Humanoid Robotics

Locomotion 3: Footstep Planning and Whole-Body Control

Maren Bennewitz

HRL

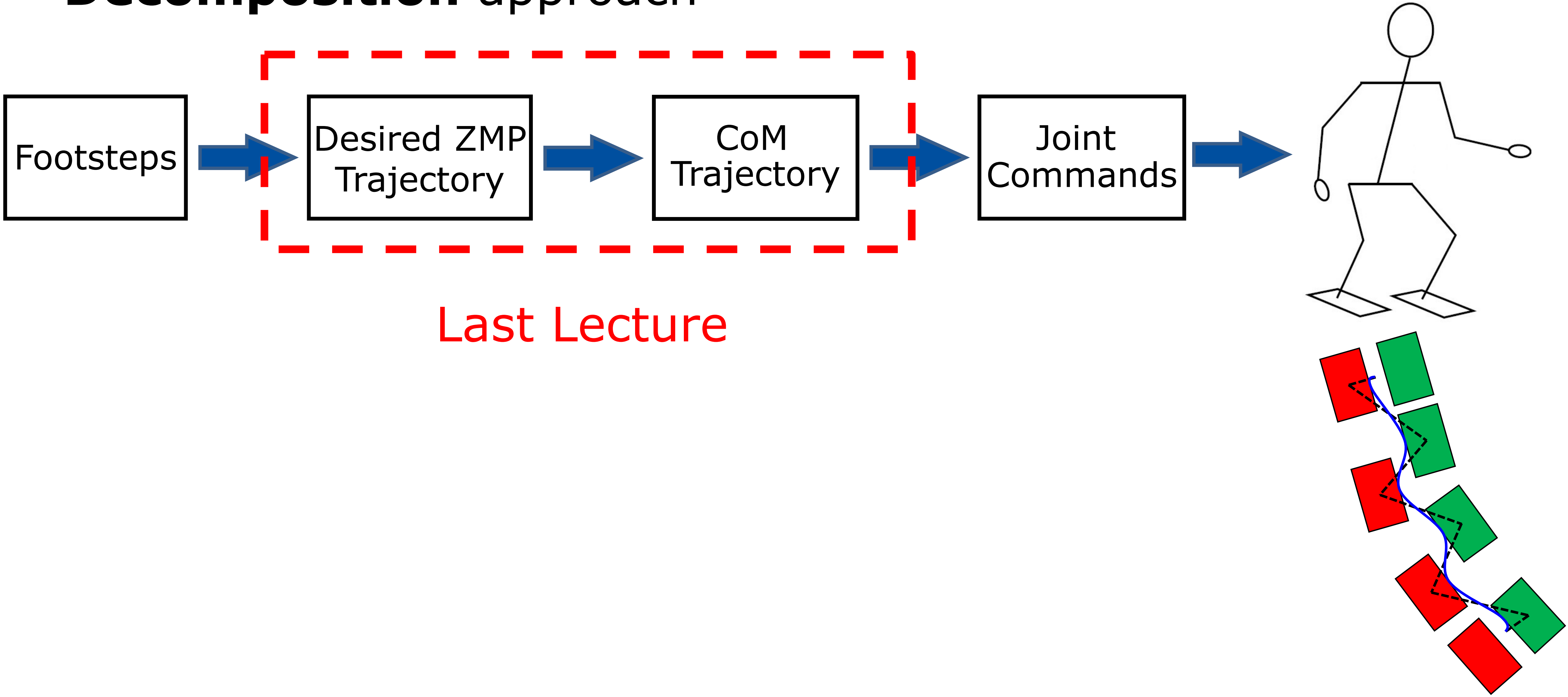
UNIVERSITÄT BONN

Goals of This Lecture

- Learn about **footstep planning** in a **2D grid map**
- Understand footstep planning on **uneven terrain** with **convex optimization**
- Understand how planned motions are executed through **real-time full-body control**

Recap – Motion Planning

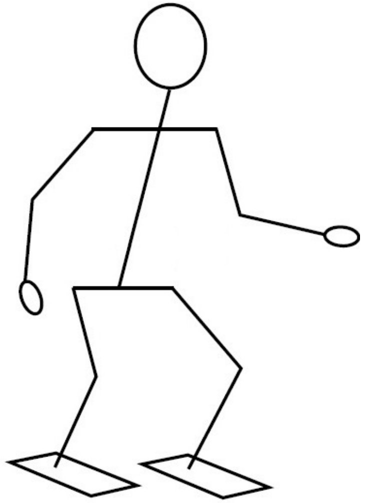
- **Decomposition** approach



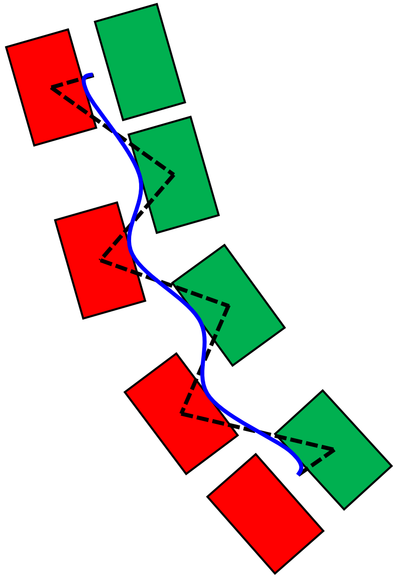
Recap – Motion Planning

- **Decomposition** approach

Footsteps → Desired ZMP Trajectory → CoM Trajectory → Joint Commands →



Now: Footstep Planning



Footstep Planning on Flat Terrain: Search-Based Methods

A*: Best First Search -- Recap

- Find a cost-optimal path to the goal state on a graph
- Expands the states according to the **evaluation function**: $f(n)=g(n)+h(n)$
- $g(n)$: **actual costs** from start to state n
- Heuristics $h(n)$: **estimated costs** from state n to the goal

[Cognitive Robotics/ Intro into MoRo]

A*: Node Expansion

- Generate successor states
- Compute their *f*-values
- Insert successors into the priority queue, or update their f-value if already in the queue

A*: General Procedure

- Expand states, starting from the current robot state
- Sort successor states in priority queue according to *f*-value
- In each iteration, **expand the state with the minimum *f*-value** until the **minimum is the desired goal state**
- Each node keeps a pointer to its parent node to keep track of how it was found
- Solution path can be reconstructed at the end

A*: Optimality Condition

- Optimal path = path with the minimum accumulated *g*-cost from the start to the goal state
- A* yields the **optimal path** if ***h* is admissible**
- Let h*(*n*) be the actual cost of the optimal path from *n* to the goal
- Heuristics *h* is **admissible** if the following holds for all *n*: $h(n) \leq h^*(n)$

A* Algorithm

```
OPEN = priority queue containing START
CLOSED = empty set
while lowest rank in OPEN is not the GOAL:
  current = remove lowest rank item from OPEN
  add current to CLOSED
  for neighbors of current:
    cost = g(current) + movementcost(current, neighbor)
    if neighbor in OPEN and cost less than g(neighbor):
      remove neighbor from OPEN, because new path is better
    if neighbor in CLOSED and cost less than g(neighbor):
      remove neighbor from CLOSED
    if neighbor not in OPEN and neighbor not in CLOSED:
      set g(neighbor) to cost
      add neighbor to OPEN
      set priority queue rank to g(neighbor) + h(neighbor)
      set neighbor's parent to current

reconstruct reverse path from goal to start
by following parent pointers
```

touched states

already expanded states

expand state

calculate f-value of successor

state touched before, check g-cost

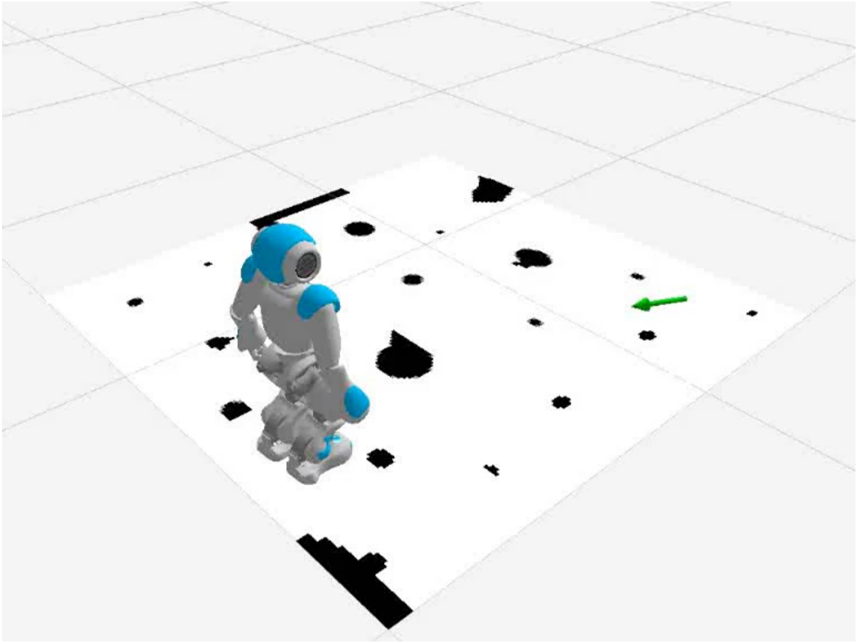
does not happen with admissible h

add state with its (new) cost

source: http://theory.stanford.edu/~amitp/GameProgramming/ImplementationNotes.html

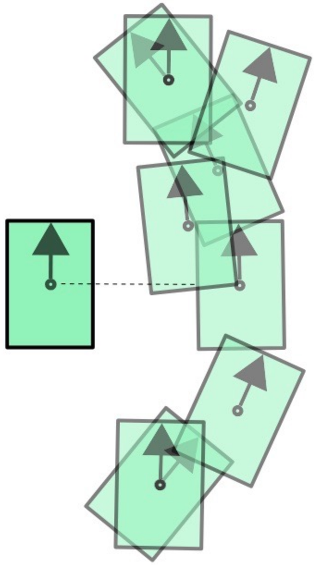
Footstep Planning with A*

- Search space: foot poses (x,y,θ)
- State: position and orientation of the stance foot
- Given: **discrete set of possible footsteps** wrt. stance foot
- Goal: Find the optimal footstep path to the goal state with A*



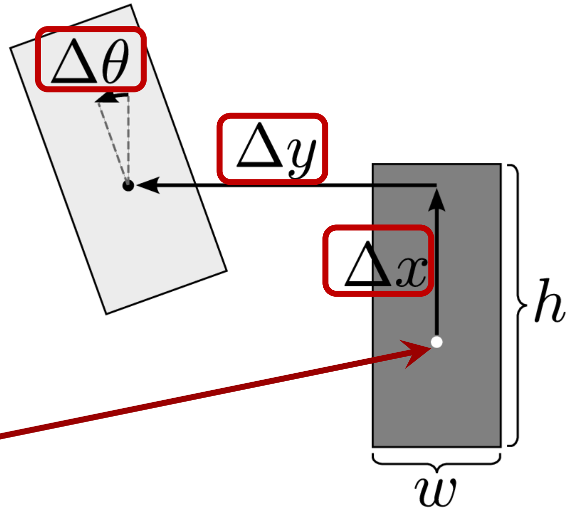
Footstep Planning with A*

- Fixed set of footsteps covering the reachable region
- Construct a search tree of successor states, starting from the current state with A*
- Check foot placements for collisions with obstacles in grid map during expansion



N. Perrin, "Biped footstep planning", Humanoid Robotics: A Reference, 2018

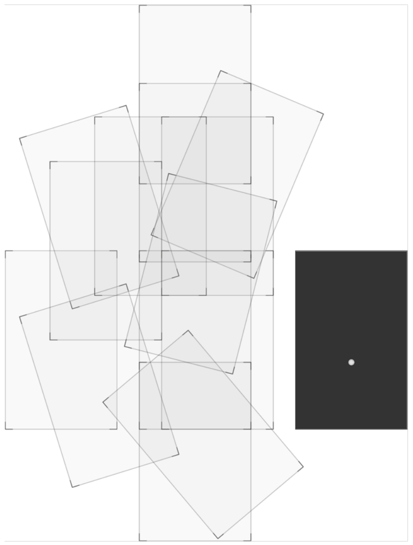
Footstep Actions & Costs



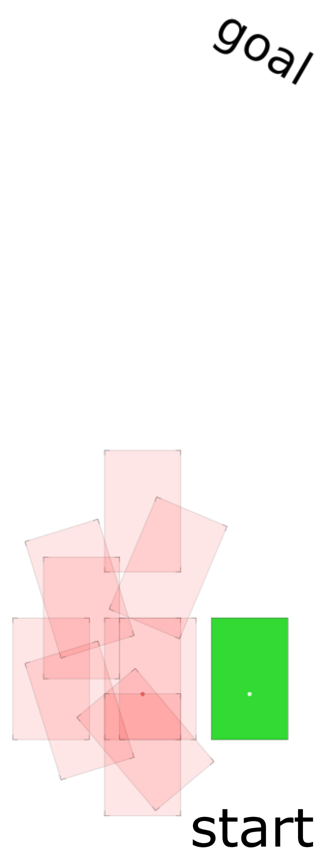
- State $s = (x, y, \theta)$
- Footstep action $a = (\Delta x, \Delta y, \Delta\theta)$
- Fixed set of footsteps $F = \{a_1, \ldots, a_n\}$
- Successor state $s' = t(s, a)$
- Transition costs:

$$\boxed{c(s, s') = \|(\Delta x, \Delta y)^\top\|}$$

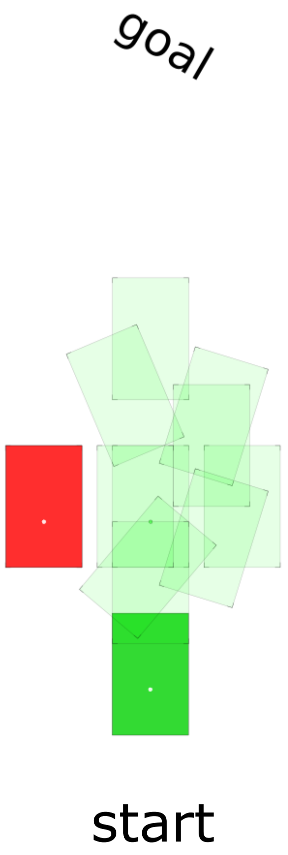
used to calculate g-value



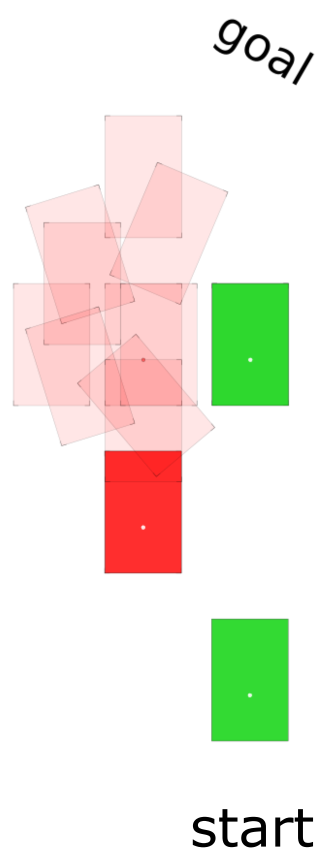
Footstep Planning



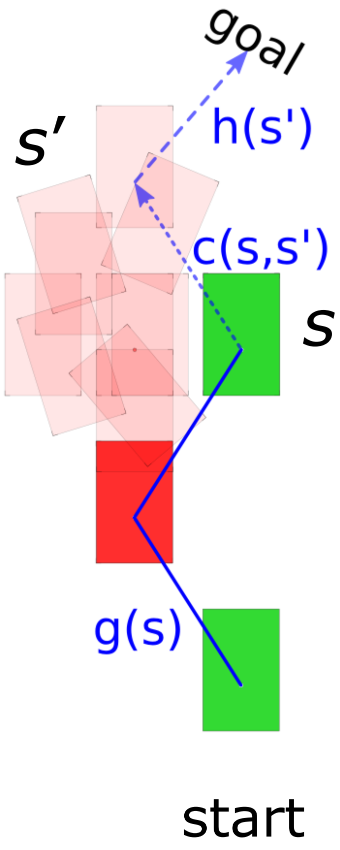
Footstep Planning



Footstep Planning



Footstep Planning



estimated costs from *s′* to goal

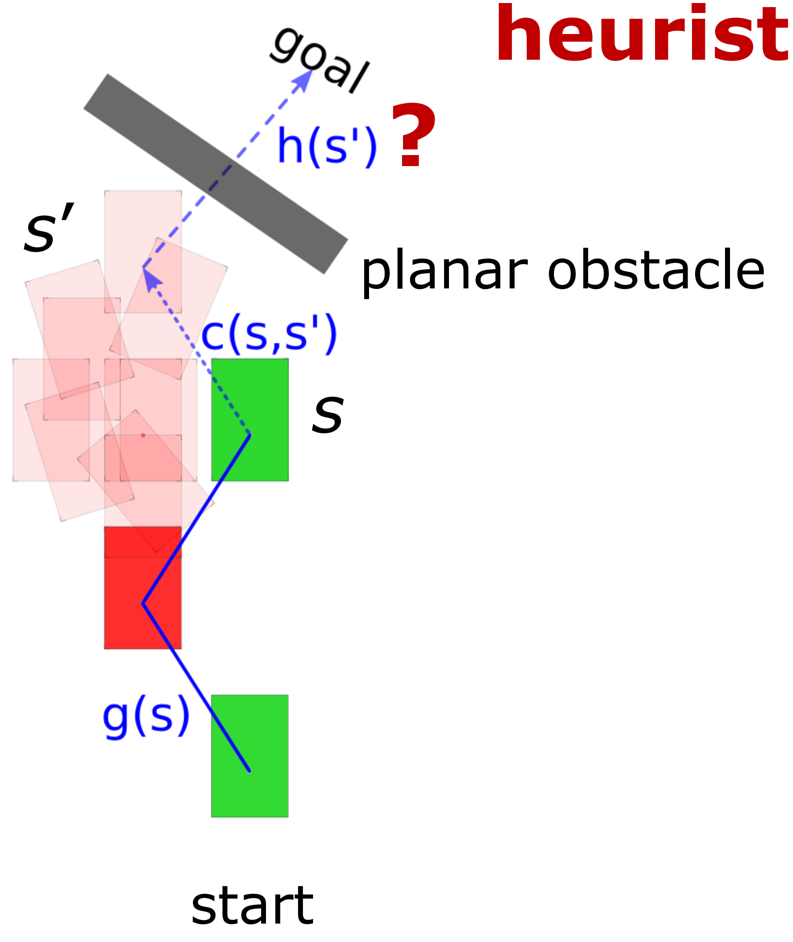
transition costs

path costs from start to *s*

Footstep Planning

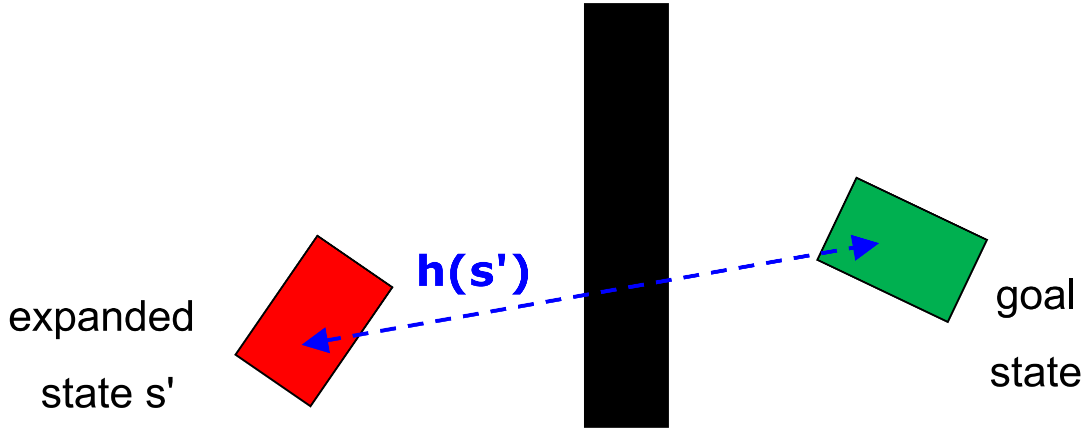
obstacles can create local minima in the search space

What is a good heuristic function?



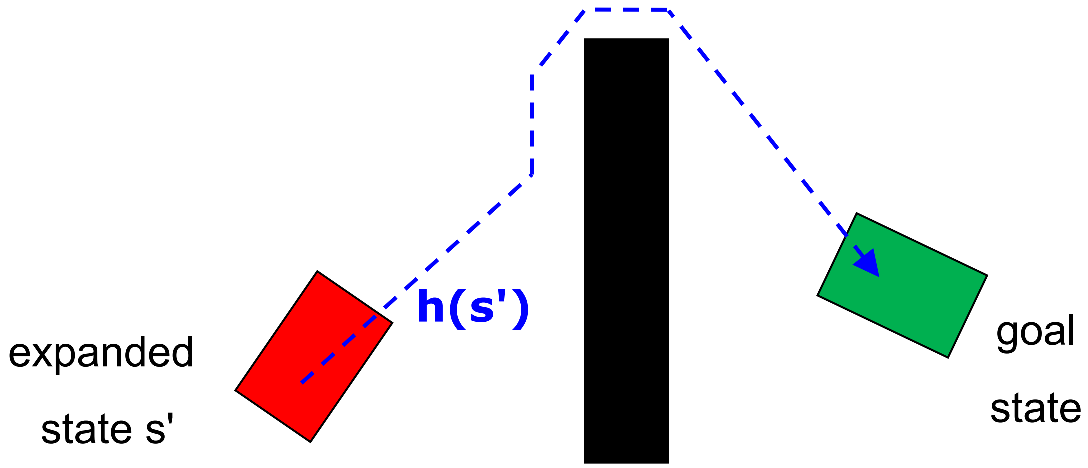
Heuristics for Footstep Planning

- Heuristic highly **influences A* performance**
- Estimates the costs to the goal
- Typical heuristic functions are based on:
  1. Euclidean distance (straight line)



Heuristics for Footstep Planning

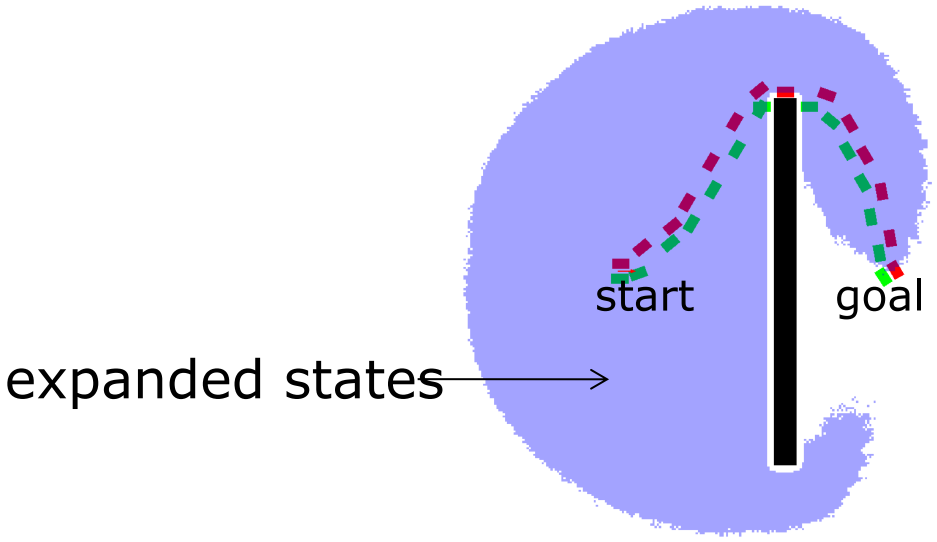
- Heuristic highly **influences A* performance**
- Estimates the costs to the goal
- Typical heuristic functions are based on:
  1. Euclidean distance (straight line)
  2. Shortest 2D path with safety margin around obstacles



- Shortest 2D path heuristic is inadmissible for humanoids
- Leads directly around obstacles, which is of advantage for **large** objects

Local Minima in the Search Space

- A* searches for the optimal path
- But sub-optimal results are often sufficient for navigation



With Euclidean distance
as heuristics

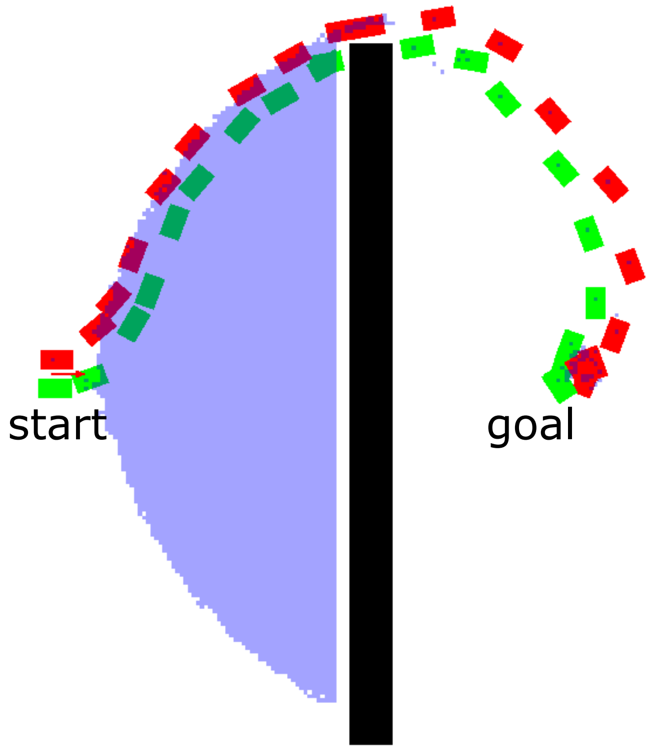
Anytime Repairing A* (ARA*)

- Weighted A* (wA*): Heuristics **“inflated” by a factor w**
- ARA* runs a **series of wA* searches**, iteratively lowering w as long as a given time limit is not met
- Resulting paths are **guaranteed** to cost no more than w times the costs of the optimal path
- ARA* reuses information from previous searches

Anytime Repairing A* (ARA*)

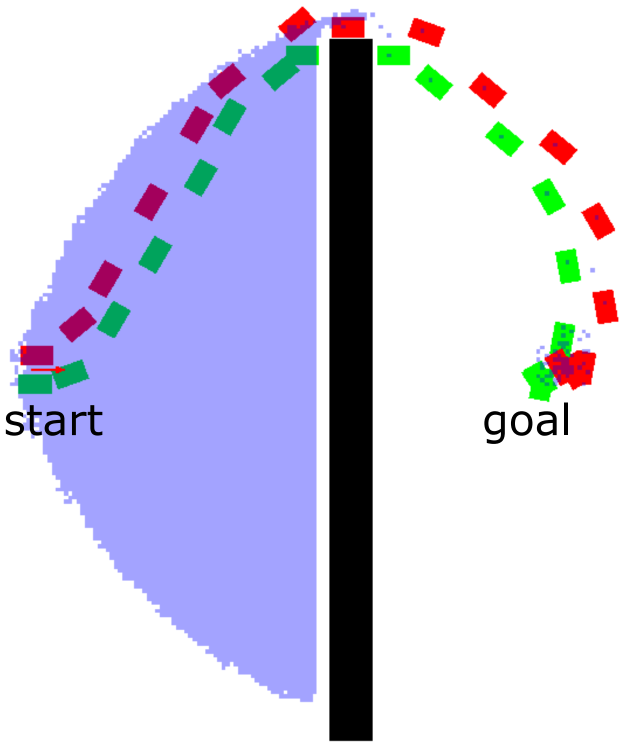
- An **initially large w** causes the search to find an initial non-optimal solution quickly
- Given enough time, ARA* finally searches with w = 1, producing an optimal path
- If time limit is met before, the cost of the best solution found is guaranteed to be **no worse than w times** the optimal solution
- ARA* finds **first sub-optimal solution faster** than standard A* and **approaches optimal one** as time allows

ARA* with Euclidean Heuristic



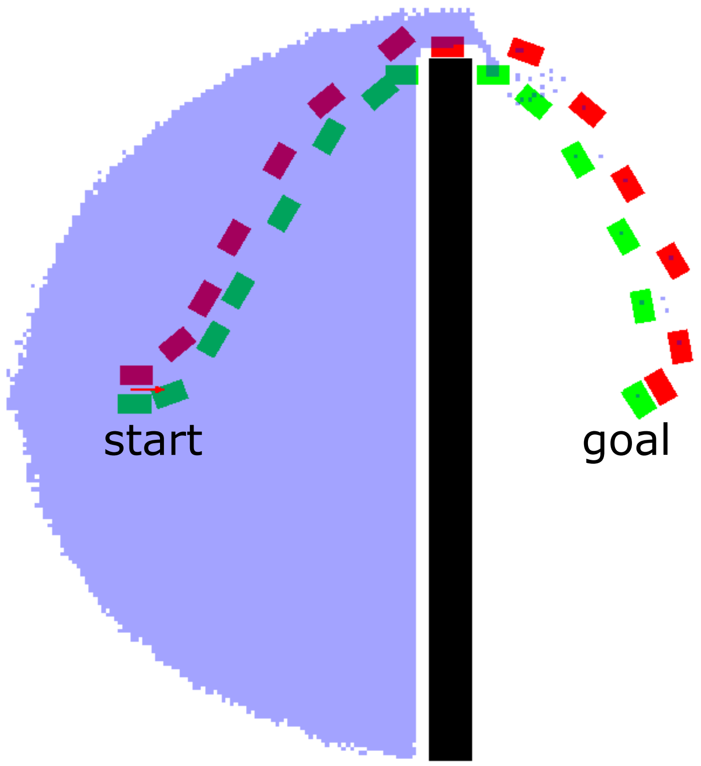
w = 10

ARA* with Euclidean Heuristic



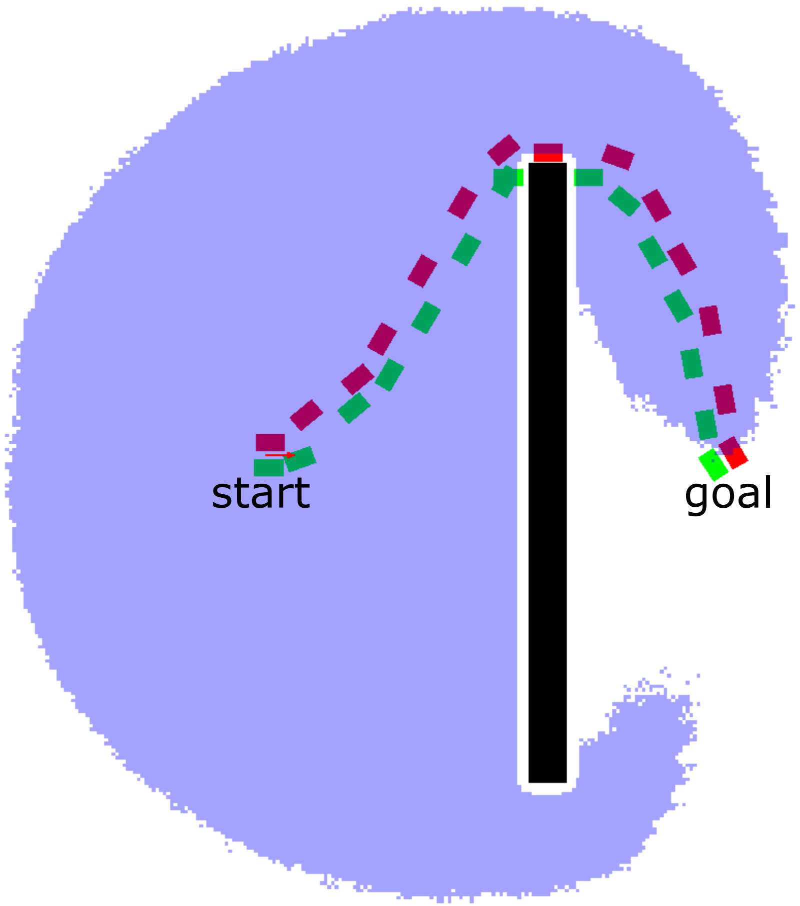
$w = 5$

ARA* with Euclidean Heuristic



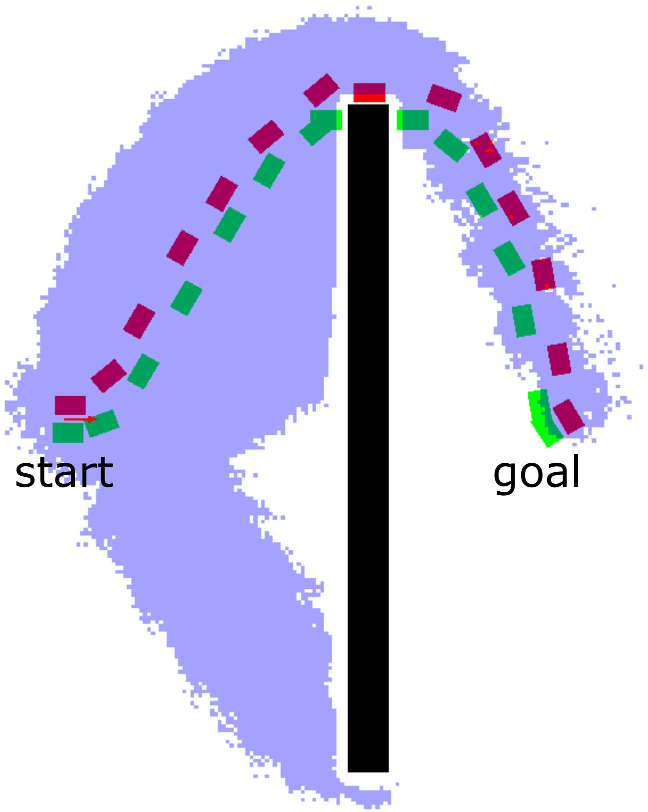
w = 2

ARA* with Euclidean Heuristic



w = 1

ARA* with Shortest 2D Path Heuristics

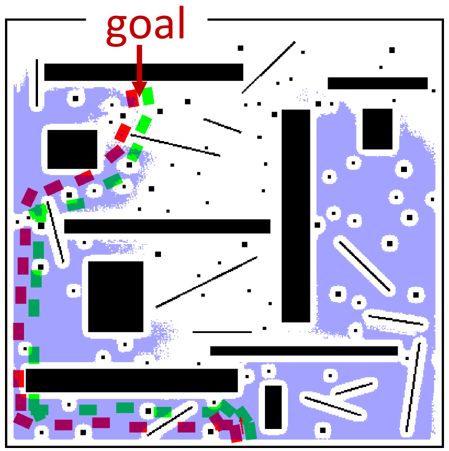


initial w=10

final w = 1

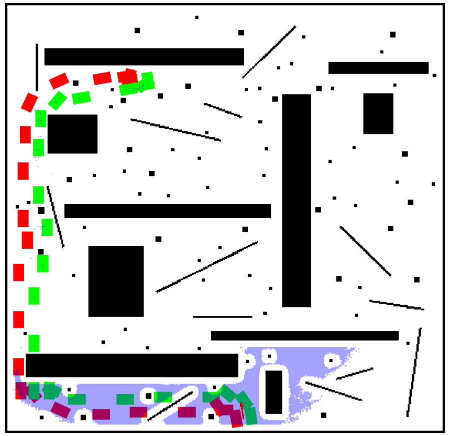
Planning in Dense Clutter Until a First Solution is Found

A*
Euclidean heuristics



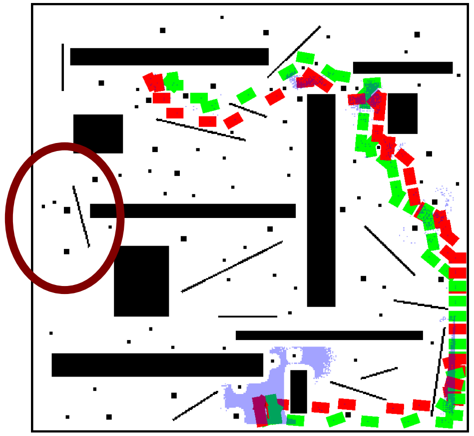
very high
planning time

ARA* (w=5)
Euclidean heuristics



lower
planning time

ARA* (w=5)
Shortest 2D path



very low
planning time

2D path heuristics:

- inadmissible and leads to longer path
- but expands much fewer states

Solution Quality and Search Time Depends on Heuristics

- Shortest 2D path heuristics might **overestimate costs** through narrow cluttered regions
- Shortest 2D path heuristics might **lead to longer paths** but much **fewer expanded nodes** than Euclidean heuristics
- "Erosion" of oversteppable objects from the map can be a solution, i.e., preprocessing of grid map
- **Trade off** between **planning time** and **solution quality**

Footstep Planning on Uneven Terrain: Optimization Methods

Why Optimization?

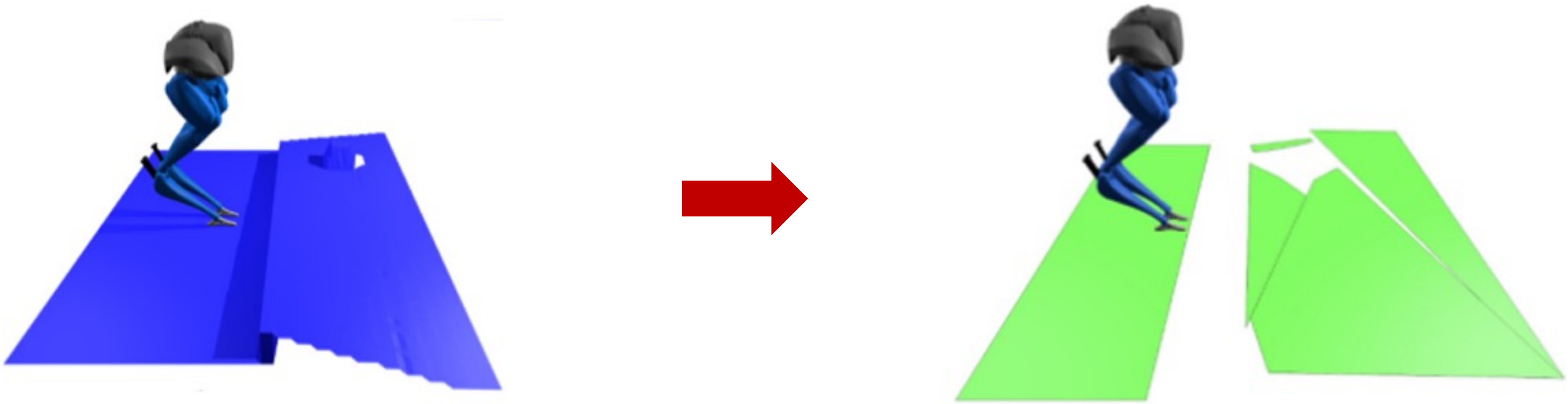
- Handle **complex** environments and uneven terrain
- Heuristics are **hard to scale** and often fail in non-trivial settings
- Need to consider **constraints and models**, e.g., terrain geometry, robot kinematics, etc.
- **Generalization** to diverse terrain types, e.g., sloped, stairs, or rough terrains
- Principled decision-making under **physical and geometric** constraints

Footstep Planning on Uneven Terrain - Challenges

- Imperfect perception and state estimation
- Need to extract **safe regions** for online footstep planning from perception
- General terrain shape is **non-convex**
- Find a **segmentation** representing **steppable regions**
- Need to extract **convex hulls**, so that footstep planning can be formulated as convex optimization

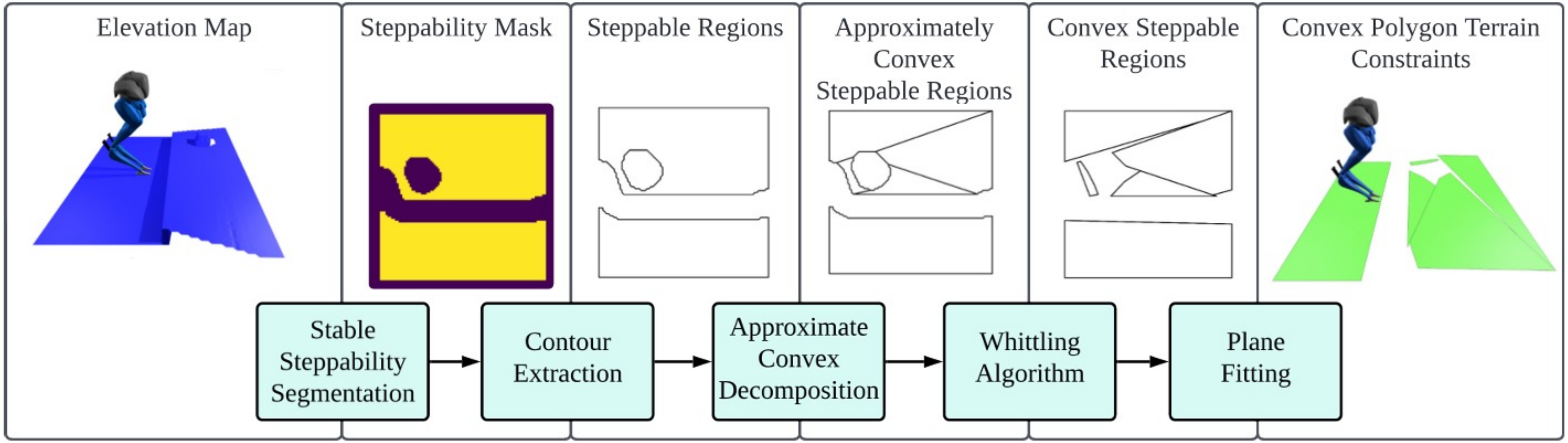
Perception on Rough Terrain

- Idea: Convert an elevation map to **convex polygons** representing terrain constraints, i.e., steppable areas



Acosta & Posa, "Perceptive mixed-integer footstep control for underactuated bipedal walking on rough terrain", arXiv, 2025

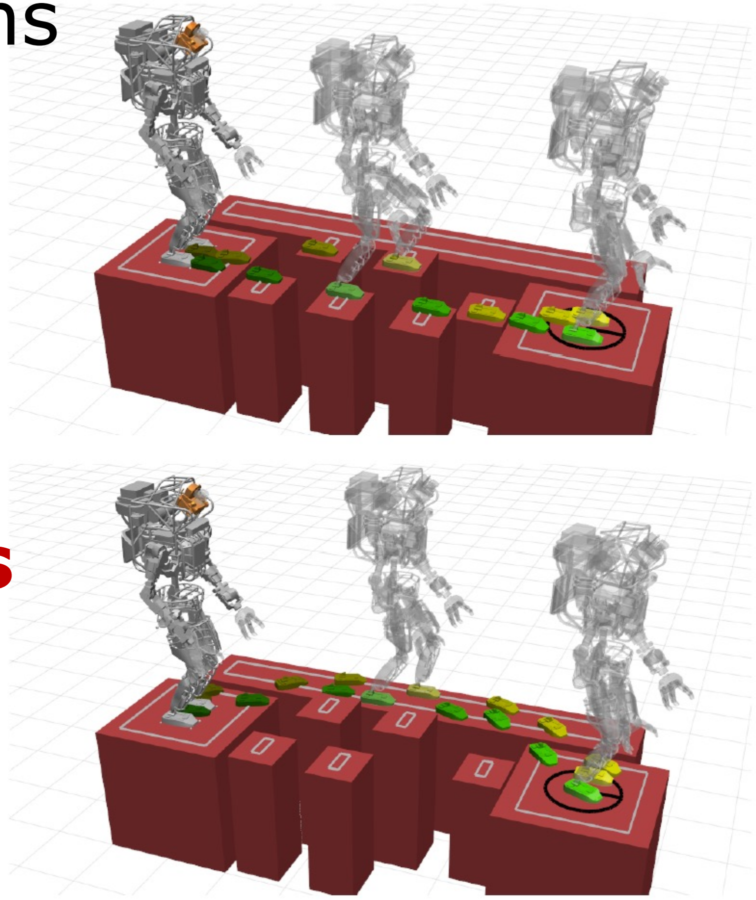
Perception on Rough Terrains



Acosta & Posa, “Perceptive mixed-integer footstep control for underactuated bipedal walking on rough terrain”, arXiv, 2025

Footstep Planning with Convex Regions

- Mixed-Integer Convex Optimization (MICP)
- Preprocess terrain into convex safe regions
- How MICP works:
  1. Select **valid foot placements** inside preprocessed safe zones
  2. Plan toward the goal while respecting **reachability** and **terrain constraints**



Deits & Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization", Humanoids, 2014

Footstep Planning with Convex Regions

- Convex region extraction: segment the terrain into **safe, flat convex regions**
- Step assignment via **integer variables**: each footstep is constrained to fall into **one of the convex regions**
- **Optimization problem**:
  - Minimize cost (e.g., distance to goal)
  - Considering constraints (footstep lies within assigned region, kinematic reachability, physics and geometry)

Footstep Planning with Convex Regions

- Determine the position $p = (x, y, z)$ and orientation θ of N footsteps, subject to the **constraints** so that:
  1. Each step does not intersect any obstacle
  2. Each step is within some convex reachable region relative to the pose of the prior step
- Formulate the problem as **assigning each footstep** to one of the precomputed **safe convex regions** of obstacle-free terrain

Footstep Planning with Convex Regions

- Simplified **objective function**:

$$\min_{p_i, \theta_i, z_{ij}} \sum_{i=1}^{N} \left(\|p_i - p_{i-1}\|^2 + c_{\text{turn}}(\theta_i, \theta_{i-1}) + c_{\text{goal}}(p_i) \right)$$

- Encourages **short, smooth, goal-directed** paths
- Turning cost **penalizes heading changes**
- Goal cost **encourages progress**
- z_{ij}: **binary indicator** if step i is in convex region j

Footstep Planning with Convex Regions

Constraints

1. Footstep in feasible convex region

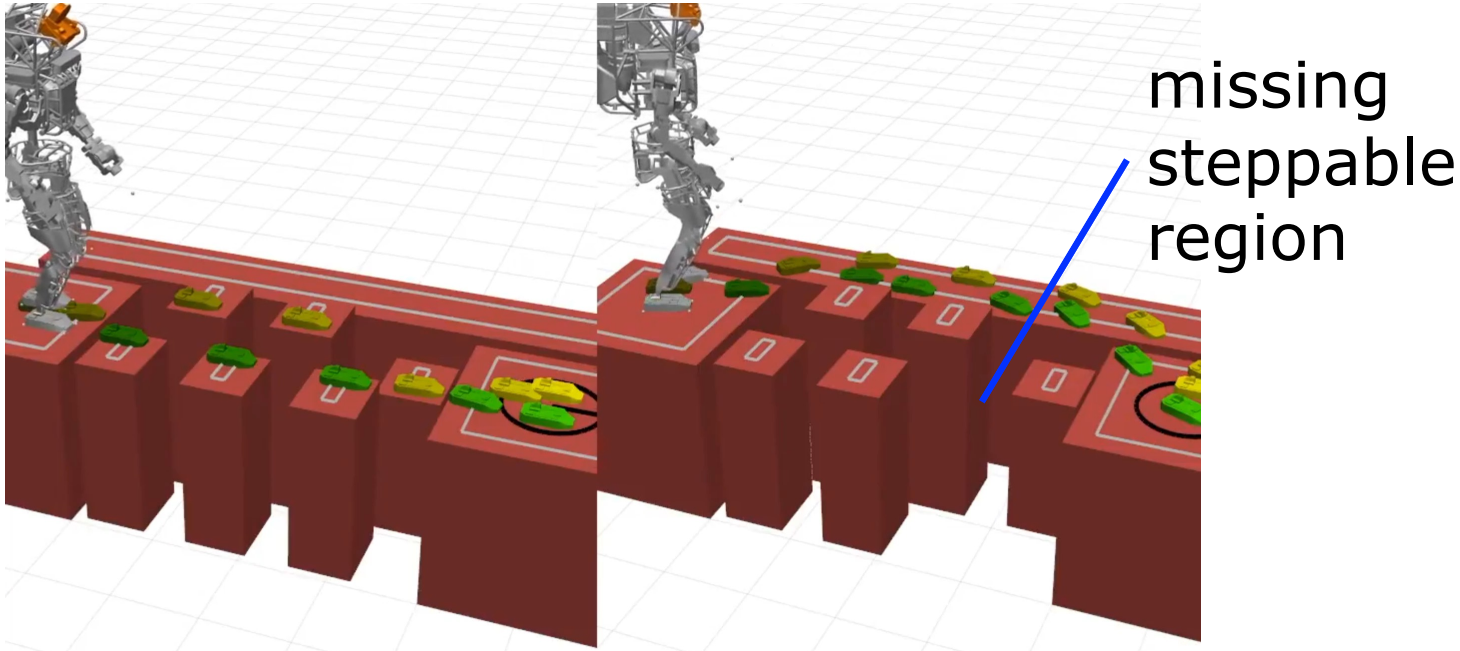
$$A_j p_j \leq b_j \text{ only if } z_{ij} = 1$$

   - A_j, b_j : define polygon j from perception
   - Planner selects which region to step onto via z_{ij}
2. Step reachability constraints
   - Enforces maximum step size and turning angle

$$\|p_i - p_{i-1}\|_2 \leq d_{\max}, \quad \angle(\theta_i - \theta_{i-1}) \leq \theta_{\max}$$

Footstep Planning with Convex Regions

- Handles arbitrary terrain
- Adaptable to changing convex regions
- Can incorporate different constraints
- But trade-off between computation time and generality



Deits & Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization", Humanoids, 2014

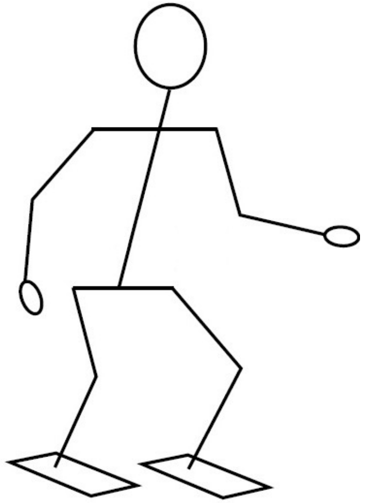
Realizing Motion Plans: Whole-Body Control

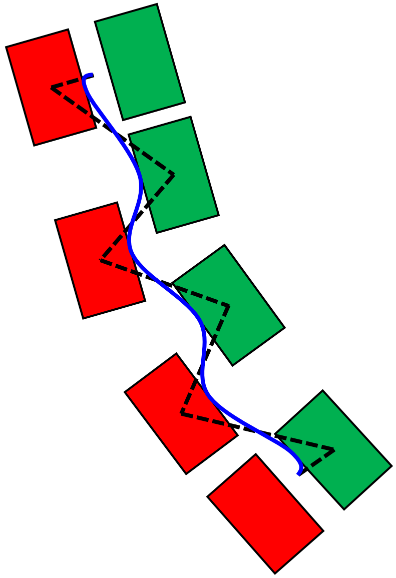
Recap – Motion Planning

- **Decomposition** approach

Footsteps → Desired ZMP Trajectory → CoM Trajectory → Joint Commands

Realizing motion plans





Whole-Body Control

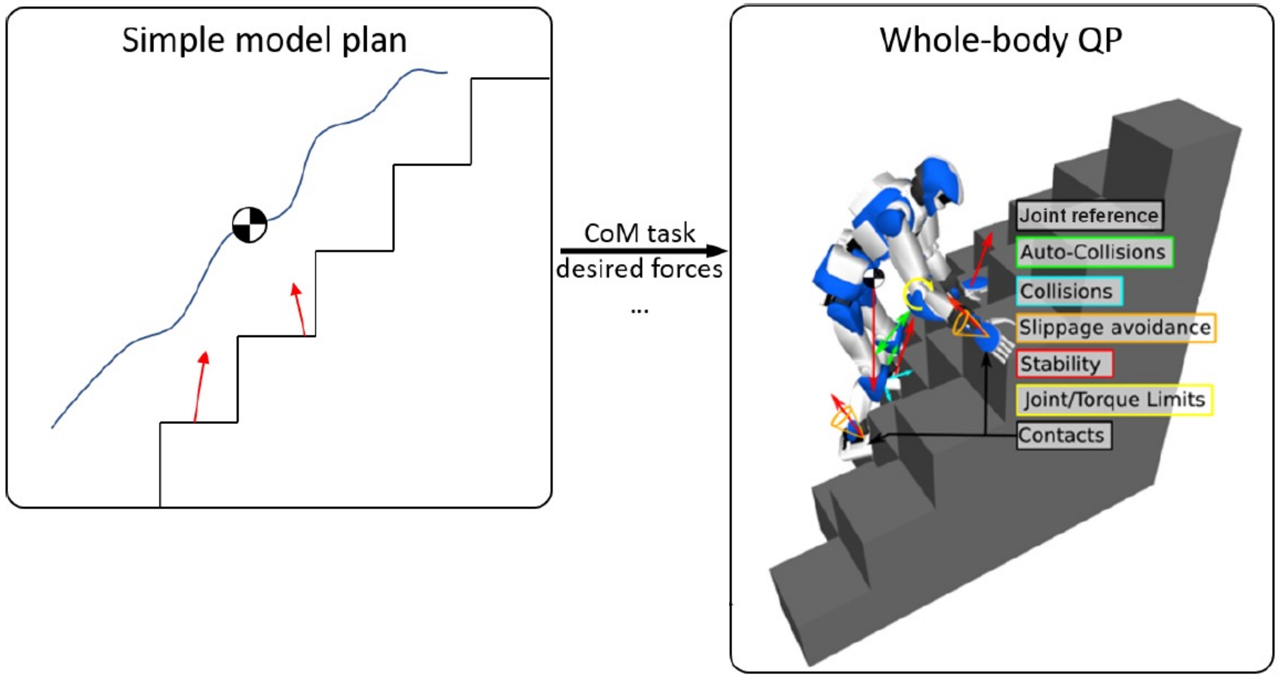
- Footstep planning and MPC generate desired footsteps and center of mass trajectory
- **Complex** optimization problems may be **too slow for real-time** control loops
- **Stabilizing controller** needed to execute and adapt the motion on the robot
- Whole-body control enforces constraints **ignored during planning** (e.g., joint limits, torque bounds, self-collisions, contact forces)

Whole-Body Control

- Formulated as **Hierarchical Quadratic Programming** (HQP)
- Use small convex QPs to compute joint torques or accelerations from state feedback
- Runs at high frequency (500–1,000 Hz) for real-time control

Whole-Body Control (WBC) using HQP

- Efficiently handles multiple constraints, e.g., COM trajectory, contacts, collisions, joint limits, torque bounds
- Enables **prioritization** of tasks (e.g., balance > foot tracking > posture)



Wensing et al.,"Optimization-Based Control for Dynamic Legged Robots", IEEE Transactions on Robotics, 2024

WBC Formulation

- Prioritized task space control as **optimization**
- Many control tasks (e.g., end-effector motion, contact forces, joint torques) can be written as **linear equations**
- Formulated as **least-squares problems**
- General task takes the form: $A_i\, x = \ b_i$, where x is the control variable (e.g., joint accelerations or torques)
- Tasks are **stacked by priority levels**, with $i = 1$ being the highest priority

WBC Formulation

- Each task is solved as:

$$\min_{\boldsymbol{x}} \ \left\| A_i \boldsymbol{x} - \boldsymbol{b}_i \right\|^2$$

- While **preserving** the solutions of **higher-priority tasks**
- Leads to a **hierarchical optimization** approach
- Can be solved with:
  1. Iterative null-space projections (solution of each task has to be inside the null-space of the previous task)
  2. Sequential Quadratic Programs (QPs)

WBC Formulation

- Sequential quadratic programs
- Each control task is formulated as a least-squares objective
- Lower-priority tasks are solved **subject to equality constraints** from higher-priority tasks

$$\min_{\boldsymbol{x}} \ \|A_i \boldsymbol{x} - \boldsymbol{b}_i\|^2 \text{ subject to } A_1 \boldsymbol{x} = \boldsymbol{b}_1 \ \ldots A_{i-1} \boldsymbol{x} = \boldsymbol{b}_{i-1}$$

- Enables structured and interpretable control by **preserving task priorities**
- Can be solved efficiently using **standard QP solvers**

Simple Example

- Control a **3-DoF planar** robotic arm with:
  1. **Task 1** (high priority): Move the end-effector to a target position
  2. **Task 2** (low priority): Control the joint angles to keep a desired posture
- Control end-effector position in 2D (x, y) → 2 constraints
- Use the remaining 1 DoF for the secondary task (joint posture)
- Define $\boldsymbol{x}$ as the vector of joint accelerations $\ddot{\boldsymbol{q}}$

Minimal Example

Task 1 (end-effector position)

- **Goal:** Move the end-effector to a target position

$$\min_{\ddot{\boldsymbol{q}}} \left\| A_1 \ddot{\boldsymbol{q}} - \boldsymbol{b}_1 \right\|^2$$

- A_1: Matrix that relates joint accelerations to the **end-effector's position**
- b_1: target position

Minimal Example

Task 2 (posture control)

- **Goal:** Achieve a desired posture (joint angle targets)

$$\min_{\ddot{\boldsymbol{q}}} \left\| A_2 \ddot{\boldsymbol{q}} - \boldsymbol{b}_2 \right\|^2$$

- A_2: Matrix that relates joint accelerations to the **joint positions**
- b_2: desired posture

Minimal Example – Hierarchical QP Formulation

- Solve Task 2 (posture) while preserving Task 1 (end-effector)

$$\min_{\ddot{\boldsymbol{q}}} \|A_2\ddot{\boldsymbol{q}} - \boldsymbol{b}_2\|^2 \text{ subject to } A_1\ddot{\boldsymbol{q}} = \boldsymbol{b}_1$$

- Here, Task 1 is enforced as a constraint to ensure that the **end-effector position is prioritized** over the posture

Minimal Example – Interpretation

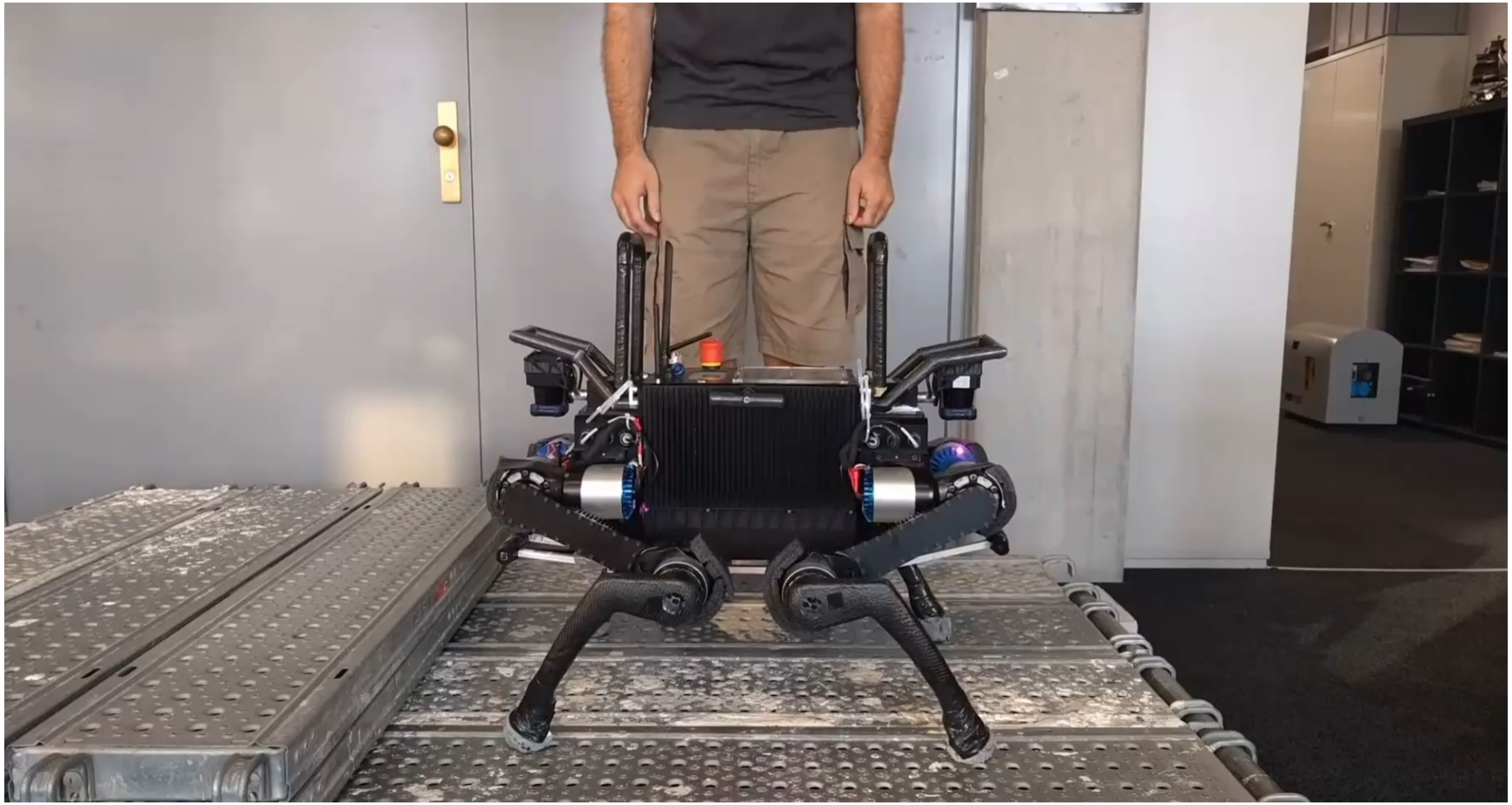
- The first task (end-effector position) has higher priority, so it is enforced as a **hard constraint**
- The second task (posture control) has lower priority, and we **minimize the error** for this task **while respecting the first task's constraint**

- Constrained optimization problem, and we can solve it using a **standard QP solver** to find the joint accelerations $\ddot{q}$ that satisfy both tasks

Implementation on Legged Robots

- **Complex locomotion behaviors** from low-level control, by combining multiple inequality and equality tasks in a hierarchical optimization
- **Robot-specific constraints** (e.g., torque limits, joint limits, …) directly encoded as inequality constraints
- **No explicit motion planning**, controller continuously adapts to terrain and contact conditions through task prioritization
- **Terrain adaptation** results from the structure of the control problem — not from high-level planning

Implementation on Legged Robots



Dario Bellicoso et al., “Perception-less terrain adaptation through whole body control and hierarchical optimization”, Humanoids, 2016

Summary

- Search-based footstep planning with **A* on flat terrain**
- **Heuristics** influence planning performance
- Footstep planning on **uneven** terrain with **Mixed-Integer Convex Optimization** (MICP)
- Terrain perception to extract convex **steppable regions** as constraints
- **Whole-body control** coordinates all limbs and joints through **hierarchical optimization**, enabling stable locomotion and terrain adaptation

Literature

- N. Perrin, “Biped footstep planning”, Humanoid Robotics: A Reference, 2018
- A. Hornung et al., “Anytime Search-Based Footstep Planning with Suboptimality Bounds”, Humanoids, 2012
- S. Boyd, L. Vandenberghe, “Convex optimization”, Cambridge University Press, 2004
- Moritz Diehl, “Numerical Optimization,” Lecture Notes, 2022, https://www.syscop.de/teaching/ws2022/numerical-optimization
- B. Acosta and M. Posa, “Perceptive mixed-integer footstep control for underactuated bipedal walking on rough terrain”, arXiv, 2025
- R. Deits and R. Tedrake, “Footstep planning on uneven terrain with mixed-integer convex optimization”, Humanoids, 2014
- O. Khatib, “A unified approach for motion and force control of robot manipulators: The operational space formulation”, IEEE Journal on Robotics and Automation, 1978
- Dario Bellicoso et al., “Perception-less terrain adaptation through whole body control and hierarchical optimization”, Humanoids, 2016