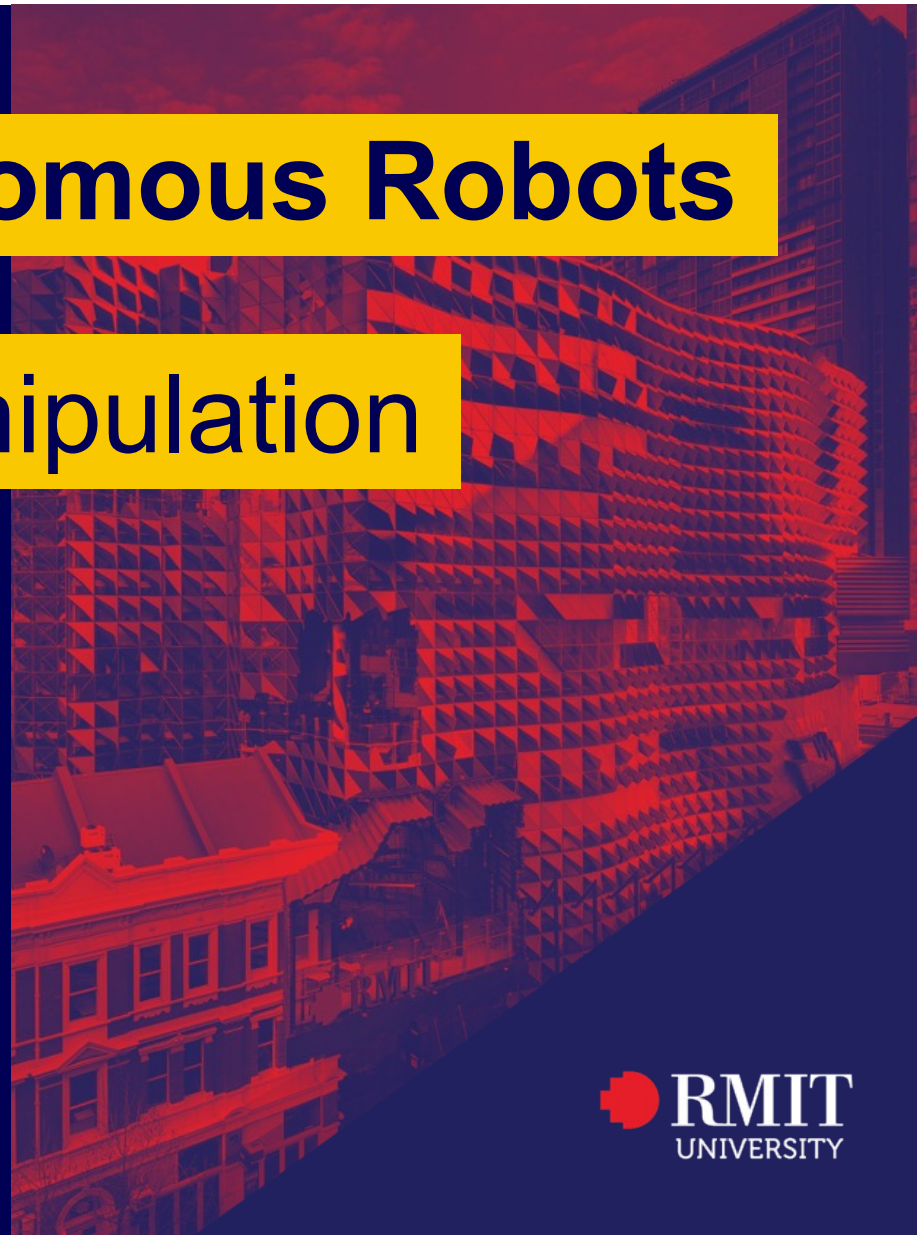
Practical AI for Autonomous Robots

Day 2: Control and Manipulation

Dr. Timothy Wiley
School of Computing Technologies
RMIT University



ESSAI July 2024

RMIT UNIVERSITY

Acknowledgement of Country

RMIT University acknowledges the people of the Woi wurrung and Boon wurrung language groups of the eastern Kulin Nation on whose unceded lands we conduct the business of the University.

RMIT University respectfully acknowledges their Ancestors and Elders, past and present.

RMIT also acknowledges the Traditional Custodians and their Ancestors of the lands and waters across Australia where we conduct our business.

Artwork ‘Luwaytini’ by Mark Cleaver, Palawa

Motivation

ESSAI July 2024

RMIT UNIVERSITY

Where is the ball?

Worked Example

Frames of Reference

ESSAI July 2024

RMIT UNIVERSITY

Frames of Reference & Poses

To describe poses a Frame is defined by:

1. Origin Point
2. 3D axes orientation
3. Following Right-hand Rule

A Pose is the tuple:

$$[position, orientation]$$

$$[x, y, z, roll, pitch, yaw]$$

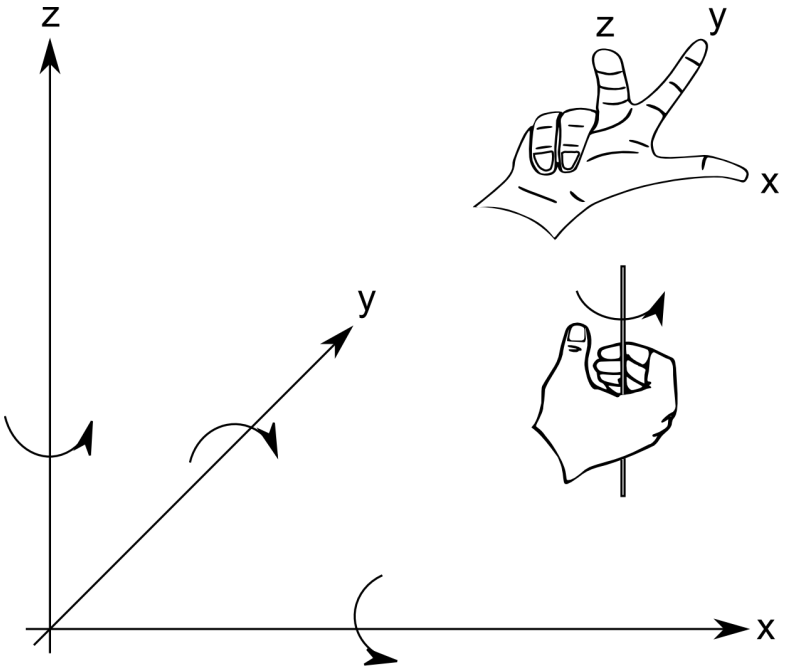


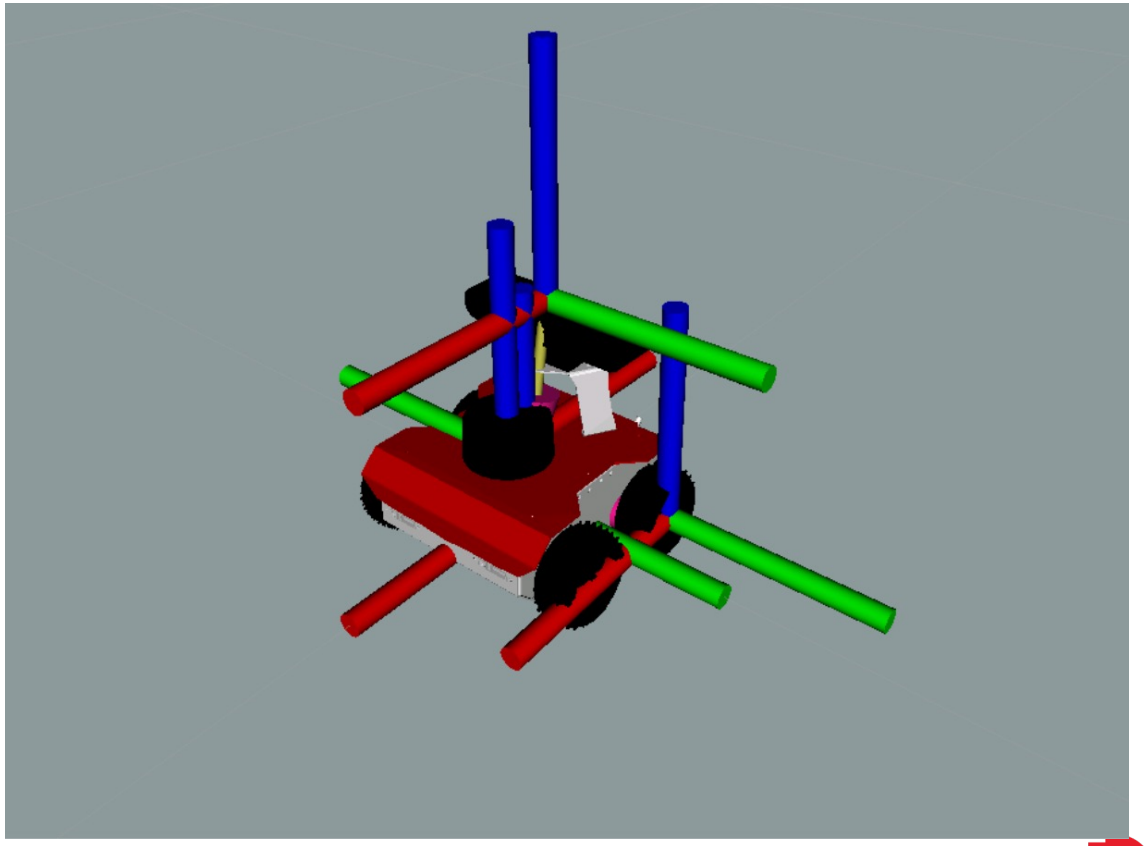
Image: Correll, 2022, introduction to Autonomous Robots

ROSBot Frames & Conventions

X – Red

Y – Green

Z - Blue



ROSBot Frames & Conventions

Conventionally:

- The “global” frame is /map
  - The x/y-axis parallel to the ground-place
  - The z-axis is vertical
- The default frame of robot is /base_link with
  - The x-axis is the ‘forward’ direction of the robot
  - The z-axis is vertical

Transforming Between Frames (links)

Defined by Linear Algebra Transformation with Homography Matrices

Point:

$$ {}^{A}P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} $$

Translation:

$$ T = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Rotations

$$ R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Transforming Between Frames (links)

Transforming a Point, Vector (or Pose) between frames:

$$^{W}P = {}^{W}_{R}T \times {}^{R}P$$

Transformation Matrices can be multiplied, but order matters

$${}^{W}_{R}T = T_x \times T_y \times R_x$$

Transforming Between Frames (links)

Defining Links

Links (between frames) may be:

- Static – such as fixed displacements of robot geometry
- Actuator – that take dynamic values from actuator (joint) measurements of the robot. Typically these are rotational links.
- Mapping/Odometry – that describe the transform from the “world” frame to a moving robot/object position. Typically these are a combination of a translation and rotation

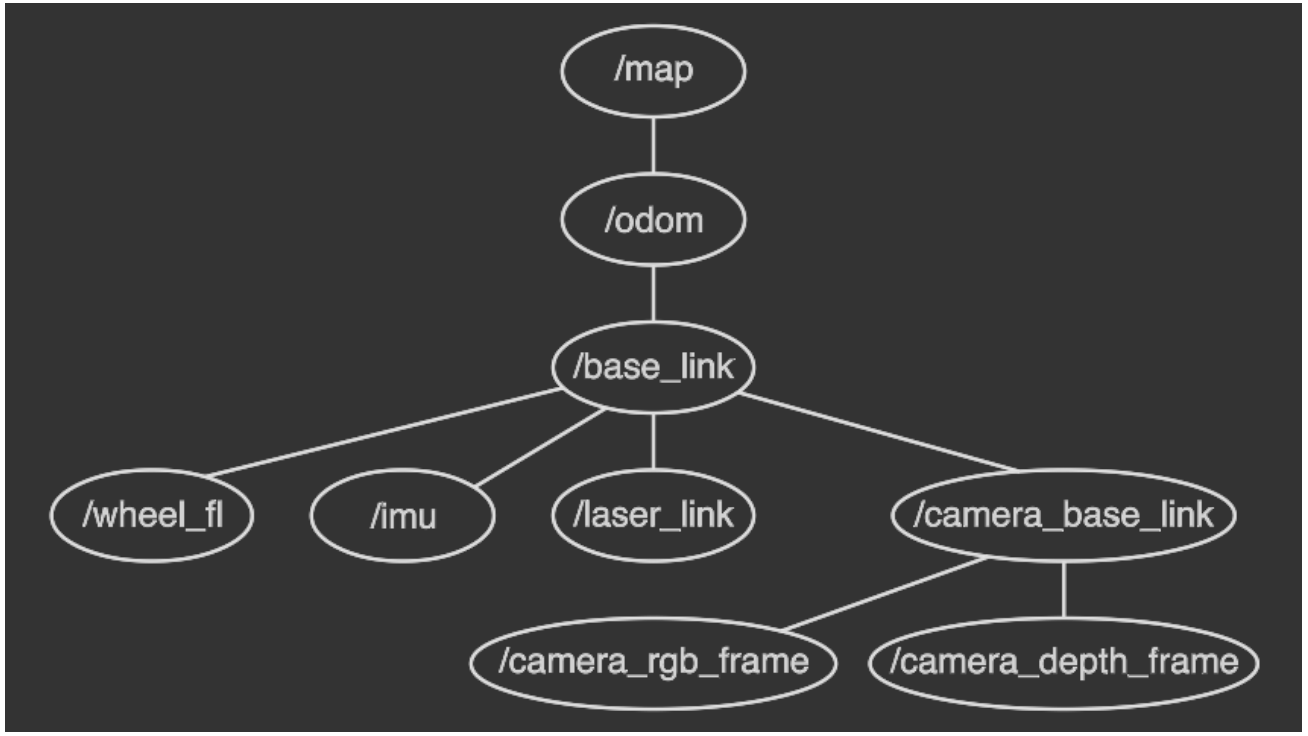
Transform Tree

ESSAI July 2024

RMIT UNIVERSITY

Defining a Complete Transform Tree

Defining a Complete Transform Tree



Forward Kinematics

ESSAI July 2024

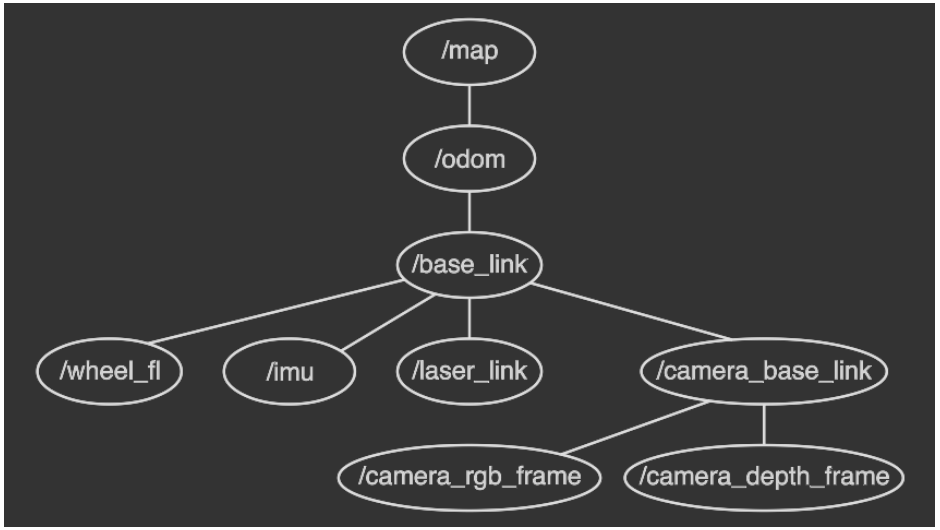
RMIT UNIVERSITY

Definition of Forward Kinematics

Forward kinematics is the “forward” application of the TF tree where:

- All TFs are known, that is, all joint angles are known
- Transform a point, vector, pose in one frame into another frame

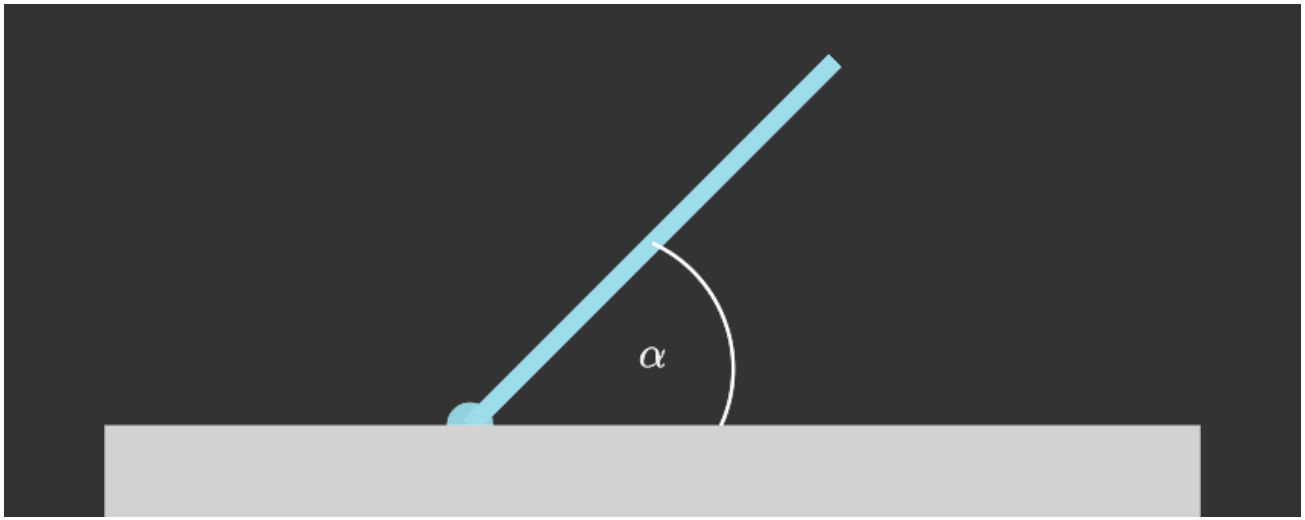
$$
\begin{aligned}
{}^{\text{camera\_rgb\_frame}}P = {}& {}^{\text{base\_link}}_{\text{laser\_link}}T^{-1} \times \\
& {}^{\text{base\_link}}_{\text{camera\_base\_link}}T \times \\
& {}^{\text{camera\_base\_link}}_{camera\_rgb\_frame}T \times \\
& {}^{\text{laser\_link}}P
\end{aligned}
$$



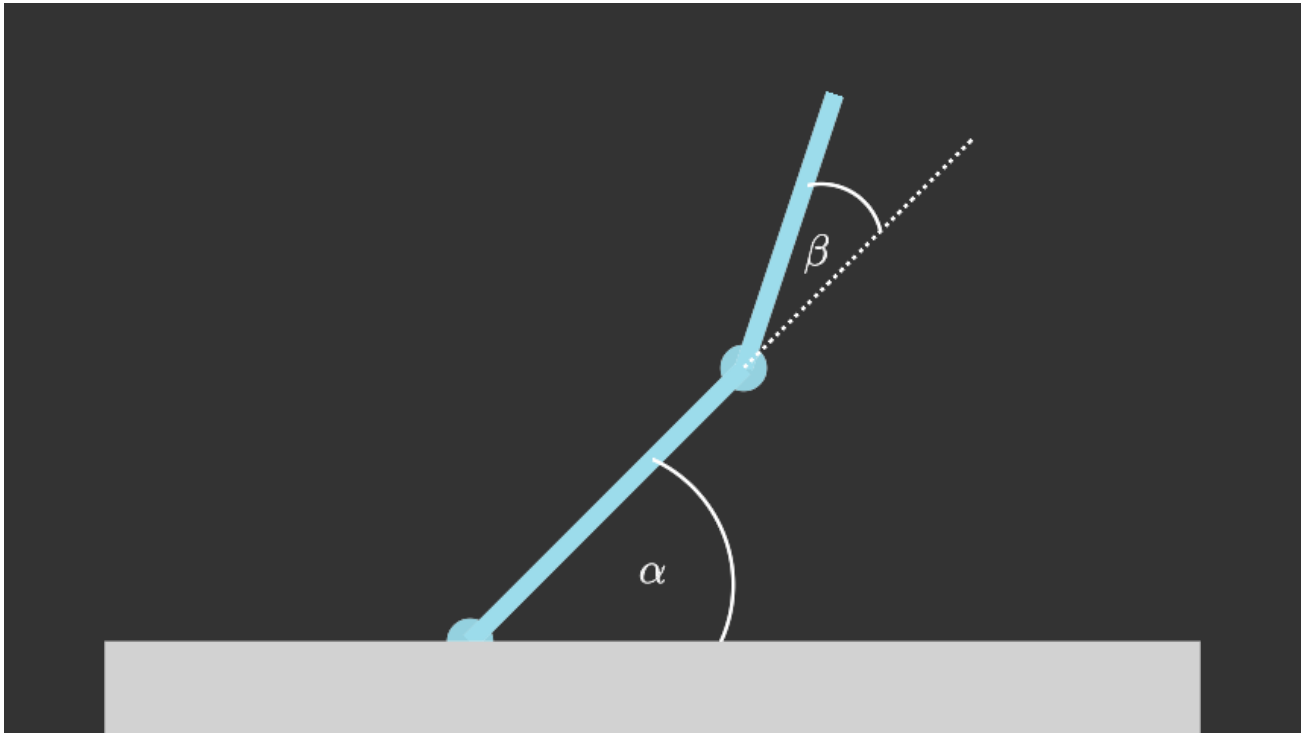
Single Joint Example

For a forward kinematic setup, what is the “world” position of the arm end-effector:

- The arm base is fixed at the “World” origin
- The angle α is known



Double Joint Example

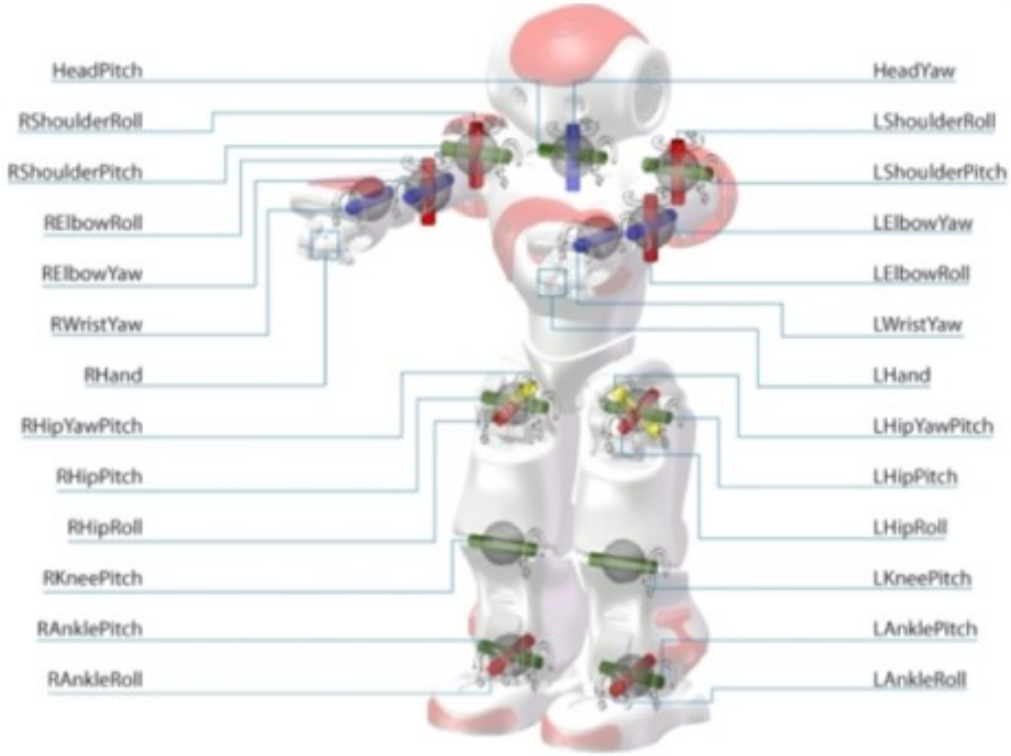


Transforms using the ROSBot in ROS

Worked Example

Nao Kinematics

A more complex robot has more transformation frames. But, this doesn't impact the overall mathematics on kinematic transforms



Real-time Kinematic Problems

The general issues to be aware of for any kinematics:

- Asynchronous updates of dynamic links
- Only approximate correspondence to sensor inputs, which also update asynchronously at a different rate to actuators
- Delays between TF update, processing, behaviour generation and final joint actuation

Inverse Kinematics

ESSAI July 2024

RMIT UNIVERSITY

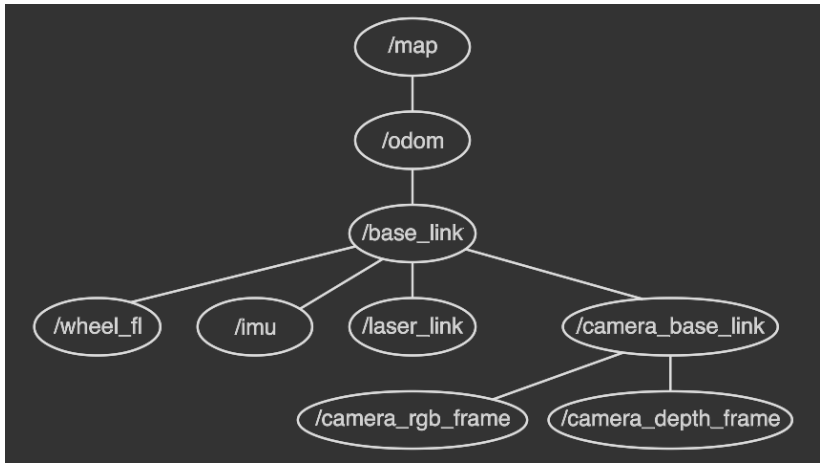
Definition of Inverse Kinematics

Inverse kinematics is the computation of required joint states to achieve a end-effector position:

- Static TFs are known
- Dynamic TFs must be computed

Requires solving for the transform of the forward kinematic equation

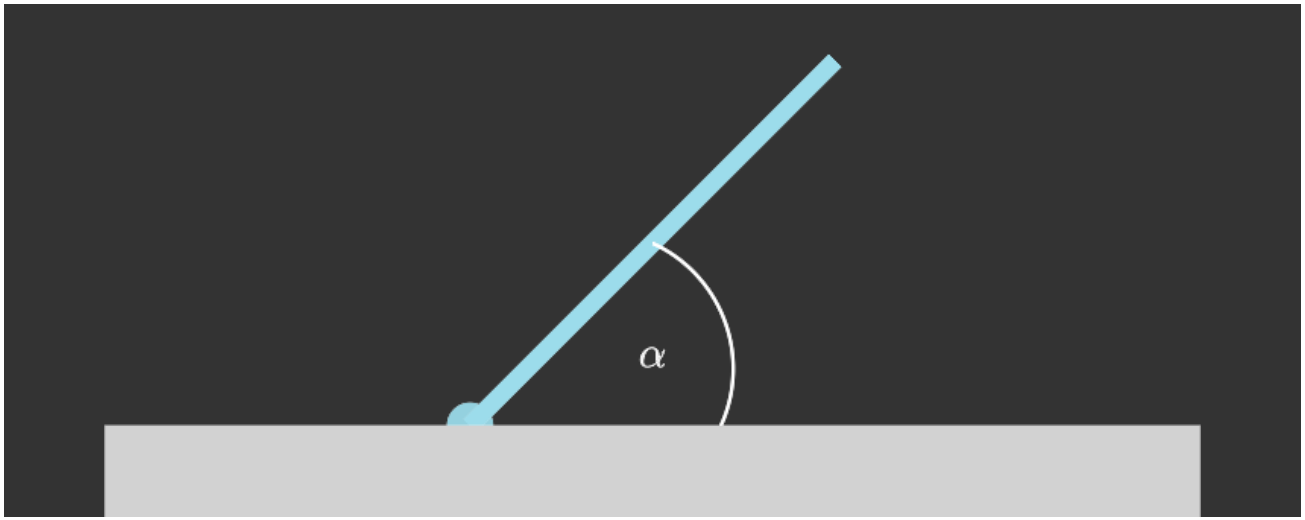
$$^{A}P = {}^{A}_{B}T \times {}^{B}P$$



Single Joint Example

For an inverse kinematic setup, what is required angle, α, for:

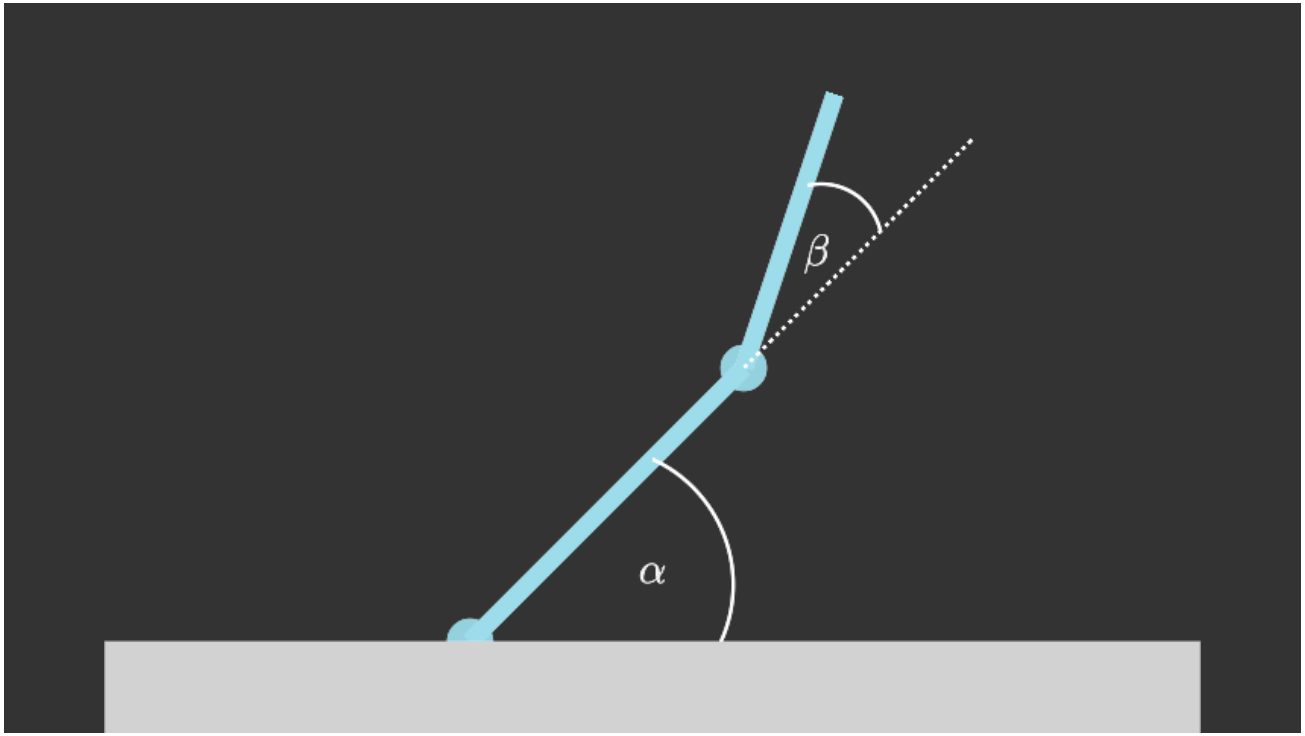
- The arm end-effector to be at: $\mathrm{P} = [3, 2]$
- Where, the arm base is fixed at the “World” origin



Double Joint Example

What is required angles, α and β, for:

- The arm end-effector to be at: $\mathrm{P} = [3, 2]$



Closed Form Solvability

It is possible to solve the equation using closed form methods:

- Observe that robotics systems are highly non-linear
- Analytical solutions can be found, depending on the complexity of the non-linear system
- Likely to have multiple solutions
- A robotic system should have at least 6DOF

Worked example in Matlab with the Puma560 arm

Closed Form Solvability

Limitations:

- Computation for complex kinematic chains
- Unsolvable solutions for under-actuated systems
- Multiple solutions for over-actuated system
- Singularities, if using Euclidean Transforms

Alternative Transform Representation

The transforms present are Euclidean Affine Transforms.

Alternative transform representations include:

- DH-Parameters, common in Mechanical Engineering
- Quaternions (for rotations)

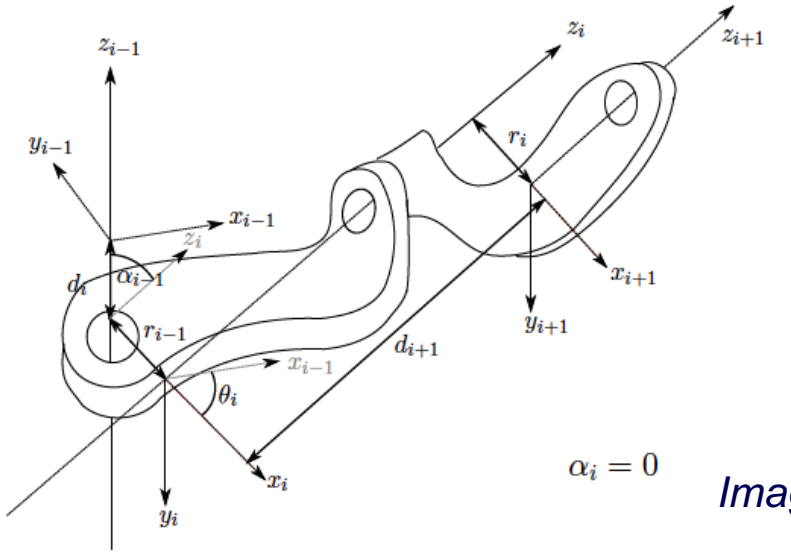


Image: Correll, 2022, introduction to Autonomous Robots

Approximate Solving via the Jacobian

Approximate closed-form solutions can be more efficient and “reasonable”:

- The kinematics define a multi-dimensional configuration space
- Finding the closed-form solution is the same as finding the minimum of this configuration space.
- This can be done by employing a technique similar to gradient descent to find the minimum
- That is, finding the derivative of the configuration space

Approximate Solving via the Jacobian

The Jacobian matrix, J, is all partial derivatives of a systems kinematics:

$$J = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$

For an analytical solution, solving still requiring inverting the Jacobian, which may not be possible. However:

- An approximation to inverting J is: $J^+ = \frac{J^T}{J.J^T} = \frac{1}{J}$
- This can be used for an iterative convergence method: $\Delta j = J^+ \epsilon$
- Where ϵ is the error between the desired and actual position

Motion Control

ESSAI July 2024

RMIT UNIVERSITY

Types of control

The general form of control for a robotic actuation system uses:

- Position control
- Velocity control
- Acceleration control

Rigid-body motion mathematically formalises a robot's motion:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}.\dot{\mathbf{q}})\dot{\mathbf{q}} + \tau_{\mathbf{g}}(\mathbf{q}) = \tau$$

Rigid-Body Motion

Rigid-body motion mathematically formalises a robot's motion:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}.\dot{\mathbf{q}})\dot{\mathbf{q}} + \tau_{\mathbf{g}}(\mathbf{q}) = \tau$$

But what is this mean?

- Control of is a function of time, that is, we are controlling a dynamic motion, not an instantaneous snapshot
- Control requires effective measurement of the joint parameters
- This motion is a function of all joint parameters
- The desired control can be computed by solving the motion equation
- There are many different motion equations depending on the robot

Open-Loop Motion Control

Rigid-body motion mathematically formalises a robot's motion:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}.\dot{\mathbf{q}})\dot{\mathbf{q}} + \tau_{\mathbf{g}}(\mathbf{q}) = \tau$$

This is what we have already looked at with inverse kinematics.

Practical Issues of Motion Control

Aside from the issue of solveability, open-loop control has issues with:

- Sensor noise
- Instantaneous actuation error
- Accumulative error
- Drift

Worked Example with ROSBot

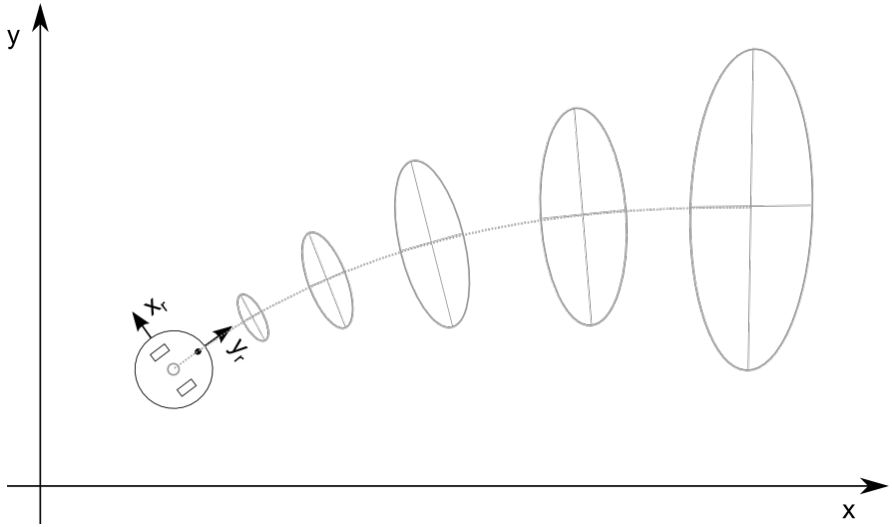


Image: Correll, 2022, introduction to Autonomous Robots

Closed-Loop Control

Closed-loop control uses actively monitors the error between a desired set-point of the actuator and actual-position. The motion-control (position, velocity and/or acceleration) is then adjusted based on the error.

This means the actuator is “servo-ed” into position gradually.

Closed-loop control can also be used in-place of inverse kinematic solving.

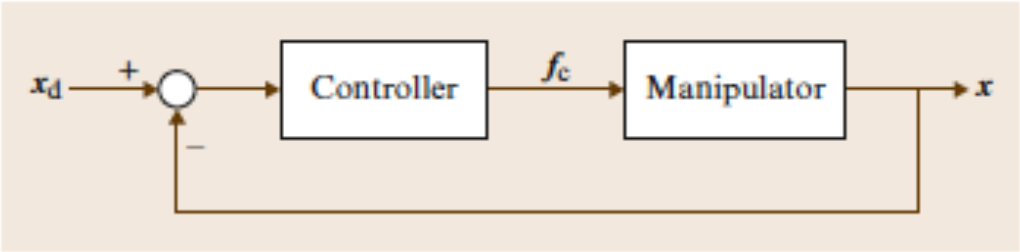


Image: Siciliano & Khatib (Eds), 2016, Springer Handbook of Robotics

PID Control

The most common form of closed-loop control is PID-control.

The general PID controller equation is:

$$\tau = K_p \mathbf{e_q} + K_i \int \mathbf{e_q} dt + K_d \frac{\mathbf{e_q}}{dt}$$

Where:

- P – Proportion of the instantaneous error
- I – Integral of the cumulative error
- D – Derivative of the instantaneous change in the error
- Each ‘K’ term is the ‘Gain’ of how much each PID term should be weighted

PID Control: Online example

The general PID controller equation is:

$$\tau = K_p \mathbf{e_q} + K_i \int \mathbf{e_q} dt + K_d \frac{\mathbf{e_q}}{dt}$$

Online PID example: http://grauonline.de/alexwww/ardumower/pid/pid.html

PID Control: ROSBot

The general PID controller equation is:

$$\tau = K_p \mathbf{e_q} + K_i \int \mathbf{e_q} dt + K_d \frac{\mathbf{e_q}}{dt}$$

ROSBot worked example of PID Control

PID Tuning

There are a wide variety of approaches for tuning the parameters of PID controllers including:

- Manual tuning
- Process Reaction Curve
- Ziegler-Nichols Method
- Cohen-Coon Method
- Lambda Tuning Method
- Internal Model Control
- Reinforcement Learning
- etc.

PID Tradeoffs

PID is still very common as it:

- Simple
- Directly uses known control dynamics of common systems
- Surprisingly robust to noise

However, the tuning of a PID system is critical. A poorly tuned system:

- May oscillate rather than converge
- May diverge with an exploding error

PID – Expanding for practical issues

Additional components can be included in PID (or closed-loop) controllers to correct for noise, error, slip, etc.

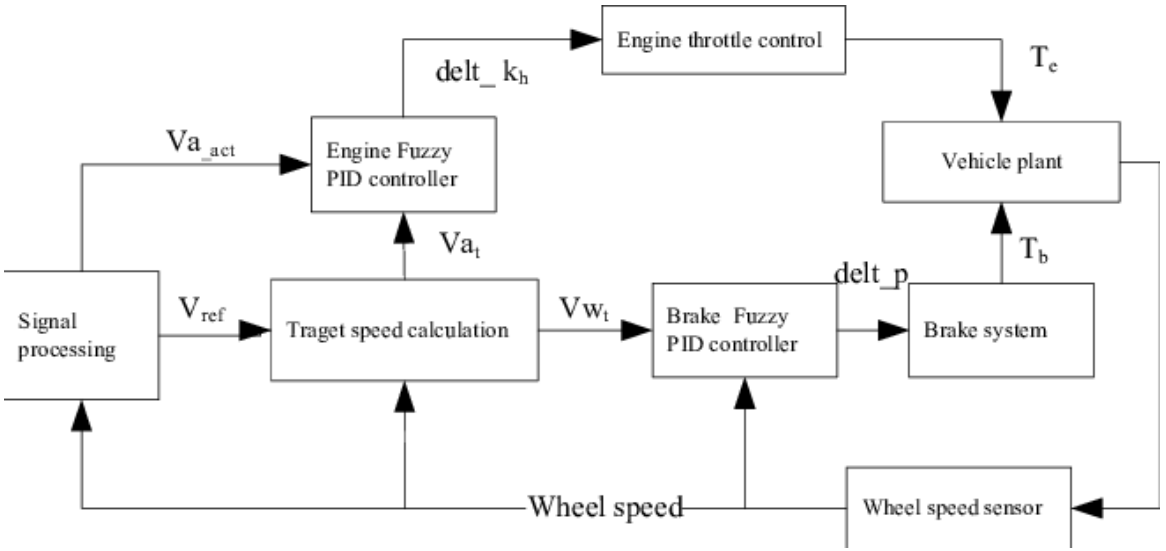


Image: Chu, Liang & Chao, Libo & Ou, Yang & Lu, WenBo. (2012). Hardware-in-the-loop Simulation of Traction Control Algorithm Based on Fuzzy PID. Energy Procedia. 16. 1685–1692. 10.1016/j.egypro.2012.01.261.

Motion Planning

ESSAI July 2024

RMIT UNIVERSITY

Motion Planning

Simple closed-loop control assumed that the robot actuators are capable of moving through all intermediate ranges without conflict.

For a complex actuation system, such as a robot arm, this assumption is not true, due to:

- Self collisions
- Environment collusions

Motion Planning: Example

Motion Planning

Motion planning overcomes this problem, by finding a sequence of intermediate positions for actuators before the end-position is reached.

A motion plan is the sequence of joint movements to move the joints from one position to another position.

Motion planning is one of the hardest problems now in robotics due to:

- Need for precise fine-grained motor control motion
- The complex nature of solving motion equations

Example with the Puma560

Worked example in Matlab with the Puma 560

Configuration Spaces

Motion planning can be solved analytically, but this is computational expensive.

An different approach to motion planning uses Configuration Spaces.

A configuration space described all valid actuator positions. When visualised, this is the actuator space, *not* the 3D space of the environment.

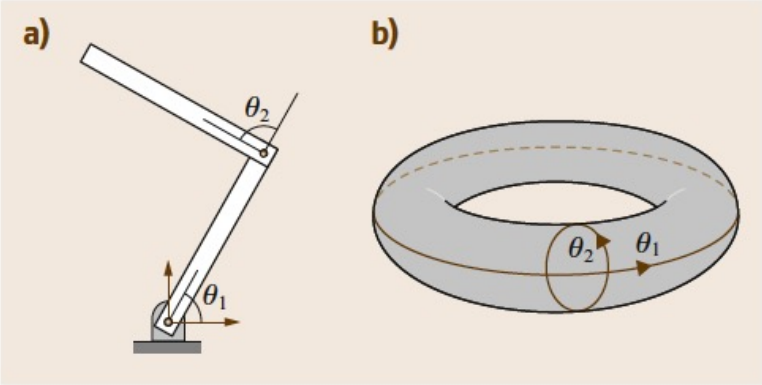


Fig. 7.2 (a) A two-joint planar arm in which the links are pinned and there are no joint limits. (b) The C-space

Image: Siciliano & Khatib (Eds), 2016, Springer Handbook of Robotics

Configuration Spaces

Configuration Spaces

Configuration spaces:

- May have complex shapes for self collisions
- Include boundaries (or breaks) for environment collisions

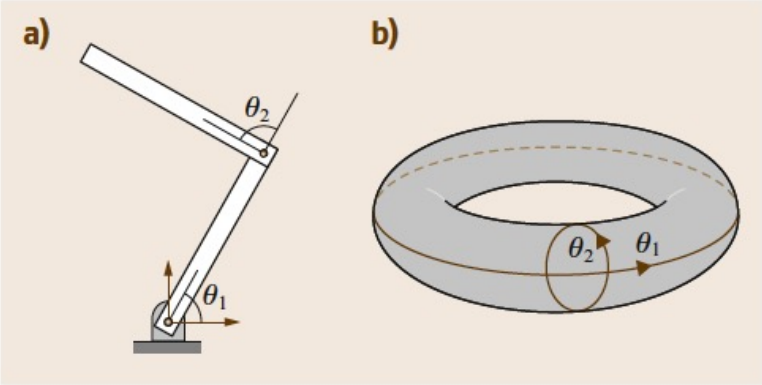


Fig. 7.2 (a) A two-joint planar arm in which the links are pinned and there are no joint limits. (b) The C-space

Image: Siciliano & Khatib (Eds), 2016, Springer Handbook of Robotics

Sampling Based Motion Planning

A common approach to Motion Planning is to randomly sample the configuration space and build a traversable graph through the space. Algorithm:

1. Initialise a Graph, G, with the starting & end configuration.
2. Randomly sample a configuration, $\alpha(i)$, from with the configuration space, and add as a vertex to G
3. Find all vertices in G within a neighbourhood of $\alpha(i)$ and connect the vertices if possible
4. Repeat (2) & (3)
5. Terminate when a path is found and at least *N* vertices are added to G

MoveIt, the ROS package, uses sampling based approaches, and you can visualise the planning

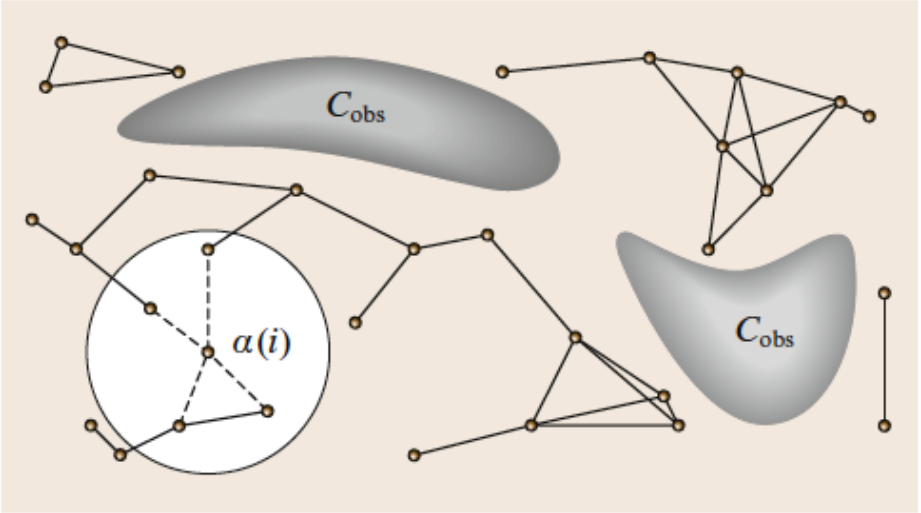


Image: Siciliano & Khatib (Eds), 2016, Springer Handbook of Robotics

Learning Motion

A practical perspective

ESSAI July 2024

RMIT UNIVERSITY

Learning Motion

Machine Learning in both simulation and online-learning is leveraged to learn complex motion. This could be a course to itself.

Learning motion is explored to handle a variety of issues in robotics:

- Noise
- Slip
- Hardware failures
- Probabilistic Actions
- Non-deterministic Actions

Reinforcement Learning is a very popular approach as various RL forms can help account for these issues without ‘manual’ intervention

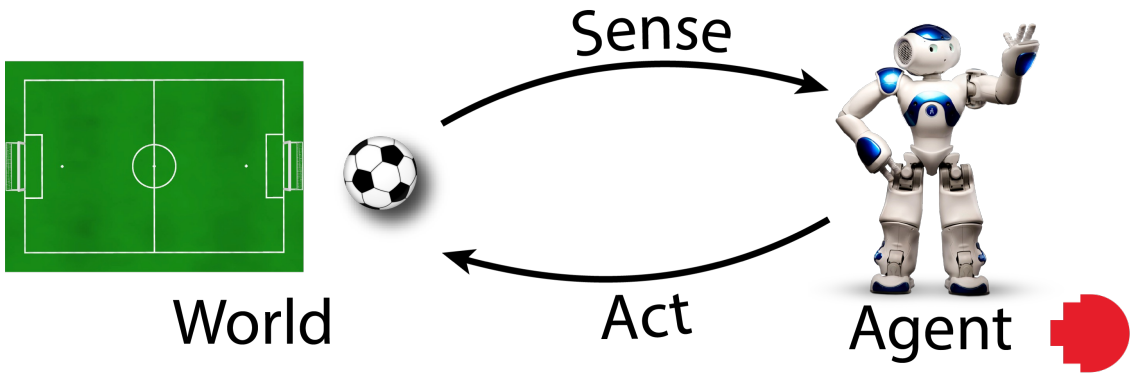
Learning Motion

As a reminder, the RL problems is defined as:

$$RL := < S, A, T, R, \gamma >$$

Where:

- S – State Space (discretised, continuous)
- A – Action Set (discrete, continuous)
- T – Transition Function (model-based, or model-free)
- R – Reward Function
- γ – Discount Factor



Learning Motion

Again, entire courses can be devoted to variations on this definition, along with structures for the RL agent. Wide varieties of RL methods have been investigated in both simulation and online learning:

- Classic Value/Policy Iteration
- Q-Learning
- PPO
- Deep Q-Learning
- Actor-Critic

… to name a few

Learning Motion: Issues

Reinforcement Learning methods encounter issues:

- Iterations required for convergence
- Balancing Exploration vs Exploitation
- Appropriate definition of the RL problem
- Transferring simulated behaviours to real-systems

This field is rapidly changing with Deep Networks and LLM approaches.

For ESSAI, we'll explore some more 'classic' techniques that are of value for a rounded understanding of leaning robot control.

Classic Example – Cart and Pole

For simplified domains, a traditional representation with any combination of value iteration, policy iteration, q-learning, etc., are successful:

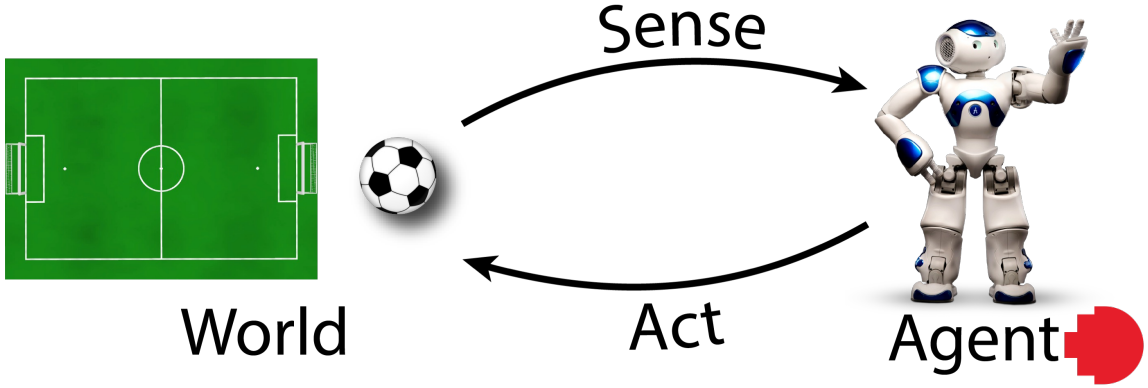
$$RL := < S, A, T, R, \gamma >$$

S – Cart/pole position

A – L/R force

T – Either model-based or model-free

R – Balancing time, distance from centre, etc.



Learning Motion: Examples

Learning Bi-Pedal Walk

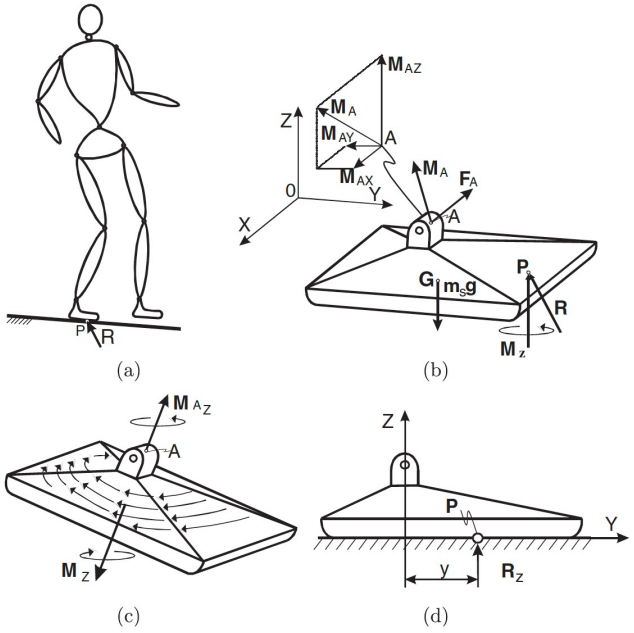
Earlier we've noted that control can be governed by known equations (models) of motion for a particular system.

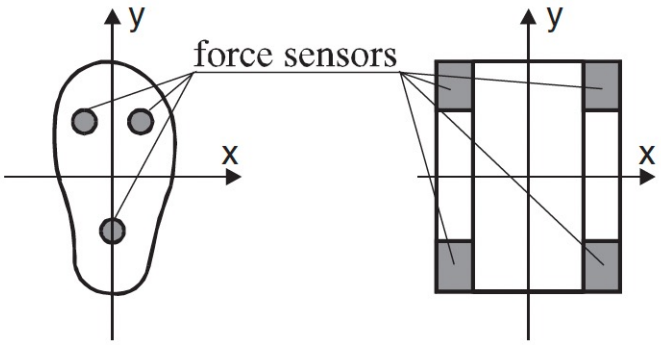
Bi-pedal motion has been studied quite some time, on Nao we typically modelled with two concepts:

- Zero-Moment Point (for foot control)
- Inverted Pendulum of dynamic walk motion

Bi-Pedal Walk (ZMP)

The ZMP concept simplifies bi-pedal walk by reducing all the forces acting on the robot to a single force termed the ‘Zero-Moment Point’.



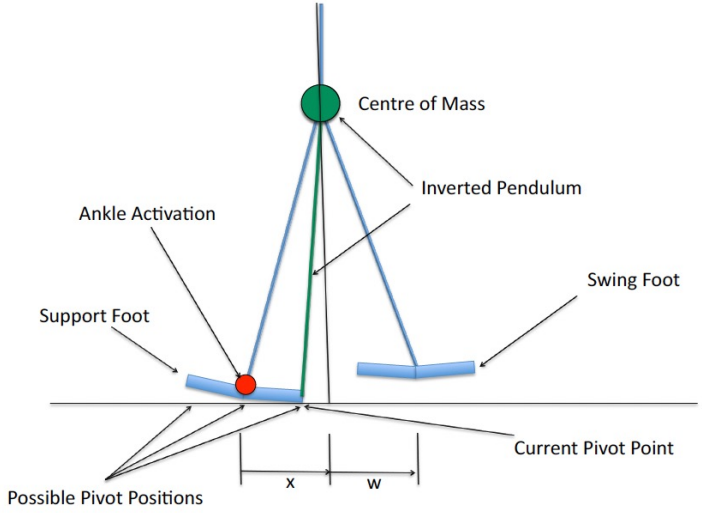


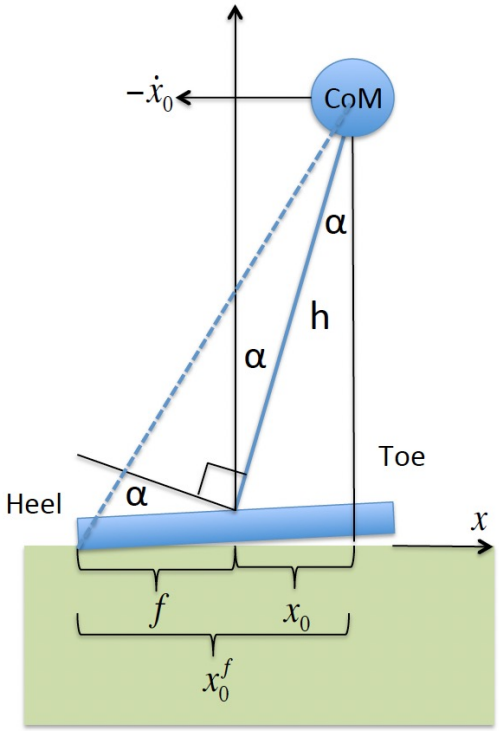
Images: Vukobratović, Miomir, and Branislav Borovac. "Zero-moment point—thirty five years of its life." International journal of humanoid robotics 1.01 (2004): 157-173.

Bi-Pedal Walk (Inverted Pendulum)

Bi-pedal walk also has a ‘rocking’ motion as:

- Weight is transferred between feet
- Single leg moving back/forwards



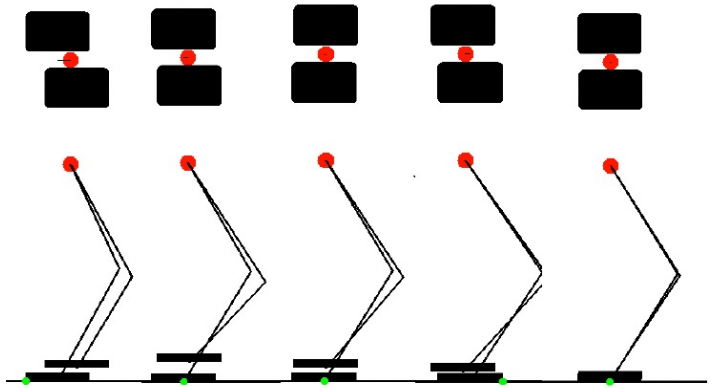


Images: Hengst, Bernhard, Manuel Lange, and Brock White. "Learning to control a biped with feet.”, Tech. Report 2014.
Hengst, Bernhard. "Reinforcement learning inspired disturbance rejection and Nao bipedal locomotion." 15th IEEE RAS Humanoids Conference. 2015.

Learning Bi-Pedal Walk

These combine to give a parameterised control equation, where the regularity of the pendulum is maintained, and the foot placement is controlled to maintain the ZMP within the foot boundary.

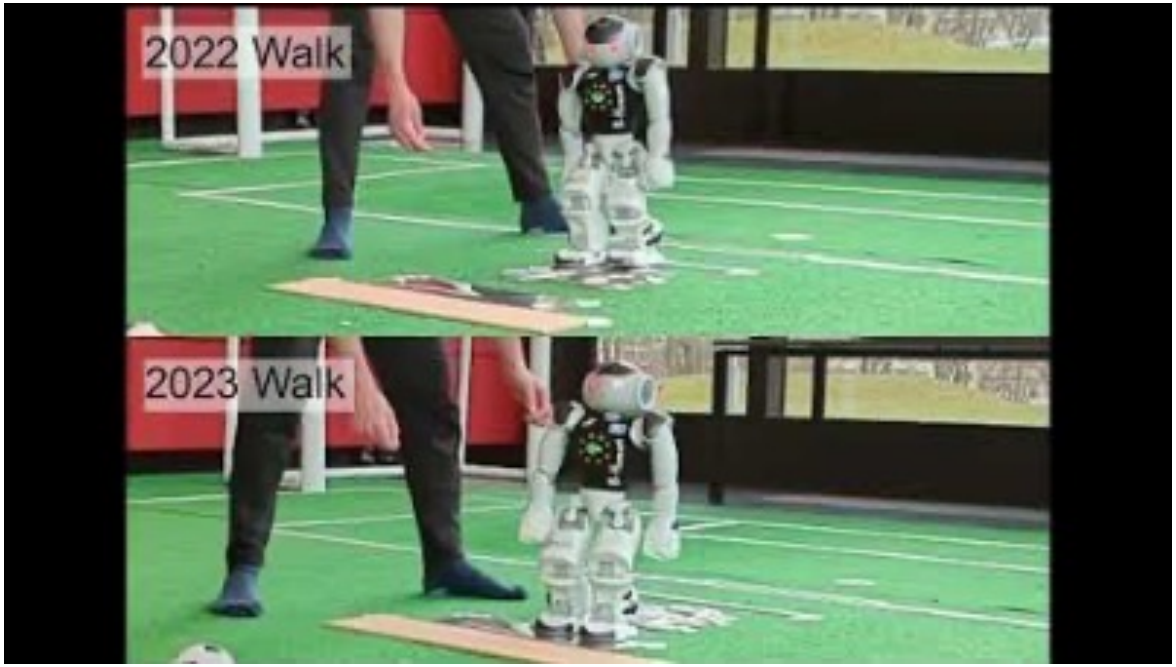
The parameters of the walk can are learnt through RL.



Hengst, Bernhard. "Reinforcement learning inspired disturbance rejection and Nao bipedal locomotion." 15th IEEE RAS Humanoids Conference. 2015.

Learning Joint Wear

Additional parameters can be introduced into the model to adjust for wear and tear in joints. Tuning (or learning) these parameters on a per-robot basis allows for correction of joint error.

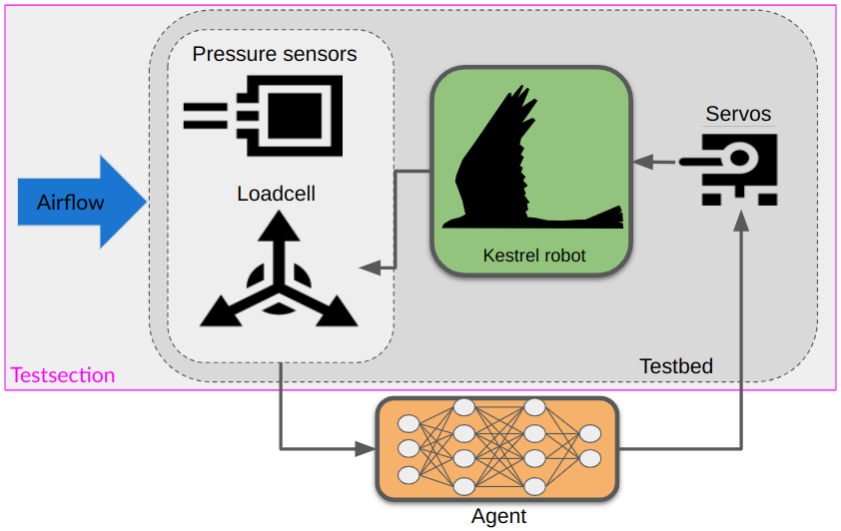


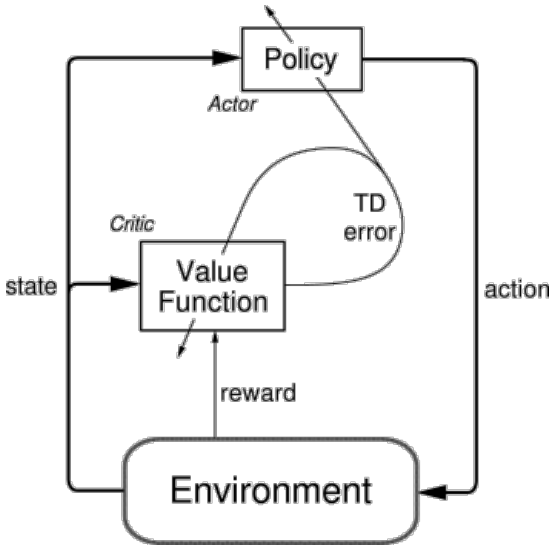
Reichenberg, Philip, and Thomas Röfer. "Dynamic joint control for a humanoid walk." Robot World Cup. Cham: Springer Nature Switzerland, 2023. 215-227.

Online Deep Reinforcement Learning

Online learning of robot control in real-world testing:

- Embedding action operators into the state space + action space
- Minimal shaping of reward function
- Actor-Critic RL structure

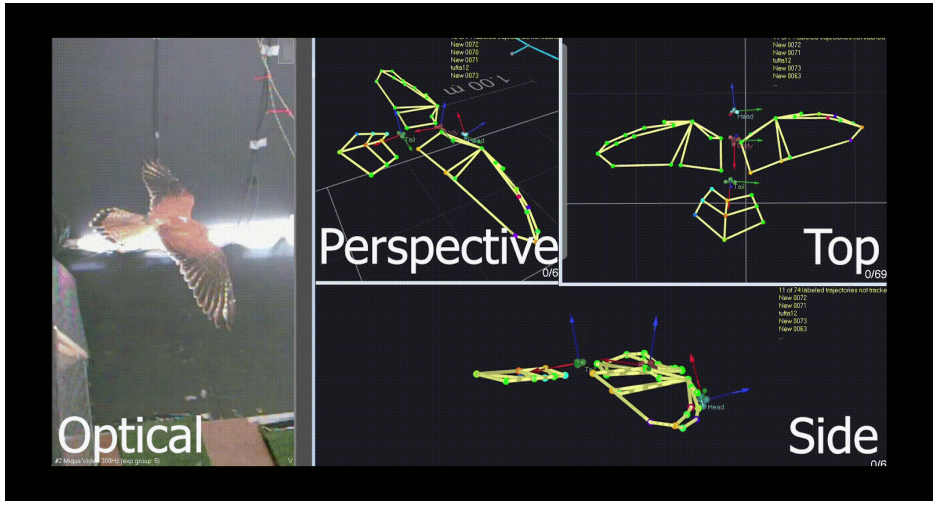




Online Deep Reinforcement Learning

Online learning of robot control in real-world testing:

- Embedding action operators into the state space + action space
- Minimal shaping of reward function
- Actor-Critic RL structure



Online Deep Reinforcement Learning

Sim2Real Learning

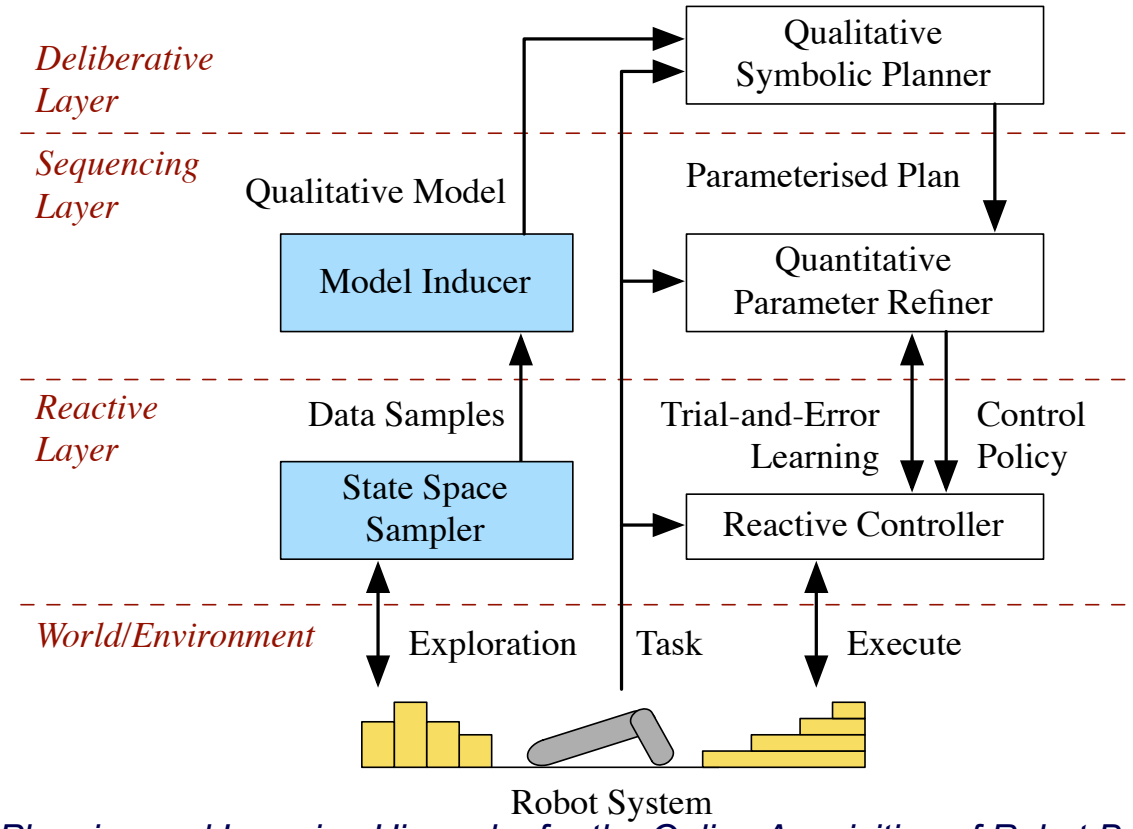
See talk by Katerina Fragkiadaki!

My favourite recent example is the training of DeepMind OP3 Soccer

- 1v1 Soccer behaviour
- Deep RL trained in Simulation, then further refinement on a real system
- Required intermediate guided rewards, such as requiring bi-pedal walk

Learning Motion: Examples (DeepMind)

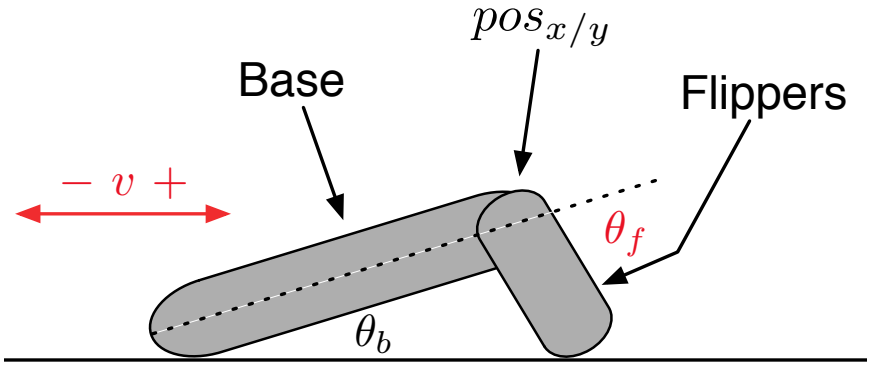
Multi-Strategy Learning

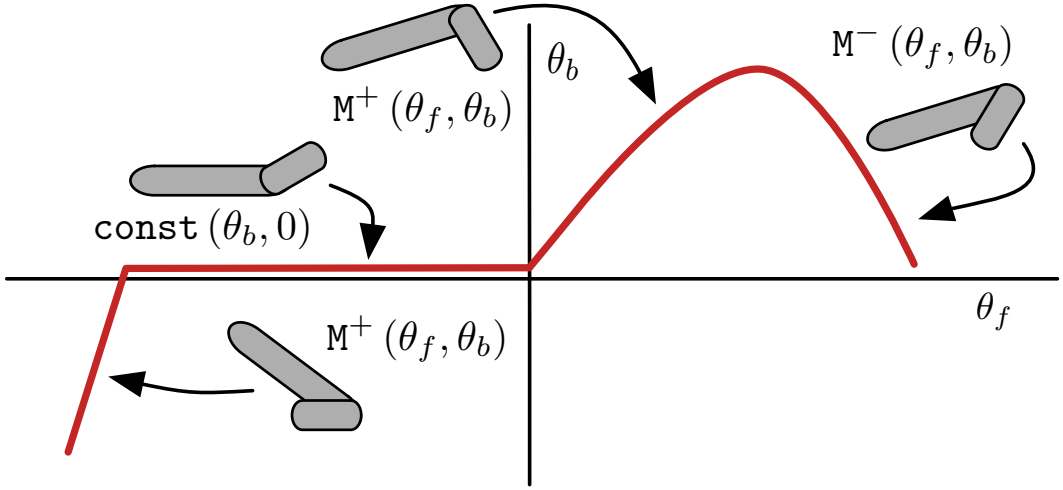


Wiley, T., A Planning and Learning Hierarchy for the Online Acquisition of Robot Behaviours. PhD Dissertation, (2017) UNSW, Australia Wiley, T., Bratko, I. & Sammut, C. A Machine Learning System for Controlling a Rescue Robot. in vol. 11175 108–119 (2018).

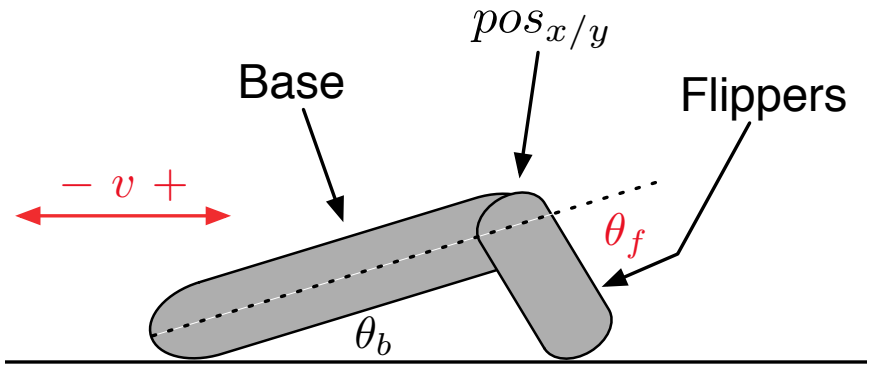
Combining Planning and Reinforcement Learning for Learning Locomotion

Negotiator Robot Model





Naive Reinforcement Learning



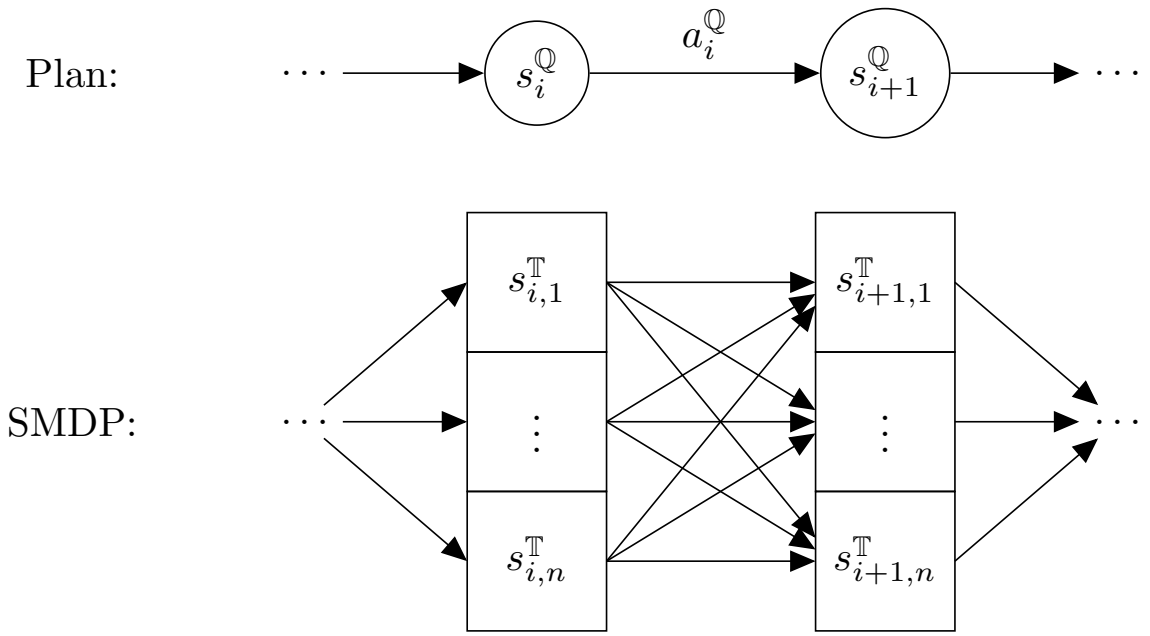
$$RL := \langle S, A, T, R, \gamma \rangle$$

$$S := [\theta_b, \theta_f, v, pos_x, pos_y, \ldots]_{discrete}$$

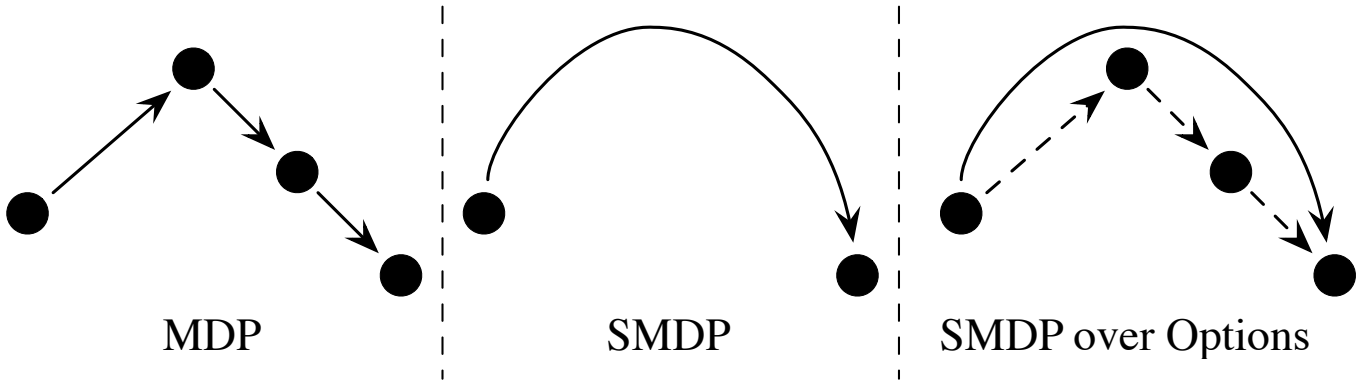
$$A := [\text{noop}, \text{inc}_{\theta_f}, \text{dec}_{\theta_f}, \text{inc}_v, \text{dec}_v]$$

$$R := \begin{cases} 100 & \text{if at goal,} \\ 0 & \text{if not at goal} \end{cases}$$

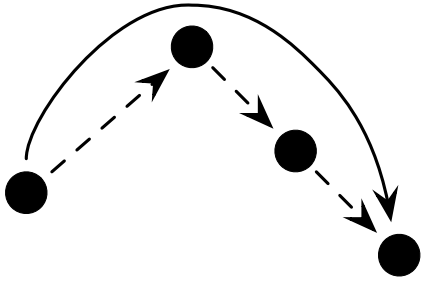
Planning into Reinforcement Learning



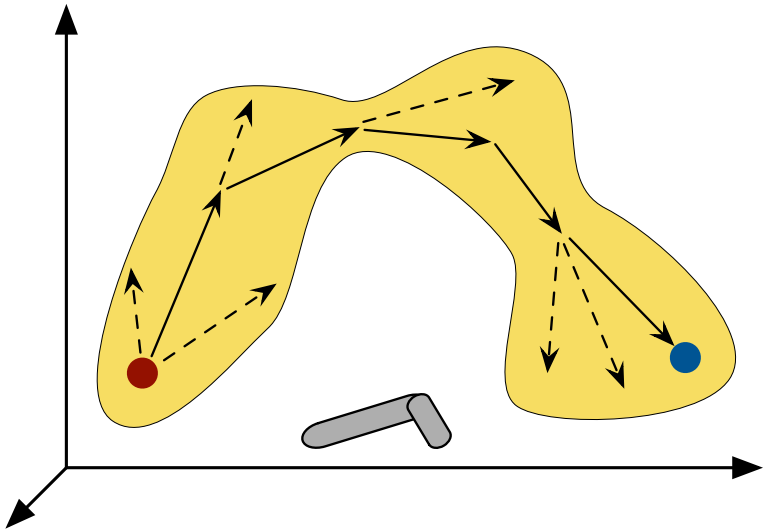
SMDP Over Options



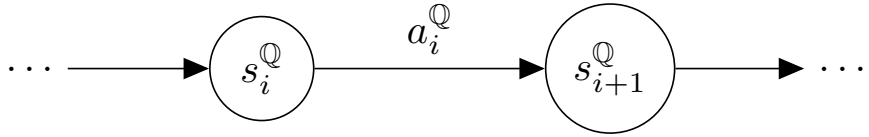
Planning into Reinforcement Learning



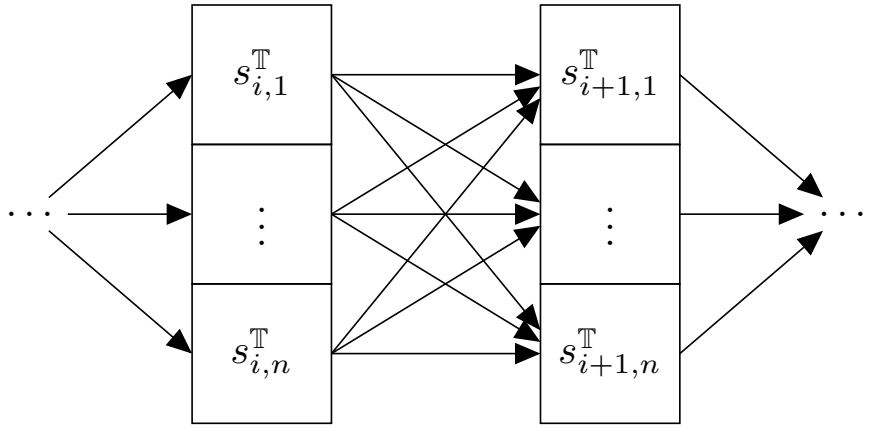
SMDP over Options



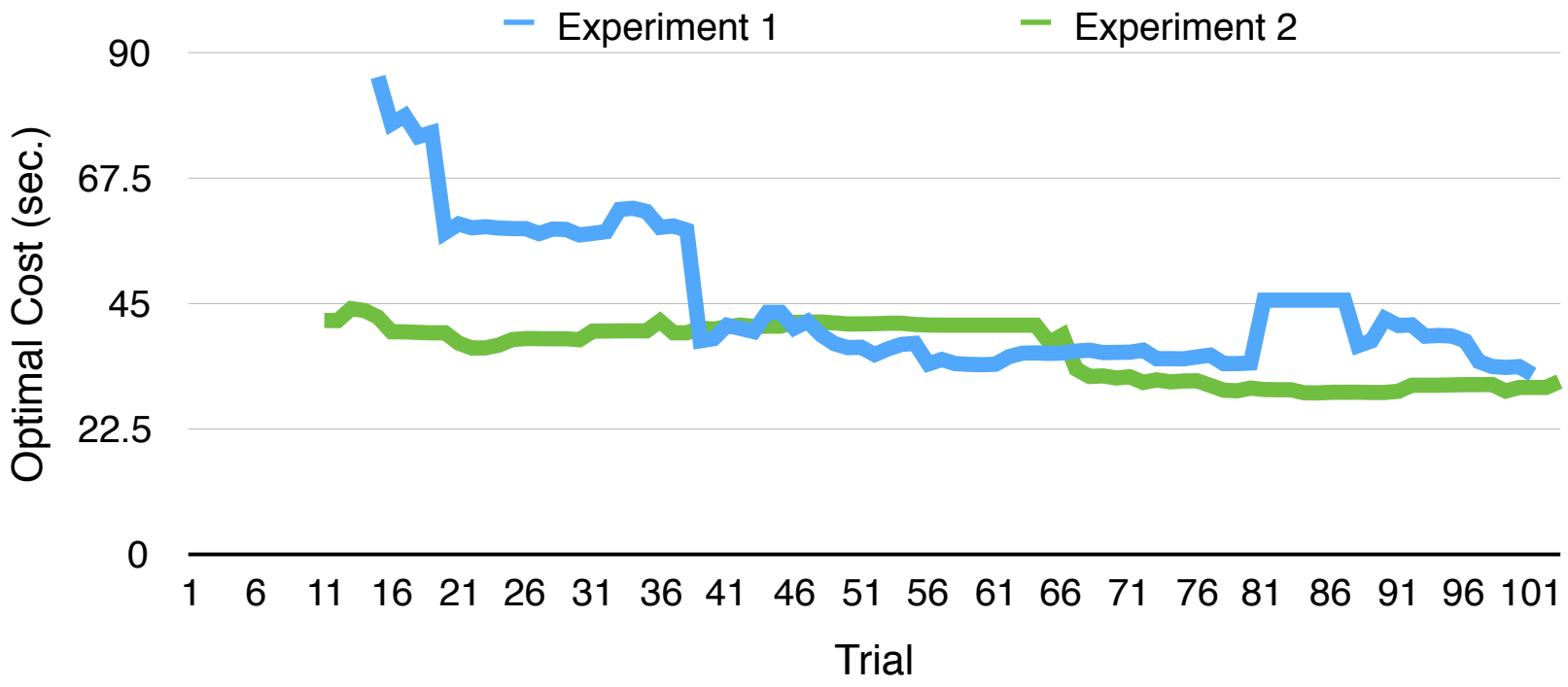
Plan:



SMDP:

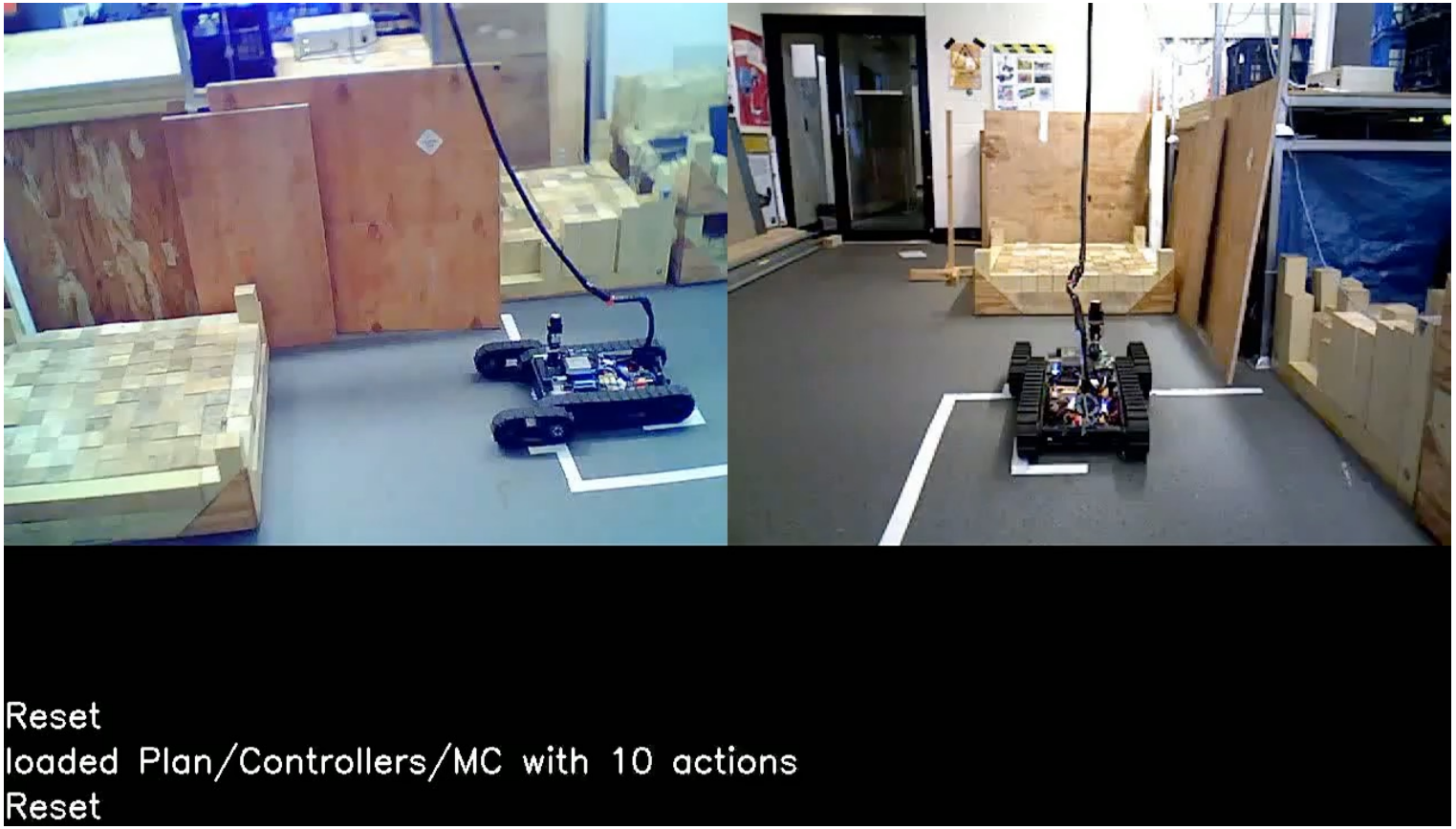


Planning into Reinforcement Learning

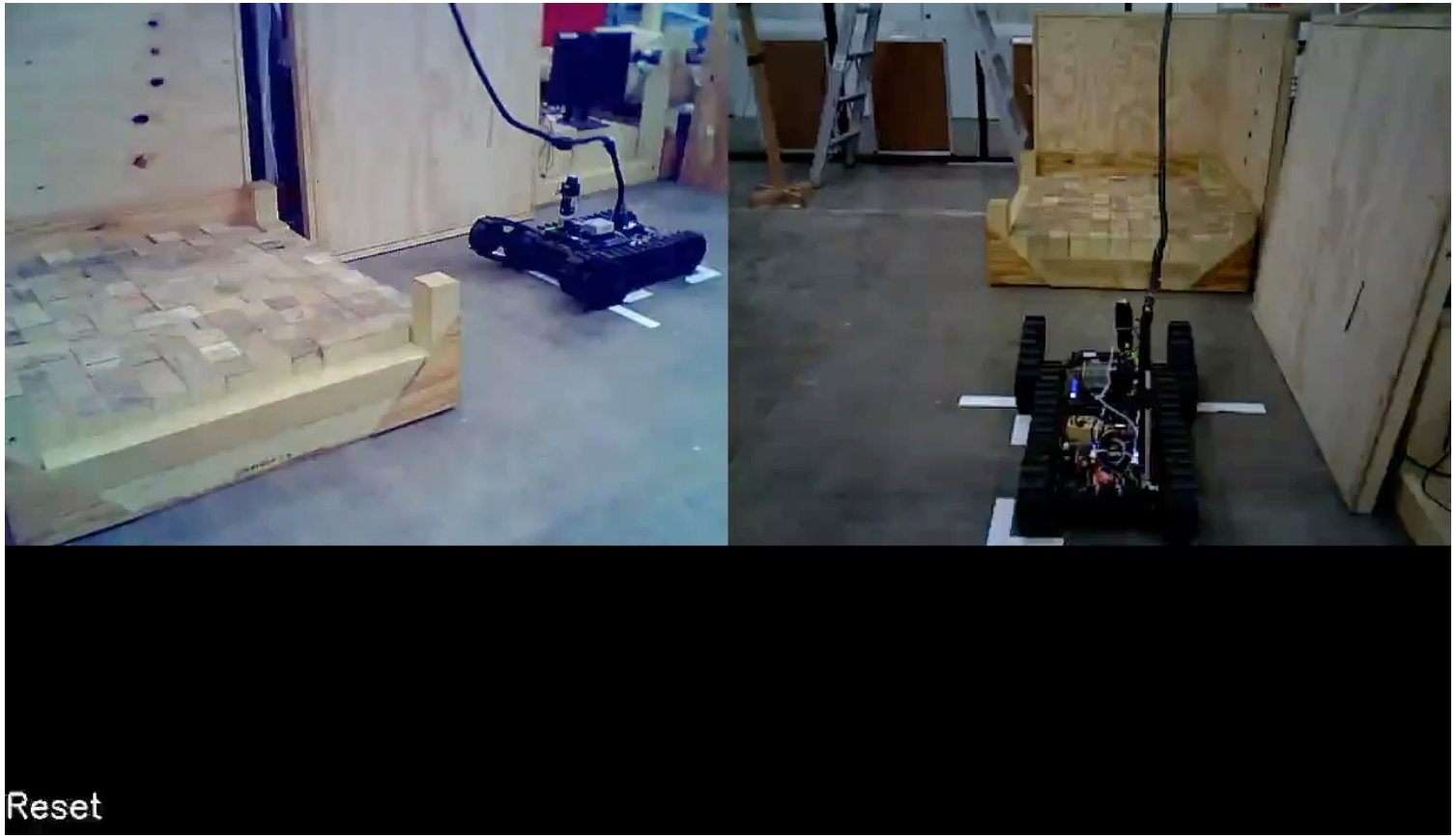


Examples of Learning

Examples of Learning



Examples of Learning



Noon Gudgin

Thank you

Day 3: Localisation, Mapping and Navigation

ESSAI July 2024

RMIT UNIVERSITY