Humanoid Robotics

Path Planning and Walking

Maren Bennewitz
Introduction

- Given the robot’s pose in a model of the environment
- Compute a path to a target location
- First: 2D path in a 2D grid map representation of the environment
- Then: Footstep path in a 2D map or height map
2D Path Planning with A*

- Given: 2D occupancy grid map
- A* finds the cost-optimal path to the goal
- Considers the 8-connected neighborhood
- Starts with the cell containing the current pose of the robot
A* Heuristic Search

- Best-first search to find a cost-optimal path to the goal state
- Expands 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 $n$ to the goal
A* Heuristic Search

- 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* yields the optimal path if $h$ is admissible** (proof: AI lect., Russell/Norvig)
A* on a 2D Grid Map

- Generate successors (8-connected neighborhood) from node with lowest $f$-cost
- Prune nodes that result in collision
- Continue until the lowest-cost node is close to the goal
- Actual costs $g =$ costs of the path from the start to the current node
- 1 per horizontal/vertical grid cell traversal, $\sqrt{2}$ per diagonal
- Estimated costs to the goal $h$: straight-line distance
Example: Path Planning with A* on a 2D Grid Map
2D Path Planning for Humanoids

- Compute a collision-free 2D path first, then follow this path with a walking controller.
- For 2D path planning use a safety margin around obstacles.
- But how large should the margin be?

![Diagram showing large clearance (detour) vs. too small clearance (collision).]
Path Planning for Humanoids

- Humanoids can avoid obstacles by stepping over or close to them
- However, planning whole-body motions has a high computational complexity
- Footstep planning reduces the computational demand
Footstep Planning with A*

- Search space: foot poses \((x,y,\theta)\)
- Global position and orientation of the stance foot
- **Given: discrete set of footsteps**
- Optimal footstep path with A*
Footstep Planning with A*

- Planning of footsteps by constructing a search tree of potential successor states
- Fixed set of possible footsteps
- Foot placements are checked for collisions during tree expansion

source: Kuffner et al.
Footstep Planning

- 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:

$$c(s, s') = k + c_{\Delta x, y}$$

constant step costs

costs based on the traversed distance
Footstep Planning

start

goal
Footstep Planning
Footstep Planning
Footstep Planning

- $s'$: estimated costs from $s'$ to goal
- $h(s')$: transition costs
- $c(s,s')$: path costs from start to $s$
- $g(s)$: start
Footstep Planning

\begin{center}
\begin{tikzpicture}
  \node (start) at (0,0) {start};
  \node (s) at (2,1.5) {s};
  \node (s_prime) at (2,-1.5) {s'};
  \node (goal) at (4,3) {goal};
  \node (obstacle) at (2,0) {planar obstacle};

  \draw[blue, dashed] (start) -- (s);
  \draw[blue, dashed] (s) -- (s_prime);
  \draw[blue, dashed] (s_prime) -- (goal);
  \draw[blue, dashed] (start) -- (goal);

  \node[above] at (goal) {h(s') \?};
  \node[below] at (s) {g(s)};
  \node[below] at (s_prime) {c(s, s')};

\end{tikzpicture}
\end{center}
Heuristics

- Estimates the costs to the goal
- Critical for planner performance
- Usual choices based on:
  - Euclidean distance (straight-line distance)
  - Shortest 2D path (Dijkstra) with safety margin around obstacles
A*-Based Footstep Planning

- Only valid states after a collision check are added to the search tree
- Goal state may not be exactly reached, but it is sufficient to reach a state close by
Local Minima in the Search Space

- A* searches for the optimal path
- Sub-optimal results are often sufficient for navigation
Anytime Repairing A* (ARA*)

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

[Likhachev et al. ’04]
Anytime Repairing A* (ARA*)

- An initially large $w$ causes the search to find a non-optimal initial solution quickly.
- Given enough time, ARA* finally searches with $w = 1$, producing an optimal path.
- If the 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 a first sub-optimal solution faster than regular A* and approaches optimality as time allows.**
ARA* with Euclidean Heuristic

$w = 1$

[start] [goal]
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 2D Dijkstra Heuristic

w = 1
Planning in Dense Clutter Until a First Solution is Found

A* Euclidean heur.

ARA* (w=5) Euclidean heur.

ARA* (w=5) Dijkstra heur.

11.9 sec.

2.7 sec.

0.7 sec.

Dijkstra heuristics: inadmissible and may lead to longer paths, but expands much fewer states
Footstep Planning on a Heightmap

- So far: planar obstacles
- Now: Consider the height of objects
- Plan footsteps that also include stepping onto objects
- Use a modified Dijkstra heuristics
Action Set for the Nao Humanoid

Standard planar steps

Extended 3D stepping capabilities
Dijkstra Heuristics for Heightmaps

- Consider the 8-neighborhood in the \((x,y)\) space of the heightmap
- Costs of an edge are defined by the height differences

![Example costs diagram](image-url)
Safe Stepping Actions

- Allow only states where all cells covered by the footprint have a small height difference

- Height difference must be within the stepping limits
Efficient Whole-Body Collision Checking

- Pre-compute the volume covered by the robot when executing the footstep actions
- Use the minimum height values for fast collision checking within the height map
Efficient Whole-Body Collision Checking

- Store minimal height values of the robot (IHM=inverse height map)
- Use IHM for fast collision checking
- An action $\alpha$ can be executed collision-free in state $s$ if

$$\forall (u, v) \in IHM^a : IHM^a_{(u,v)} + z_s > h(u,v)$$

- $z_s$: height of stance foot at $s$
- $h(u,v)$: heightmap value
- Volume covered by the robot
Heightmap learned online from the robot's depth camera data
Adaptive Level-of-Detail Planning

- Planning the whole path with footsteps may not always be needed in large open spaces
- Adaptive level-of-detail planning: Combine fast grid-based 2D planning in open spaces with footstep planning near obstacles
Adaptive Planning Results

2D Planning
<1 s planning time
high path costs

Footstep Planning
29 s planning time

Adaptive Planning
<1 s planning time,
costs only 2% higher

Fast planning times and efficient solutions
with adaptive level-of-detail planning
Walking

HRP-4C and ASIMO
Posture Stability

A pose is **statically stable** if

- The robot’s **center of mass (COM)** lies above the support polygon on the floor
- **Support polygon**: the convex hull of all contact points of the robot on the floor

source: T. Asfour
Statically Stable Walking

- The robot will stay in a stable position whenever the motion is stopped

- The projection of the COM of the robot on the ground must be contained within the support polygon

- Support polygon:
  - Either the foot surface in case of one supporting leg, or
  - the minimum convex area containing both foot surfaces when both feet are on the ground
Statically Stable Walking

Leads to robust but slow walking performance

source: T. Asfour
Human Walking

- Body as an inverted pendulum pivoting around the ankle joint
- The body is represented as a point mass
- The body weight acts at the COM and generates momentum
- The ground reaction force acts at the center of pressure (COP)

source: T. Asfour
Zero Moment Point

- Stability is achieved if the zero moment point (ZMP) is in the support area
- A robot standing on the ground applies a force and moment to the ground
- At the same time, the ground applies a force and a moment to the robot (ground reaction force)
- The ZMP is the point on the ground where the total moment generated due to gravity and inertia equals zero
Zero Moment Point

\[ \vec{F} + \vec{G} + \vec{R} = 0 \]
Zero Moment Point
Zero Moment Point

\[ \vec{F} + \vec{G} + \vec{R} = 0 \]
\[ M_F + M_G + M_R = 0 \]
Zero Moment Point

\[ \vec{F} + \vec{G} + \vec{R} = 0 \]
\[ M_F + M_G + M_R = 0 \]

Zero-Moment-Point (ZMP)
Zero Moment Point

- For stable walking, the support foot must rest on the ground
- Forces and torques acting on support foot must sum up to 0
- The ZMP must remain inside the footprint of the support foot
- Then, the ZMP coincides with the COP
- ZMP must not be on the edge of the support polygon, this will cause instability
- During the movement, the projection of the COM can leave the support polygon
Dynamically Stable Walking

Stopping the motion may result in falling

source: S. Kajita
Following Footsteps

- Footsteps
  - Desired ZMP Trajectory
    - CoM Trajectory
      - Joint Angles

Pattern Generator (e.g. Kajita et al. 2003)
ZMP-Based Walking Pattern Generator

- Parameters: COM height, step speed, single/double support phase duration

- Calculate feet trajectory: polynomial trajectories with zero velocity and acceleration at start and end of each step
ZMP-Based Walking Pattern Generator

- Define **reference ZMP**:
  - In single support phase: always in center of foot
  - In double support phase: fast switch from previous standing foot to the next one

- Calculate **COM trajectory** to follow the foot placements and reference ZMP

- Solve inverse kinematics (IK) for given feet position and COM position to get the **leg joint angles** (IK will be detailed in the next chapter)
ZMP-Based Walking Pattern Generator

footstep positions

feet trajectory

reference ZMP

resulting ZMP

COM trajectory

source: T. Asfour
Summary

- Search-based path planning and footstep planning with A*
- Anytime search-based footstep planning with sub-optimality bounds: ARA*
- Heuristics influences planning performance
- Adaptive level-of-detail planning to combine 2D with footstep planning
- Extensions to 3D obstacle traversal
- Basic concepts of stability and walking
Literature

- Introduction to Humanoid Robotics, Ch. 3 (ZMP) Shuji Kajita et al., 2014
- Search-based planning library: www.ros.org/wiki/sbpl
- Footstep planning implementation based on SBPL: www.ros.org/wiki/footstep_planner
Acknowledgment

- Previous versions of parts of the slides have been created by Armin Hornung, Tamim Asfour, and Stefan Osswald