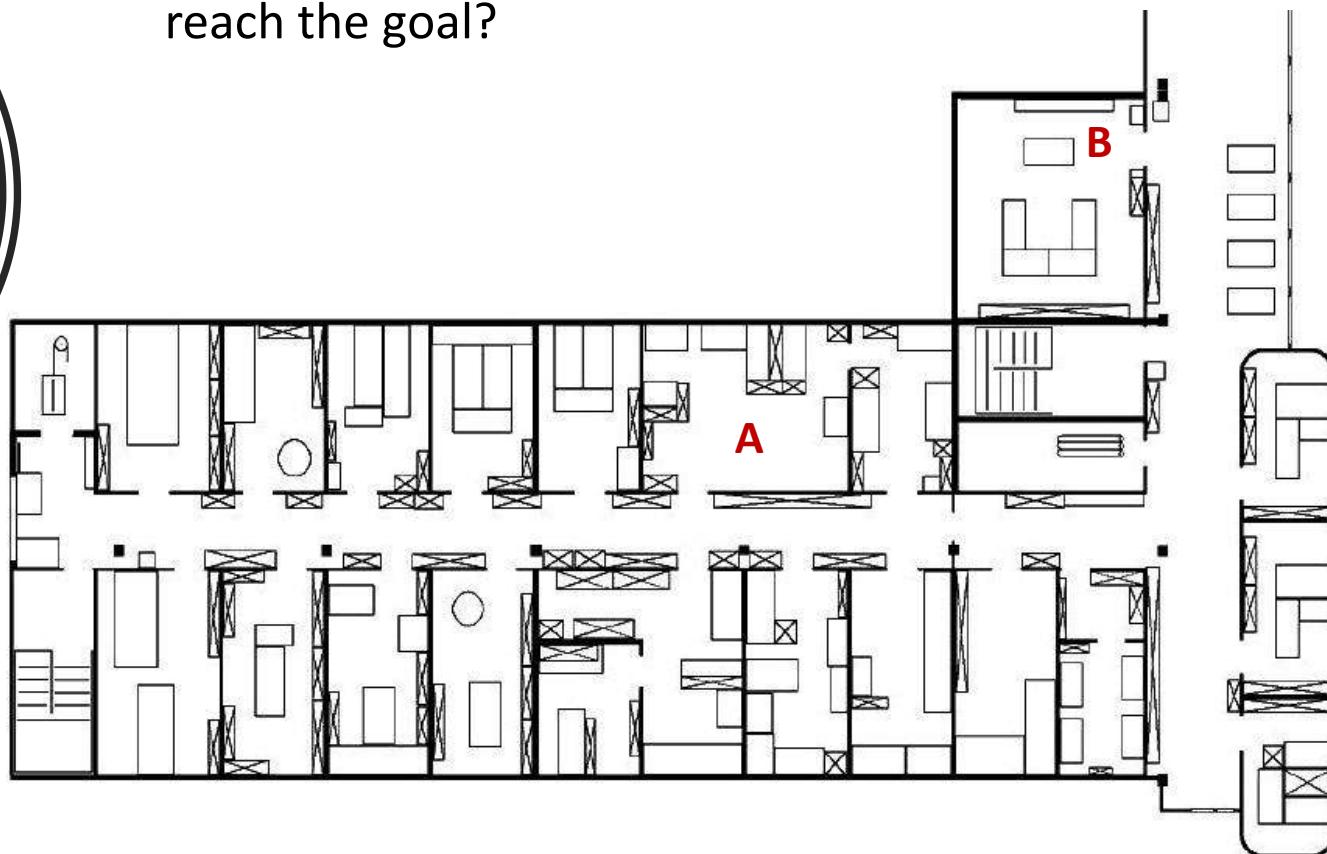


Task: Go from
A to B

Behaviour-based navigation:

Procedure: Always follow the left wall

- Does it work in all environments?
- How do we know that the goal is reached?
- With which accuracy and reliability do we reach the goal?



Task: Go from
A to B

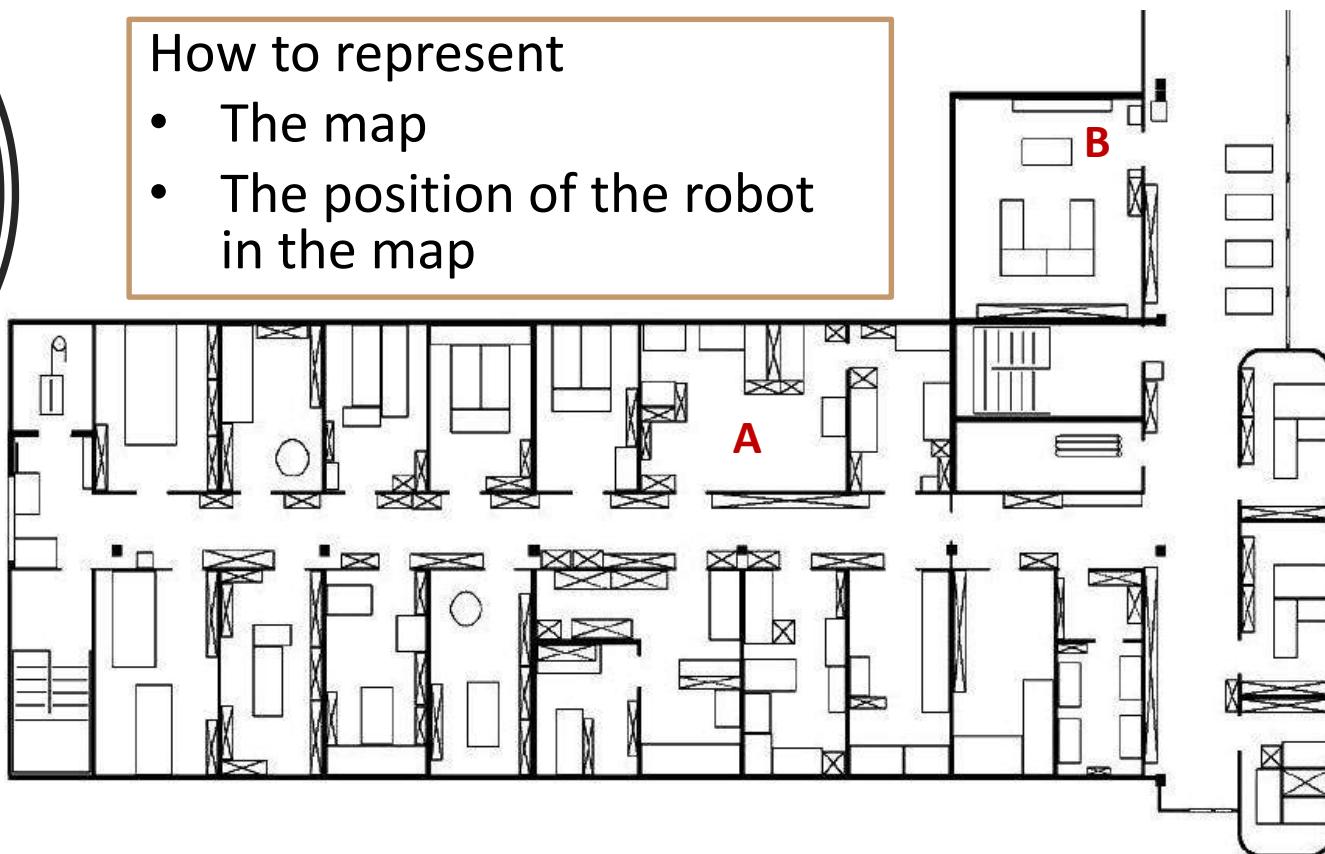
Map based navigation:

If map is known, at every step the robot needs to know where it is. How?

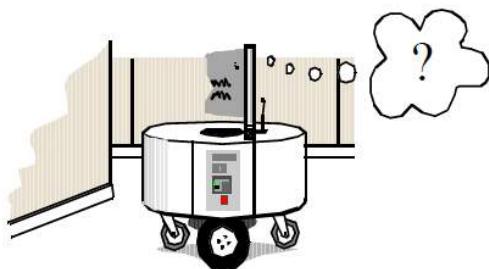
- Wheel odometry?
- Dead reckoning?

How to represent

- The map
- The position of the robot in the map



Definitions



- **Global navigation**
 - Robot is not told its initial position
 - Position should be estimated from scratch
- **Position tracking**
 - Robot knows initial position
 - It has to accommodate small errors in odometry as it moves

Localisation: How to?

- Based on external sensors, beacons, landmarks
- Odometry
 - Only proprioceptive sensors
- Map-based
 - No external sensors and landmarks
 - Only onboard sensors

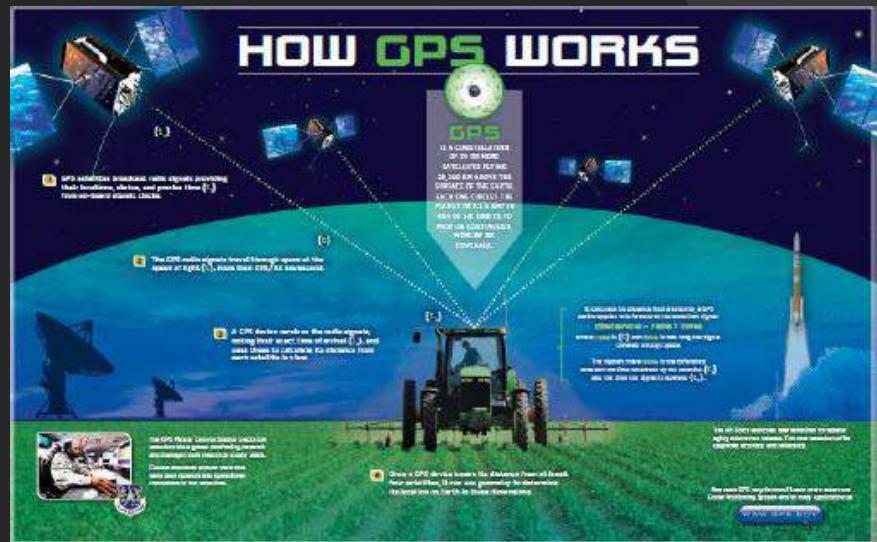


GPS positioning

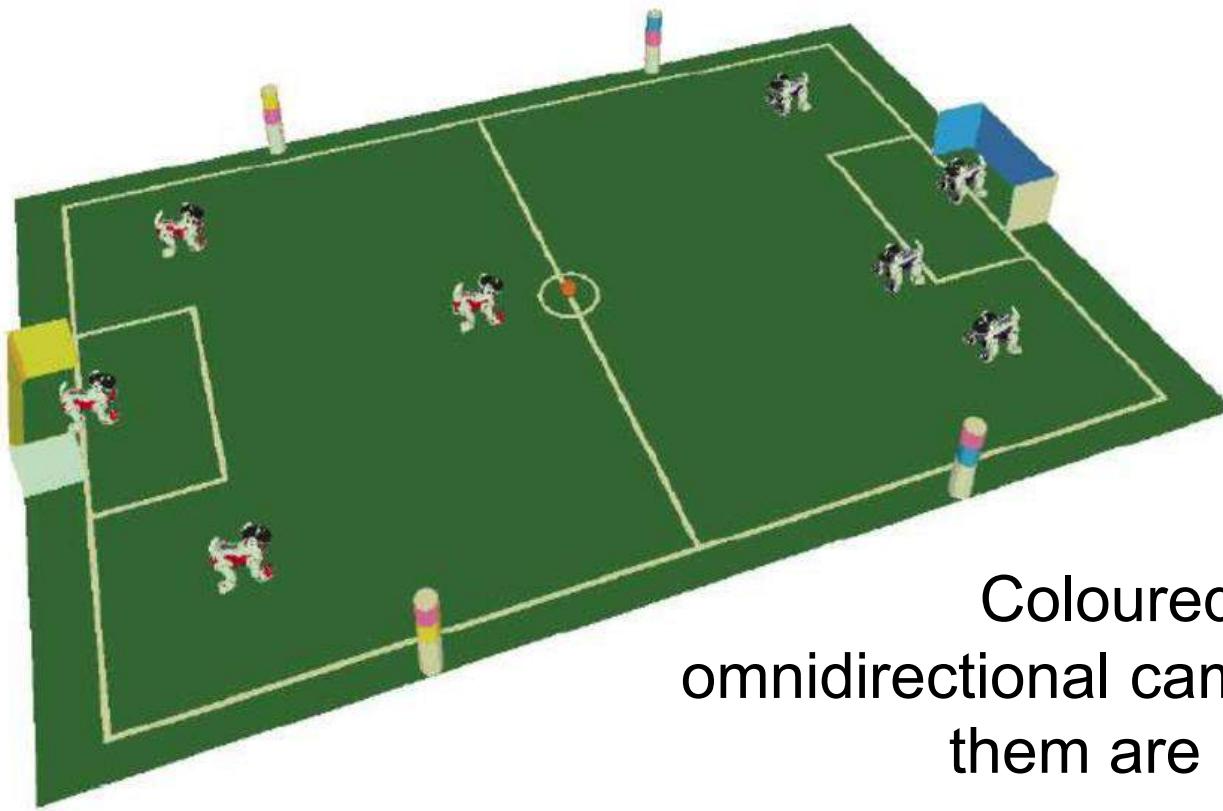
- GPS provides metre-level outdoor positioning using information from multiple satellites
- Atmospheric conditions limit precision unless corrected using a reference (DGPS)
- Can be power hungry
- Similar methods used from mobile phone masts.

Limitations:

- Accuracy
 - ~2m at the very best
 - Poor in urban environments
- Need for infrastructure



Beacon based localisation



Coloured beacons and an omnidirectional camera for detecting them are used in RoboCup

Examples: Beacon-Based Localisation

KIVA Systems, acquired by Amazon, 2011

Unique markers indicate absolute 2D position in the map

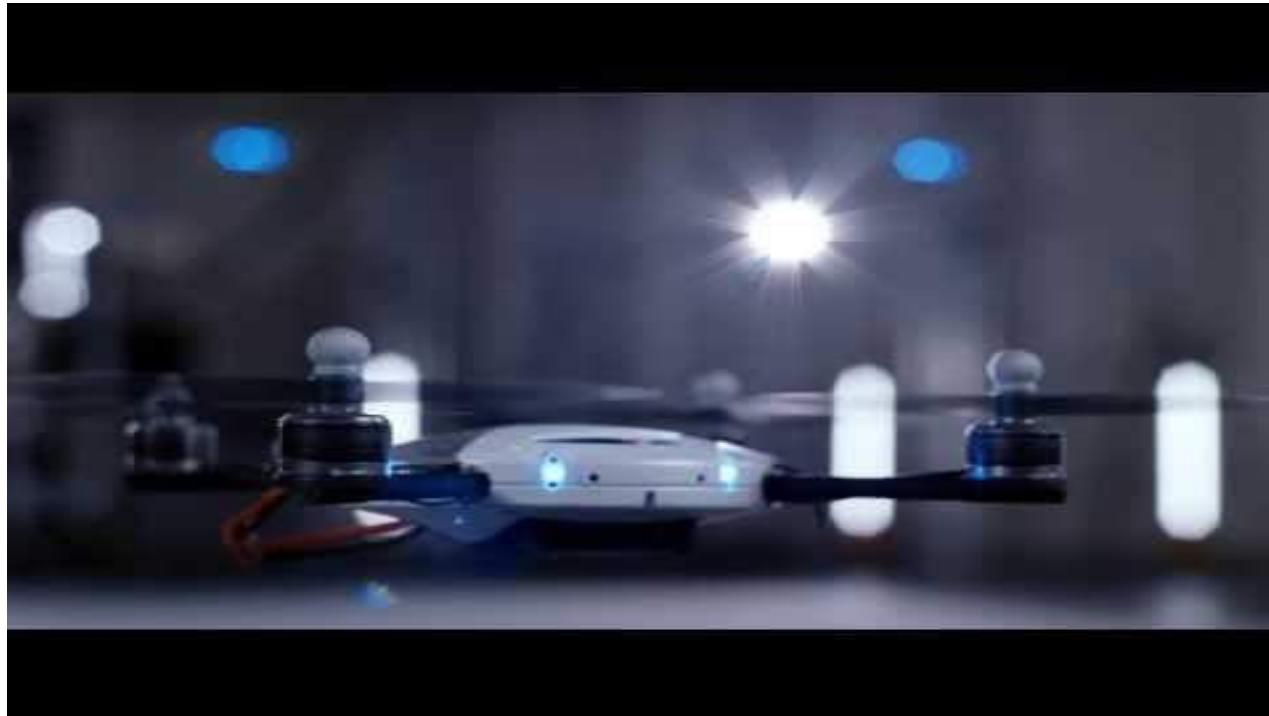


<https://www.youtube.com/watch?v=8gy5tYVR-28>

Examples: Motion Capture Localisation

High resolution, high speed IR cameras

Good for ground truth reference and multi-robot control



<https://www.youtube.com/watch?v=M1ShuAEIfGw&feature=youtu.be>

Beacon based localisation



Generally limited accuracy



External references can be used for triangulation



City Line of sight may be blocked (indoors, cities)



Signals may be jammed

ODOMETRY

Cataglyphis fortis lives in extremely flat and featureless salt planes in the Sahara

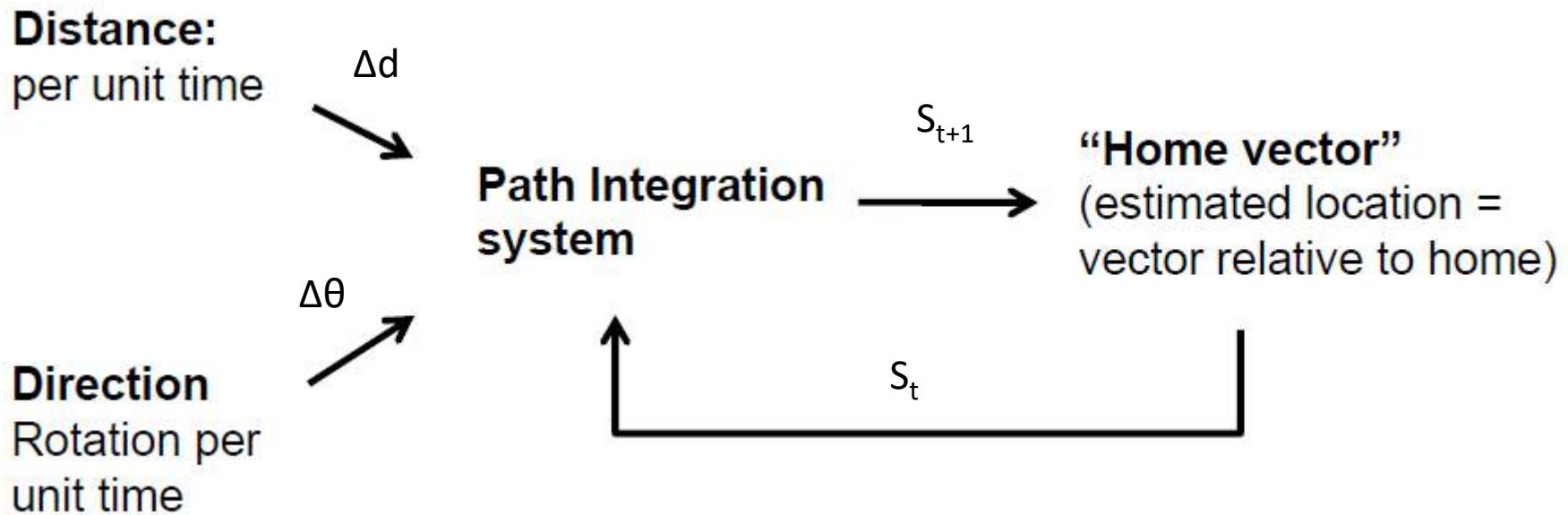


A motivating example:
Path Integration in Desert Ants

- Many ants use pheromones to mark their trails
- However, desert ants use path integration:

As they travel from their nests, they integrate their 2D movements to obtain a global vector (home vector) pointing back towards the nest

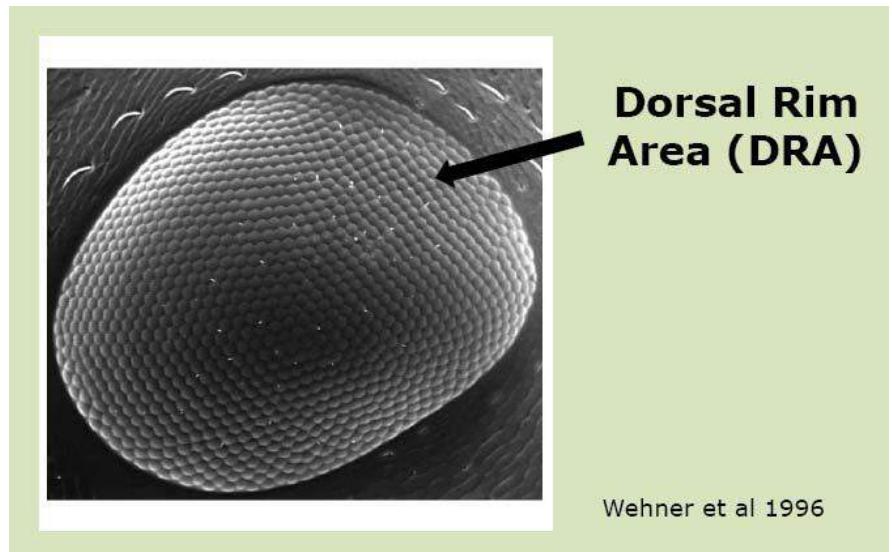
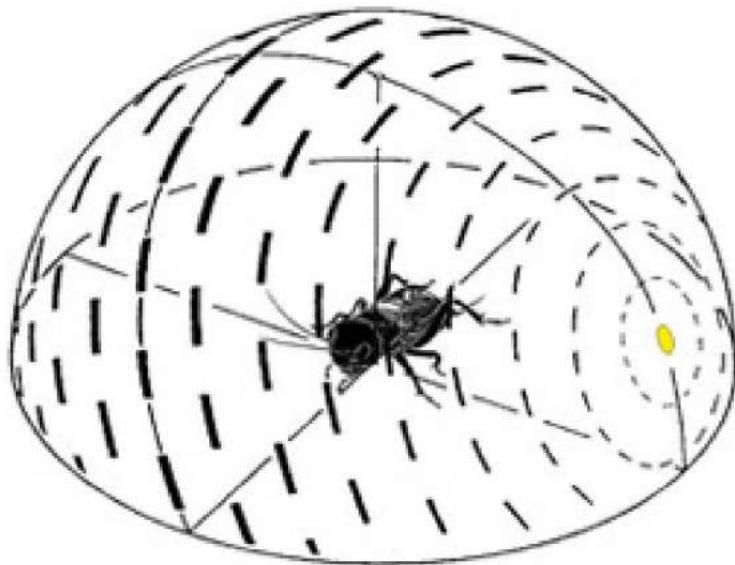
Outline of a Path Integration System



Need mechanisms for measuring

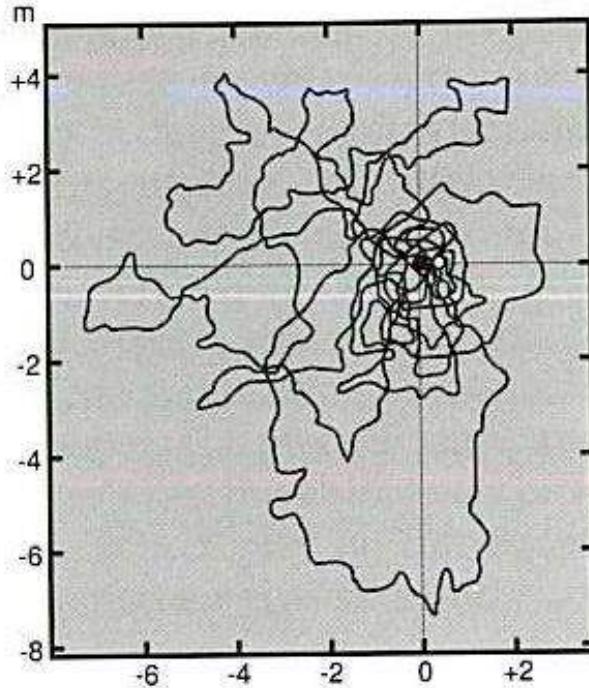
- distance (per unit time) – step counter
- direction (per unit time) – biological compass

Polarization Compass in *Cataglyphis fortis*



Wehner et al 1996

A visual compass sense based on polarization patterns in the sky – insects can see the polarization pattern of the sky in the Dorsal Rim Area of their compound eyes



Homing Behaviour in *Cataglyphis fortis*

- Path integration is prone to cumulative error (as with robot odometry)
 - e.g. random errors of $\sim 10^\circ$ in rotation, and 25% in distance
- Back-up strategy in the desert ant: systematic search
 - ever increasing loops, spiralling outwards

Odometry

- **Odometry**: An approximation of the location of a robot can be obtained by repeatedly computing the distance moved and the change direction from the velocity of the wheels in a short period of time.
 - Odos: route; metron : measure
 - Also called **deduced reckoning** or **dead reckoning**
- Robot motion is recovered by integrating **proprioceptive** sensor velocities readings
 - Pros: Straightforward
 - Cons: Errors are integrated -> unbound
- Heading sensors (e.g., gyroscope) help to reduce the accumulated errors but drift remains :(

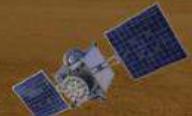
Navigation by Odometry

Example : piloting barren Martian surface
with no external guidance cues e.g. GPS



Navigation by Odometry

Example : piloting barren Martian surface
with no external guidance cues e.g. GPS

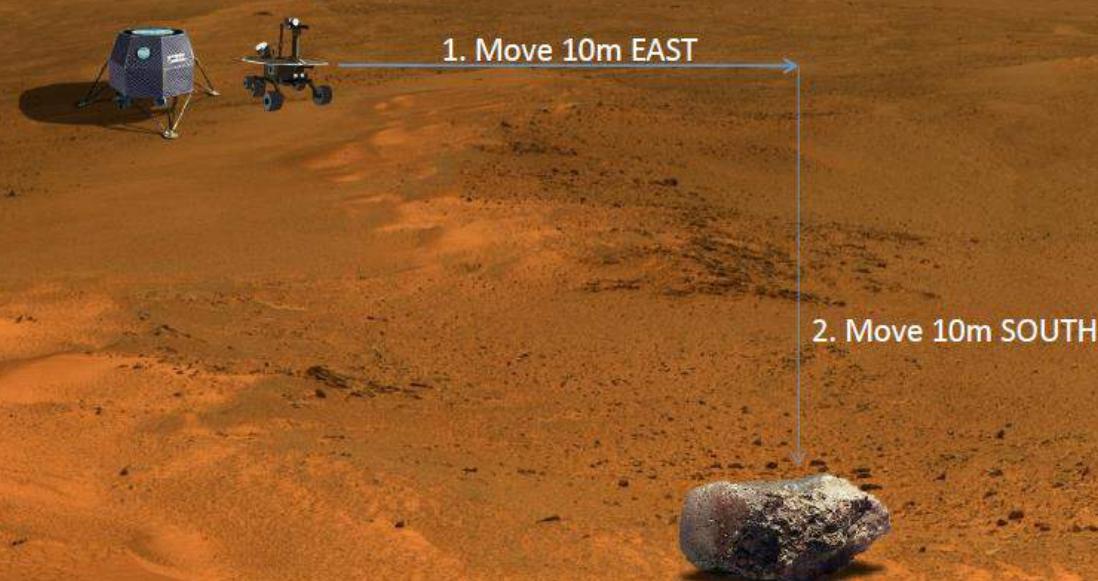


!! NASA COMMANDS !!

1. Move 10m EAST
2. Move 10m SOUTH
3. Bring back minerals

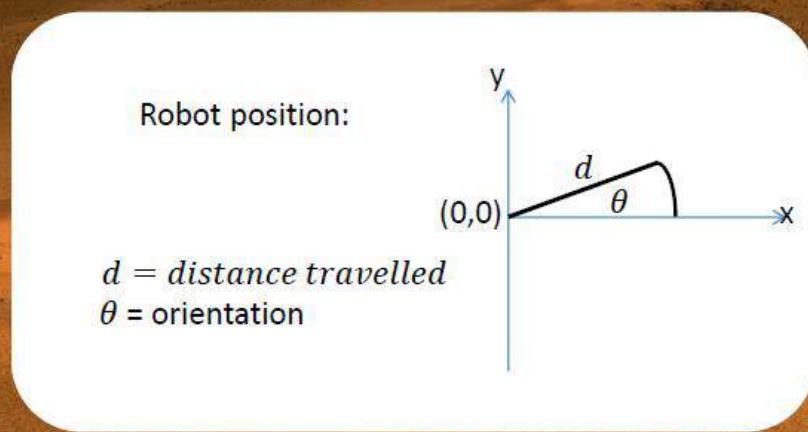
Navigation by Odometry

Example : to execute commands the robot must continuously calculate it's position wrt to base



Navigation by Odometry

Example : formulation



1. Move 10m EAST

2. Move 10m SOUTH

Navigation by Odometry

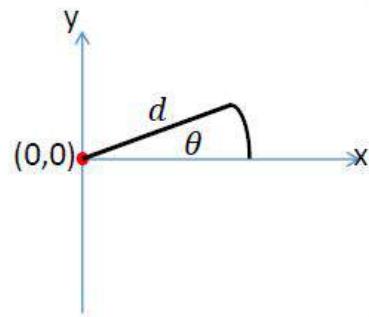
Example : formulation

Robot position:

$$x(t) = x(t-1) + \Delta x$$
$$y(t) = y(t-1) + \Delta y$$

where

$$\Delta x = d * \cos(\theta)$$
$$\Delta y = d * \sin(\theta)$$



1. Move 10m EAST

2. Move 10m SOUTH

Navigation by Odometry

Example : monitoring orientation

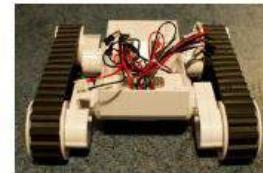
Tracking Orientation



External
Compass



Directional
wheels



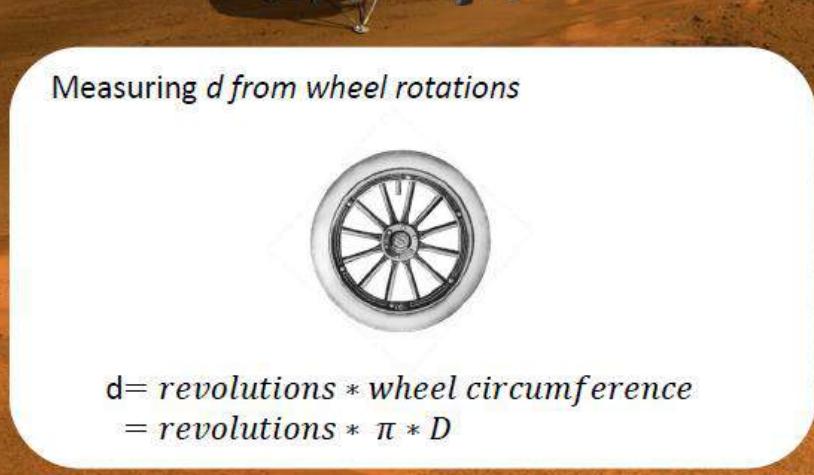
Difference in
wheel speeds*

1. Move 10m EAST

2. Move 10m SOUTH

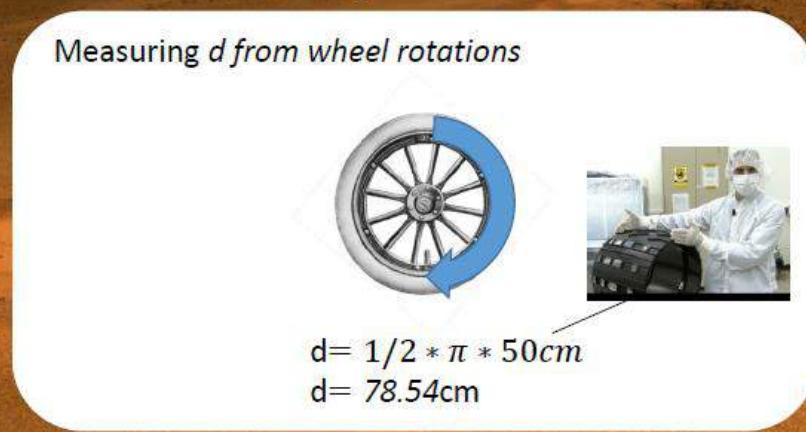
Navigation by Odometry

Example : measuring distance travelled



Navigation by Odometry

Example : measuring distance travelled

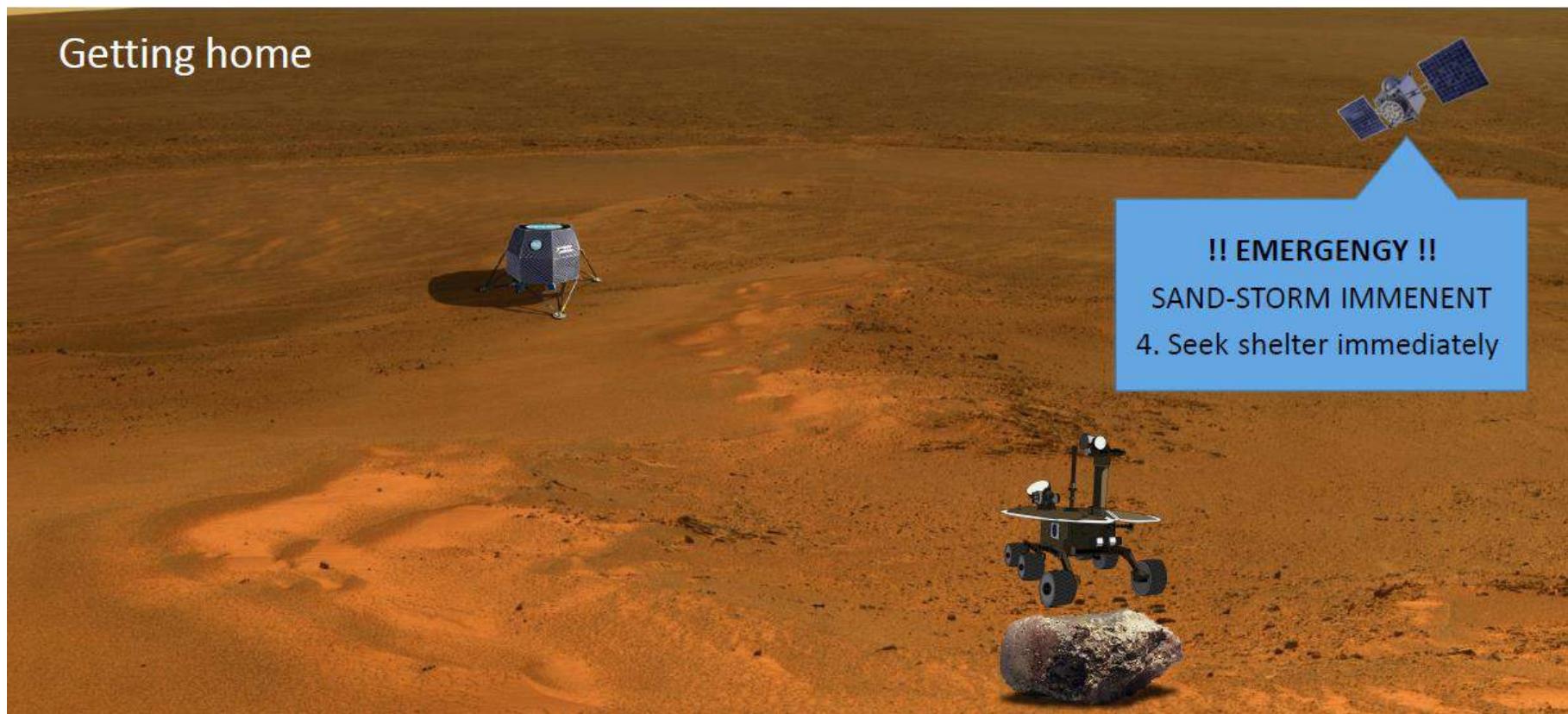


Navigation by Odometry

Example : monitoring orientation



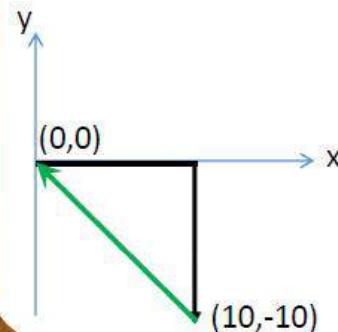
Navigation by Odometry



Navigation by Odometry

Getting home

Calculating the home vector



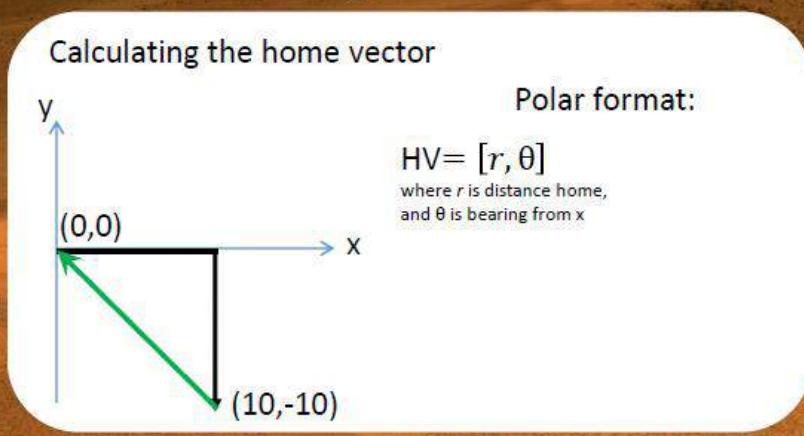
Cartesian format:

$$\begin{aligned} \text{HV} &= [x_1 - x_2, y_1 - y_2] \\ &= [0 - 10, 0 - (-10)] \\ &= [-10, 10] \end{aligned}$$



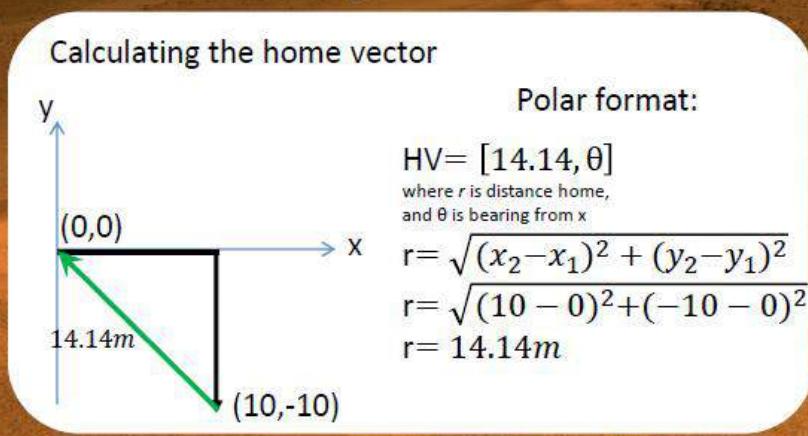
Navigation by Odometry

Getting home



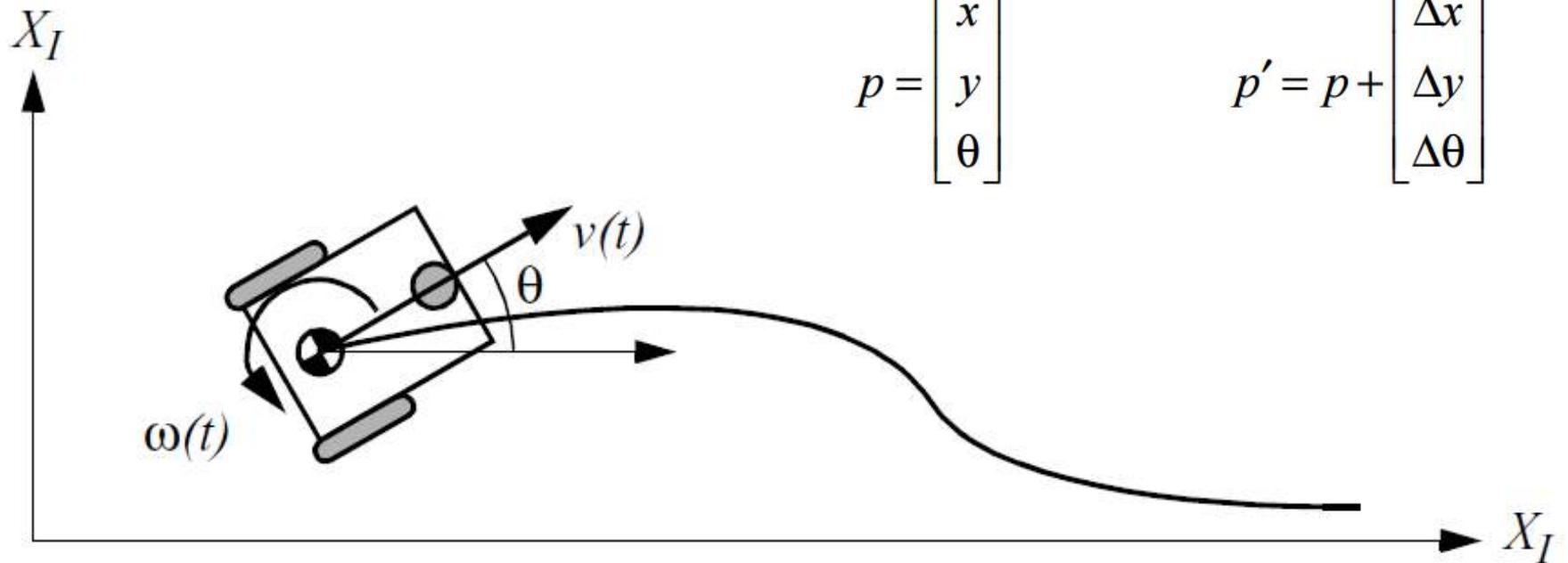
Navigation by Odometry

Getting home

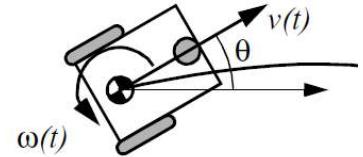


Odometry

Differential drive robot



Wheel Odometry



Incremental travel distances for a discrete system with fixed sampling interval:

$$p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}$$

$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$

$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$

where

$(\Delta x; \Delta y; \Delta \theta)$ = path traveled in the last sampling interval;

$\Delta s_r; \Delta s_l$ = traveled distances for the right and left wheel respectively;

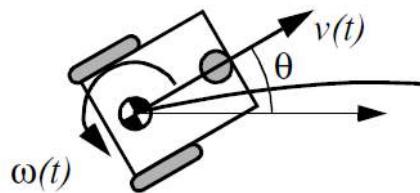
b = distance between the two wheels of differential-drive robot.

These terms comes from the application of the Instantaneous Center of Rotation

Mobile Robot Odometry

Putting it all together....

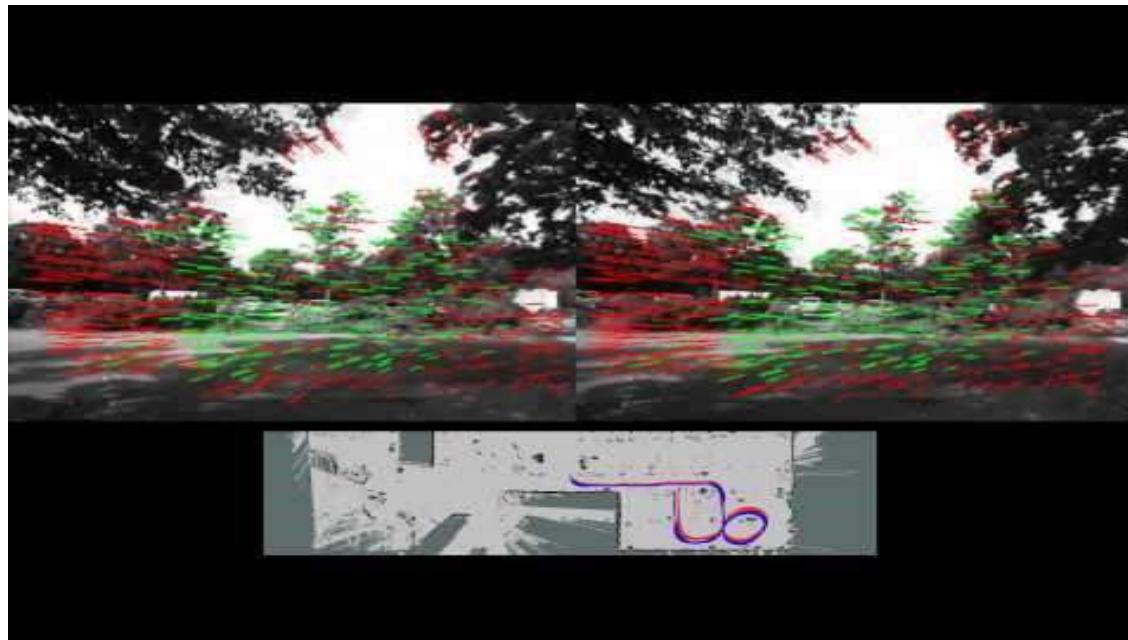
$$p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}$$



$$p' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = p + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix}$$

Visual Odometry

- Odometry is not limited to wheel encoders
- In visual odometry consecutive camera images are used to estimate velocity

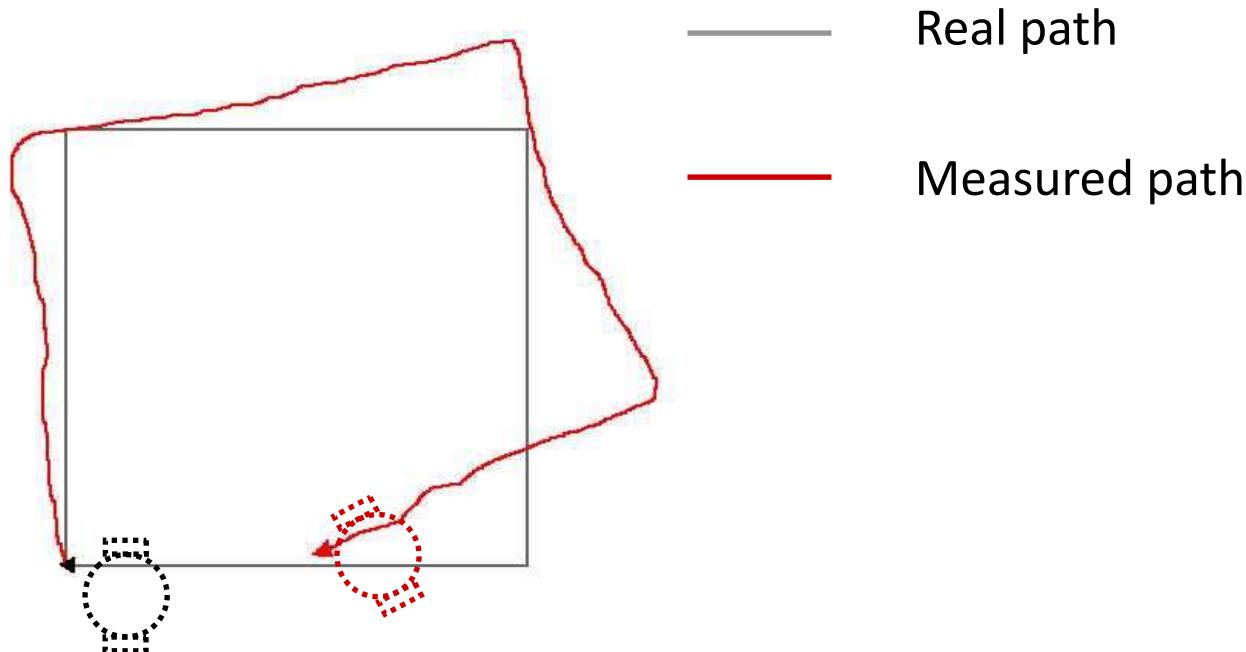


<https://www.youtube.com/watch?v=0sq1O8U2TWk>

“So where am I?!”

(The problem of Self-Localisation)

... a.k.a. What's wrong with odometry (just proprioception) ...

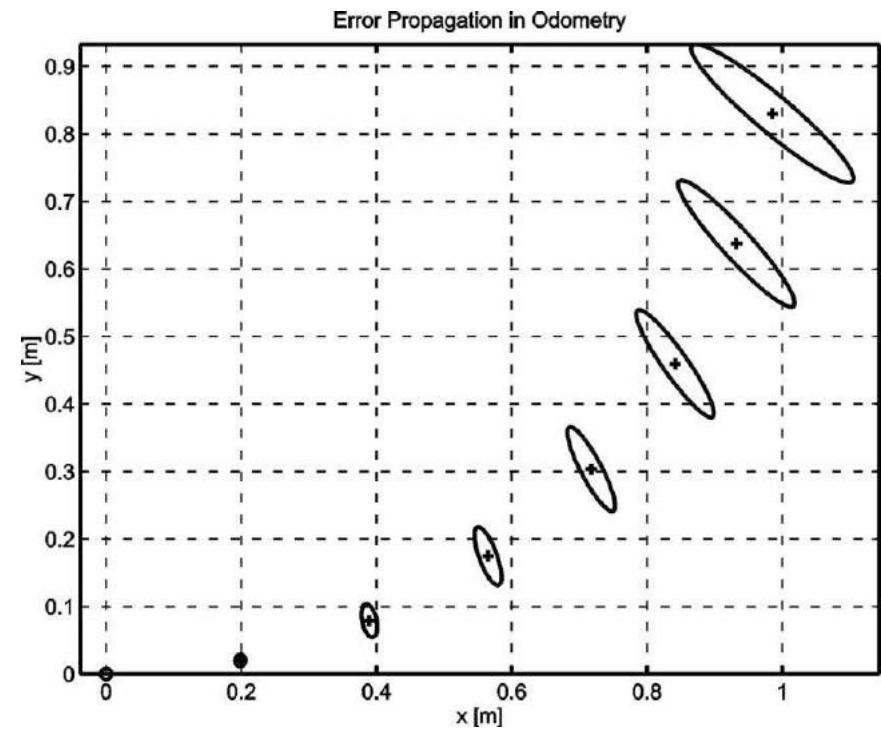
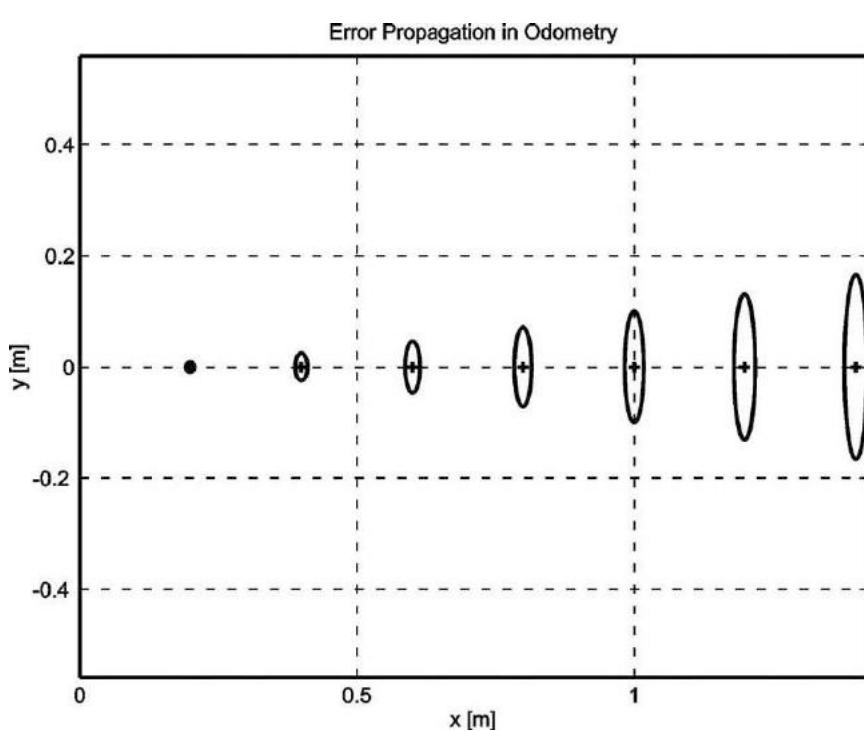


Solution: use a map! combine odometry with sightings on known landmarks / environmental features

Demonstration!

Sensor errors

Error Propagation in Odometry



Odometry in ROS

nav msgs/Odometry Message

```
# This represents an estimate of a position and velocity in free space.
# The pose in this message should be specified in the coordinate frame given by header.frame_id.
# The twist in this message should be specified in the coordinate frame given by the child_frame_id
Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

geometry msgs/PoseWithCovariance Message

```
# This represents a pose in free space with uncertainty.
Pose pose
# Row-major representation of the 6x6 covariance matrix
# The orientation parameters use a fixed-axis representation.
# In order, the parameters are:
# (x, y, z, rotation about X axis, rotation about Y axis, rotation about Z axis)
float64[36] covariance
```

Odometry Error Sources

- Misalignment of the wheels (systematic)
- Unequal wheel diameter (systematic)
- Variation in the contact point (non-systematic)
- Unequal floor contact of the wheel (slippage, non planar contact) (non-systematic)
- Limited resolution during integration (time increments, measurement resolution)

Odometry Error Types

Range error

- Sum of the wheel motions leads to an error in the integrated path distance of the robot's movement

Turn error

- Difference of the wheel motions leads to an error in the robot's final orientation

Drift error

- Difference in the error of the wheels leads to an error in the robot's angular orientation

Odometry

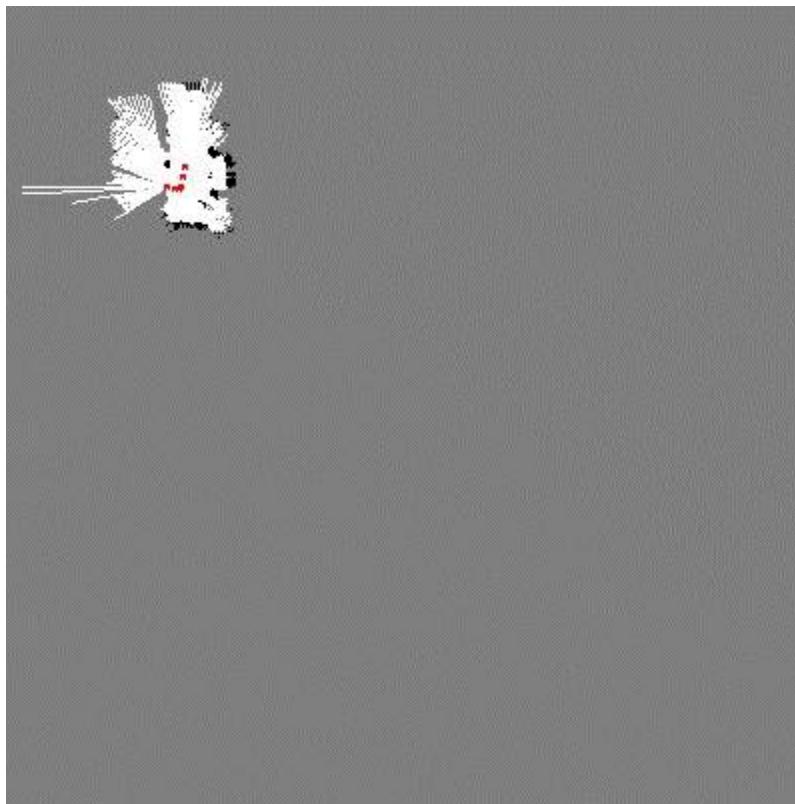
Can function independent of external cues and sensors (e.g. GPS) and without maps

...BUT accumulates error

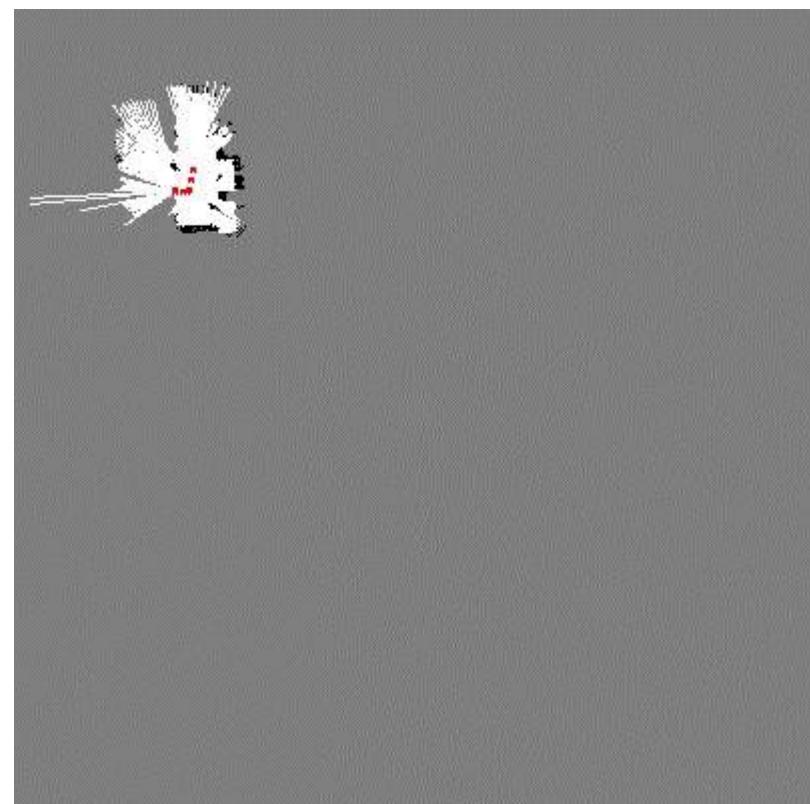
SLAM provides the best of both worlds by using odometry for rapid position updates which are realigned periodically using the map (more on this later)

Teaser: Why is self-localisation needed?

Example of mapping using odometry



Example of simultaneous localisation and mapping



... We will discuss more next week!

MAP-BASED LOCALISATION

What is a map?

Collection of elements or features at some scale of interest, and a representation of the spatial and semantic relationships among them

LEGEND			
	Tropical Wet	Humid Subtrop.	
	Tropical Dry	Cool Summer	
	Semiarid	Subarctic	
	Arid	Tundra	
	Mediterranean	Highlands	
	Marine W Coast	Ice	

Why maps?

- Useful for a wide variety of robotic activities, e.g.:
 - localisation,
 - planning,
 - mobile manipulation,
 - human-robot interaction

but...

- Mapping is highly labour intensive
 - Exploration (global coverage)
 - Measurement (local coverage)
 - Validity (correctness, error bounds)
 - Currency (freshness)

Types of Maps

Metric Maps

- Record the location of objects in an absolute coordinate system

Topological Maps

- Record the connections (edges) between a set of places (nodes)

Semantic Maps

- Record semantic information (metadata), includes segmentation, place/object naming, function, etc.

Hybrid Maps

- Combine two or more of the map types above

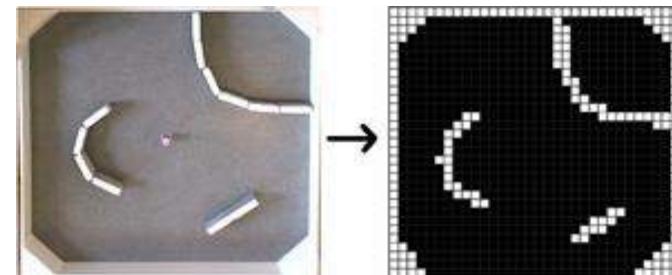
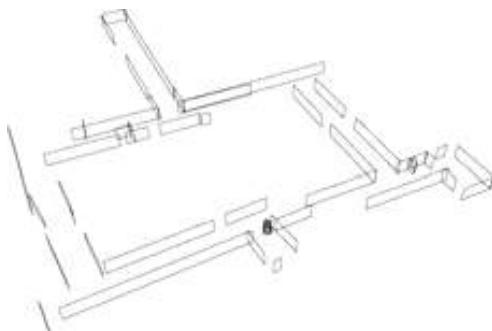
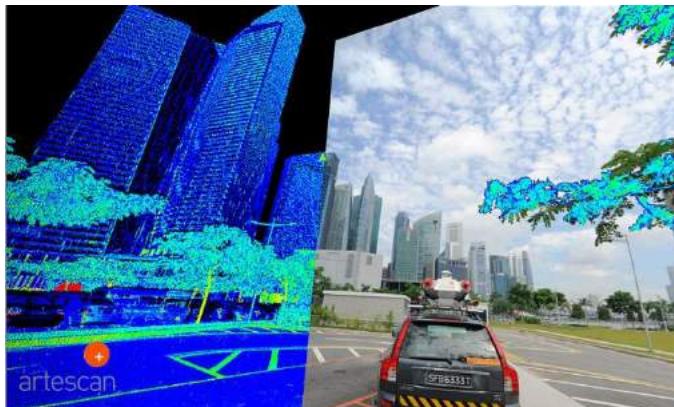


METRIC MAPS

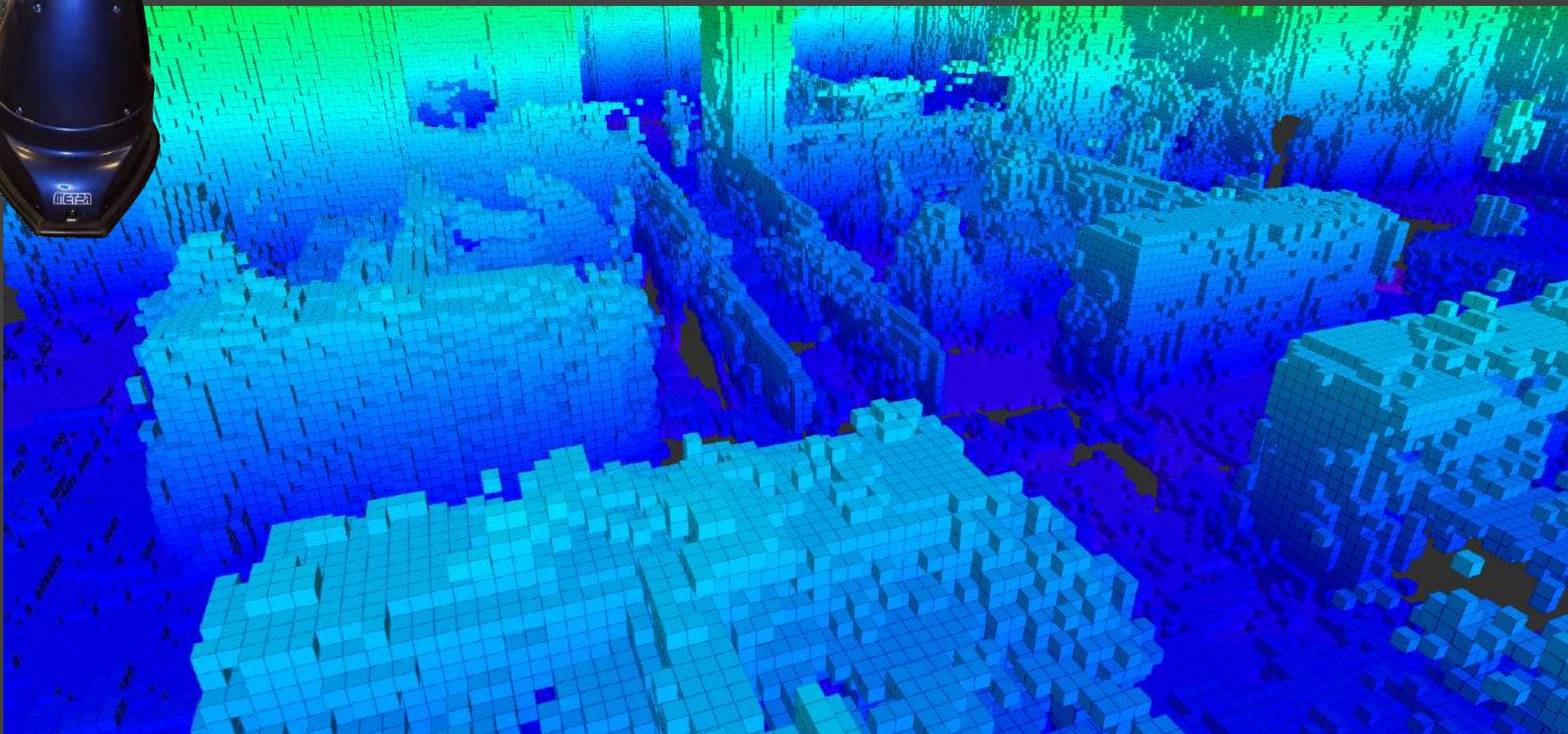
Record of the location of objects in an absolute coordinate system

Metric Maps

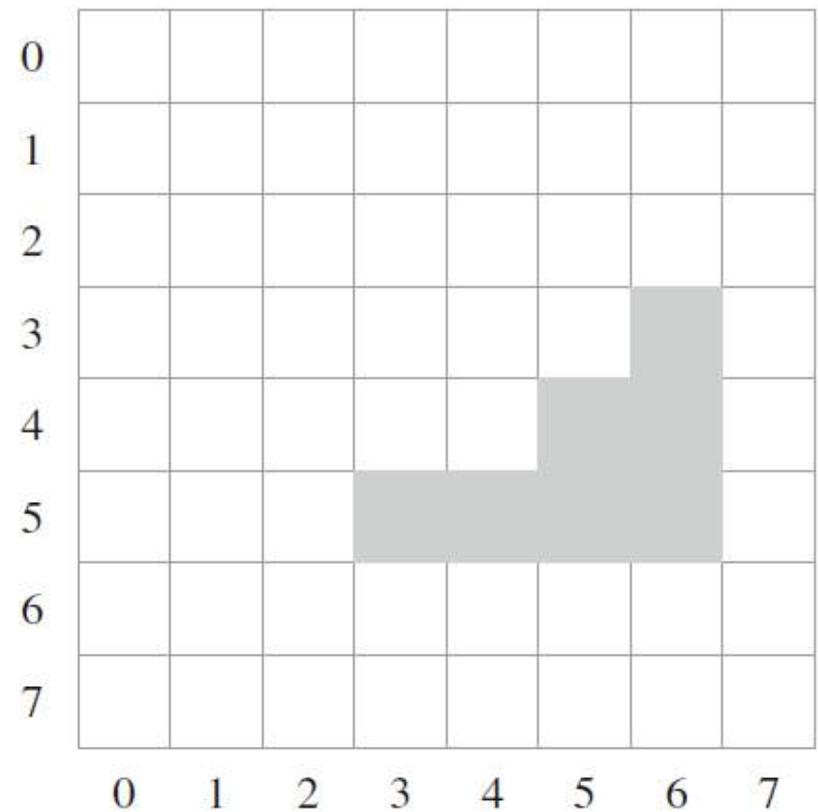
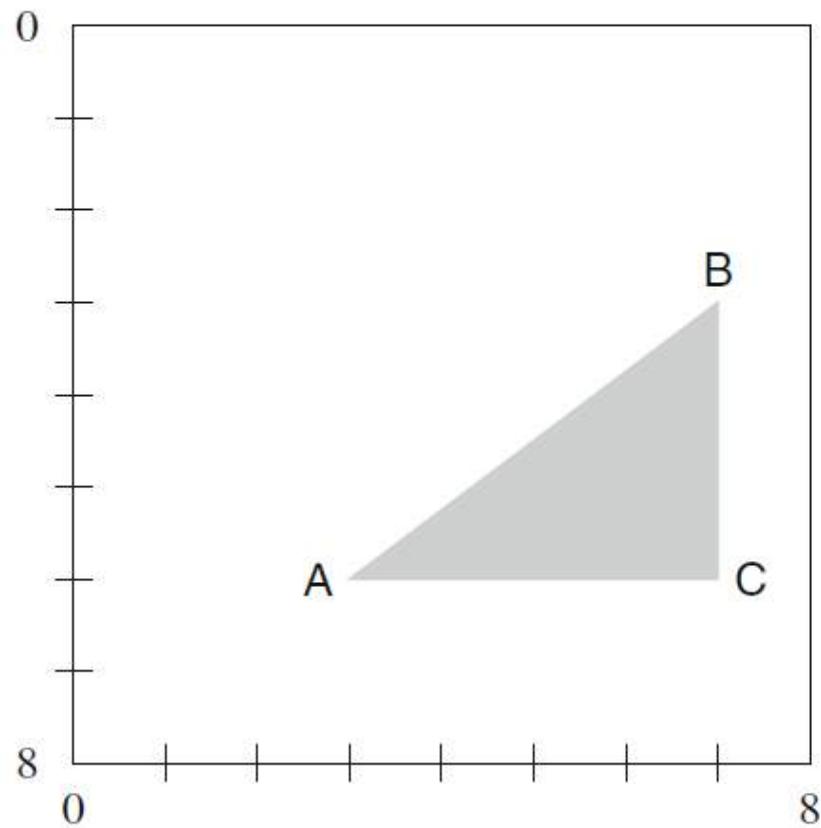
- Continuous / “vector” format
 - Points, linear or curved segments, surface patches
- Discrete / “raster” format
 - Occupancy grids



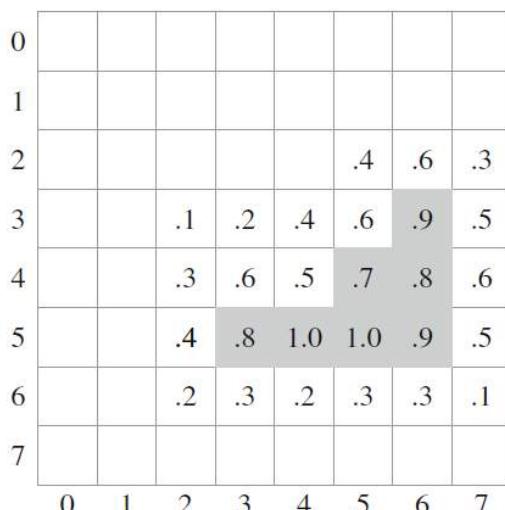
Linda's voxel map of Witham Wharf



Continuous vs Discrete Maps



Metric Maps – Occupancy Grids

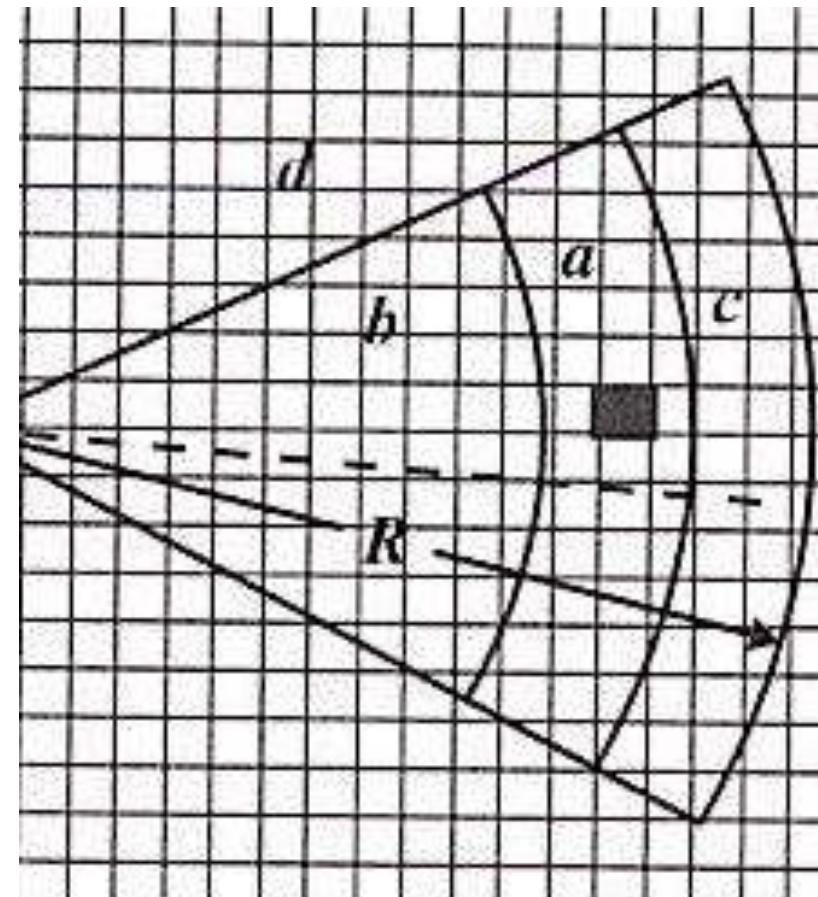


- Robot doesn't have complete and accurate prior knowledge about obstacles
- Each cell is associated with a probability that the cell is occupied, $P(\text{occ}_{x,y})$
- Updated using current robot pose (x , y , θ) and depth measurements from range-finder sensors, e.g. sonar, laser, stereo-vision

Model of Sonar Beam

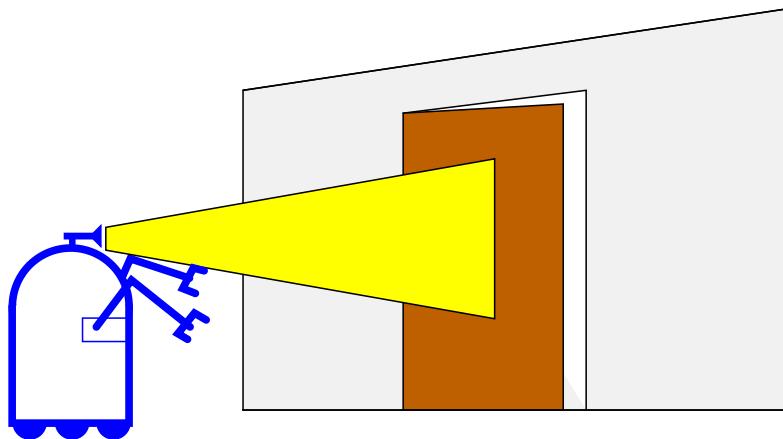
We need a model of the range sensor, e.g. for sonar we would define the following regions for a range measurement R :

- Probably occupied
- Probably empty
- In the shadow of the detected object, so status unknown
- Outside the beam, so status unknown.



Probabilistic robotics

- Explicit representation of uncertainty using the calculus of probability theory



$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z)}$$

Reasoning with probabilities

- Probabilistic reasoning formalizes the process of accumulating evidence and updating probabilities based on new evidence.
- *Prior* probability – belief before the new evidence
- *Posterior* probability – belief after the new evidence

Bayes' Rule

- General formula for Bayes' Theorem (discrete case)

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}$$

- (expresses the relation between a conditional probability and its inverse)
- Or, another way of writing it:

$$P(A | B) = \frac{1}{C} P(B | A)P(A)$$

Bayes' Rule

- The quantities in Bayes rule are often described as follows:

$$P(A | B) = \frac{1}{c} P(B | A)P(A)$$

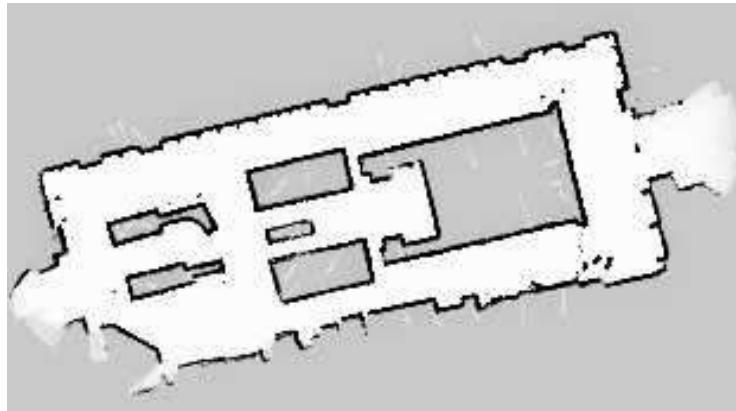
posterior probability

prior probability

likelihood

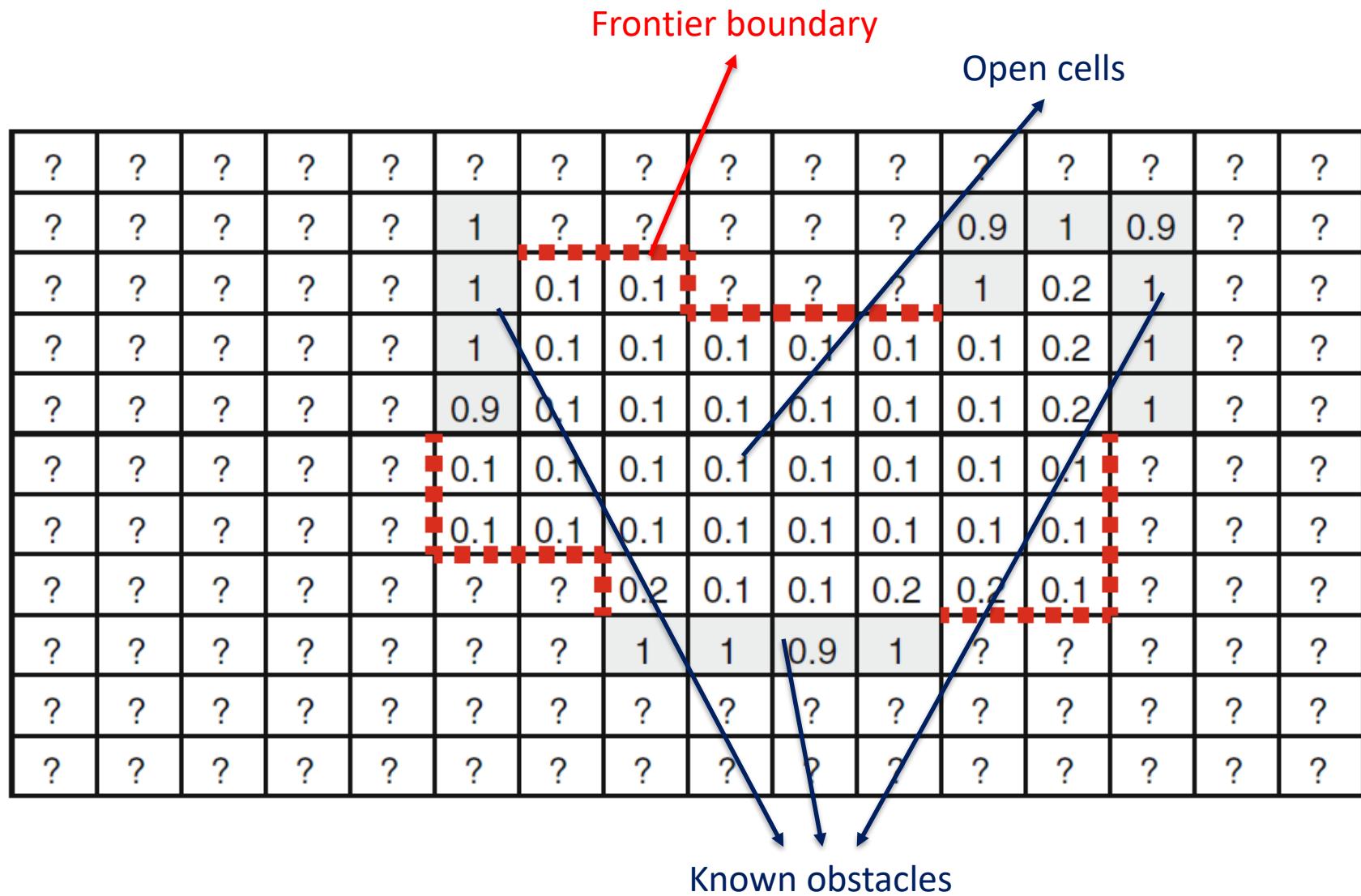
normalization factor

Example Occupancy Grid

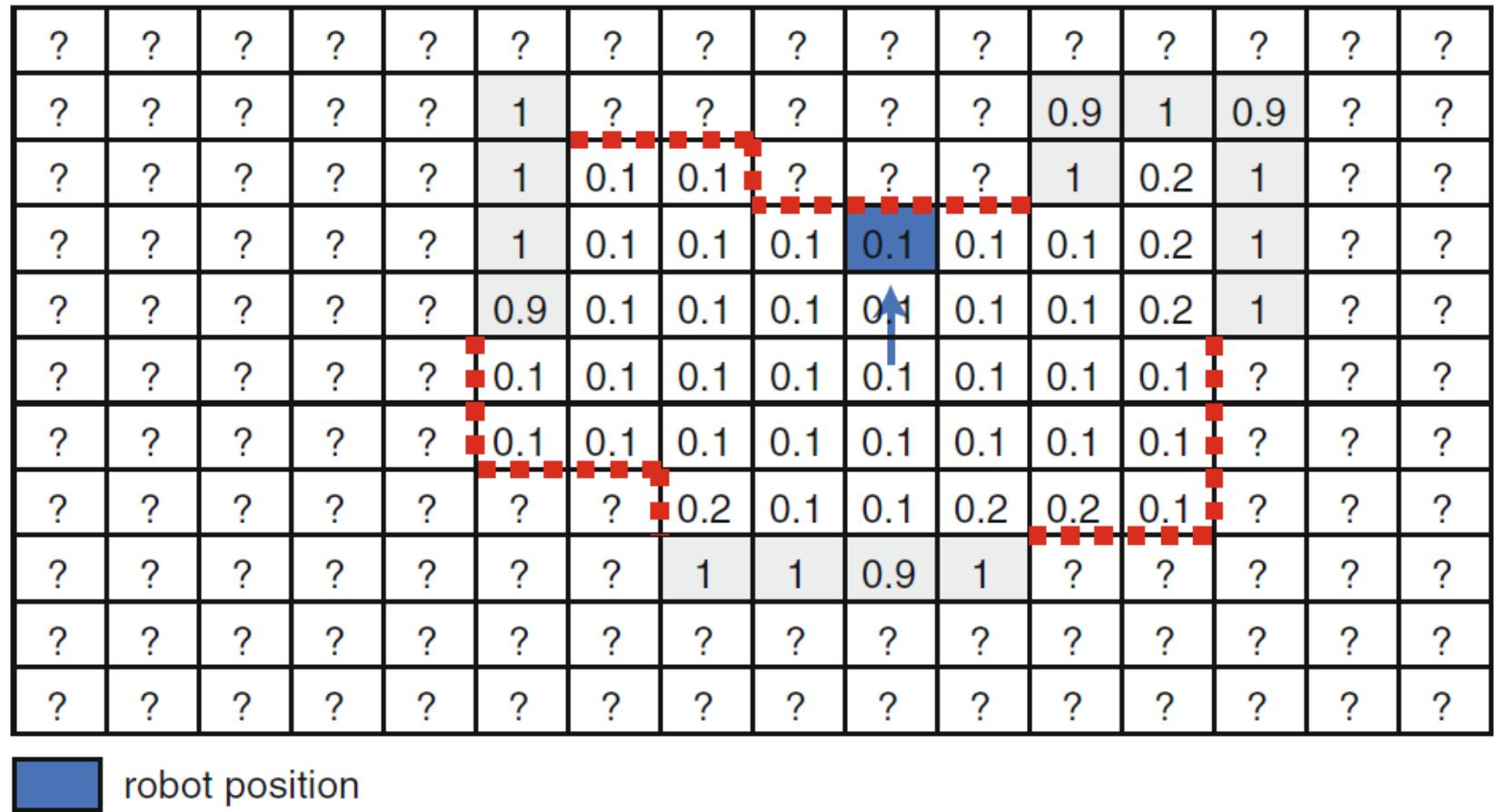


- 0 indicates that the cell has not been hit by any ranging measurements (free space)
- 1 indicates that the cell has been hit one or multiple times by ranging measurements (occupied space)
- Can change over time (e.g. dynamic obstacles)

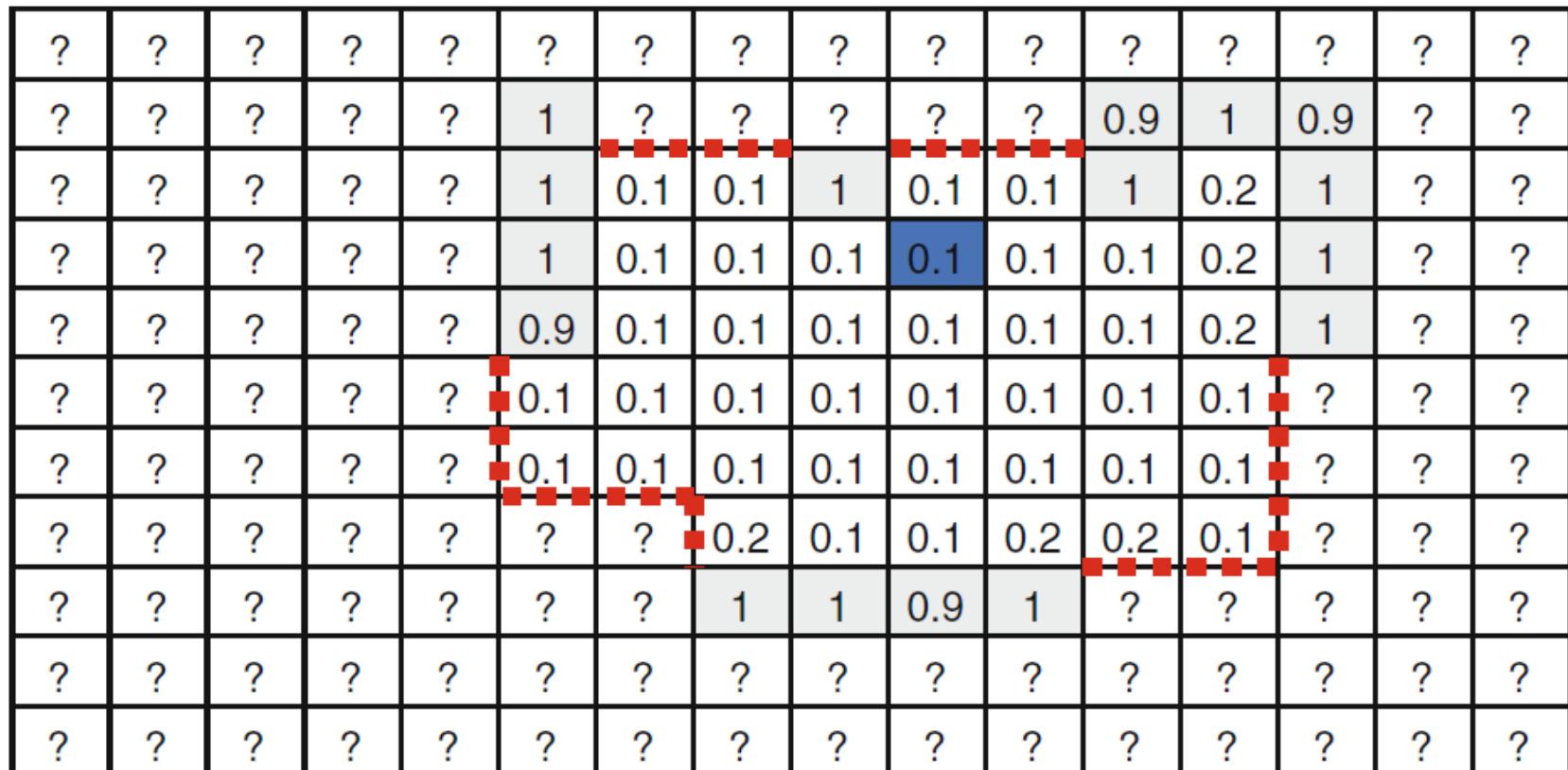
Frontier Algorithm – Grid Maps with Occupancy Information



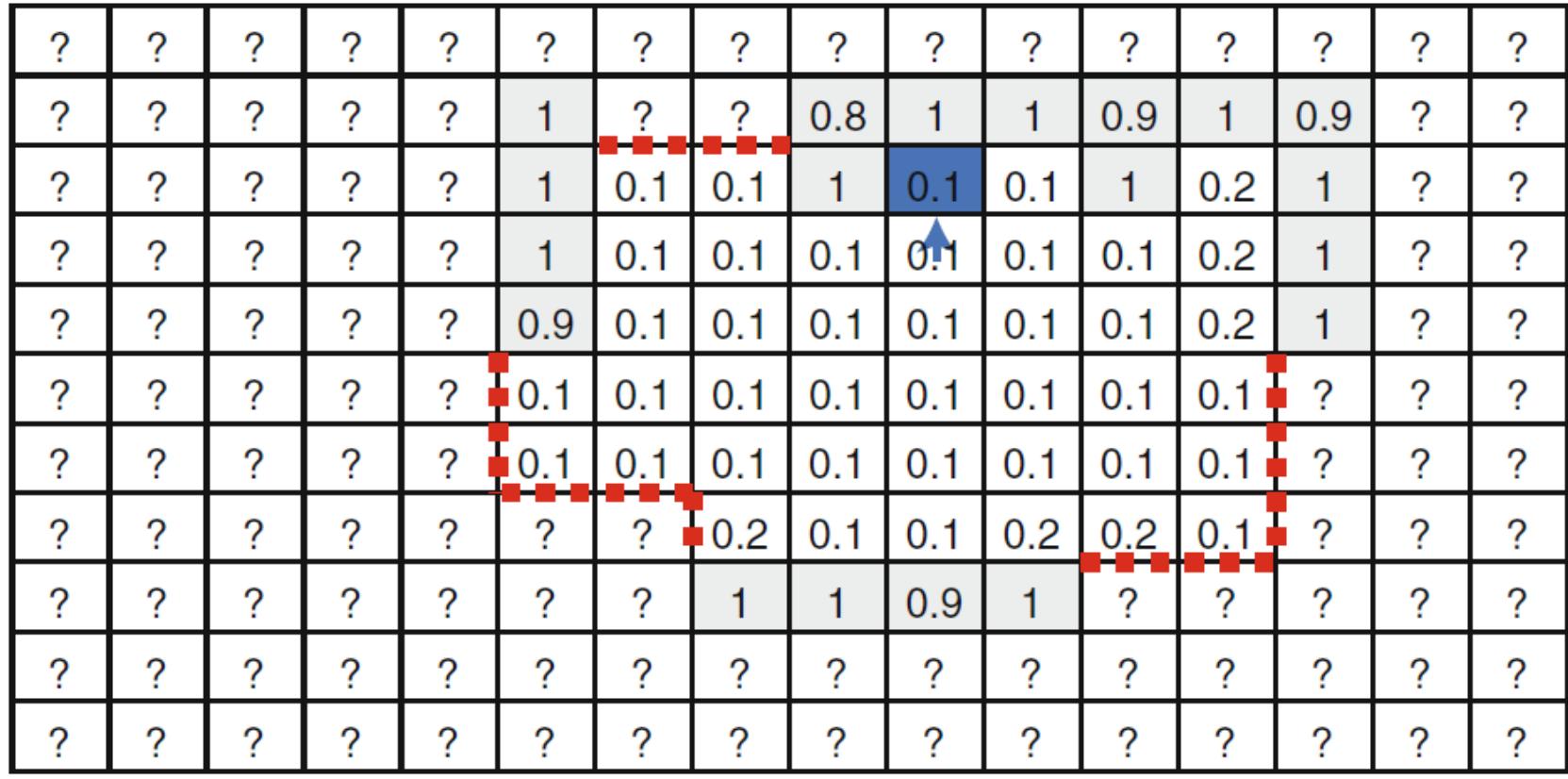
Frontier Algorithm – Grid Maps with Occupancy Information



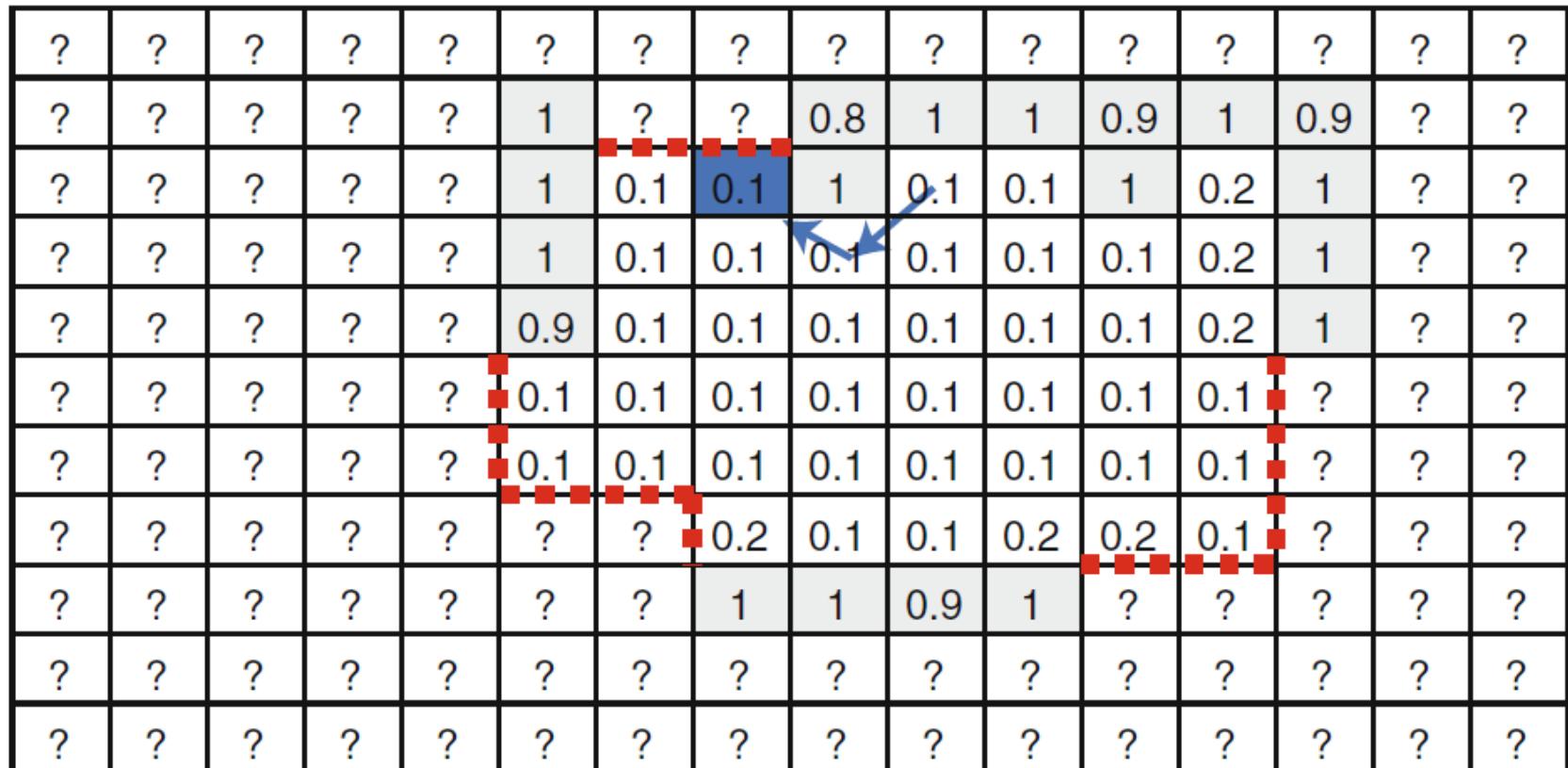
Frontier Algorithm – Grid Maps with Occupancy Information



Frontier Algorithm – Grid Maps with Occupancy Information

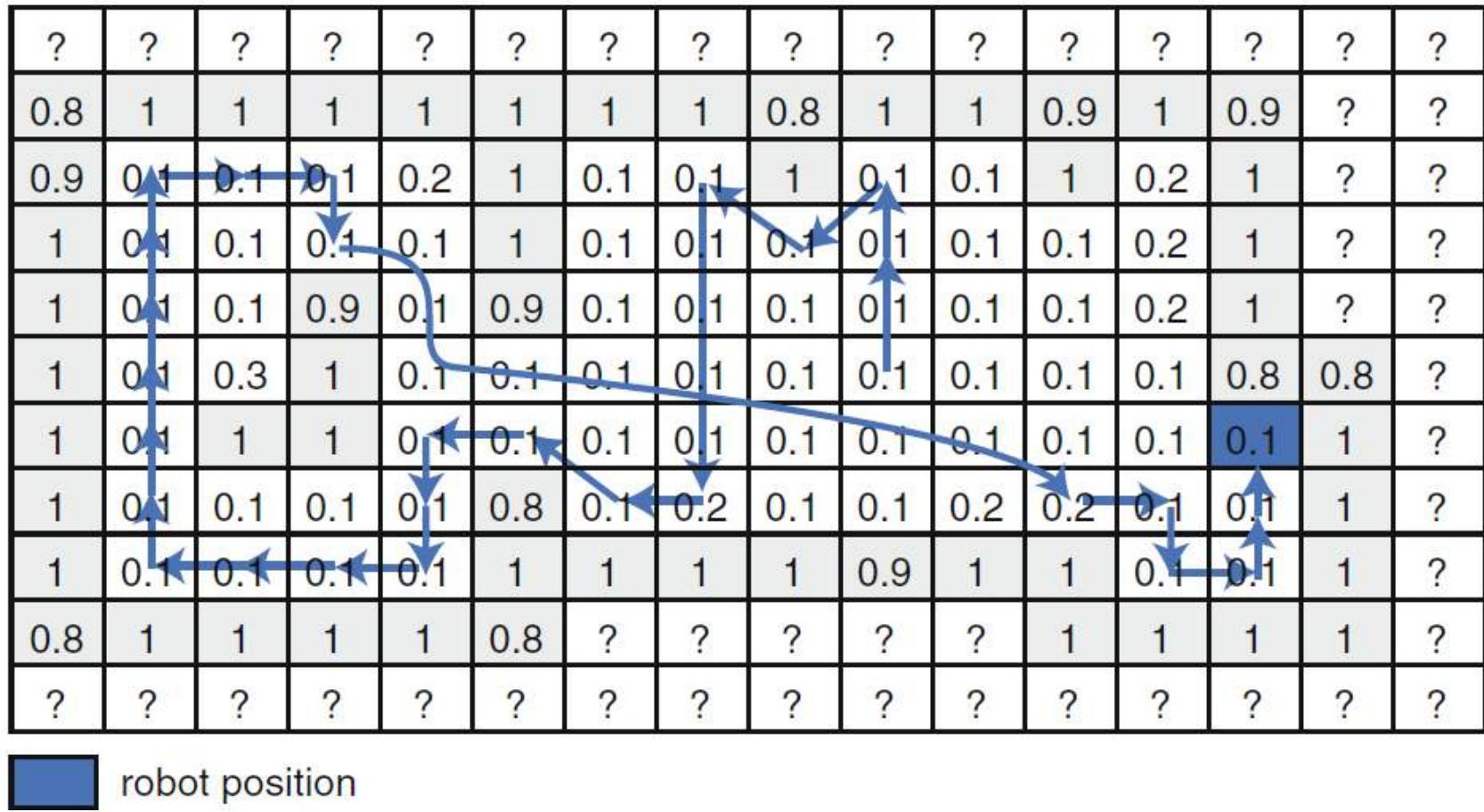


Frontier Algorithm – Grid Maps with Occupancy Information



robot position

Frontier Algorithm – Grid Maps with Occupancy Information

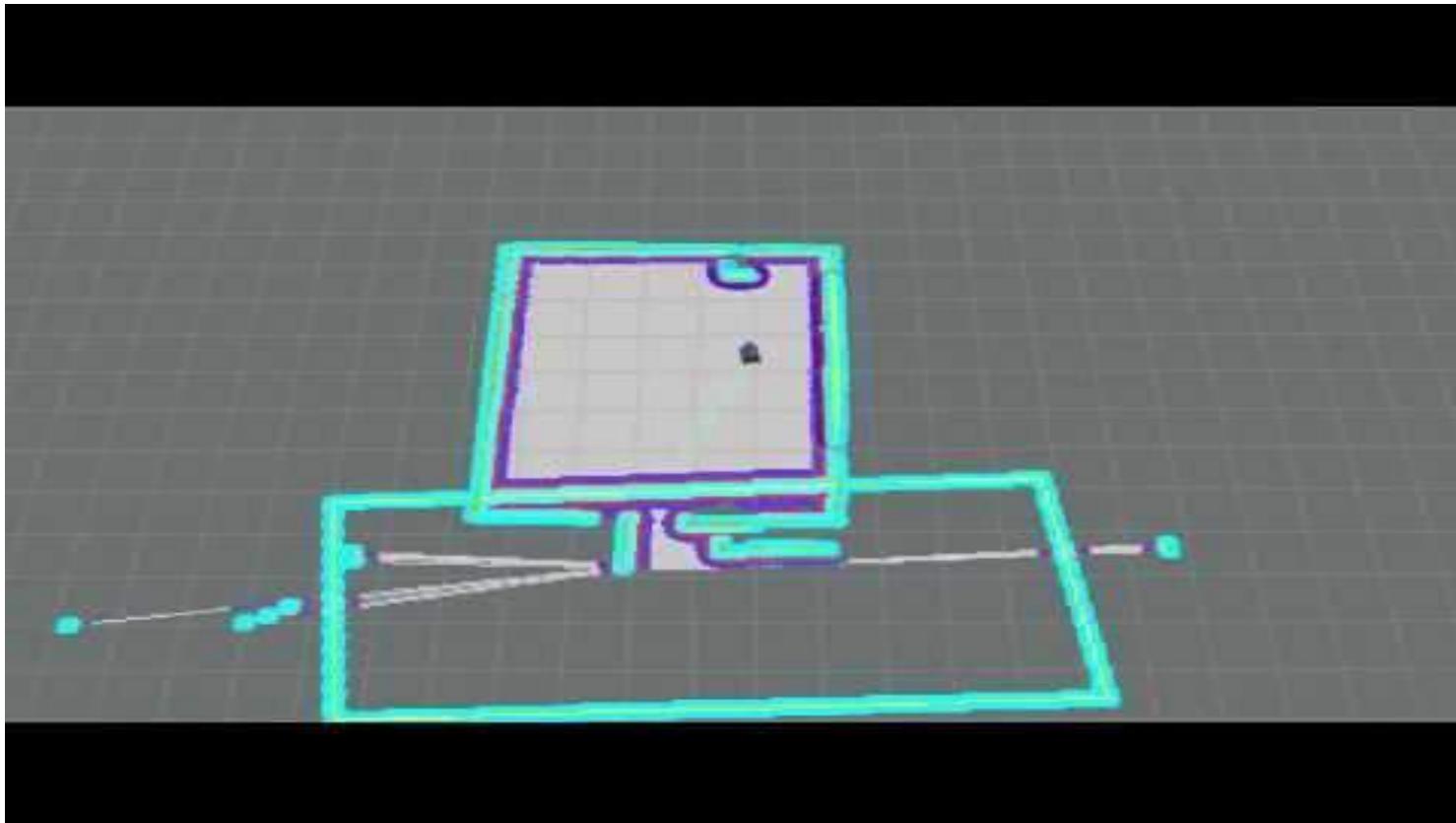


Priority of a frontier cell

- Distance
- The number of unknown cells adjacent to a frontier cell

Frontier Exploration

- http://wiki.ros.org/frontier_exploration

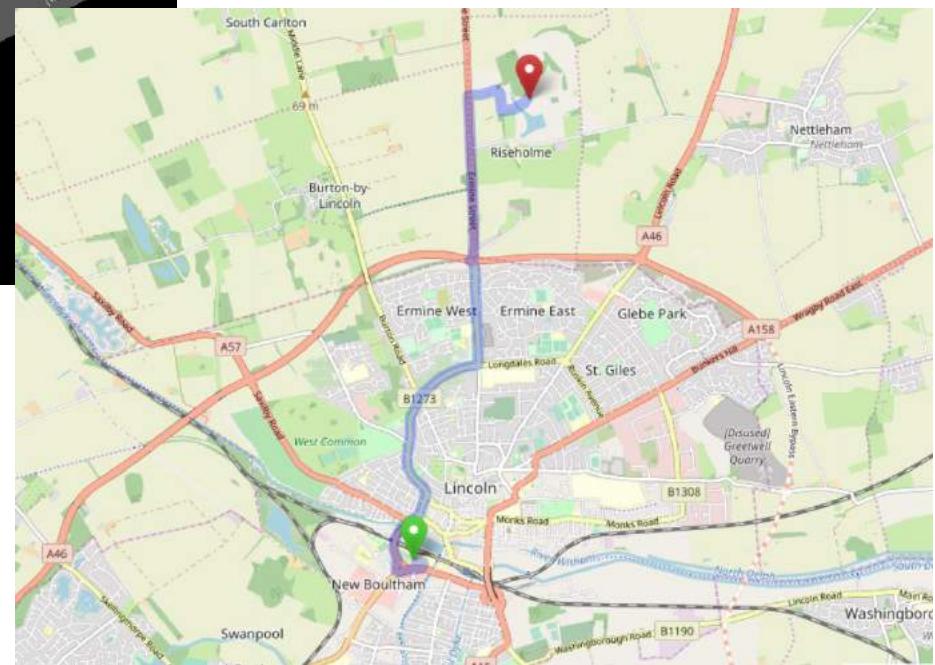
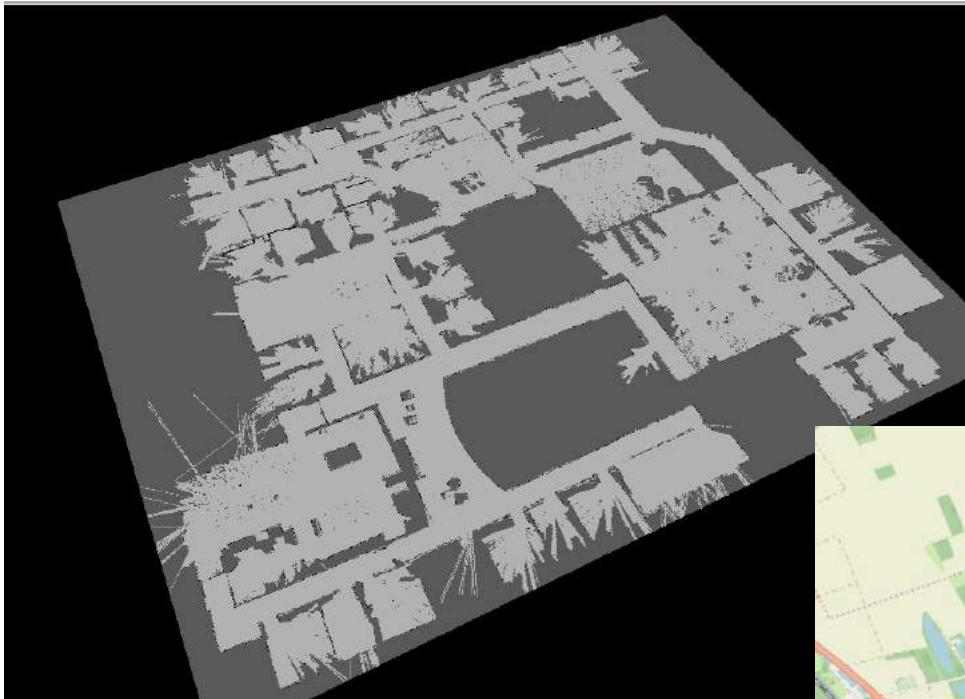




TOPOLOGICAL MAPS

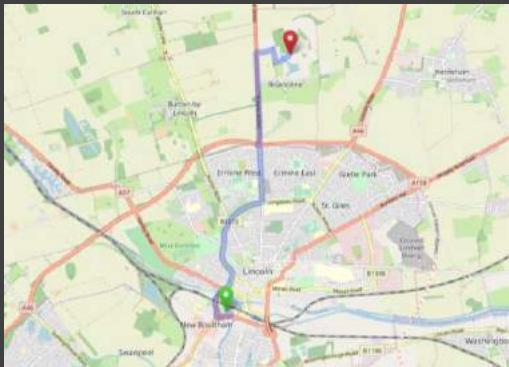
Represent known locations and the connections between them a set of nodes and edges in a graph

Complex, large, structured environments?

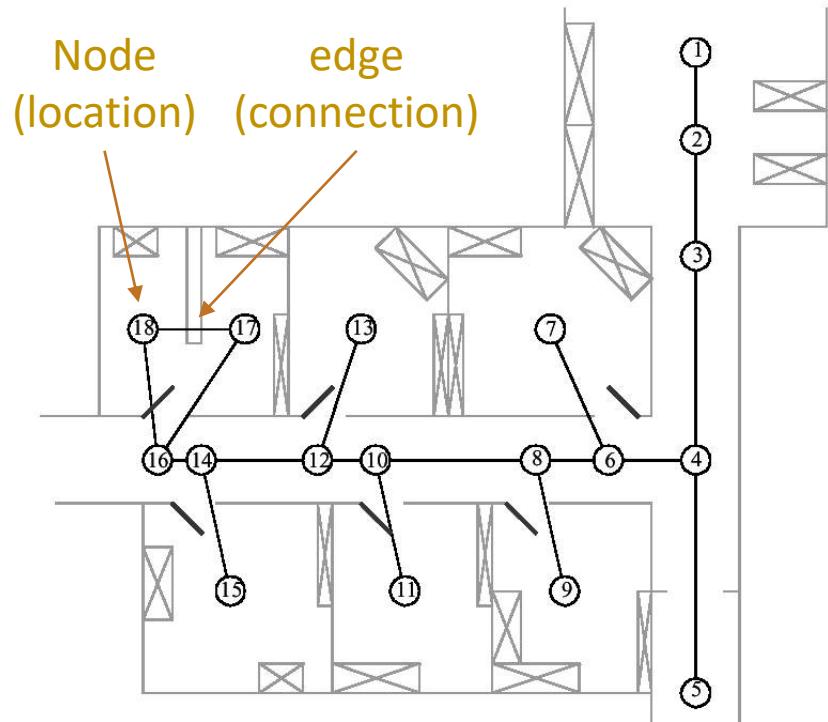


Complex, large, structured environments?

- specific navigation behaviours depending on the location.
- planning complexity over such a gridmap may be huge



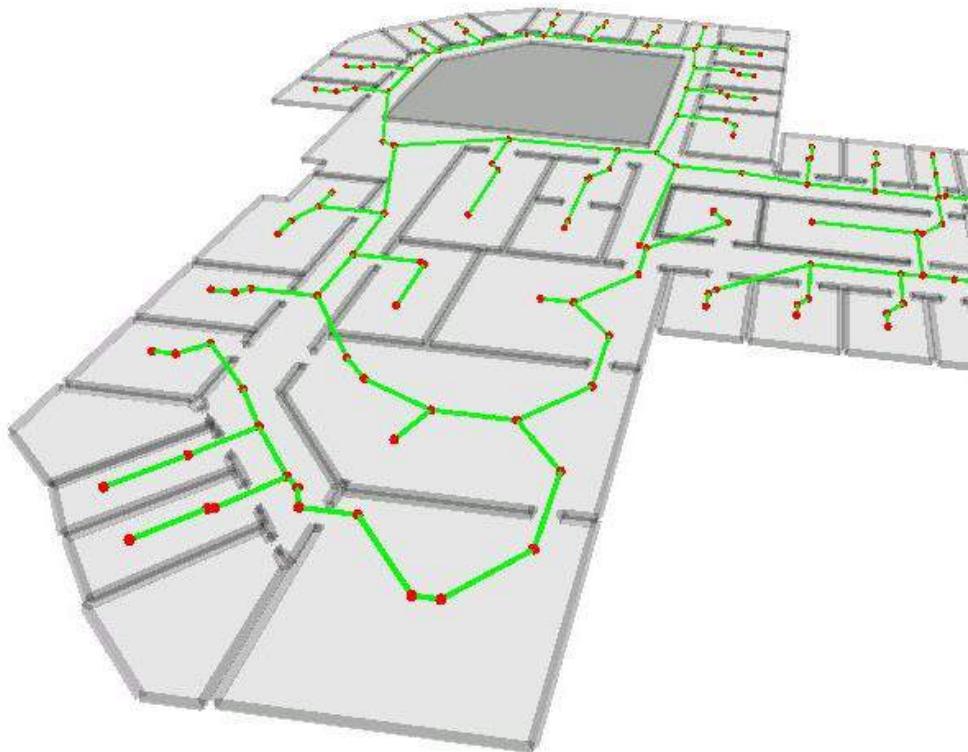
Topological Maps



- A topological map represents the environment as a graph with nodes and edges.
 - Nodes correspond to spaces
 - Edge correspond to physical connections between nodes
- Topological maps lack scale and distances, but topological relationships (e.g., left, right, etc.) are maintained

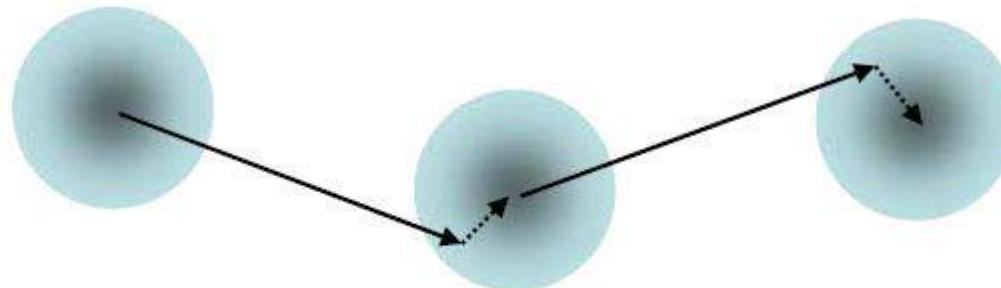
Topological Maps

- Can represent known locations and the connections between them as **nodes** and **edges** in a **graph**
- The edges could represent **actions** needed to get from one node to the next, or direction and distance
- Can determine a route by standard graph search methods (A*, Dijkstra) if distances (or costs) are added to the edges



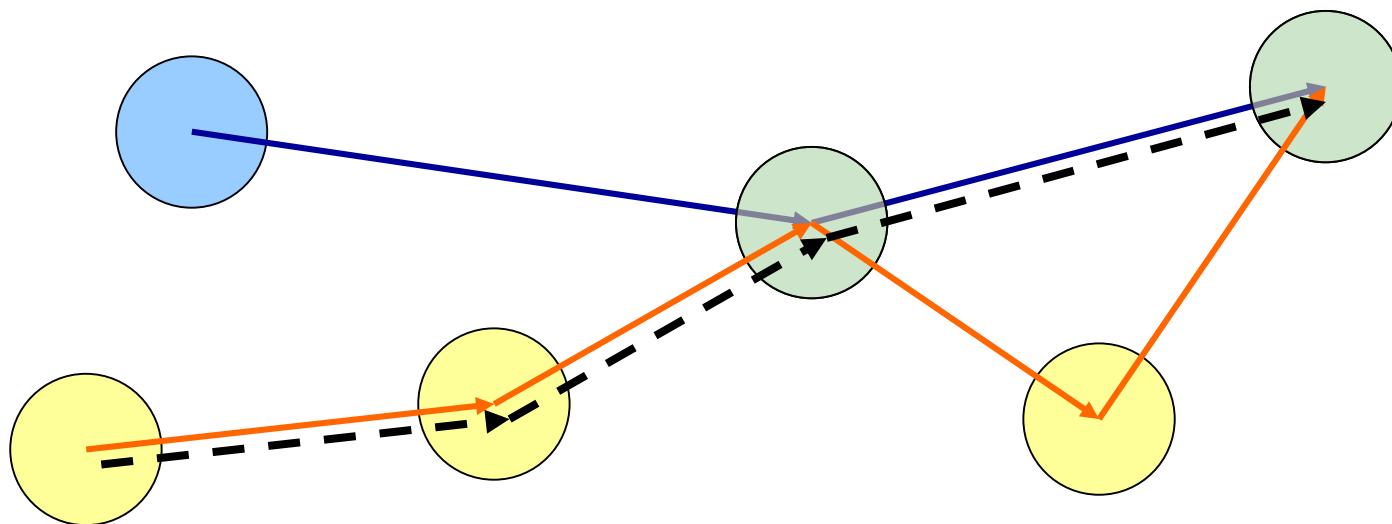
From local strategies to routes...

- Local strategies
 - Target location can be found from surrounding catchment area using a memory of the target location
 - Outbound route can be used to calculate vector direction to return to starting point
- Multiple memories and/or vectors can be linked together to form route memories



... to Topological Maps

- If we can combine routes by recognising overlapping locations from different routes, we have information we can use for finding novel routes
 - (often considered as a defining property of maps vs. routes)



Constructing A Topological Map

- New node = new place.
- Two nodes are connected when travelling from one node to another (unless already connected)
- Localisation: use adjacency information in the graph. Given tracked position, search is limited to the nodes in the adjacency.

Metric vs. Topological Maps

Metric Maps

- Detailed, quantitative, “sub-symbolic” representation
- Good for representing (and avoiding) known, static obstacles
- High computational cost of storage and processing
- Require very accurate position tracking → reliance on accurate odometry and range-finder sensors
- How to determine an appropriate resolution?

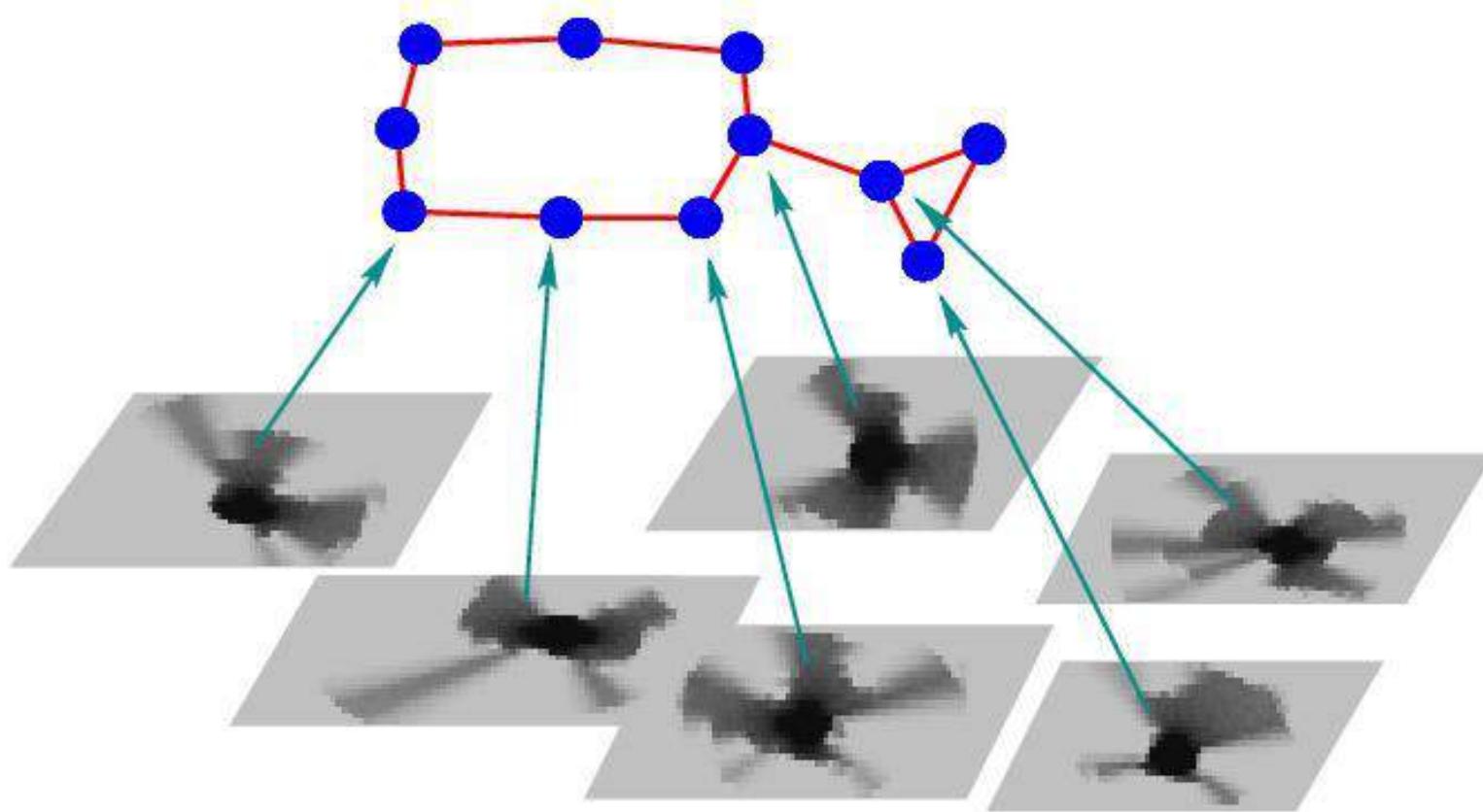
Topological Maps

- Abstract, qualitative, “symbolic” representation
- May be more persistent/robust to environment dynamics
- Low computational cost → efficient path planning, scale better to large environments
- Require accurate place recognition → problem of perceptual aliasing (what if 2 or more places look alike?)
- How to determine what makes a “place” ?



SEMANTIC MAPS

Record of semantic information (metadata), includes segmentation, function etc.

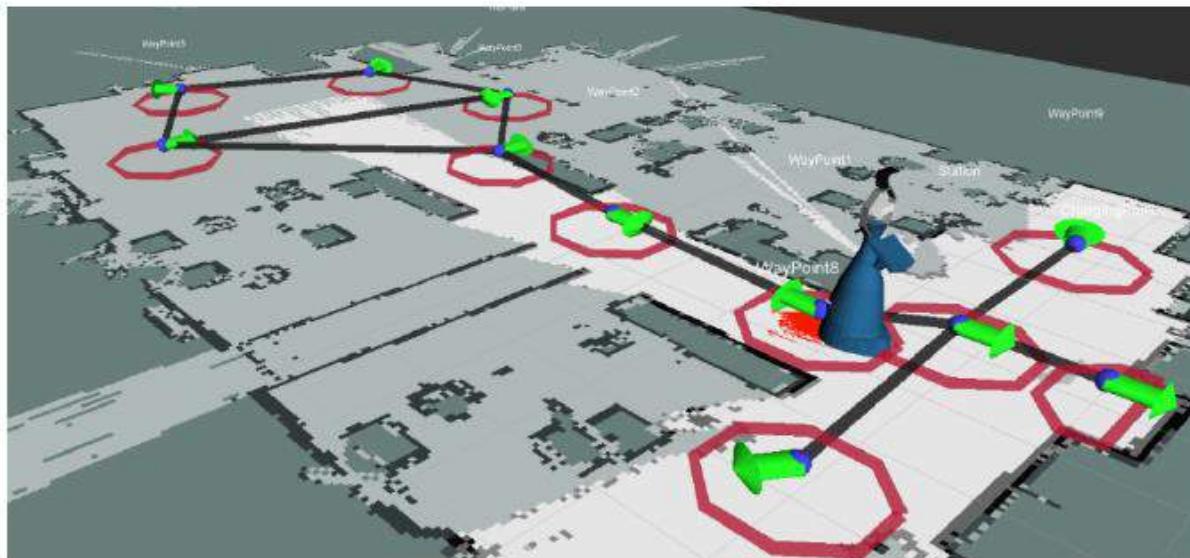


HYBRID MAPS

combine the complementary strengths of different representations (metric, topological, and semantic maps)

Hybrid Maps

- Attempt to combine the complementary strengths of different representations (metric, topological, and semantic maps)
- Example of a hybrid metric-topological map
 - Topological level: connected set of places
 - Metric level: each place is associated with a local metric map

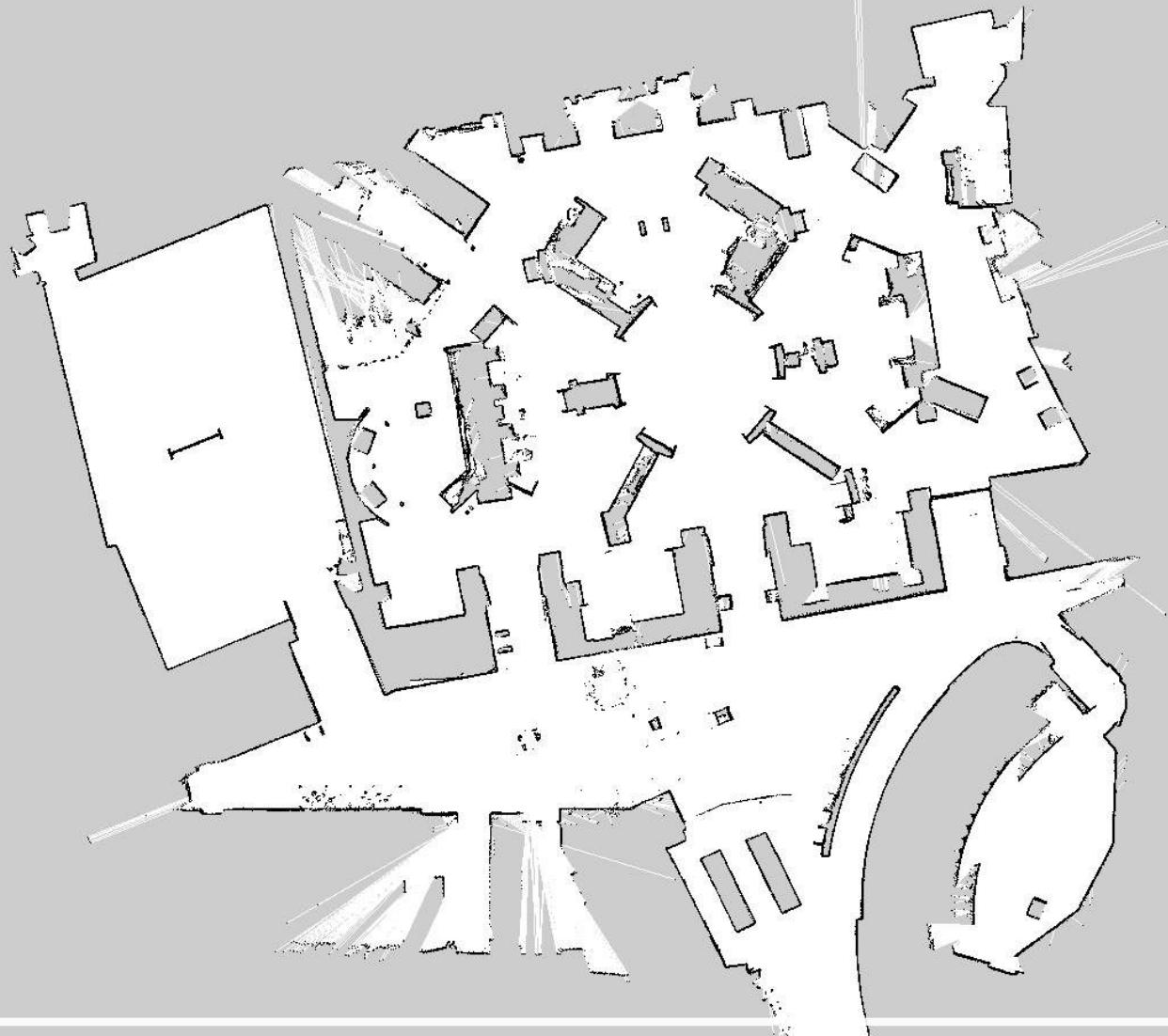




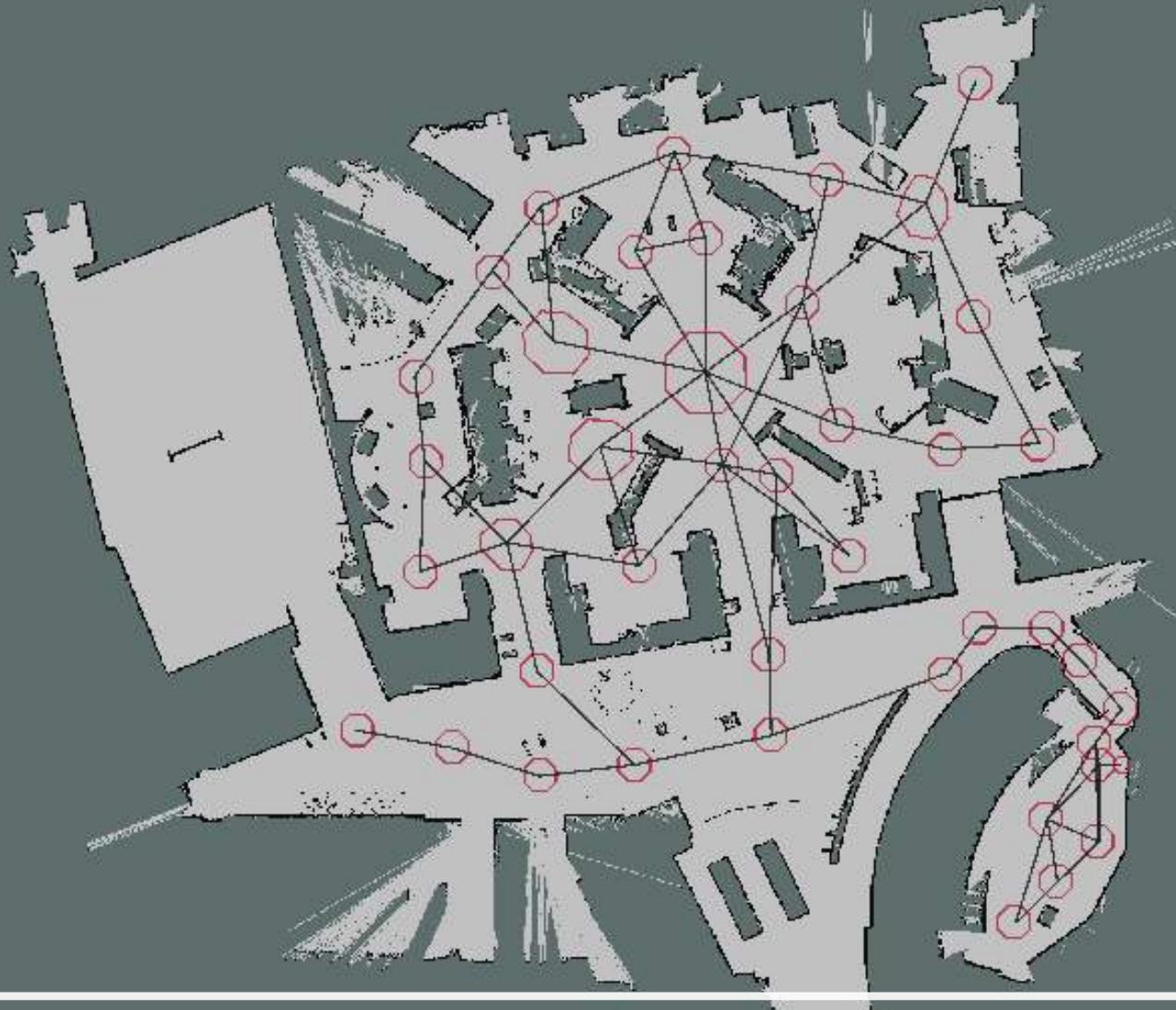
Lindsey at the Collection

<https://cityx.co.uk/2018/11/meet-lindsey-your-new-robotic-tour-guide-at-the-collection/>

https://www.youtube.com/watch?v=XRocFBA28Is&fbclid=IwAR0UJiZu_XyPaJaIR33OpCivRG147IZtaIRKR-R0SyaEm3mkJPtRbOZQ4zc



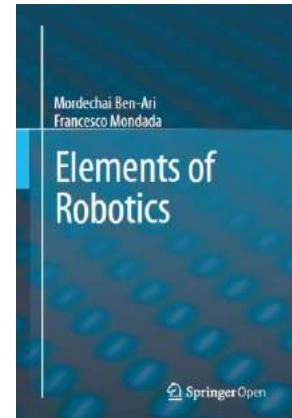
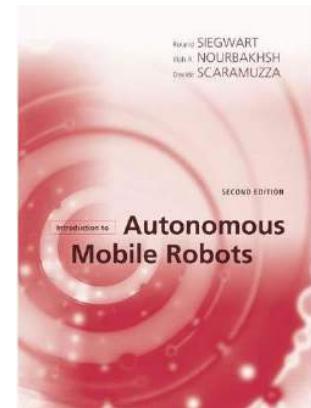
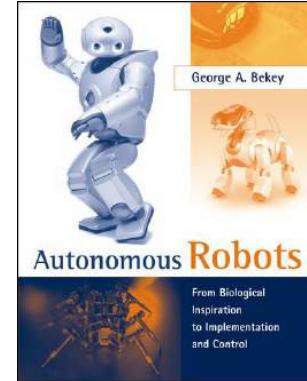
Lindsey's obstacle map at the Collection



Lindsey's navigation map at the Collection

Suggested reading

- Bekey, G.A., *Autonomous Robots: From biological inspiration to implementation and control*, (MIT Press). Chapter 14.
- Siegwart R. et al., *Autonomous Mobile Robots*, (MIT Press). Chapter 5.
- Ben-Ari, Mordechai, and Francesco Mondada. *Elements of robotics*. Springer International Publishing, 2018. Chapter 9.





Questions?

CMP3103M/CMP9050M

Autonomous Mobile Robotics

Dr Ayse Kucukyilmaz

University of Lincoln
Centre for Autonomous Systems

INB3201

akucukyilmaz@lincoln.ac.uk
<http://webpages.lincoln.ac.uk/akucukyilmaz/>



UNIVERSITY OF
LINCOLN

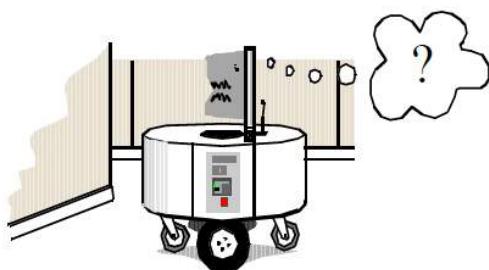


Syllabus

1. Introduction (MH)
2. Robot Programming (MH)
3. Robot Sensing (MH)
4. Motion and Control (MH)
5. Robot Behaviours and Navigation (AK)
6. Localisation and Mapping (AK)
7. **Self-Localisation (AK)**
8. Planning (AK)
9. Control Architectures (PB)
10. Human Robot Interaction 1 (PB)
11. Human Robot Interaction 2 (PB)
12. Applications (PB)



Key questions



- **Where am I?**
- Where do I go?
- How do I go there?
- To answer these, the robot needs
 - A model of the environment (either given or autonomously built)
 - To perceive and understand the environment
 - **Find its position and situation in the environment**
 - Plan and execute the movement



**Based on external
sensors, beacons,
landmarks**



Odometry



Map-based

Only proprioceptive
sensors

No external sensors
and landmarks

Only onboard sensors

Localisation: How to?



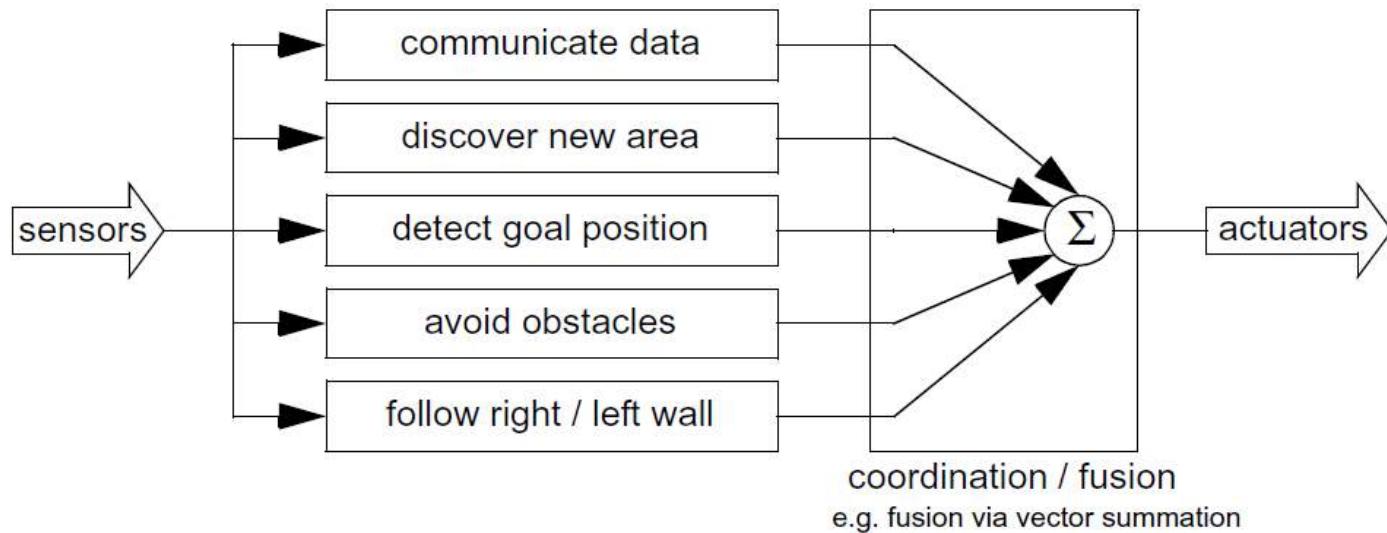
Odometry



- **Odometry**: The process of calculating vehicle's current position by using a previously determined position and estimated speeds over the elapsed time
 - Odos: route; metron : measure
 - Also called **deduced reckoning** or **dead reckoning**
- Robot motion is recovered by integrating **proprioceptive** sensor velocities readings
 - Pros: Straightforward
 - Cons: Errors are integrated -> unbound
- Heading sensors (e.g., gyroscope) help to reduce the accumulated errors but drift remains :(

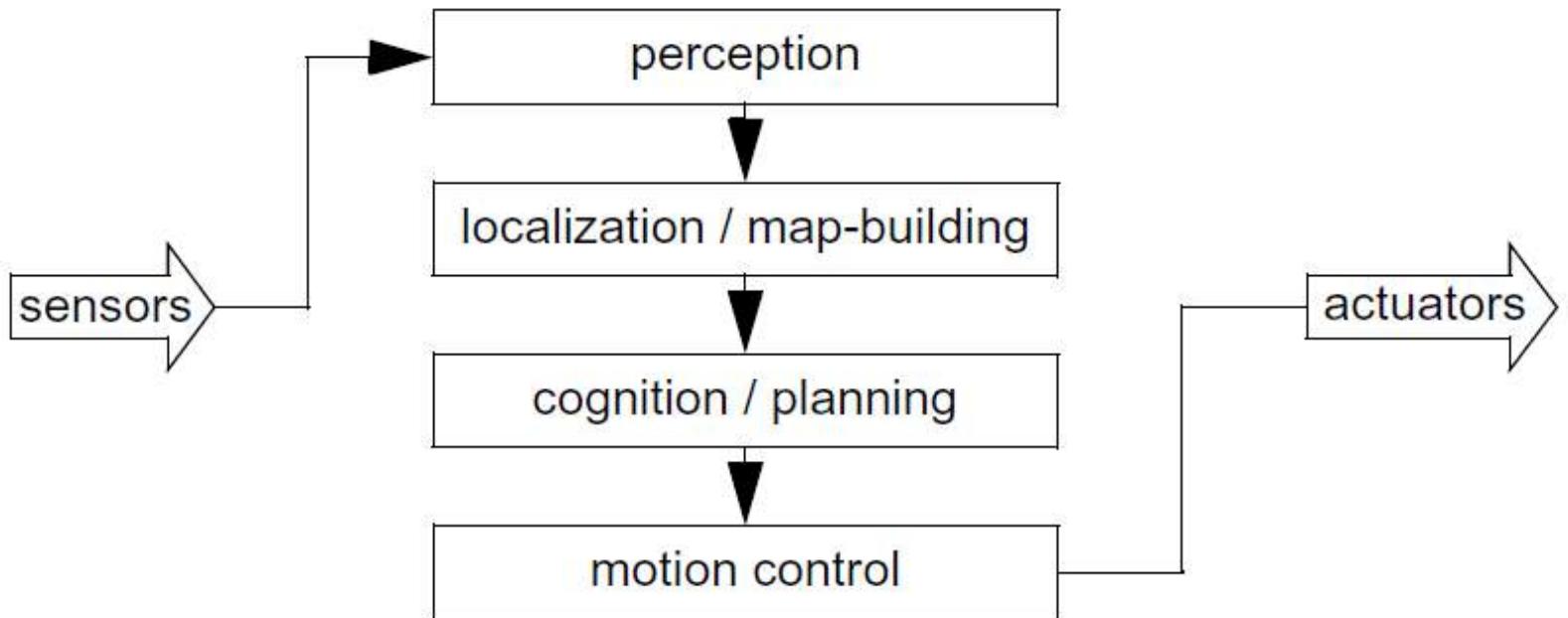
To localise or not to localize

A behavior-based approach



To localise or not to localize

A behavior-based approach



MAP-BASED LOCALISATION



Mobile robot self-localisation

- Often divided into 3 main problems:
 - Position tracking (good prior estimate)
 - Global localisation (no prior estimate)
 - “Kidnapped robot problem” (prior estimate is wrong)
- Particle Filters can address all of these cases
 - a.k.a. Monte Carlo localization

“So where am I?!”

(The problem of Self-Localisation)

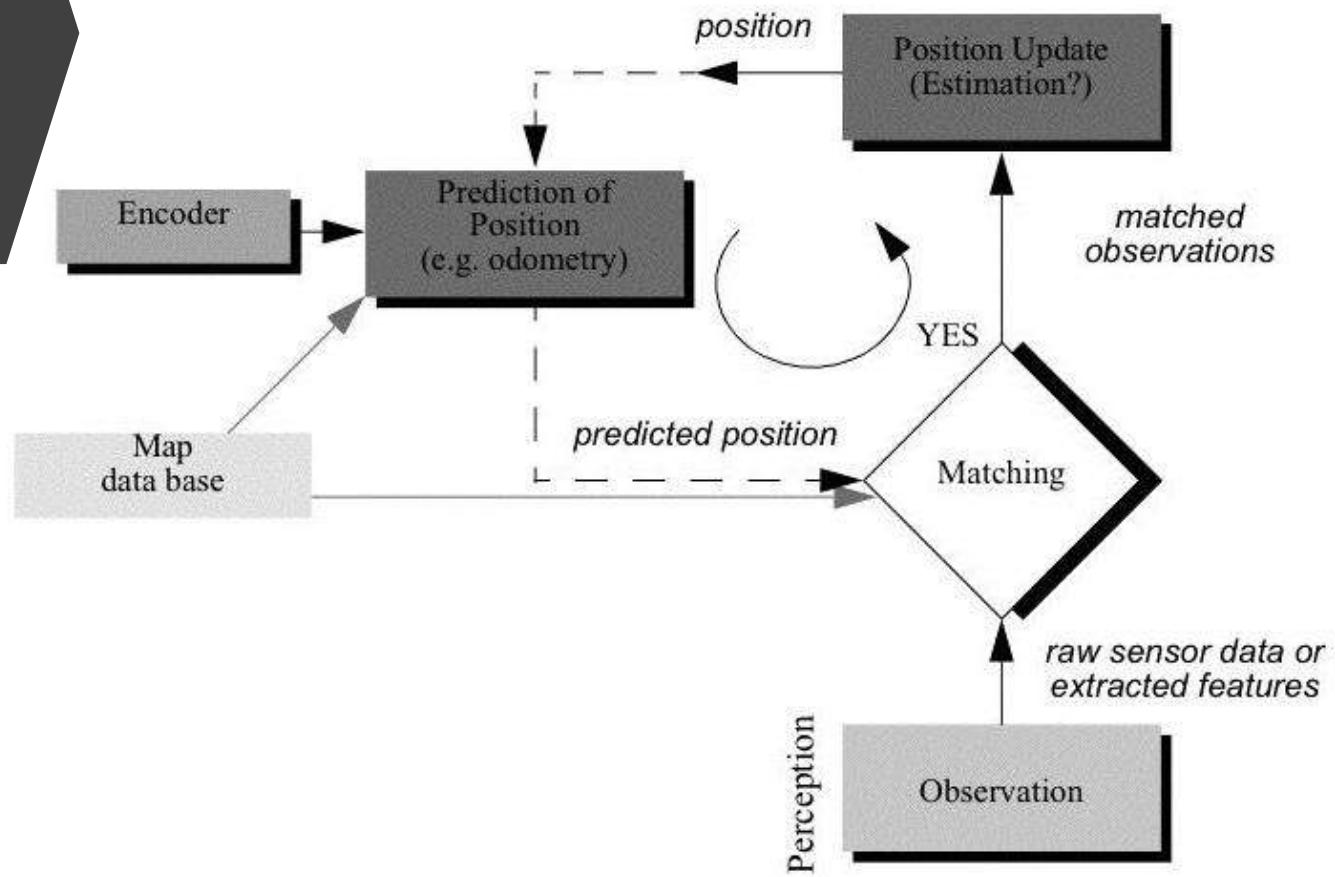
- a.k.a What’s wrong with odometry (just proprioception) ...



Solution: use a map! combine dead reckoning with sightings on known landmarks/environmental features

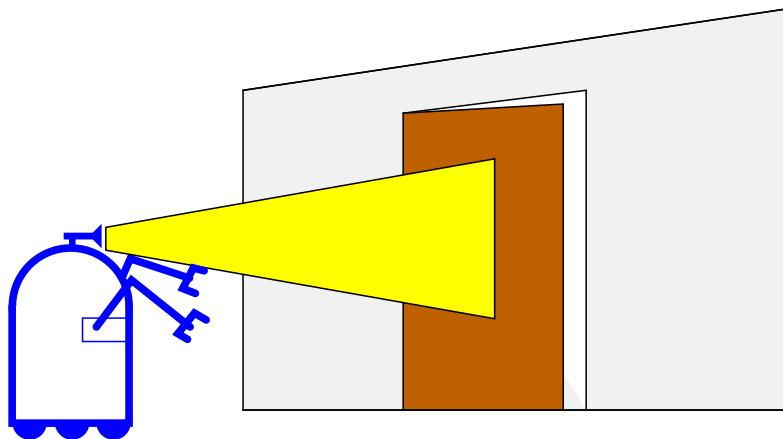
METRIC LOCALISATION

General process of map-based self-localisation



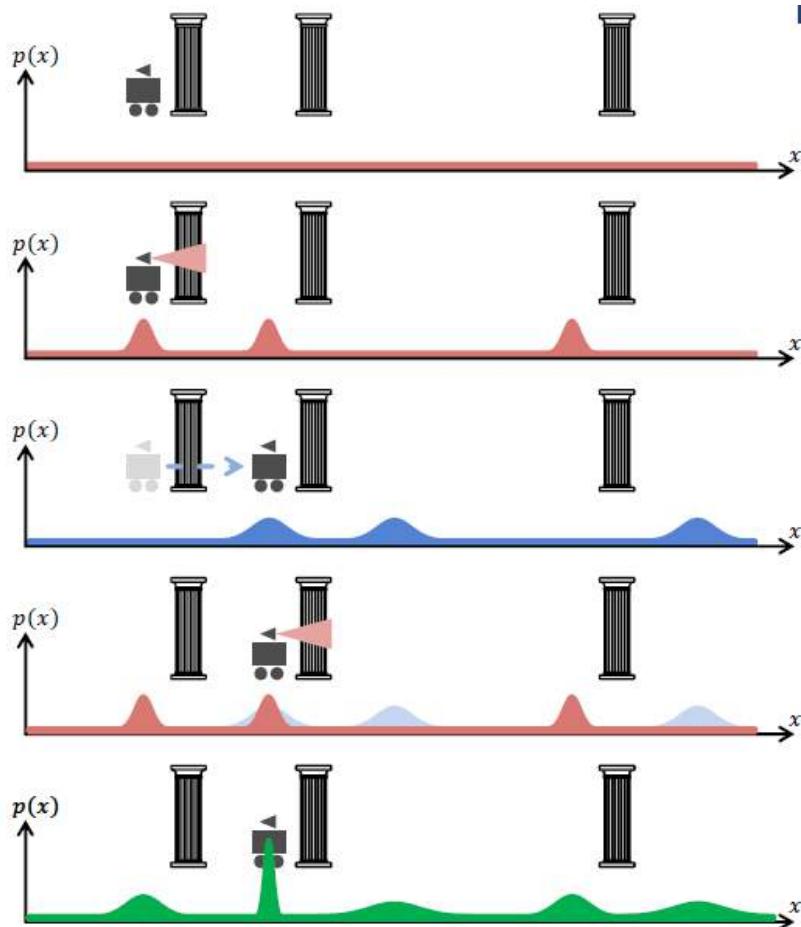
Probabilistic robotics

- Explicit representation of uncertainty using the calculus of probability theory



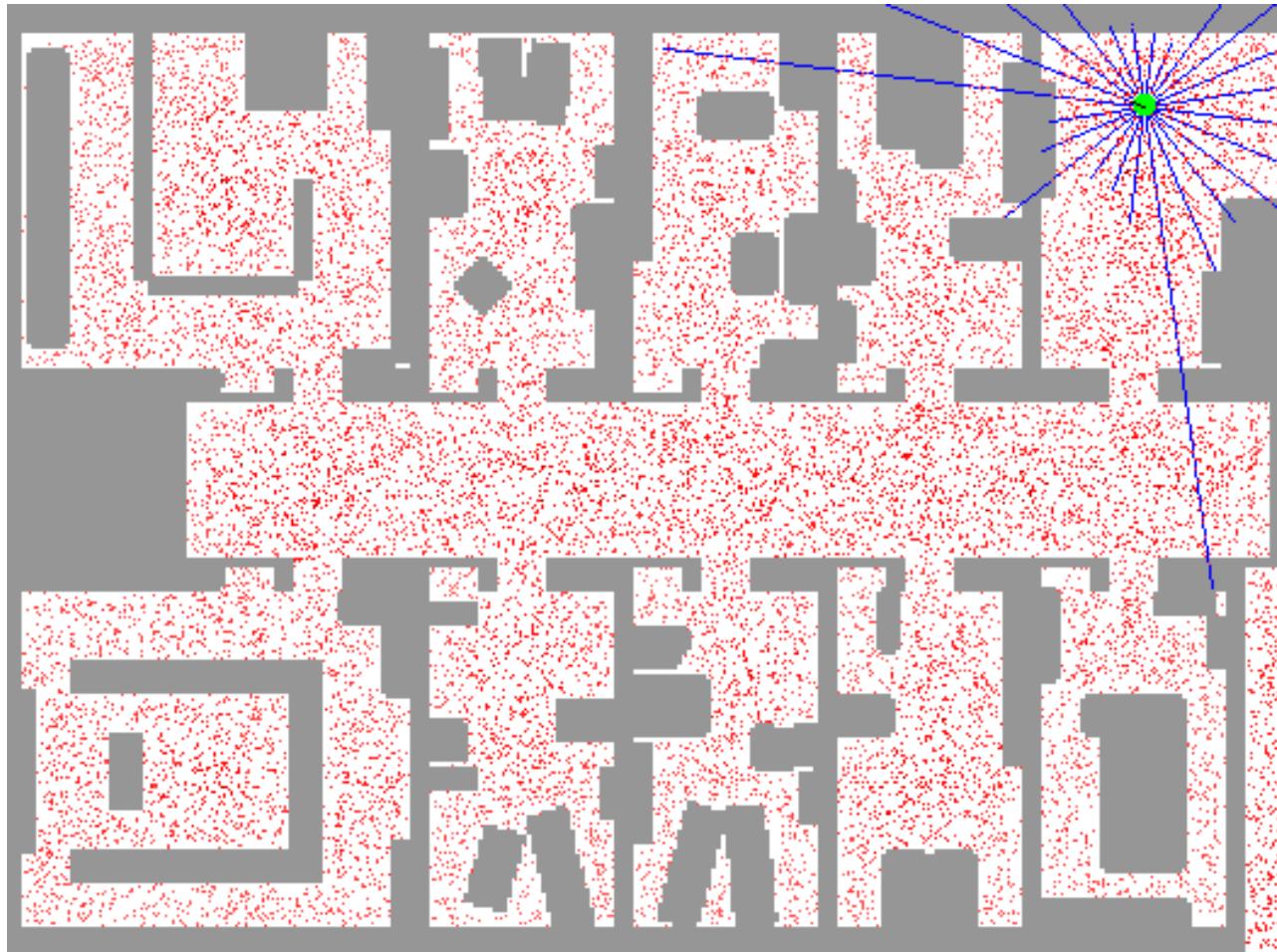
$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z)}$$

(Markov) Localisation: where am I?



- PERCEPTION (MEASUREMENT / CORRECTION) UPDATE: The robot queries its sensors
 - finds itself next to a pillar
- PREDICTION (ACTION) UPDATE: Robot moves one meter forward
 - Motion estimated by wheel encoder
 - Accumulation of uncertainty
- PERCEPTION UPDATE: The robot queries its sensors again
 - finds itself next to a pillar
- BELIEF UPDATE (information fusion)

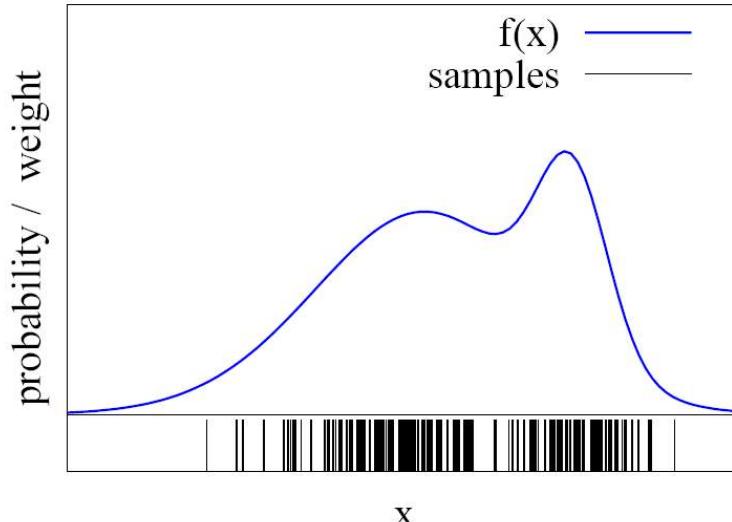
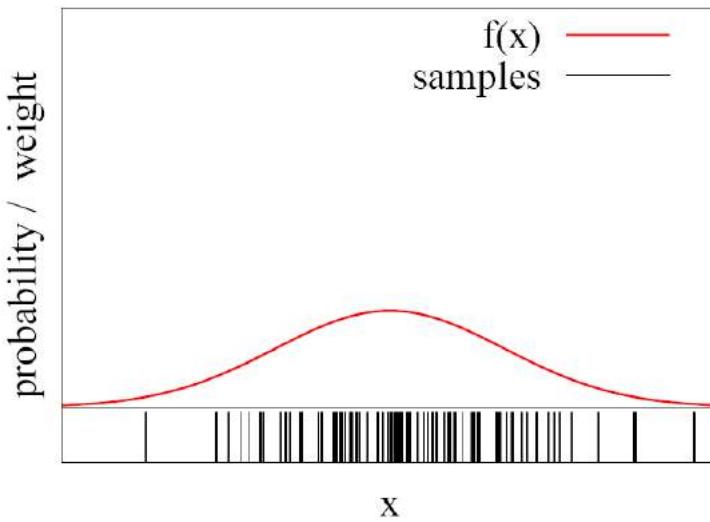
Randomised Sampling: 2D Monte Carlo Localisation



<http://mobilerobotics.cs.washington.edu/mcl/>

Function Approximation

- Particle sets can be used to approximate functions



- The more particles fall into an interval, the higher the probability of that interval

What is a particle?

- A particle is an individual state estimate
- A particle is defined by its:
 1. State values that determine the **robot's pose** (position and orientation), e.g. $[x, y, \theta]$ for 2D self-localisation “in the plane”
 2. A weight that indicates its **likelihood**
- Particle filters use many particles to represent the **belief state** (“where am I?”)

Start



Loop

Initialisation: sample from initial distribution

- No idea where robot is. Throw particles everywhere!

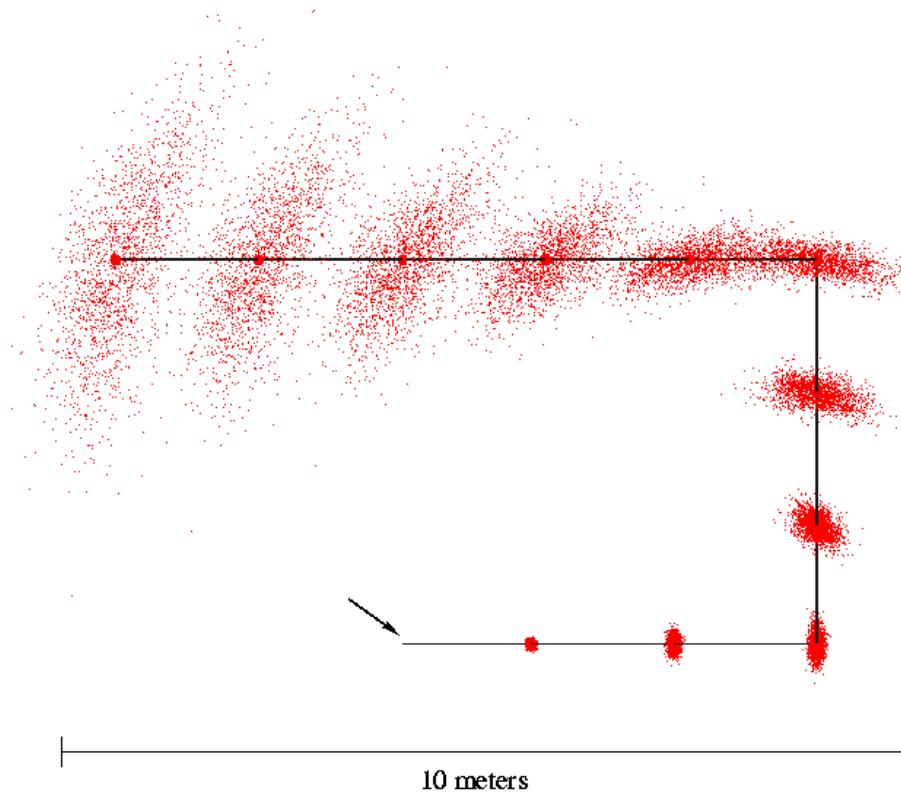
For each time step, loop with three phases:

- **Prediction**
- **Update**
- **Resample**

Monte Carlo Localisation (MCL)
Particle Filter Algorithm – Main Steps

1) Prediction step

- For each particle, sample and add random, noisy values from the motion model



Motion model (actions disperse the distribution...)

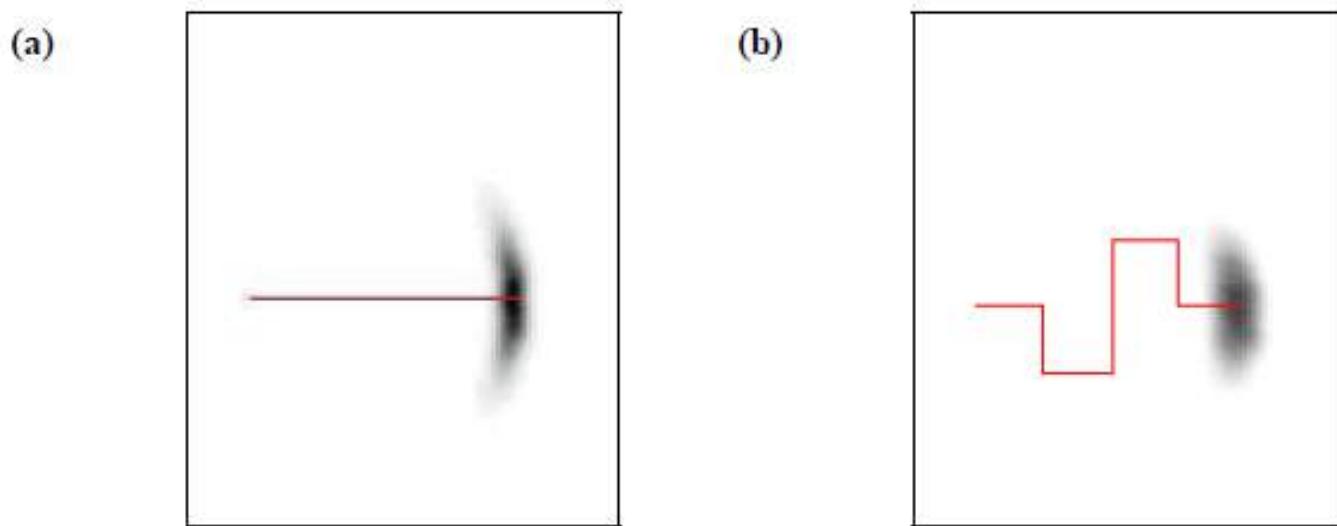
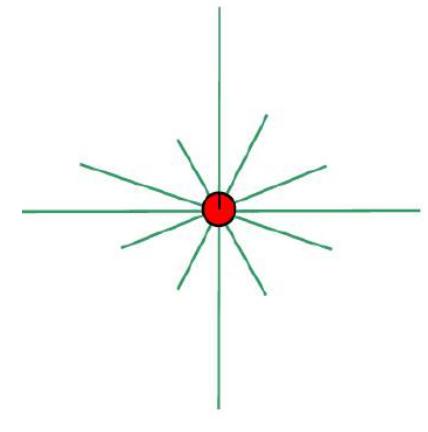
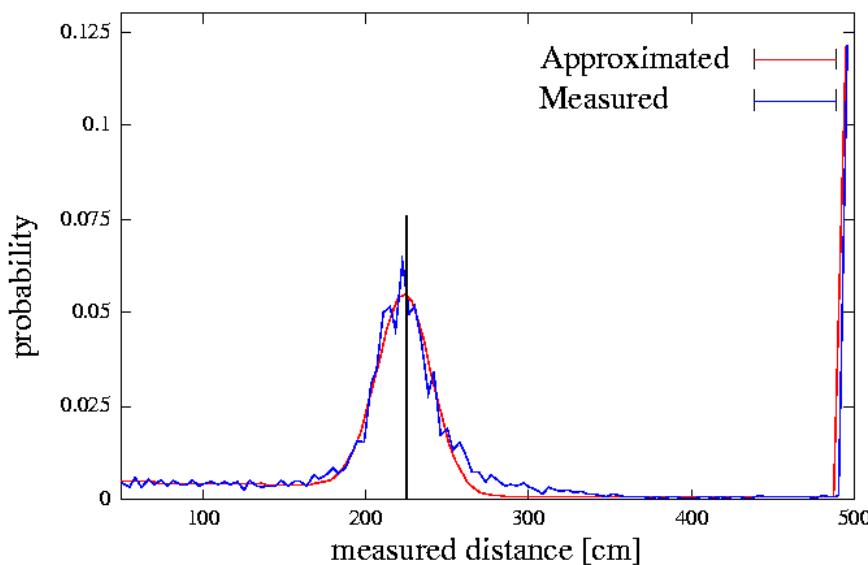


Figure 5.2 The motion model: Posterior distributions of the robot's pose upon executing the motion command illustrated by the solid line. The darker a location, the more likely it is. This plot has been projected into 2D. The original density is three-dimensional, taking the robot's heading direction θ into account.

S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

2) Update step

- Each particle's weight is the likelihood of getting the current sensor readings from that particle's hypothesis (compared to the predicted readings from the map)

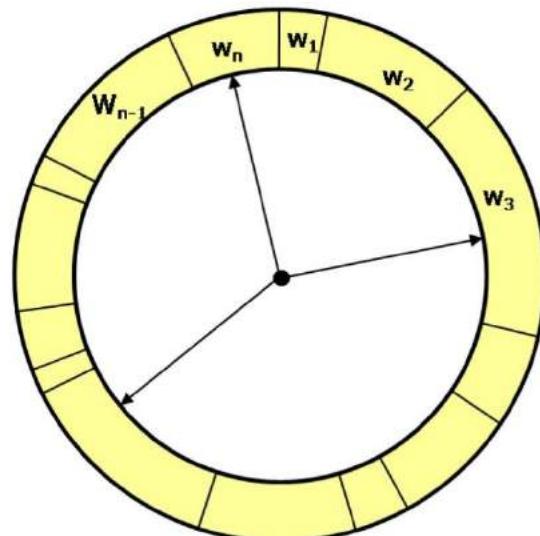


Laser sensor

3) Resample step

- A new set of particles is chosen such that each particle survives in proportion to its weight
- “Survival of the fittest”: replace unlikely samples by more likely ones

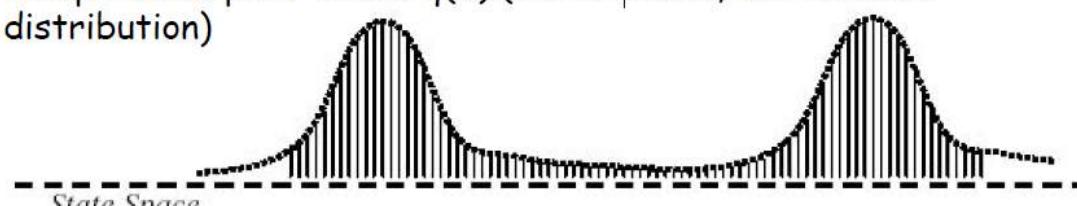
“Roulette wheel” resampling



Resampling



Sample from prior belief $q(x)$ (for instance, the uniform distribution)



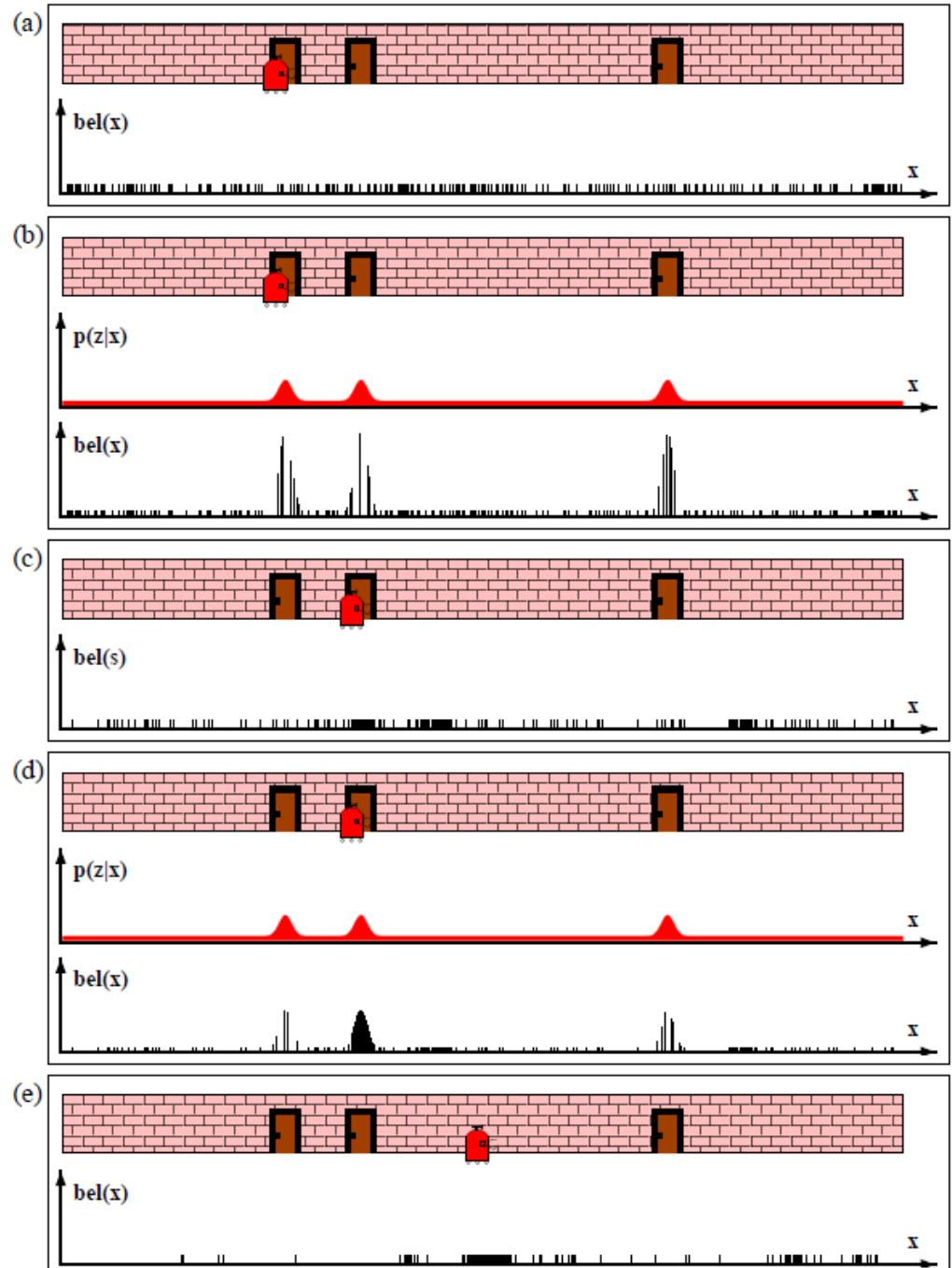
Compute importance weights, $w(x) = p(x) / q(x)$



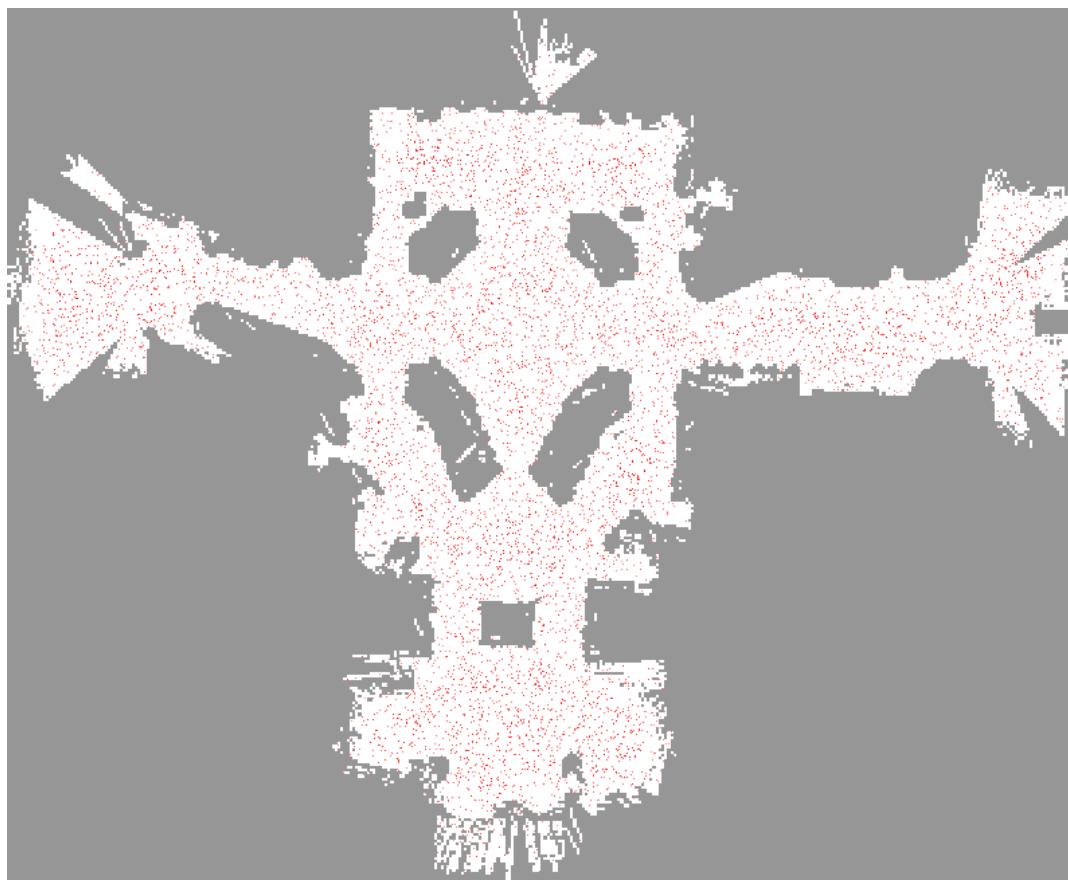
Resample particles according to importance weights to get $p(x)$
Samples with high weights chosen many times; density reflects pdf

- 1-dimensional localisation along a corridor

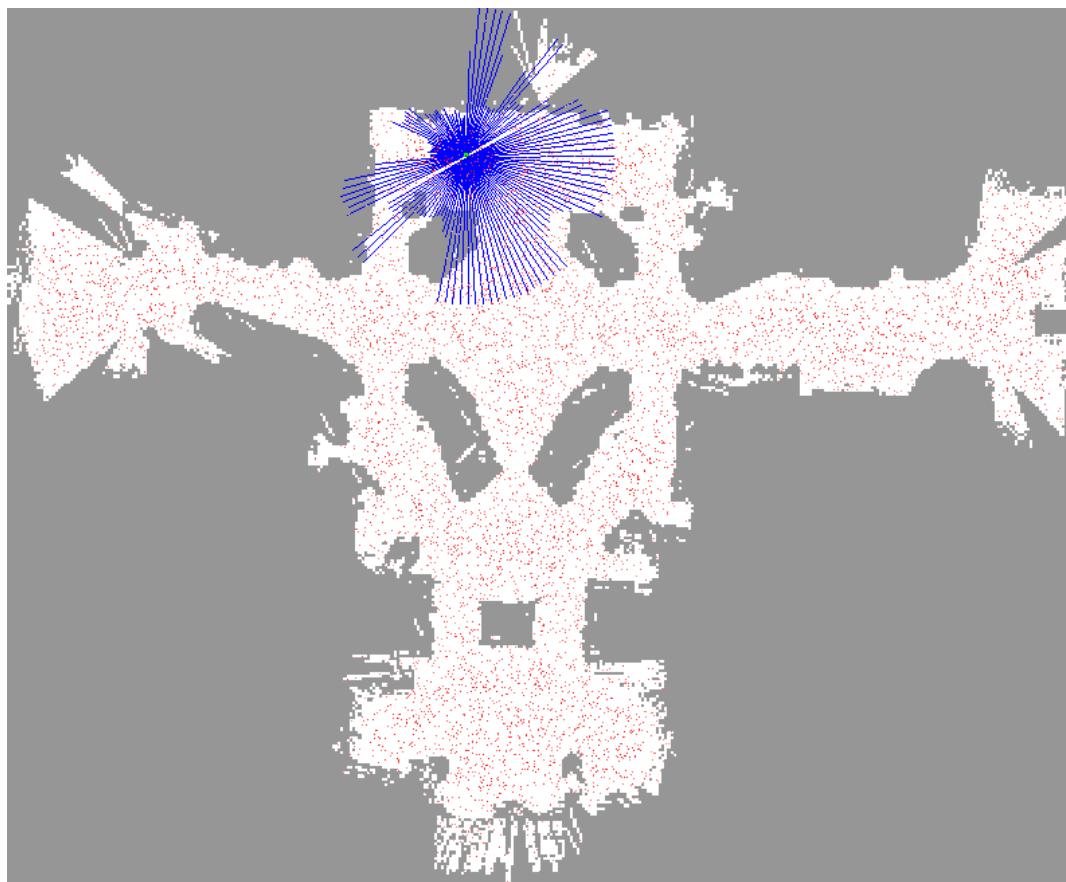
Exa



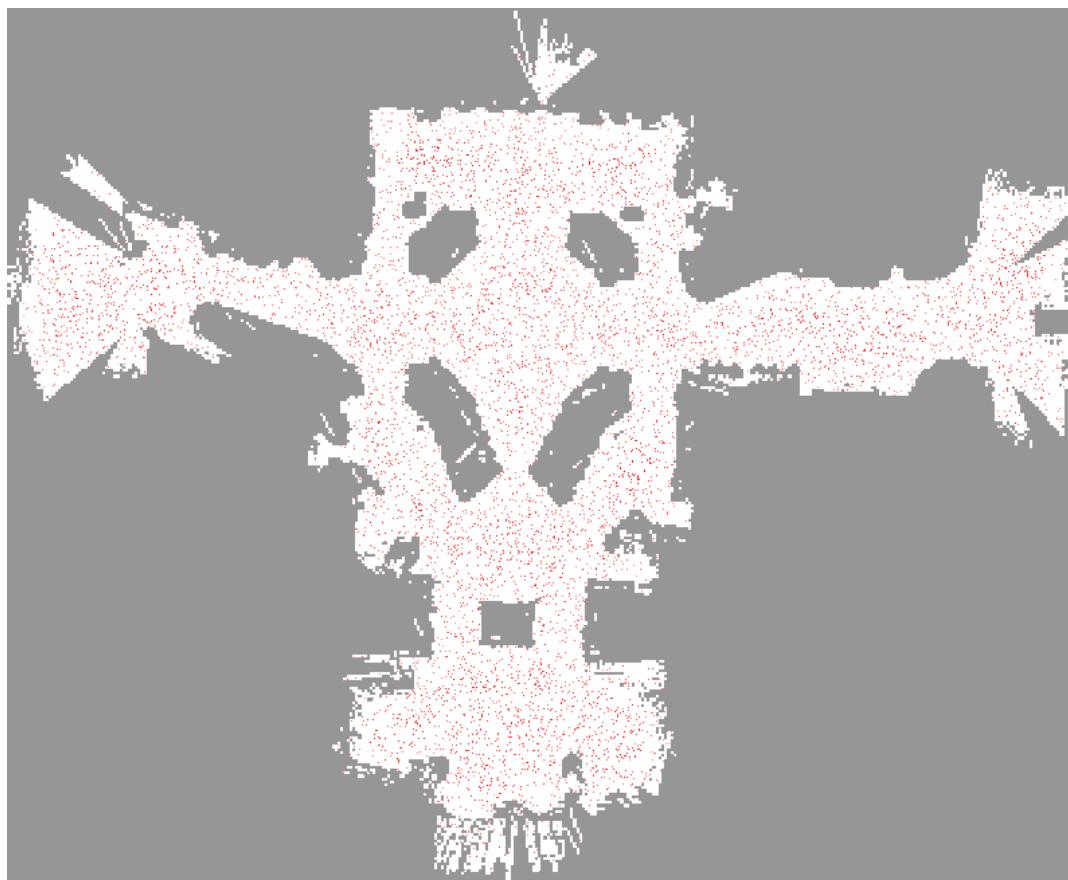
Example 2: Initialisation



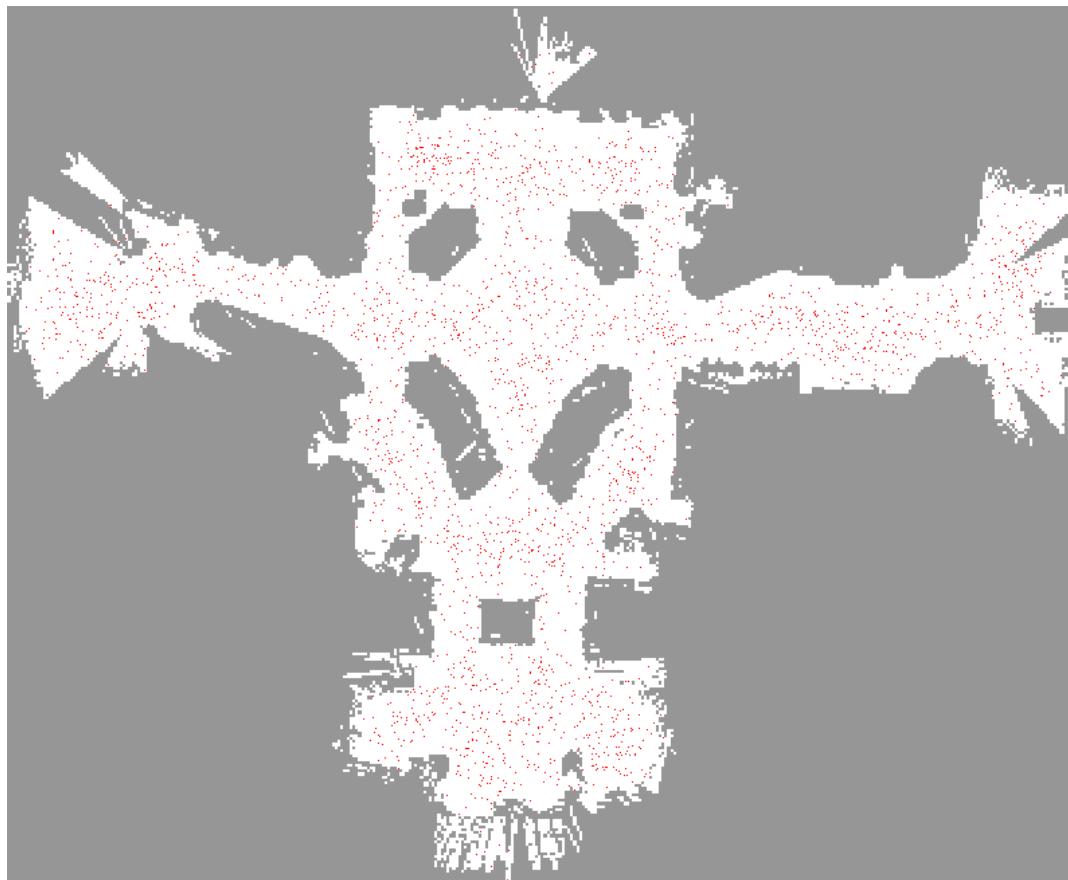
Measurement



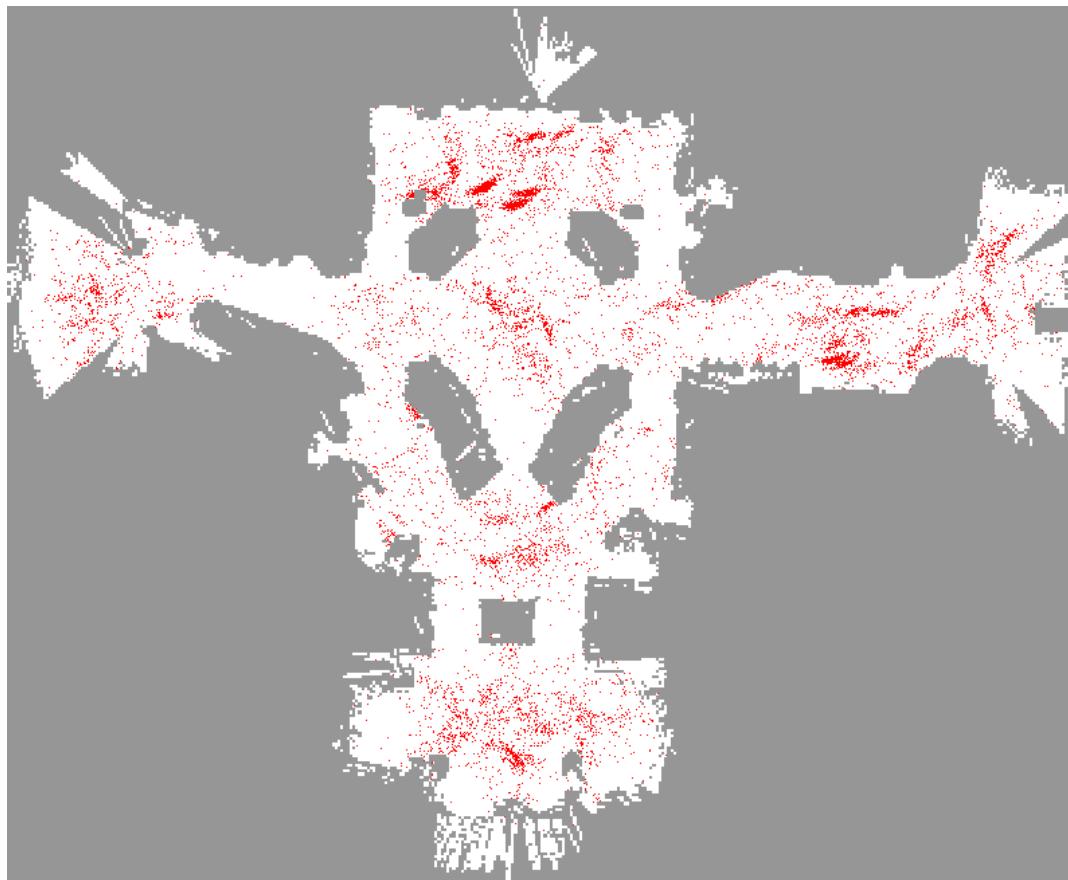
Weight update



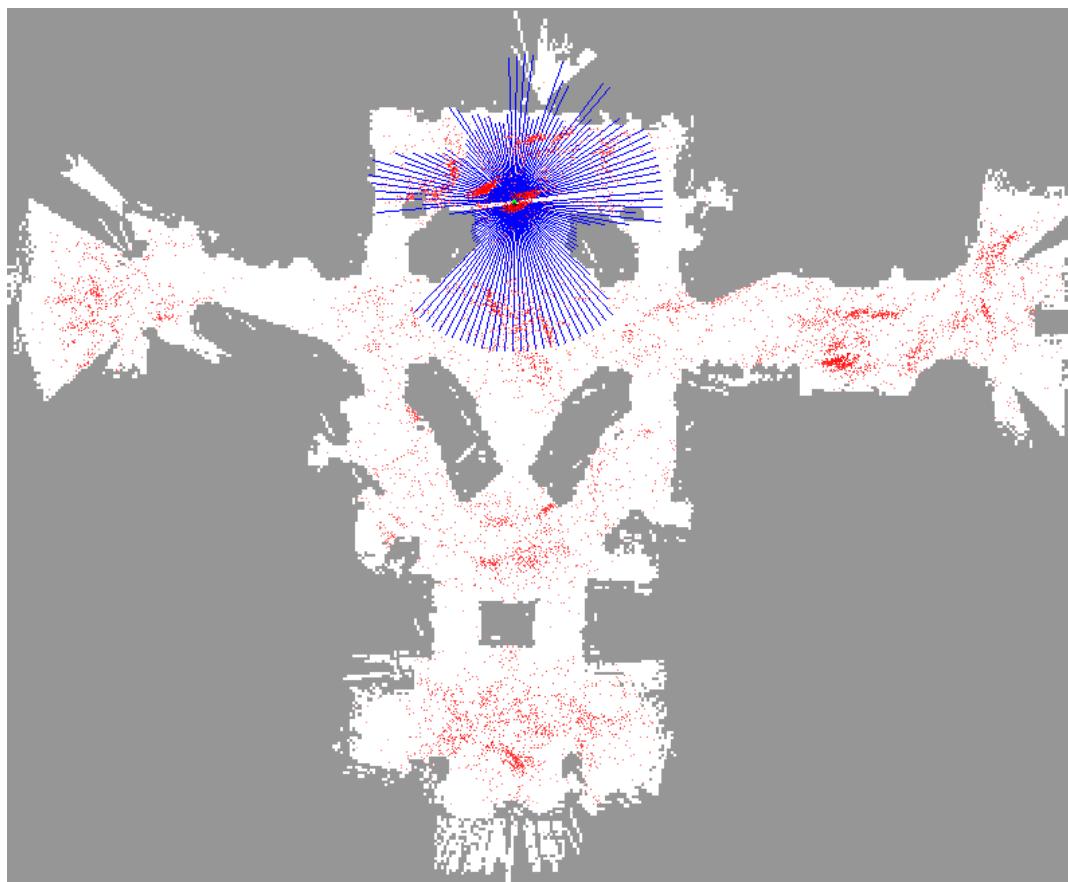
Resampling



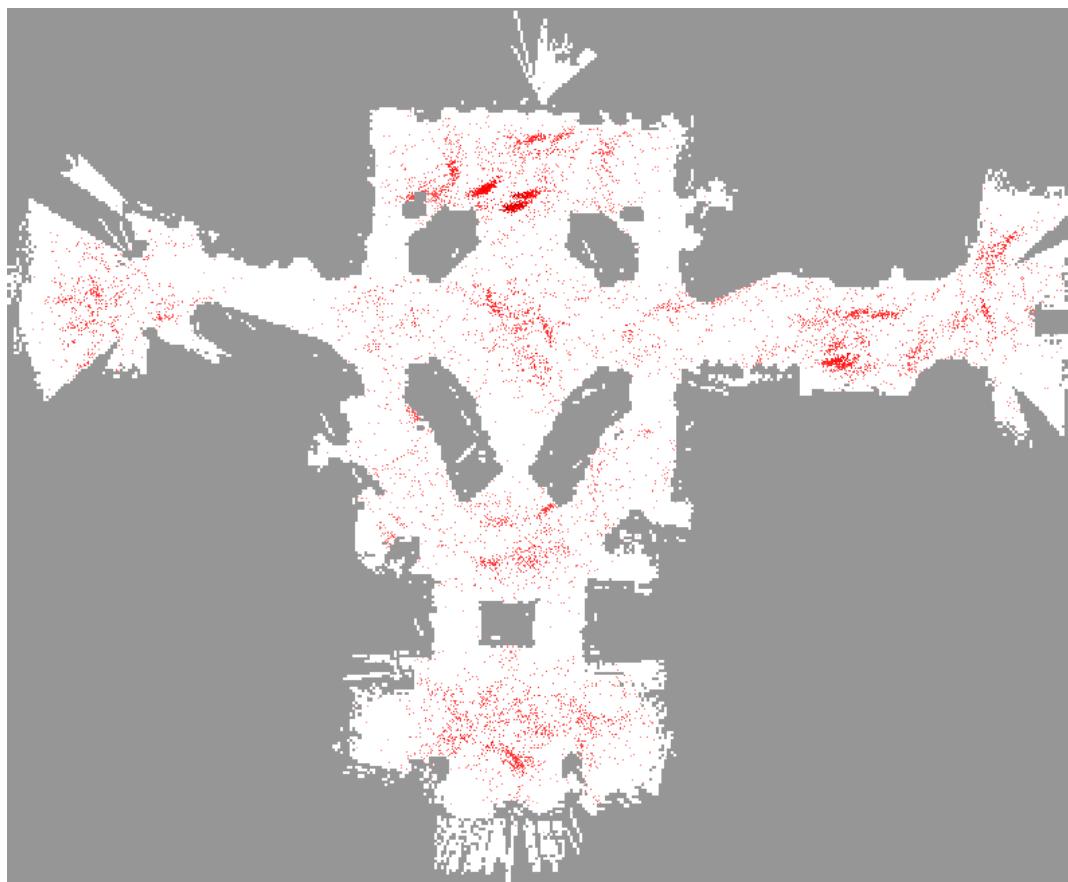
Motion update



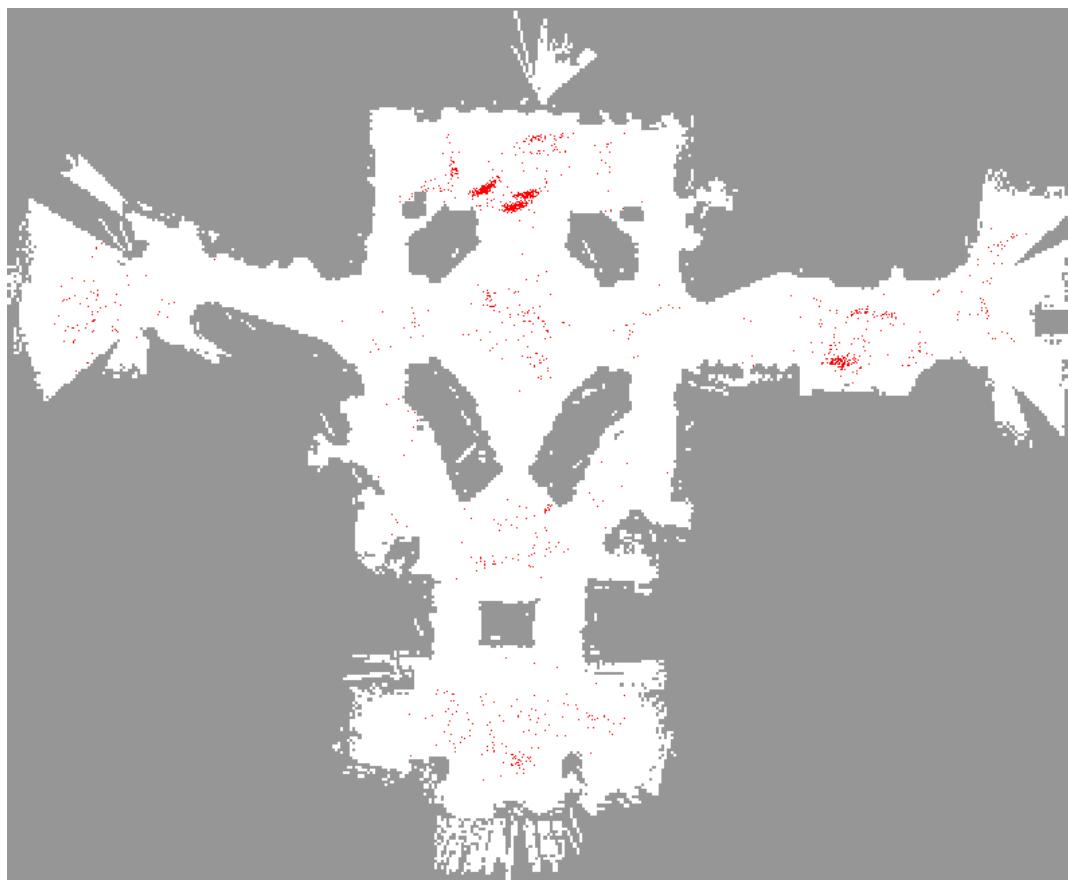
Measurement



Weight update



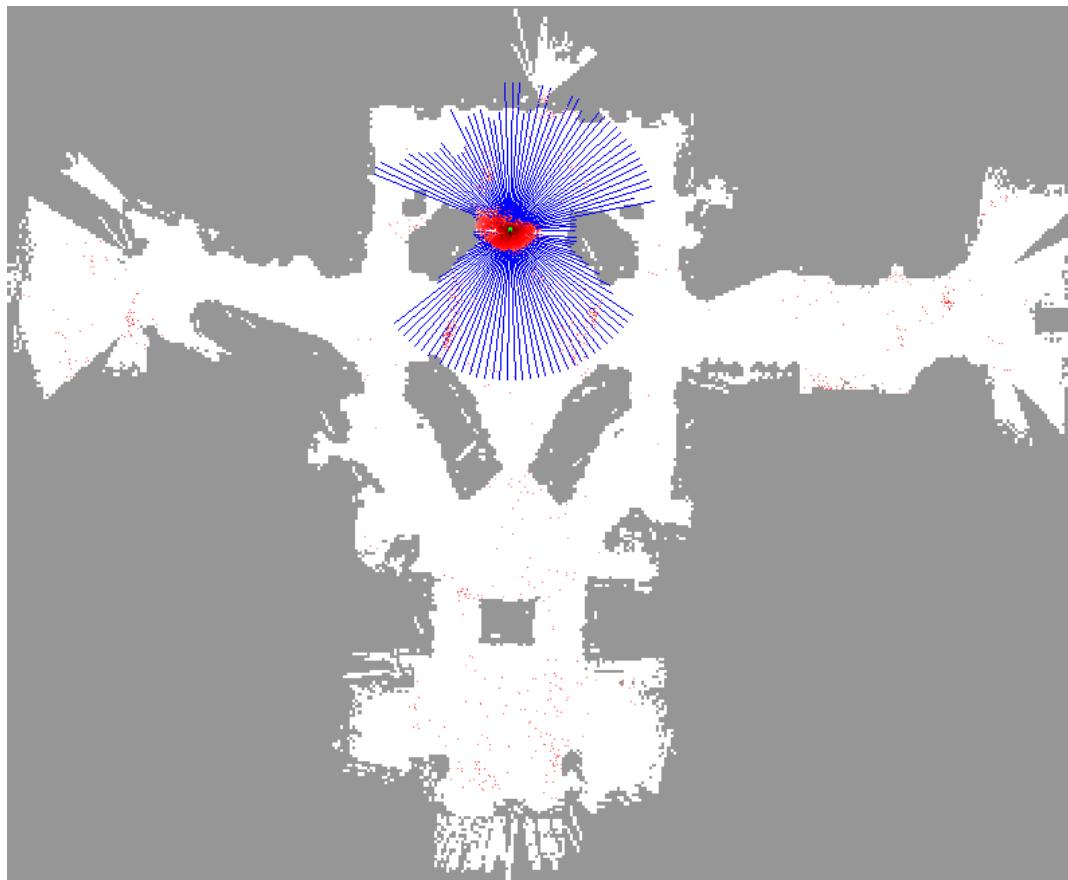
Resampling



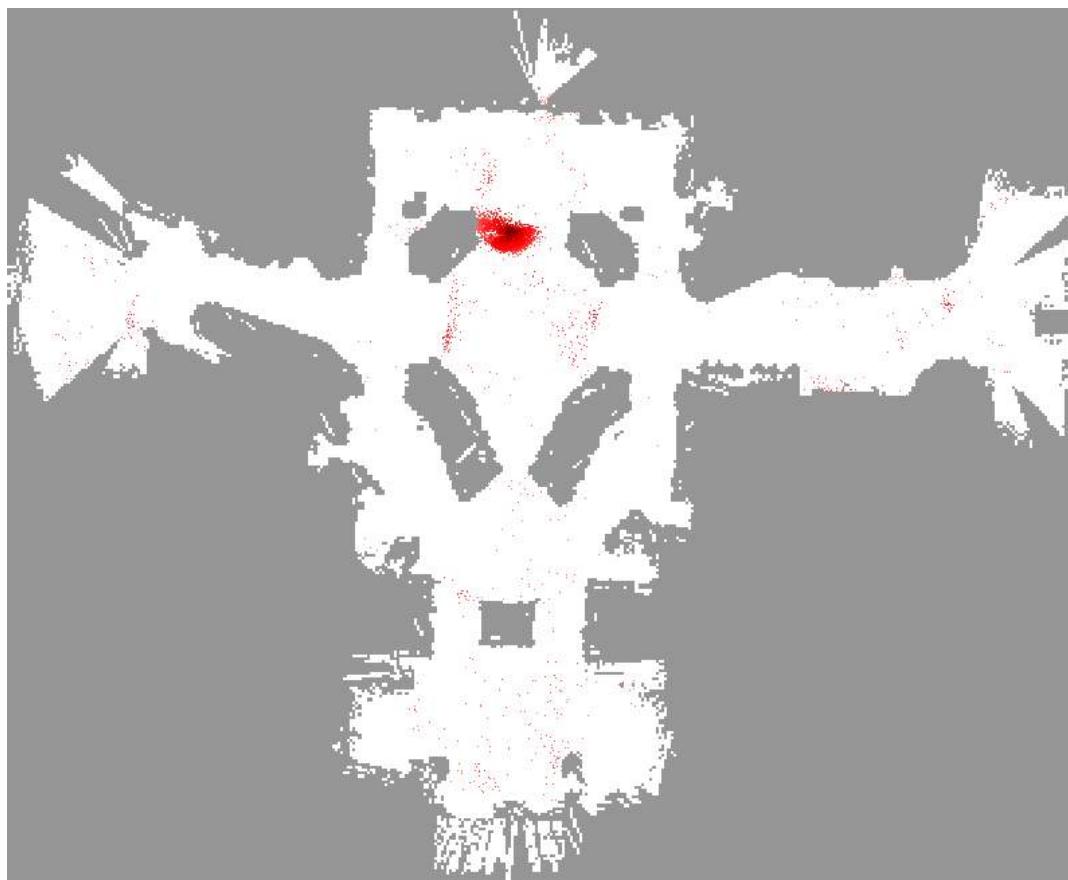
Motion update



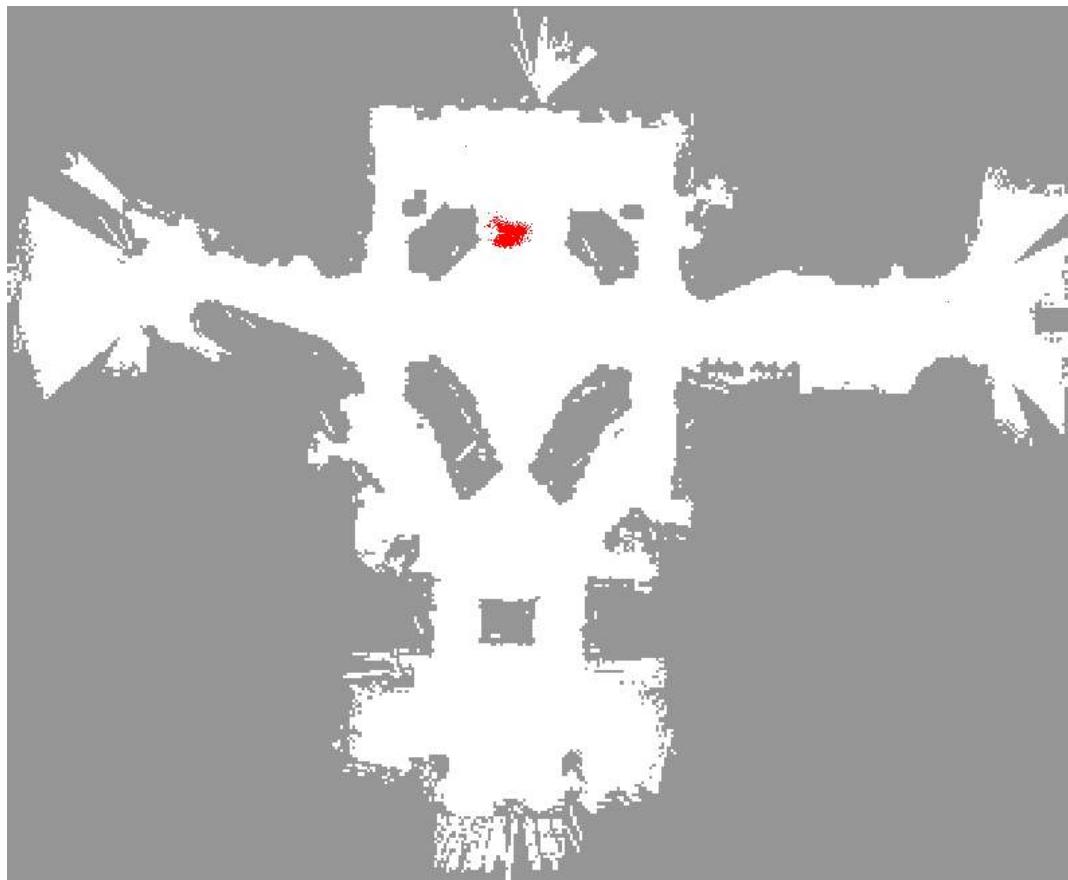
Measurement



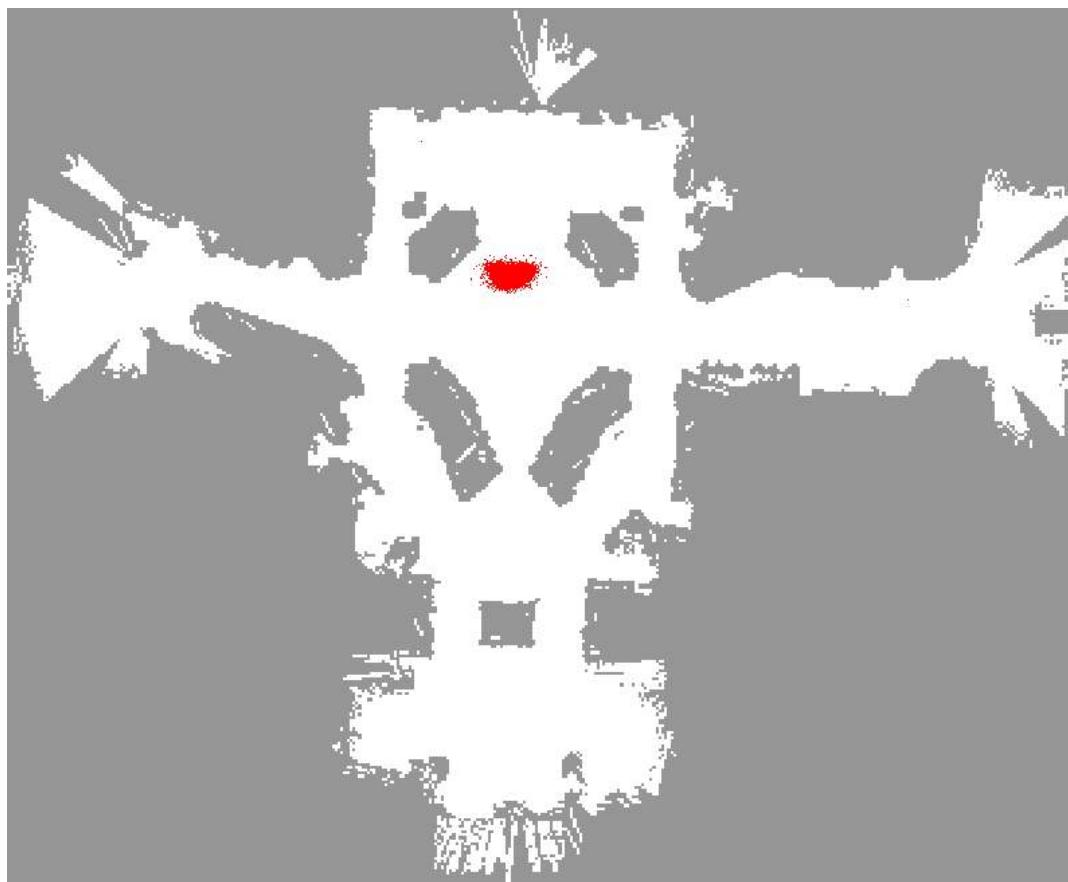
Weight update



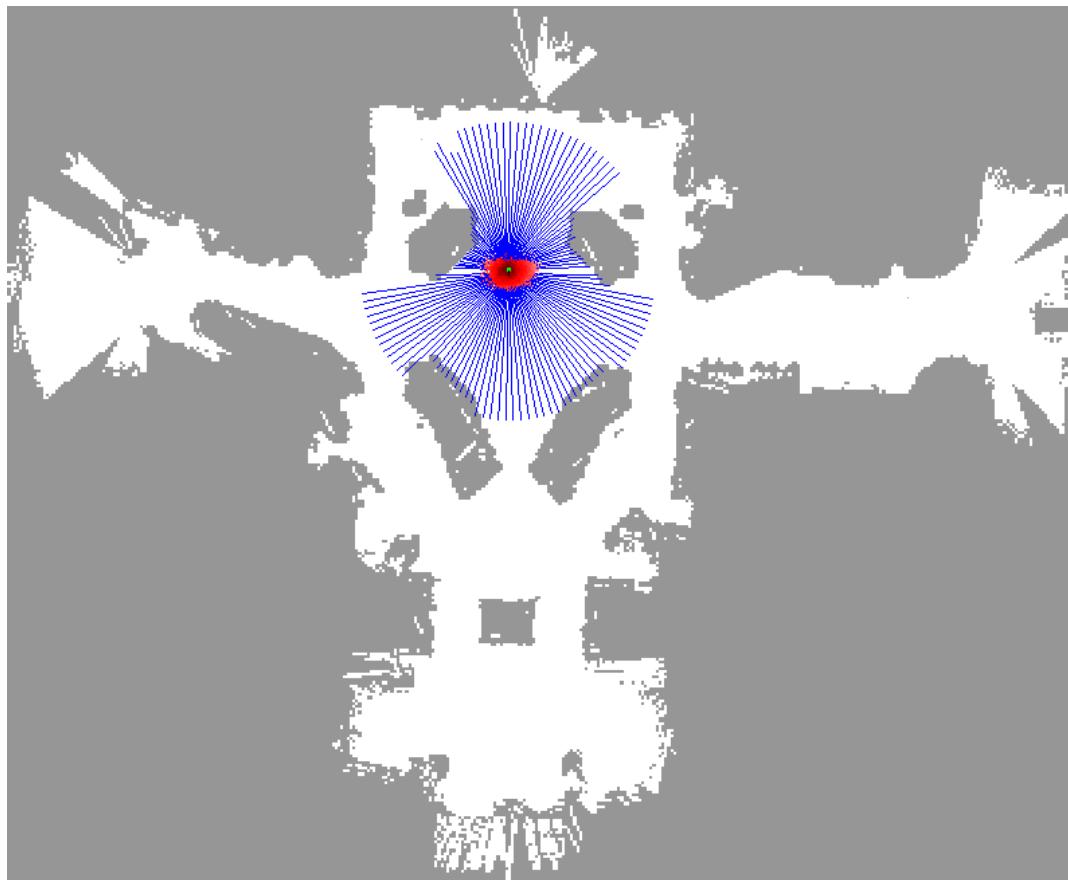
Resampling



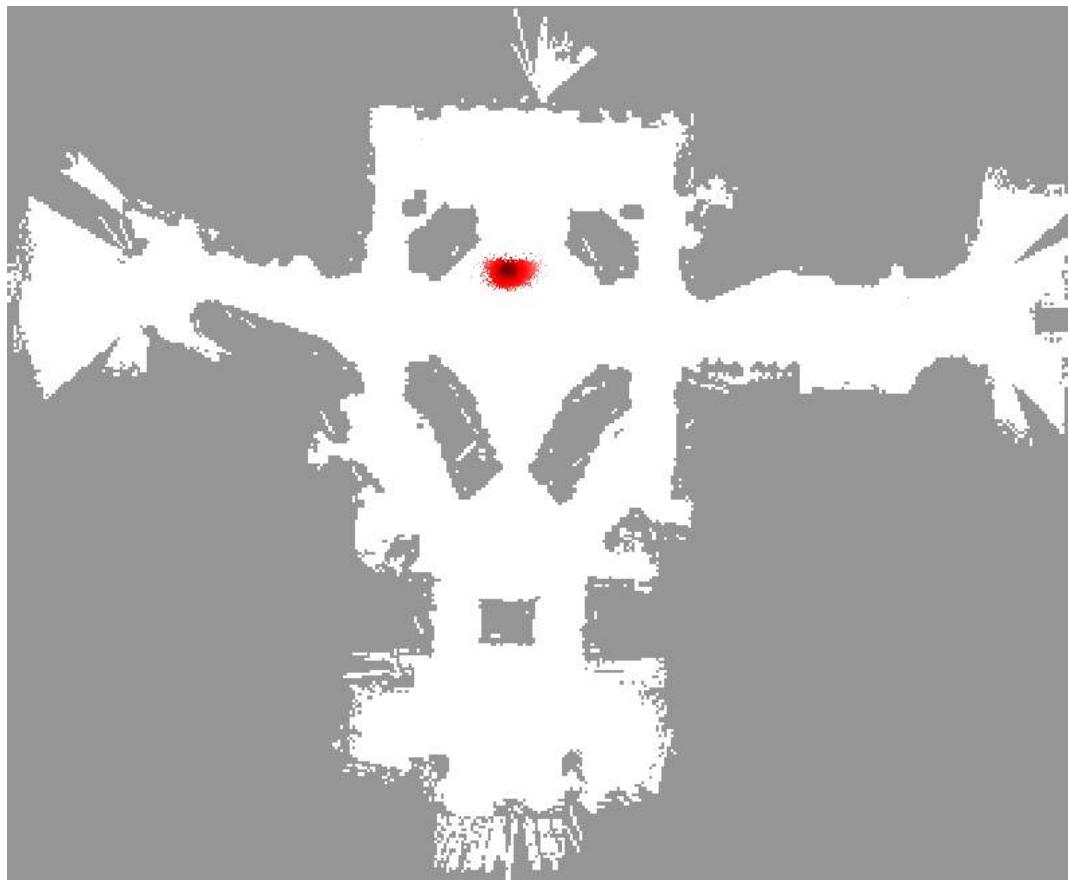
Motion update



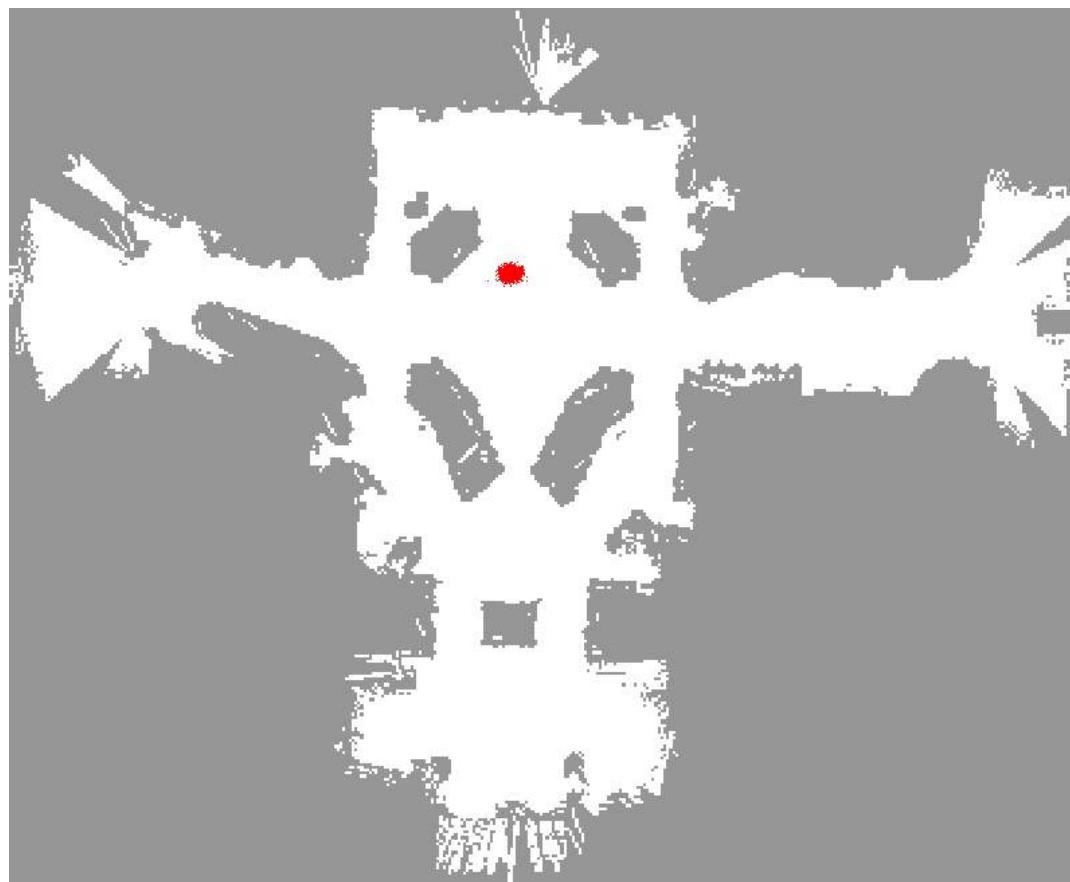
Measurement



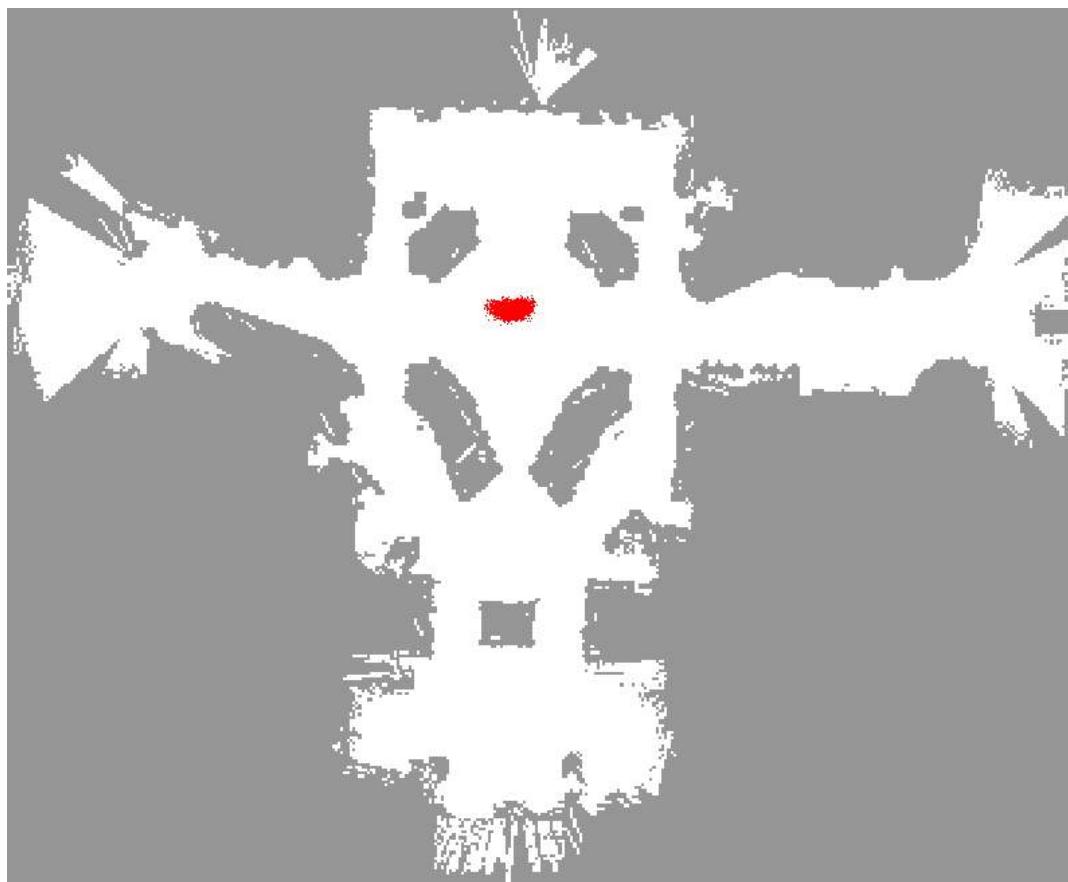
Weight update



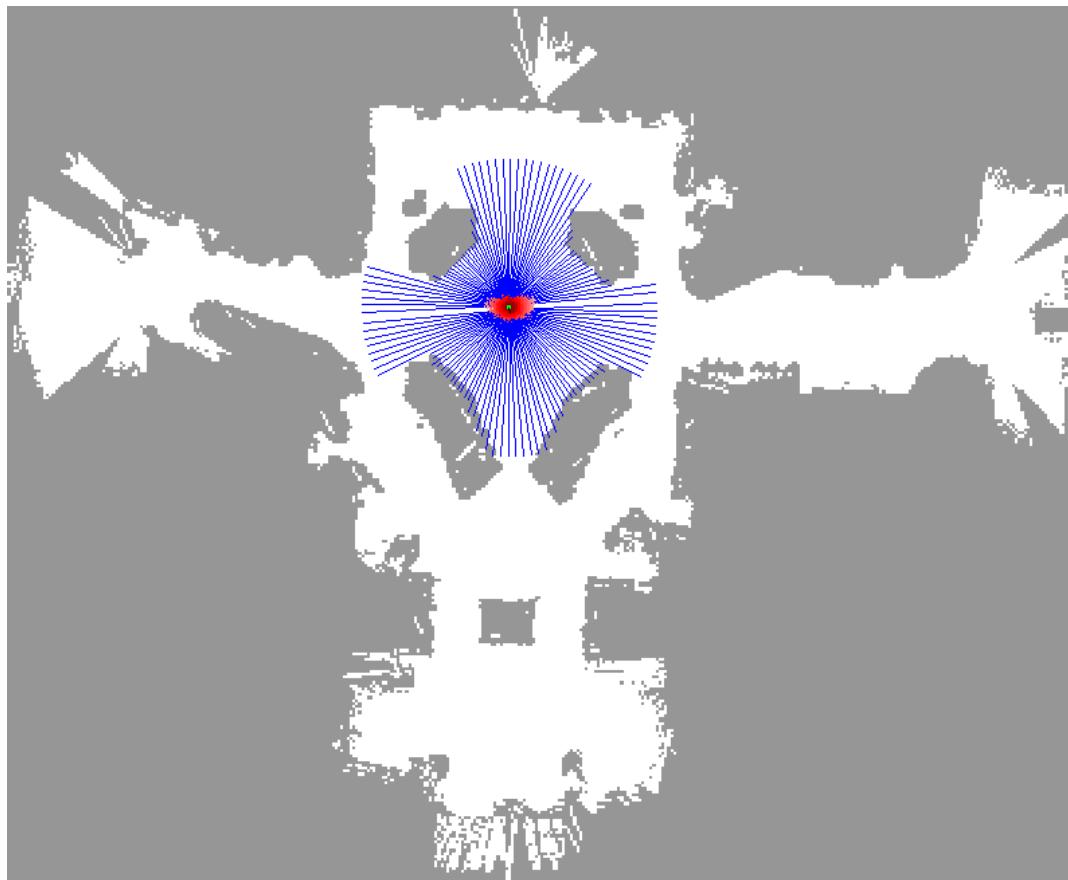
Resampling



Motion update



Measurement



Summary of Monte Carlo localisation

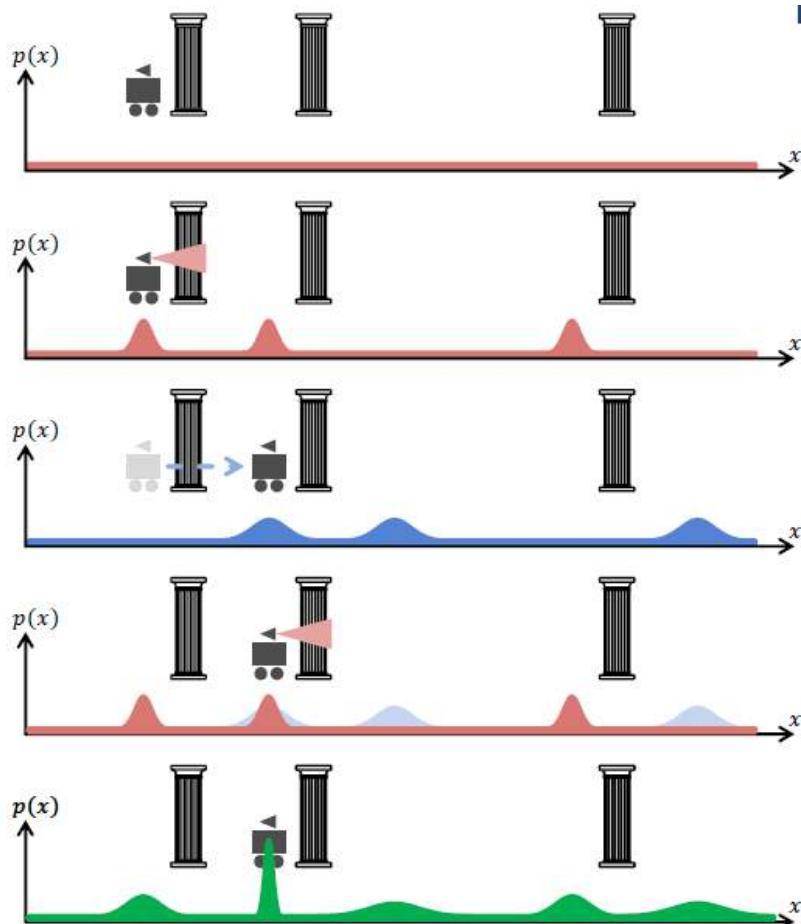
- Particle Filters (PFs) can represent arbitrary probability density functions (distributions) using samples
- PFs use sample importance resampling, with 4 main steps:
 - Initialisation
 - Predict
 - Update
 - Resample
- PFs can solve the “kidnapped robot problem” and handle perceptually aliased environments
- Also quite easy to implement...



Kalman Filter (KF)Localisation

- Instead of an arbitrary density function, KF uses Gaussians for robot belief, motion and measurement models.
- Only mean and covariance need to be updated – efficient computation
- Since initial belief is also Gaussian, initial position shall be known with a certain approximation
- KF localization addresses position tracking, not global localization or kidnapped robot problem.

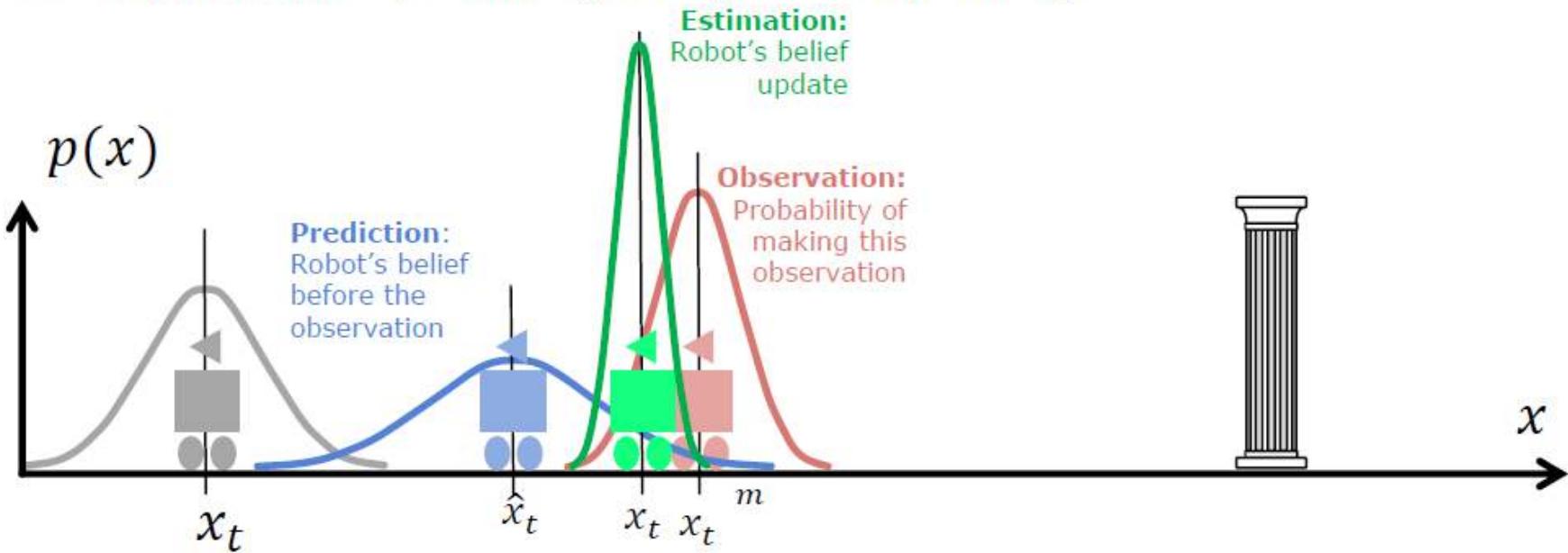
Revisit Markov Localisation:



- PERCEPTION (MEASUREMENT / CORRECTION) UPDATE: The robot queries its sensors
 - finds itself next to a pillar
- PREDICTION (ACTION) UPDATE: Robot moves one meter forward
 - Motion estimated by wheel encoder
 - Accumulation of uncertainty
- PERCEPTION UPDATE: The robot queries its sensors again
 - finds itself next to a pillar
- BELIEF UPDATE (information fusion)

KF Localisation

1. **Prediction (ACT)** based on previous estimate and odometry
 2. **Observation (SEE)** with on-board sensors
 3. **Measurement prediction** based on prediction and map
 4. **Matching** of observation and map
 5. **Estimation** → position update (posteriori position)



Probabilistic Map-Based Localisation

Markov localization

uses an explicitly specified probability distribution across all possible robot positions

Pros:

- allows for localization starting from any unknown position
- can recover from ambiguous situations

Cons:

- requires a discrete representation of the space, such as a geometric grid or a topological graph
- requires memory and computational power

Kalman filter localization

uses a Gaussian probability density representation of robot position and scan matching for localization

Pros:

- tracks the robot from an initially known position
- precise and efficient
- can be used in continuous world representations

Cons:

- if uncertainty of robot becomes large (e.g. collision with an object) it can fail to capture the multitude of possible robot positions and can become irrevocably lost

ROS Navigation Stack

<http://wiki.ros.org/navigation> - taking information from odometry, sensors, and a goal pose, safely issues velocity commands

http://wiki.ros.org/map_server - map_server provides the map_server ROS Node, which offers map data as a ROS Service. It also provides the map_saver command-line utility, which allows dynamically generated maps to be saved to file.

<http://wiki.ros.org/gmapping> - provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

<http://wiki.ros.org/amcl> - implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.

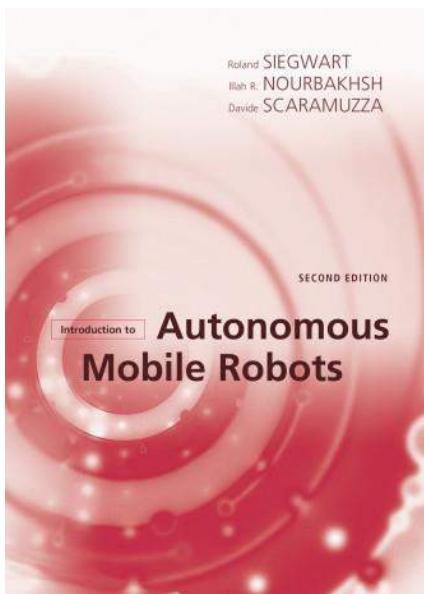
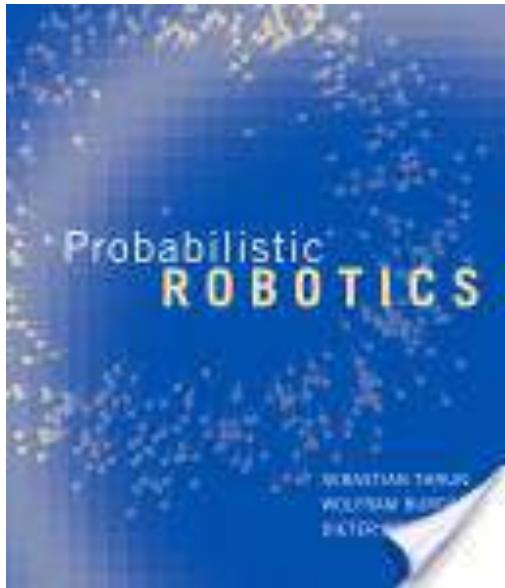
http://wiki.ros.org/global_planner - provides an implementation of a fast, interpolated global planner for navigation. Dijkstra/A*

http://wiki.ros.org/base_local_planner - provides implementations of the Trajectory Rollout and Dynamic Window approaches to local robot navigation on a plane.

http://wiki.ros.org/move_base - links together a global and local planner to accomplish its global navigation task.

Robot navigation

- Navigation goal determines the trajectory from a start point – global planner
- Local planner avoids obstacles not present in the global map – sensor driven information about obstacles
- Recovery behaviours if robot gets stuck



Suggested reading

- S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005. Ch. 4
- Siegwart, R. Nourbakhsh, I., and Scaramuzza, D. (2004). *Introduction to autonomous mobile robots*, (MIT Press). Ch.5.

CMP3103M/CMP9050M

Autonomous Mobile Robotics

Dr Ayse Kucukyilmaz

University of Lincoln
Centre for Autonomous Systems

INB3201

akucukyilmaz@lincoln.ac.uk
<http://webpages.lincoln.ac.uk/akucukyilmaz/>



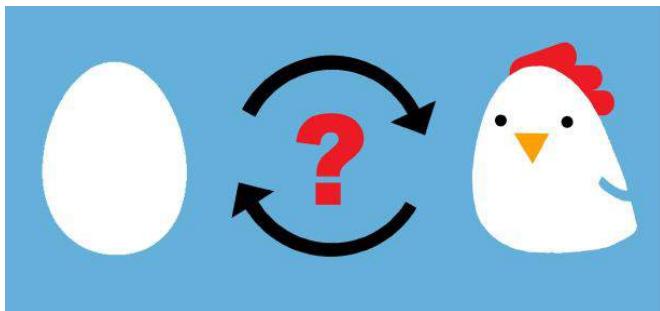
UNIVERSITY OF
LINCOLN



Autonomous map building

- Starting from an arbitrary initial point, a mobile robot should be able to autonomously explore the environment with its on-board sensors, gain knowledge about it, interpret the scene, build an appropriate map, and localize itself relative to this map.

SLAM



- Making maps is a chicken or egg
- problem: if we can't know where we are, we cannot make a good model, but if we don't have a good model we can't know where are.
- This is known as the Simultaneous Localisation And Mapping, a.k.a. **SLAM**

SLAM: Simultaneous Localization AND Mapping

The SLAM problem:

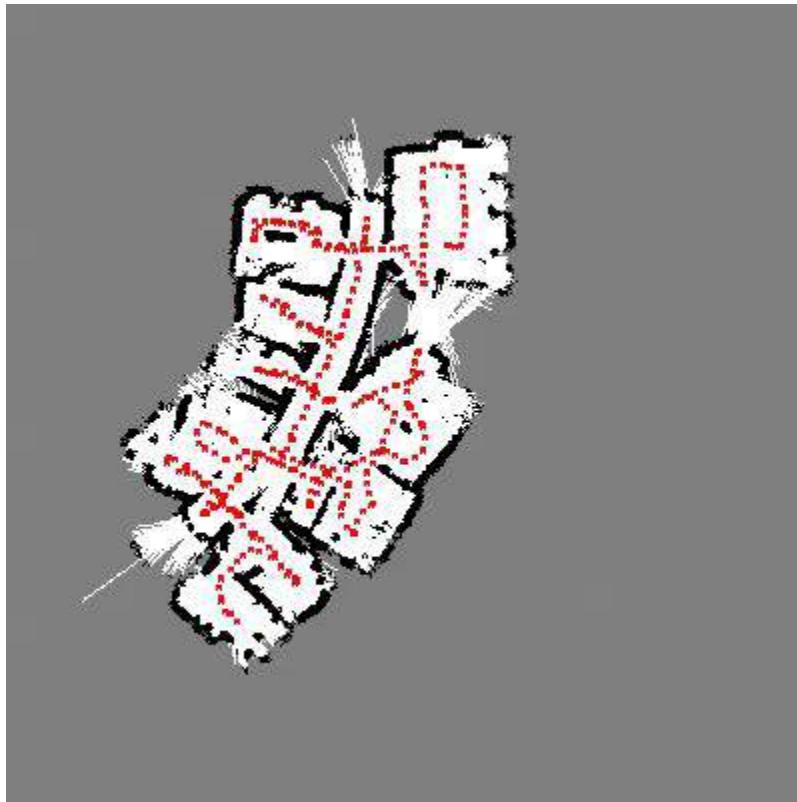
How can a body **navigate** in a previously unknown environment, while constantly building and updating a **map** of its workspace using onboard sensors & onboard computation?

WHEN IS IT NECESSARY?

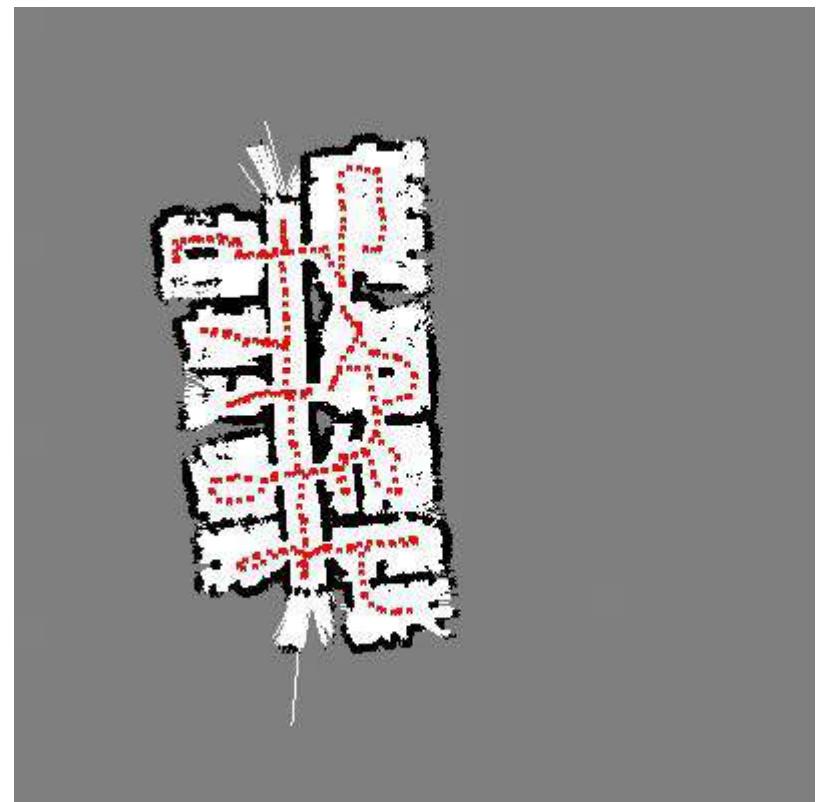
- When a robot must be truly **autonomous** (no human input)
- When there is **no prior knowledge** about the environment
- When we **cannot** rely exclusively on pre-placed **beacons** or **external positioning systems** (e.g. GPS)
- When the robot needs to know where it is

Why is self-localisation needed?

Example of mapping using odometry



Example of simultaneous localisation and mapping

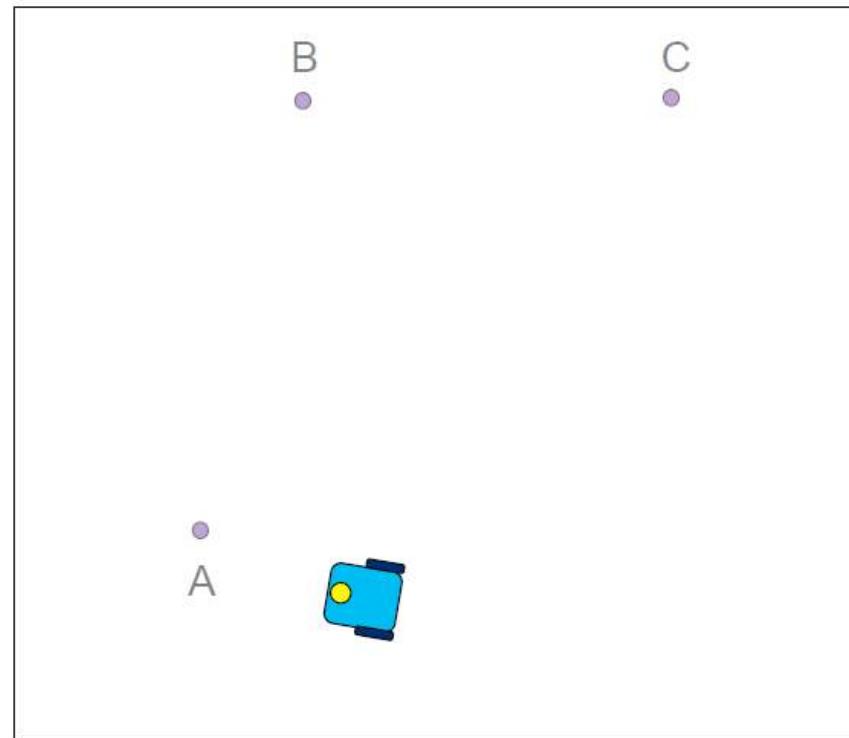


SLAM – with a Gaussian Filter

Use internal representations for

- The positions of landmarks (: map)
- The camera parameters

Assumption: Robot's uncertainty at starting point is zero



Start: robot has zero uncertainty

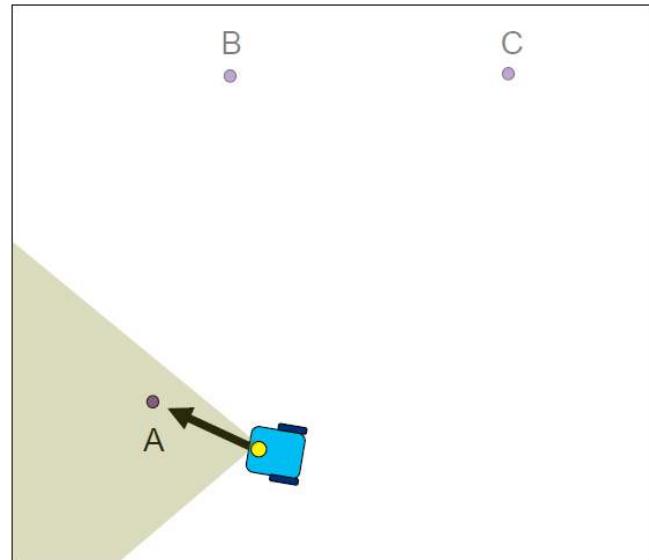
SLAM – with a Gaussian Filter

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



First measurement of feature A

SLAM – with a Gaussian Filter

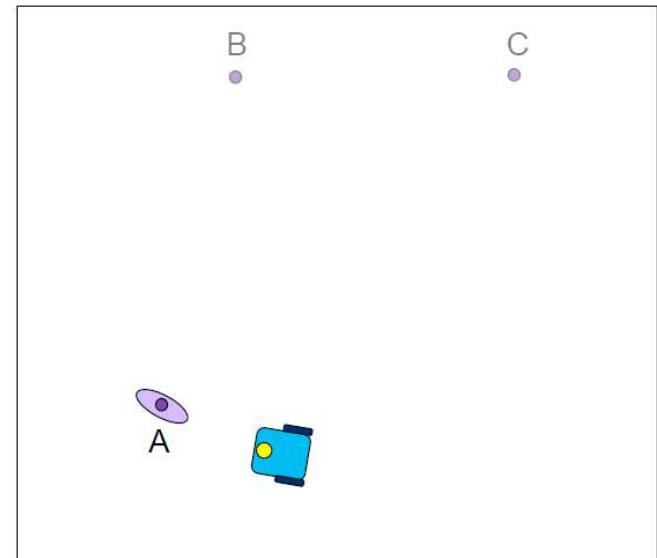
- The robot observes a feature which is mapped with an uncertainty related to the **measurement model**
e.g. the camera model, describing how world points map into pixels in the image

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



SLAM – with a Gaussian Filter

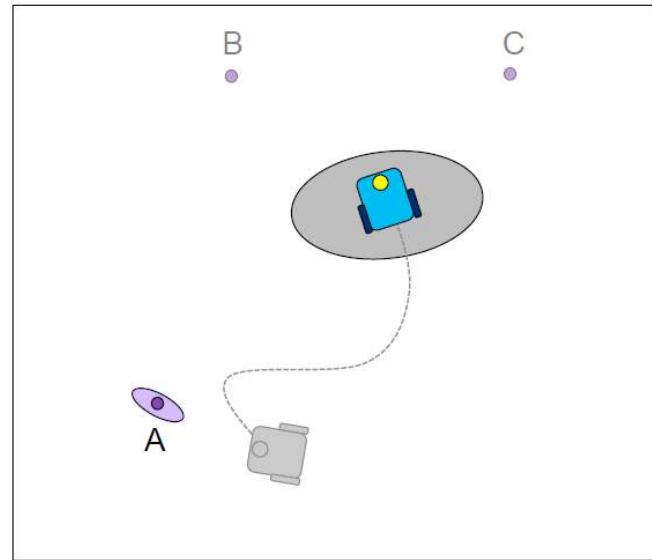
- As the robot moves, its pose uncertainty increases, obeying the robot's **motion model**.
e.g. the driver's commands: turn right, drive on for 1m – uncertainty is added due to wheel slippage and other imprecisions

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



Robot moves forwards: uncertainty grows

SLAM – with a Gaussian Filter

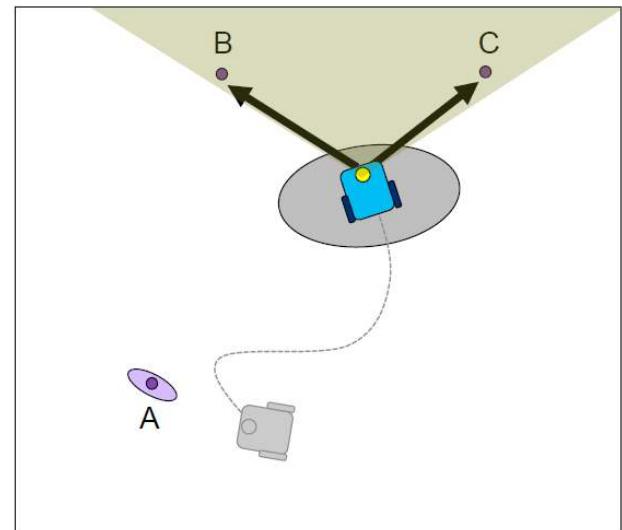
- Robot observes two new features

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



Robot makes first measurements of B & C

SLAM – with a Gaussian Filter

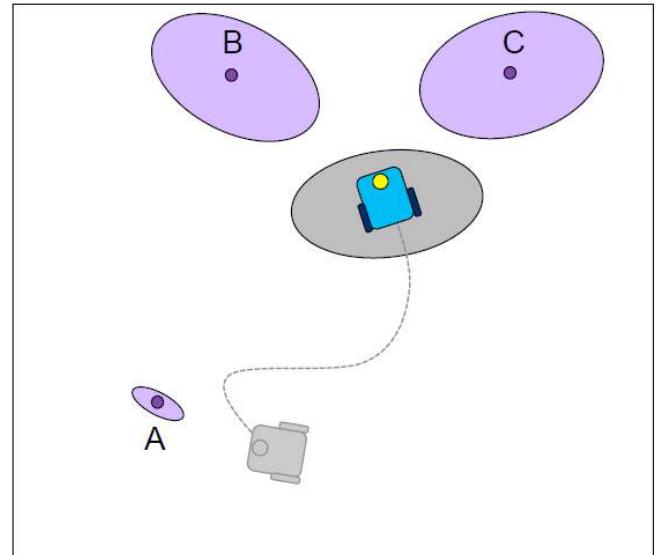
- Their position uncertainty results from the **combination** of the measurement error with the robot pose uncertainty.
- map becomes **correlated** with the robot pose estimate.

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



Robot makes first measurements of B & C

SLAM – with a Gaussian Filter

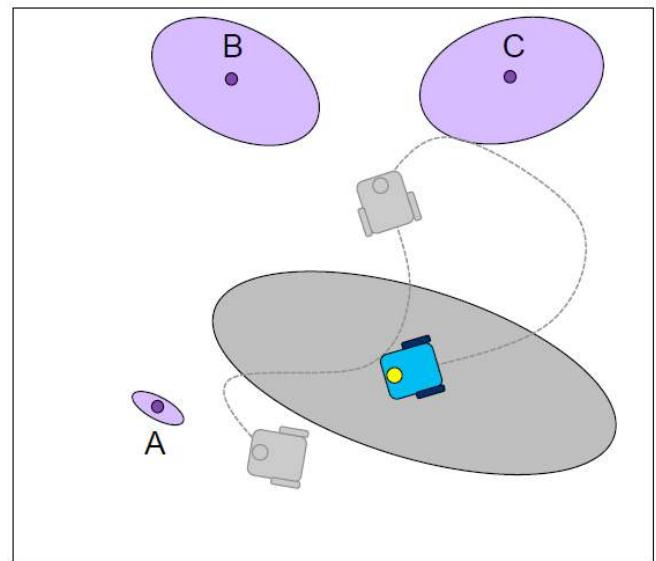
Robot moves again and its uncertainty increases (motion model)

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



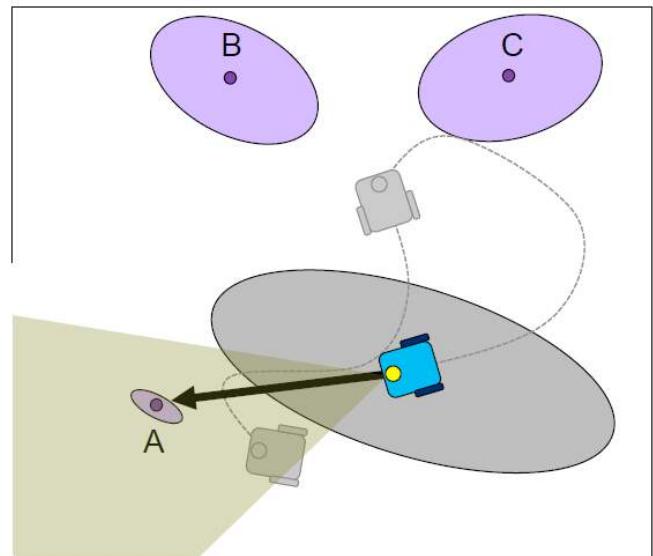
Robot moves again: uncertainty grows more

SLAM – with a Gaussian Filter

Robot re-observes an old feature
→ **Loop closure** detection

On every frame:

Predict how the robot has moved
Measure
Update the internal representations



SLAM – with a Gaussian Filter

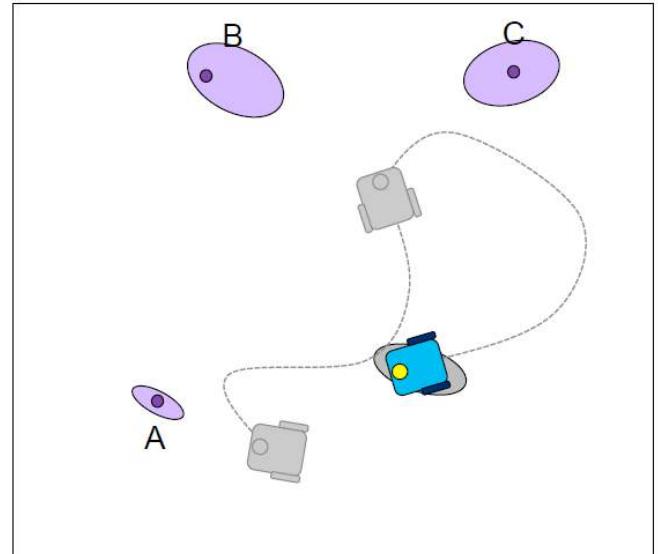
- Robot updates its position: the resulting **pose** estimate becomes **correlated** with the feature **location estimates**.
- Robot's uncertainty **shrinks** and so does the uncertainty in the rest of the map

On every frame:

Predict how the robot has moved

Measure

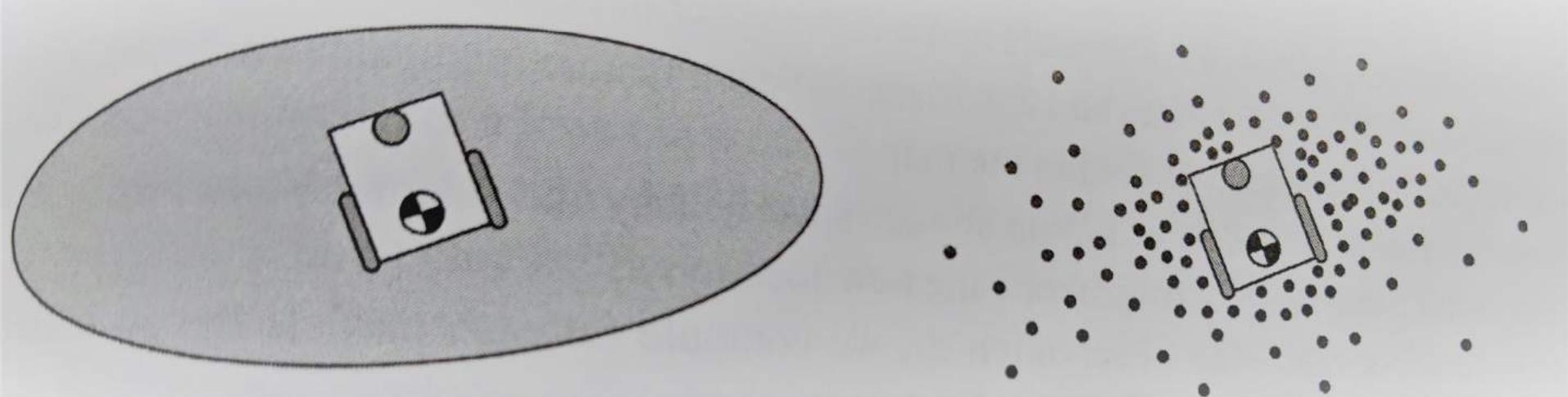
Update the internal representations



Robot re-measures A: “**loop closure**”
uncertainty shrinks

EKF SLAM vs Particle Filter SLAM

- Standard EKF SLAM represents the probability distribution in parametric form with a Gaussian
- Particle filter SLAM represents the probability distribution as a set of particles drawn randomly from the parametric distribution (the density of particles is higher toward the centre of the Gaussian)



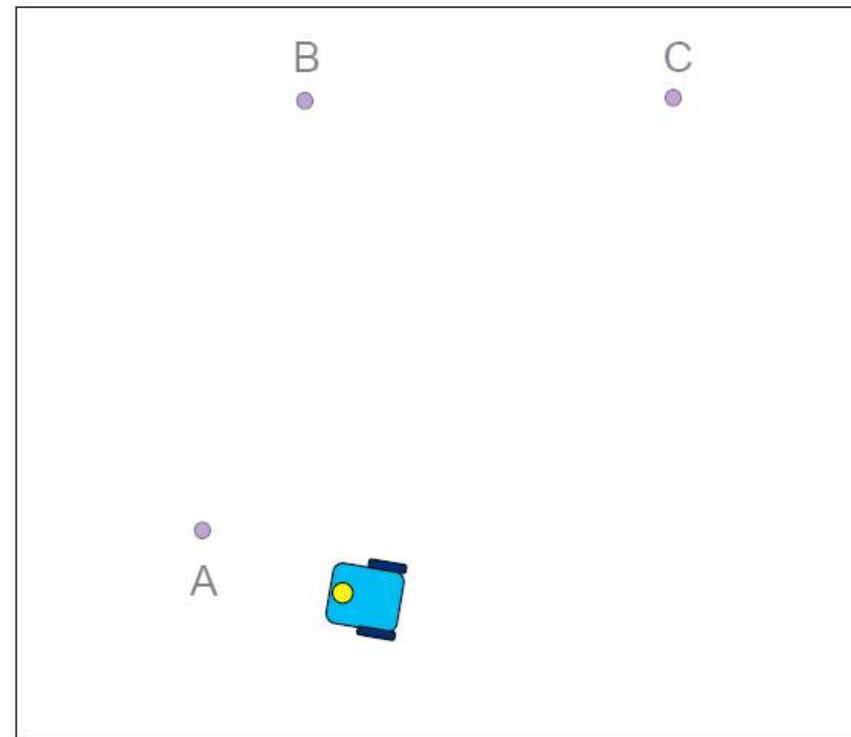
SLAM – with a Particle Filter

Use internal representations for

- The positions of landmarks (: map)
- The camera parameters

Assumption: Robot's uncertainty at starting point is zero

Initialize N particles at the origin, with weight $1/N$



Start: robot has zero uncertainty

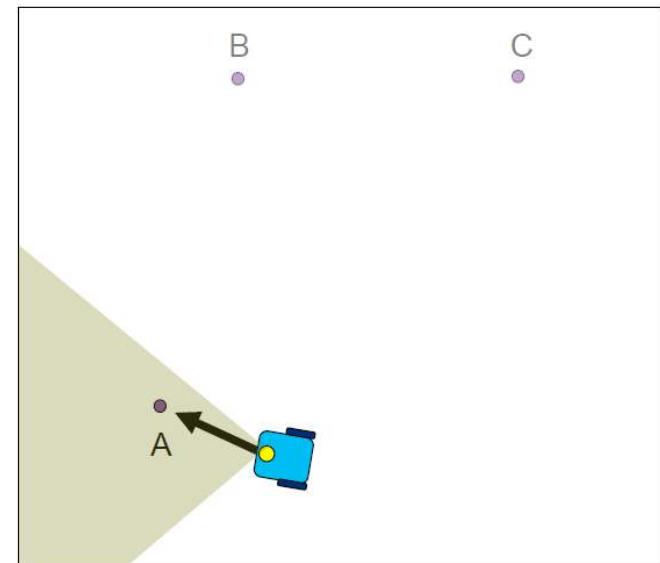
SLAM – with a Particle Filter

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



First measurement of feature A

SLAM – with a Particle Filter

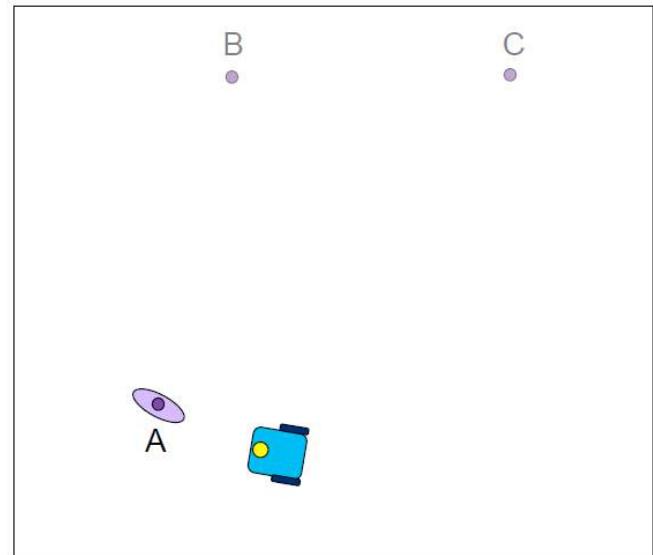
Robot observes a feature which is mapped with an uncertainty related to the **measurement model**.

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



SLAM – with a Particle Filter

As the robot moves pose uncertainty increases

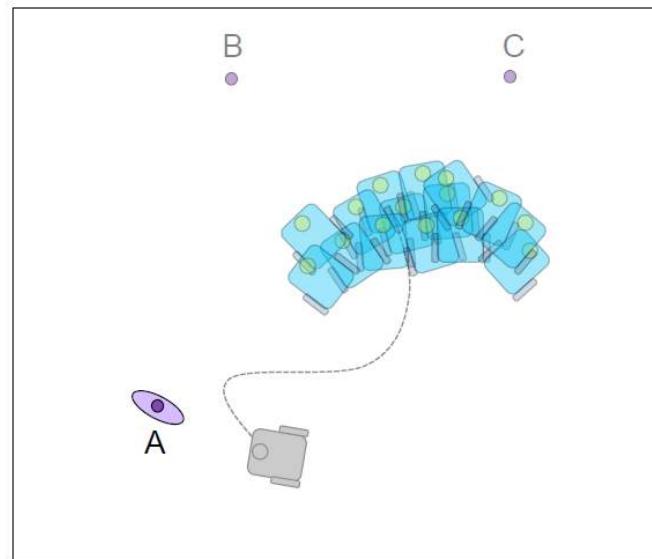
Apply motion model to each particle

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



Robot moves forwards: uncertainty grows

SLAM – with a Particle Filter

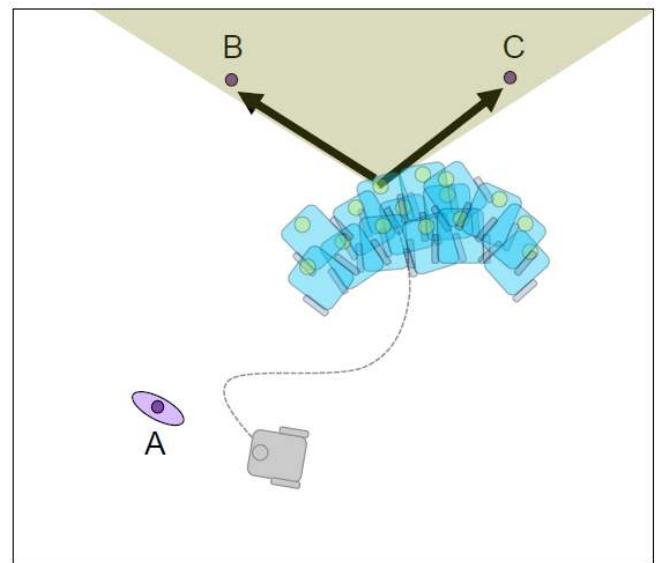
Robot observes two new features

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



SLAM – with a Particle Filter

Position uncertainty is encoded for each particle individually:

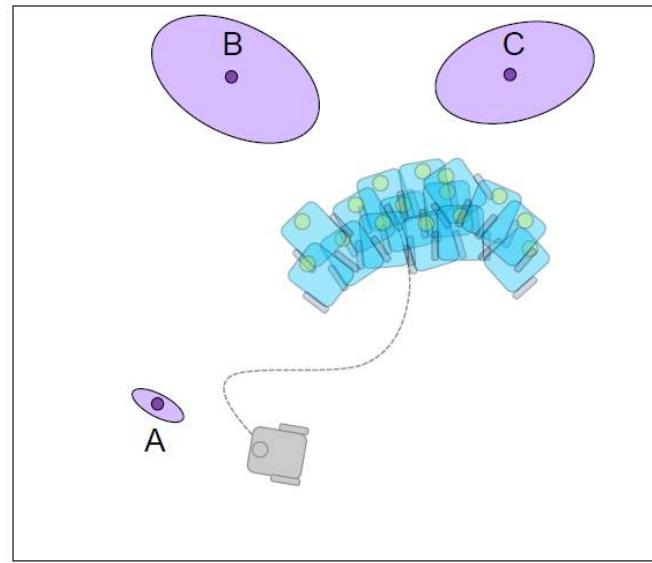
- Compare the particle's predicted measurements with actual measurements
- Re-weigh s.t. particles with good predictions get higher weight
- Renormalize particle weights
- Resample

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



SLAM – with a Particle Filter

Robot moves again and its uncertainty increases

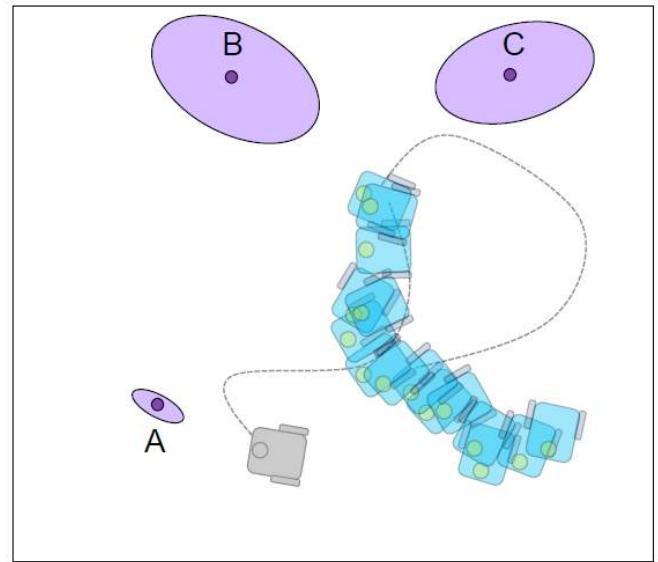
Apply motion model to each particle

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



Robot moves again: uncertainty grows more

SLAM – with a Particle Filter

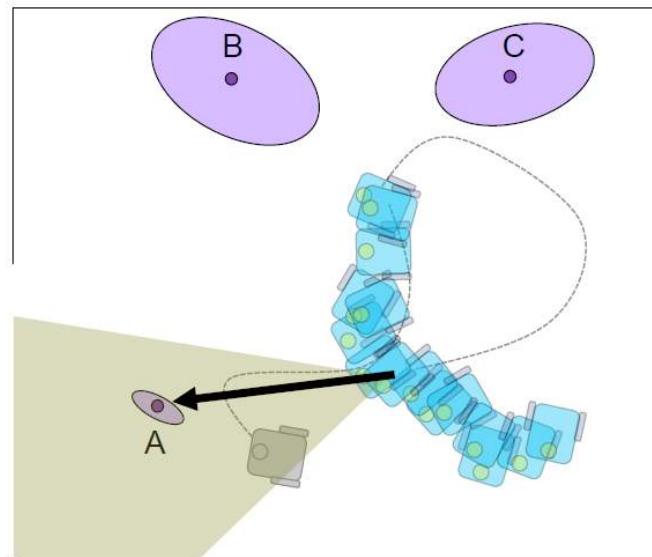
Robot moves again and its uncertainty increases

On every frame:

Predict how the robot has moved

Measure

Update the internal representations

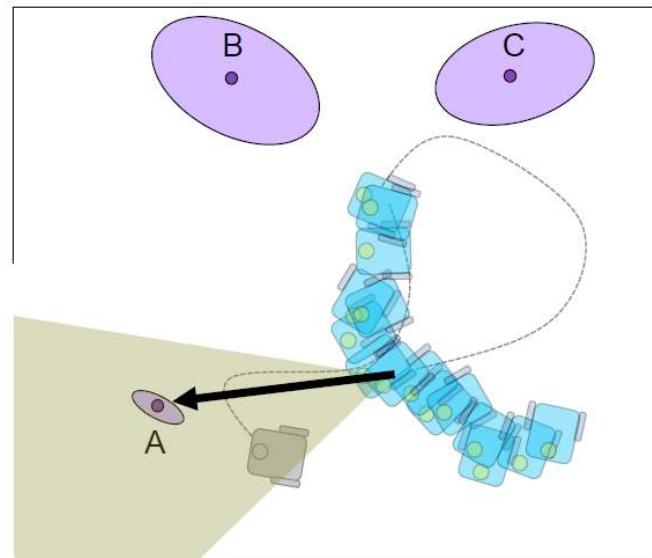


SLAM – with a Particle Filter

Robot re-observes an old feature
Loop closure detection!

On every frame:

Predict how the robot has moved
Measure
Update the internal representations



SLAM – with a Particle Filter

For each particle

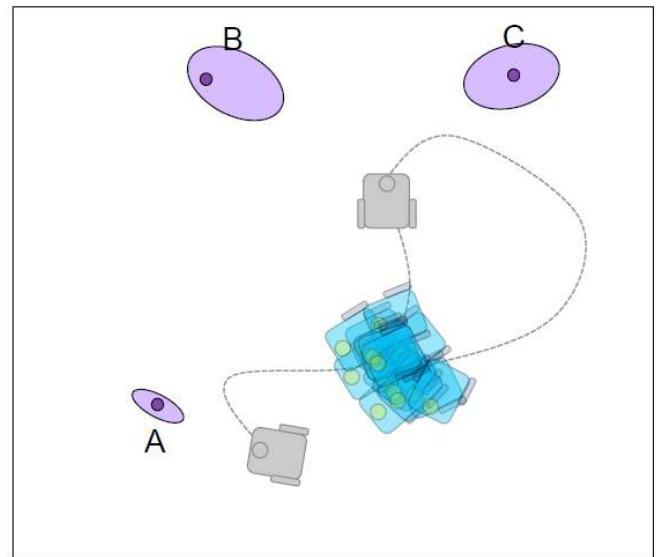
- Compare the particle's predicted measurements with actual measurements
- Re-weigh s.t. particles with good predictions get higher weight
- Renormalize particle weights
- Resample

On every frame:

Predict how the robot has moved

Measure

Update the internal representations



Robot re-measures A: “loop closure”
uncertainty shrinks

SLAM – with a Particle Filter

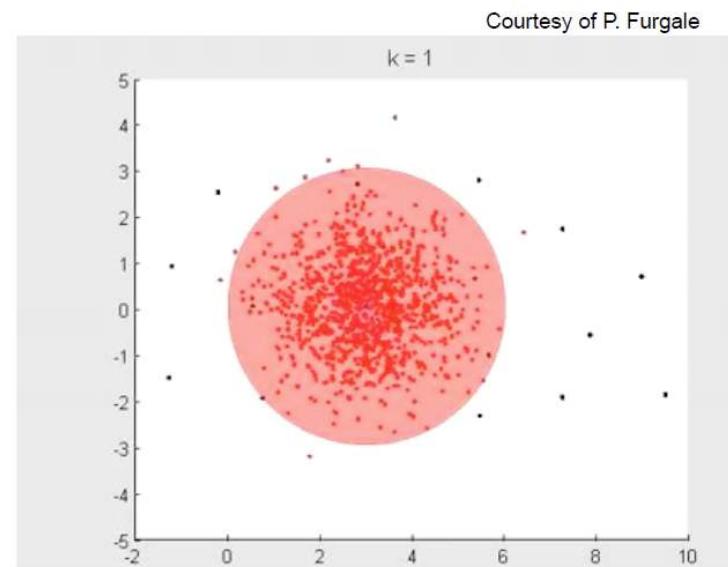
- Represents belief by a series of samples
- Each particle denotes a hypothesis of the state with an associated weight
- Predict/measure/update

Pros

- Noise densities from any distribution
- Works for multimodal distributions
- Easy to implement

Cons

- Does not scale to high dimensional problems
- Requires many particles to have good convergence

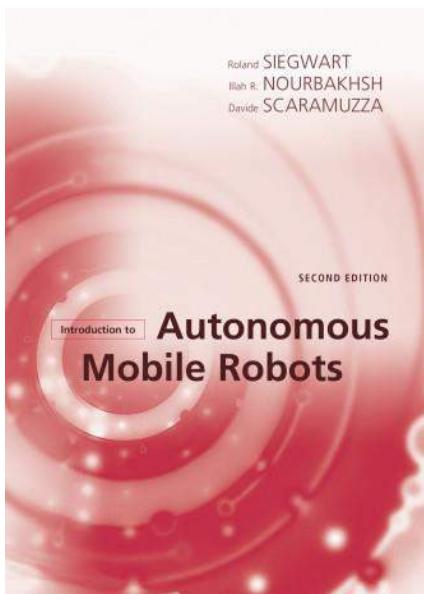
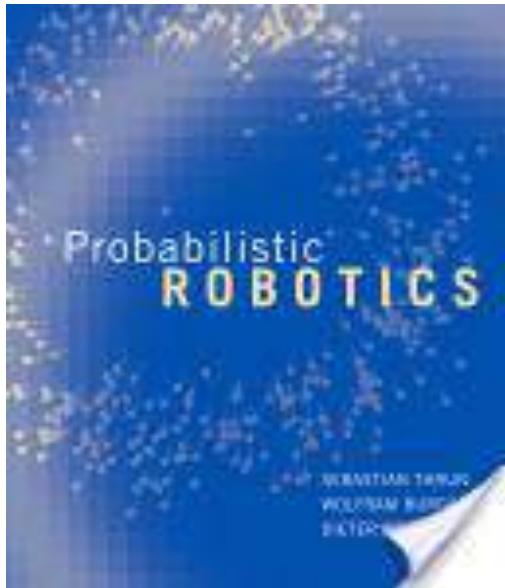


Distribution in the robot's position estimate:

- red dots – particle filtering
- red ellipse – EKF filtering

Open source SLAM software

- <http://www.openslam.org> – Comprehensive list of SLAM software
- <http://www.doc.ic.ac.uk/~ajd/software.html> - Andrew Dawson's real-time monocular SLAM
- <http://wiki.ros.org/gmapping> - ROS wrapper for OpenSlam's Gmapping



Suggested reading

- S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005. Ch. 4
- Siegwart, R. Nourbakhsh, I., and Scaramuzza, D. (2004). *Introduction to autonomous mobile robots*, (MIT Press). Ch.5.

CMP3103M/CMP9050M

Autonomous Mobile Robotics

Dr Ayse Kucukyilmaz

University of Lincoln
Centre for Autonomous Systems

INB3201

akucukyilmaz@lincoln.ac.uk
<http://webpages.lincoln.ac.uk/akucukyilmaz/>



UNIVERSITY OF
LINCOLN



ROS Navigation Stack

<http://wiki.ros.org/navigation> - taking information from odometry, sensors, and a goal pose, safely issues velocity commands

http://wiki.ros.org/map_server - map_server provides the map_server ROS Node, which offers map data as a ROS Service. It also provides the map_saver command-line utility, which allows dynamically generated maps to be saved to file.

<http://wiki.ros.org/gmapping> - provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

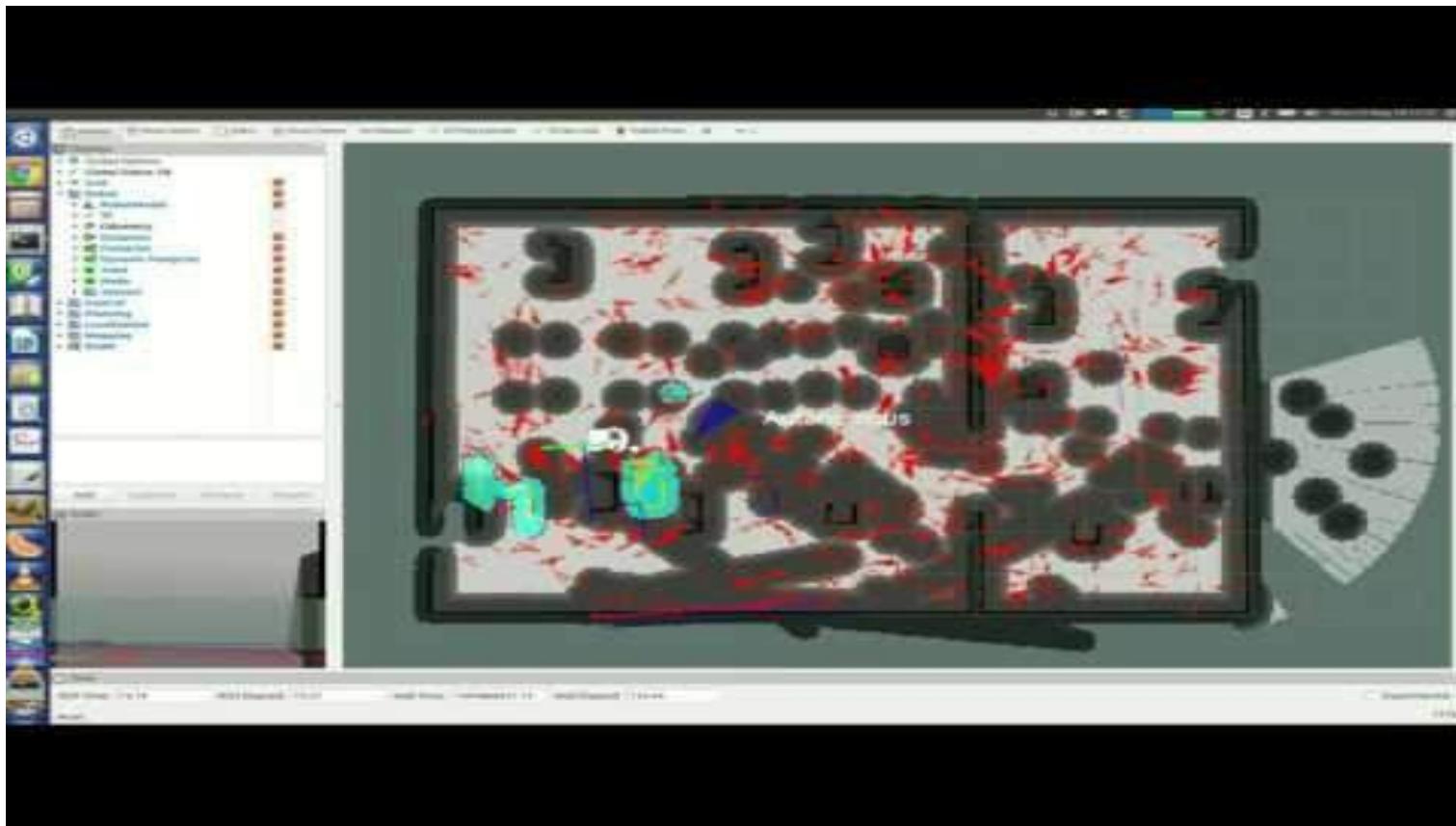
<http://wiki.ros.org/amcl> - implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.

http://wiki.ros.org/global_planner - provides an implementation of a fast, interpolated global planner for navigation. Dijkstra/A*

http://wiki.ros.org/base_local_planner - provides implementations of the Trajectory Rollout and Dynamic Window approaches to local robot navigation on a plane.

http://wiki.ros.org/move_base - links together a global and local planner to accomplish its global navigation task.

Localization and autonomous path planning with ROS



- <https://www.youtube.com/watch?v=JxVXtOY2g-o>

Robot Motion Planning

Representing planning
problems, configuration
space

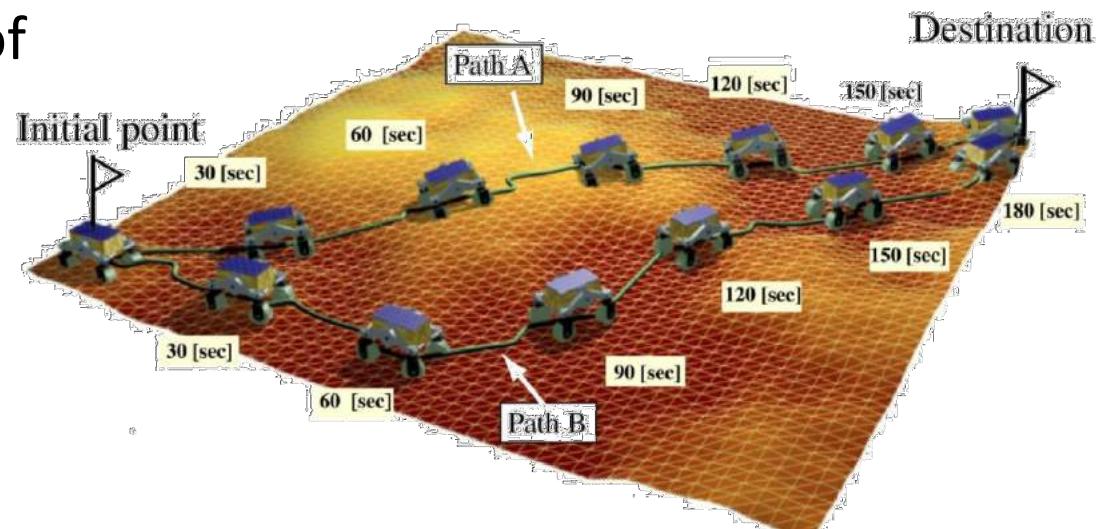
Graph search methods

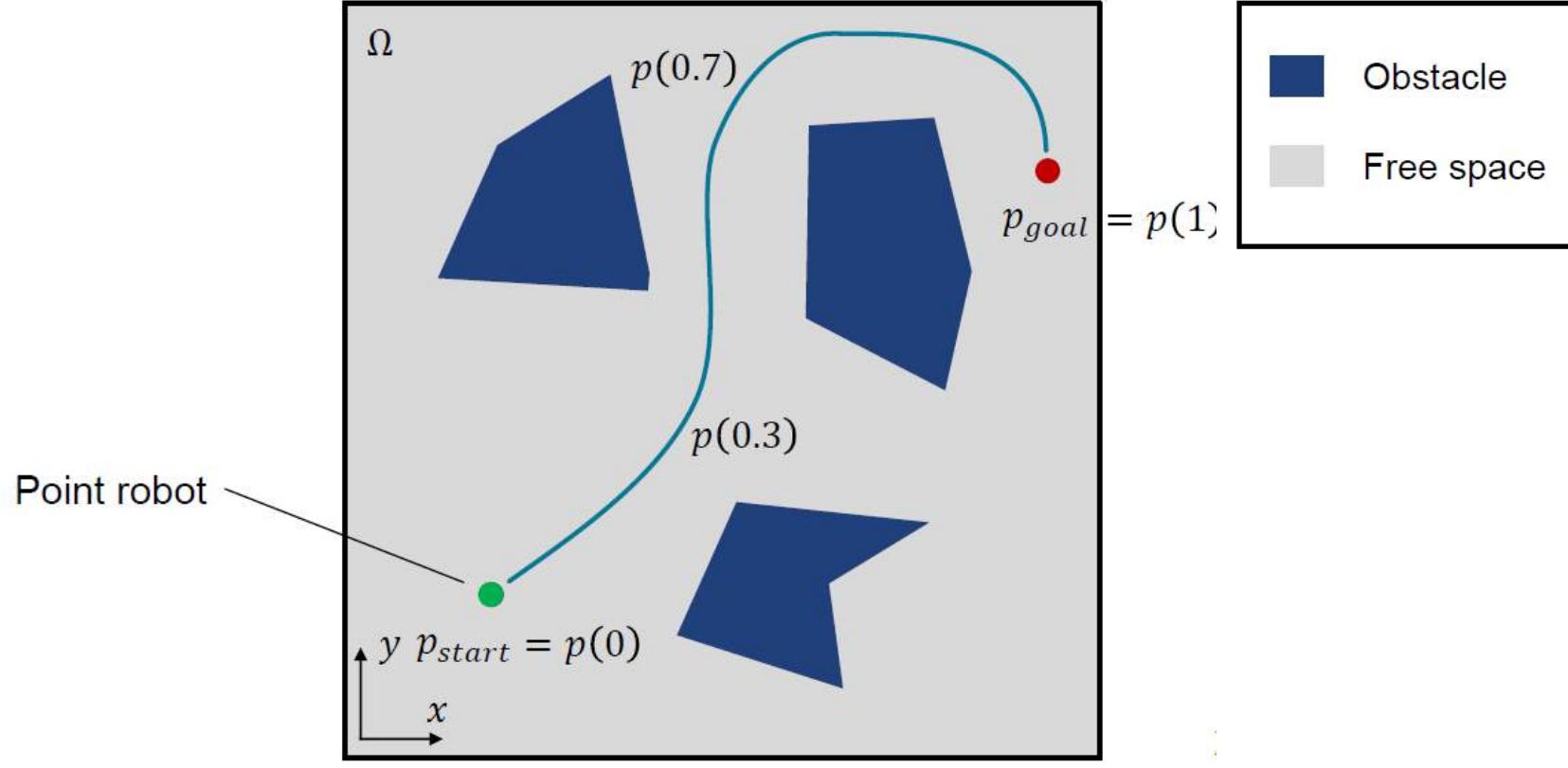
Potential fields

How can our robot feasibly move from one position to another?

Assumptions:

- Representation of robot and world
- Where I am and where do I go?
- Motion model

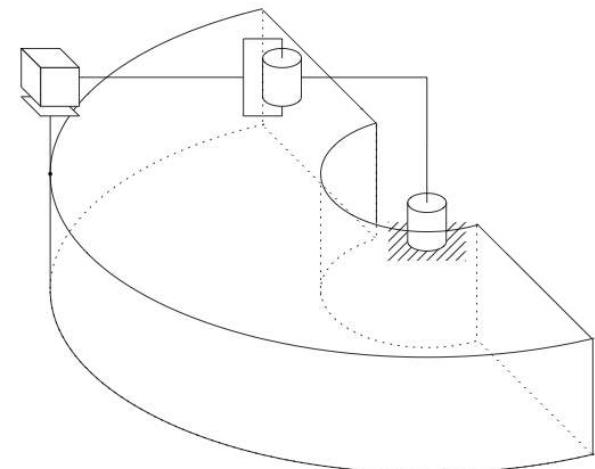




Representation

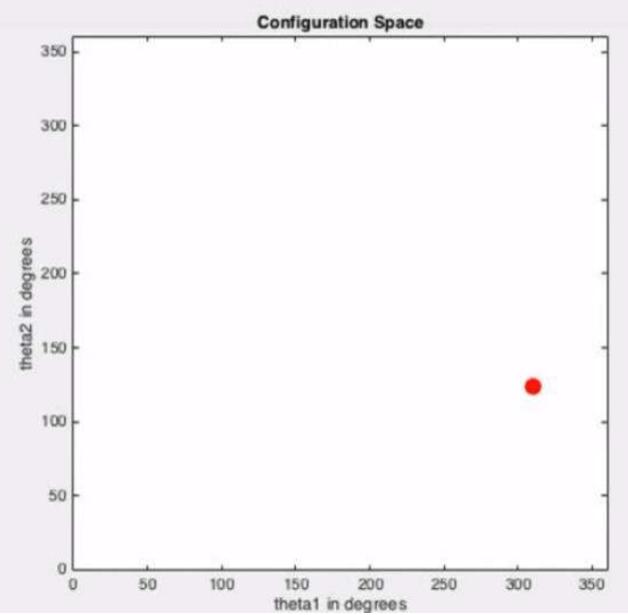
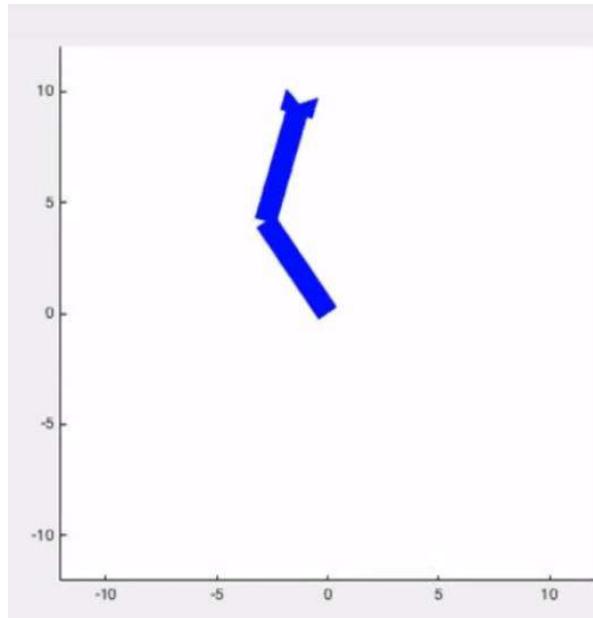
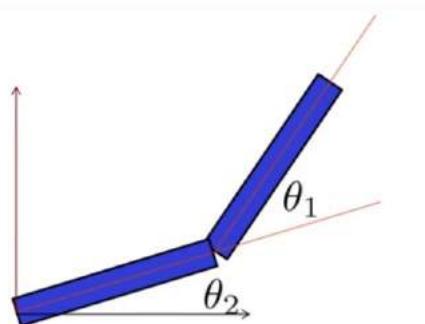
Workspace and Configuration Space

- Workspace: Reachable space within the environment (independent of the robot)
- Configuration space: Full state of the robot in the environment



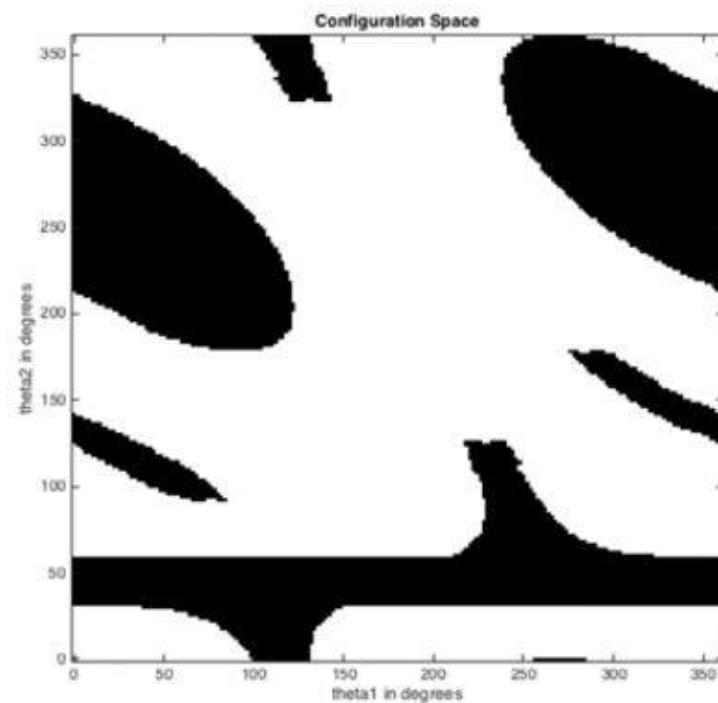
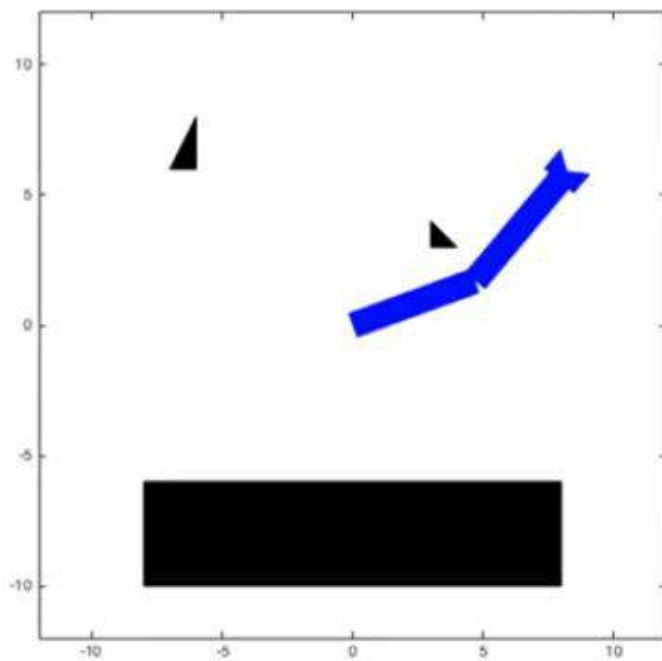
Configuration (C-)Space

alternative morphologies



Configuration (C-)Space

alternative morphologies

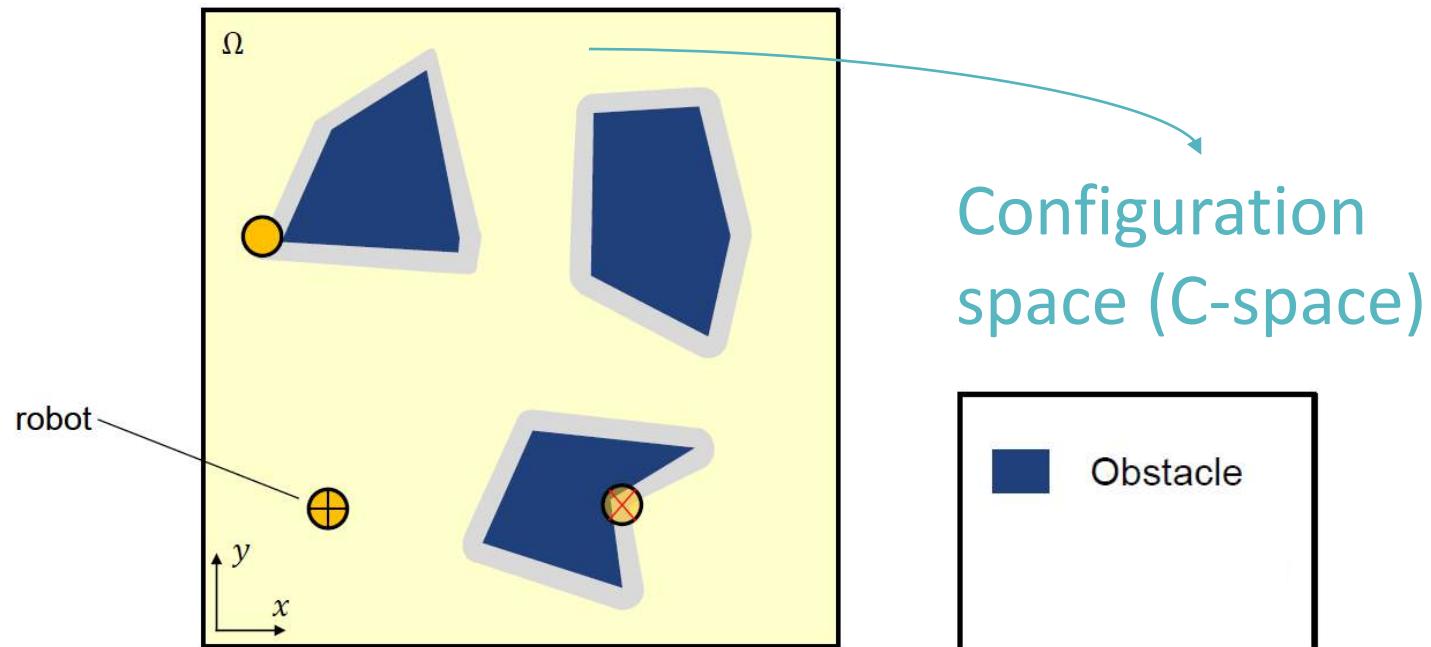


https://www.youtube.com/watch?time_continue=146&v=DU-EfHU0NEM

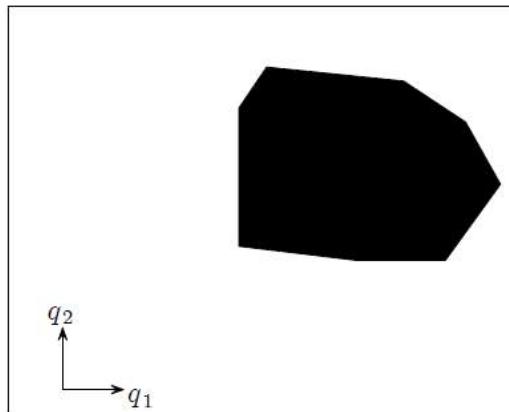
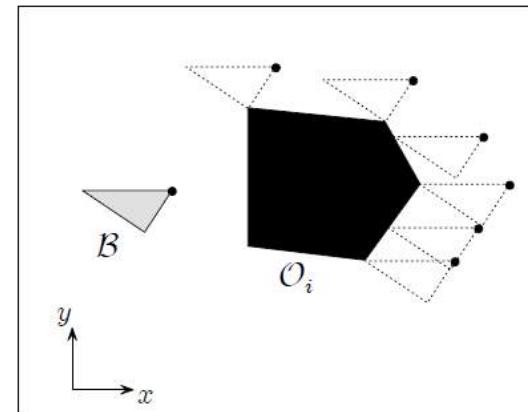
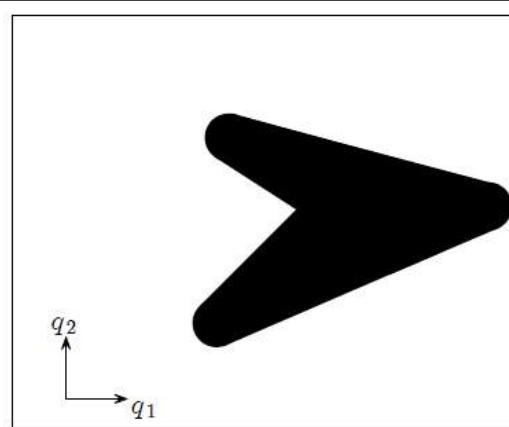
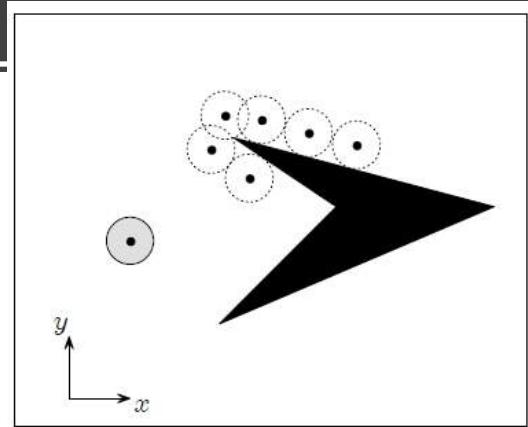
Why use a C-space

- Positions in configuration space tend to be close together for the robot
- Can be easier to solve collision checks, and join nearby poses
- Allows a level of abstraction that means solution methods can solve a wider range of problems
- Sometimes helps with wraparound conditions (rotational joints)

Configuration (C-)Space



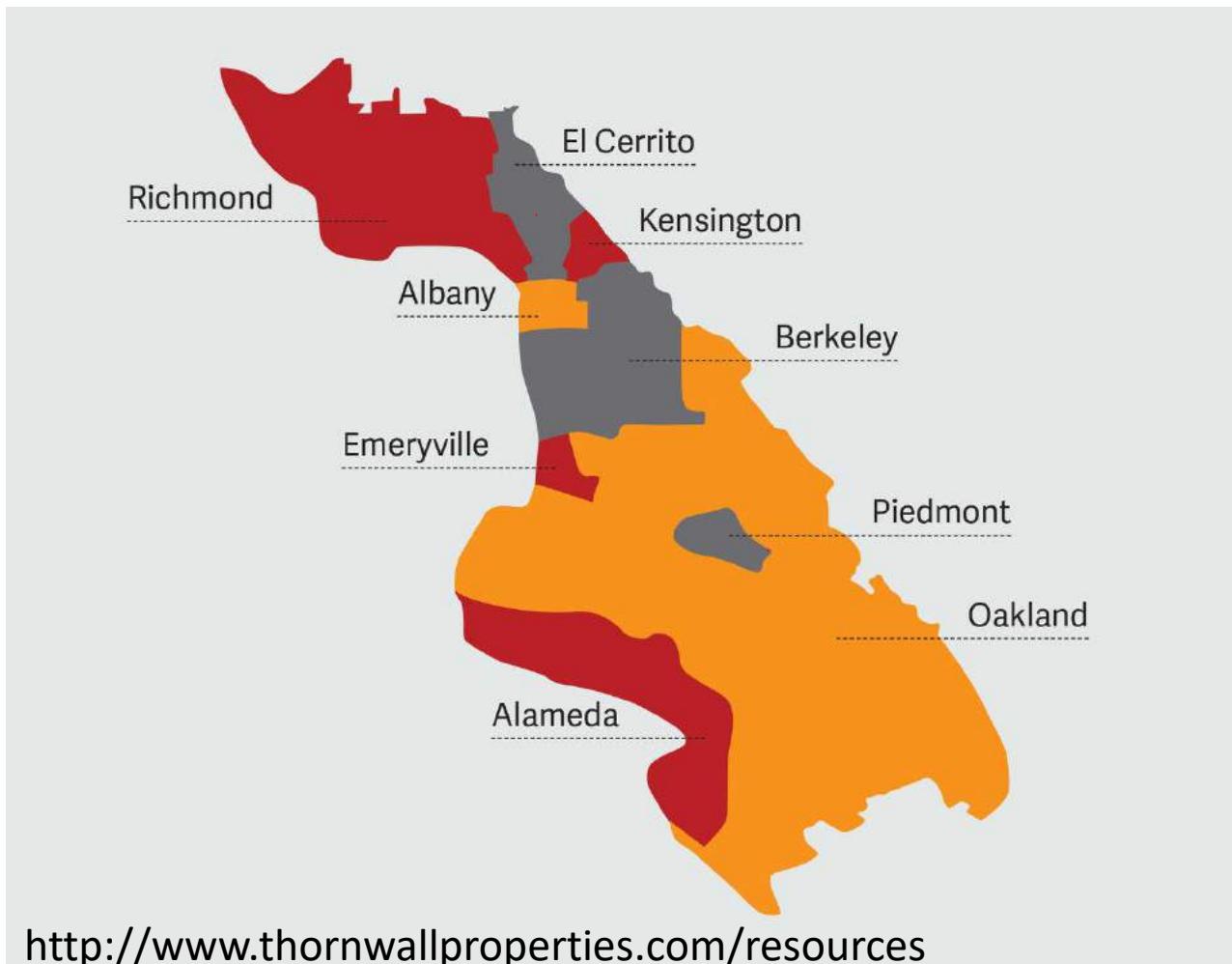
Configuration (C-)Space



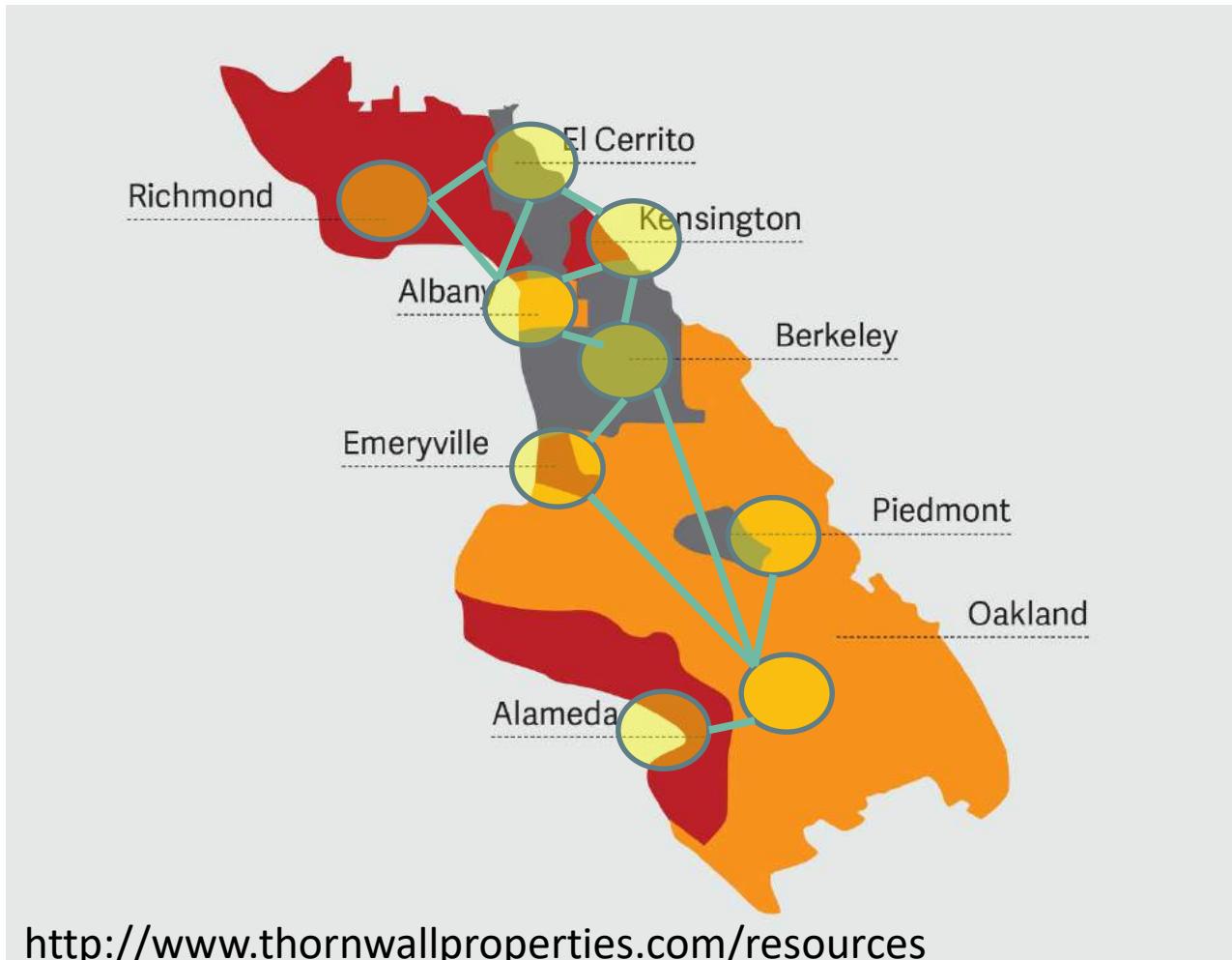
Continuous vs Discrete State Space Representations

- Convert a planning problem to some kind of discrete representation
- then use the discrete decomposition for path planning
 - Graph search: a connectivity graph is constructed (offline) and searched
 - Potential field planning: a mathematical function is imposed on the free space. The gradient of the function is followed to reach goal

An Example Graph: Neighborhood in the East Bay

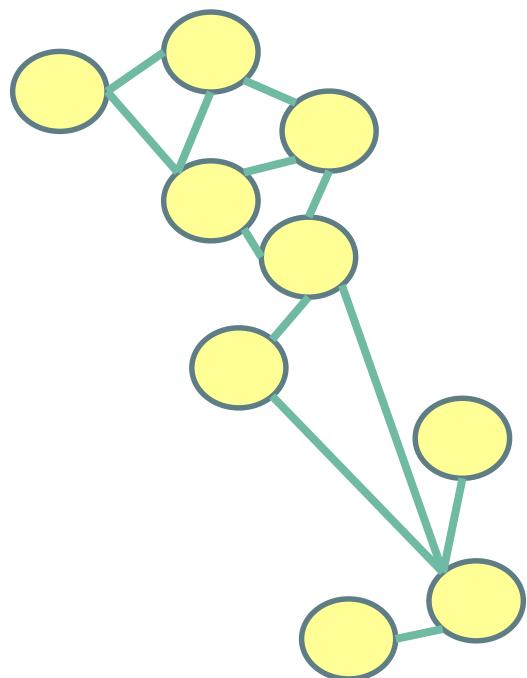


An Example Graph: Neighborhood in the East Bay

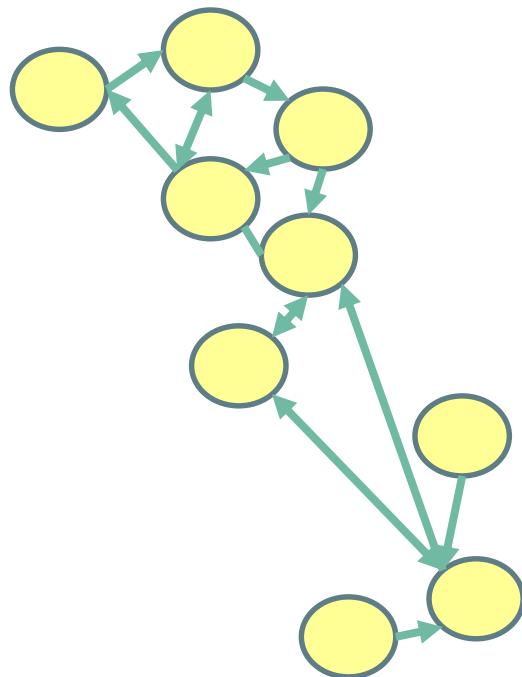


Types of Graphs

Undirected Graph

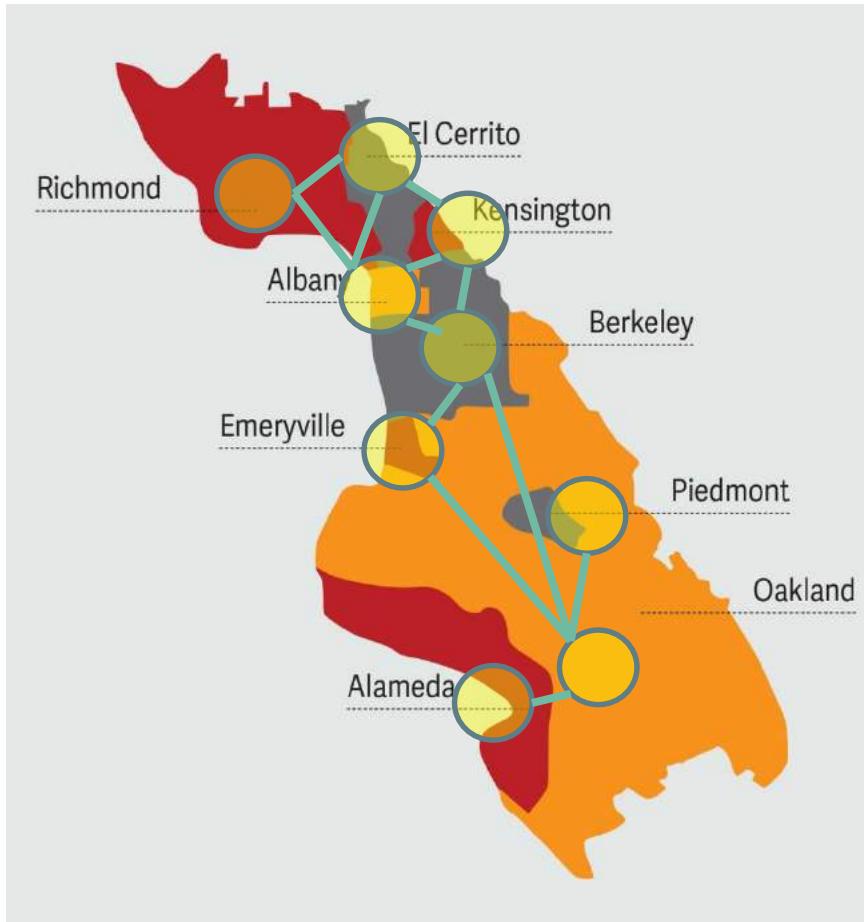


Directed Graph (Digraph)

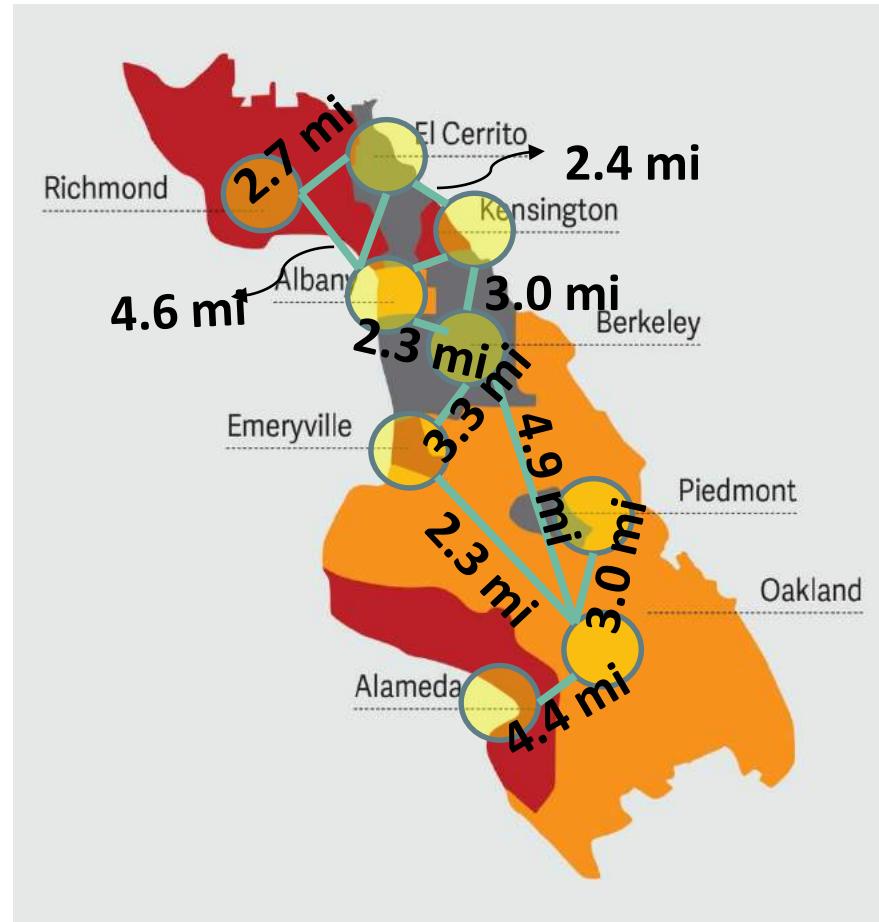


Types of Graphs

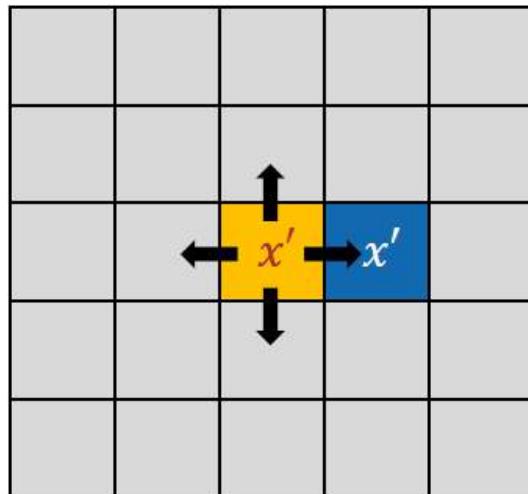
Unweighted Graph



Weighted Graph

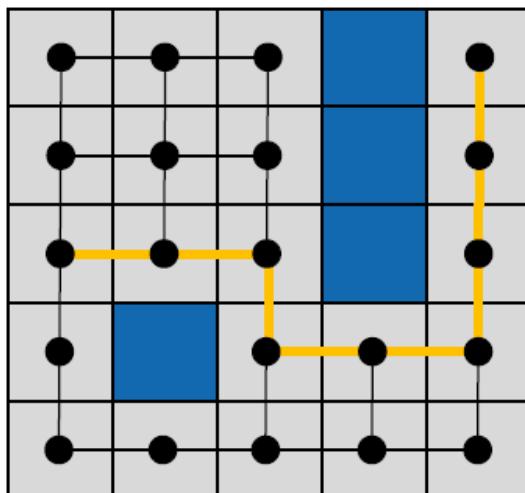


Discrete state space representation



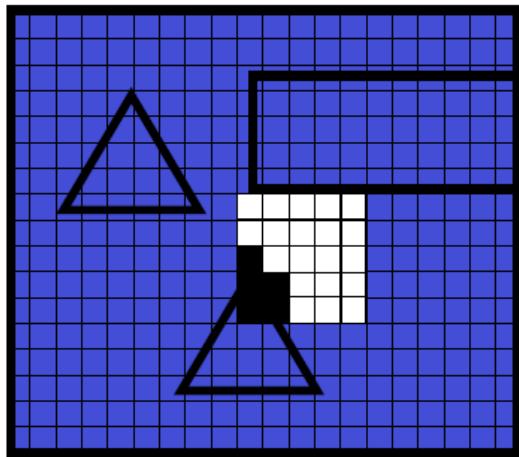
- Reduce continuous state space to a finite set of discrete **states (discrete state space representation)**
 - $x \in X$
- Define feasible **actions** from each state
 - $A(x) = \{a_0, a_1, \dots, a_n\}$
- And an associated **transition function**
 - $f(x, a) = x'$

Grid to graph



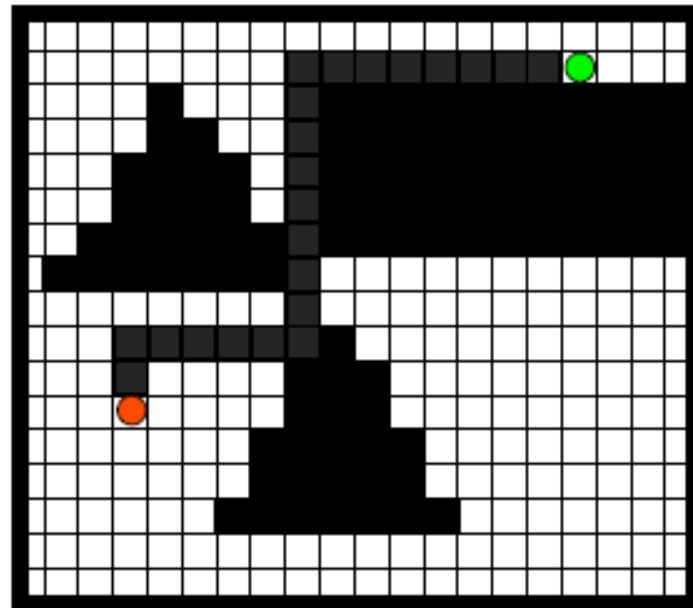
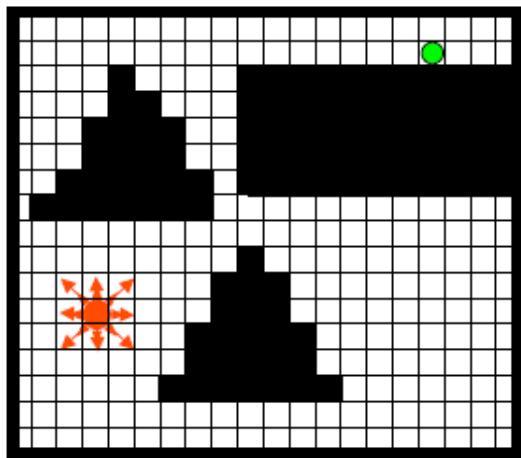
- Consider:
 - States as vertices
 - Transitions as directed edges
- Add:
 - Start node, x_s
 - Goal node, x_g
 - Cost function $C: X \times A \rightarrow \mathbb{R}^+$
- Finding the shortest path can be treated as a graph search problem

Turning a polygonal C-space into a grid?

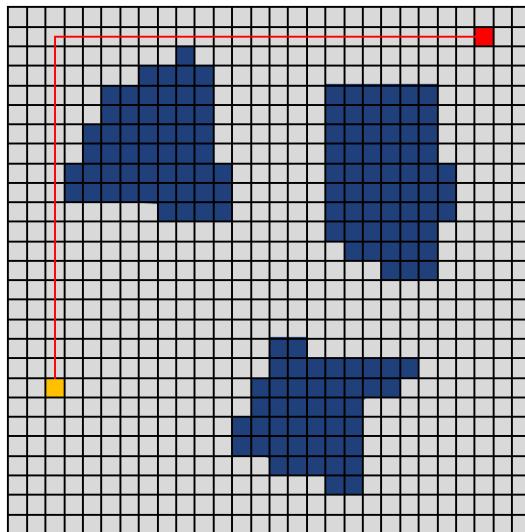


- A grid square is in the c-space if it is:
 - not inside an obstacle
 - further than the radius of the robot from all obstacle edges
- Algorithm:
 - Pick a grid square you know is in free space
 - Do breadth-first search (“flood-fill”) from that start square
 - As each square is visited by the search, compute the distance to all obstacle edges
 - label as “free” if the distance is greater than the radius of the robot or “occupied” if the distance is less
 - Once breadth-first search is done, also label all unlabelled squares as “occupied”

Perform search



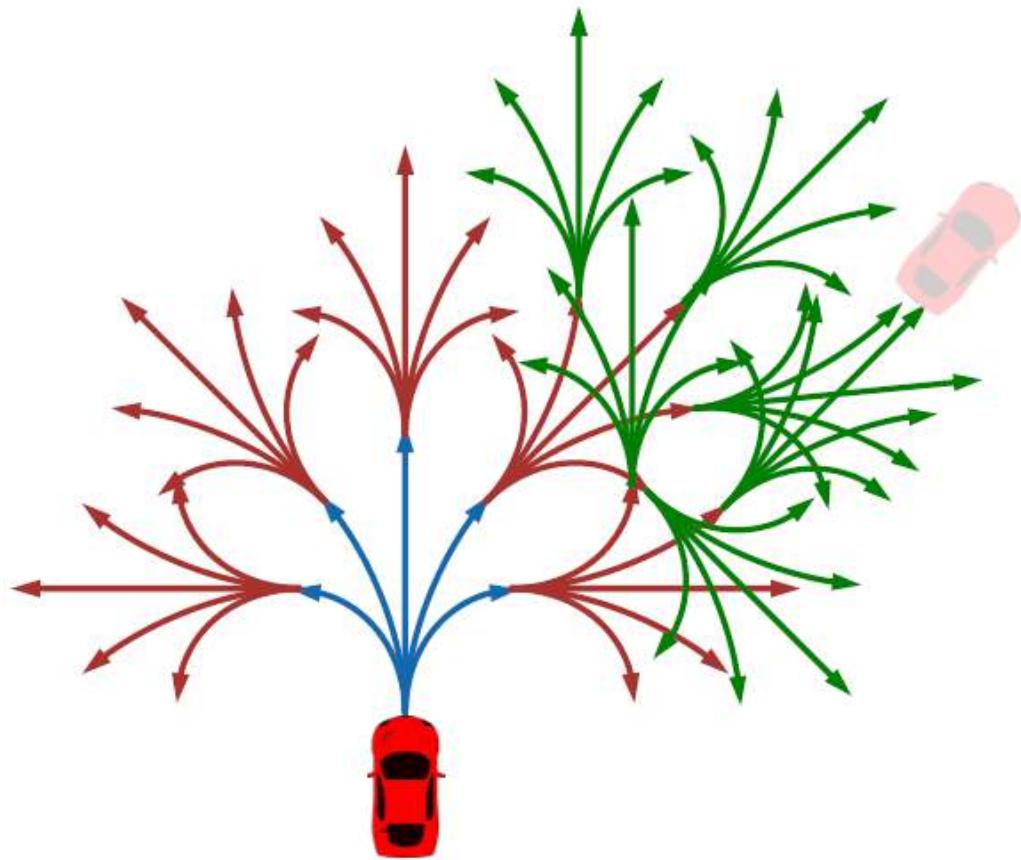
Issues with grid-based representations



- Loss of precision
- Selecting grid resolution
- Type of output path
- Poor scaling in higher dimensions

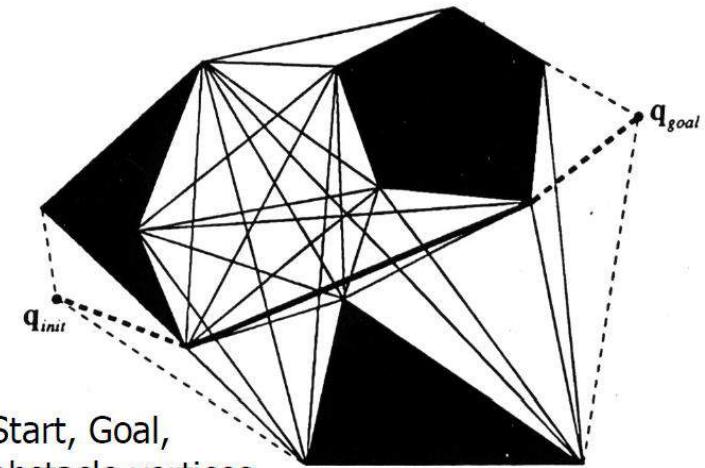
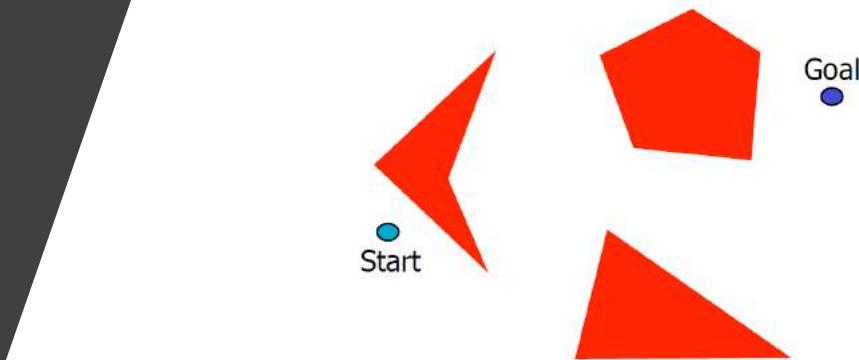
Grid lattice

- create a set of feasible motion primitives, and construct a tree (graph) that chains the motions into a sequence (plan)



Graph Construction: Visibility Graph

- Create edges between all pairs of mutually visible vertices
 - Search resulting graph
- ✓ Optimal plan!
- ✓ Good in sparse environments
- ✗ Limited to straight 2D motion
- ✗ Need polygonal obstacles
- ✗ Safety at stake



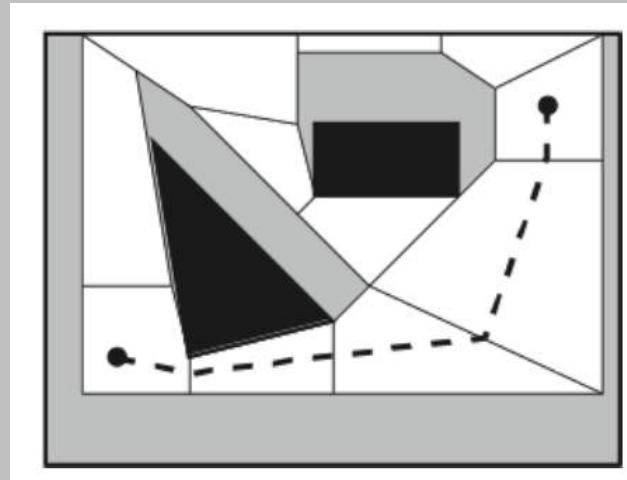
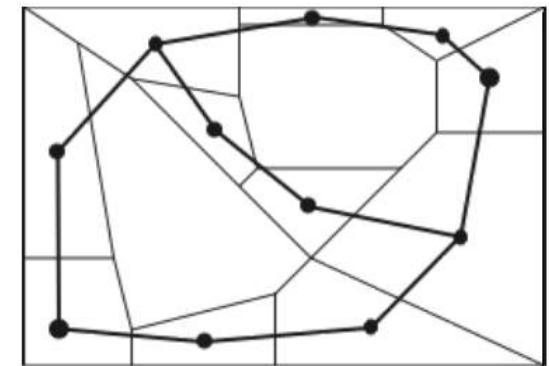
Vertices: Start, Goal, obstacle vertices

Edges: all combinations (v_i, v_j) that do not intersect any obstacle

Graph Construction: Voronoi Diagram

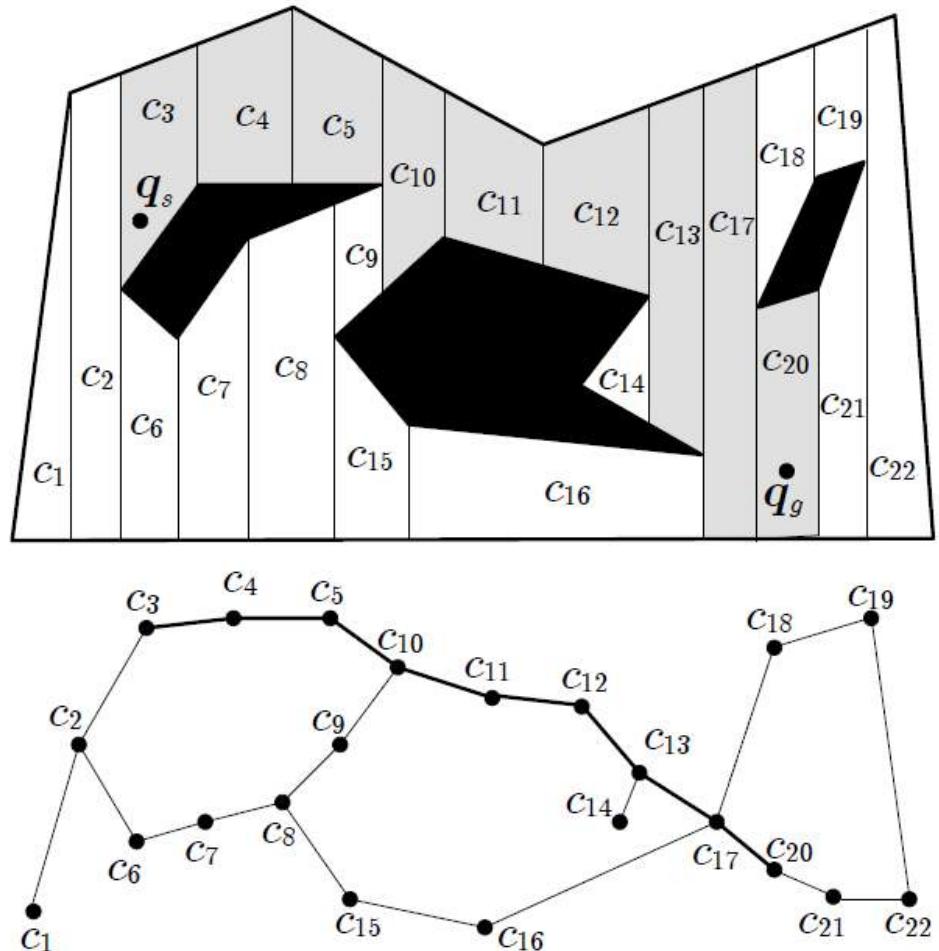
- Maximise the distance between the robot and the obstacles
- Draw equidistance lines
- Search resulting graph
-

- ✓ Complete
- ✓ Executability
- ✗ Not optimal
- ✗ Need long-range sensing



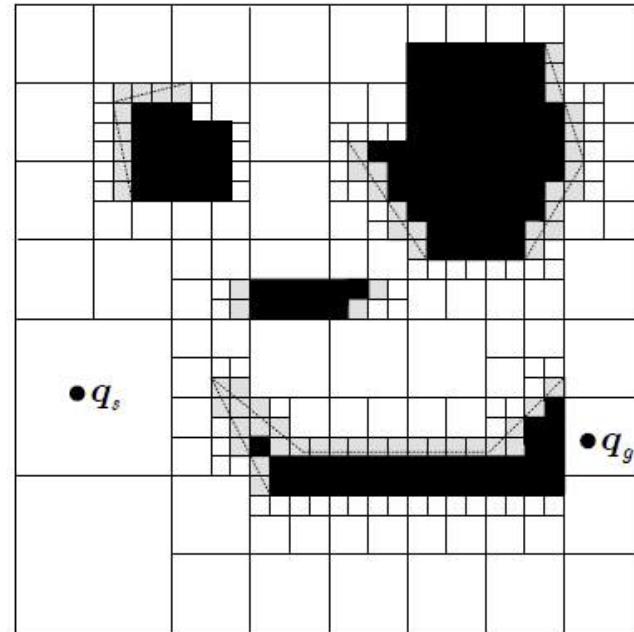
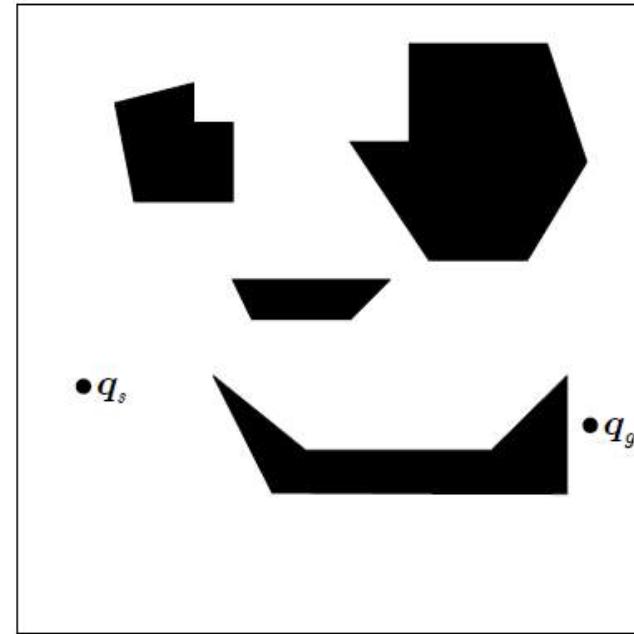
Graph Construction: Exact Cell Decomposition

- ✓ Complete
- ✗ Good in extremely sparse environments

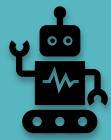


Graph Construction: Approximate Cell Decomposition

- Low computational complexity



Planning as Search



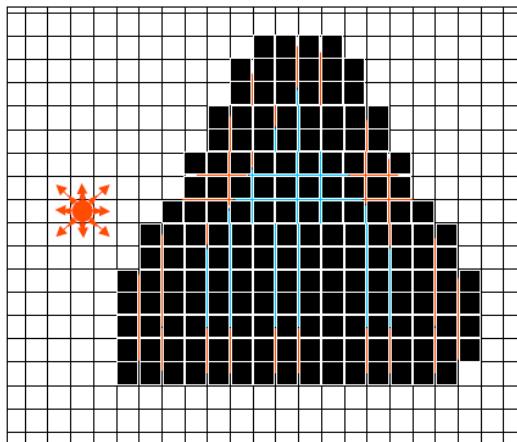
Given a representation, a start, a goal, and a motion model, how do we actually generate a plan?



We know how to search graphs and computers are good at it!

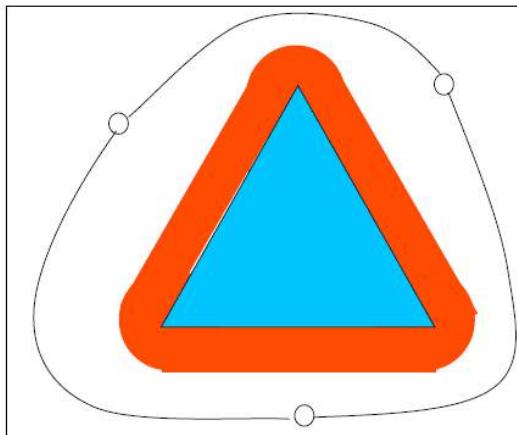
Convert problem to a graph
Search the graph
Profit!

Setting up the State Space



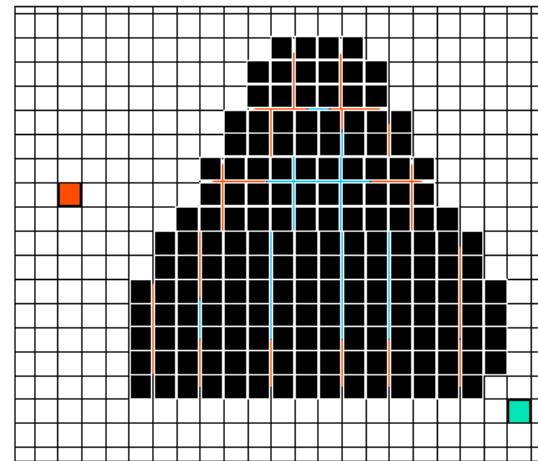
- Real space
- Configuration space
- State space
- Actions get you from one state to another
- Objective is to find a path from the start to the goal

Topological Discretizations

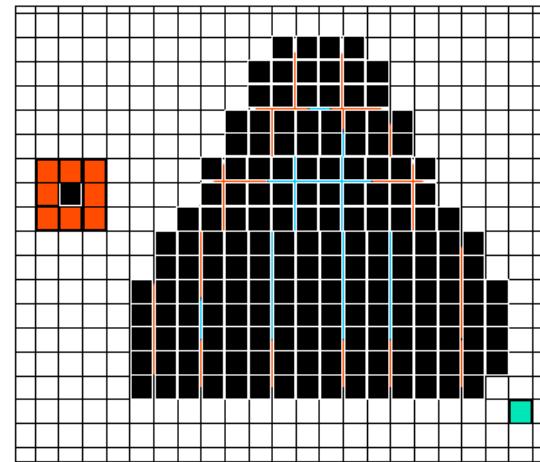
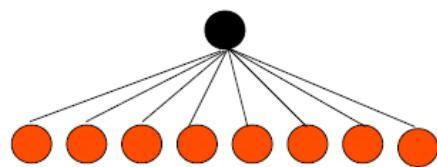


- State space could be states chosen from the c-space at random
- Sampling states at random is the “probabilistic roadmap”
- Visibility graph is optimal (in 2 dimensions and for point robots only)
- PRM is only optimal in the limit of infinite number of samples
- Trade-off: optimality vs. difficulty of computing configuration space exactly

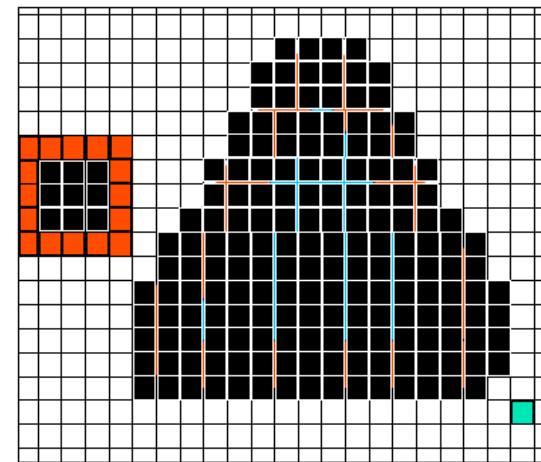
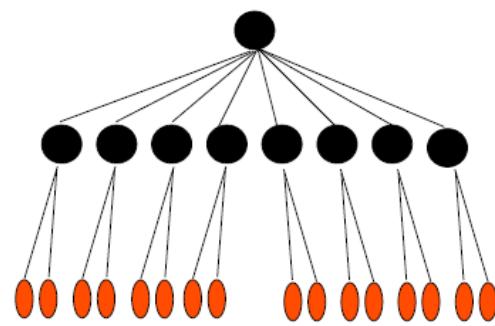
Tree Search



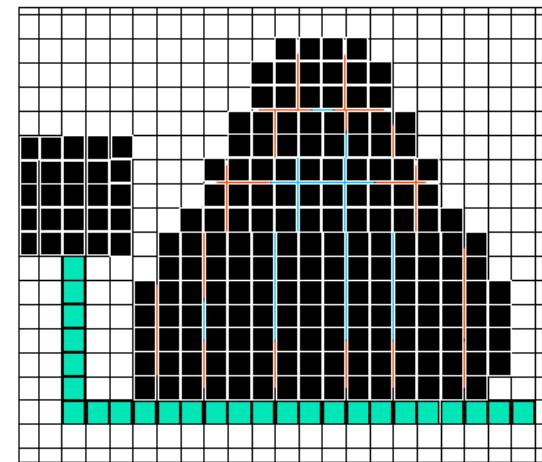
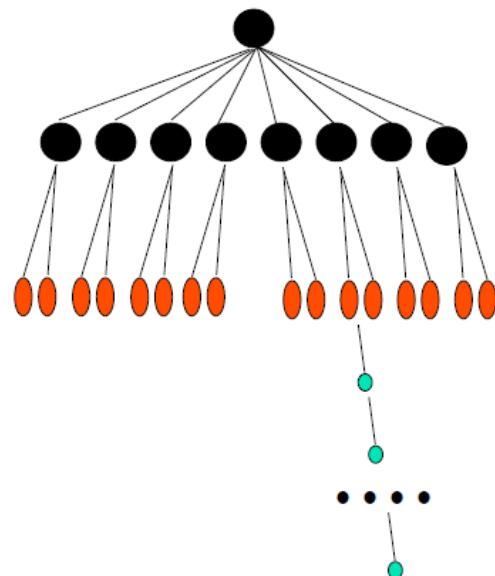
Tree Search



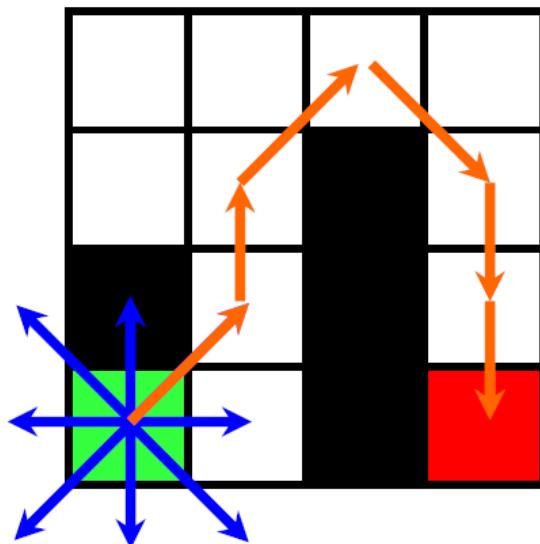
Tree Search



Tree Search

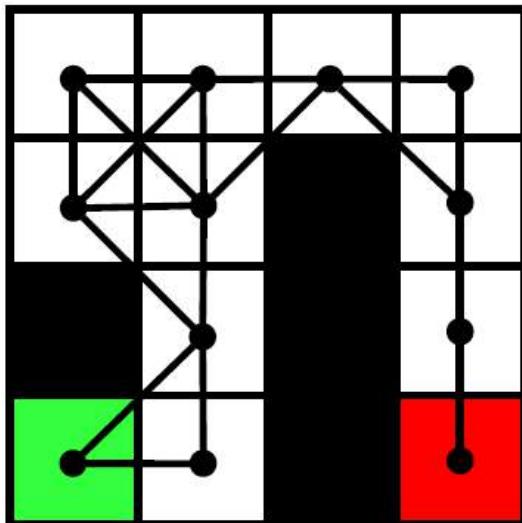


Setting up the State Space



- Real space
- Configuration space
- State space
- Actions get you from one state to another
- Objective is to find a path from the start to the goal

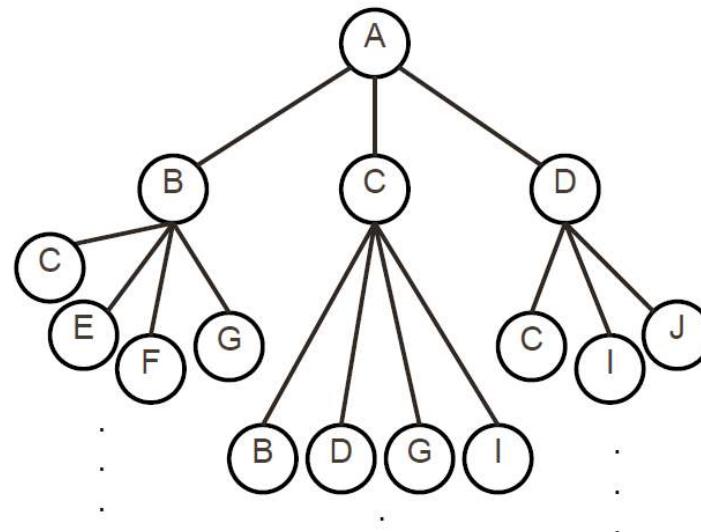
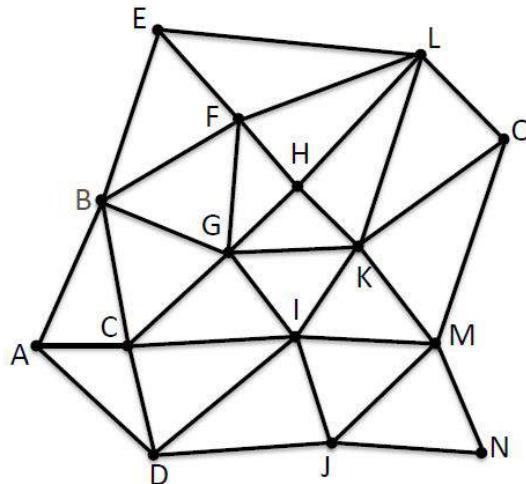
Setting up the State Space



- Search over the underlying graph
- Solve for paths from any point to any other point
- Assume all edge transitions are dynamically feasible

Search trees

- Construct a “tree” to search for optimal paths through the environment



Forward search

Node expansion:

- Mark a node “active”
- Explore its neighbours and mark them as “open”
- Mark the parent node as “visited”

Forward search

Node expansion:

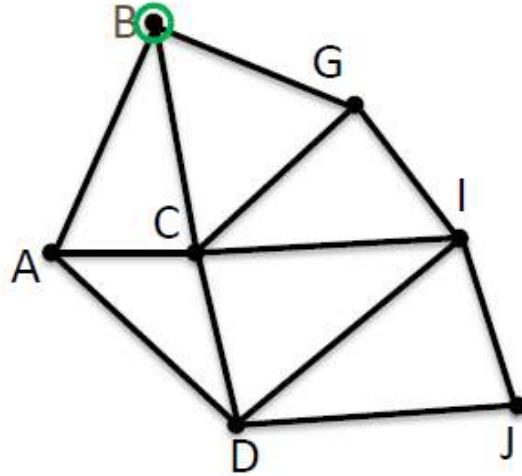
- Mark a node “active”
- Explore its neighbours and mark them as “open”
 - Open set: list of frontier (unexpanded) plans
 - Keeps track of what nodes to expand next
 - For each node in the open list, we know of at least one path to it from the start
- Mark the parent node as “visited”

Forward search

Node expansion:

- Mark a node “active”
- Explore its neighbours and mark them as “open”
 - Open set: list of frontier (unexpanded) plans
 - Keeps track of what nodes to expand next
 - For each node in the open list, we know of at least one path to it from the start
- Mark the parent node as “visited”
 - Closed set: nodes that have been expanded
 - For each node in the closed list, we’ve already found the lowest-cost path to it from the start

Breadth-First Search



Open (Q): Closed:

{B}

{}

Our (BFS) queue will be FIFO:

- push ($Q.Insert$) onto the end
- pop ($Q.GetFirst$) from the front



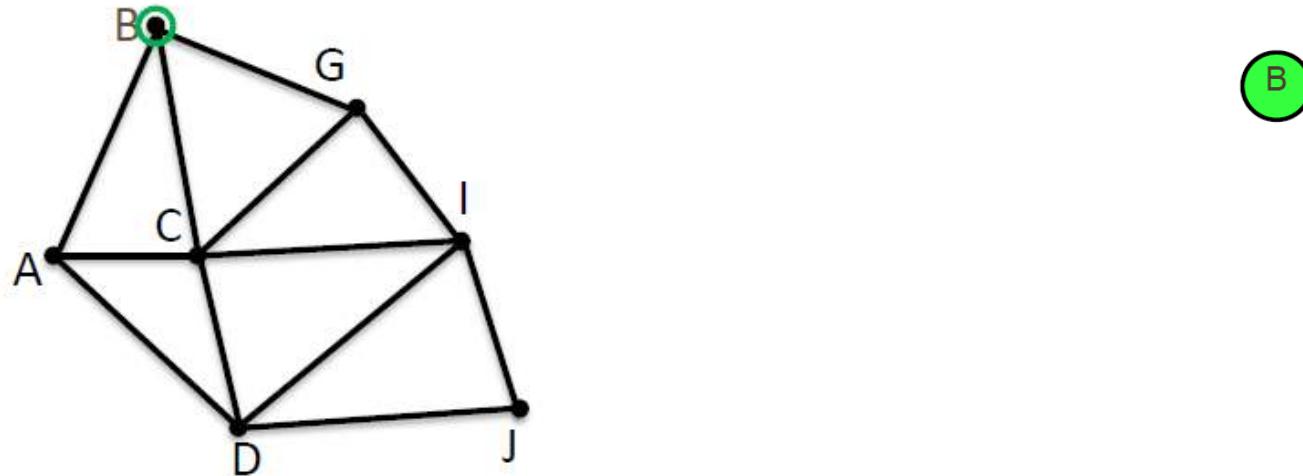
FORWARD_SEARCH

```
1   $Q.Insert(x_I)$  and mark  $x_I$  as visited
2  while  $Q$  not empty do
3       $x \leftarrow Q.GetFirst()$ 
4      if  $x \in X_G$ 
5          return SUCCESS
6      forall  $u \in U(x)$ 
7           $x' \leftarrow f(x, u)$ 
8          if  $x'$  not visited
9              Mark  $x'$  as visited
10              $Q.Insert(x')$ 
11         else
12             Resolve duplicate  $x'$ 
13     return FAILURE
```

Figure 2.4: A general template for forward search.

LaValle, Steven M. *Planning algorithms*. Cambridge university press, 2006, p. 33

Breadth-First Search



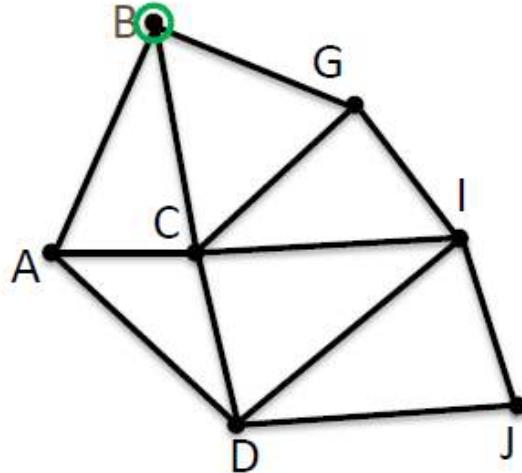
BFS

- Complete (will find the solution if it exists)
- Guaranteed to find the shortest path (number of edges, no weights)
- First solution that is found is the optimal path
- Time complexity $O(|V|+|E|)$
- Names in robotics:
 - Wavefront
 - Forest fire

Depth-First Search

- DFS starts at the root node and explores as far as possible along each branch
- Similar implementation to BFS, but with a stack (last-in first-out) queue

Depth-First Search



Open (Q): Closed:

{B}

- Our (DFS) queue will be LIFO:
- push ($Q.Insert$) onto the front
 - pop ($Q.GetFirst$) from the front



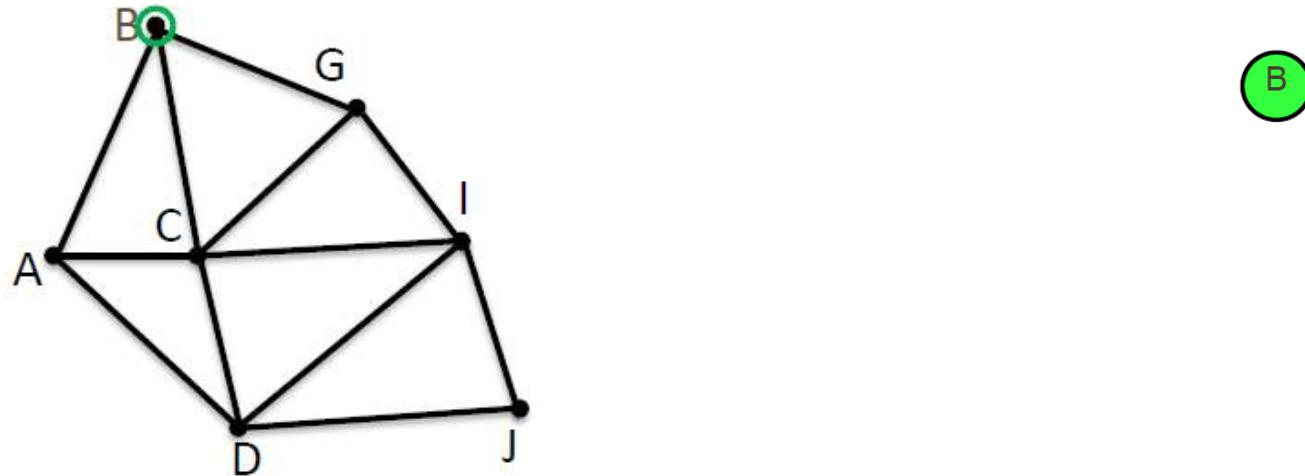
FORWARD_SEARCH

```
1   $Q.Insert(x_I)$  and mark  $x_I$  as visited
2  while  $Q$  not empty do
3       $x \leftarrow Q.GetFirst()$ 
4      if  $x \in X_G$ 
5          return SUCCESS
6      forall  $u \in U(x)$ 
7           $x' \leftarrow f(x, u)$ 
8          if  $x'$  not visited
9              Mark  $x'$  as visited
10              $Q.Insert(x')$ 
11         else
12             Resolve duplicate  $x'$ 
13     return FAILURE
```

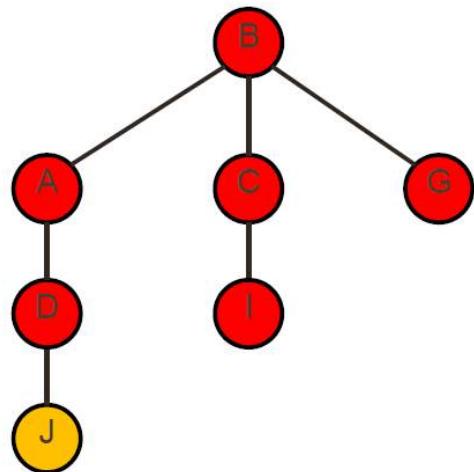
Figure 2.4: A general template for forward search.

LaValle, Steven M. *Planning algorithms*. Cambridge university press, 2006, p. 33

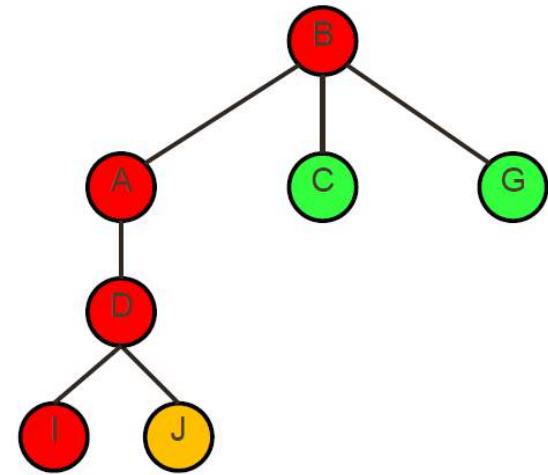
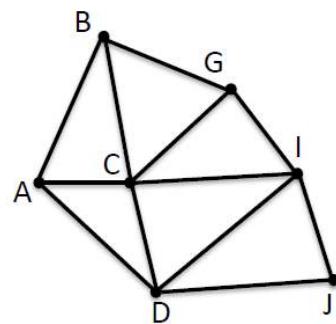
Depth-First Search



BFS vs DFS



BFS

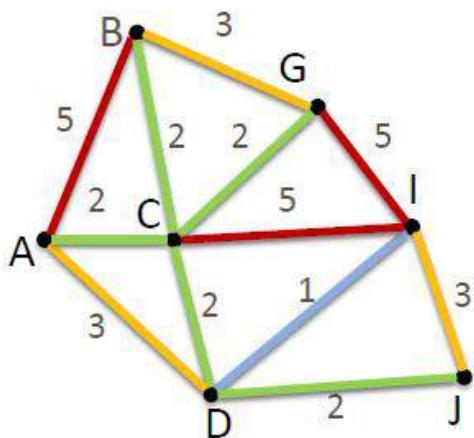


DFS

DFS vs BFS

- DFS not complete for infinite trees
 - may explore an incorrect branch infinitely deep
 - never come back up
- BFS *is* complete
- DFS has lower memory footprint than BFS with high-branching
- Not often used for path search, but to completely explore a graph
- Both are simple to implement
- Both has time complexity $O(|V|+|E|)$

Dijkstra's Algorithm



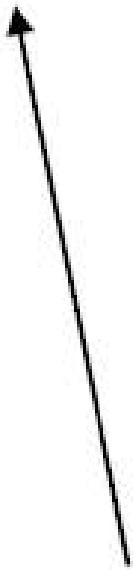
- BFS with edge costs: Expanding in order of closest to start
- Asymptotically the fastest known single-source shortest path algorithm for arbitrary directed graphs
- **Open** queue is ordered according to currently known best cost to arrive

Open (Q):

{B(0)}

Closed:

{B(0)}

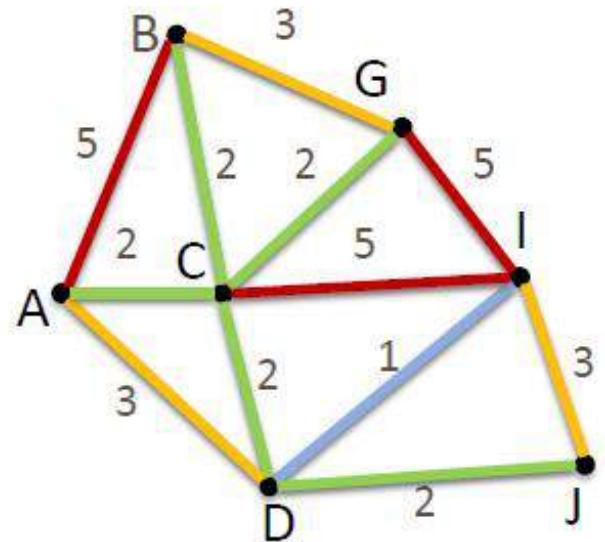


Our Dijkstra queue will be ordered by cost to arrive:

- push (*Q.Insert*) by cost
- pop (*Q.GetFirst*) from the front, and add it to the closed list

Dijkstra's Algorithm

B(0)



Dijkstra's Algorithm

- We can recover the lowest-cost route from the start to any node (or any node with cost $<$ goal if we terminate at a goal)
- Easy to implement, with management with the priority queue
- Due to heap operations, time complexity becomes $O(|V| \log |V| + |E|)$
- Doesn't really know the goal exists until it reaches it
 - Can we incorporate our knowledge of the goal to expand nodes that are closer to the goal earlier?
 - Can we do it without breaking the condition that a node is only accepted with its lowest cost of arrival?

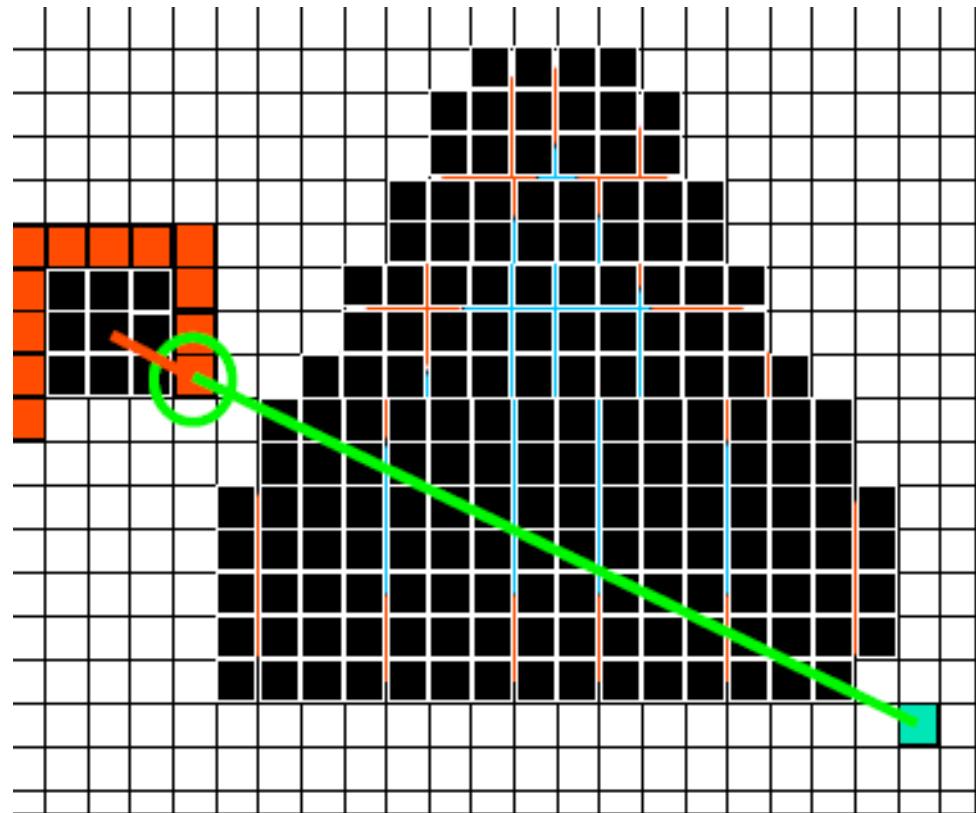
Informed Search – A*

Use domain knowledge to bias the search

Favour actions that might get closer to the goal

Each state gets a value
 $f(x) = g(x) + h(x)$

Choose the state with best f



Informed Search – A*

Use domain knowledge to bias the search

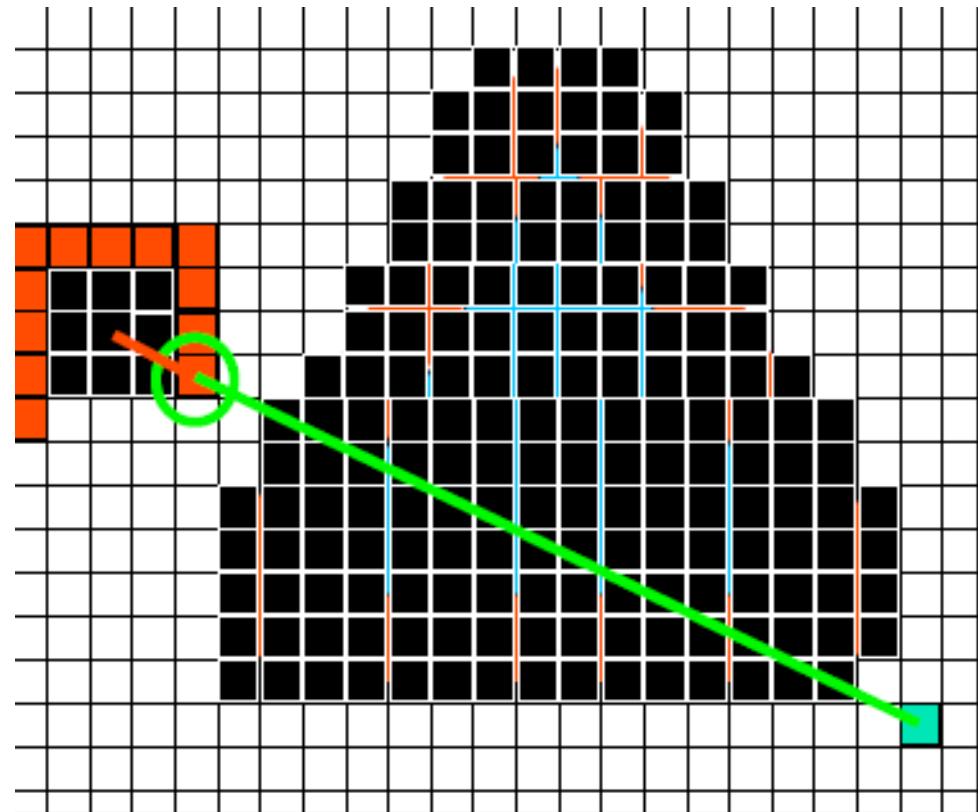
Favour actions that might get closer to the goal

Each state gets a value

$$f(x) = g(x) + h(x)$$

Cost incurred so far, from the start state

Estimated cost from here to the goal: “heuristic” cost



Informed Search – A*

Use domain knowledge to bias the search

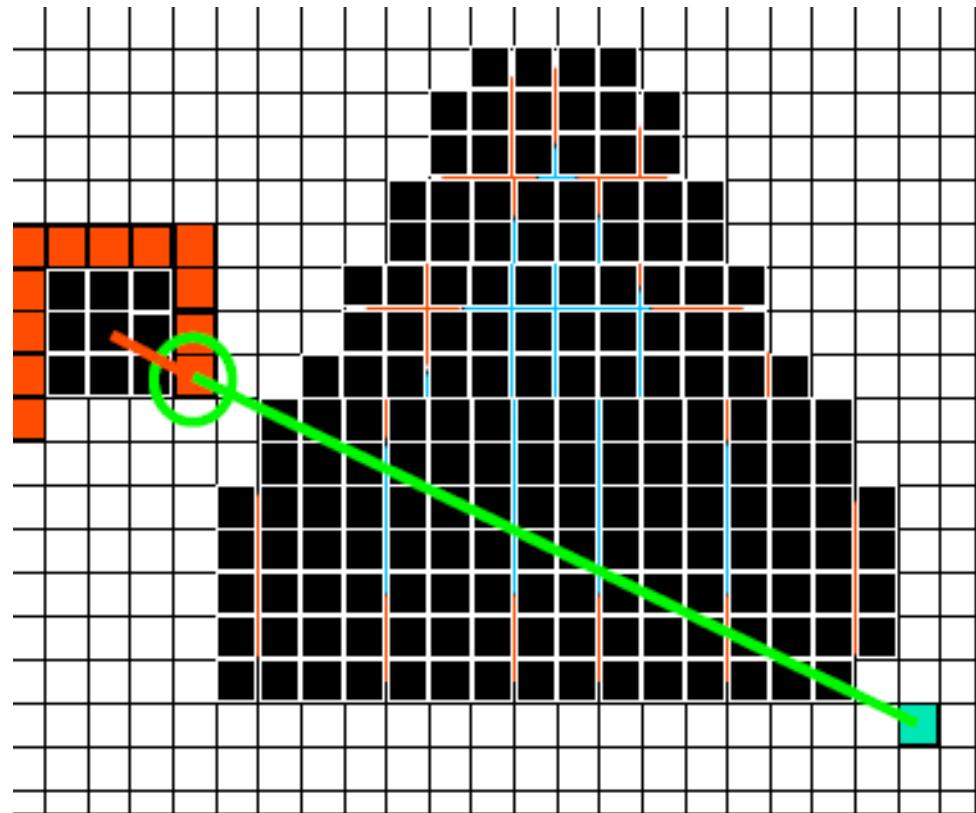
Favour actions that might get closer to the goal

Each state gets a value

$$f(x) = g(x) + h(x)$$

Cost incurred so far, from the start state

Estimated cost from here to the goal: "heuristic" cost



Example:

$$g(x) = 3$$

$$h(x) = ||x-g|| = \sqrt{8^2+11^2} = 19.7$$

$$f(x) = 22.7$$

Informed Search – A*

Use domain knowledge to bias the search

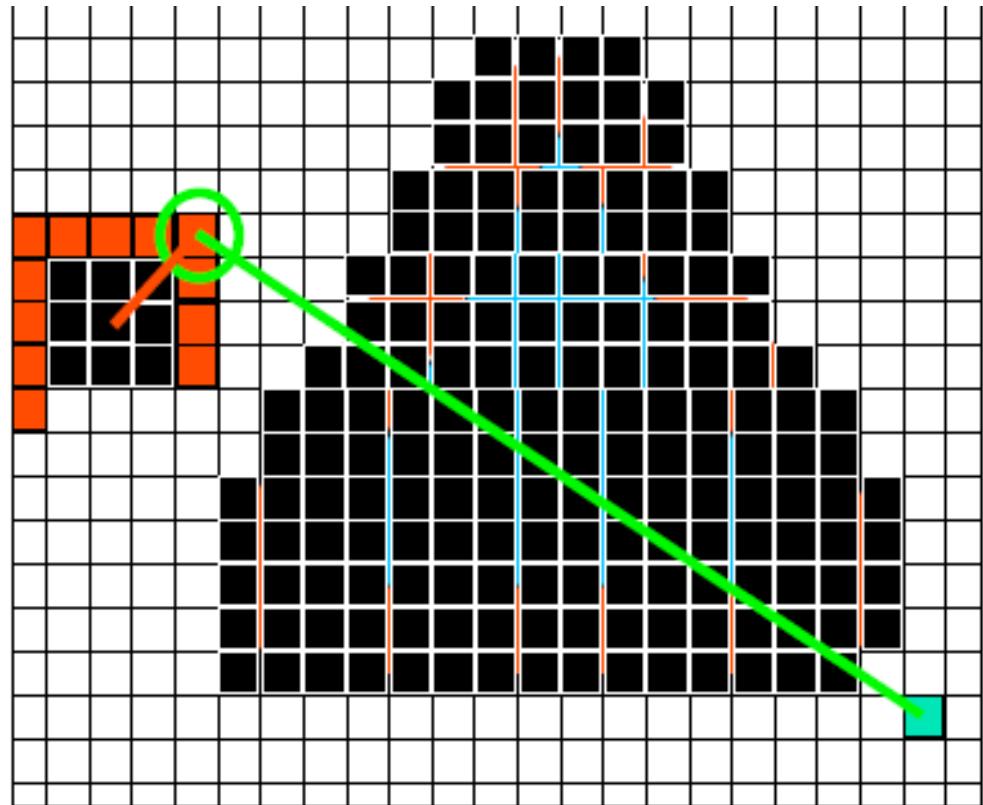
Favour actions that might get closer to the goal

Each state gets a value

$$f(x) = g(x) + h(x)$$

Cost incurred so far, from the start state

Estimated cost from here to the goal: “heuristic” cost



Example:

$$g(x) = 4$$

$$h(x) = ||x-g|| = \sqrt{11^2+18^2} = 21.1$$

$$f(x) = 25.1$$

How to Choose Heuristics?

- The closer $h(x)$ is to the optimal cost to the goal, $h^*(x)$, the more efficient the search!
- The heuristic must be ***admissible***
 - It never overestimates the cost
 $h(x) \leq h^*(x)$ to guarantee that A* finds the lowest-cost path
- The heuristic must be ***consistent***
 $h(x) \leq d(x,y) + h(y)$ for any pair of adjacent nodes x and y,

Decisions... decisions 😊

How is your map described?

- Is it a grid map?
- Is it a list of polygons?

What kind of controller do you have?

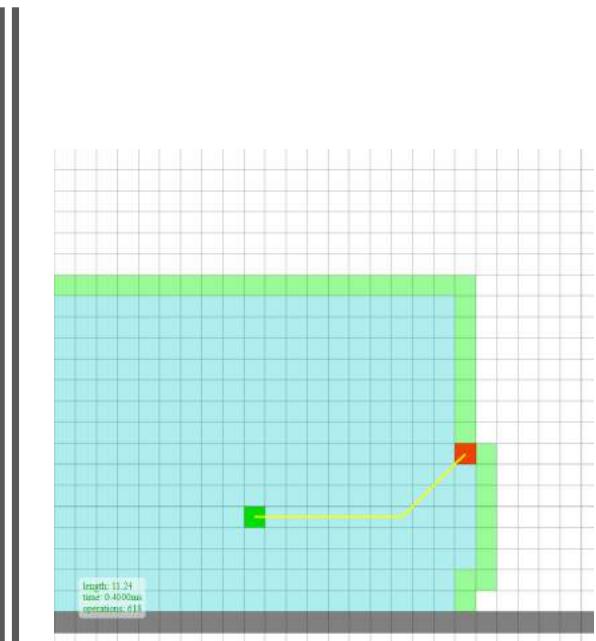
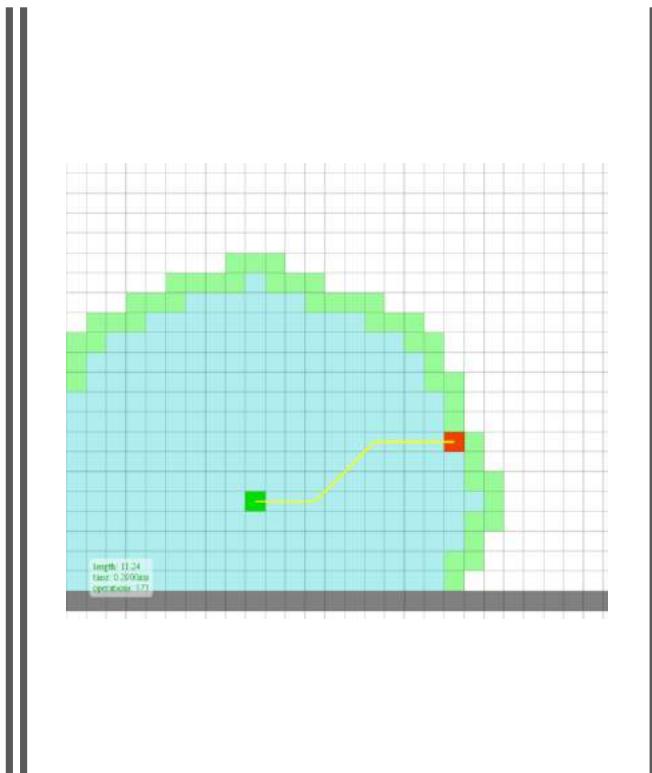
- Do you just have controllers on distance and orientation?
- Do you have behaviours, e.g. follow walls?

What do you care about?

- The shortest path?
- The fastest path?

What kind of search to use?

- Do you have a good heuristic? If so, then maybe A* is a good idea.

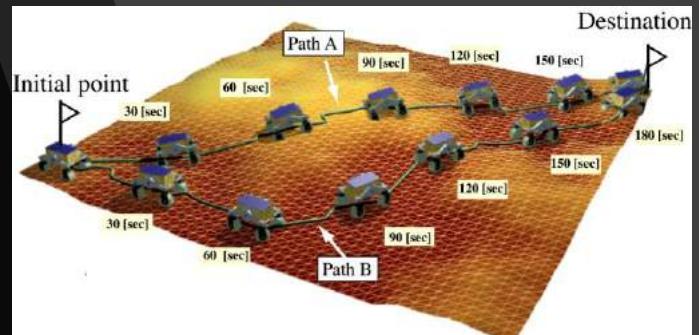


Comparison

<https://qiao.github.io/PathFinding.js/visual/>

Randomised Graph Search

- Complexity of breadth-first algorithm in a uniform grid as a function of the number of dimensions $O(|V|+|E|)$
- Number of nodes in a
 - 2D grid $100 \times 100 = 10^4$
 - 3D grid $100 \times 100 \times 100 = 10^6$
 - 6D grid 100 cells per d is 10^{12}



Randomised Graph Search



divide the region **uniformly** into small cells, one approach is to **randomly sample** locations in the space and try and connect the samples

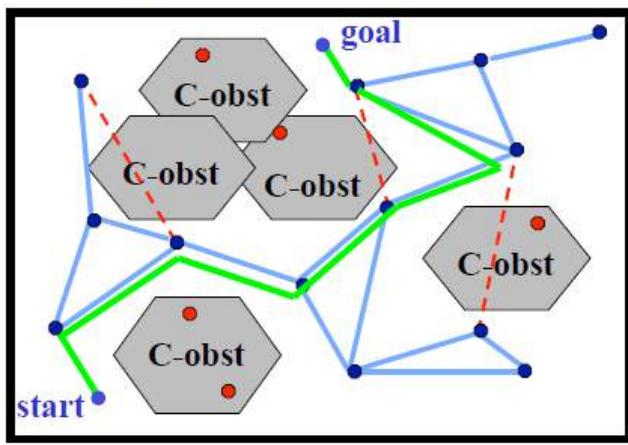


Often, a larger proportion of the working volume is free space, so if two points are 'near' each other, it is often the case that they can be connected by a simple path (e.g. straight line)

Probabilistic Road Maps (PRM)

[Kavraki et al. 96]

C-space



Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes)
 - discard nodes that are invalid
2. Connect pairs of nodes to form **roadmap**
 - simple, deterministic *local planner* (e.g., straight line)
 - discard paths that are invalid

Query processing

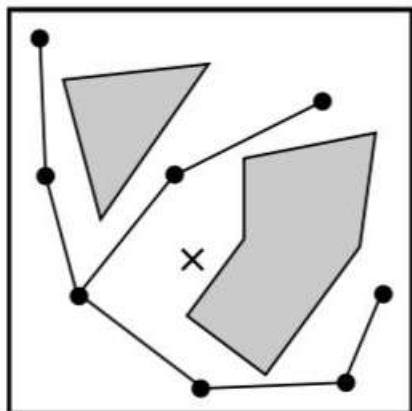
1. Connect *start* and *goal* to roadmap
2. Find path in roadmap between *start* and *goal*
 - regenerate plans for edges in roadmap

Primitives Required:

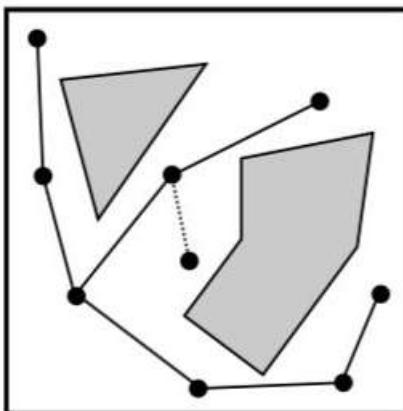
1. Method for Sampling points in C-Space
2. Method for “validating” points in C-Space

PRM Algorithm

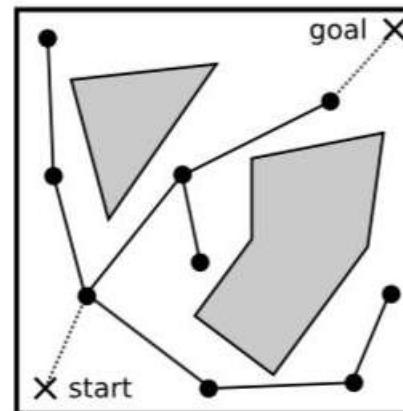
(1) PRM Algorithm



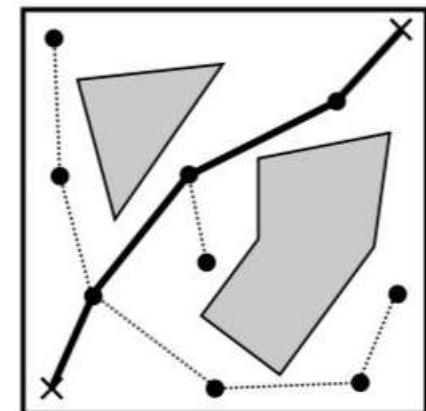
(a) The *learning* phase: a random sample, denoted by \times , is generated



(b) A local planner is used to connect the new sample to nearby roadmap vertices.



(c) The *query* phase: the start and goal configurations are added to the roadmap.

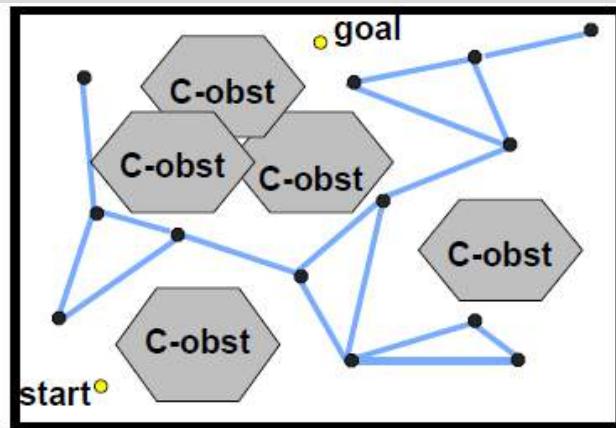


(d) A graph search algorithm is used to connect the start and goal through the roadmap.

PRMs

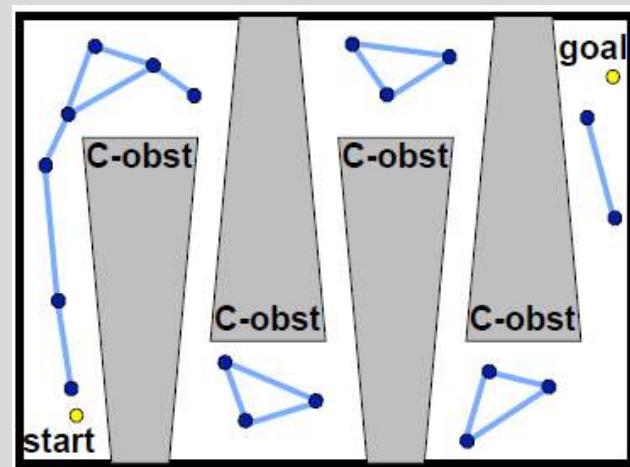
Pros

- *probabilistically complete*
- applied easily to high-dimensional C-space
- support fast queries with enough pre-processing



Cons

- don't work as well for some problems:
 - unlikely to sample nodes in *narrow passages*
 - only *probabilistically complete*

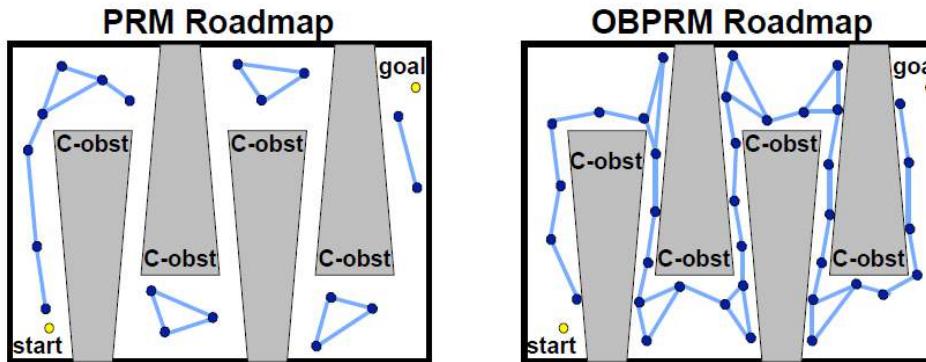


Sampling Around Obstacles

[Amato et al 98]

To Navigate Narrow Passages we must sample in them

- most PRM nodes are where planning is easy (not needed)

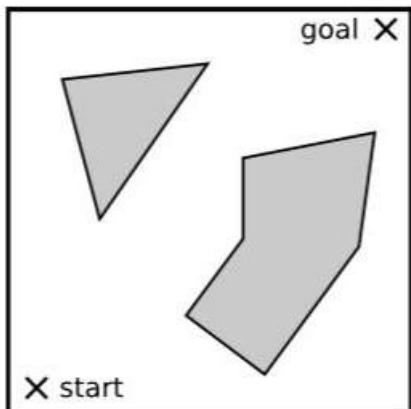


Idea: Can we sample nodes near C-obstacle surfaces?

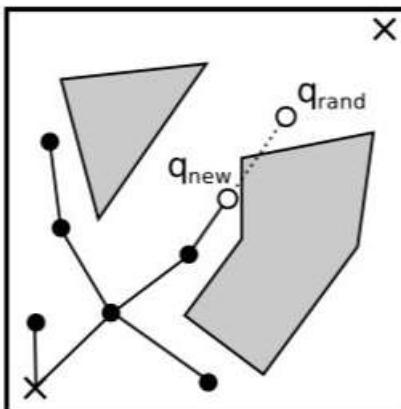
- we cannot explicitly construct the C-obstacles...
- we do have models of the (workspace) obstacles...

PRM Algorithm

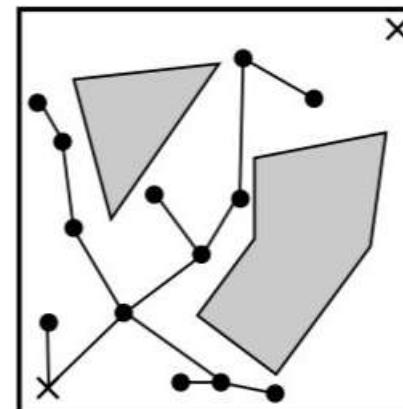
(2) RRT Algorithm



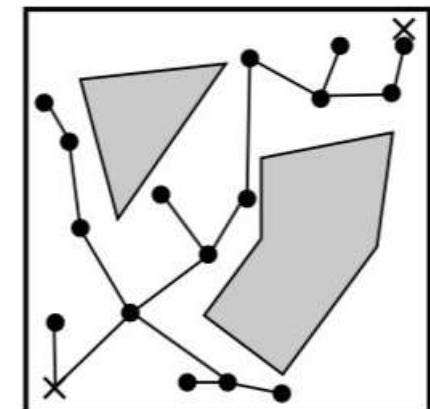
(a) A tree is grown from the start configuration towards the goal.



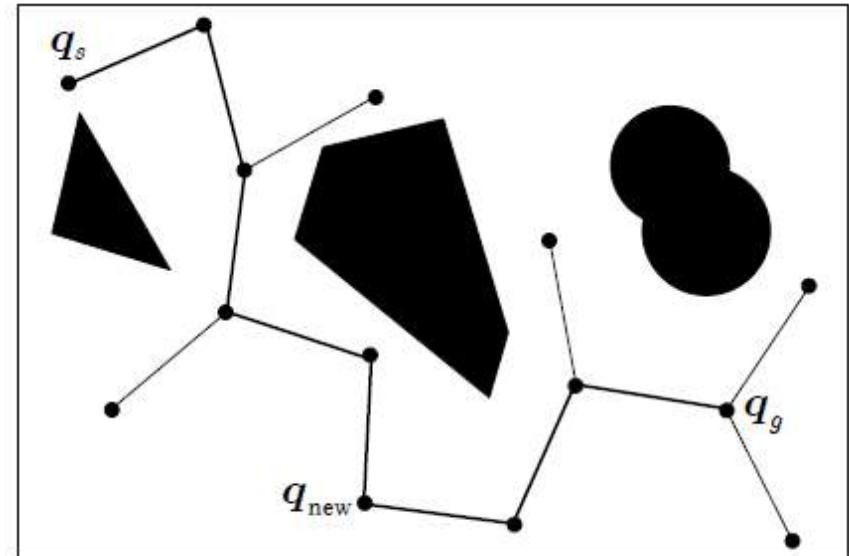
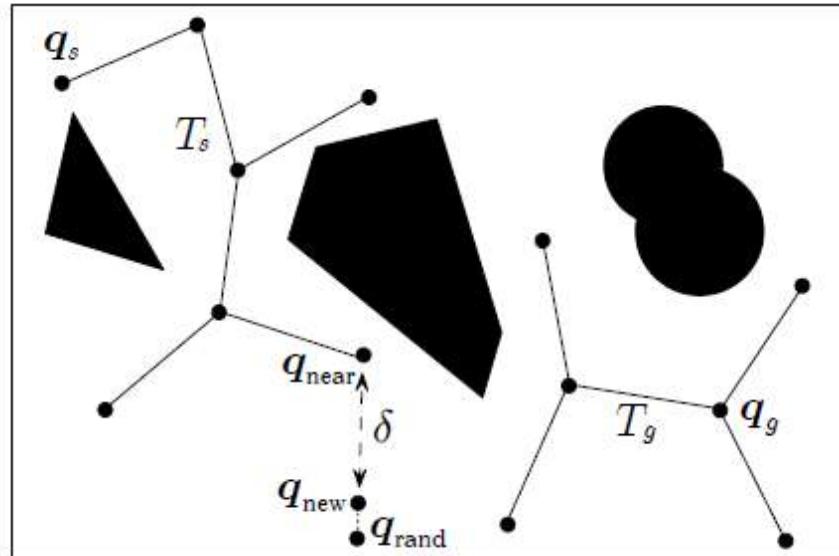
(b) The planner generates a configuration q_{rand} , and grows from the nearest node towards it to create q_{new} .



(c) The tree rapidly explores the free space.



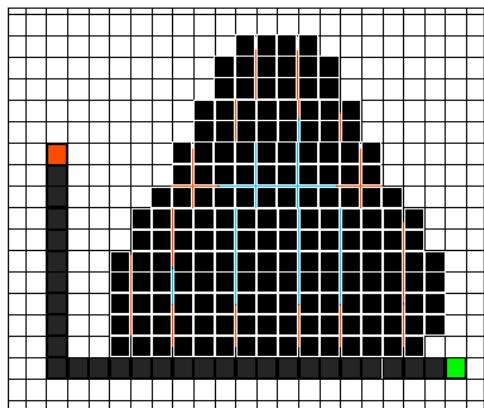
(d) The planner terminates when a node is close to the goal node. Common implementations will connect directly to the goal.



Bidirectional RRT Method

RRT: *Rapidly-exploring Random Tree*

Divergence from a plan?



- What happens if we take an action that causes us to leave the plan?
 - a) Use behaviors!
 - b) Replan
 - c) Keep a cached conditional plan
 - d) Keep a policy

Collision Avoidance

- Try to move back onto the planned trajectory (global plan), while avoiding collisions (local planning)
- **Potential field methods:** create a field (or gradient) that pushes the robot away from obstacles, and towards the goal

Potential Fields

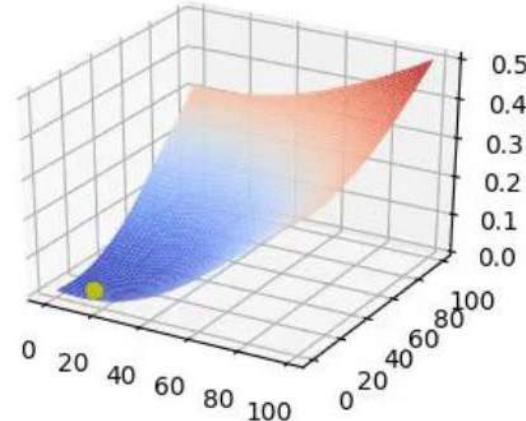
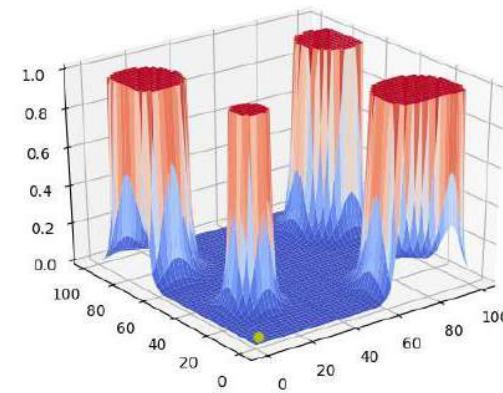
The potential of each obstacle generates a repulsive force

$$U_{rep} = \frac{1}{\|x - x_c\|}$$

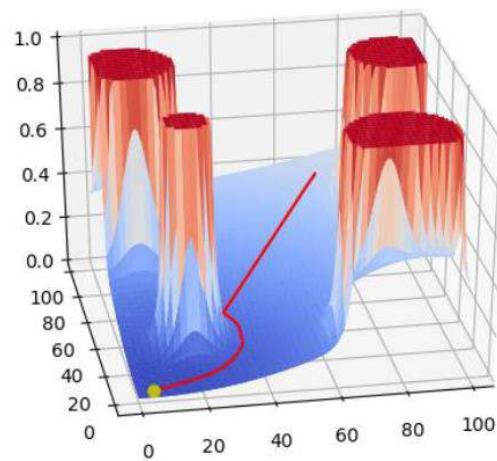
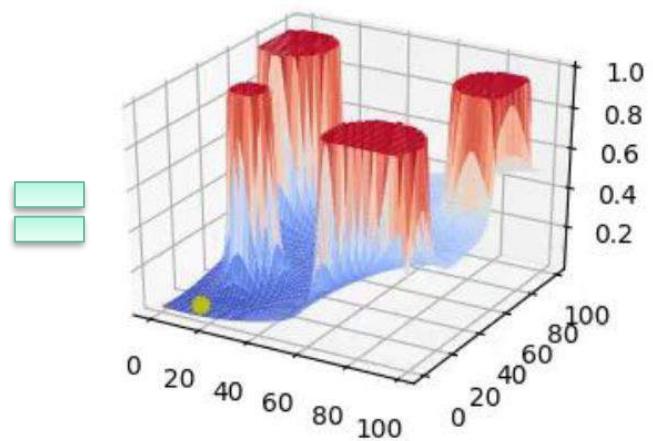
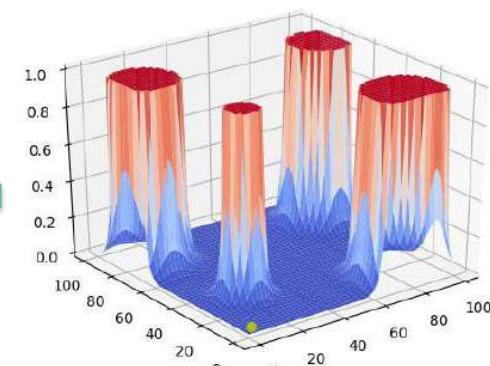
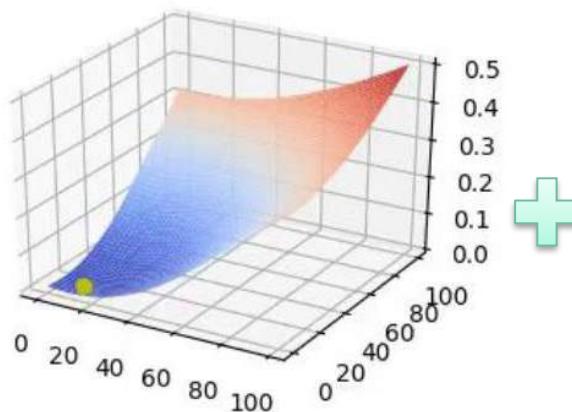
and the potential of the goal generates an attractive force

$$U_{att} = \frac{1}{2} \|x - x_{goal}\|^2$$

Easy and fast to compute
Susceptible to local minima



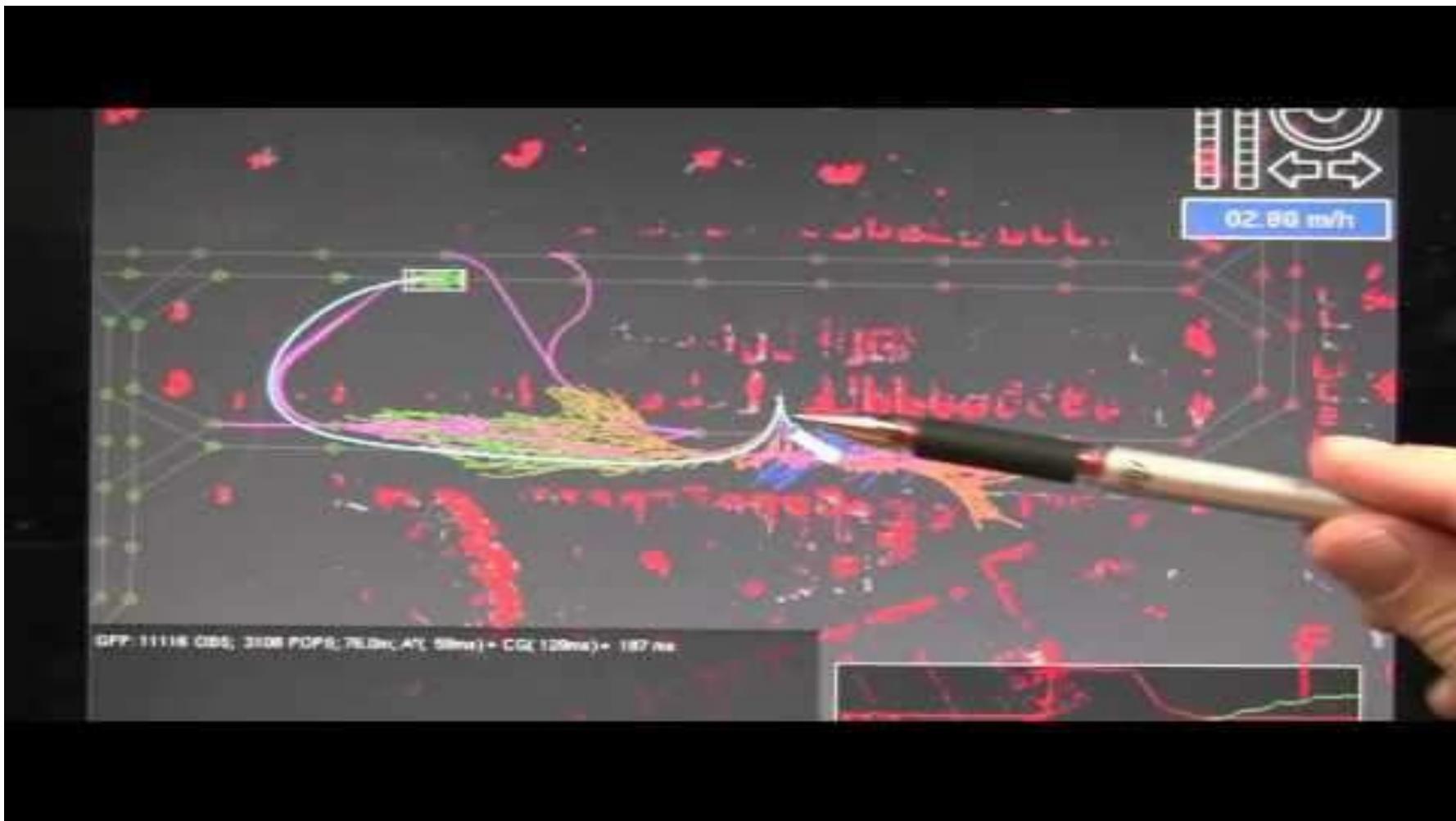
Potential Fields



Planning in Practice

- In general planning is done in a hierarchical manner
 - Global planner
 - Construct a path from initial position to the goal
 - A*, RRT, etc
 - Path smoothing is usually performed to clean up solutions
- Local planning
 - Continuously run to adapt the planned global path to changes
 - Avoids the need to compute the entire global path
- Reactive
 - For collision avoidance in case of fast dynamic objects

Planning in Practice



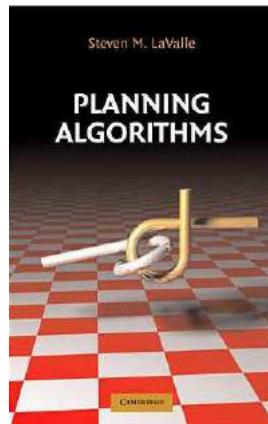
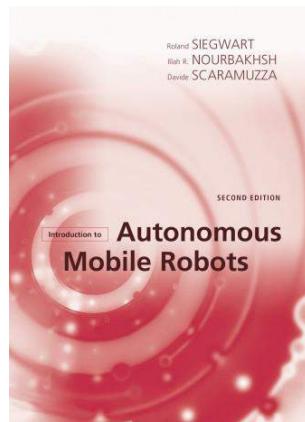
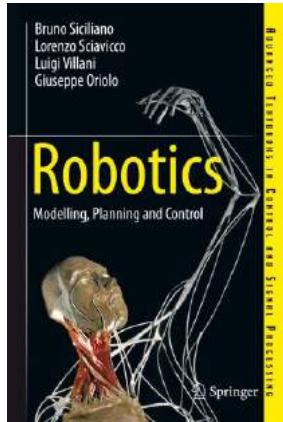
https://www.youtube.com/watch?v=zS3st_7og3A

Planning in Practice: Dynamic obstacles



http://wiki.ros.org/teb_local_planner

<https://www.youtube.com/watch?v=e1Bw6JOgHME>



Suggested reading

- LaValle, Steven M. *Planning algorithms*. Cambridge university press, 2006. (Chapter 4) <http://planning.cs.uiuc.edu/>
- Siciliano, Bruno, et al. Robotics: modelling, planning and control. Springer Science & Business Media, 2010. (Chapter 12)
- Siegwart, R. Nourbakhsh, I., and Scaramuzza, D. (2004). *Introduction to autonomous mobile robots*, (MIT Press). Ch.6.



Thank you
for listening!

Any
questions ?

https://www.youtube.com/watch?v=3_S3GPxAMYA

CMP3101M AMR – Week 9

Control Architectures

Dr. Paul Baxter

Location: INB3119

pbaxter@lincoln.ac.uk

Office hours: Tuesdays 9am-11am



MSc in Robotics and Autonomous Systems

- New (taught) MSc starting this October, specialises in Robotics, with plenty of hands-on applications
- See: <http://www.lincoln.ac.uk/home/course/ROBASYMS/>
- Robotics, Programming, AI, Applications, etc...
- Also see <https://lcas.lincoln.ac.uk/wp/study-with-us-towards-an-msc-in-robotics-and-autonomous-systems/>



My background

- Developmental Cognitive Robotics
- Human-Robot Interaction
- Social Robots for Children
- Applications:
 - Educational Robotics
 - Robots for Therapy
 - Robots in Healthcare



Syllabus review

Week	Topic	Lecturer
1	Introduction	Marc Hanheide
2	Robot Programming (ROS)	Marc Hanheide
3	Robot Sensing	Marc Hanheide
4	Motion & Control	Marc Hanheide
5	Robot Behaviour	Ayse Kucukyilmaz
6	Navigation 1	Ayse Kucukyilmaz
7	Navigation 2	Ayse Kucukyilmaz
8	Robot mapping & SLAM	Ayse Kucukyilmaz
9	Control Architectures	Paul Baxter
10	Human-Robot Interaction 1	Paul Baxter
11	Human-Robot Interaction 2	Paul Baxter
12	Applications	Paul Baxter

Syllabus review

Week	Topic	Lecturer
1	Introduction	Marc Hanheide
2	Robot Programming (ROS)	Marc Hanheide
3	Robot Sensing	Marc Hanheide
4	Motion & Control	Marc Hanheide
5	Robot Behaviour	Ayse Kucukyilmaz
6	Navigation 1	Ayse Kucukyilmaz
7	Navigation 2	Ayse Kucukyilmaz
8	Robot mapping & SLAM	Ayse Kucukyilmaz
9	Control Architectures	Paul Baxter
10	Human-Robot Interaction 1	Paul Baxter
11	Human-Robot Interaction 2	Paul Baxter
12	Applications	Paul Baxter

Who said robots were easy?

Source: IEEE Spectrum – full video at <https://www.youtube.com/watch?v=g0TaYhjp0fo>

Problems...

Complex Environment

Noisy / Unpredictable
Environment

Actions don't always lead
to intended consequences

Misleading sensors

Responses are too slow

Why Control Architectures?

- Seen a range of competencies in previous weeks...
 - Sensing, motion, navigation, mapping, localisation, etc
 - And how to use these in code
- The issue remaining is how to bring it all together into a single system that can operate autonomously
 - ...and reliably, in a complex world
- Using a set of organising principles for the robot control system (primarily the software)
 - Building blocks
 - Requirements and Constraints

Control Architecture Paradigms

- Robot control paradigm:

“...a philosophy or set of assumptions and/or techniques which characterize an approach to a class of problems.”

R. Murphy, 2000, p5

- Three main paradigms:

1. Deliberative
2. Reactive
3. Hybrid

- Each have advantages and disadvantages: in some cases, one approach may be more appropriate than another

- Typical means of characterising these paradigms is through the fundamental primitives: **sense**, **plan** and **act**

Sensing, Planning, Acting

SENSE

Sense the Environment

- In: Raw sensor data
- Out: The sensed information (some processing)

PLAN

Decide what to do (using some model of the world)

- In: sensory information
- Out: directives

ACT

Act on the Environment

- In: Information (sensory or directives)
- Out: actuator commands

**DELIBERATIVE
ARCHITECTURES**

**HYBRID
ARCHITECTURES**

**REACTIVE
ARCHITECTURES**

**COGNITIVE
ARCHITECTURES**

DELIBERATIVE ARCHITECTURES

Planning what action to take, assuming you have a world model, then doing this

- Symbolic AI

Emphasis on this ‘top-down’ (hierarchical) planning process

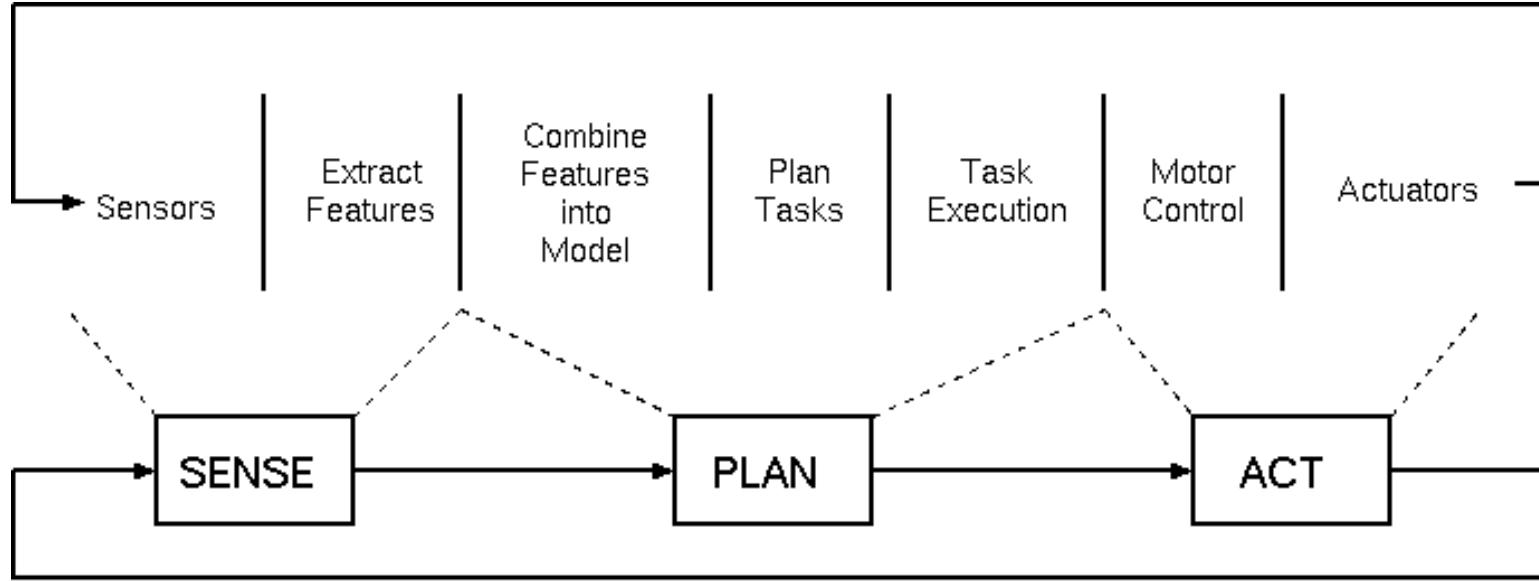
- Partly inspired by human introspection

Basic process:

1. Gather currently available information, integrate into world model
2. Plan what to do
3. Execute the plan; return to step 1



Horizontal decomposition

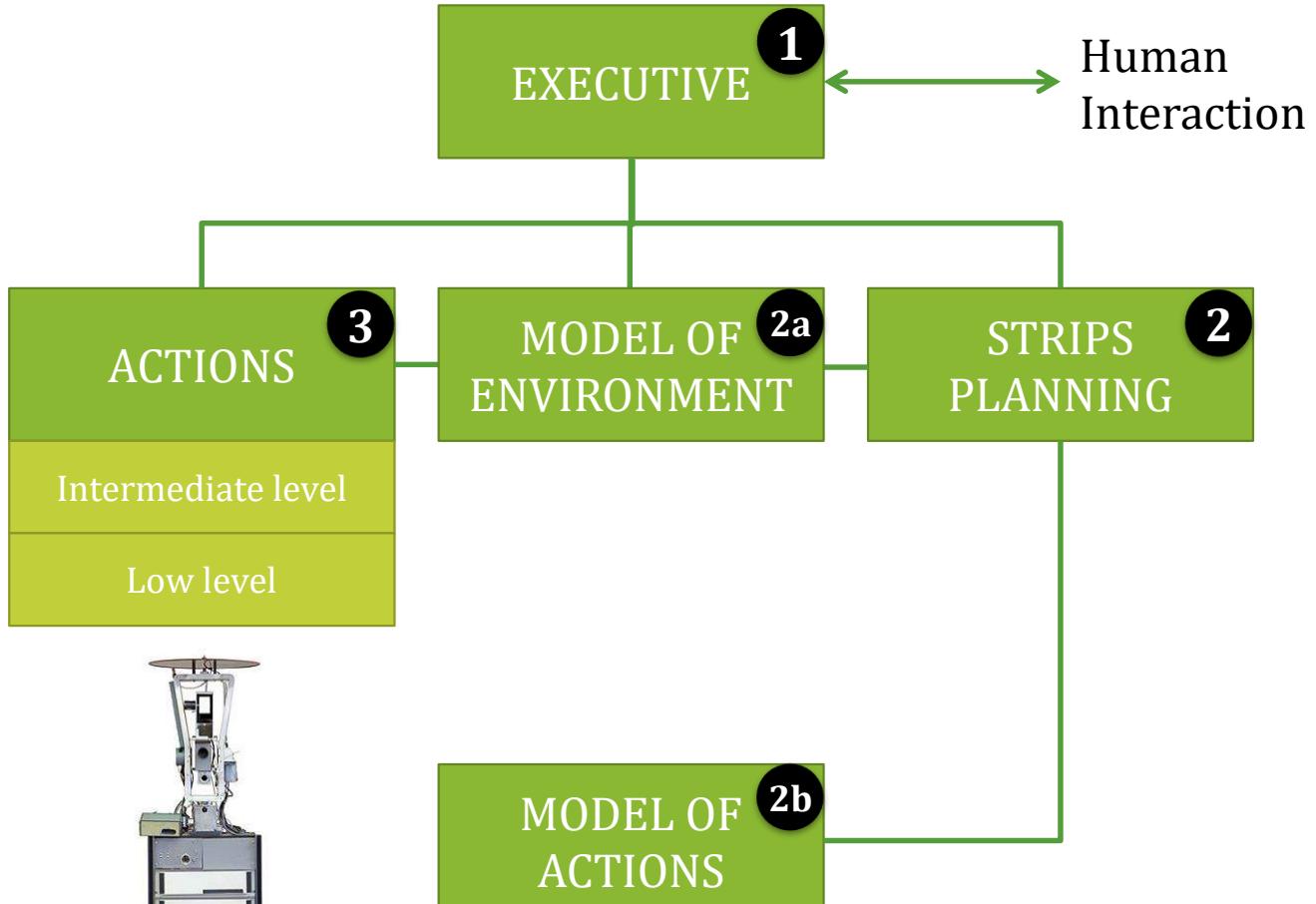


- Horizontal decomposition of tasks
 - A pipeline model: planning follows sensing, acting follows planning
- Plan first, then act out plan (open-loop)

Shakey the robot

See full (HQ) video at: <https://www.youtube.com/watch?v=7bsEN8mwUB8>

Shakey Control Architecture



Planning: STRIPS

- Stanford Research Institute Problem Solver
- Planning to accomplish a goal
 - Break down into sub-goals to reduce difference between current state and goal state
- Symbolic representation of all information
 - The world model: everything about the state of the environment
 - The capabilities/properties of the robot itself (operators)
 - Initial and goal states
 - Difference evaluator (how close to the goal state am I?)

```

Open ... / Close ...
Open door dx.
OPEN(dx)

Preconditions: NEXTTO(ROBOT, dx), TYPE(dx,DOOR), STATUS(dx,CLOSED)
Deletions:      STATUS(dx,CLOSED)
Additions:     *STATUS(dx,OPEN)

Close door dx.
CLOSE(dx)

Preconditions: NEXTTO(ROBOT,dx), TYPE(dx,DOOR), STATUS(dx,OPEN)
Deletions:      STATUS(dx,OPEN)
Additions:     *STATUS(dx,CLOSED)

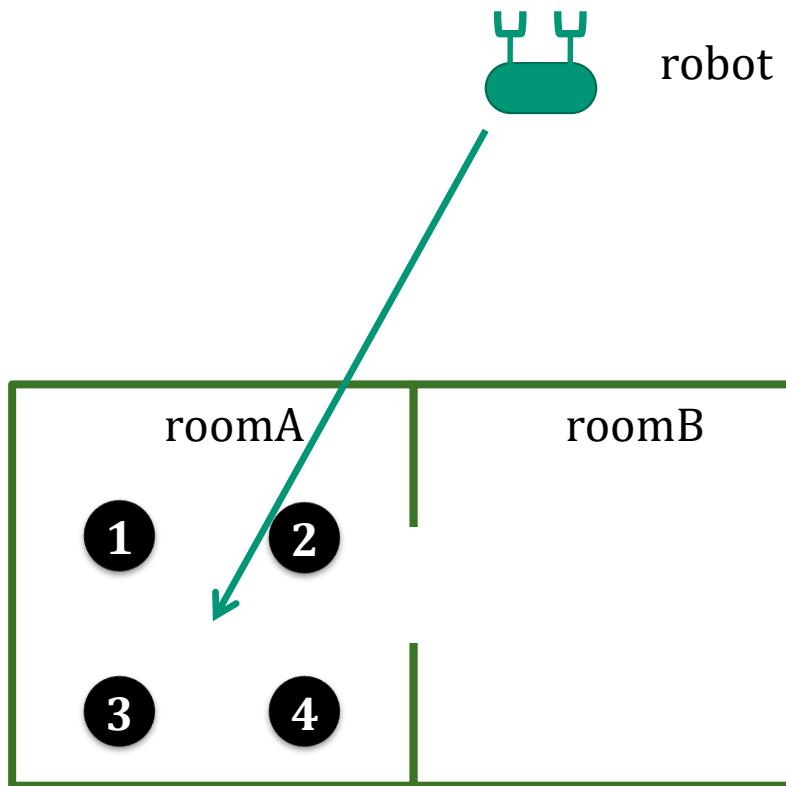
```

Standardised Planning with PDDL

- Where STRIPS is a specific planner/language, a more recent standardised planner has been created
- Planning Domain Definition Language (PDDL)
 - STRIPS plus extensions, common assumptions, benefits/shortfalls...
- See <http://lcas.lincoln.ac.uk/fast-downward/> for a planner that you can play around with
- Example from this planner: moving objects
 - Robot domain
 - Robot problem

A brief PDDL example: World

```
(define (problem strips-gripper-x-1)
  (:domain gripper-strips)
  (:objects rooma roomb ball4 ball3 ball2 ball1 left right)
  (:init (room rooma)
         (room roomb)
         (ball ball4)
         (ball ball3)
         (ball ball2)
         (ball ball1)
         (at-robbby rooma)
         (free left)
         (free right)
         (at ball4 rooma)
         (at ball3 rooma)
         (at ball2 rooma)
         (at ball1 rooma)
         (gripper left)
         (gripper right)))
```



A brief PDDL example: Robot

```


:ine (domain gripper-strips)
:predicates (room ?r)
  (ball ?b)
  (gripper ?g)
  (at-roddy ?r)
  (at ?b ?r)
  (free ?g)
  (carry ?o ?g))

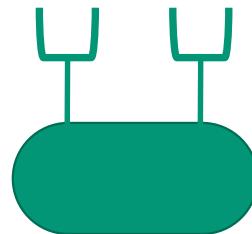
:action move
:parameters (?from ?to)
:precondition (and (room ?from) (room ?to) (at-roddy ?from))
:effect (and (at-roddy ?to)
  (not (at-roddy ?from)))))

:action pick
:parameters (?obj ?room ?gripper)
:precondition (and (ball ?obj) (room ?room) (gripper ?gripper)
  (at ?obj ?room) (at-roddy ?room) (free ?gripper))
:effect (and (carry ?obj ?gripper)
  (not (at ?obj ?room))
  (not (free ?gripper)))))

:action drop
:parameters (?obj ?room ?gripper)
:precondition (and (ball ?obj) (room ?room) (gripper ?gripper)
  (carry ?obj ?gripper) (at-roddy ?room))
:effect (and (at ?obj ?room)
  (free ?gripper)
  (not (carry ?obj ?gripper)))))



```



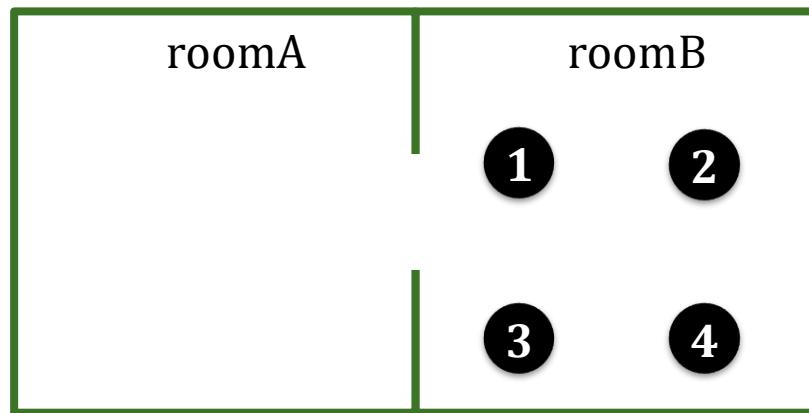
Robot can:

- Move
- Pick
- Drop

With left and
right grippers

A brief PDDL example: Goal

```
(:goal (and (at ball4 roomb)
             (at ball3 roomb)
             (at ball2 roomb)
             (at ball1 roomb))))
```

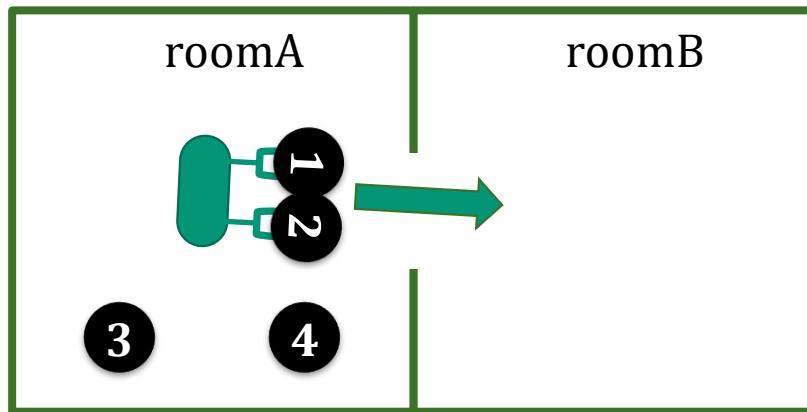


A brief PDDL example: Plan

```

(pick ball1 rooma left)
(pick ball2 rooma right)
(move rooma roomb)
(drop ball1 roomb left)
(drop ball2 roomb right)
(move roomb rooma)
(pick ball3 rooma left)
(pick ball4 rooma right)
(move rooma roomb)
(drop ball3 roomb left)
(drop ball4 roomb right)
; cost = 11 (unit cost)

```



Deliberative Architecture Limitations

- Closed World problem
 - All information is present – nothing unexpected, no unanticipated consequences, etc
- The Frame problem
 - What is and is not relevant? Should enumerate all states, even if unchanged – becomes intractable...
- Brittleness problem
 - Can't handle change not effected by the agent (wrong model?)
- Uncertainty problem
 - How should this be handled in a symbolic planner that assumes crisp knowledge, and true/false conditionals?
- Computational load
 - High load leads to slow reactivity

Direct reaction against deliberative models

Emphasis on fast reaction to low-level sensory information, without involved processing and planning

- Partly inspired by work in biology/CogSci
- Integration of sensory information not necessary

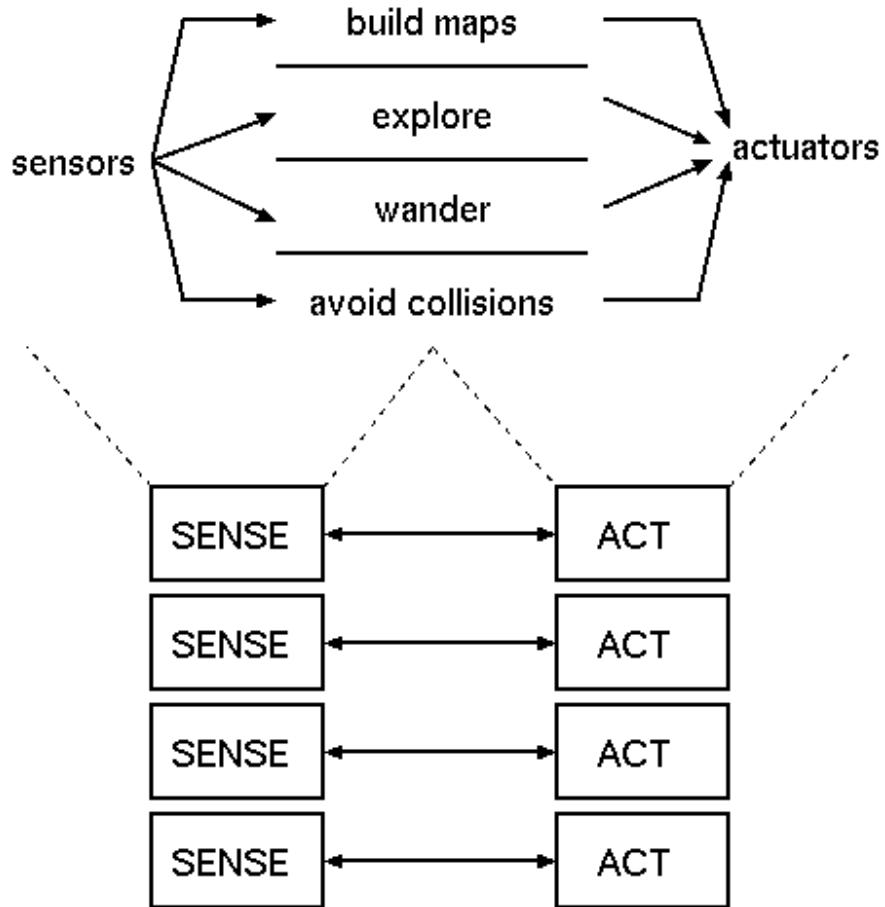
REACTIVE ARCHITECTURES

Basic process:

1. Sensory input acquired
2. Multiple parallel behaviours result in overt agent action(s)



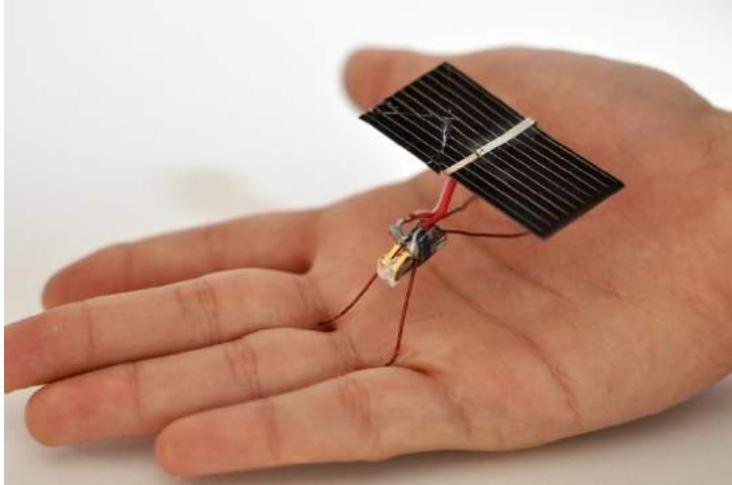
Vertical decomposition



- Contrast to the deliberative approach
- Vertical decomposition of tasks
 - Simultaneously operating pairs of sensing and acting

Benefits of Reactive Architectures

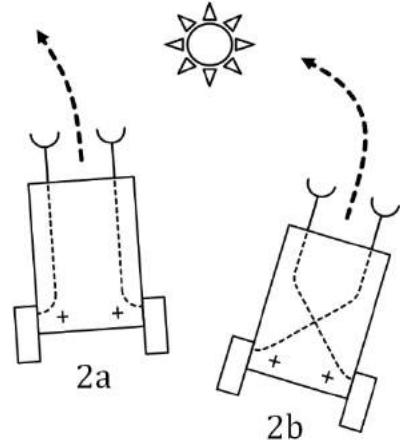
- No internal world model needed
 - “the world is its own best model”
 - Cheap and fast!
- Real-time behavioural control
- Can have emergence of complex behaviour with little design effort



<https://www.youtube.com/watch?v=k05V29ayVNU>

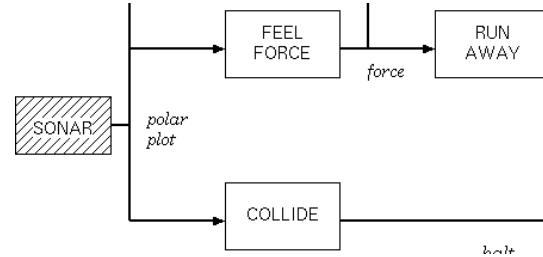
Behaviour-based Robotics

- These are typically reactive
 - Tightly coupled to sensory information (no “planning”)
 - Lacking in (or only minimal) internal state
 - Hence fast acting
- In the design of the behaviours, strong interaction with the embodiment of the robot
 - i.e. the behaviour depends on the specific architecture and body used
 - Remember the Braitenberg vehicles...
- Hence also interaction with the environment
 - Cannot necessarily fully account for behaviour just by looking at the control architecture, also need to consider the environment

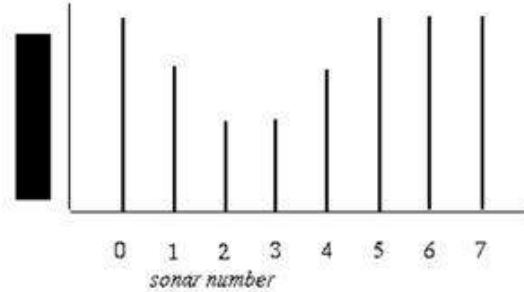
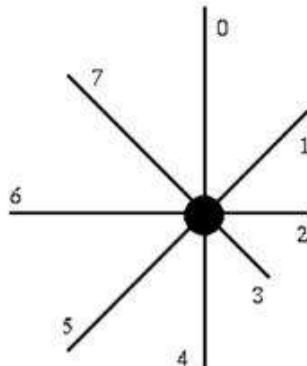


No World Model...

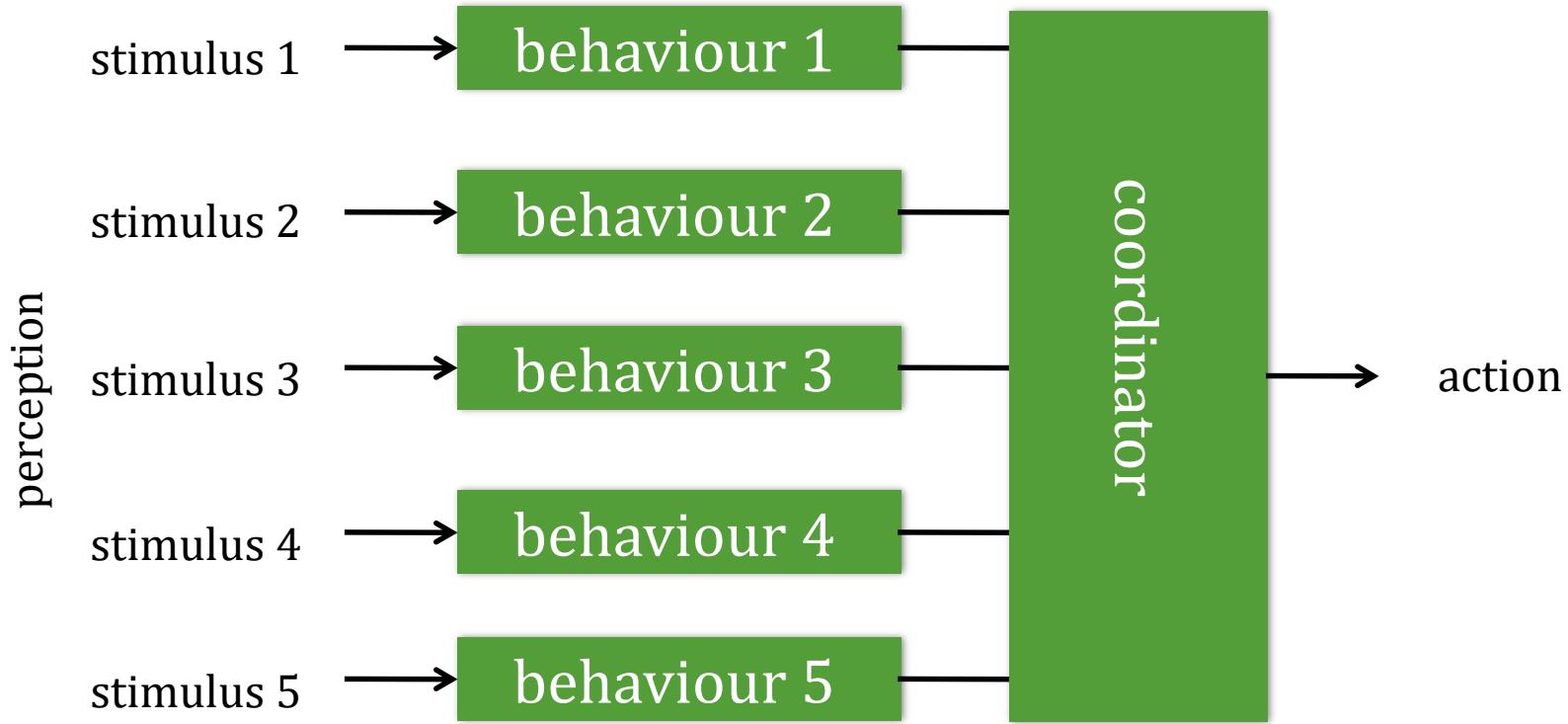
- no memory (no internal state, no model of the world)
 - Can be difficult to choose the most appropriate behaviour



- one way to deal with this limitation is to use ego-centric representations
 - Provides some structure: helps with choosing what to do



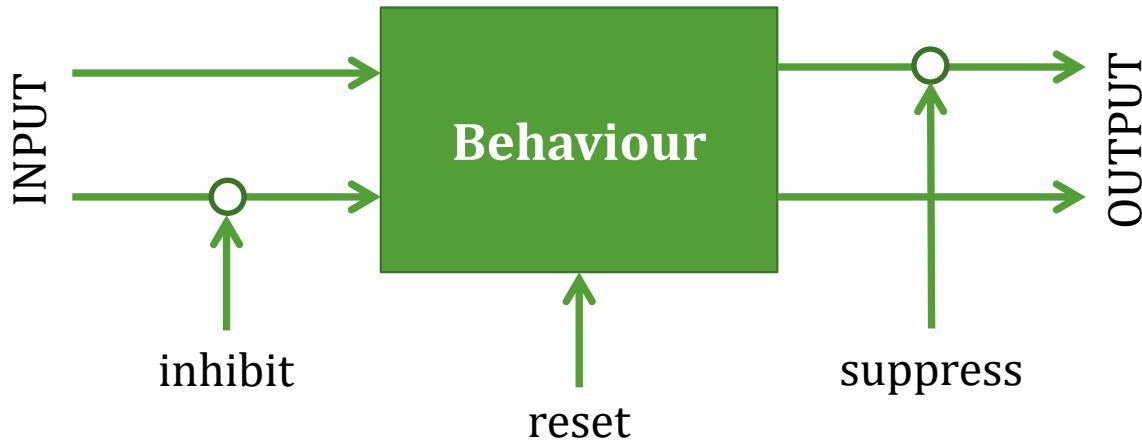
Multiple Behaviours (*recap*)



- What role for the coordinator?
 - Competitive: e.g. winner-takes-all
 - Cooperative: e.g. blending outputs through addition
 - Hybrid: e.g. activation/inhibition dynamics (Maes, 1989)

Subsumption Architecture

- A single example case of behaviour-based control architecture
 - Though the best known
 - *A design methodology*
- Gets around the coordinator problem by having higher level behaviours “subsume” lower level behaviours
 - i.e. some behaviours can over-ride others



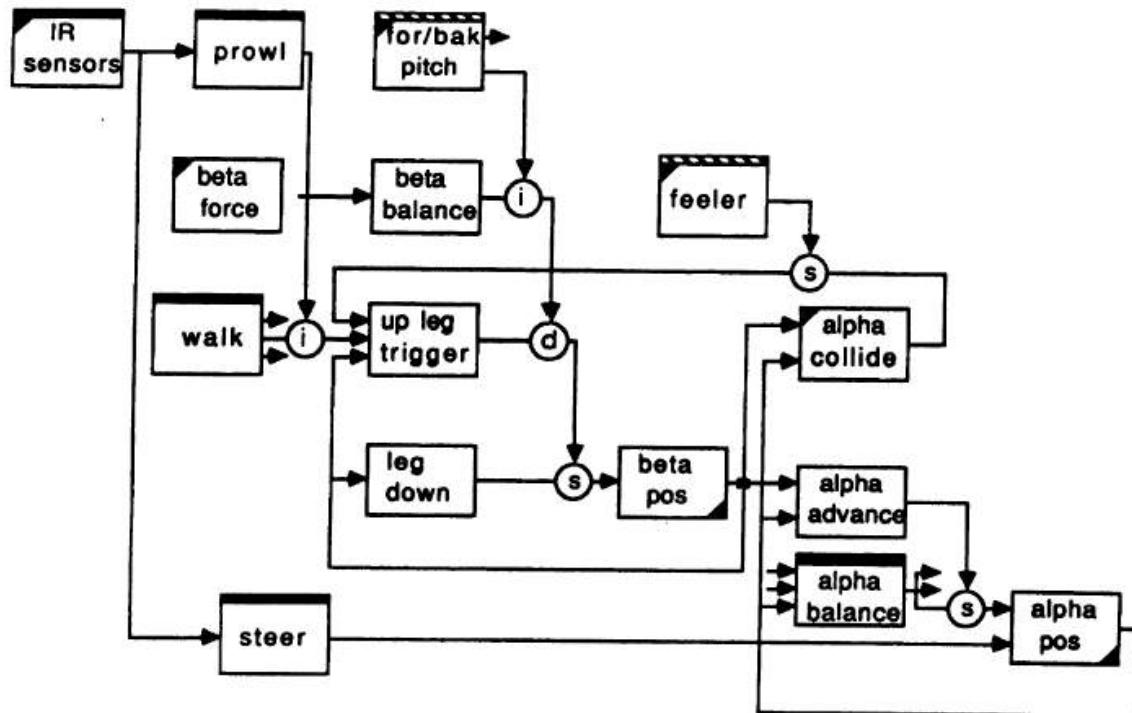
Example: Genghis



- Rodney Brooks, late 80's
 - Use of the subsumption architecture
 - (nearly) independent behaviour-based control for each leg

Example: Genghis

- Example architecture of Genghis
- Blocks:
 - Sensor input
 - Behaviours
 - ...
- Note the relative complexity of the inter-connections
- Hand-designed and tuned behaviours



Multi-agent example

Steels' cooperative Mars Exploration (1991):

<https://www.youtube.com/watch?v=H68YF9YKKJ8>

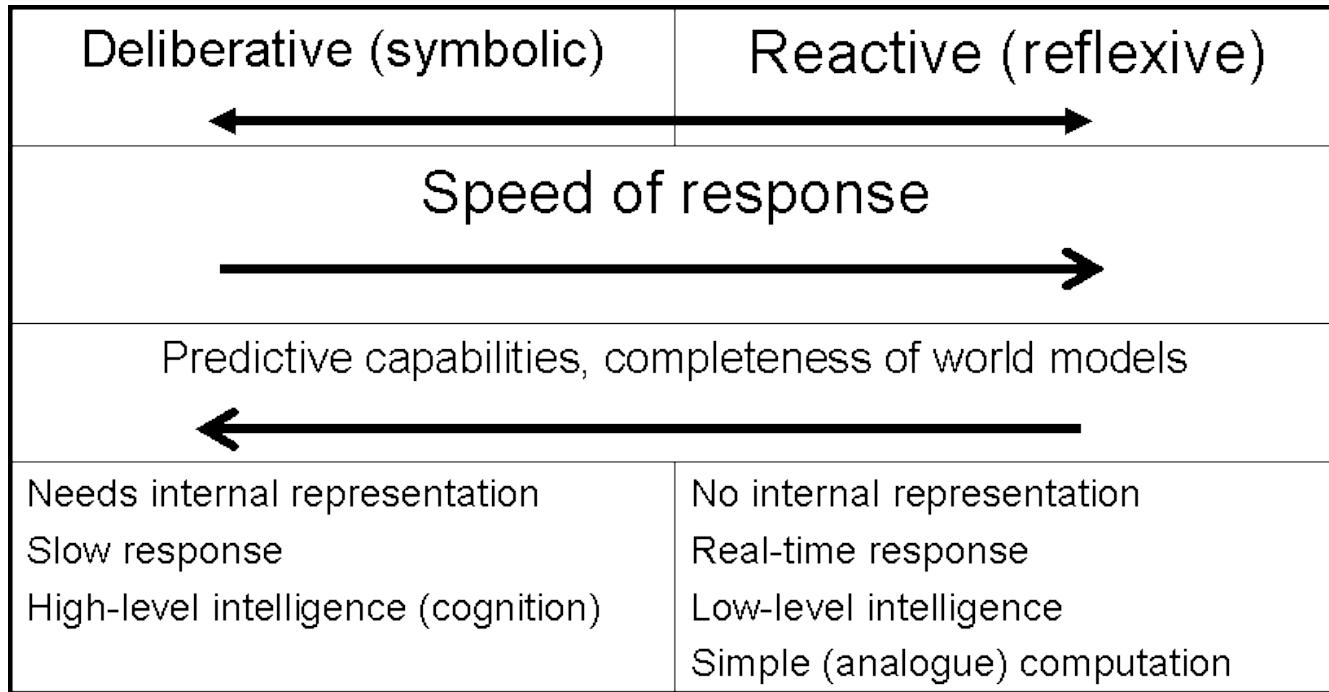
Reactive Architecture Limitations

- Oriented to specific Task (lack of generalisation)
- Based on, and constrained by, particular robot embodiment
- Sensitivity to Sensor noise
- Lack of Planning
 - Lack of internal state
 - Learning as a problem
- Stimulus-Response alone insufficient to account for intelligence
- Emergence of complex behaviour is a design problem

Assignment Advice

- See FAQ on Blackboard (announcement 15/03/19), e.g.:
 - *What if I don't find all objects/hit a wall/...*
 - *Am I allowed to use...*
 - *What should I submit...*
- Also see previous announcements on accessibility of software in other labs, as well as lab availability
 - All in announcement 14/03/19
 - Also refer to the tutorials we provided you in the first few weeks

Story so far...

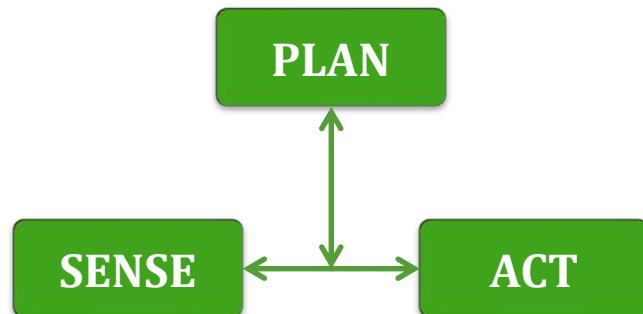


Trying to get the best of both Deliberative and Reactive paradigms

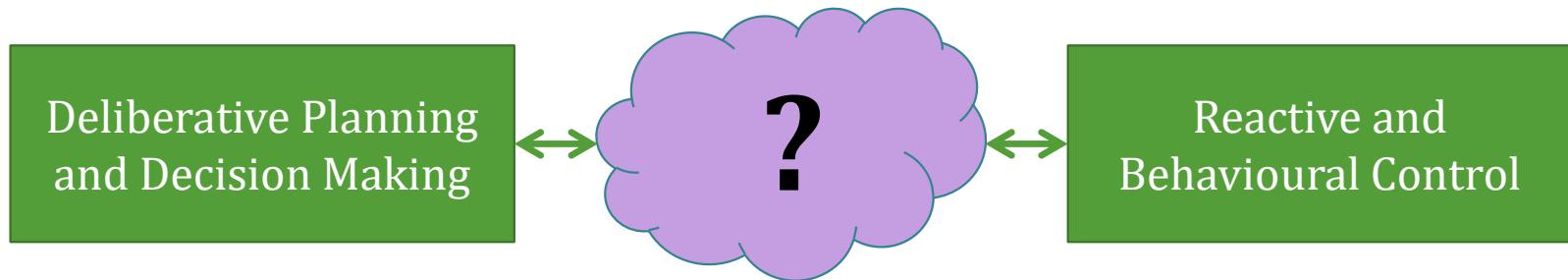
Some planning where appropriate, but maintaining ability to respond quickly to the environment

Multiple levels of control, each focused on a different aspect

HYBRID ARCHITECTURES



Linking Deliberative with Reactive

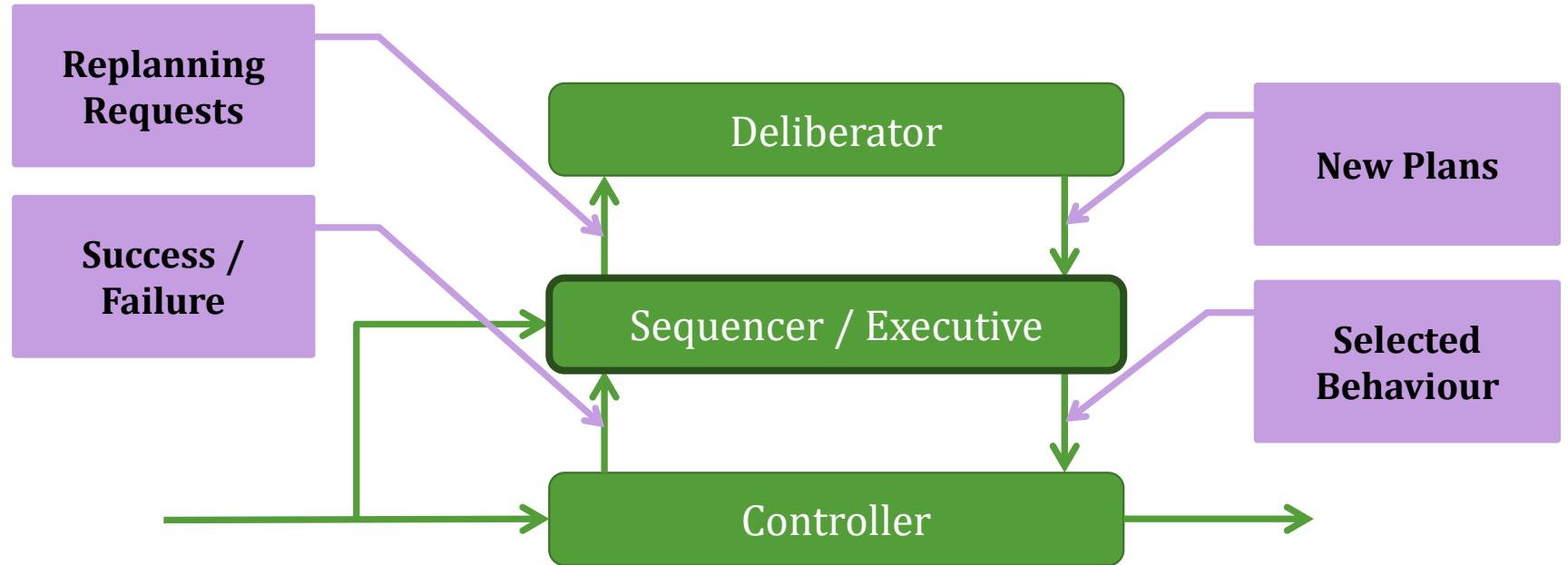


- To get best of both, use both...
 - Symbolic processing and world models/maps for planning
 - Reactive behaviours for fast, responsive action
- How best to link the two together?

Temporal decomposition

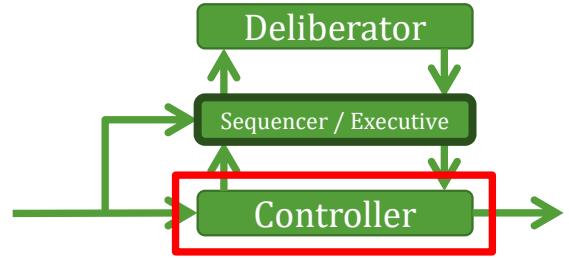
- In other architectures:
 - Horizontal decomposition of the task in Deliberative architectures
 - Can be slow...
 - Vertical decomposition of tasks in Reactive architectures
 - Can be too quick? (sensitive to noise)
- Resolve this by using time-appropriate processes:
 - Different layers with different 'speeds' of processing

Three Tier (3T) Architectures

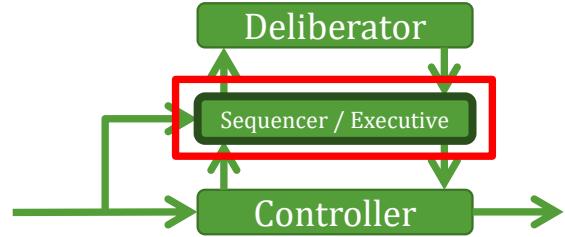


Controller

- Library of Behaviours
 - May be handcrafted
 - E.g. the behaviour-based approach
- Must be fast:
 - Avoiding internal state (except to estimate state), planning, etc
 - Stable closed-loop control
- Must be able to detect failure
 - This allows the Sequencer/Executive to call another behaviour, or call for a re-planning

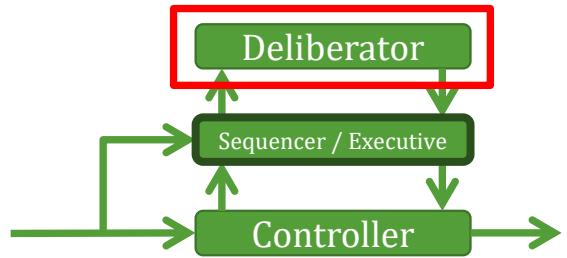


Sequencer / Executive



- This drives the control of the system
 - Not strictly speaking hierarchical, since this middle layer is doing the coordination...
- Selects which behaviour will be active
 - Sequences, loops, conditionals, threads, etc
- Initiates behaviour and planning:
 - Examines state of the world
 - Gets success/failure of controller
 - Queries deliberator if necessary
- E.g find and follow maze wall, remember sequence of turns

Deliberator



- Operates on its internal state – the world model
 - No sensing
- Time consuming and computationally intensive tasks
 - Maps and route planning
- No commitment to the means of processing here
 - Could be STRIPS-style, or anything else
- Its operation is directed (start/stop) by the Sequencer/Executive
- Example:
 - In a maze, once the location is recognised, plan a route to the exit

Performing a sample task

- I want coffee...
And I want it now...
- What do I need to take into account?

static • Where is it?

static • How to get there?

dynamic • Walk and Open doors... (physical conventions)

dynamic • Don't walk into people... (personal conventions)

dynamic • Queue... (cultural conventions)

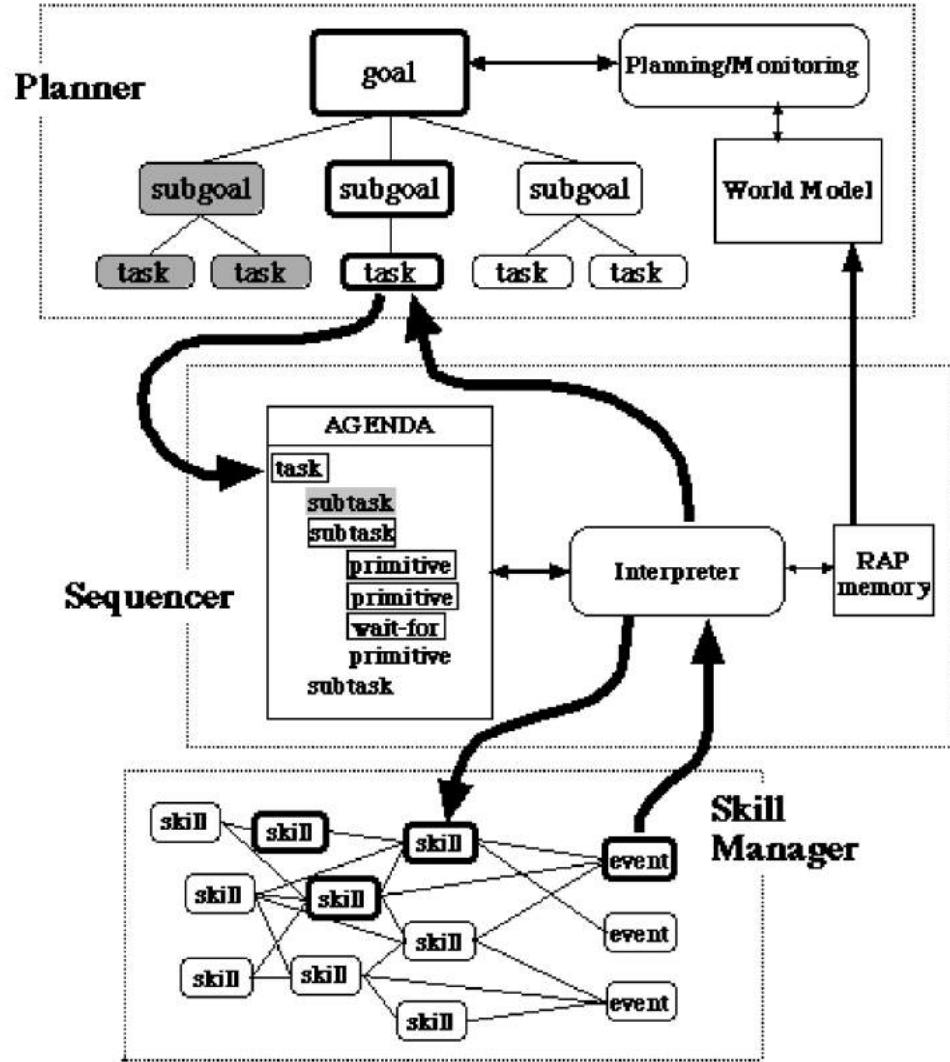
dynamic • Pay... (legal conventions)

target! • Refreshment



Both planning and reactive components required to solve the task

Example Systems: NASA



- A 3T architecture used by NASA (left)
 - Automated control of systems and devices
 - Also shared autonomy at various layers in the hierarchy
- Generally, 3T architectures are among the most prevalent
 - Hybrid approaches in general form the basis of the majority of systems in use today

From Kortenkamp et al (1998)

Example Systems: Google Car

See Waymo website: <https://waymo.com/>

Advantages

- Part of the advantage is the specification of overall structure, and principle of operation, rather than precise mechanism
 - Maintains flexibility
 - If one algorithm not appropriate, swap it out for another
- According to Erann Gat (1998):

“lines between the components of the three layer architecture can be blurred to accommodate reality”

 - Flexibility depending on the application

“If, as seems likely, there is no One True Architecture, and intelligence relies on a hodgepodge of techniques, then the three layer architecture offers itself as a way to help organise the mess”

 - Guiding principle rather than prescriptive on mechanism

Hybrid Architecture Issues

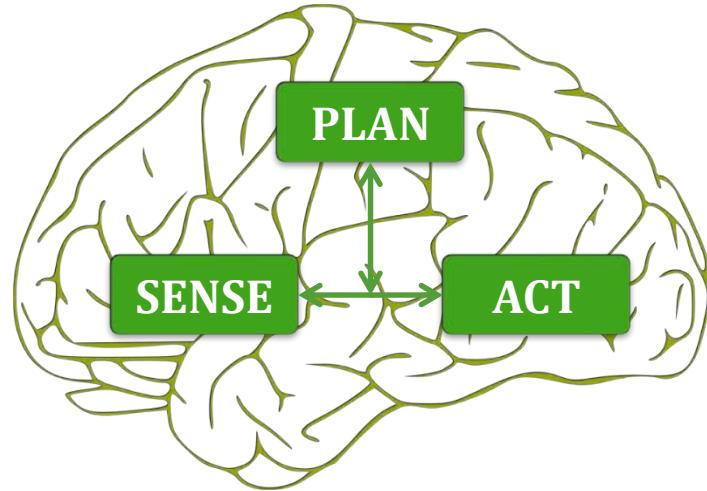
- The central role of the Sequencer / Executive
 - Clearly of central importance in the 3T approach
 - Thus needs particular attention
- “Symbol grounding”
 - Relevance of the representations (e.g. symbolic) to the instantiation/actions of the robot system it is planning for
 - Circumvented by appropriate design

Explicitly taking into account the way that humans may process information and act

- Not introspection, but evidence...

Broad category, including inspiration from psychology, CogSci, neuroscience, etc

Overlaps with the previous paradigms, particularly hybrid



COGNITIVE ARCHITECTURES

What is a Cognitive Architecture?

Cognitive Architecture...

“...is the overall, essential structure and process of a domain-generic computational cognitive model, used for a broad, multiple-level, multiple-domain analysis of cognition and behavior..”

(Sun, 2004)

Unpacking the definition

- Cognition and Behaviour
 - Both an account of the internal workings, and also how this generates overt behaviour
- Multiple-level
 - Macro and micro aspects, over multiple time-scales
 - Similar in this sense to hybrid architectures
- Domain-general
 - Not restricted to a specific task, but general modes of operation applicable across domains
 - Compare/contrast with deliberative/reactive architectures
- Structure and Process
 - The mechanisms (and knowledge) required to achieve this

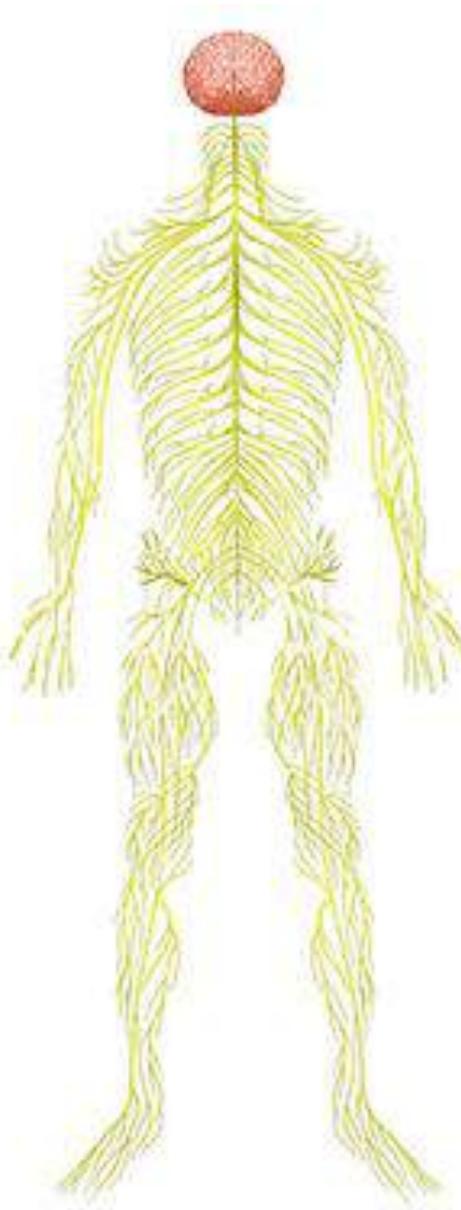
Why use Cognitive Architectures?

Prof. Ashok Goel

Udacity/Georgia Tech: <https://www.youtube.com/watch?v=bjEyYKaXUvw>

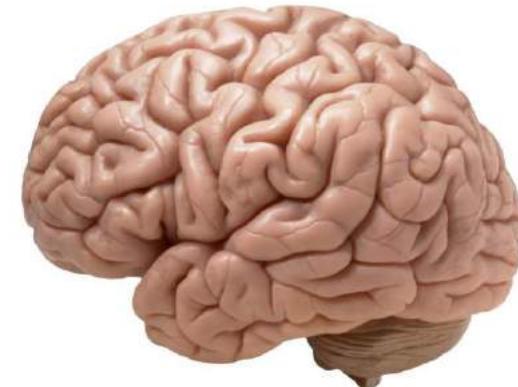
Levels of control

- Brain
 - Cognitive processing
 - Memory, etc
- Joint between the two...
 - Embodied cognition (influence of the body)
 - Influence of drives (e.g. hunger)
 - Sensory information
- Body
 - Reflexes
 - Reactions
 - Independent of brain...



Why take inspiration from humans?

- Humans demonstrate the best example of highly complex intelligence, and so could form useful design guides
 - To solve the difficult problem of general-purpose intelligence, start somewhere...
- If robots are to interact with humans in human environments, then having them endowed with some human-like cognitive features could be useful
 - To understand humans and their behaviour, to facilitate interaction
- We will return to this in the coming weeks...



Two main approaches (and others besides...)

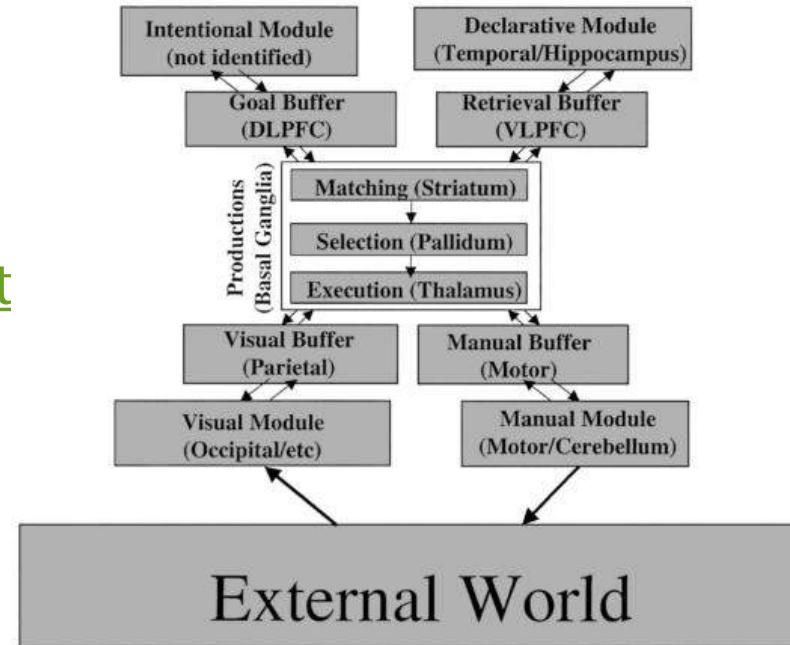
1. Derive set of mechanisms to use from human behavioural or other data
 - A “top-down” perspective
 - Can use this as a model of human behaviour
 - Typically based on data from psychology (overt behaviour)
2. Try to model fundamental principles of organisation of cognition, and implement these
 - A “bottom-up” perspective
 - Try to match to certain aspects of (human) behaviour
 - Typically derived from data closer to biology

Note...

There are many, many cognitive architectures, these are only two examples...

Example of Top-down: ACT-R

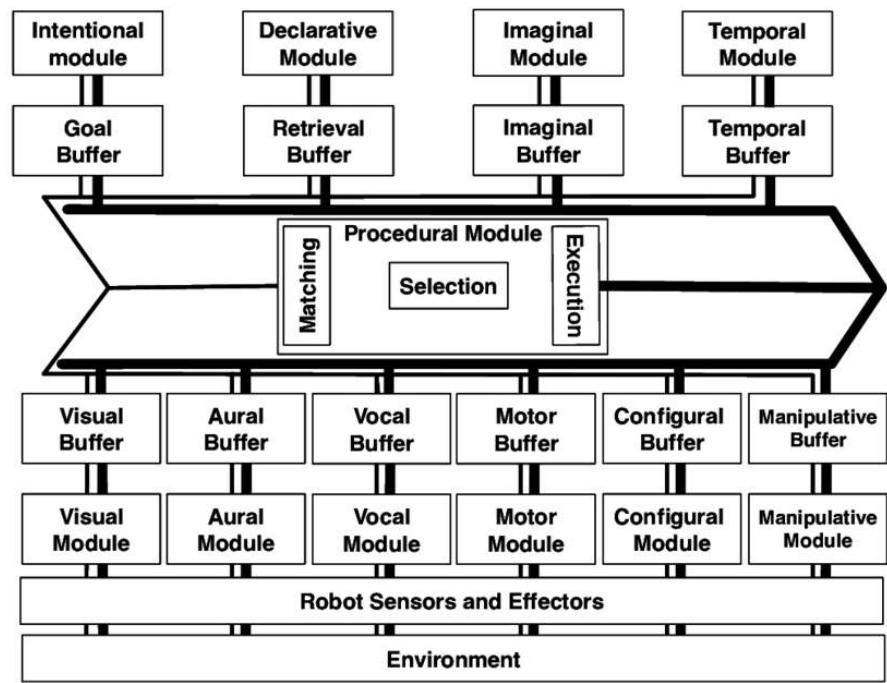
- ACT-R is an established cognitive architecture that has been applied to numerous models of human behaviour (reaction times, cognitive load, etc)
- Based on evidence from psychology, physiology, etc
- Hybrid Symbolic/
Sub-symbolic processing
- <http://act-r.psy.cmu.edu/about>



Example of Top-down: ACT-R

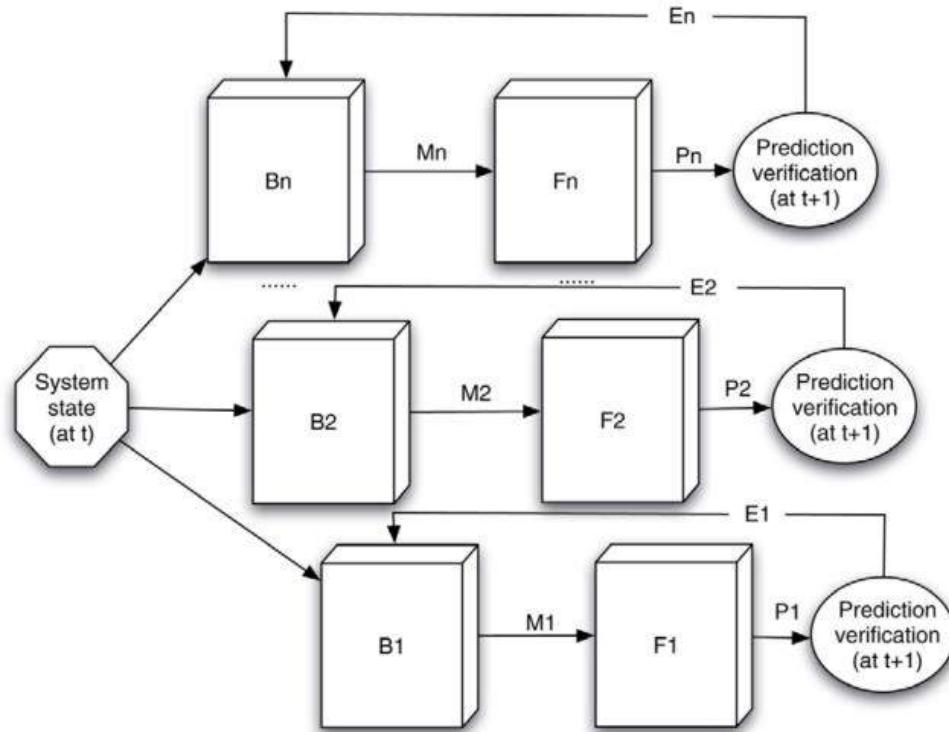


Images taken from (Trafton et al, 2013)



Example of Bottom-up: HAMMER

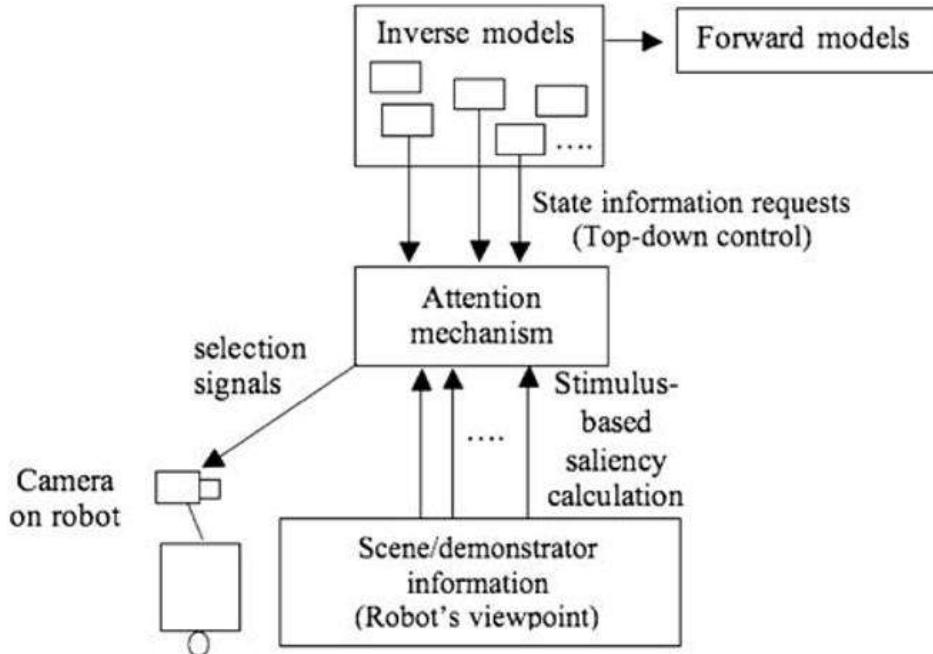
- Hierarchical, Attentive, Multiple Models for Execution and Recognition (HAMMER)
- Fundamentally based on Forward /Inverse model couplings, and applied to imitation, learning, assistance, etc
 - Strong theoretical foundations in psychology, neuroscience...
 - Comparing the different applications in a principled manner



Example of Bottom-up: HAMMER



Cooperative
wheelchair control

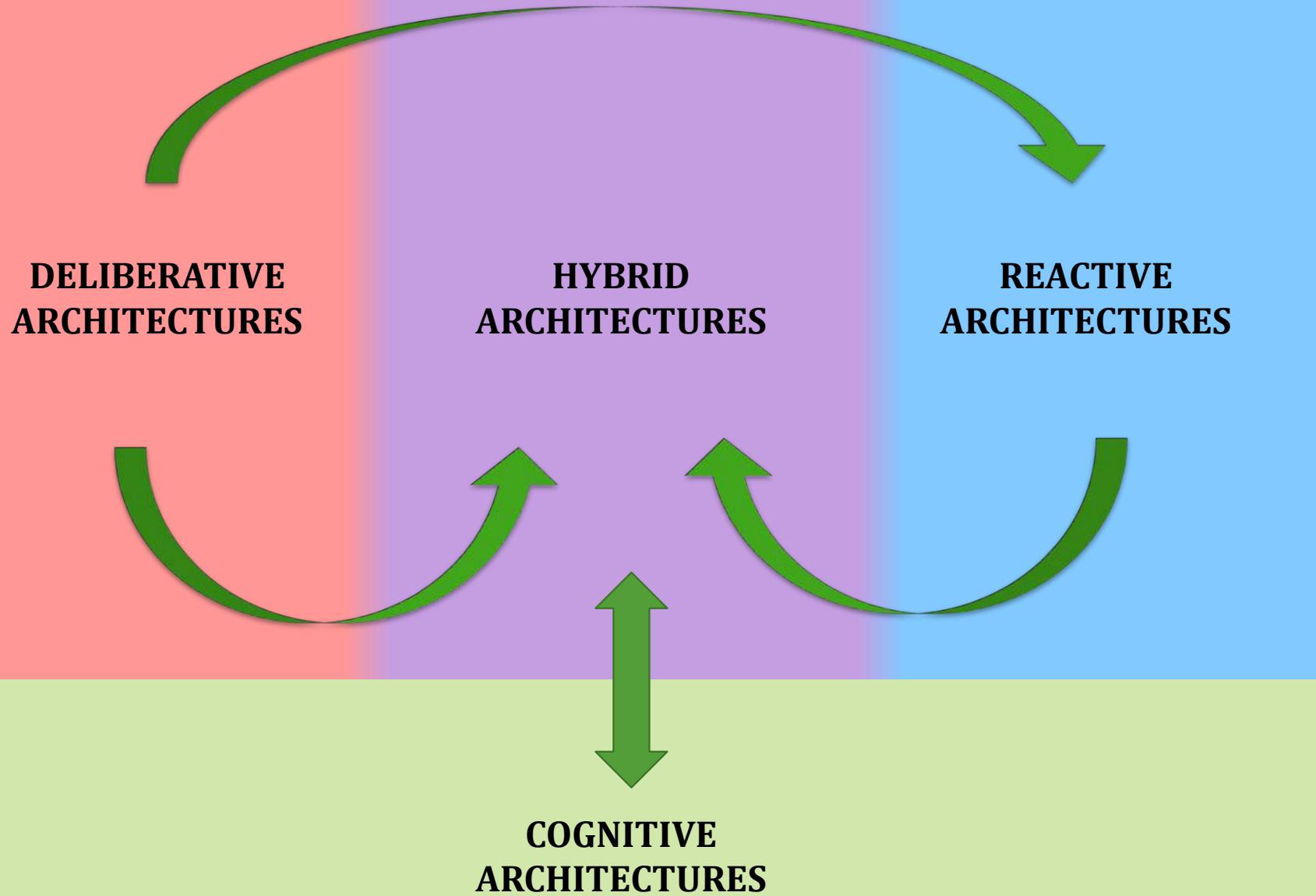


Prediction
of
intentions

See https://www.youtube.com/watch?v=CaH82_WzAeQ for a lecture by Prof. Yiannis Demiris on this, and other, work

Cognitive Architecture Issues

- How related to biology should it be?
 - Inspiration or constraints?
- Suitable level of abstraction?
 - Behaviour or mechanism?
 - E.g. ANN from week 5
- What is the purpose of using a Cognitive Architecture?
 - Functional or explanatory?



Tip of the Iceberg...

- Many individual examples of systems, each with variations, for each of the types of architecture mentioned
 - Have focused on control of individual robots
- Some control architectures/methods not detailed:
 - Multi-Agent Systems
 - Coordinating multiple robots
 - Emergent behaviour from swarms of robots
 - Inter-Agent Communication/Synchronisation
 - Social interaction, etc
 - Though something related next week...
 - Etc...

E.g. <https://www.youtube.com/watch?v=G1t4M2XnIhI>

Workshop this week

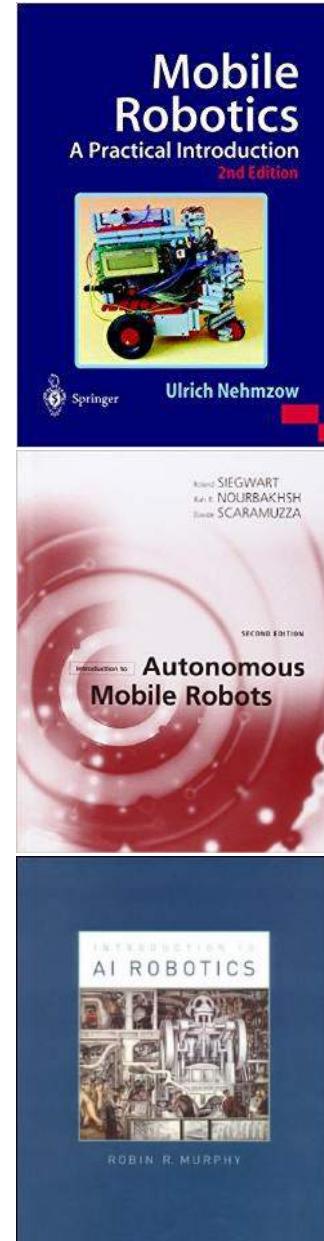
- There will be workshops as normal this week (Friday and next Monday)
 - Timetabled and with registers as normal
- Task this week will further explore some of the concepts from today
 - Part practical using the simulation environment from the assignment
 - Part theoretical where you try to map some of the concepts covered to today to the (simulated) robot

Assignment Marking

- Will start in workshops next Friday (29th March)
 - This will continue for the following workshops
 - You will be assigned to one of the two marking weeks (in your normal scheduled workshop)
- Allocations of individuals are on Blackboard
 - See announcement 08/02/19
 - This may be subject to change until just after the assignment deadline
- No need to attend the workshop you have not been assigned to
 - Do not worry about attendance monitoring for this, we will make the necessary arrangements

References / Reading

- Brooks, R.A., 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1), pp.14–23.
- Brooks, R.A., 1990. Elephants don't play chess. *Robotics and Autonomous Systems*, 6, pp.3–15.
- Gat, E., 1998. "On Three-Layer Architectures." Artificial Intelligence and Mobile Robotics, in D. Kortenkamp, R. P. Bonnasso and R. Murphy (eds.), AAAI Press, pp.195-210.
- Kortenkamp, D. et al, 1998. Three NASA application domains for integrated planning, scheduling and execution. *AAAI AIPS Workshop*, pp.88-93
- Maes, P., 1989. How to do the right thing. *Connection Science*, 1(3), pp.291–232.
- Murphy, R., 2000. *Introduction to AI Robotics*, MIT Press.
- Sun, R., 2004. Desiderata for cognitive architectures. *Philosophical Psychology*, 17(3), pp.341–373.



Syllabus review

Week	Topic	Lecturer
1	Introduction	Marc Hanheide
2	Robot Programming (ROS)	Marc Hanheide
3	Robot Sensing	Marc Hanheide
4	Motion & Control	Marc Hanheide
5	Robot Behaviour	Ayse Kucukyilmaz
6	Navigation 1	Ayse Kucukyilmaz
7	Navigation 2	Ayse Kucukyilmaz
8	Robot mapping & SLAM	Ayse Kucukyilmaz
9	Control Architectures	Paul Baxter
10	Human-Robot Interaction 1	Paul Baxter
11	Human-Robot Interaction 2	Paul Baxter
12	Applications	Paul Baxter

CMP3101M AMR – Week 10

Human-Robot Interaction

part 1

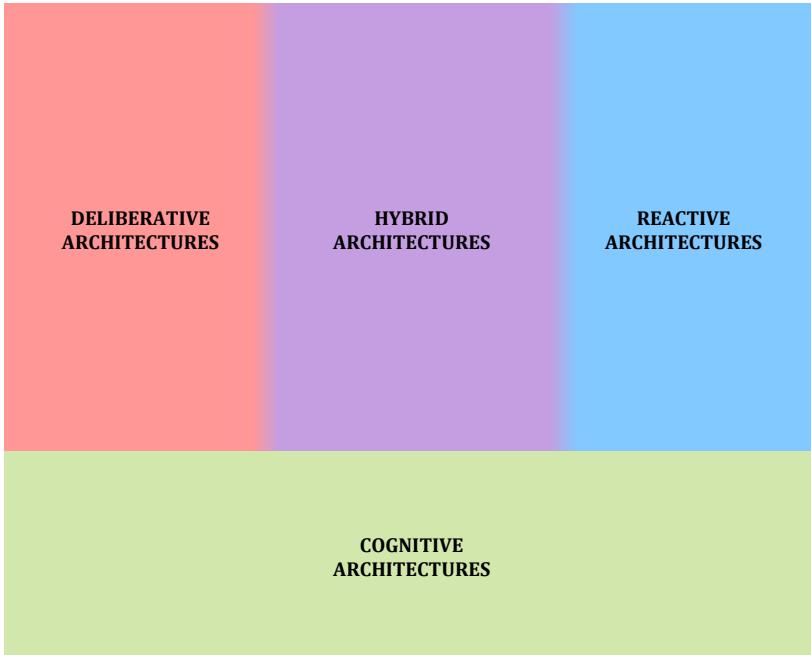
Dr. Paul Baxter

Location: INB3119

pbaxter@lincoln.ac.uk

Office hours: Tuesdays 9am-11am

Last Week...



- Control Architectures for Autonomy
- Why they are necessary
- Three (+1) main paradigms

Last Workshop...

- Exploring Control Architectures in more detail
 - Subsumption architectures (*reactive*): paper for background reading on Blackboard
 - A PDDL implementation (*deliberative*): exploring the example we went through in the lecture
 - How the two can be combined: advantages/disadvantages...
- Making use of the turtlebot simulator to implement an instance of the subsumption architecture
- *In support of exam preparation*

Marking Sessions

- Start this Friday 29th and continue in timetabled workshop sessions until Monday 8th April
- **See Blackboard (and email) announcement on 20th March for details**
- Reminder:
 - You must attend the time slot you have been allocated to
 - 5 minutes to run simulation (continuous, may restart within this): you will start this
 - Deliver presentation – may be useful to do in parallel with the simulation
 - We will ask questions/explore your code and reasoning
 - We will have all submitted code/presentations ready
 - Shared register: only attend the session you are assigned to

Syllabus review

Week	Topic	Lecturer
1	Introduction	Marc Hanheide
2	Robot Programming (ROS)	Marc Hanheide
3	Robot Sensing	Marc Hanheide
4	Motion & Control	Marc Hanheide
5	Robot Behaviour	Ayse Kucukyilmaz
6	Navigation 1	Ayse Kucukyilmaz
7	Navigation 2	Ayse Kucukyilmaz
8	Robot mapping & SLAM	Ayse Kucukyilmaz
9	Control Architectures	Paul Baxter
10	Human-Robot Interaction 1	Paul Baxter
11	Human-Robot Interaction 2	Paul Baxter
12	Applications	Paul Baxter

Syllabus review

Week	Topic	Lecturer
1	Introduction	Marc Hanheide
2	Robot Programming (ROS)	Marc Hanheide
3	Robot Sensing	Marc Hanheide
4	Motion & Control	Marc Hanheide
5	Robot Behaviour	Ayse Kucukyilmaz
6	Navigation 1	Ayse Kucukyilmaz
7	Navigation 2	Ayse Kucukyilmaz
8	Robot mapping & SLAM	Ayse Kucukyilmaz
9	Control Architectures	Paul Baxter
10	Human-Robot Interaction 1	Paul Baxter
11	Human-Robot Interaction 2	Paul Baxter
12	Applications	Paul Baxter

What is HRI?

Human-Robot Interaction (HRI) is:

... a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans. Interaction, by definition, requires communication between robots and humans.

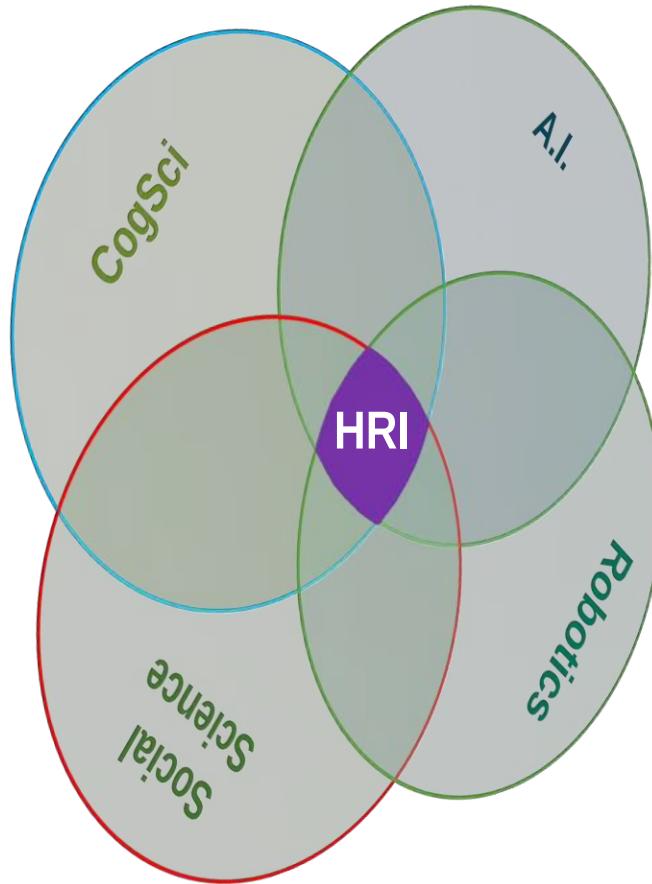
<http://humanrobotinteraction.org/1-introduction/>

Some overlaps with Human-Computer Interaction, so refresh your memory of last year's HCI module...

- In fact, origins of HRI lie in HCI
- Particularly in terms of development and evaluation methodologies

Aspects of HRI

- Psychology
 - Human Factors
-
- Sociology / Social Science
 - Human-Computer Interaction
-
- Computer Science
 - Mechatronics
-
- ...



From (Baxter et al, 2016)

Humans and Robots

Humans

- Interaction partner
- Social agent <----->
- Target of research <----->
- Source of knowledge
- Recipient of help
- Caregiver
- Companion
- ...

Robots

- Interaction partner
- Social agent?
- Target of development
- Source of knowledge
- Recipient of help
- Caregiver
- Companion
- ...



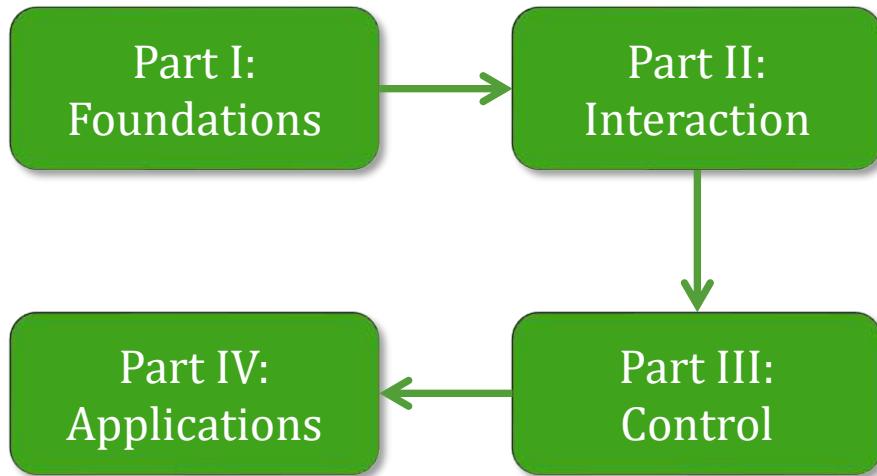
Why HRI in AMR?

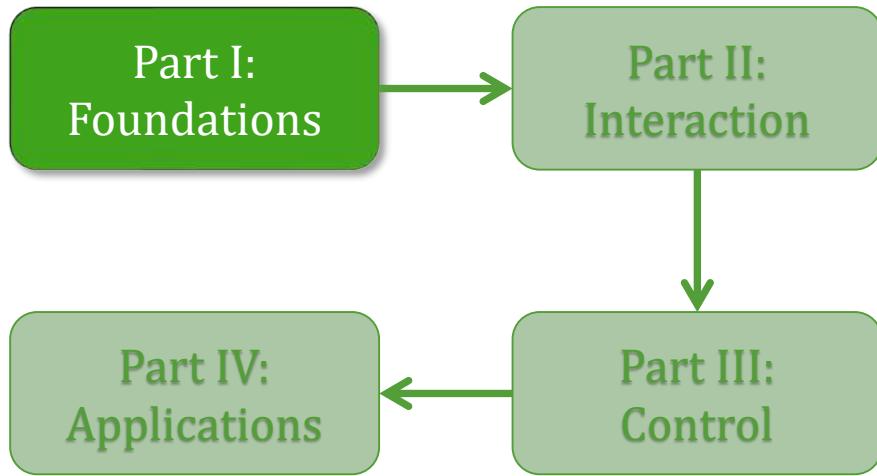
- Purpose of robotics:
 - Automation: doing (boring/repetitive) jobs for people
 - Replacing people in risky/dangerous situations
 - ...
- Could also use for helping people:
 - Physical rehabilitation (exoskeletons, haptics, etc)
 - Social therapy, etc (e.g. socially-assistive robotics - SAR)
 - ...
- And as a tool for understanding people...?
 - The psychology/social science perspective

The interaction of humans and robots is pervasive in robotics

-> hence HRI...

Today...



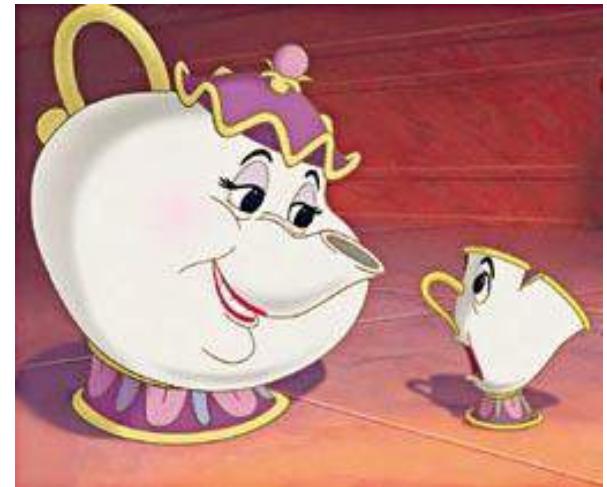


Part I: Foundations

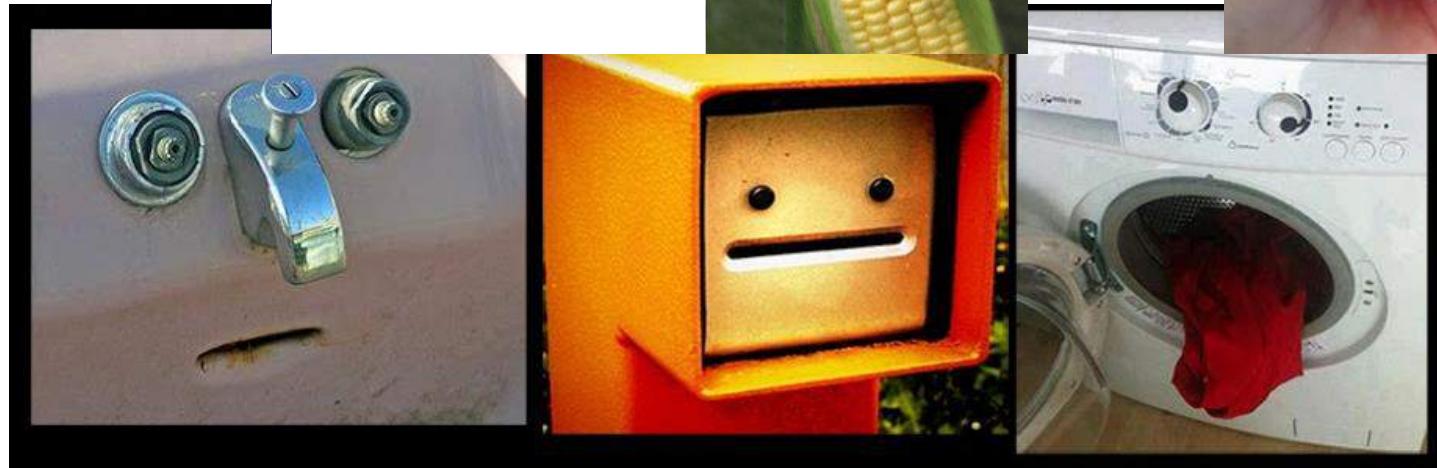
Assumptions and characteristics

Anthropomorphism

- Is the tendency to attribute human characteristics to inanimate objects, animals and others with a view to helping us rationalise their actions
 - (Duffy, 2003)
- Many examples in cartoons (Disney being particularly prolific)
- “the strategy of interpreting the behaviour of an entity (person, animal, artefact, whatever) by treating it as if it were a rational agent who governed its ‘choice’ of ‘action’ by a ‘consideration’ of its ‘beliefs’ and ‘desires’”
 - (Dennet, 1996): the intentional stance
- Embracing this concept in HRI
 - Taking advantage of it rather than trying to avoid it
 - ***Appearance*** and ***Behaviour***
- Related concepts: active perception, and gestalt psychology from HCI – wanting to find and group information in ‘meaningful’ ways

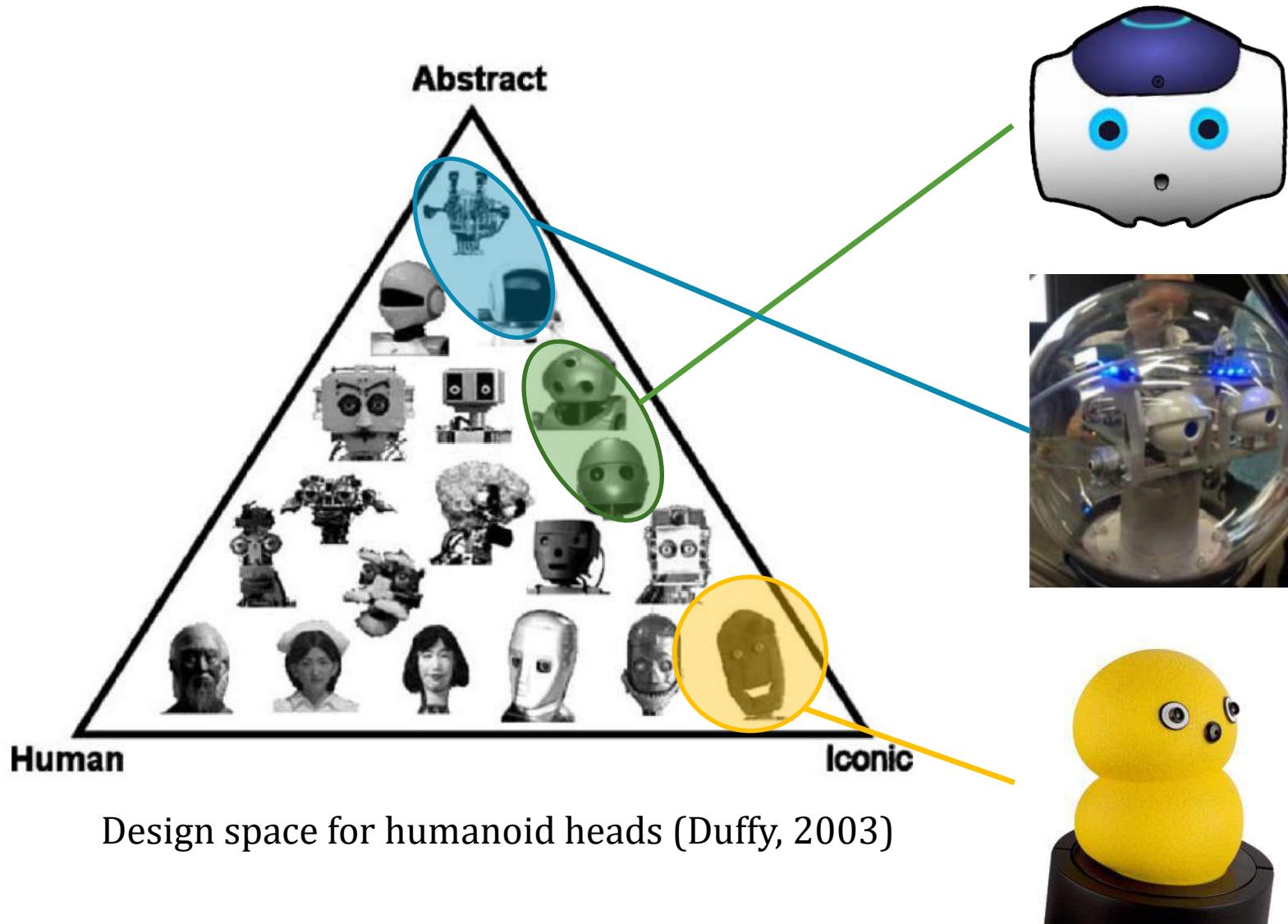


Anthropomorphism: Appearance



Pareidolia – perceiving a familiar pattern (typically face) where there is none

Anthropomorphism: Robot Faces



Anthropomorphism: Behaviour

Apparent Behaviour (Heider & Simmel, 1944)

Anthropomorphism: Behaviour



While working on your assignment, how many of you:

- Swore at your robot?
- Complimented your robot?
- Referred to your robot as he/she?
- Gave your robot a name?

Anthropomorphism: Behaviour

Full video: <https://www.youtube.com/watch?v=VWeRC6j0fW4>
Also see: http://cosmo.nyu.edu/hogg/lego/braitenberg_vehicles.pdf

Anthropomorphism: Robots



Human-Likeness

Affetto (Osaka University)

Saya (Kobayashi et al)



KR1000 (Kuka Robotics)



Nao (Softbank Robotics)



iCub (IIT et al)



Diego-san (UCSD et al)



The Uncanny Valley

+

familiarity

-

<http://sp>

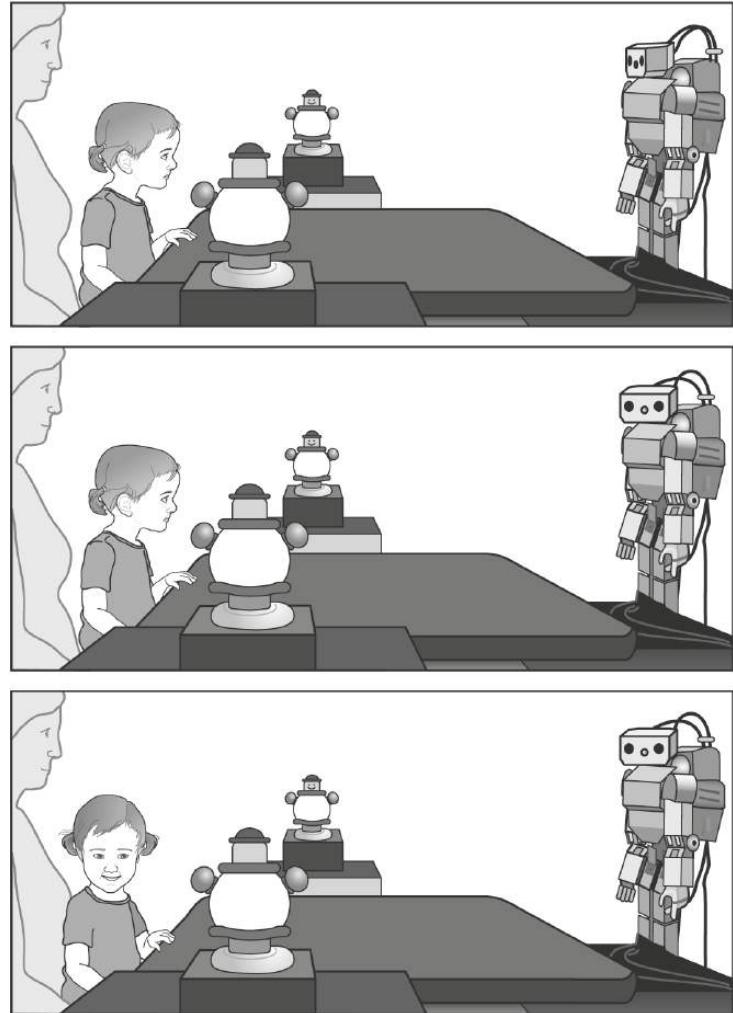


- One possible explanation: a conflict between cues (Moore, 2012)
 - Hence why the “moving” curve more pronounced
 - Greater mismatch between movement and appearance (see anthropomorphism notes above)

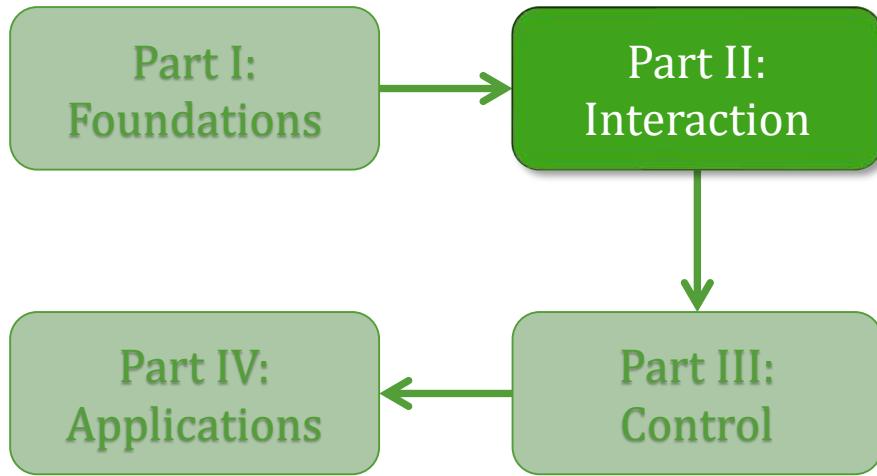
- The “Uncanny Valley”
 - Increasing human likeness increases familiarity...
 - ...however, at a certain point, a sharp drop in familiarity, resulting in revulsion, etc
- Refer to (Mathur et al, 2016) for an overview
- Reasons for this not fully understood

Attribution of Agency to Robots

- Particularly if certain characteristics are present, people will naturally attribute agency to an inanimate object
 - Though this illusion can be broken in the interaction
- This is (at least partly) learned from experience
- Seen in children with a robot for example
 - Meltzoff et al (2010)
 - Children watch adult interact with robot (joint attention)
 - These children then more likely to follow gaze when they interact with robot themselves



From Meltzoff et al (2010)



Part II: Interaction

Types and contexts

Fundamentally two modes...

Proximate

- The human and the robot are in the same space
- Physical and social interactions fall in this category
 - E.g. service robots



Remote

- The human and the robot are in different locations
 - Maybe even temporally removed
- E.g. teleoperation, supervised control, ...



Two types of Interaction

Explicit

- An action performed with the purpose of eliciting a reaction
- Dialogue: e.g. asking a question
- Manipulation: e.g. handing over an object, or pointing

Implicit

- Actions are modified due to presence of an other, but no reaction is expected
- Navigation: e.g. avoiding humans in the way

Capacity for overlap between Explicit and Implicit: e.g. avoiding humans, but engaging in some strategies to encourage humans to move out of the way (next week!)

Explicit Interactions: example

- Leonardo robot with Andrea Thomasz (MIT, 2006)
- Explicit interaction
 - Pointing from human
 - Gaze gestures from robot to indicate ready for next instruction
- See: <https://www.youtube.com/watch?v=GHIIFrL7dKM>

Explicit Interactions: example

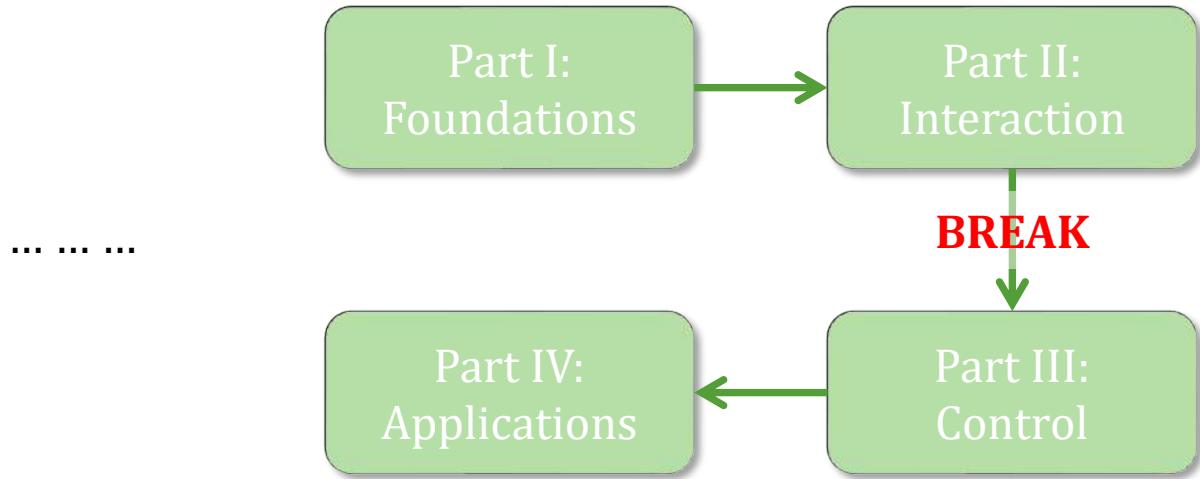
- Explicit interaction scenario
 - Robot and human facing each other
 - Using the same workspace
 - Human providing explicit instruction, with gestures
 - Robot performs actions, looks back at human
- Human teaching:
 - Names, attributes, affordances
- Robot learns from:
 - Speech, observation

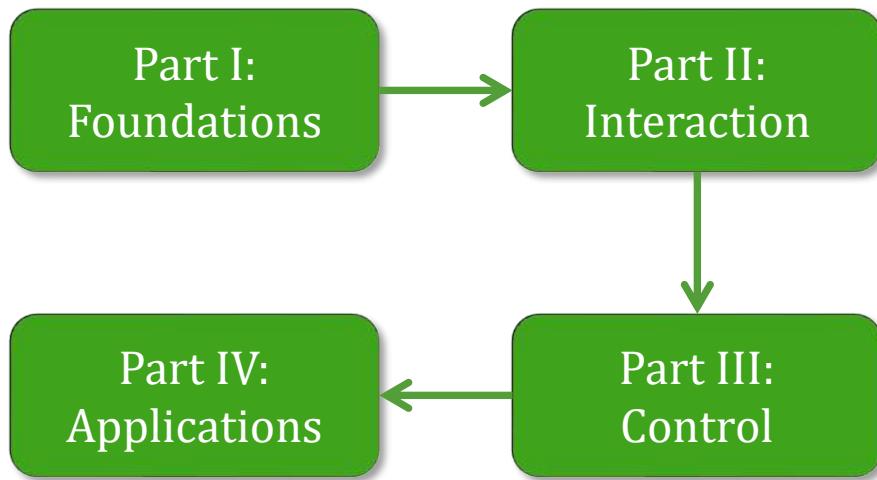
Implicit Interactions: example

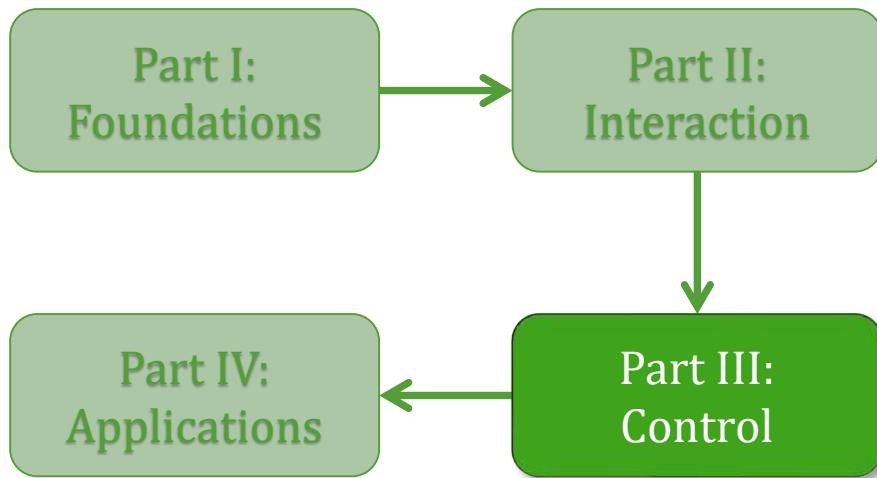
- FROG project: robot museum guide
- Video from HRI 2014 conference
- Implicit interaction in terms of navigation, avoidance, spatial positioning, and other aspects of behaviour (more on this next week!)

Implicit Interactions: example

- E.g. robot navigates through populated environment
- Humans change their behaviour
 - They stop
 - Deviate path to avoid robot
- Interaction not strictly necessary for the task of navigation however:
 - If done correctly, can improve efficiency
 - E.g. shorter path found/planned due to person moving out of the way
 - (this may require explicit interaction strategies – e.g. “please move out of my path”)
- Queueing has similar implicit interaction characteristics







Part III: Control

Autonomy and architectures

Control for Autonomous Robots

This should be familiar from last week!

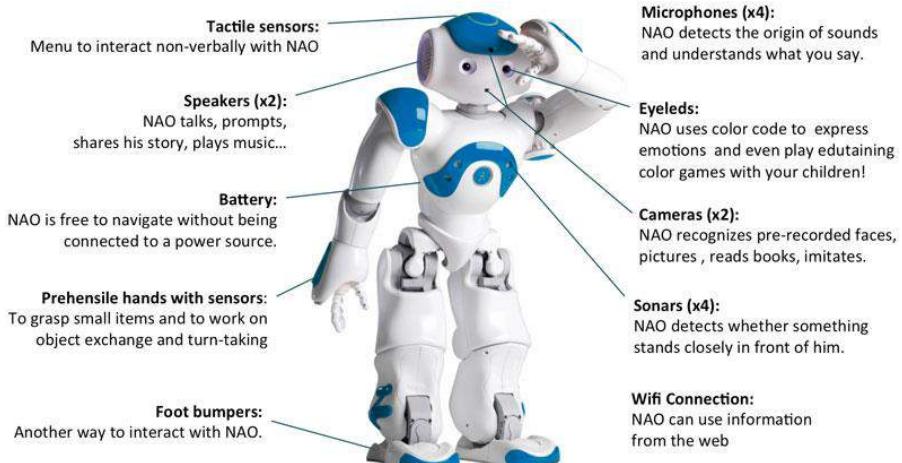
- Sensing
- Decision making
- Acting



Sphero Robotics

Sensing

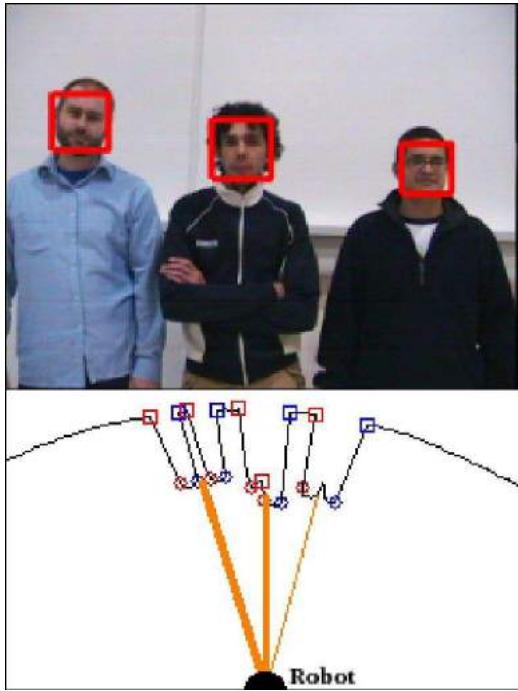
- Recall the sensing you did in your assignment:
 - Array of sensors (depth, vision, bump, ...)
 - This environment was static (nothing moving except the robot)
- Taking into account humans adds significant difficulty
 - Not just because people move...
 - ... also unpredictability, occlusions, etc
 - And: the meaning of underlying expressions, gestures, etc
- People are not good at being reliable...
- Sensors suffer from noise...



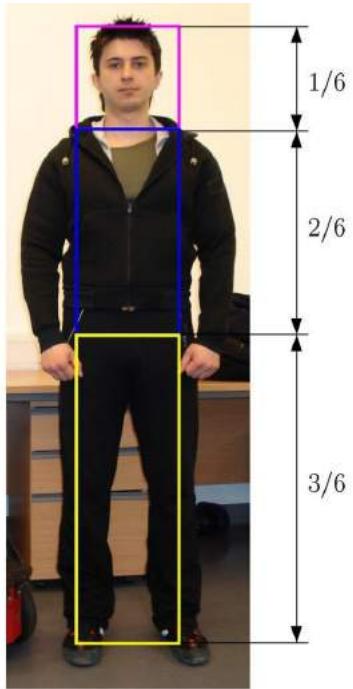
Sensing is difficult...

See paper and video (2012) at <http://dl.acm.org/citation.cfm?doid=2559636.2559650>

Human Detection

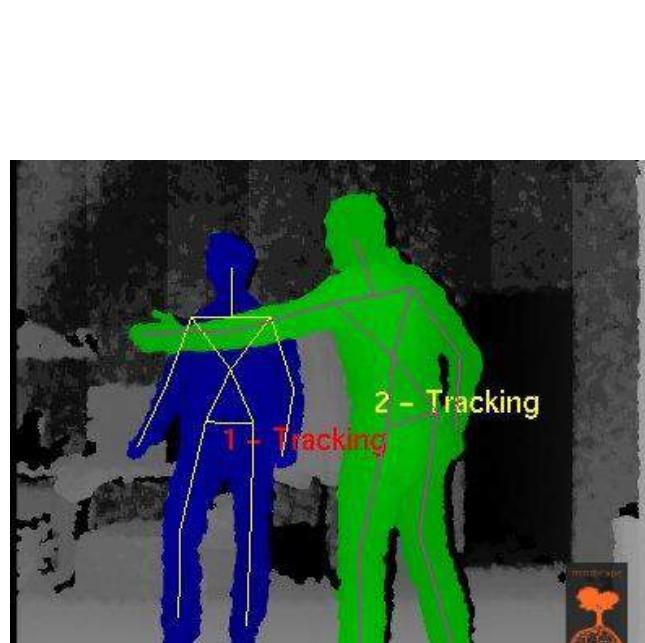


Person detection from
laser scans of legs



Person detection from
camera-based clothing
and face recognition

Both images from Dr. Bellotto homepage:
<http://webpages.lincoln.ac.uk/nbellotto/software.html>



Person/skeleton
tracking from RGB-D
information (e.g.
Kinect):
MS Kinect SDK, OpenNI
e.g. <https://structure.io/openni>

Decision Making: what is Autonomy?

- Implicit assumption so far (see last week) that our robots are *autonomous*
- Many (very involved) definitions that are philosophically-inclined...
 - E.g. based on autopoiesis...
- Practical characterisations:
 - <http://humanrobotinteraction.org/autonomy/>
 - **The amount of time that a robot can be 'neglected' by the designer/operator**
 - High autonomy: long periods acting on its own
 - Low autonomy: no/short periods of acting alone

Levels of Autonomy

Teleoperation

- 
1. Wizard of Oz
 - Teleoperation
 2. Scripted Robot Behaviour
 - Operator runs pre-scripted behaviours
 3. Partial Autonomy
 - Hybrid autonomous/controlled system
 4. Supervised Autonomous Behaviour
 - System acts autonomously, but is supervised, with human take-over in uncertainty
 5. Full Autonomy
 - No human intervention required

Autonomous

1. Wizard of Oz

- Remote control of a robotic system, or aspects thereof
 - May be mixed with varying levels of autonomy
 - Typically used to stand in for technical aspects that are currently too difficult/unreliable/under test
- From 2012:
 - Most uses of WoZ for Natural Language Processing

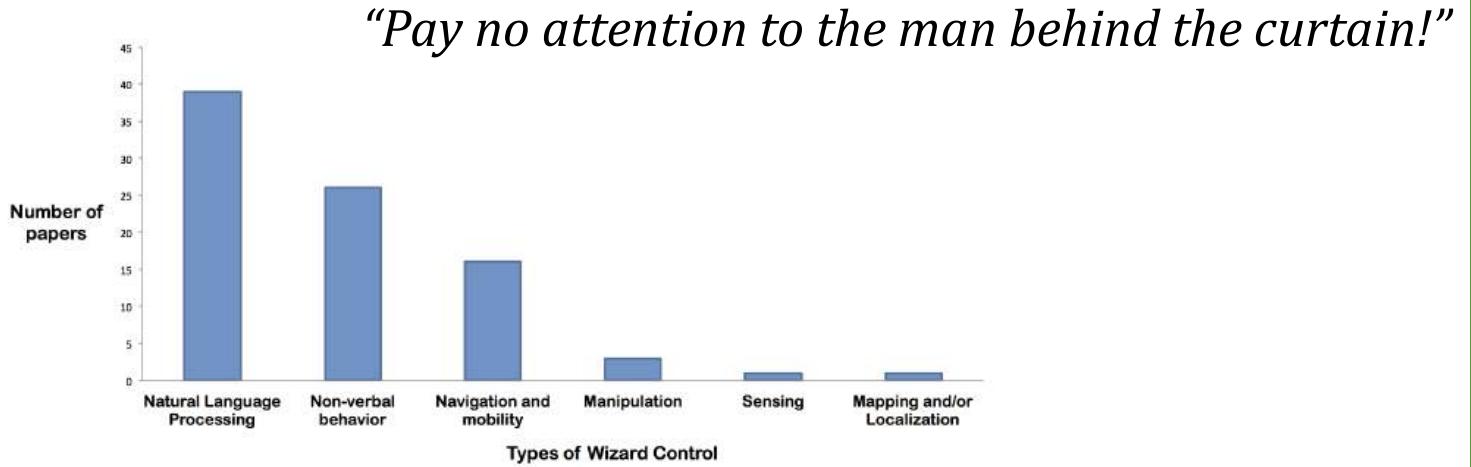
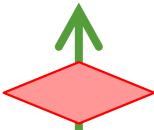


Figure 4. Chart depicting the types of Wizard control employed in the included papers. Some papers described using more than one type of control.

1. Wizard of Oz

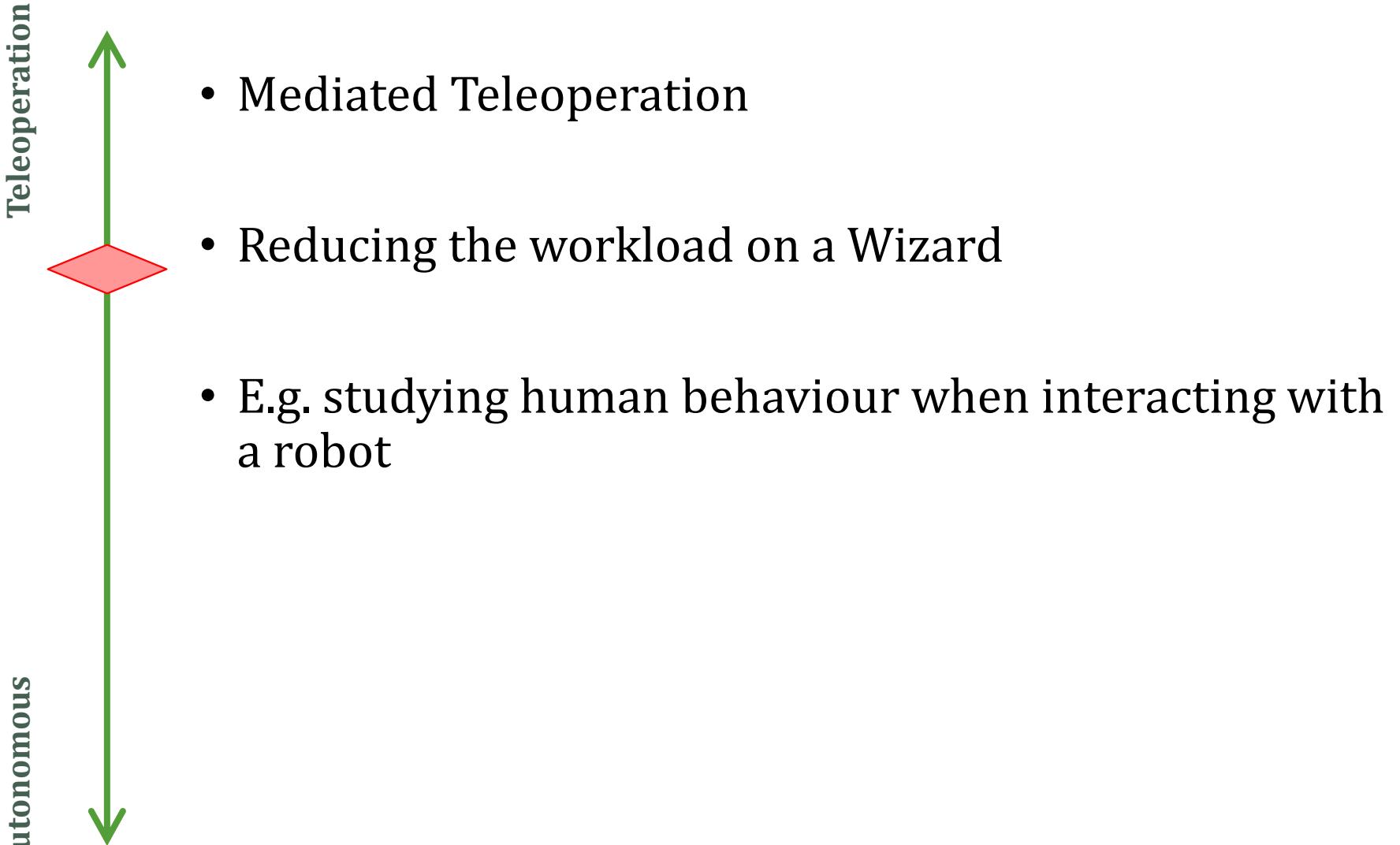
Teleoperation



- Teleoperation
- Giving the impression of autonomy
- The Wizard is hidden from view, or not obviously associated with the control of the robot
 - Either way, the participant is unaware of the remote control.
- Complete WoZ entails full remote control of all aspects of behaviour

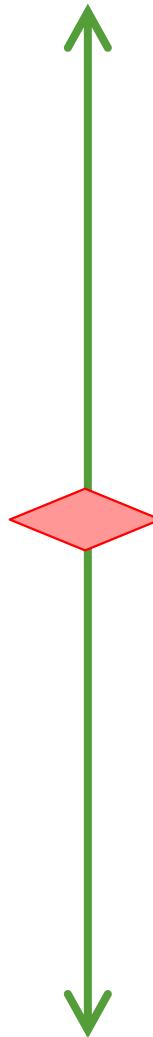
Autonomous

2. Scripted Robot Behaviour



3. Partial Autonomy

Teleoperation



- Testing subsystems of a robot control system
 - Components that are ready can be run autonomously
 - Those that are not can be wizarded
- Use of shared autonomy
 - Hand-off between robot and human team-mates
 - E.g. in disaster scenarios:
partially autonomous
search prior to rescue



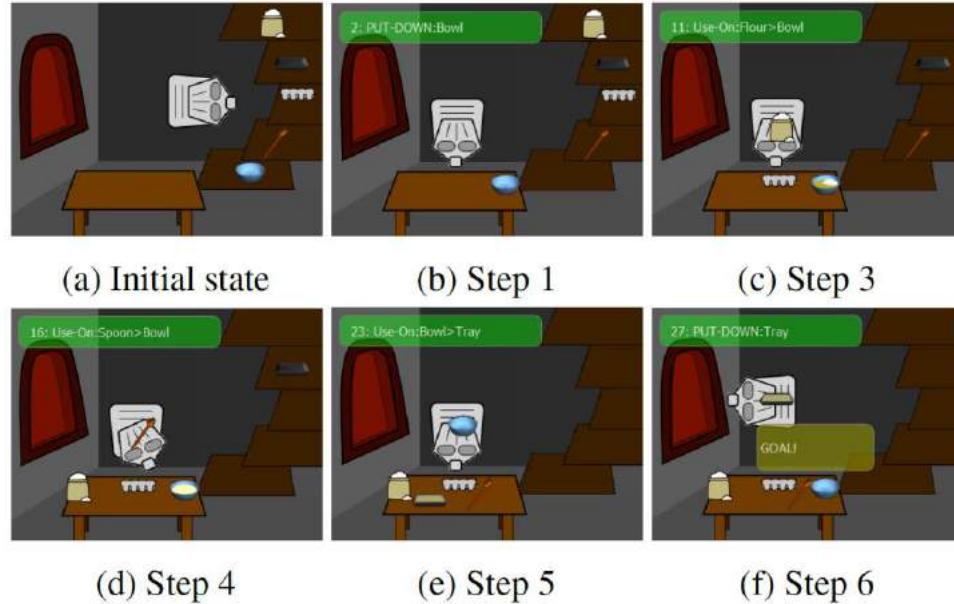
<http://www.tradr-project.eu/>

4. Supervised Autonomous Behaviour

Teleoperation



- Attempting to overcome noisy sensors
 - If very unreliable, then a helping hand may be needed
- Guidance for a learning robot system
 - Helping it to learn



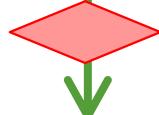
5. Full Autonomy

Teleoperation



- The typical goal for robot development
 - No requirement for human remote-controller present
 - Though environment/context may require someone close by for safety...
- Full autonomy implies capacity for adaptation
 - E.g. only using predefined behaviours or waypoints may be autonomously executed, but useless if something changes (e.g. obstacles)
 - Adaptation suggests learning...

Autonomous



Contrasting WoZ and Autonomy

Social Sciences Point of View

WoZ

- Pros:
 - Remove uncertainty
 - Focus on evaluation of interaction rather than robot
 - Full control over robot behaviour
 - Repeatable experiment setup (??)
 - Easier to implement
- Cons:
 - Human-human interaction with a robot in the middle?
 - Not consistent...

Autonomous

- Pros:
 - Study with state-of-the-art robot instead of dummy
 - Testing system robustness
 - How to replicate human-like learning on robot
- Cons:
 - High uncertainty
 - Not necessarily repeatable
 - High maintenance
 - Can be slow...

Contrasting WoZ and Autonomy

Computer Science Point of View

WoZ

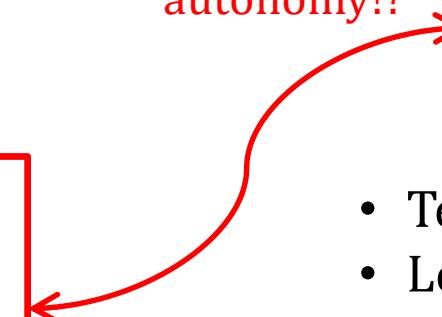
- Pros:
 - Remove uncertainty
 - Evaluate robot design
 - Repeatable experiment

- Cons:
 - No evaluation of:
 - Perception
 - Reasoning
 - Learning
 - World Model
 - Action selection
 - Evaluates only robot design

Autonomous

- Pros:
 - Evaluation of:
 - Perception
 - Reasoning
 - Learning
 - World Model
 - Action selection
 - Testing system robustness
 - Learning from interaction
- Cons:
 - High uncertainty
 - Not necessarily repeatable
 - Can be slow...

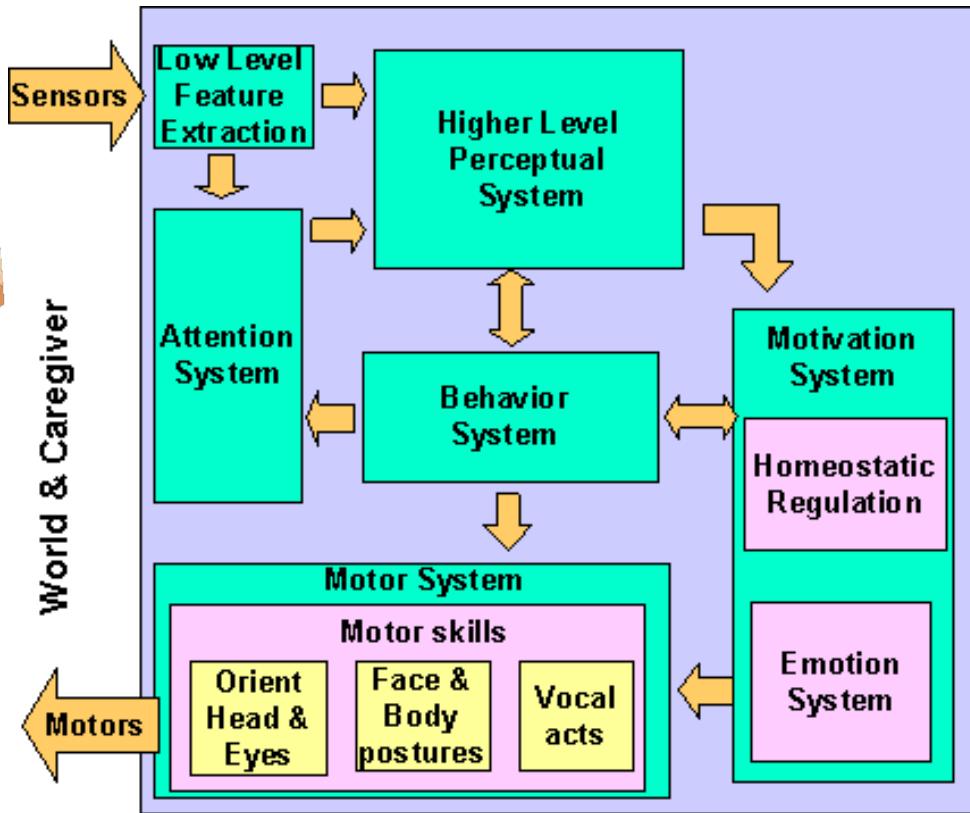
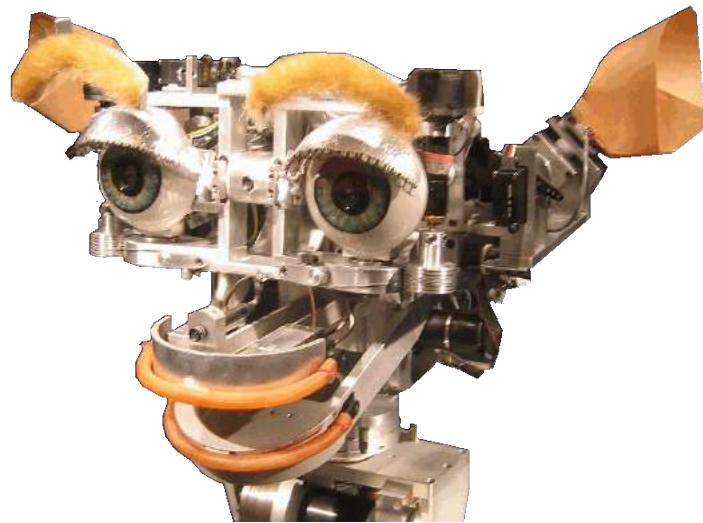
Level of
autonomy!?



Control Architectures: Reactive

- Kismet (MIT, 1990's)
- See video at: <https://www.youtube.com/watch?v=8KRZX5KL4fA>
- Note the brief discussion of anthropomorphisation...

Control Architectures: Reactive



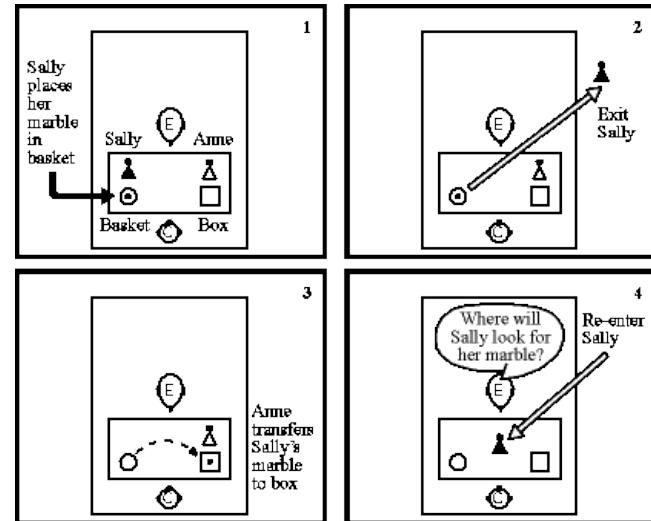
- Kismet control architecture
- See <http://www.ai.mit.edu/projects/sociable/kismet.html>

Control Architectures: Cognitive

- MDS – (Octavia @ NRL)
- Video: <https://www.youtube.com/watch?v=aQS2zxmrrrA>
- Controlled with cognitive architecture ACT-R/E (Trafton, 2013)

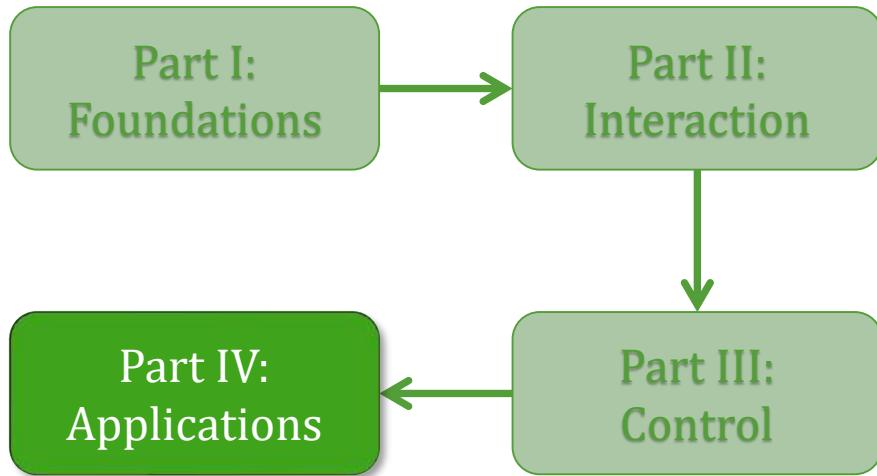
Control Architectures: Cognitive

- Octavia case study: Theory of Mind
 - The capacity of humans to attribute mental states to others and to oneself, and to understand that others may have different states than oneself



<http://www.holah.karoo.net/baronstudy.htm>

- Further relevant concepts to CogArch:
 - Belief-Desire-Intention architectures, e.g.
<http://www.aaai.org/Papers/ICMAS/1995/ICMAS95-042.pdf>
 - Affective Computing related to HRI, e.g.
<http://cseweb.ucsd.edu/~lriek/papers/riek-acd-aisb09.pdf>



Part IV: Applications

Robot roles and ethics

Robot Roles: Learning Peer

ROBOTICS
WITH
PLYMOUTH
UNIVERSITY



The 'Sandtray'

Mediating Social Human-Robot Interaction

Plymouth University, U.K.

- Robot as a learning partner for a child:
<http://ieeexplore.ieee.org/abstract/document/6249477/>
- Robot and child co-located, facing each other over a touchscreen, and can both interact with the touchscreen
- Robot plays the role of a peer: not perfect performance, makes mistakes, and adapts its behaviour

Robot Roles: Security and Nurse Support

Robot as an assistant

See the STRANDS project <http://strands.acin.tuwien.ac.at/>

Robot Roles: Therapy Aid

- Probo robot: a green elephant-like cuddly robot
- Used in autism therapy for children, under the supervision of a therapist
- Used in Robot-Enhanced Therapy, see <http://www.dream2020.eu/>

Other Robot Roles

Have seen:

- Robot as learning partner
- Robot as assistant
- Robot as therapy tool

Could be:

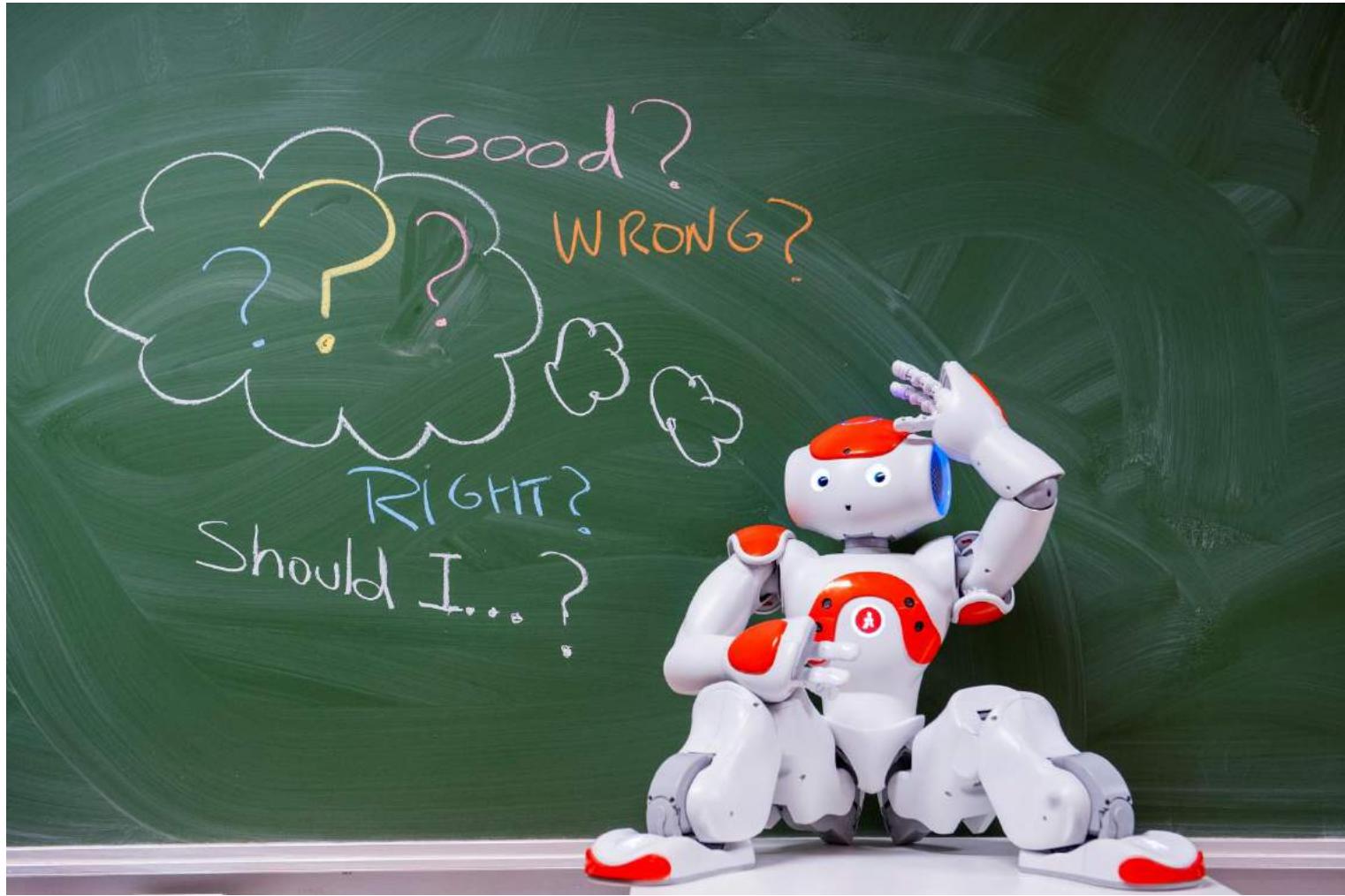
- Robot as teacher/tutor
- Robot as tool
- Robot as therapist
- Robot as team member

(Autonomous) Tool

Social Agent

- Above, level, below in social hierarchy

Ethical Issues

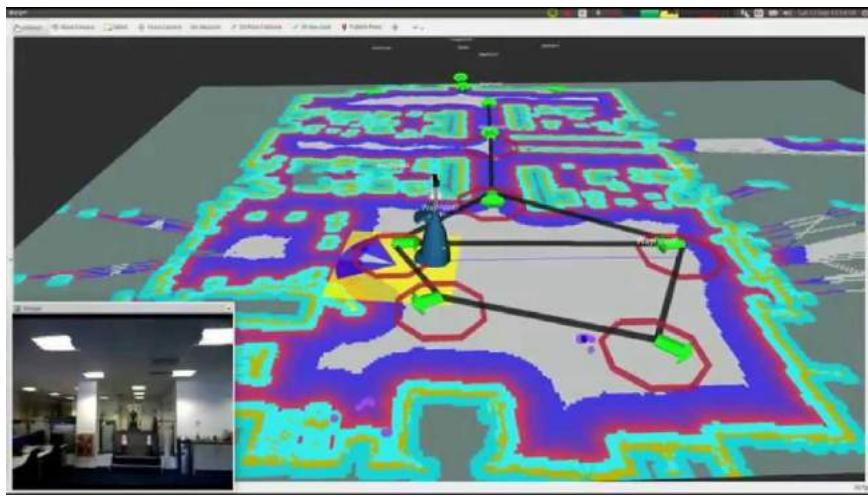


Ethical Issues

- Is it right to use robots as social agents in this way?
 - Issue of deception?
 - Are we replacing social contact with other humans if we use these devices?
 - In the case of child therapy, are the problems made worse (e.g. getting used to interacting with robot rather than people)?
 - Attachment to robots rather than humans...
- Technical/Legal concerns:
 - Data protection, and the role of data recording/capture
 - Memory of prior interactions and privacy of this information
- We'll consider some more aspects in week 13...
- ...

Next Week...

- Engagement in HRI
- Navigation in HRI
 - Human-aware planning
- Methods for HRI Evaluation



Reminder: Marking Sessions

- Start this Friday 29th and continue in timetabled workshop sessions until Monday 8th April
- **See Blackboard (and email) announcement on 20th March for details**
- Reminder:
 - You must attend the time slot you have been allocated to
 - 5 minutes to run simulation (continuous, may restart within this): you will start this
 - Deliver presentation – may be useful to do in parallel with the simulation
 - We will ask questions/explore your code and reasoning
 - We will have all submitted code/presentations ready
 - Shared register: only attend the session you are assigned to

Workshops after marking sessions

- Friday/Monday 12th/29th April
- Workshop activity related to lectures both today and next week: mobile robot behaviours relevant to human-robot interaction
 - Part paper-based exercise: working from an academic paper
 - Part implementation: starting from the workshop code and assignment simulation environment
- Academic paper on Blackboard in workshop materials for week 11:
 - Will return to this paper in next week's lecture
 - Recommend you at least take a look at the paper before the workshop...

References / Reading

- Baxter, P. et al., 2016. From Characterising Three Years of HRI to Methodology and Reporting Recommendations. In *HRI 2016*. Christchurch, New Zealand: ACM Press, pp. 391–398.
- Duffy, B.R., 2003. Anthropomorphism and the social robot. *Robotics and Autonomous Systems*, 42(3–4), pp.177–190.
- Heider, F., Simmel, M., 1944. An experimental study of apparent behavior. *The American Journal of Psychology*, Vol 57, 243-259
- Mathur, M.B. & Reichling, D.B., 2016. Navigating a social world with robot partners: A quantitative cartography of the Uncanny Valley. *Cognition*, 146, pp.22–32.
- Meltzoff, A.N. et al., 2010. “Social” robots are psychological agents for infants: a test of gaze following. *Neural networks : the official journal of the International Neural Network Society*, 23(8–9), pp.966–72.
- Moore, R.K., 2012. A Bayesian explanation of the “Uncanny Valley” effect and related psychological phenomena. *Scientific Reports*, 2, p.864.
- Riek, L., 2012. Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. *Journal of Human-Robot Interaction*, 1(1), pp.119–136.
- Senft, E., Baxter, P., Kennedy, J., Lemaignan, S., & Belpaeme, T., 2017. Supervised Autonomy for Online Learning in Human-Robot Interaction. *Pattern Recognition Letters*, 99, p77–86.
- Trafton, J.G. et al., 2013. ACT-R/E : An Embodied Cognitive Architecture for Human-Robot Interaction. *Journal of Human-Robot Interaction*, 2(1), pp.30–54.

CMP3101M AMR – Week 11

Human-Robot Interaction

part 2

Dr. Paul Baxter

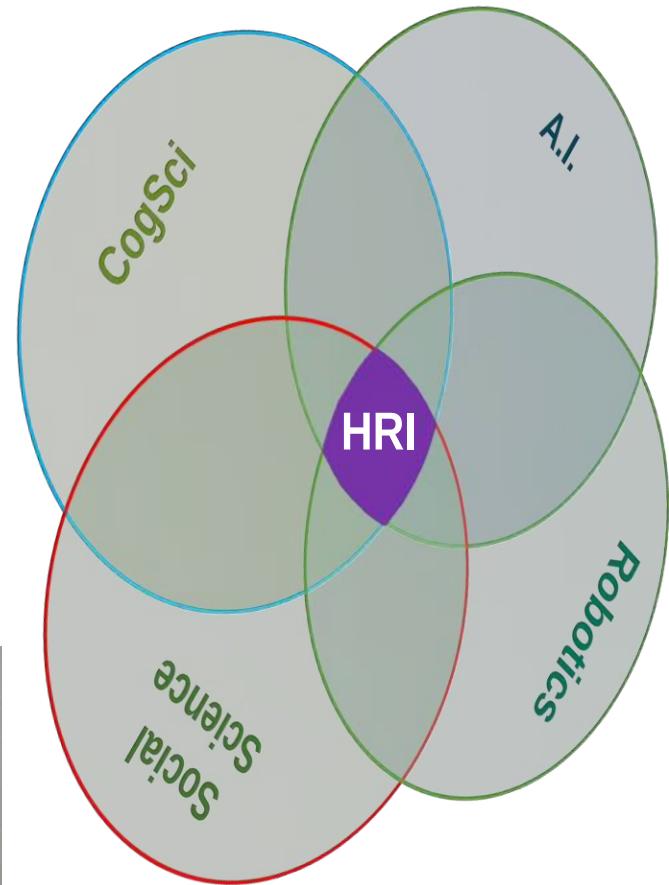
Location: INB3119

pbaxter@lincoln.ac.uk

Office hours: Tuesdays 9am-11am

Last Week: Intro to HRI

- HRI is “*dedicated to understanding, designing, and evaluating robotic systems for use by or with humans*”
- Anthropomorphism as an important factor:
 - Is the “tendency to attribute human characteristics to inanimate objects, animals and others with a view to helping us rationalise their actions”
 - (Duffy, 2003)
- The Uncanny Valley



From (Baxter et al, 2016)

Modes of Interaction

Proximate

- The human and the robot are in the same space
- Physical and social interactions fall in this category
 - E.g. service robots



Remote

- The human and the robot are in different locations
 - Maybe even temporally removed
- E.g. teleoperation, supervised control, ...



Two types of Interaction

Explicit

- An action performed with the purpose of eliciting a reaction
- Dialogue: e.g. asking a question
- Manipulation: e.g. handing over an object, or pointing

Implicit

- Actions are modified due to presence of an other, but no reaction is expected
- Navigation: e.g. avoiding humans in the way

Capacity for overlap between Explicit and Implicit: e.g. avoiding humans, but engaging in some strategies to encourage humans to move out of the way (more today!)

Levels of Autonomy

Teleoperation

- 
1. Wizard of Oz
 - Teleoperation
 2. Scripted Robot Behaviour
 - Operator runs pre-scripted behaviours
 3. Partial Autonomy
 - Hybrid autonomous/controlled system
 4. Supervised Autonomous Behaviour
 - System acts autonomously, but is supervised, with human take-over in uncertainty
 5. Full Autonomy
 - No human intervention required

Robot Roles and Ethics

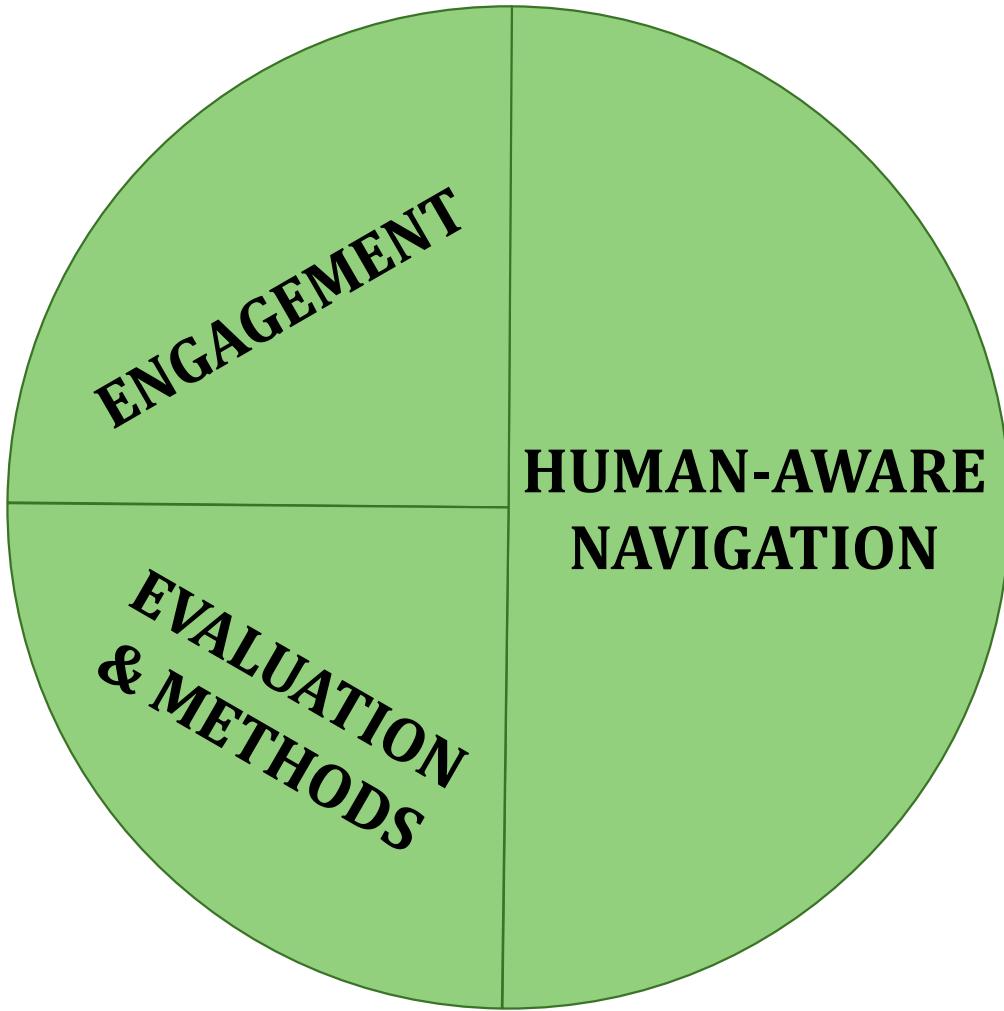
- Robot could take on a number of roles in an interaction, both social and non-social:
 - Learning peer / tutor
 - Therapy assistant / tool
 - Remote presence
 - Team member
 - ...
- Ethical issues can arise in each of these:
 - Deception
 - Attachment (particularly, but not just, with children)
 - There are consequences for technical implementation: e.g. data protection and deletion

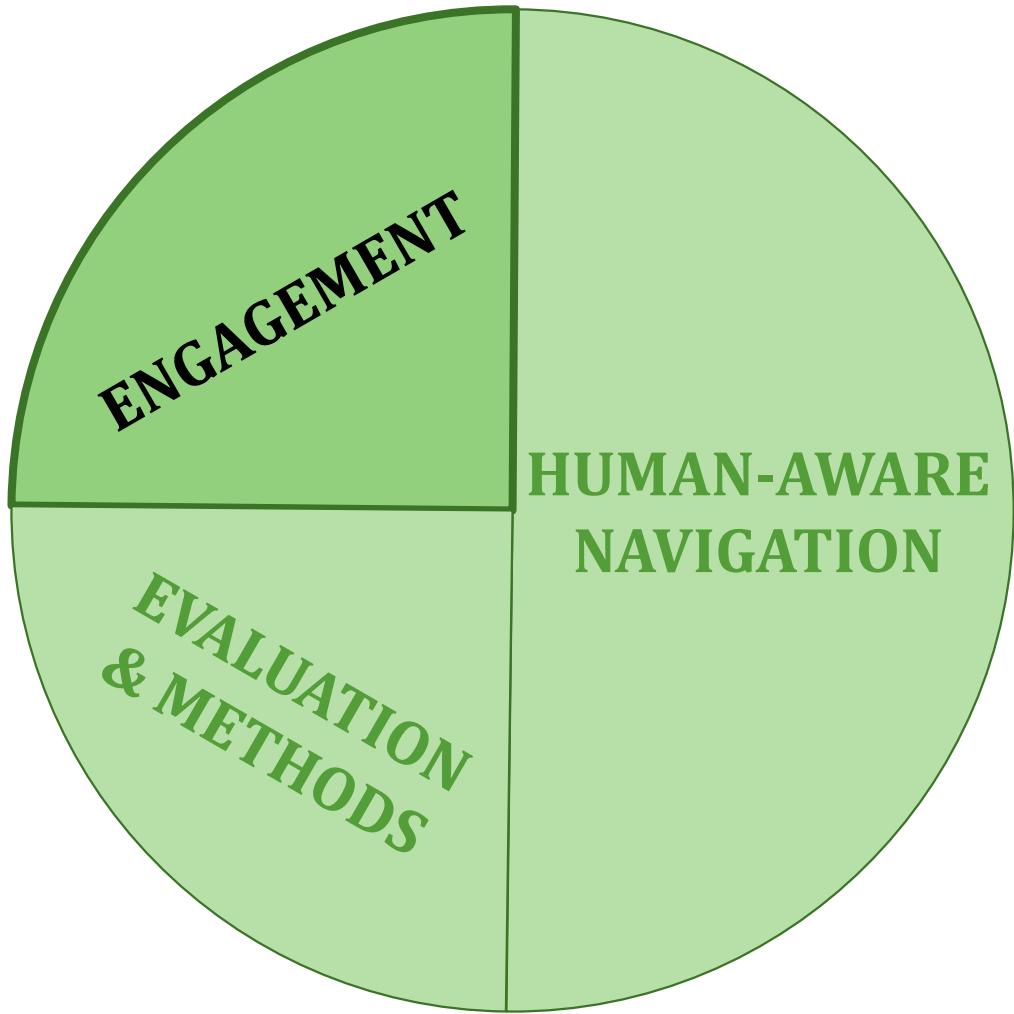


Syllabus review

Week	Topic	Lecturer
1	Introduction	Marc Hanheide
2	Robot Programming (ROS)	Marc Hanheide
3	Robot Sensing	Marc Hanheide
4	Motion & Control	Marc Hanheide
5	Robot Behaviour	Ayse Kucukyilmaz
6	Navigation 1	Ayse Kucukyilmaz
7	Navigation 2	Ayse Kucukyilmaz
8	Robot mapping & SLAM	Ayse Kucukyilmaz
9	Control Architectures	Paul Baxter
10	Human-Robot Interaction 1	Paul Baxter
11	Human-Robot Interaction 2	Paul Baxter
12	Applications	Paul Baxter

Today



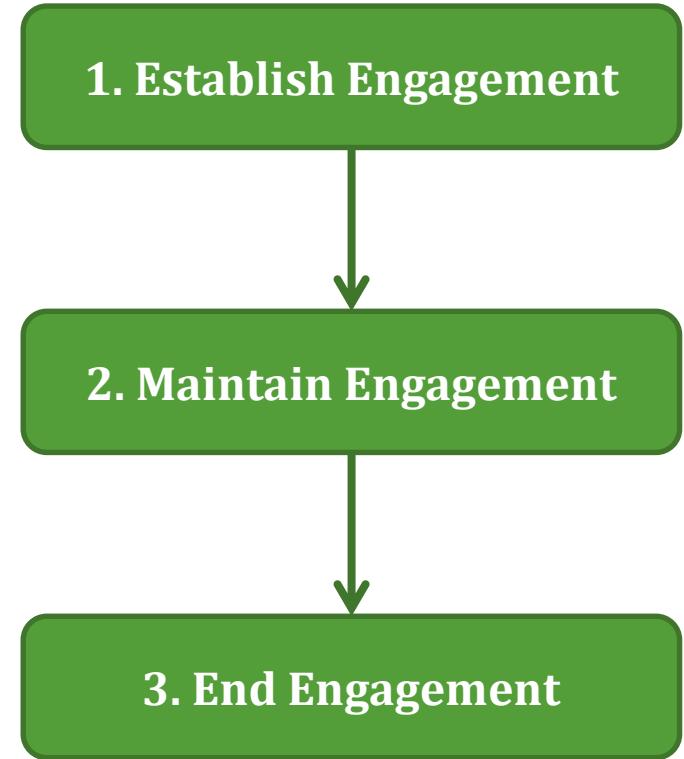


What is engagement?

- Many different definitions...
See e.g. Glas & Pelachaud, 2015 for an overview of these
- Sidner et al 2005:
“...the process by which two (or more) participants establish, maintain and end their perceived connection during interactions they jointly undertake.”
- It encompasses all types of behaviour:
 - Implicit/Explicit
 - Verbal/Nonverbal

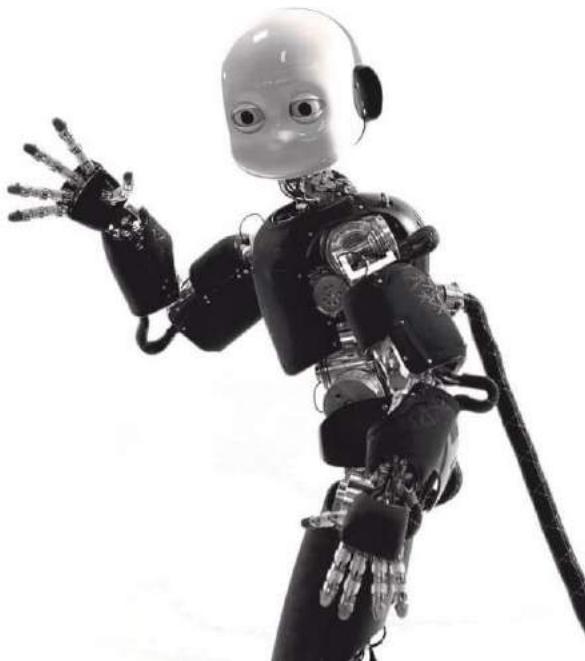


Stages in Engagement



1. Establish Engagement

- Attracting attention
- Drawing into an (ongoing?) interaction



1. Establish Engagement

Social Strategies

- Speech
 - Saying something to someone
- Gesture
 - Only useful if have limbs...
 - Waving, etc
- Behaviour
 - Could engage in attention seeking behaviour
- Physical Interaction
 - Equivalent of a tap on the shoulder... (clearly safety implications)

Non-Social Strategies

- Sound
 - Pre-recorded speech
 - Alarm / beeps / motor noise
 - White noise
- Vision
 - Flashing lights
- Behaviour
 - Bumping into human (not advisable...)



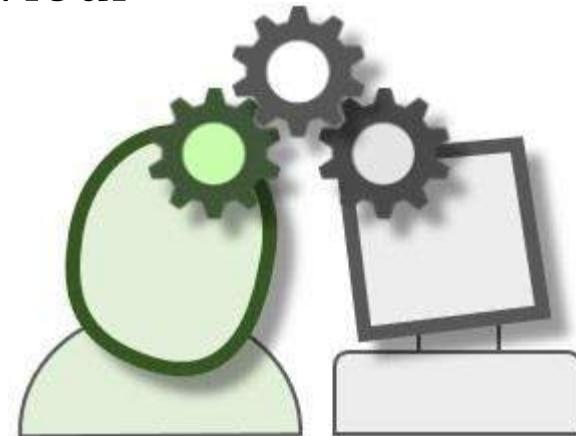
Video from:
<https://www.youtube.com/watch?v=asHaWo04mIo>

2. Maintain Engagement

- Recall that we are at least partially relying on:
 - Anthropomorphism: appearance and behaviour
 - Attribution of agency

This can only get you so far!

- The necessity for going beyond this with “deeper” complexity and adaptivity of behaviour
 - Refer to Control Architectures Lecture, Week 9
 - This also underlies part of the motivation for incorporating Cognitive Architectures into HRI systems



See: <https://sites.google.com/site/cogarch4socialhri2016/>

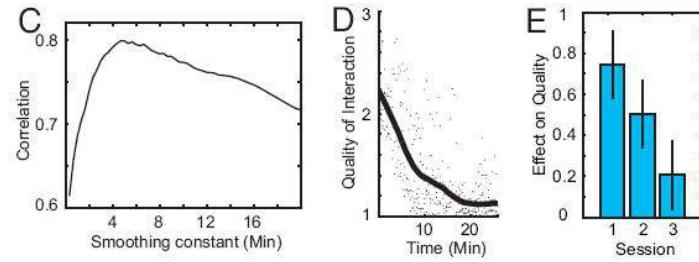
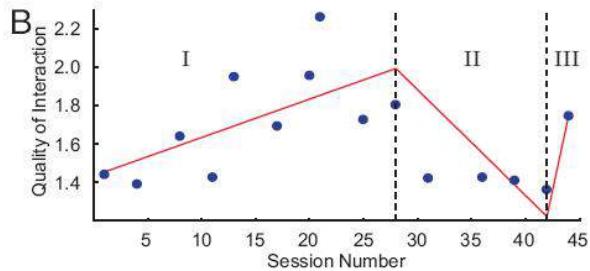
3. End Engagement

- Ending the interaction, and hence any engagement in the interaction
 - Possibly to be resumed at a later time
- Task related:
 - Joint task has completed successfully/unsuccessfully
 - Task no longer relevant
- Personal related
 - Illusion of agency has gone
 - Boredom!



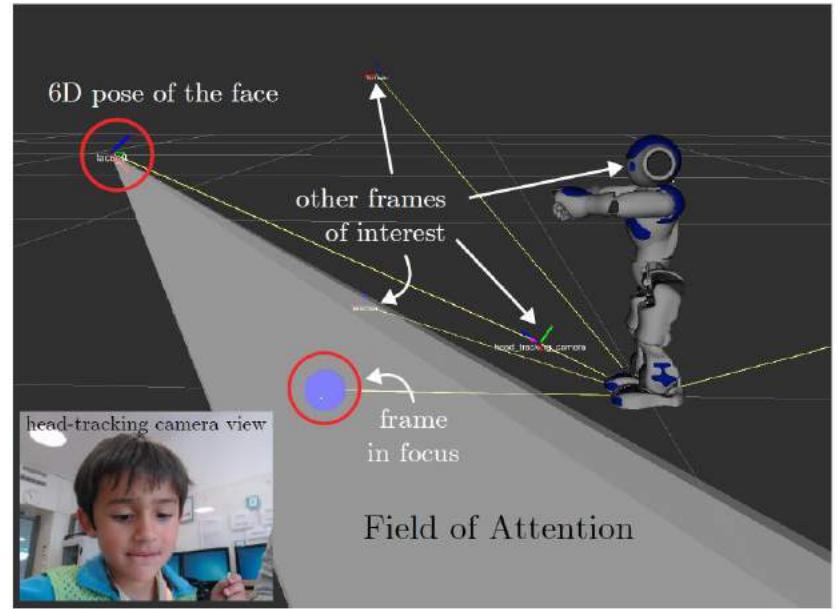
How to detect Engagement?

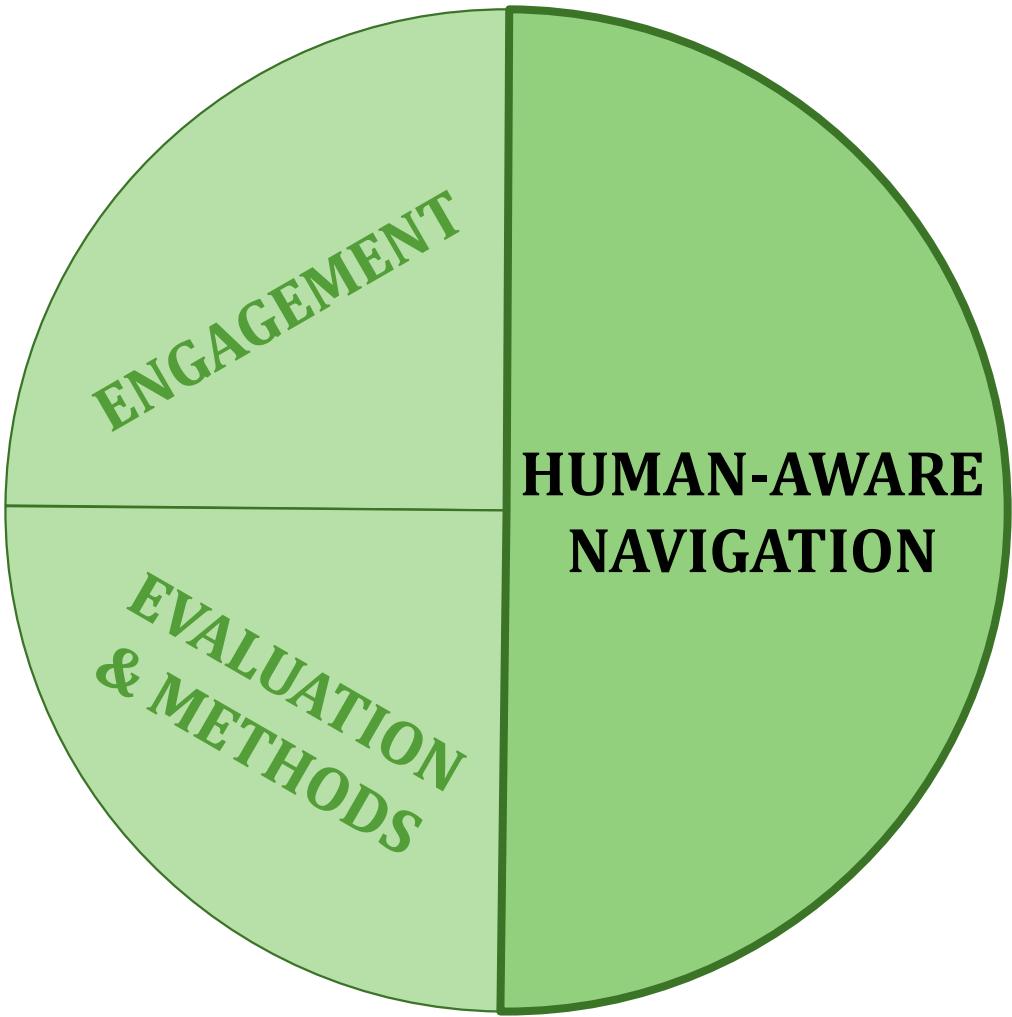
- How to tell whether someone is engaged in an interaction?
 - And, ideally, how to do so automatically?
 - What should you examine?
- This is a difficult task!
- Humans have an intuitive sense of engagement, based on extensive experience:
 - Developed from baby onwards
 - Can take advantage of this
 - E.g. Tanaka et al (2007): rating using a 'slider'



How to detect Engagement?

- Gaze is often used as an indicator of engagement
 - E.g. Baxter et al, 2014
- The problem is that this is often post hoc analysis
 - I.e. not in real-time, but only afterwards
- Recent developments trying to achieve this in real-time, using robot sensors:
 - E.g. the GAZR gaze estimator, which can be used for an estimation of engagement (Lemaignan et al, 2016)
 - ROS package:
<https://github.com/severin-lemaignan/gazr>





“Will I bother here? - A robot anticipating its influence on pedestrian walking comfort”, HRI 2013, Tokyo, Japan

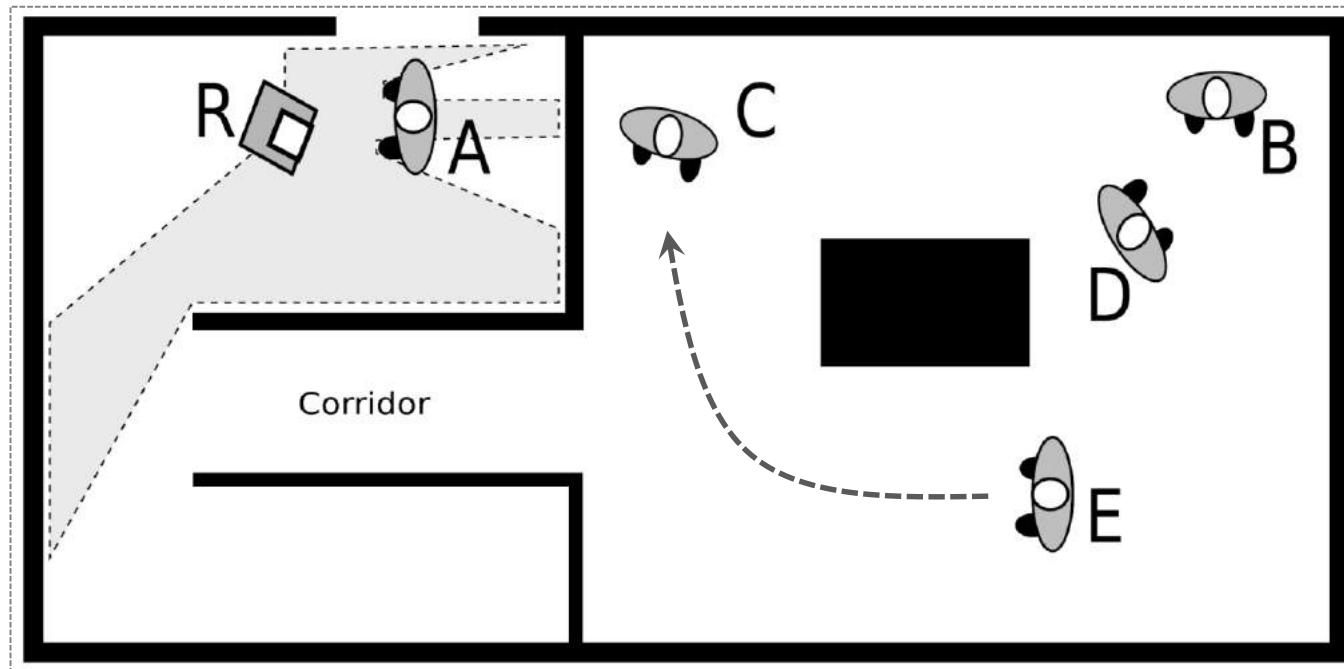
Paper: <http://ieeexplore.ieee.org/abstract/document/6483597/>

Human-Aware Navigation

- Autonomy for mobile robots requires navigation capabilities
- Obstacle avoidance clearly required
 - Preventing robot damage
 - Human safety!
- Goals (Kruse et al, 2013):
 1. Comfort: absence of annoyance and stress for humans
 2. Naturalness: similarity of robot behaviour to humans
 3. Sociability: adherence to high-level cultural constraints
- May be necessary to reduce efficiency (in terms of speed/distance to goal) in the service of these goals

An example

From Kruse et al, 2013:

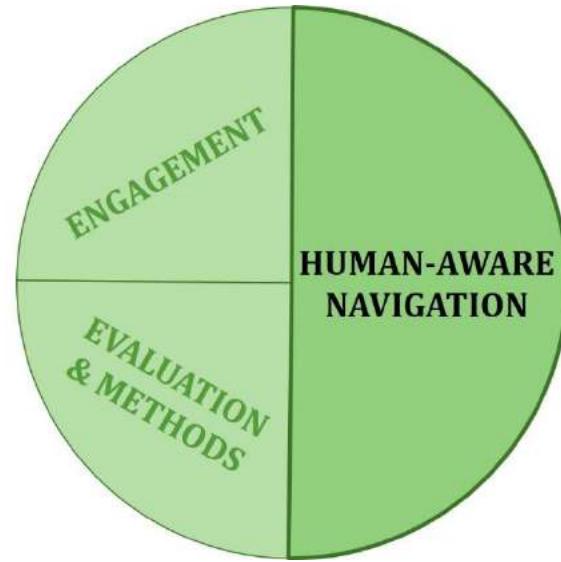


How should the robot guide person A to person B?

Humans are Awkward Obstacles...

Also see this L-CAS video:

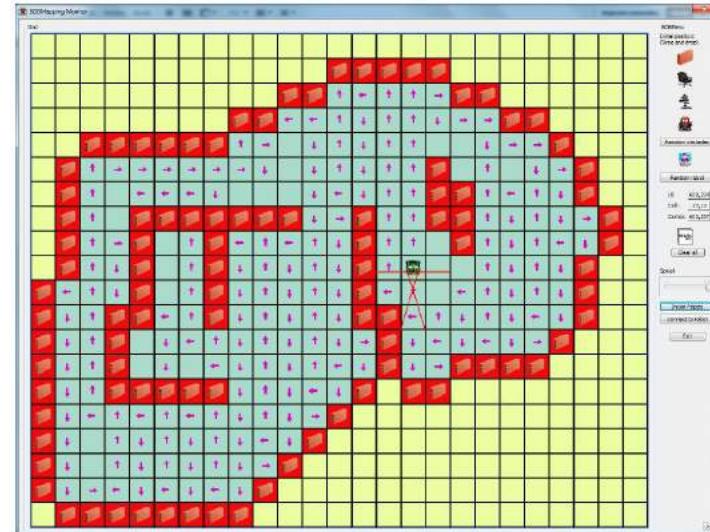
<https://www.youtube.com/watch?v=zdnhQU1YNo>



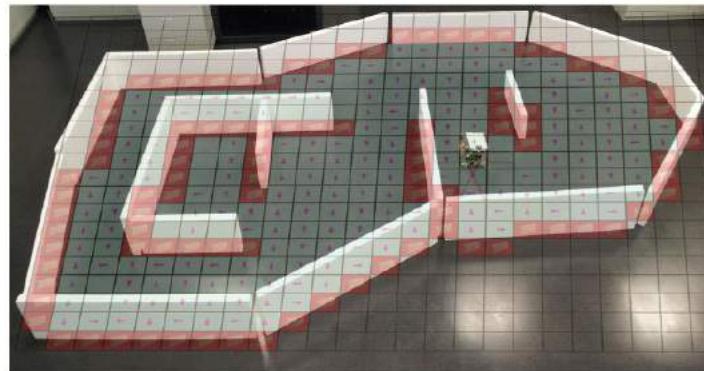
(a) Navigation and Costmaps

Recap: path planning

- Refresh your memory of Lecture Week 7: Navigation 1
 - Dijkstra and A*
 - Include movement cost into node
- Application to discrete states, in this example a network of nodes
- Equally applies to a 2D space discretised in a grid
 - “Occupancy” grid
 - Image from (Gonzalez et al, 2013)



(a)



(b)

See <http://qiao.github.io/PathFinding.js/visual/>

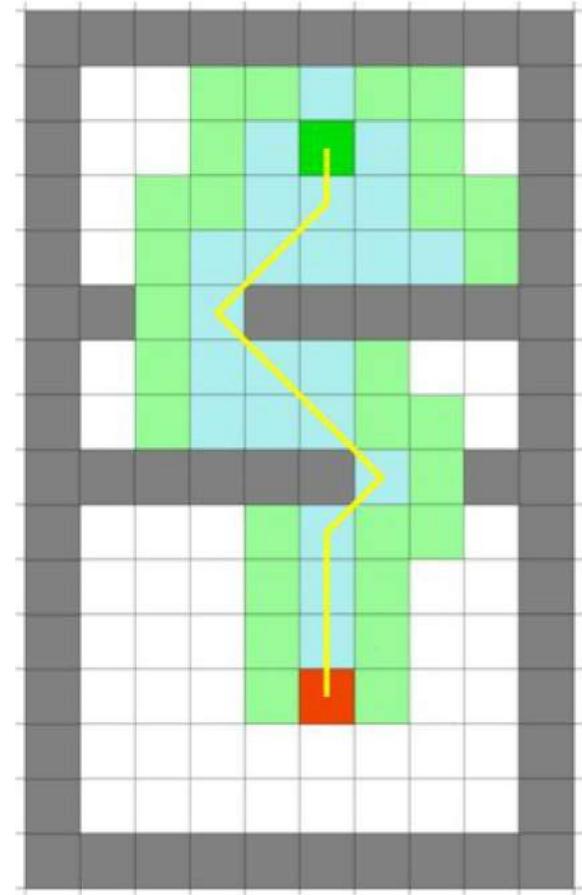
An aside: 3D discretisation (voxels)

- Regular grid in 3D space
- Can use to discretise a 3D space in a similar way as done for 2D spaces

Video: https://www.youtube.com/watch?v=CUT3t-ico5Y&list=PLnS6TQ_QsDUxCrv1fbzPHn0LoTwc_n8NF&index=2

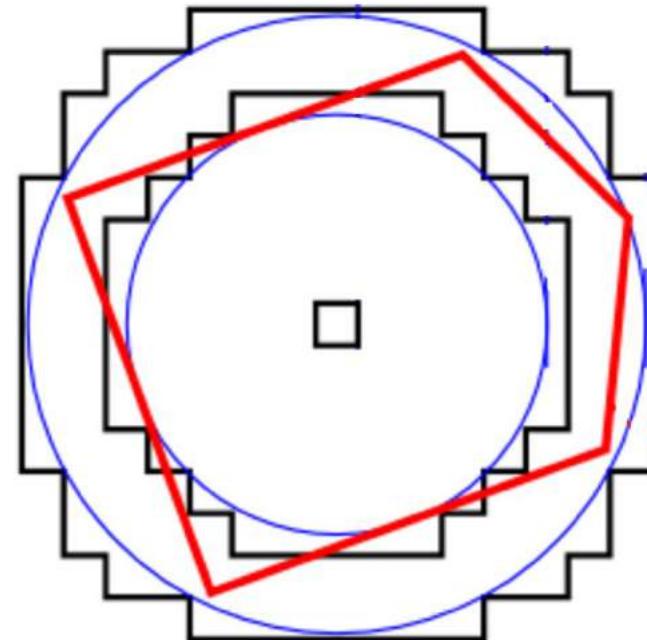
Path Planning

- Have what we need to plan path
 - E.g. Dijkstra
- However, Dijkstra not ideal:
 - Path close to obstacles
 - Next to walls
 - Not good for robots!
- Want to keep our robot safe:
 - If in corridor, drive in middle
 - Keep distance from obstacles

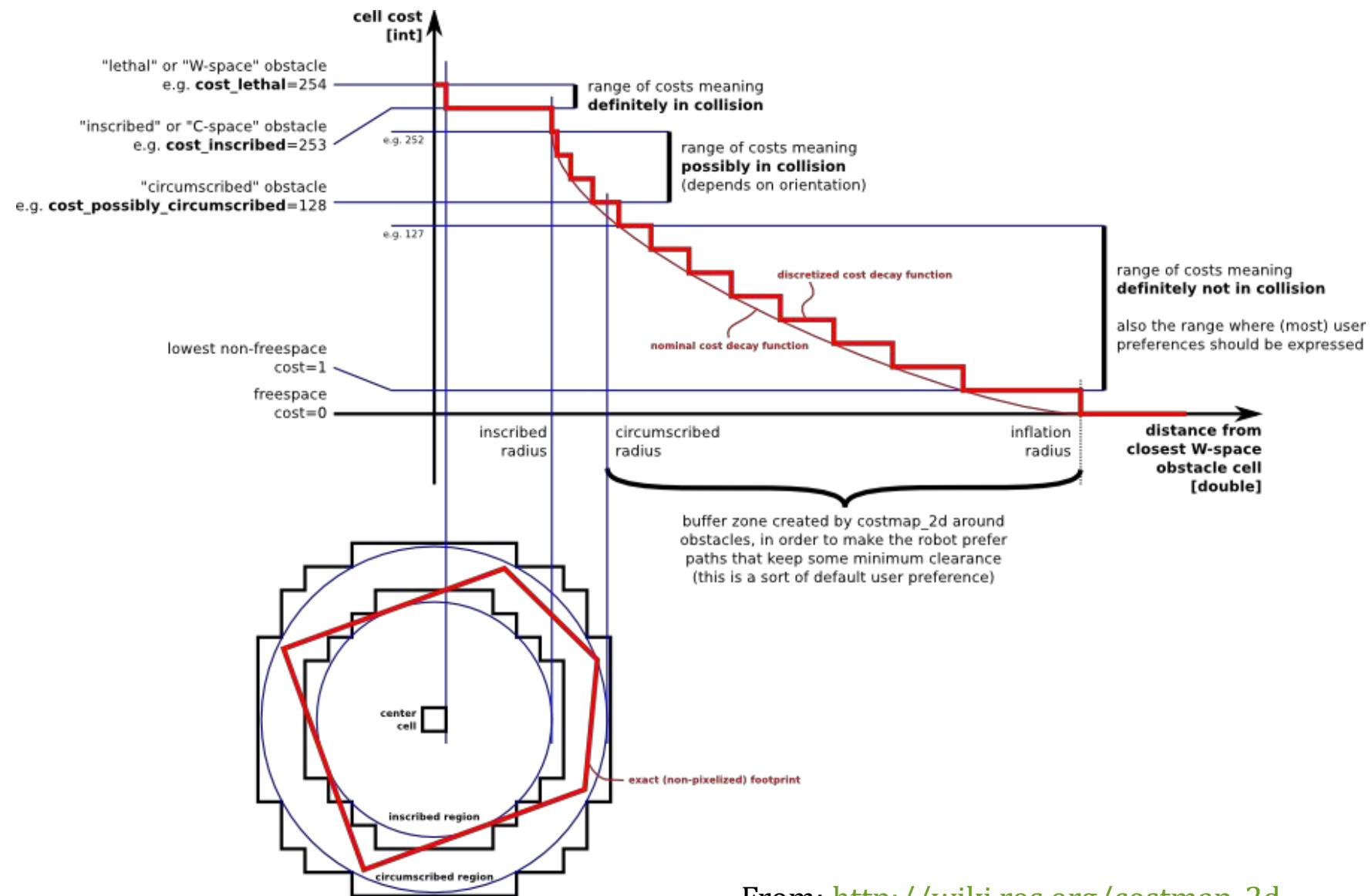


Robot Distances and Costs

- Robot centre point assumed as point of rotation
 - E.g. the turtlebots
- Obstacles are “lethal”
 - In map
 - Sensed by laser
- Lethal obstacles “inflated” based on the robot size
 - Helps determine distance from obstacles



RED – actual robot footprint
OUTER CIRCLE – circumscribed region
INNER CIRCLE – inscribed region

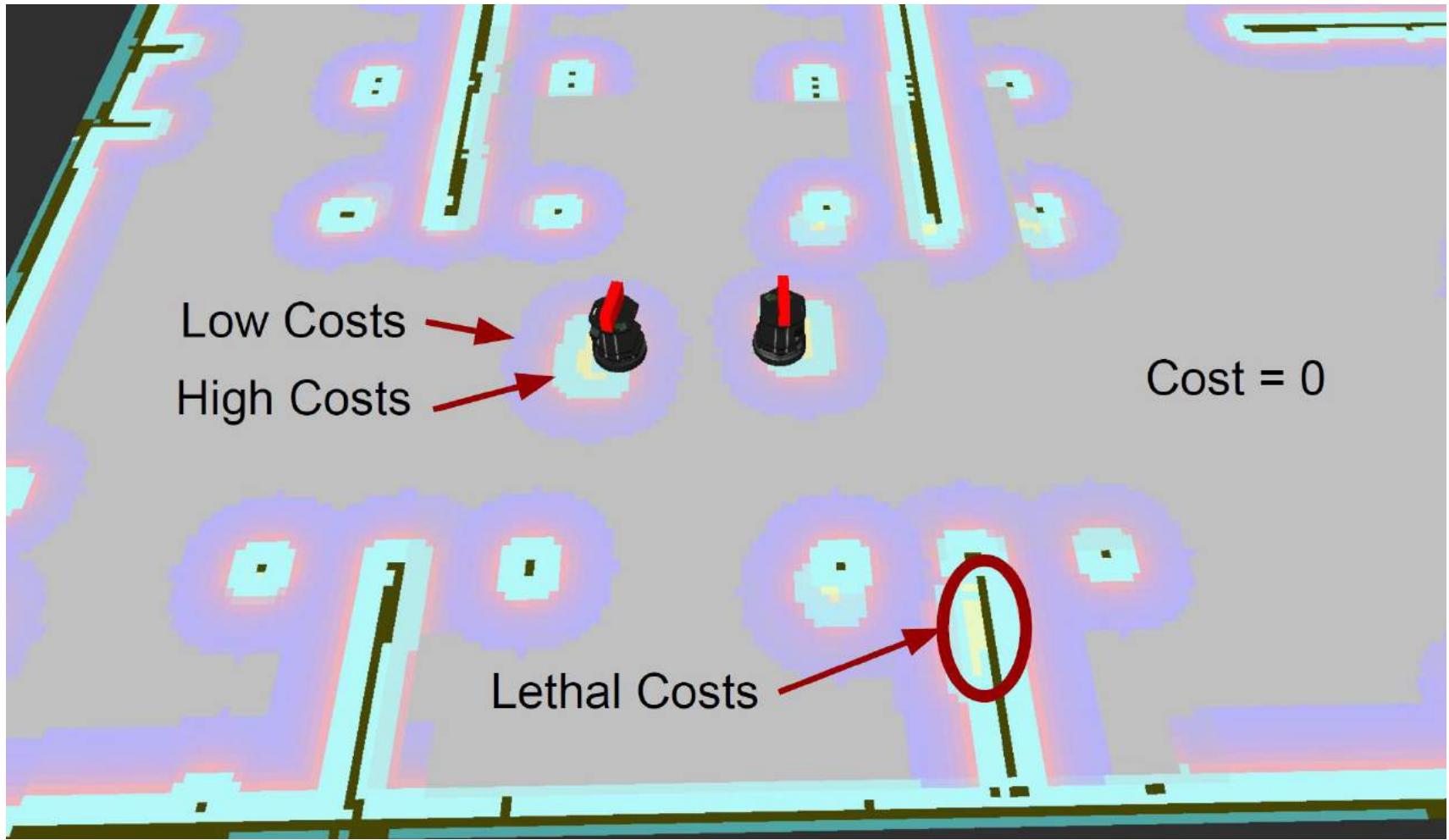


From: http://wiki.ros.org/costmap_2d

Using these costs

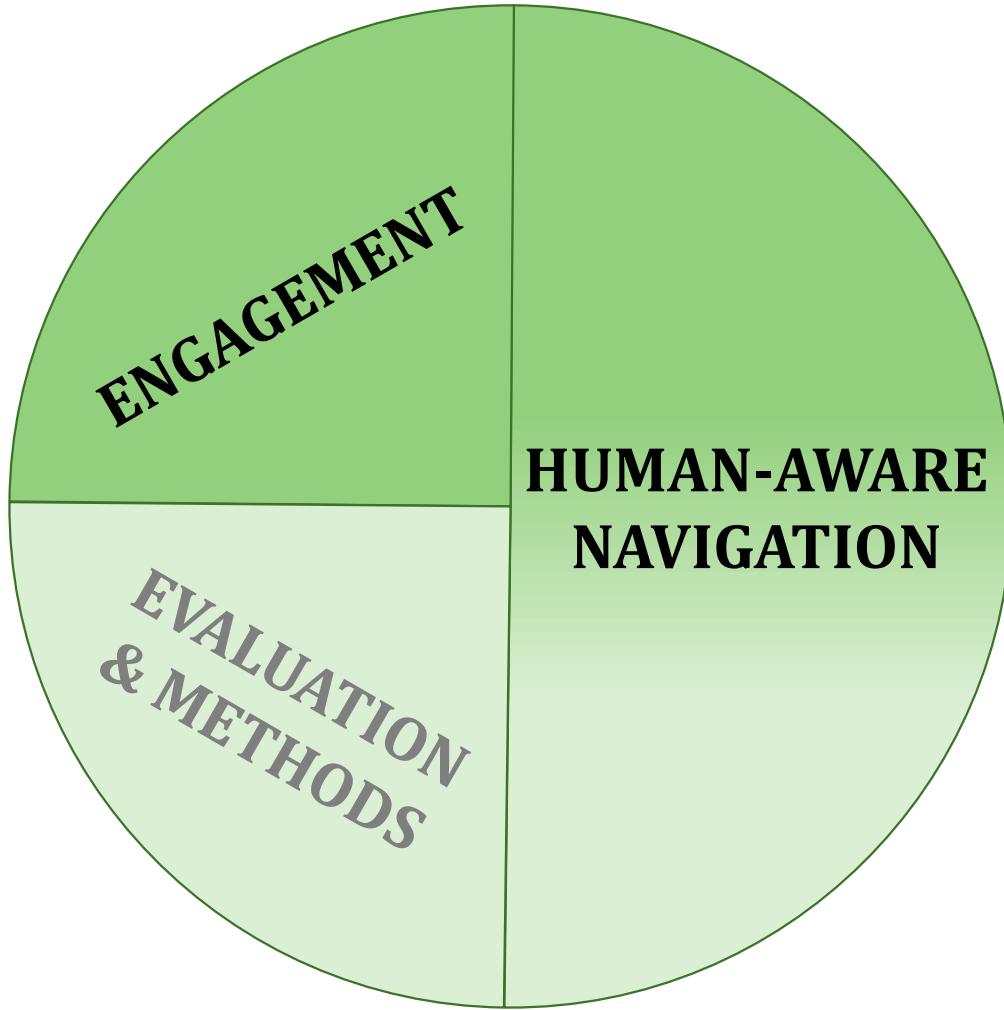
- Costs encoded into the cost-map, for each obstacle
- In Dijkstra example:
 - Using movement as part of the cost (1 for straight, $\sqrt{2} = 1.414$ for diagonal)
 - Now also use the cost of the relevant cells in the costmap to calculate the next neighbour
- Result:
 - For navigation, the robot can stay clear of obstacles, even when using Dijkstra for planning

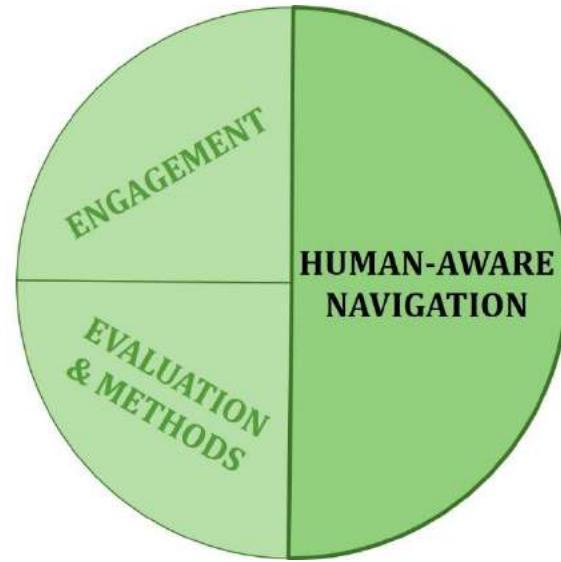
Costmaps



.....

Before the break...





(b) The Human Obstacle

Humans are not just obstacles...

- Human-Robot Spatial Interaction

The study of joint movement of robots and humans through space and the social signals governing these interactions

- Movement of robots and humans
- Focus on social signals

- Human-Aware Navigation

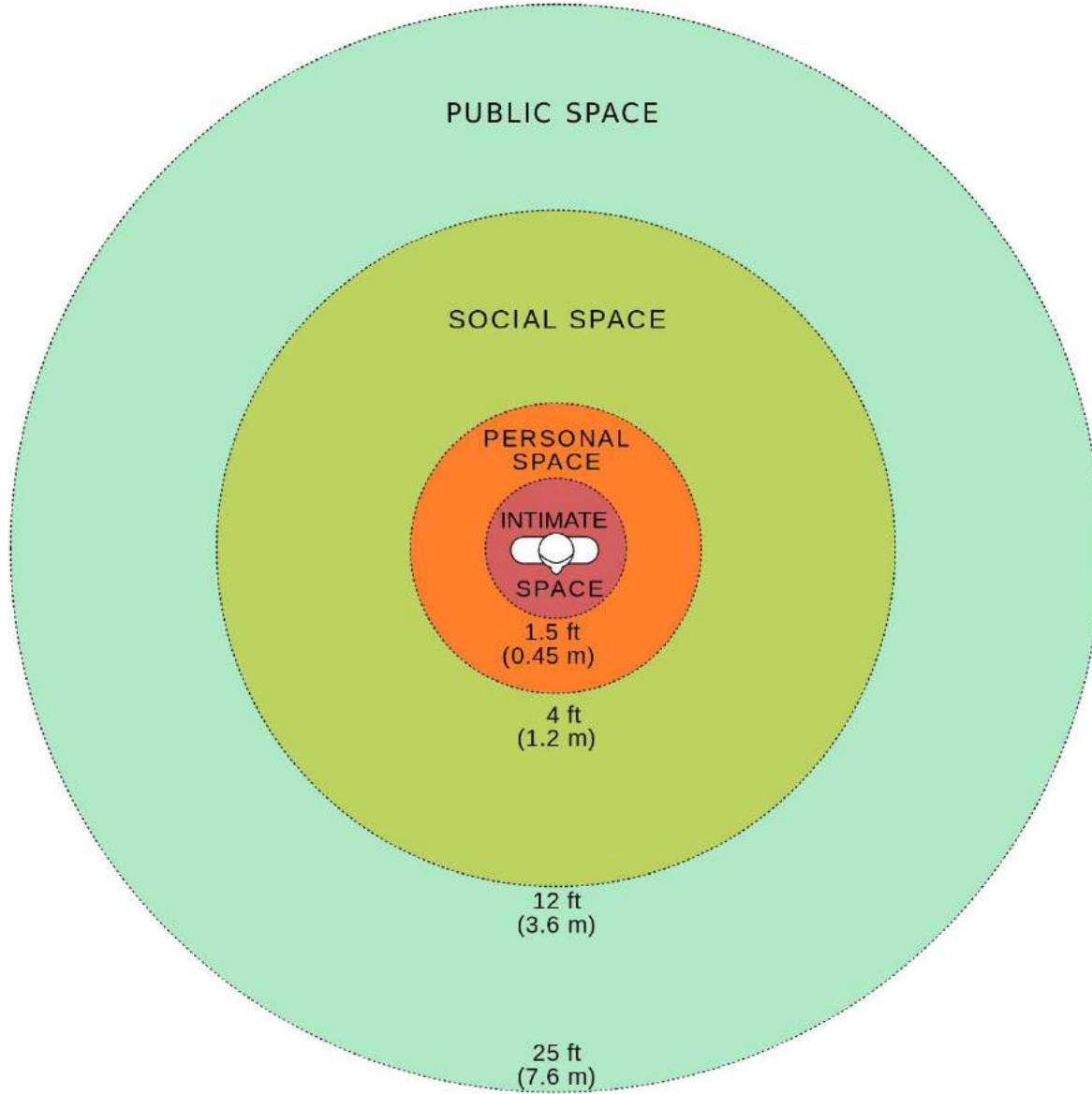
- Specifically taking the human into account
- Recall the three goals: comfort, naturalness, and sociability
- Two main methods:
 1. Stop-and-wait: human does all the hard work
 2. Cost functions based on principles of Proxemics

Proxemics

- A virtual personal space around an individual
 - Edward Hall, 1966
- Divided into four main zones
 - Each zone at a different distance, and with different interaction characteristics
 - Also dependent on relationship
 - Two 'phases' per zone

The four zones:

1. Intimate
2. Personal
3. Social
4. Public



Intimate Space

- 0 – 45cm: Intimacy
- Close Phase (0-15cm)
 - Intimacy, comforting
 - Vision blurred, vocalisations whispered
 - Senses of smell and radiant heat effective
 - Arms can encircle
- Far Phase (15-45cm)
 - Hand can reach and grasp extremities
 - Heads, thighs, and pelvis are not easily brought into contact
 - Able to focus the eye easily, peripheral vision includes the outline of the head and shoulders
 - Heat and odour of the other person's breath might be detected

Personal Space

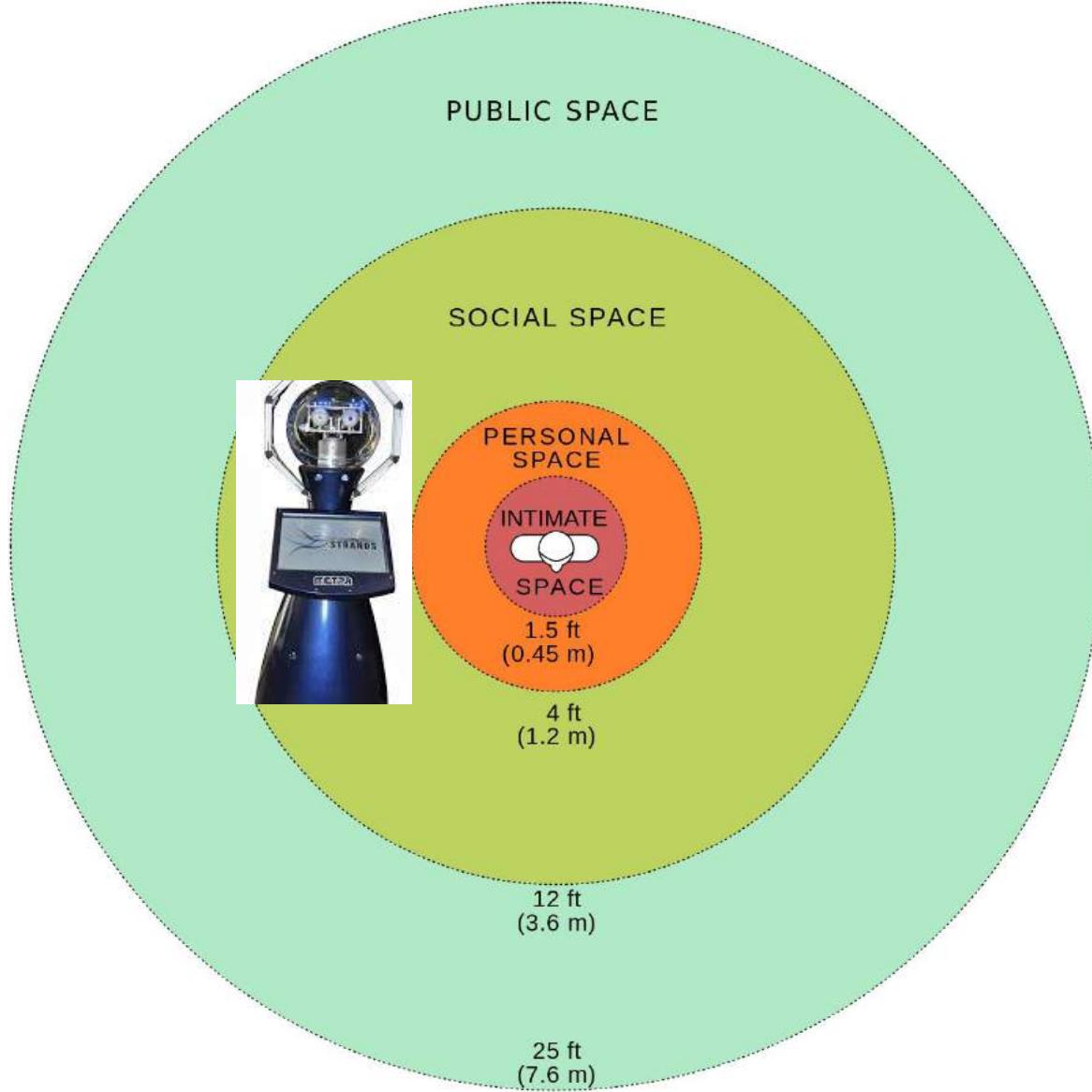
- 45 – 120cm: Family and good friends
- Close Phase (45-75cm)
 - Can grasp other person: peripersonal space
 - No visual distortion of other's features
 - Perception of 3-dimensional qualities of objects
- Far Phase (75-120cm)
 - Outside of grasping distance: at arm's length
 - Other's features clearly visible
 - Moderate voice volume
 - No perception of body heat
 - Lower levels of olfaction

Social Space

- 1.2m – 3.6m: Interactions with acquaintances/strangers
- Close Phase (1.2-2.1m)
 - No touching without special effort
 - Normal voice volume – can be heard from a moderate distance
 - Visual focus extends to nose and parts of both eyes, or, nose, mouth and one eye
- Far Phase (2.1-3.6m)
 - Fine details of face are lost
 - Skin texture, hair, teeth, and condition of clothes readily visible
 - Odour not detectable

Public Space

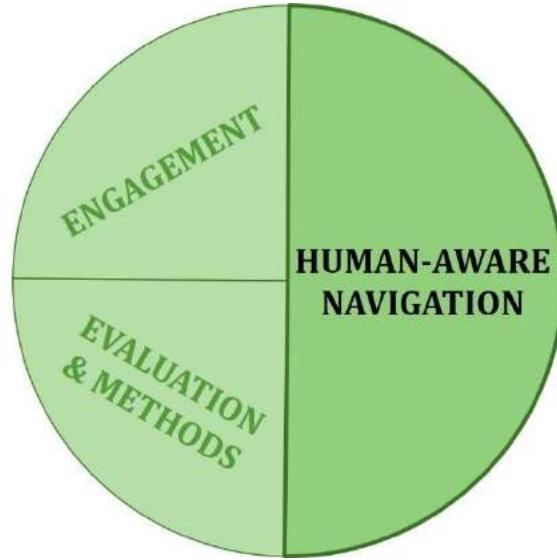
- > 3.6m: public speaking
- Close Phase (3.7-7.6m)
 - Can take evasive actions
 - Voice loud but not full volume
 - Can see whole face, fine details not visible
 - Only whites of eyes visible
- Far Phase (>7.6m)
 - Subtleties of meaning in voice is lost, as are details of facial expressions
 - Vocal, facial, and bodily expression must be exaggerated
 - Foveal vision takes in increasingly more of the other person



Caveats

- Dependent on culture (Hall's studies were in US)
- Equal spacing around individual
 - No difference between front and back
 - No accounting for movement (e.g. warping of space when walking)
- Do not take into account environmental conditions
 - E.g. dim lighting, loud environments, etc
- Enforced violations?
 - E.g. public transport
 - Changes behaviour
- These zones may be applicable to humans, but do they apply to robots?





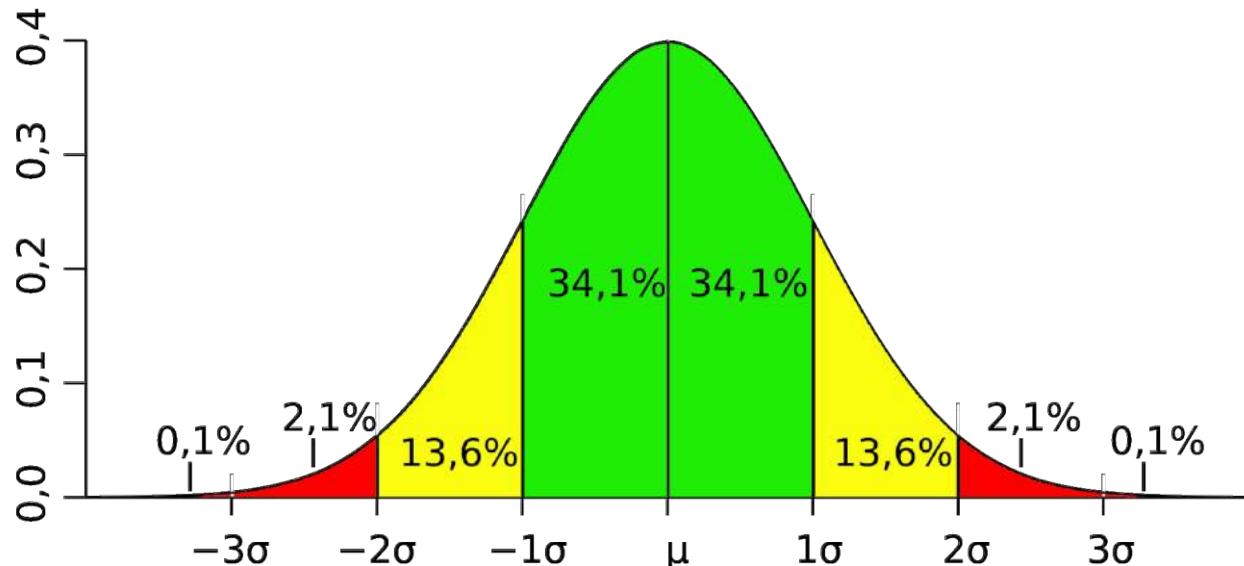
(c) Bringing These Together

Combining Costmaps and Proxemics

- Including costmaps as part of the human representation in the map:
 - Including proxemics as part of this
 - What is the benefit?
- Keeping a greater distance to the human
 - Perceived as safer
 - Reduced stress
 - Even though less “efficient”
- Not necessarily either of the other two goals
 - Doesn’t guarantee more natural behaviour
 - Doesn’t incorporate societal/cultural norms (e.g. drive on the left or the right?)
- How to put Proxemics into Costmap?

Gaussian Distribution and Proxemics

- Sigma – standard deviation (mean of zero)
 - Mean could be position in one dimension (x or y)
- Theoretically infinite, so cut-off at 3-sigma
- Set sigma to size of the Intimate Space
 - This is in only one dimension...
 - For quick visualisation: <http://homepage.stat.uiowa.edu/~mbognar/applets/normal.html>

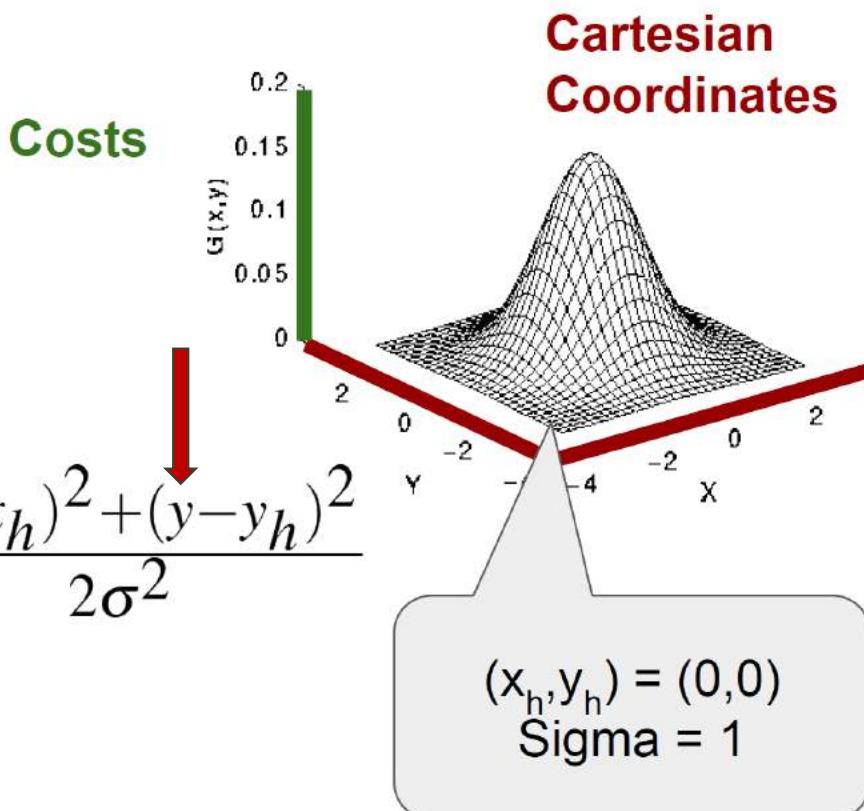


Extending to 2D...

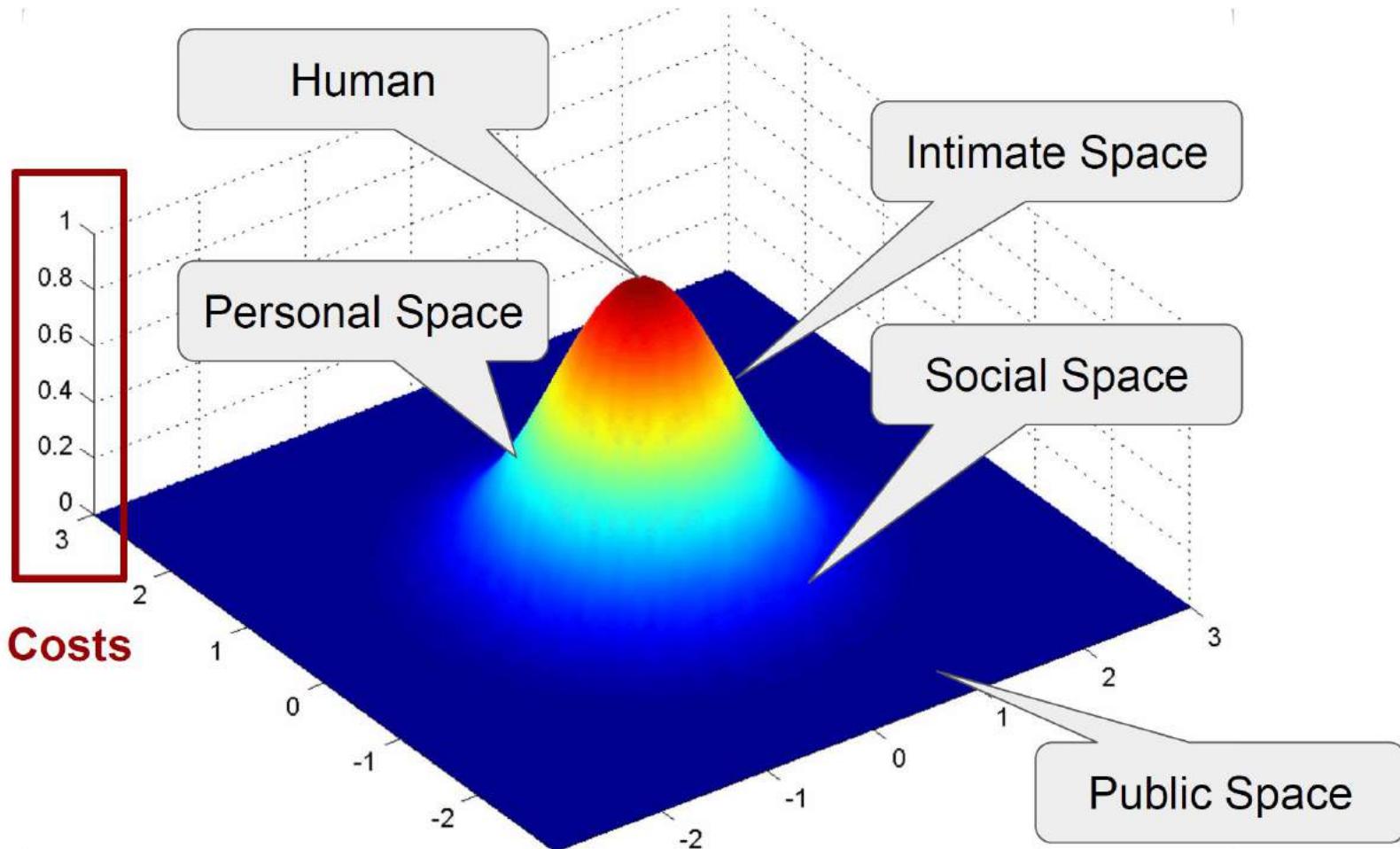
- 2D Gaussian equation – example parameters shown
- Cartesian coordinates centred on the position of the human

Height of the peak

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_h)^2+(y-y_h)^2}{2\sigma^2}}$$

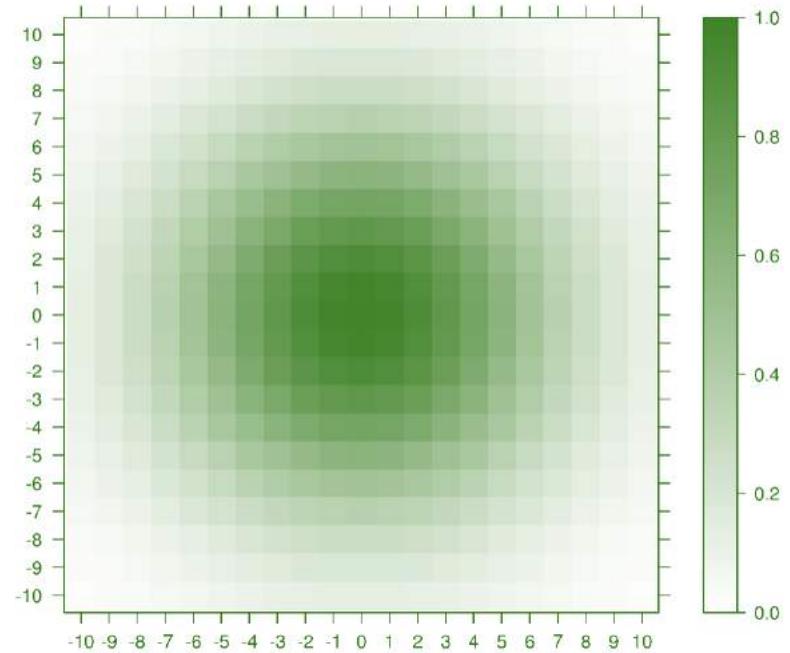
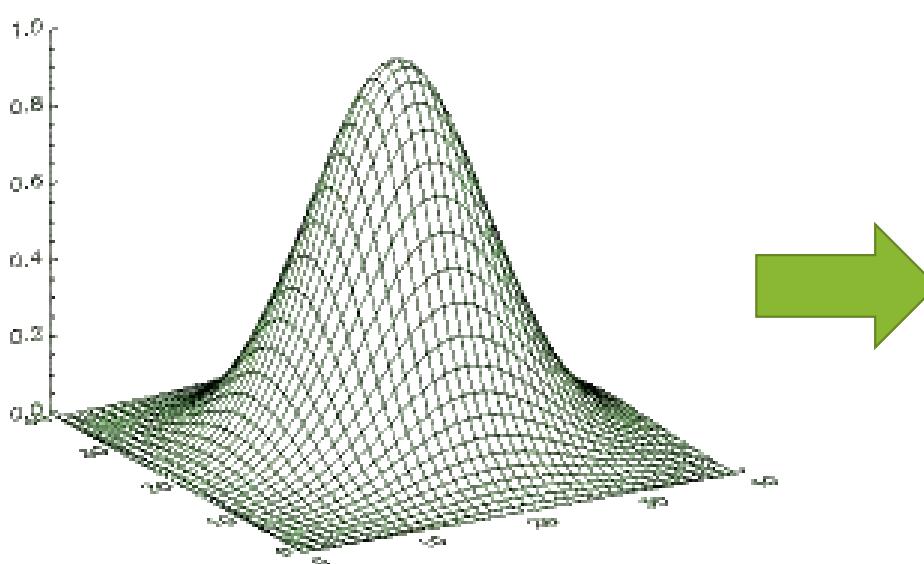


Proxemic Gaussian



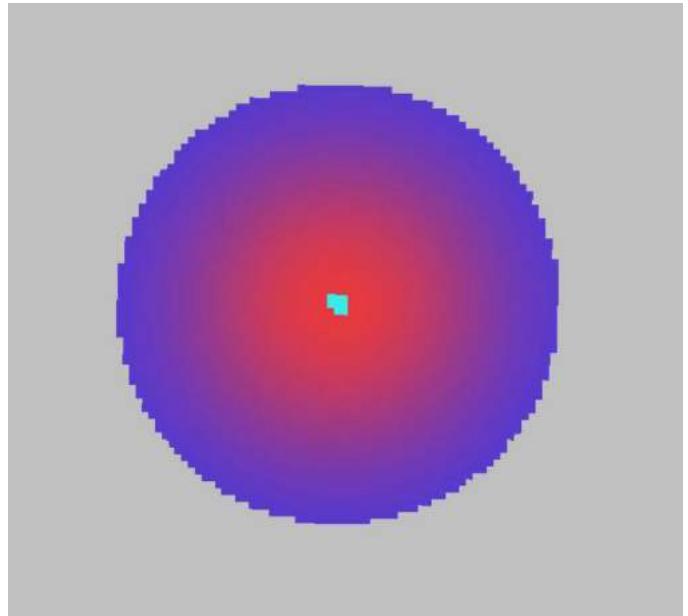
Discretisation

- Have a continuous function, need it to be discretised



And back to Costmaps...

- This information can now be added to the overall costmap, with the other obstacles
 - Or added to a separate layer of the costmap (Lu et al, 2014)
- Path planning in this space as described before
 - Dijkstra



Navigation in Open Space

Full video: <https://www.youtube.com/watch?v=pg7g7qv80MU>

Navigation in Corridor

Full video: <https://www.youtube.com/watch?v=pg7g7qv80MU>

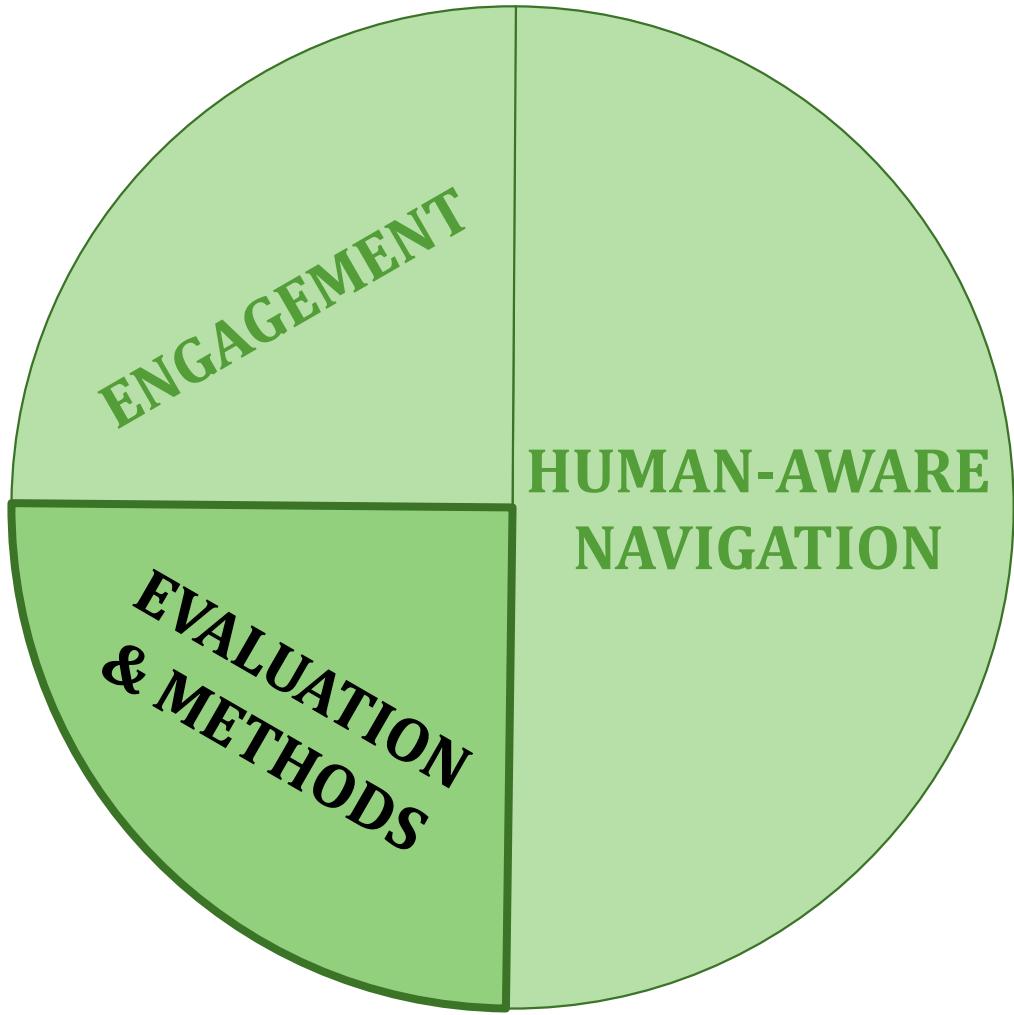
Gaussian Cost functions

Positives

- Straightforward implementation
- Is relevant for all environments
- Takes into account proxemics
- Ensures interaction is safe, and perceived to be as such

Negatives

- Only influences distances
- Sociability and naturalness not guaranteed
- Does not take into account social context
 - Only based on proxemics
 - Could drive through groups?
- Works with Dijkstra
 - Slow/inefficient



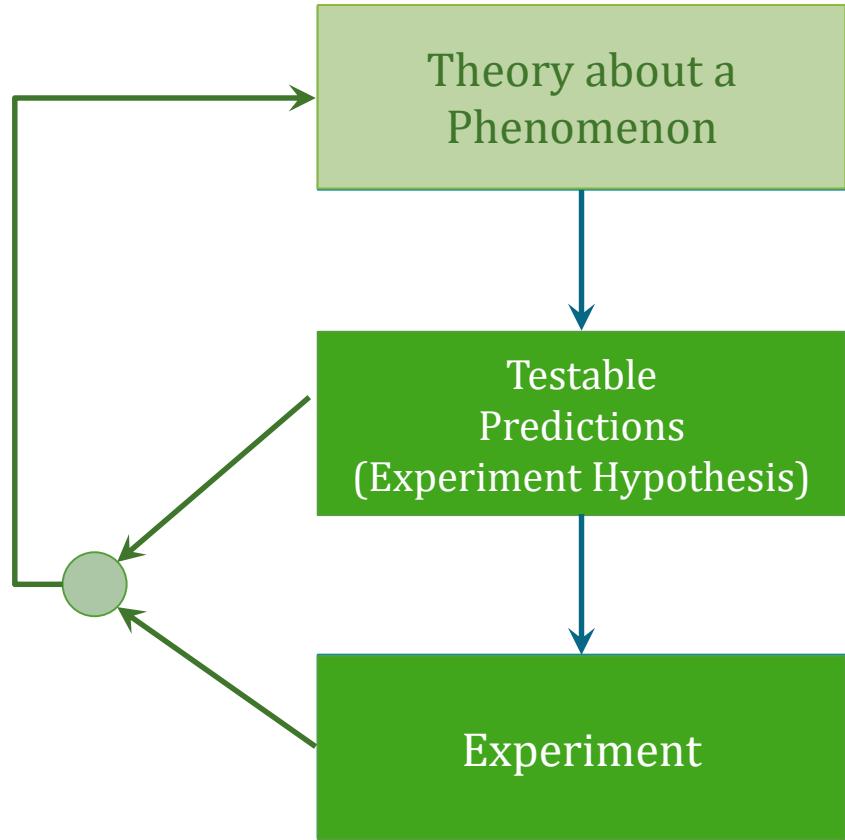
Is my Robot Successful?

- You have implemented a complete robotic system – how do you know if it is successful?
 - Performance metrics (remember your assignment!)
 - Test with real robots/simulation, etc
- But what if it is an HRI system?
 - Humans are problems...
 - Non-predictable, they come with prior expectations/experience
- Need to evaluate your system with people...
 - Running experiments
 - To verify that the robot has an effect (or is perceived) in the way you (as a designer) intended

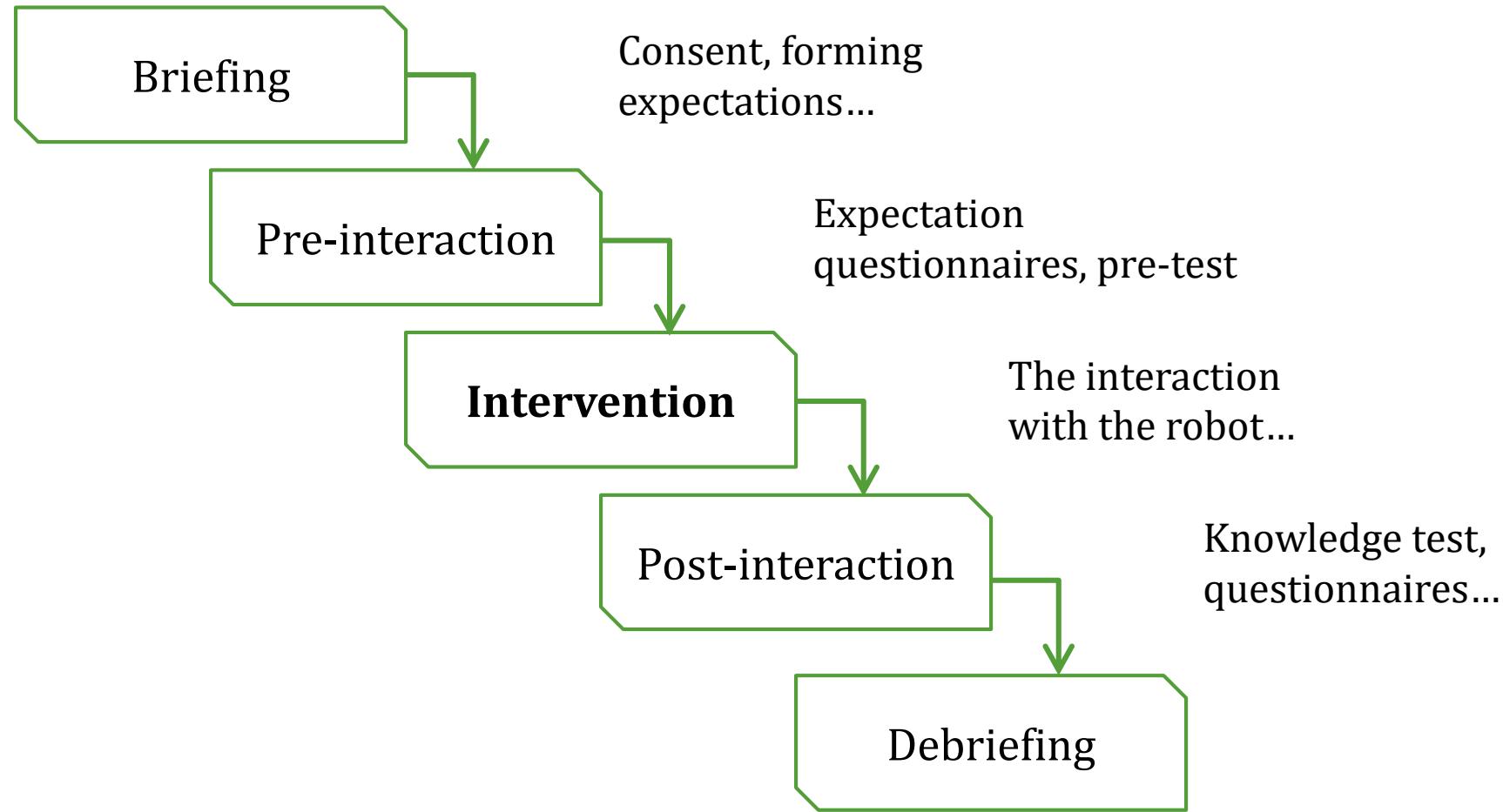


The Scientific Method

- The “classic” view of the scientific method
 - Generate testable predictions from a theory
 - Perform experiment specifically generated to address the hypothesis
 - Assess the experimental hypotheses: observations in context of the theory
 - Update/refine the founding theory as necessary
- Does not deny role for exploratory studies (later)

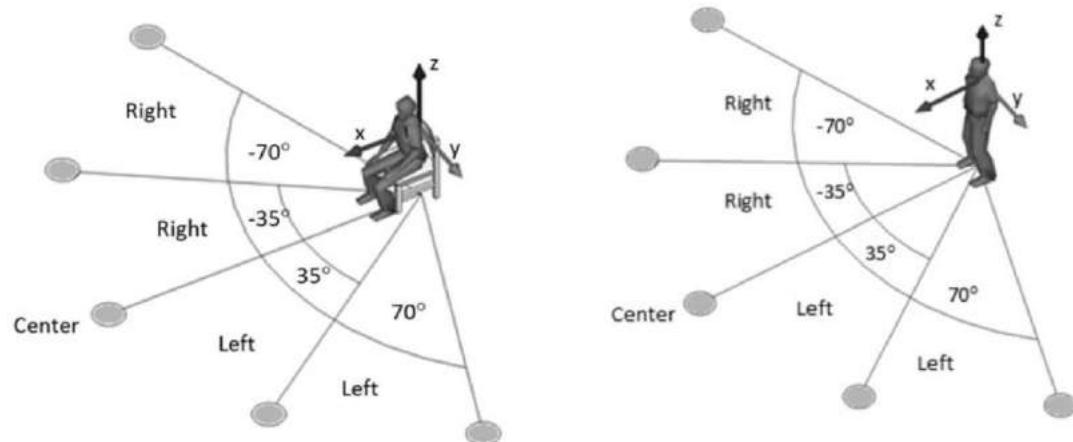


The structure of an experiment...



Example HRI Experiment (1)

- Study by (Torta et al, 2013): Design of a parametric model of personal space for robotic social navigation
- Small humanoid (Nao) approached human participants in one of two conditions: sitting or standing
 - Approach from a range of angles: humans would stop it when they wanted to
- Indication that the HRI distance is longer than would be expected in HHI
 - Also see (Walters et al, 2009)
- Effect of approach angle, and whether standing or sitting



Example HRI Experiment (2)

- Study by (Beck et al, 2010): exploring how body posture of a Nao robot relates to the interpretation of emotion
 - Displaying “key poses” and assessing interpretation
- Research question:

Is the interpretation of the key poses displayed consistent with their positions in the Affect Space?
- Participants better than chance at interpreting the five key poses
- Some blending was also possible, though this made it more difficult to identify

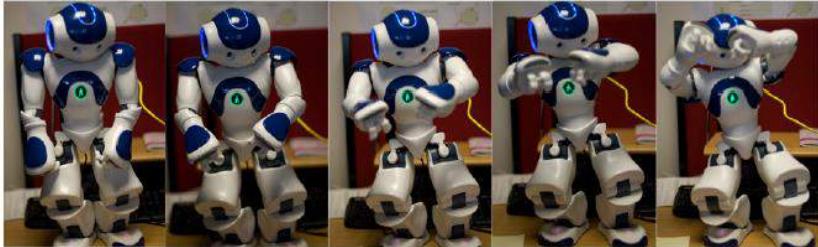


Figure 3: Five Key poses generated by the system (A: 100% Sadness. B: 70% Sadness 30% Fear. C: 50% Sadness 50% Fear. D: 30% Sadness 70% Fear. E: 100% Fear).

Table 2: Postures and their main interpretations.

“None” indicates that the question was left unanswered.

Posture	Most common Primary Interpretation (PI)	Most common Secondary Interpretation (SI)
100% Sadness	Sadness (74%)	None (70%)
50% Sadness 50% Neutral	Neutral (52%)	None (70%)
70% Sadness 30% Pride	Sadness (61%)	None (70%)
50% Sadness 50% Pride	Neutral (52%)	None (52%)

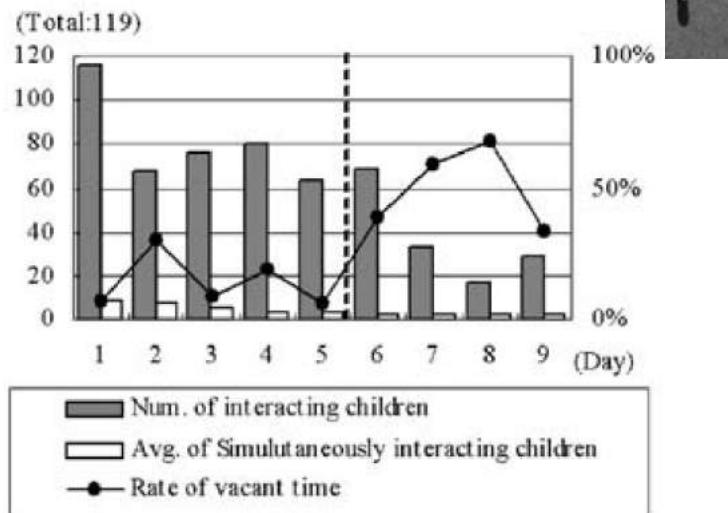
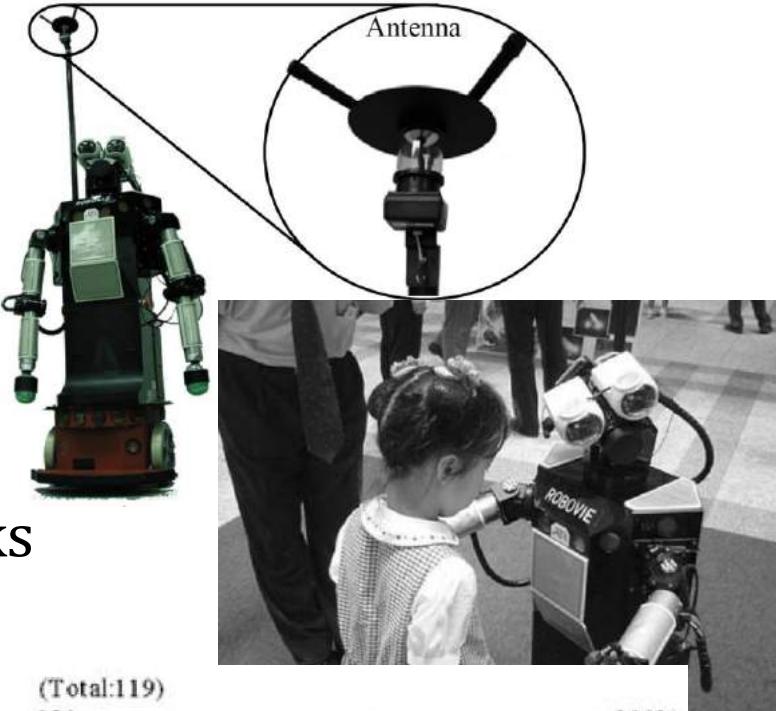
Pilot studies

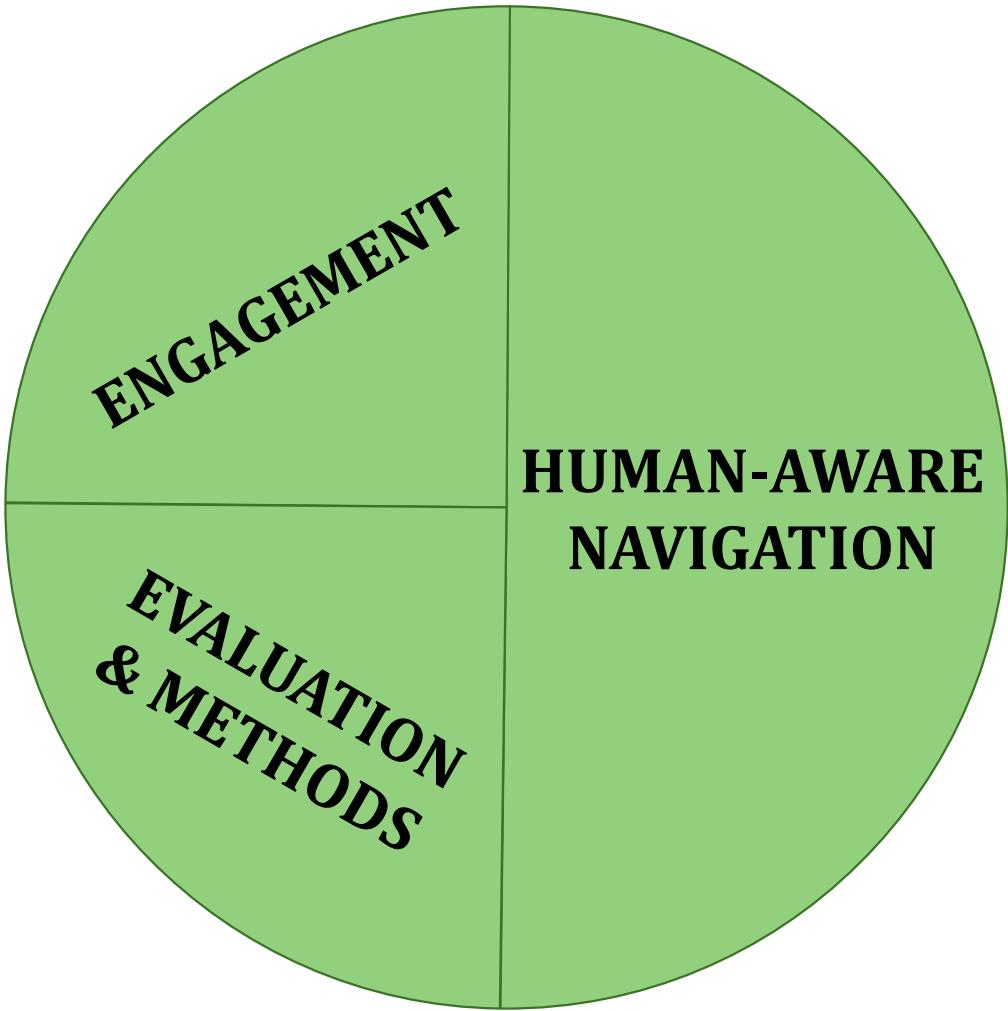
... or *Exploratory Studies*

- Can learn from studies that don't formally fit within the scientific method outlined:
 - Studies with no hypothesis (though would still have some expectations of what will/could happen – if only for ethics)
 - Perhaps a hypothesis, but only a single experimental condition
- Some degree of control and rigour are still required to draw meaningful observations (e.g. Kanda et al, 2007)
 - Reduce the incidence of confounds, or even discover what the confounds are
 - Finding data from which hypotheses can be formed

Example Pilot Study

- Field trial:
 - (Kanda et al, 2004)
 - Two weeks in a corridor speaking English with Japanese children
 - Watch to see what happens
- Various pre/post expt checks
 - E.g. English level etc
- No explicit hypothesis, one experimental condition
 - Could examine rates of interaction over time
 - And a look at learning...





Reminder: Marking Sessions

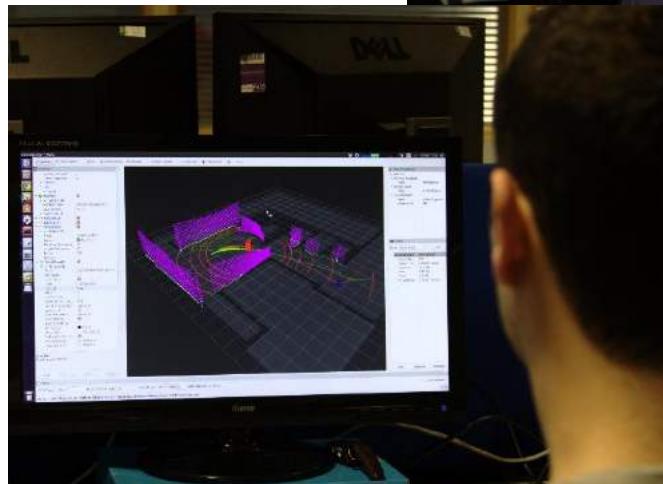
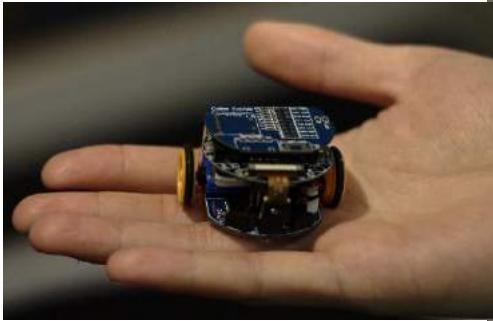
- Continue this Friday 29th and continue in timetabled workshop sessions until next Monday 8th April
- **See Blackboard (and email) announcement on 20th March for details**
- Reminder:
 - You must attend the time slot you have been allocated to
 - 5 minutes to run simulation (continuous, may restart within this): you will start this
 - Deliver presentation – may be useful to do in parallel with the simulation
 - We will ask questions/explore your code and reasoning
 - We will have all submitted code/presentations ready
 - Shared register: only attend the session you are assigned to

Workshops after marking sessions

- Friday/Monday 12th/29th April
- Workshop activity related to lectures both last week and today: mobile robot behaviours relevant to human-robot interaction
 - Part paper-based exercise: working from an academic paper
 - Part implementation: starting from the workshop code and assignment simulation environment
 - Towards preparation for the exam
- Academic paper on Blackboard in workshop materials for week 11:
 - Will return to this paper in next week's lecture
 - Recommend you at least take a look at the paper before the workshop...

Next Lecture...

- Applications
- Focus on the activities of L-CAS
- What the future holds
... and a bit more ethics



References / Reading

- Baxter, P. et al., 2014. Tracking gaze over time in HRI as a proxy for engagement and attribution of social agency. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction - HRI '14*. Bielefeld, Germany: ACM Press, pp. 126–127.
- Beck, A., Hiolle, A., Mazel, A., & Cañamero, L. (2010). Interpretation of emotional body language displayed by robots. *Proceedings of the 3rd International Workshop on Affective Interaction in Natural Environments - AFFINE '10*, 37.
- Glas, N. & Pelachaud, C., 2015. Definitions of Engagement in Human-Agent Interaction. In *International Conference on Affective Computing and Intelligent Interaction (ACII)*. Xi'an, China: IEEE Press, pp. 944–949.
- Gonzalez-Arjona, D. et al., 2013. Simplified Occupancy Grid Indoor Mapping Optimized for Low-Cost Robots. *ISPRS Int. J. Geo-Information*, 2, pp.959–977.
- Hall, E., 1966. The Hidden Dimension, Anchor Books
- Kanda, T. et al., 2004. Interactive Robots as Social Partners and Peer Tutors for Children: A Field Trial. *Human-Computer Interaction*, 19(1), pp.61–84.
- Kruse, T. et al., 2013. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12), pp.1726–1743.
- Lemaignan, S. et al., 2016. From real-time attention assessment to “with-me-ness” in human-robot interaction. In *ACM/IEEE International Conference on Human-Robot Interaction*. Christchurch, New Zealand.
- Lu, D. V, Hershberger, D. & Smart, W.D., 2014. Layered Costmaps for Context-Sensitive Navigation. In *IROS 2014*. Chicago, USA: IEEE, pp. 709–715.
- Sidner, C.L. et al., 2005. Explorations in engagement for humans and robots. *Artificial Intelligence*, 166(1–2), pp.140–164.
- Tanaka, F., Cicourel, A., & Movellan, J. R. (2007), “Socialization between toddlers and robots at an early childhood education center”, *PNAS*, 102(46), 17954–17958.
- Torta, E., Cuijpers, R.H. & Juola, J.F., 2013. Design of a Parametric Model of Personal Space for Robotic Social Navigation. *International Journal of Social Robotics*, 5(3), pp.357–365.
- Walters , M L , et al, 2009 , 'An empirical framework for human-robot proxemics' in *Procs of New Frontiers in Human-Robot Interaction : symposium at the AISB09 convention* . pp. 144-149 .

CMP3101M AMR – Week 12

Applications

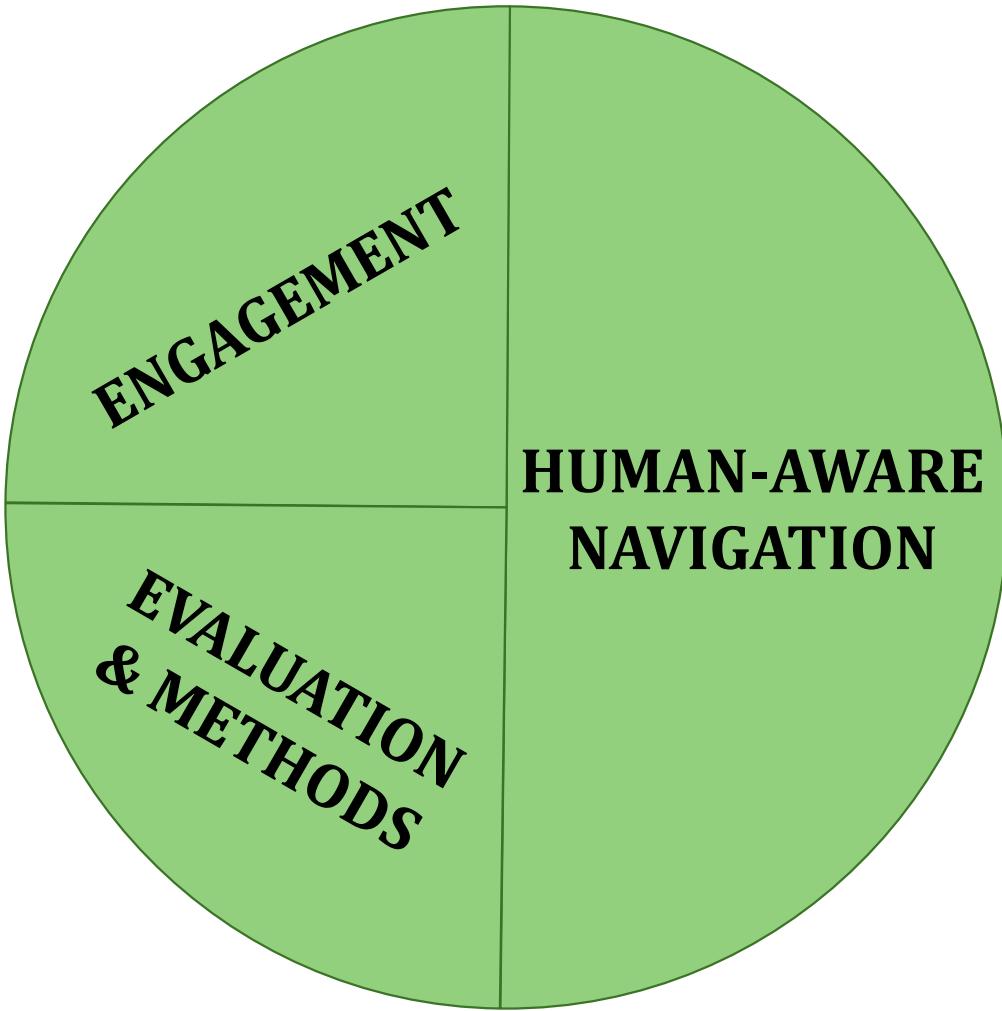
Dr. Paul Baxter

Location: INB3119

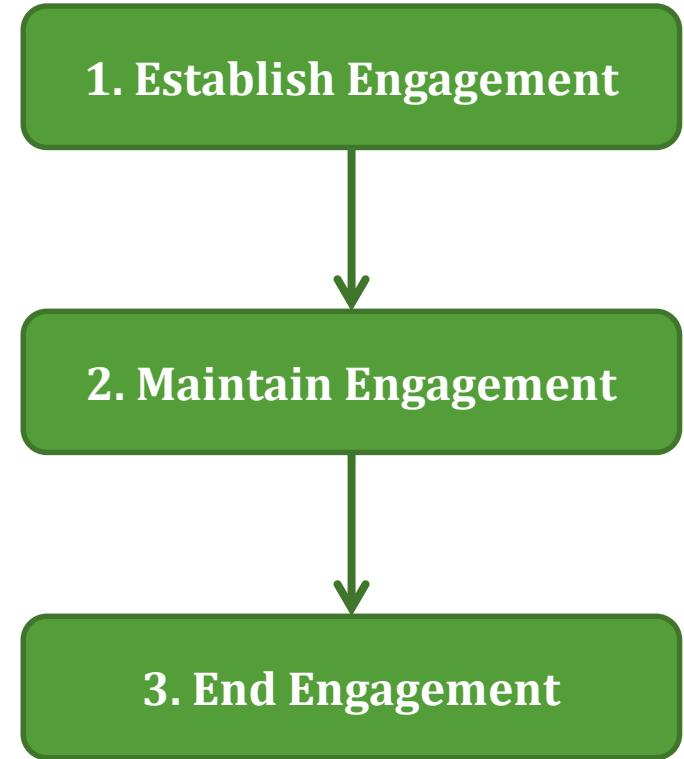
pbaxter@lincoln.ac.uk

Office hours: Tuesdays 9am-11am

Last Week: HRI 2



Stages in Engagement



Establishing Engagement

Social Strategies

- Speech
 - Saying something to someone
- Gesture
 - Only useful if have limbs...
 - Waving, etc
- Behaviour
 - Could engage in attention seeking behaviour
- Physical Interaction
 - Equivalent of a tap on the shoulder... (clearly safety implications)

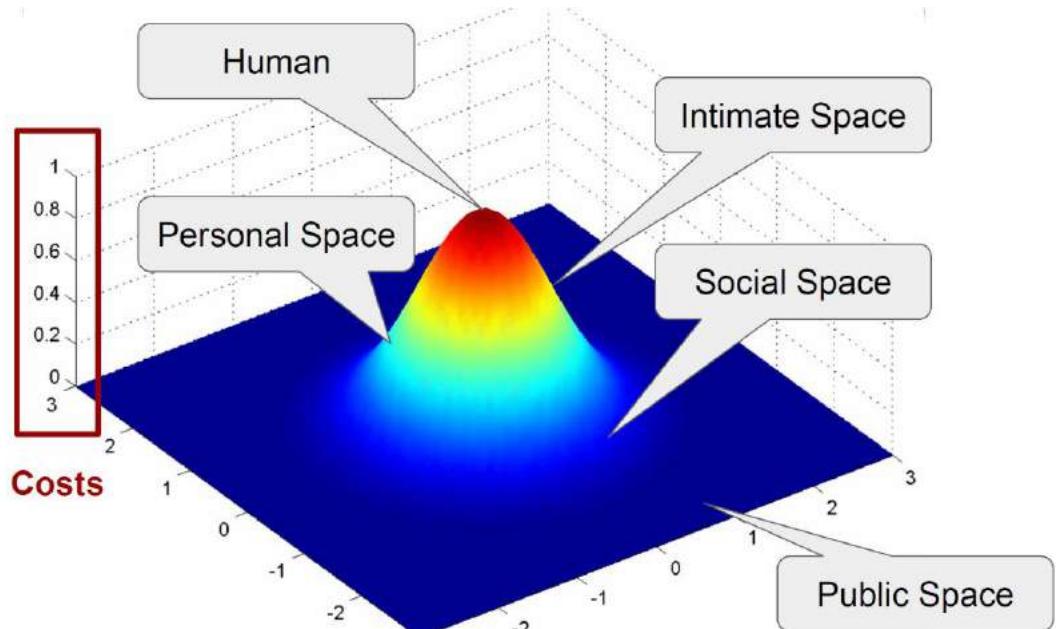
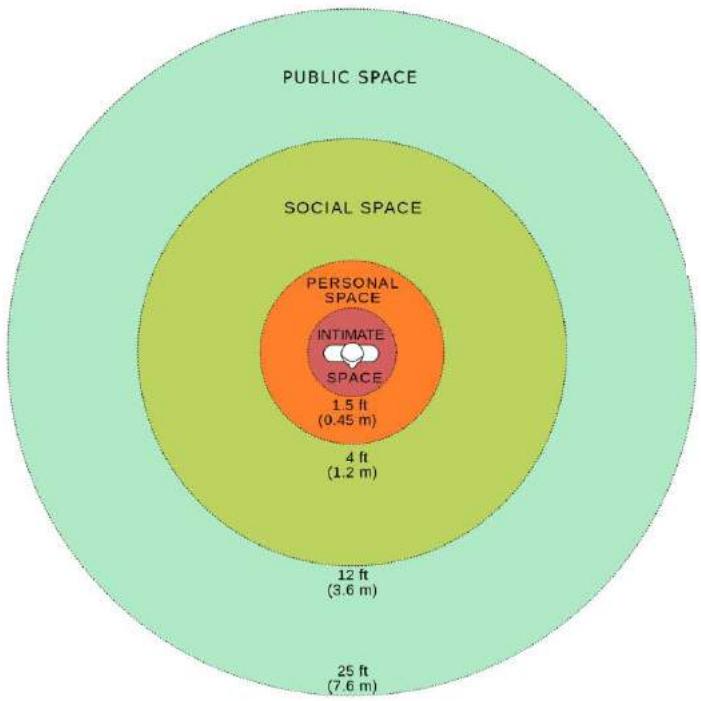
Non-Social Strategies

- Sound
 - Pre-recorded speech
 - Alarm / beeps / motor noise
 - White noise
- Vision
 - Flashing lights
- Behaviour
 - Bumping into human (not advisable...)

Video from:

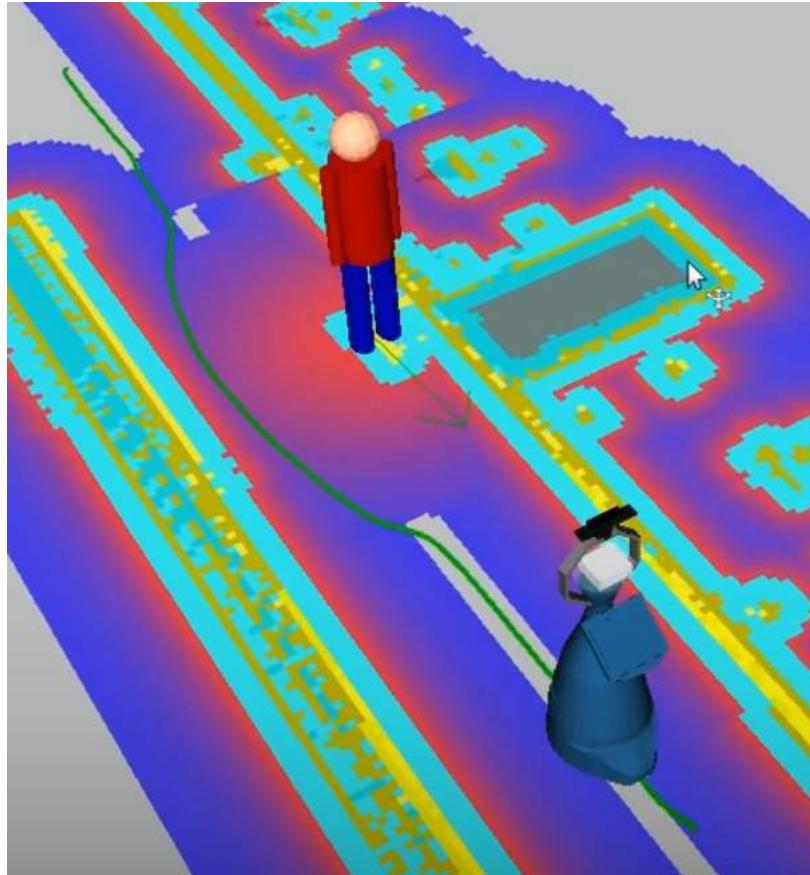
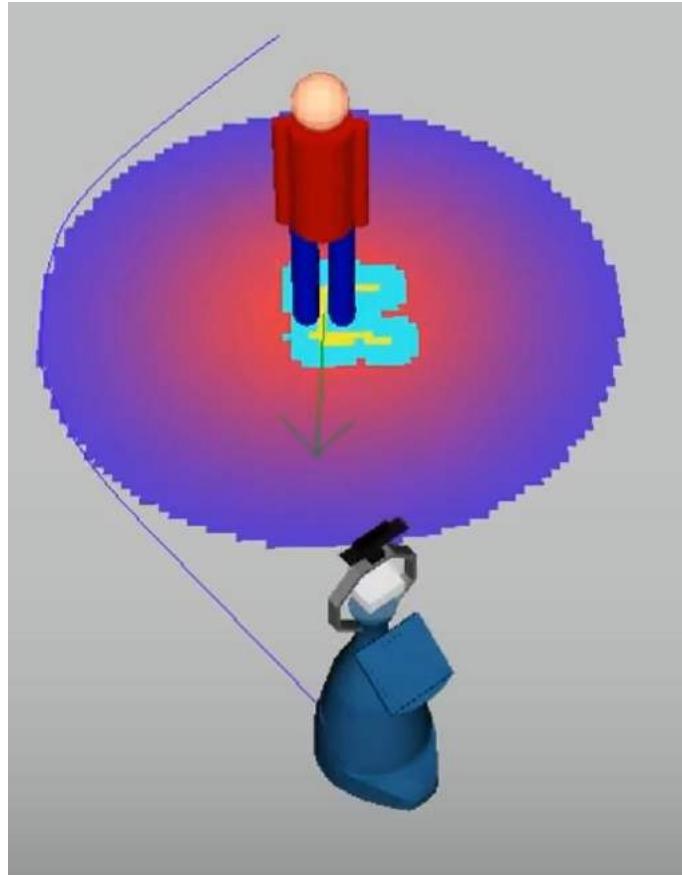
<https://www.youtube.com/watch?v=asHaWo04mIo>

Proxemics and Robot Model



$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_h)^2 + (y-y_h)^2}{2\sigma^2}}$$

Integration into Costmaps

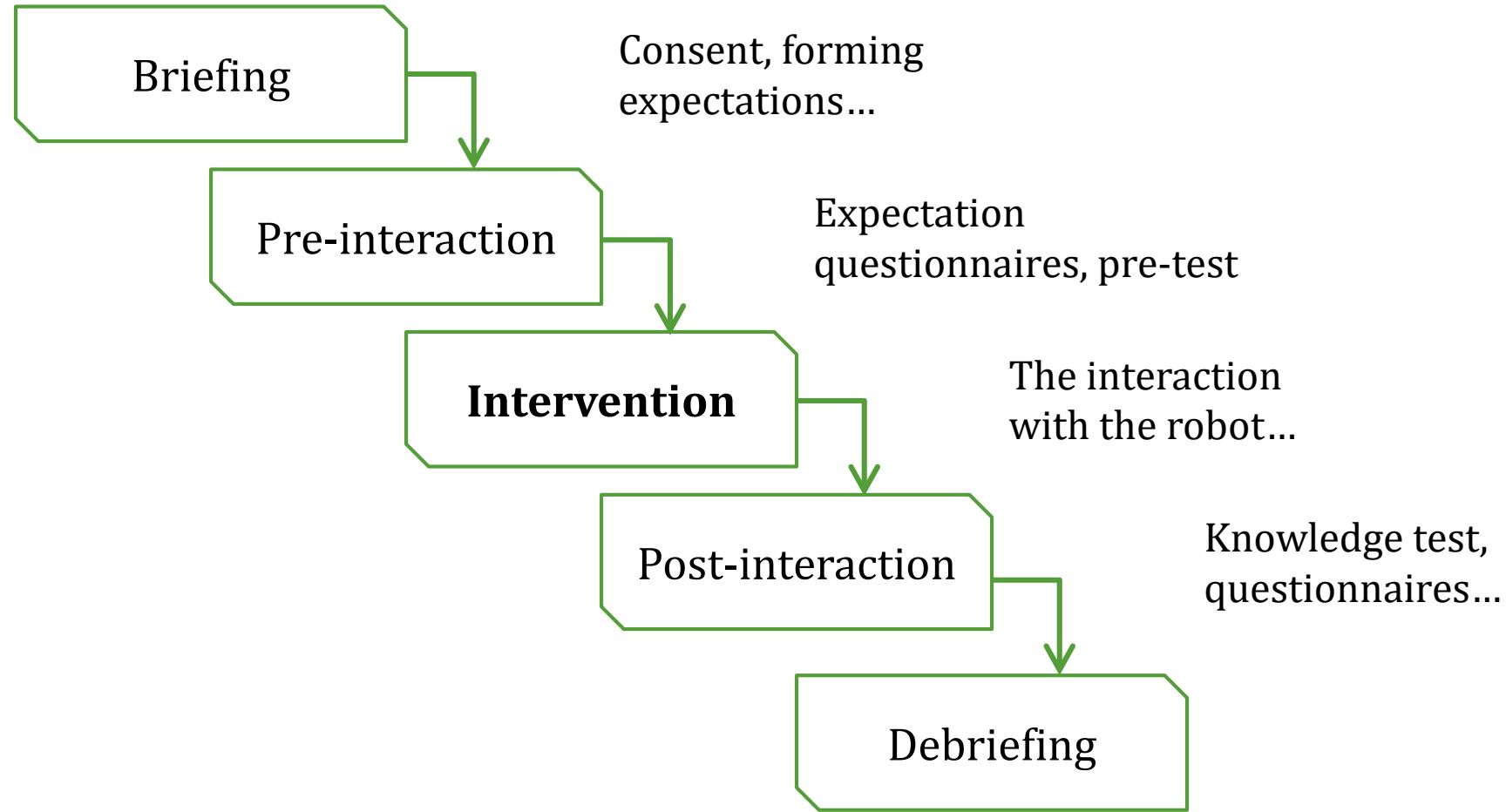


Is my Robot Successful?

- You have implemented a complete robotic system – how do you know if it is successful?
 - Performance metrics (remember your assignment!)
 - Test with real robots/simulation, etc
- But what if it is an HRI system?
 - Humans are problems...
 - Non-predictable, they come with prior expectations/experience
- Need to evaluate your system with people...
 - Running experiments
 - To verify that the robot has an effect (or is perceived) in the way you (as a designer) intended



The structure of an experiment...

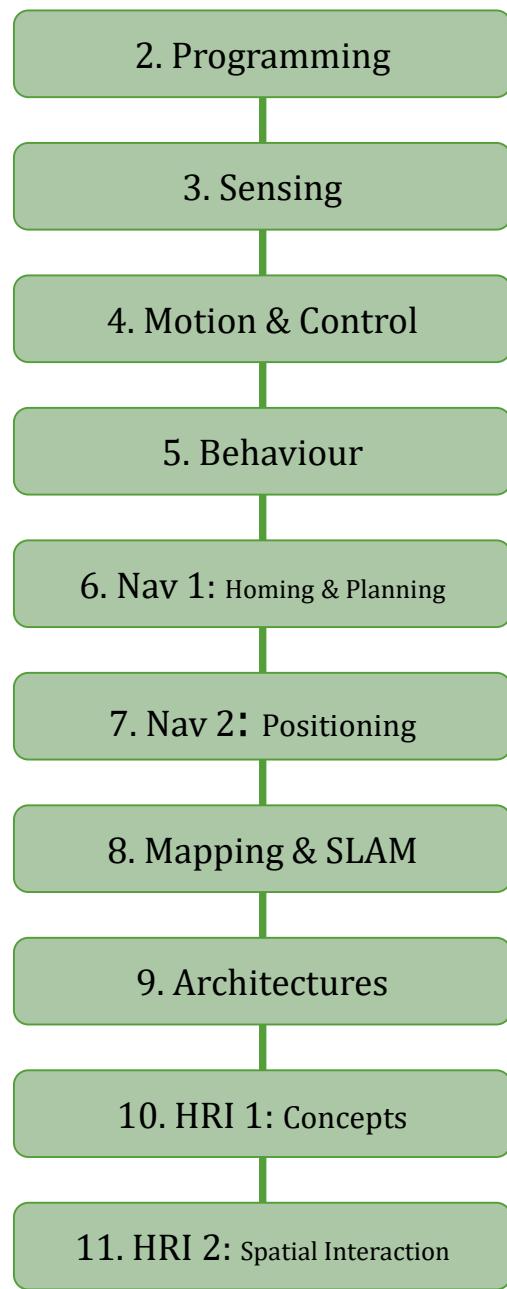


Syllabus review

Week	Topic	Lecturer
1	Introduction	Marc Hanheide
2	Robot Programming (ROS)	Marc Hanheide
3	Robot Sensing	Marc Hanheide
4	Motion & Control	Marc Hanheide
5	Robot Behaviour	Ayse Kucukyilmaz
6	Navigation 1	Ayse Kucukyilmaz
7	Navigation 2	Ayse Kucukyilmaz
8	Robot mapping & SLAM	Ayse Kucukyilmaz
9	Control Architectures	Paul Baxter
10	Human-Robot Interaction 1	Paul Baxter
11	Human-Robot Interaction 2	Paul Baxter
12	Applications	Paul Baxter

Today

- Purpose is to give you an overview of Robotics Applications, using examples of real Research and Systems
- Research in L-CAS will be used as the main examples
 - Recent/current research by your SoCS lecturers and demonstrators
 - Possible opportunities for you to be involved in this work as a postgraduate (especially the new MSc in Robotics and Autonomous Systems)...
- L-CAS research covers a wide range of topics, most of which is related in some way to the content of this module



Lincoln Centre for Autonomous Systems

- L-CAS specialises in:
 - perception, learning, decision-making, control and interaction for autonomous systems, such as robots,
 - the integration of these capabilities in application domains including agri-food, healthcare, intra-logistics, intelligent transportation, personal robotics and security
- 12 academic staff (or parts thereof...), and 30+ postdocs and research students
- <https://lcas.lincoln.ac.uk/>



L-CAS staff members

- [Prof Tom Duckett](#)
- [Prof Shigang Yue](#)
- [Prof Marc Hanheide](#)
- [Prof Gerhard Neumann](#)
- [Prof Pal From*](#)
- [Dr Nicola Bellotto](#)
- [Dr Paul Baxter](#)
- [Dr Grzegorz Cielniak](#)
- [Dr Heriberto Cuayahuitl](#)
- [Dr Charles Fox](#)
- [Dr Amir Ghalamzan Esfahani](#)
- [Dr Ayse Kucukyilmaz](#)

...and of course your friendly demonstrators!

See <https://lcas.lincoln.ac.uk/wp/people/> for full list



L-CAS Research Themes

AFT

Agri-Food Technology

BES

Bio-Inspired Embedded Systems

HCR

Human-Centred Robotics and Assistive Technology

LAS

Learning for Autonomous Systems

See L-CAS website: <https://lcas.lincoln.ac.uk/wp/projects>



AFT

BES

HCR

LAS

Agri-Food Technology

Combining Agriculture, Food, Autonomous Systems, Robotics, Logistics, etc...



Detection: sensing broccoli

- In-field and real-time detection of broccoli heads for automated monitoring and harvesting
 - Various imaging and classification methods used
 - Field trials literally in a field
- See video:
https://www.youtube.com/watch?v=MUTddzcWERs&list=PLnS6TQ_QsDUXCrv1fbzPHn0LoTwc_n8NF&index=13



Detection: Potato Anomalies

- Automatically detecting anomalies on food given human expert knowledge
 - Example of potatoes, and vision-based detection
 - See: https://www.youtube.com/watch?v=v8YiL_Zaf4Q&index=4&list=PLwJcdKVRxEGKMTZmYMu2ggPWE_SeJRHRr

Outdoor Mobile Robots



- The Thorvald and Husky outdoor robots
 - Rather more tricky conditions than your assignment simulations
 - Uneven ground can make sensing more difficult too



Example: Ibex

- See: <http://www.ibexautomation.co.uk/>
- (video above at: https://www.youtube.com/watch?time_continue=31&v=3Y0x7jYzQ5I)



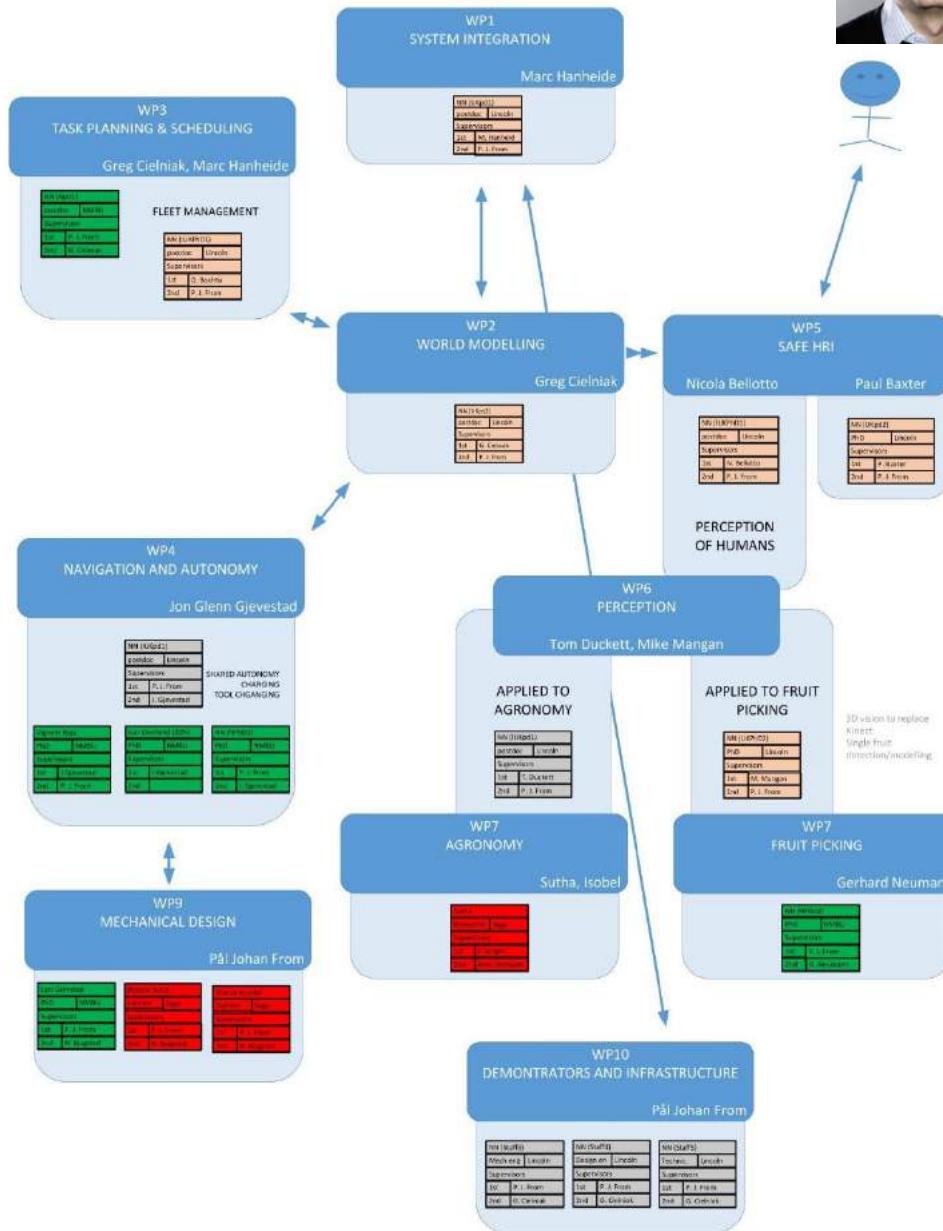
Integrated Robotic Systems

- The RASberry project
 - Fleets of autonomous robots for in-field food transportation
 - Must have human- and environment-aware behaviour
 - <https://rasberryproject.com/>



RASberry

- A large collaborative project involving two institutions and many different areas of work
- For example:
 - Navigation
 - Scheduling
 - Modelling
 - Perception
 - Picking
 - HRI
 - ...



RASberry



- Autonomous mobile robots for in-field transportation

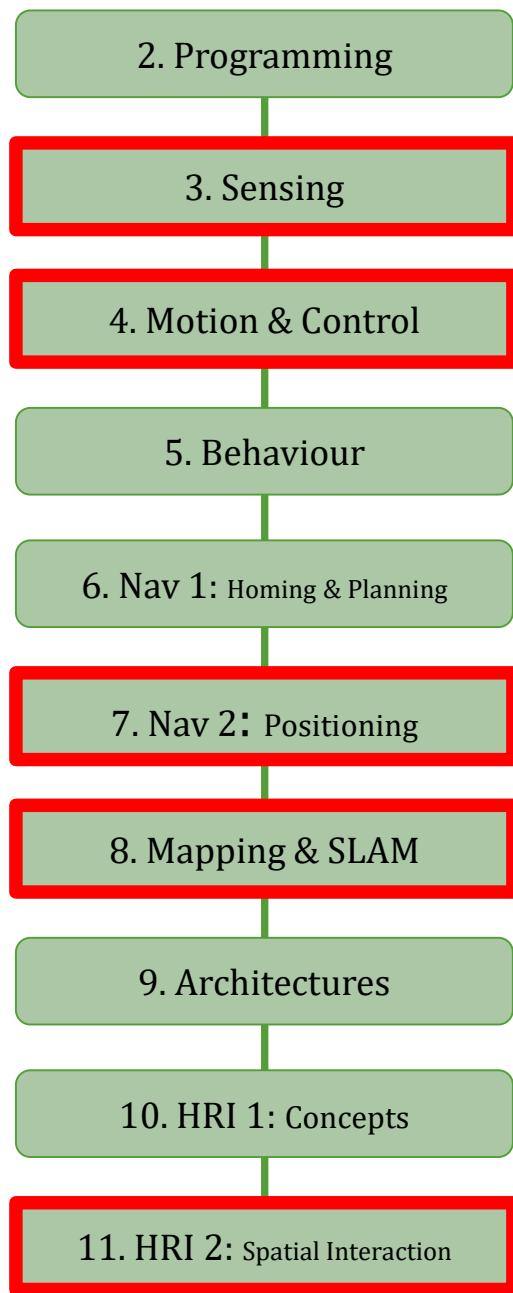




Logistics

- The ILIAD project
 - The development of autonomous warehouse robots for food distribution warehouses
 - See for more details: <http://iliad-project.eu/>
 - Particularly see: <http://iliad-project.eu/about/objectives/>







Bio-Inspired Embedded Systems

Learning from natural systems to improve synthetic systems...



Colias robots

- Small mobile robots, designed in Lincoln
- Have been used in a range of research projects, looking at various bio-inspired phenomena
 - Implement on a robot to help understand the biology
 - Also improve robot functionality



Bee-like clustering

- Bee-inspired clustering algorithm
 - Basic reactive obstacle avoid: if meet another bee, waiting time dependent on luminance
- Video from: <https://www.youtube.com/watch?v=BurlFPGJlIE>



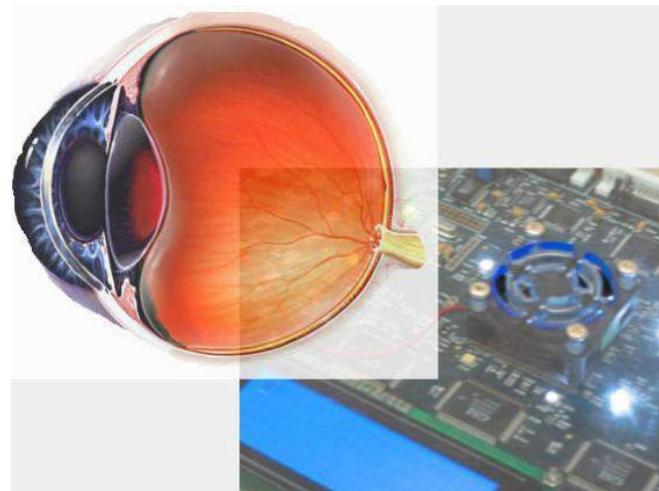
Pheromone trails

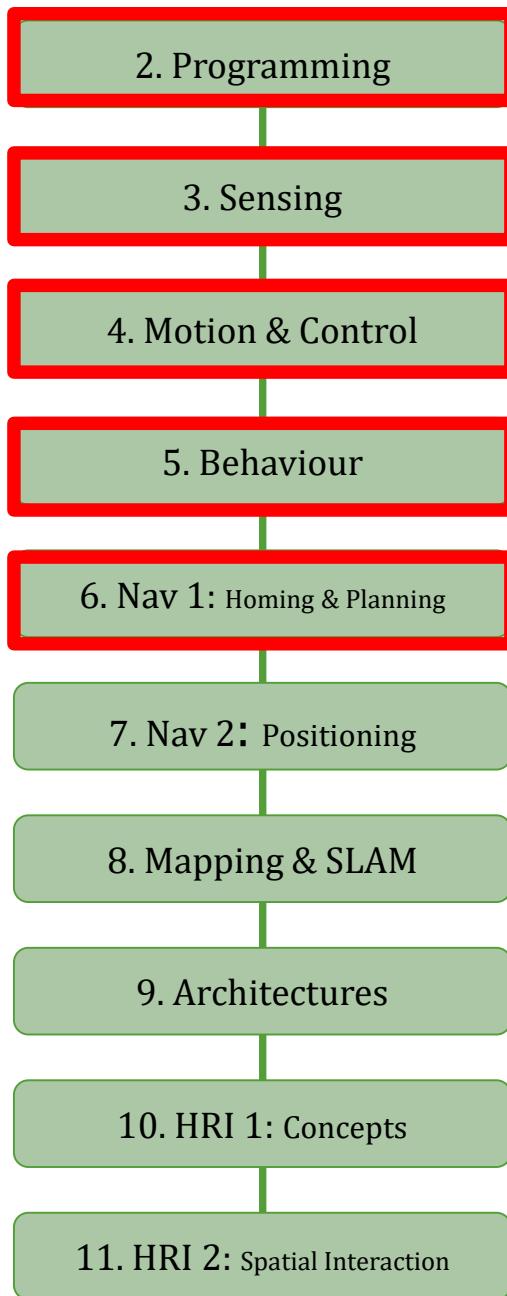
- Virtual pheromones, as inspired by ants
- Video from: <https://www.youtube.com/watch?v=HdbxjcYX7fg>



STEP2DYNA

- Collaborative project (EU and China)
 - Taking inspiration from insect brains (e.g. locust) to design new collision avoidance methods, particularly for robots and UAVs
 - Involves computational models of parts of insect brains and the design of new microchips to take advantage of aspects of these models
 - Uses the Colias robots and UAV devices as the test platforms
 - See: http://cordis.europa.eu/project/rcn/199944_en.html







AFT

BES

HCR

LAS

Human-Centred Robotics

Assistive Technology, Human-Robot Interaction, etc...



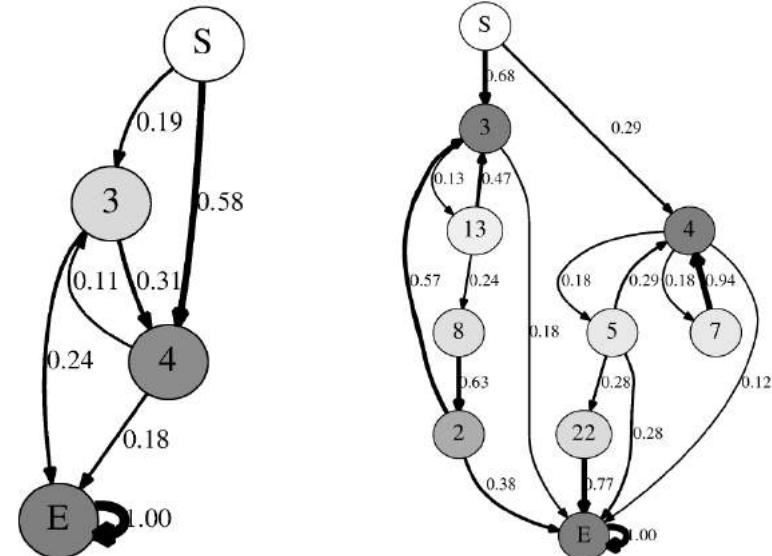
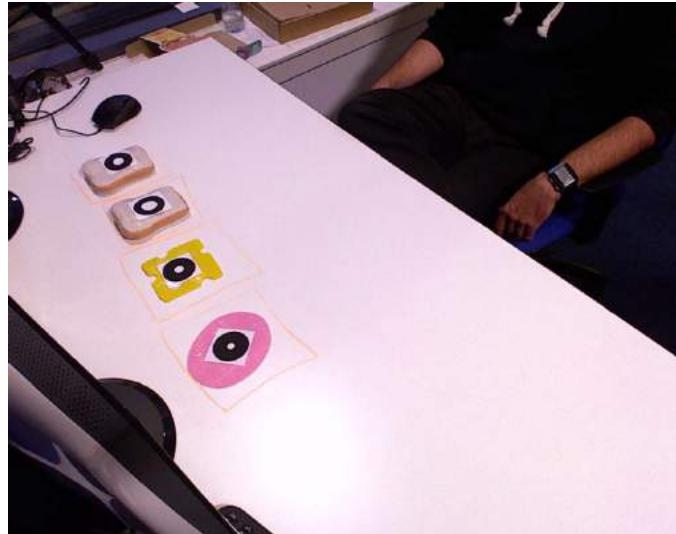
FloBot Human-Tracking

- Floor Washing Robot in commercial situations
- Needs to operate in environments with people, so needs to detect them
- Video from: <https://www.youtube.com/watch?v=H2dBDKZMFTE>



Individualised Collaboration

- FInCoR: Facilitating Individualised Collaboration with Robots
- Identifying individuals by the way in which they complete a task
 - E.g. making a sandwich (Lightbody et al, 2015)
 - Probabilistic sequence of states, Hidden Markov Model (HMM)
- Can use the result to personalise to individuals
 - Robot can learn by observing the human, then can figure out how best to help





Frequency enhanced maps

- Allowing temporal dynamics to be encoded within spatial models (maps, ...)
 - Makes use of fourier transforms to approximate thee dynamics from observation
 - See: <https://github.com/strands-project/fremen>

Both videos from T. Krajnik



Adaptive Info-Terminal

- Linda the robot learning about the most appropriate times/locations for the provision of what information
 - See: <http://strands.acin.tuwien.ac.at/>





Robot Assisted Living

- ENRICHME project:
 - Ambient Assisted Living, with robot, for monitoring and interaction to support independence for the elderly with declining cognitive capacity
 - See: <http://www.enrichme.eu/>



Assistive Robotics

- Shared control (recall levels of autonomy...)
- Learning from demonstration

See: <http://webpages.lincoln.ac.uk/akucukyilmaz/research.html>



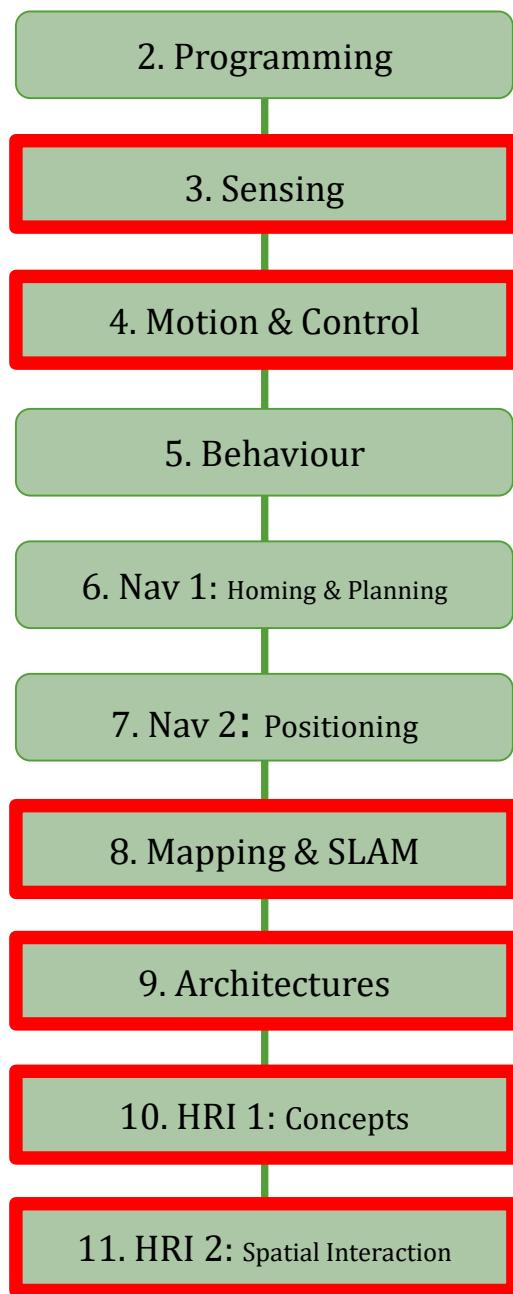
Robot Learning Peers

- Child and robot interacting over a touchscreen, with robot pretending to move items on the screen
- Incorporated a cognitive model of behavioural alignment
 - Allowed the children to learn better
 - See example for diabetic children:
<https://www.youtube.com/watch?v=K7pDGKKlxVU>



Robots for therapy

- For example, helping children with Autism Spectrum Disorders
- Adding to existing therapeutic interventions
 - Robot can play the role of a patient interaction partner: doesn't get tired or bored of repetition
 - Low expressivity is typically beneficial
- Big technical problems to be overcome:
 - How to tell what the child is doing?
 - How to respond appropriately?
 - What is best for the child?





AFT

BES

HCR

LAS

Learning for Autonomous Systems

Learning Algorithms, Machine Learning, Deep Learning, etc...



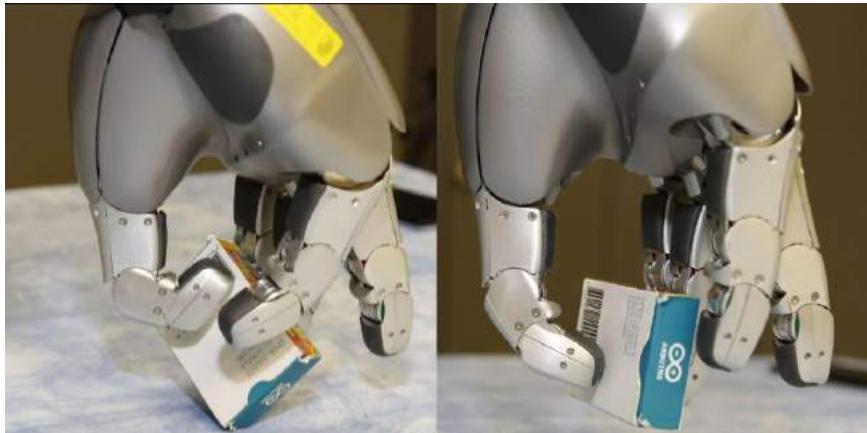
Game Playing with Deep RL

- Learning multimodal information to play noughts and crosses with deep reinforcement learning
 - Speech, game move, and head pose recognition
 - Learning to perceive -> learning to interact
- See the following:
 - Paper: <https://arxiv.org/pdf/1611.08666.pdf>
 - Video: <https://youtu.be/25jdV8FN4ic>



ML-Assisted Teleoperation

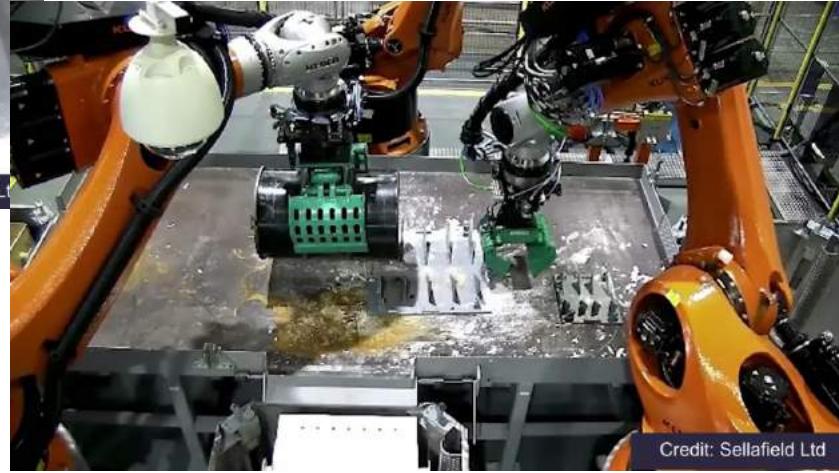
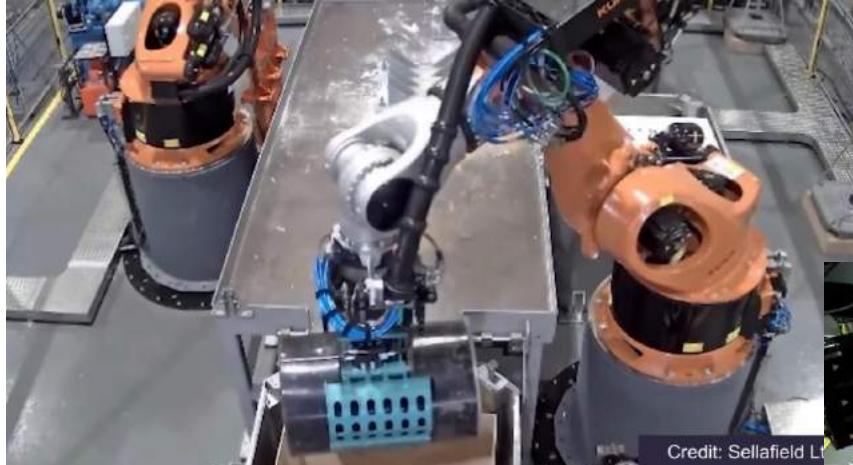
- RoMaNS project
 - Machine learning to assist with teleoperated manipulation tasks: related to nuclear waste handling
 - See: <https://www.h2020romans.eu/>

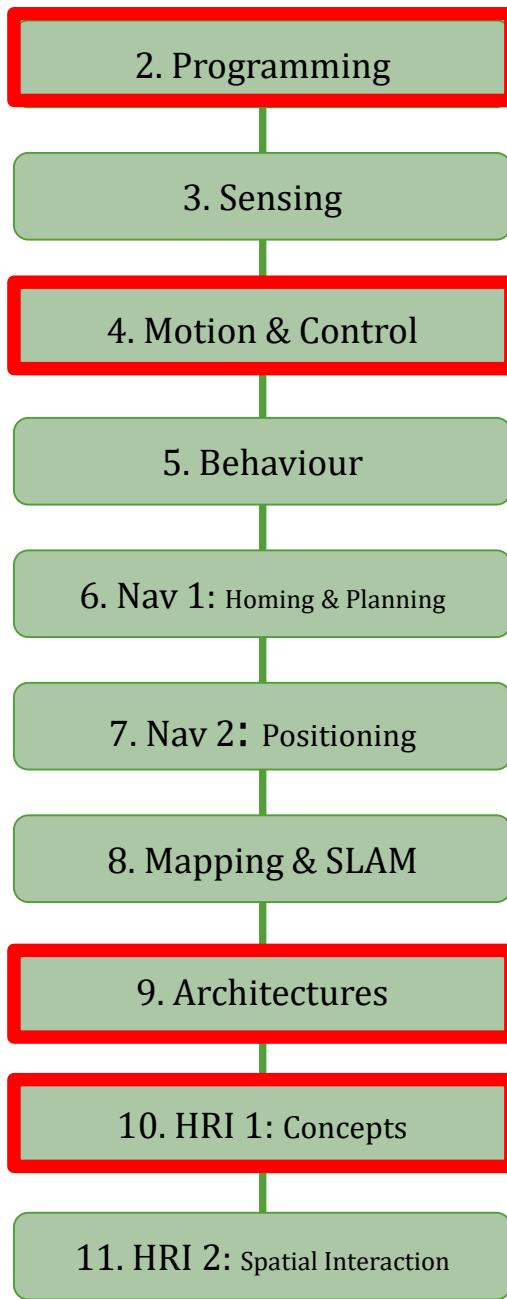


NCNR



- National Centre for Nuclear Robotics
 - <https://www.ncnr.org.uk/>
 - See: https://www.youtube.com/watch?v=YCg_XQYxUw





L-CAS Research Themes

AFT

Agri-Food Technology

BES

Bio-Inspired Embedded Systems

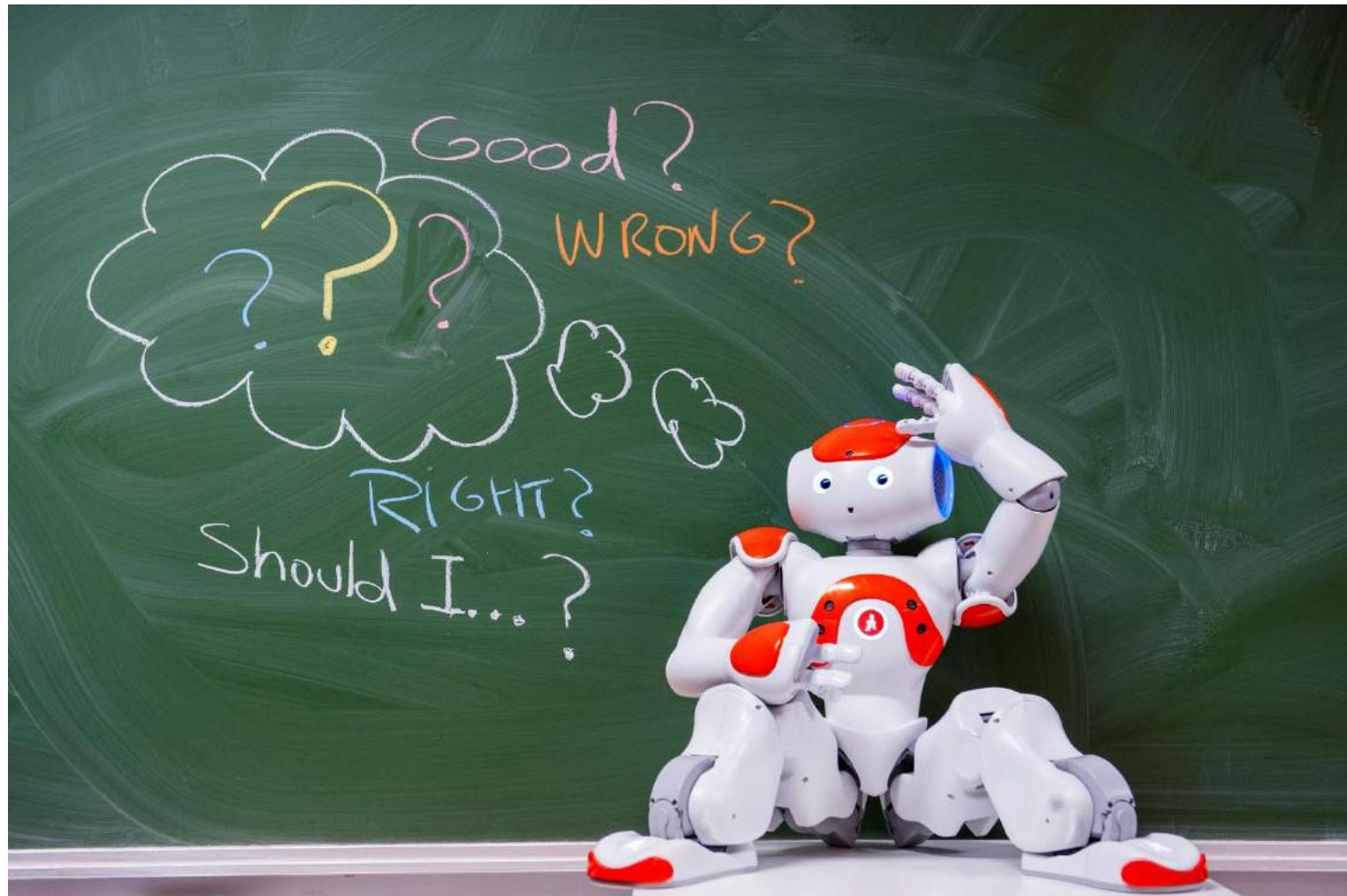
HCR

Human-Centred Robotics and Assistive Technology

LAS

Learning for Autonomous Systems

See L-CAS website: <https://lcas.lincoln.ac.uk/wp/projects>



Robots and Jobs

- Relatively recent big news on how robots will reduce human employment
 - Resulting from a US national report:
<https://www.nber.org/papers/w23285.pdf>
 - Various news articles following this: e.g.
https://www.theregister.co.uk/2017/03/28/robots_are_killing_jobs_after_all/
- Others worried not so much about jobs, but about increasing wealth imbalance:
 - The rich/powerful have less need to rely on other people for manufacturing, concentrating power further
 - E.g.
<https://www.theguardian.com/technology/2017/mar/02/robot-tax-job-elimination-livable-wage>

Mitigation?

- The beneficial role for robots:
 - Too dirty/dangerous for humans (e.g. the human waste)
 - Too monotonous/boring for humans (e.g. production lines)
 - Not enough humans for the job (front page news on Financial Times today...)
 - Robots offer a certain advantage over humans (e.g. some cases for ASD therapy, though not universal)
- Robots don't replace jobs, they *improve* them
 - History of automation suggests that people will do different types of job
 - E.g.
 - <https://techcrunch.com/2016/10/09/industrial-robots-will-replace-manufacturing-jobs-and-thats-a-good-thing/>
 - https://www.washingtonpost.com/opinions/workers-dont-fear-the-robot-revolution/2016/08/16/28c1606e-631f-11e6-96c0-37533479f3f5_story.html?utm_term=.175d6156e6e0

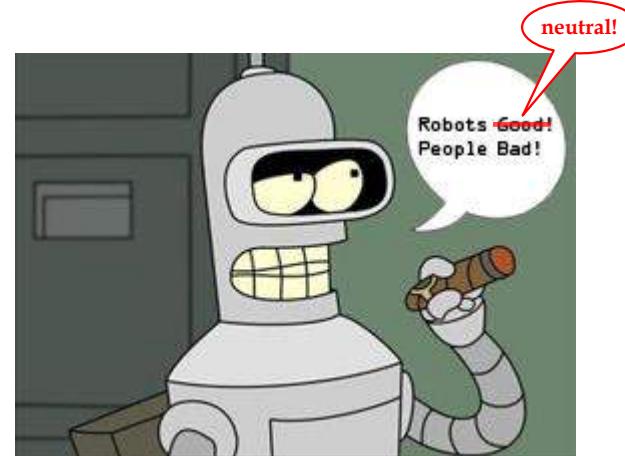
So how should we view robots?

- (eventually) autonomous agents?
- Just inanimate tools?
- Does this depend on the application context?

Guidelines from the community...

Principles of Robotics 2017

1. Robots are multi-use tools. Robots should not be designed solely or primarily to kill or harm humans, except in the interests of national security
2. Humans, not robots, are responsible agents. Robots should be designed; operated as far as is practicable to comply with existing laws, fundamental rights & freedoms, including privacy
3. Robots are products. They should be designed using processes which assure their safety and security
4. Robots are manufactured artefacts. They should not be designed in a deceptive way to exploit vulnerable users; instead their machine nature should be transparent
5. The person with legal responsibility for a robot should be attributed

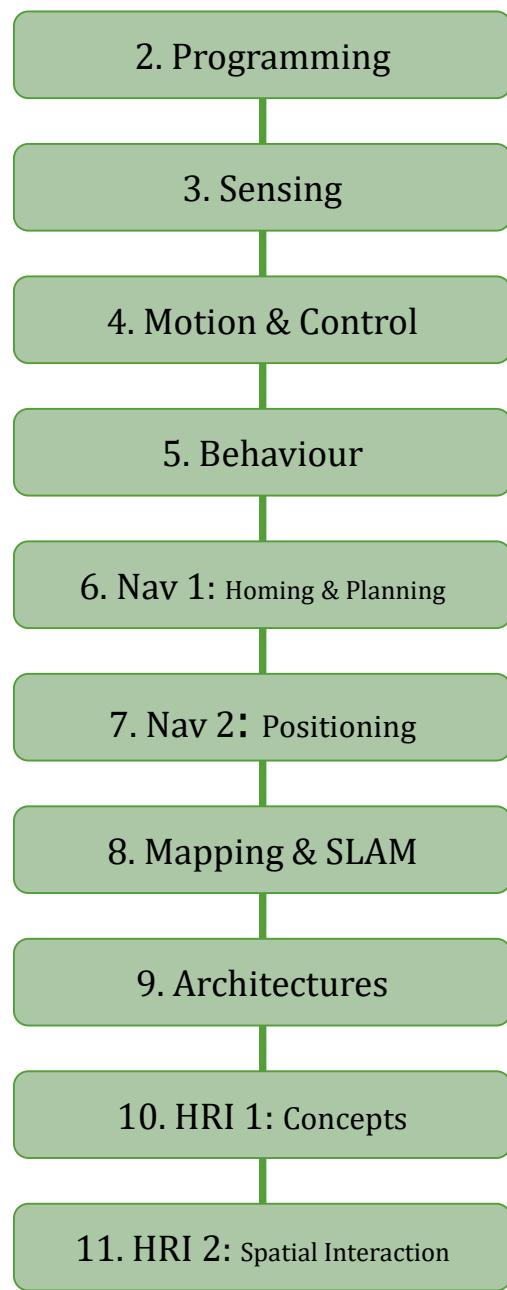


See full paper (open access): <http://www.tandfonline.com/doi/full/10.1080/09540091.2016.1271400>

Responsible Robotics

	Message
1	We believe robots have the potential to provide immense positive impact to society. We want to encourage responsible robot research
2	Bad practice hurts us all
3	Addressing obvious public concerns will help us all make progress
4	It is important to demonstrate that we, as roboticists, are committed to the best possible standards of practice
5	To understand the context and consequences of our research, we should work with experts from other disciplines including social sciences, law, philosophy and the arts
6	We should consider the ethics of transparency: are there limits to what should be openly available
7	When we see erroneous accounts in the press, we commit to take the time to contact the reporting journalists

See full paper (open access): <http://www.tandfonline.com/doi/full/10.1080/09540091.2016.1271400>



Workshops this week

- Friday/Monday 12th/29th April
- Workshop activity related to lectures both last week and today: mobile robot behaviours relevant to human-robot interaction
 - Part paper-based exercise: working from an academic paper
 - Part implementation: starting from the workshop code and assignment simulation environment
 - *Towards preparation for the exam*
- Academic paper on Blackboard in workshop materials for week 11:
 - Recommend you at least take a look at the paper before the workshop...

Lecture Week 13

- After Easter...
- The Week B13 lecture will serve as a brief review of the mock exam
- The mock exam will be released on Blackboard before the end of this week
 - ...and so available to you to work on for a few weeks before the mock exam review lecture

MSc in Robotics and Autonomous Systems

- Taught MSc starting in October, specialises in Robotics, with plenty of hands-on applications
- See: <http://www.lincoln.ac.uk/home/course/ROBASYMS/>
- Module list:

Advanced AI	Machine Learning
Advanced Robotics	Research Methods
Computer Vision	Research Project
Foundations of Robotics	Robot Programming
Frontiers of Robotics Research	
- Also see <https://lcas.lincoln.ac.uk/wp/study-with-us-towards-an-msc-in-robotics-and-autonomous-systems/>

References / Reading

- Boden, M., et al (2017), “Principles of robotics: regulating robots in the real world”, *Connection Science*, 29(2): 124-129
- T. Krajnik, J. P. Fentanes, G. Cielniak, C. Dondrup, T. Duckett: Spectral Analysis for Long-Term Robotic Mapping. ICRA 2014
- Lightbody, Peter and Dondrup, Christian and Hanheide, Marc (2015) *Make me a sandwich! Intrinsic human identification from their course of action*. In: Towards a Framework for Joint Action, 26th October 2015, Paris, France.