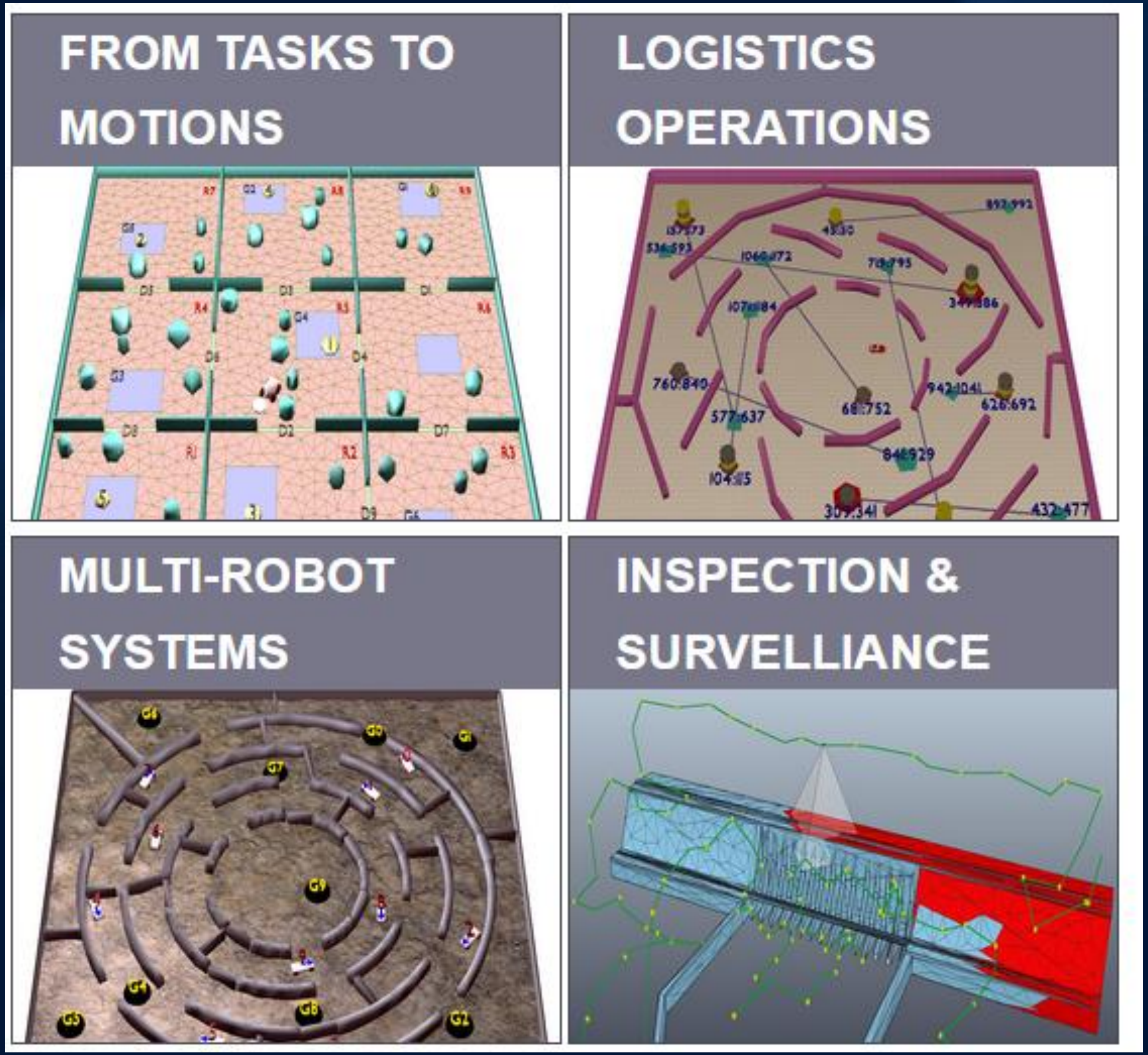
# MULTI-GOAL MOTION PLANNING

**Stefan Edelkamp**
**PUI**

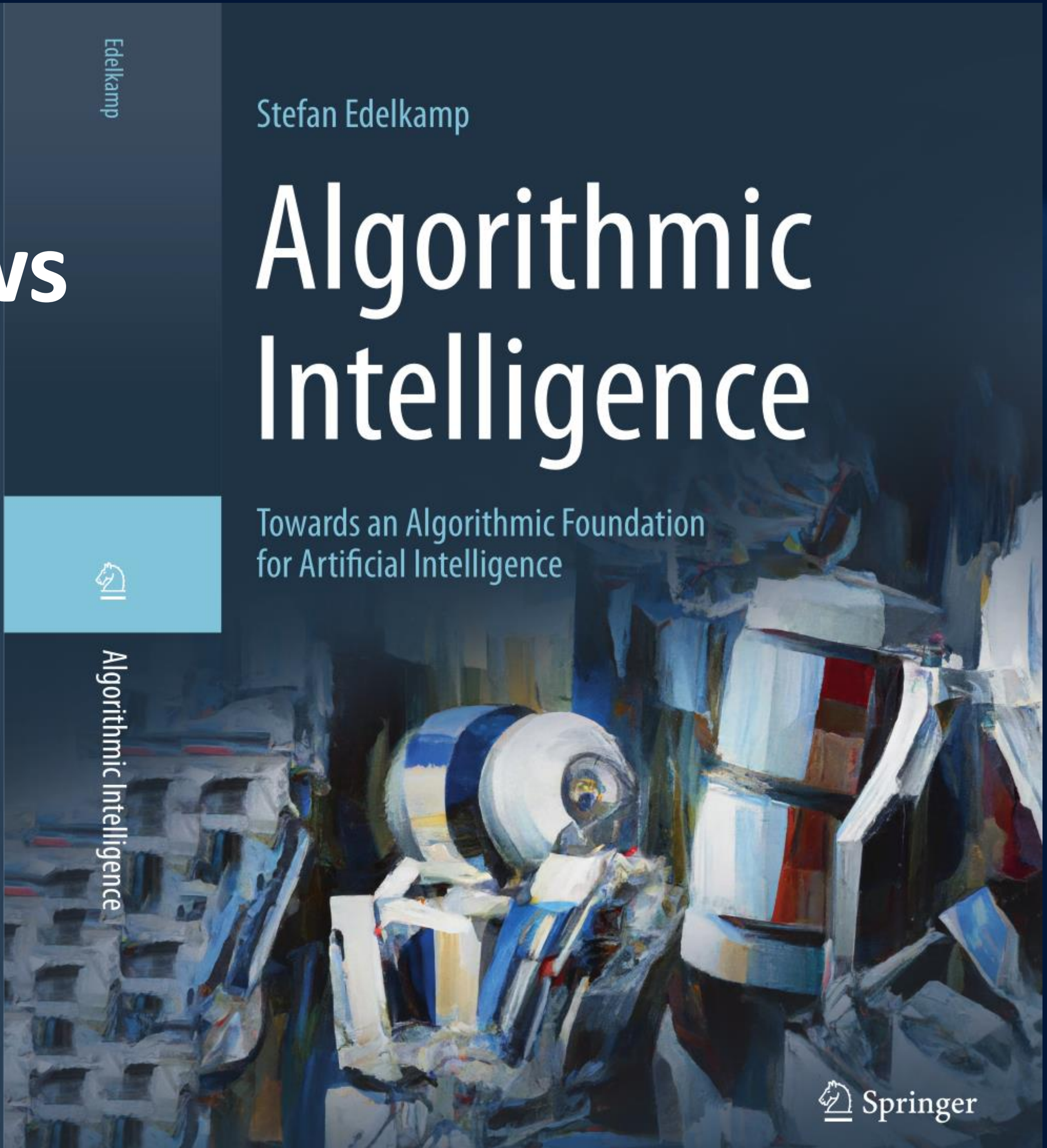
Artificial Intelligence Center

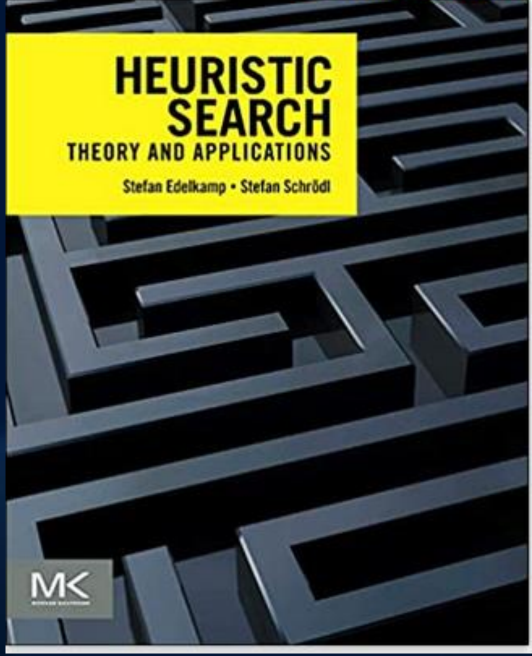
Faculty of Electrical Engineering

Czech Technical University in Prague



# News




- Demonstrates the algorithmic foundations of computer intelligence
- Integrates programming, theoretical computer science, optimization, machine learning, data mining, data analytics
- Covers searching, sorting and deep learning with applications to big data, games, biology, robotics, IT security

AI CENTER
FEE CTU

Stefan Edelkamp

# Algorithmic Intelligence

## Towards an Algorithmic Foundation for Artificial Intelligence

In this book the author argues that the basis of what we consider computer intelligence has algorithmic roots, and he presents this with a holistic view, showing examples and explaining approaches that encompass theoretical computer science and machine learning via engineered algorithmic solutions.

Part I of the book introduces the basics. The author starts with a hands-on programming primer for solving combinatorial problems, with an emphasis on recursive solutions. The other chapters in the first part of the book explain shortest paths, sorting, deep learning, and Monte Carlo search. A key function of computational tools is processing Big Data efficiently, and the chapters in Part II of the book examine traditional graph problems such as finding cliques, colorings, independent sets, vertex covers, and hitting sets, and the subsequent chapters cover multimedia, network, image, and navigation data. The highly topical research areas detailed in Part III are machine learning, problem solving, action planning, general game playing, multiagent systems, and recommendation and configuration. Finally, in Part IV the author uses application areas such as model checking, computational biology, logistics, additive manufacturing, robot motion planning, and industrial production to explain how the techniques described may be exploited in modern settings.

The book is supported with a comprehensive index and references, and it will be of value to researchers, practitioners, and students in the areas of artificial intelligence and computational intelligence.

# Two Meanings of Planning

AI CENTER
FEE CTU

## Long-Standing Challenge: Bridge the Gap

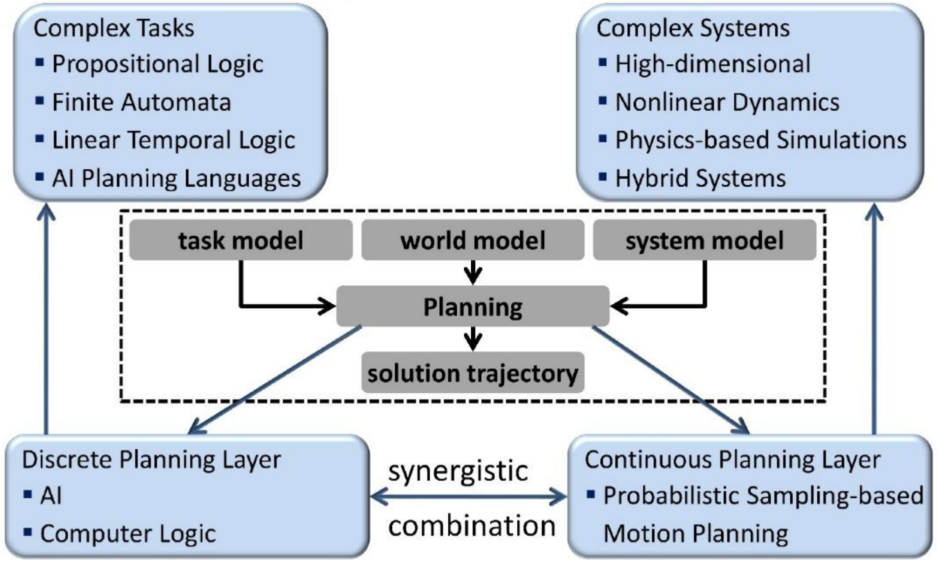
given a high-level task description compute the motions to enable the robot accomplish the task

**AI Planning**

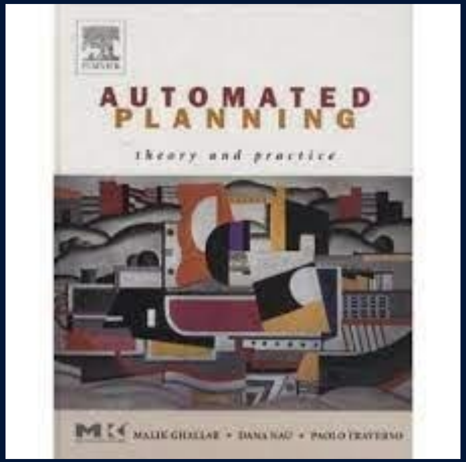
- ▷ Discrete world
- ▷ Discrete actions
- ▷ Complex task specifications

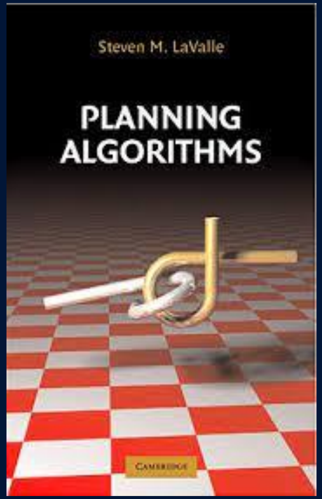
**Motion Planning**

- ▷ Continuous world
- ▷ Continuous controls/dynamics
- ▷ Simple task specifications




**Synergistic Combination of Discrete and Continuous Layers of Planning**



AI CENTER
FEE CTU

# Driving Research Questions


# Steer Robots of the 21st Century

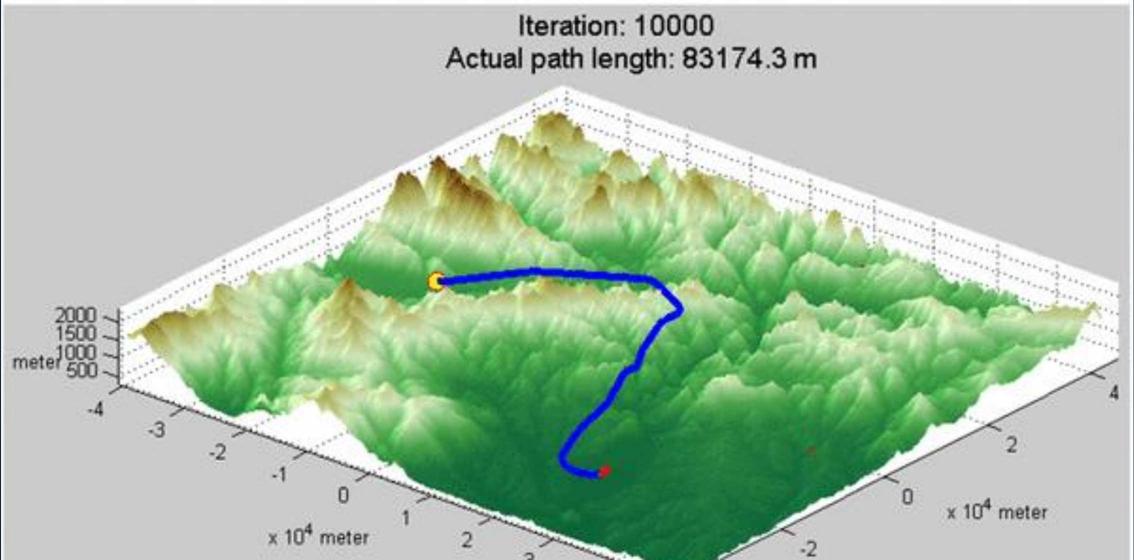
- How can we improve motion planning for complex systems?
- How can we develop motion planners that are generally applicable?
- How can we achieve planning efficiency even with nonlinear dynamics?
- How far back can we push the “curse of dimensionality”?
- How to integrate domain-independent planners for more flexible task planning?
- How to solve the inspection problem?
- How to combine efficient discrete solving with searching the continuous space of feasible motions?
- Is there Pareto optimality between efficiency and solution quality?
- What formal guarantees can we provide?
- How can we take resource and energy constraints into account?

delkamp

AI CENTER
FEE CTU

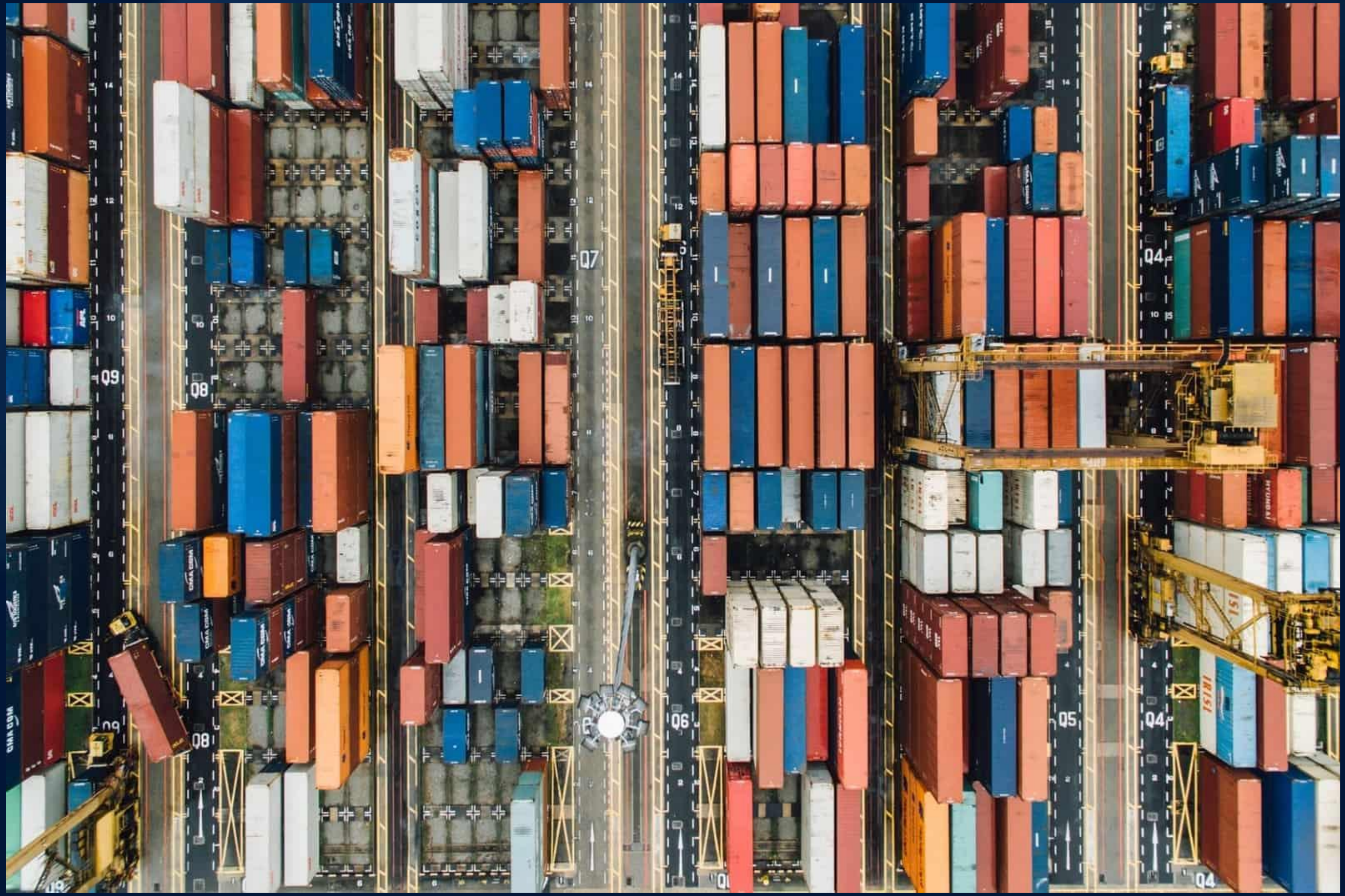
# Ongoing Work

SubT Challenge, Complex Resources, DeepRRT*, Emergency Planning, Location Routing, …



AI CENTER
FEE CTU

# Prelude: Planning Tours

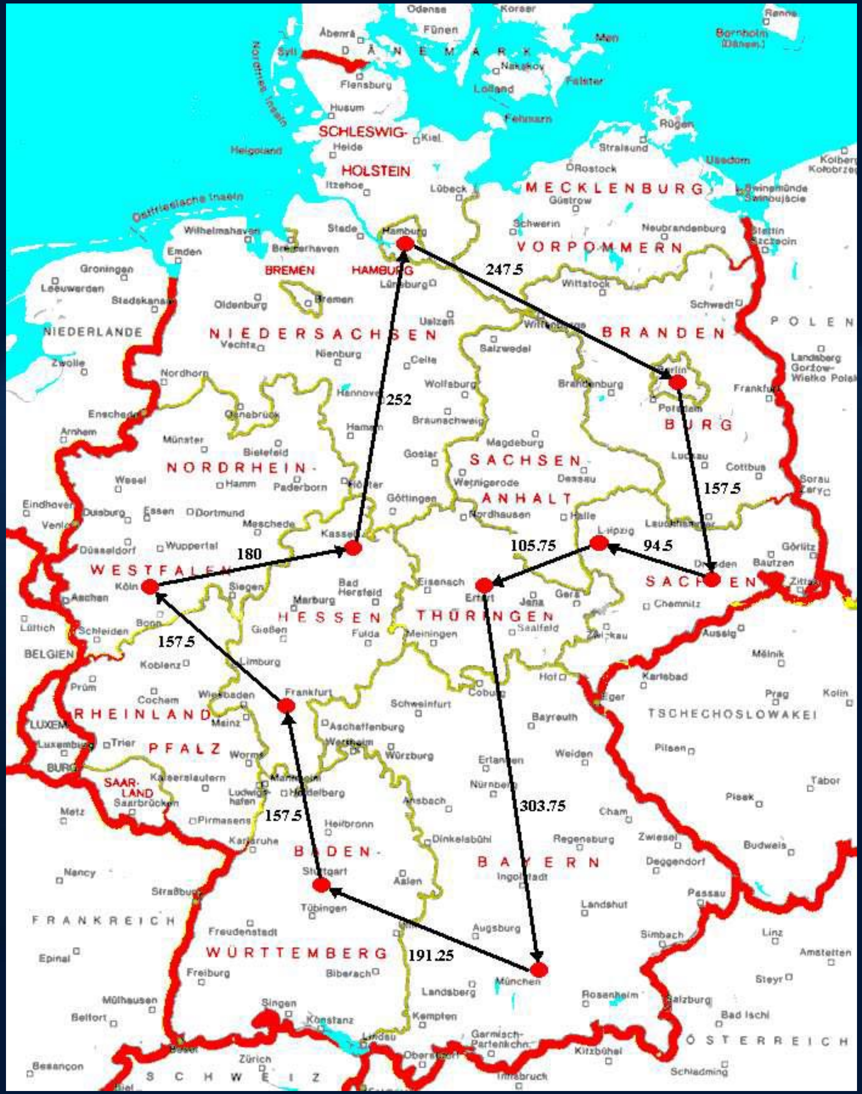
AI CENTER
FEE CTU

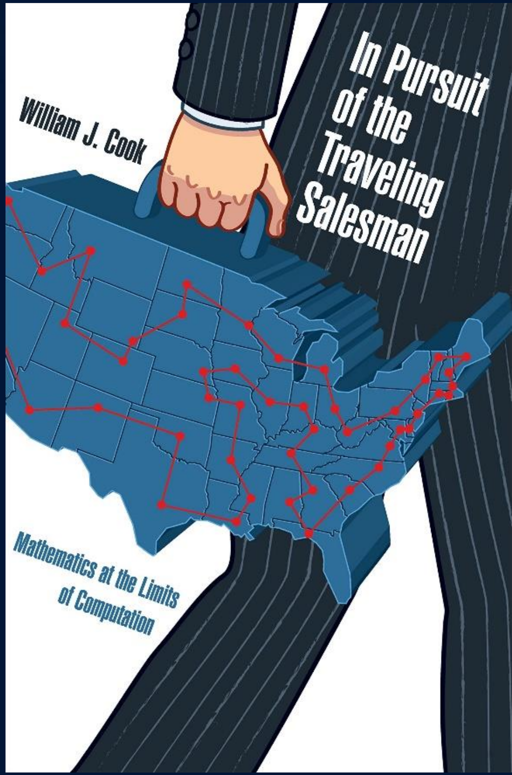
# Traveling Salesman Problem

Task

**Given a map, compute a minimum-cost round trip visiting certain cities**
**Shortest paths graph reduction**

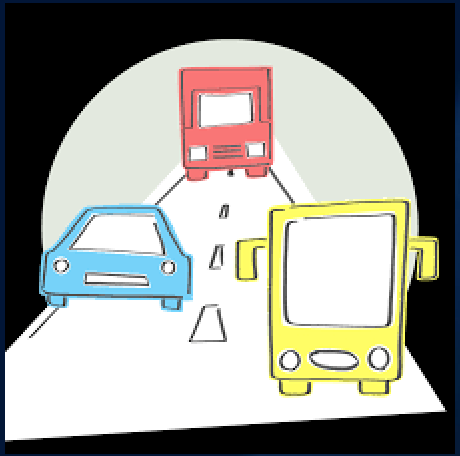
precompute all-pairs-shortest-paths
with Dijkstra's SSSP algorithm

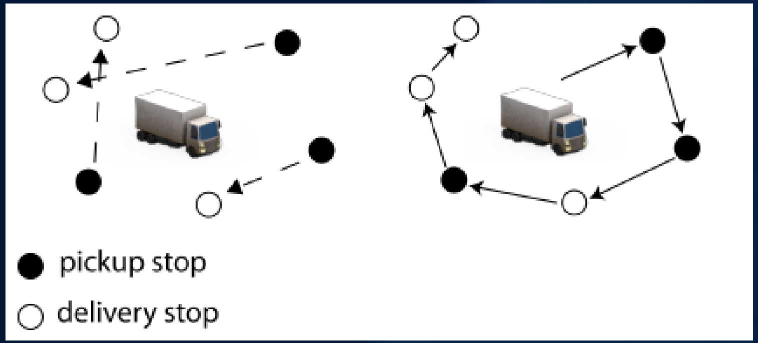


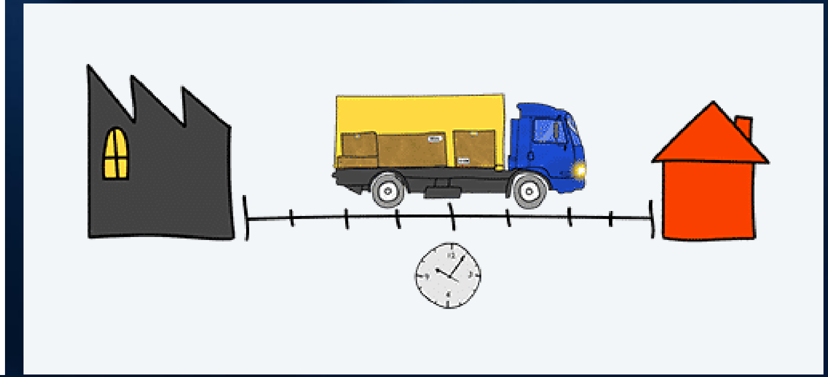



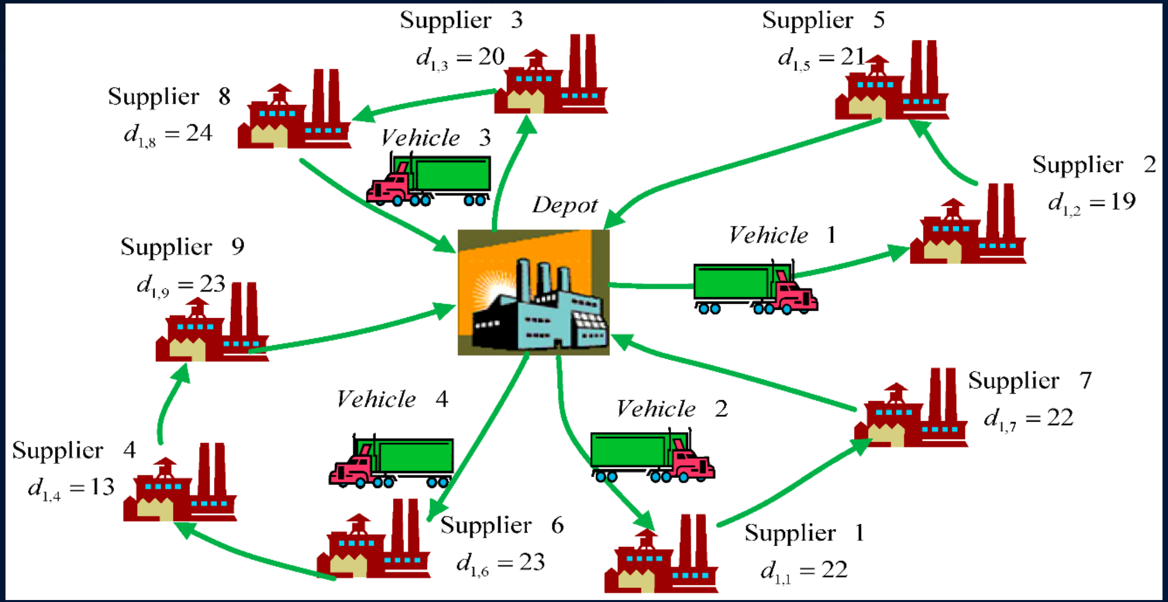
AI CENTER
FEE CTU

# TSP Variants





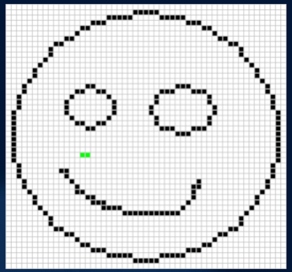





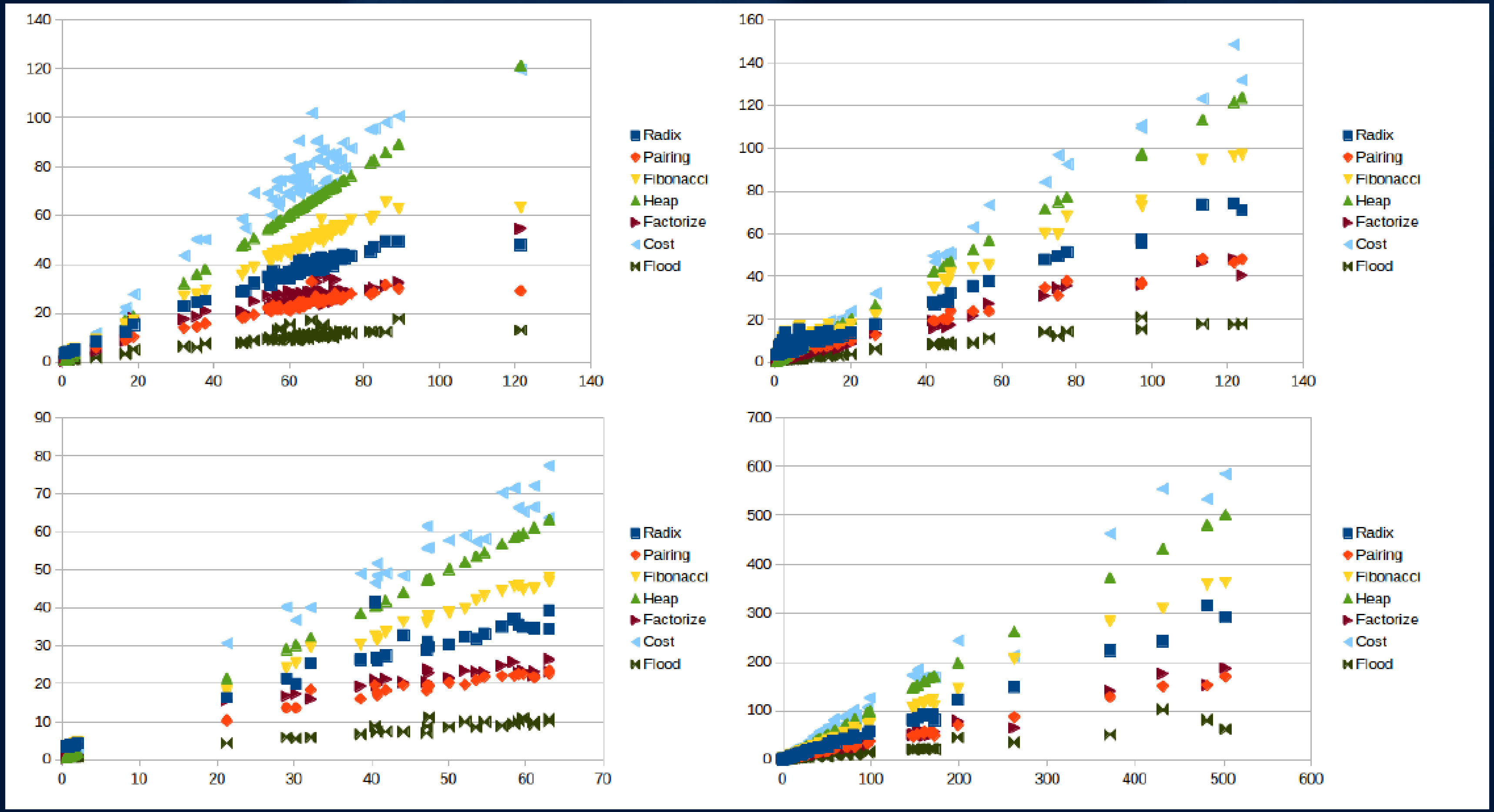

**Time Windows, Capacities, Premium Services, Pickup and Deliveries**

TSP+TW: Restricted time intervals / service times
C+TSP: Limited vehicle load
TSP+PD: Pickup and deliveries (PDP)
TSP*: Premium service – same-day delivery preferred
VRP: Vehicle routing – several vehicles
…

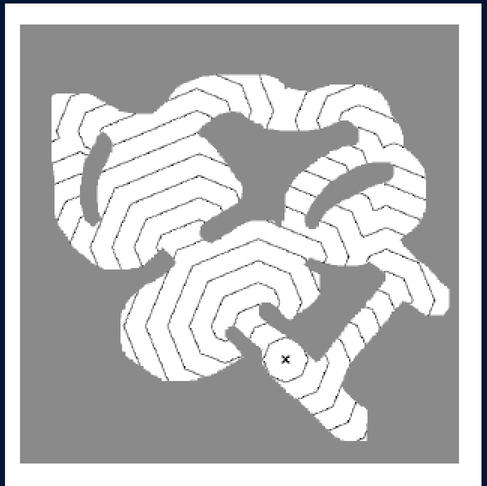
AI CENTER
FEE CTU



# Shortest Paths



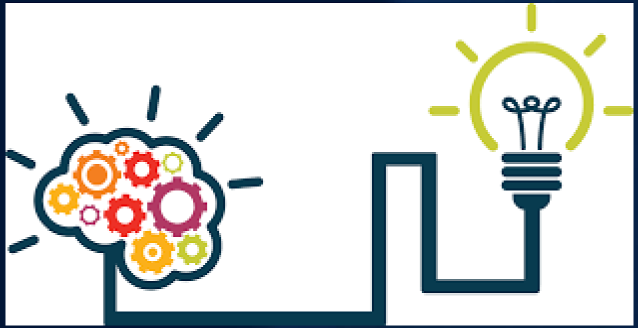

## Cache-efficient Implementation of Dijkstra‘s algorithm



Time in milliseconds for shortest paths search in the game maps of Baldur’s Gate II, Starcraft, Warcraft III,and Dragon Age using scatter plots wrt the performance of shortest paths search with heaps.

AI CENTER
FEE CTU

# Solving TSPs



Given a distance matrix, compute a minimum-cost trip

## Traditional

Model problem as an IP and call solver (CPLEX, IPSolve,. . . )

Neighborhood search (xOPT: SA; GA; AA; PSO; LNS,. . . )

**Depth-First Branch and Bound** with

**DFBnB**0 No Heuristic – incremental O(1) time

**DFBnB**1 Column/Row Minima – incremental O(1) time

**DFBnB**2 Assignment Problem – incremental $O(n^2)$ time

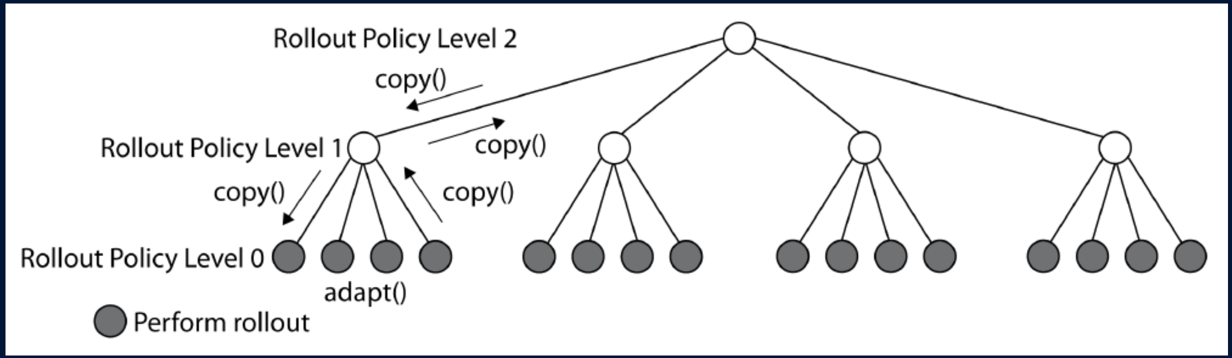
…

## NRPA (Reinforcement Learning)

**Nested-Monte-Carlo Tree Search (with Policy Adaptation)**

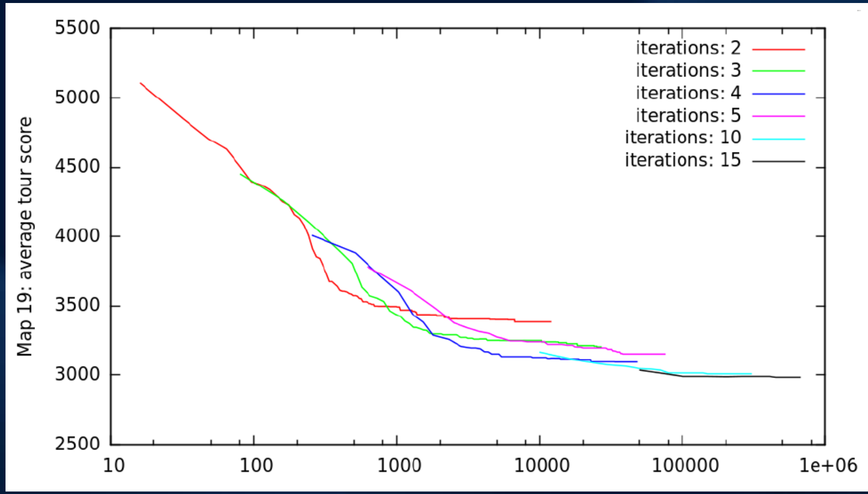
**Input:** Iteration **width** (exploitation), **nestedness** (exploration)

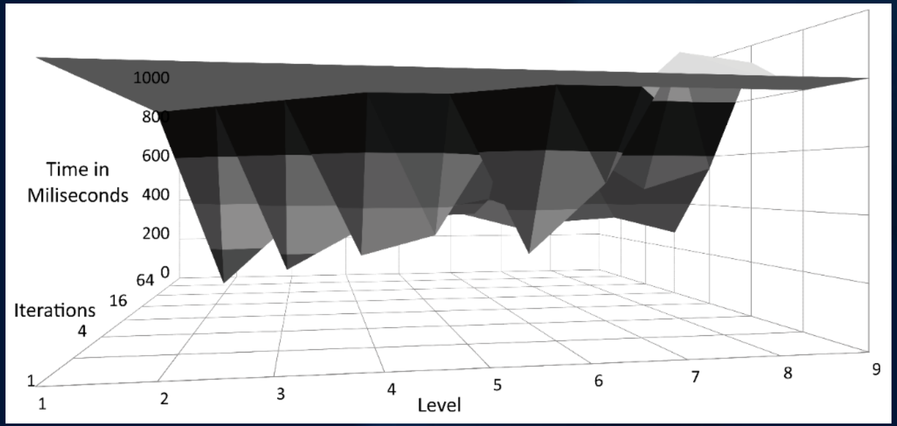
**Policy:** (city-to-city) Mapping IN x IN -> IR to be learnt



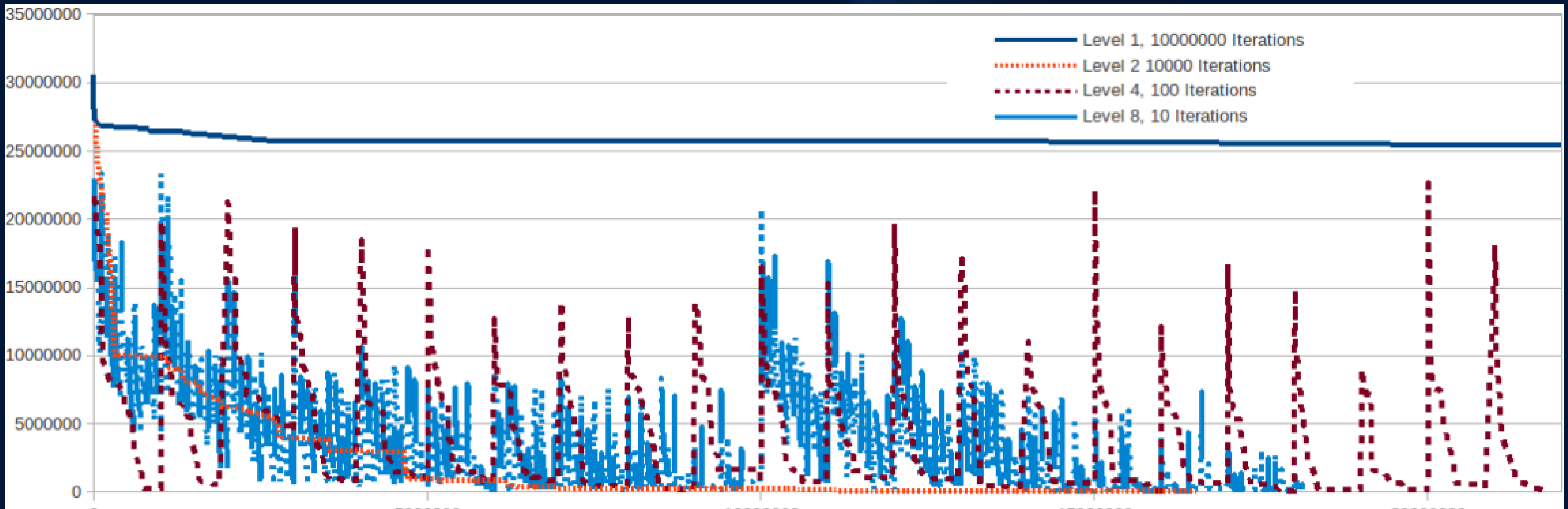
AI CENTER
FEE CTU

# Planning & Optimization with NRPA

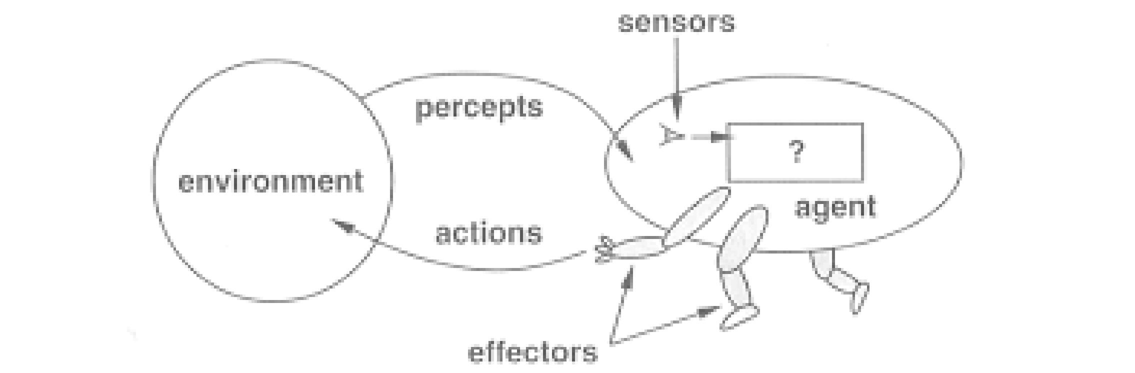


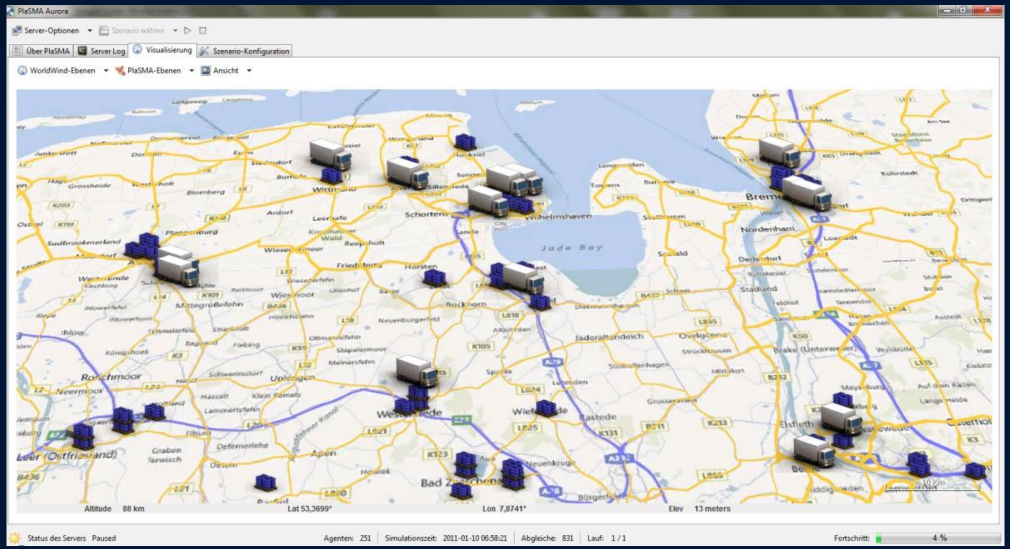




```
Tour search(level)
  best = new Tour();
  best.score = MAXVALUE;
  if (level == 0)
    best.score = rollout();
    best.tour = tour;
  for (int i = 0; i < ITERATIONS; i++)
    Tour r = search(level - 1);
    if (r.score < best.score)
      best.score = r.score;
      best.tour = r.tour;
      adapt(best.tour,level);
  return best;
```



# Integration in Multiagent System

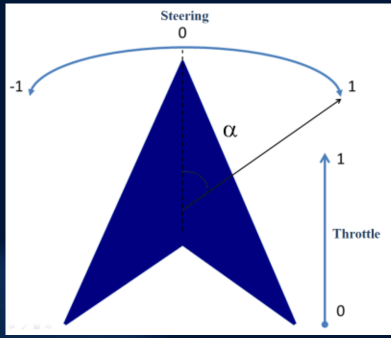



AI CENTER
FEE CTU

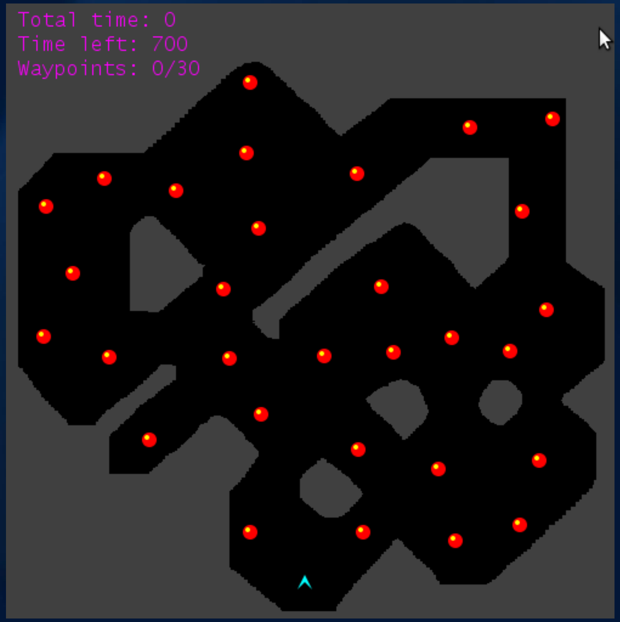
# In Motion

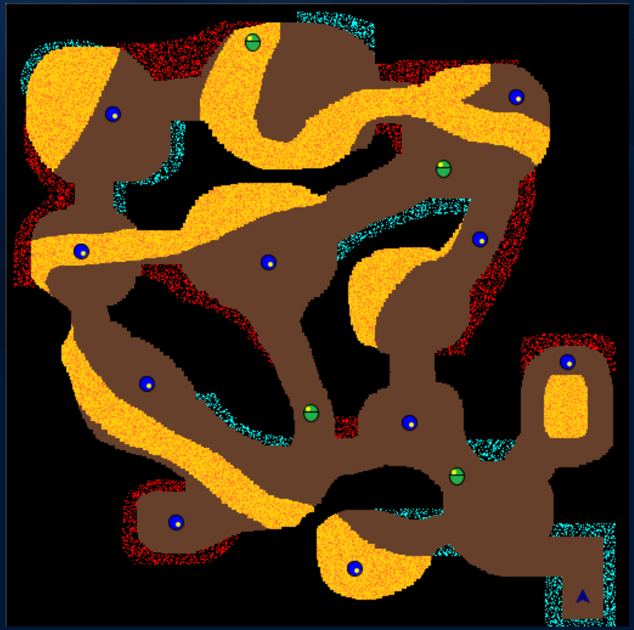
AI CENTER
FEE CTU




| Action ID | Steering | Throttle |
|---|---|---|
| 0 (*Neutral* action) | *Straight* (0) | *Off* (0) |
| 1 | *Left* (-1) | *Off* (0) |
| 2 | *Right* (1) | *Off* (0) |
| 3 | *Straight* (0) | *On* (1) |
| 4 | *Left* (-1) | *On* (1) |
| 5 | *Right* (1) | *On* (1) |

# Physical Traveling Salesman Problem



The main purpose of the PTSP is to provide a benchmark for combined task and motion planning in interactive computer games.

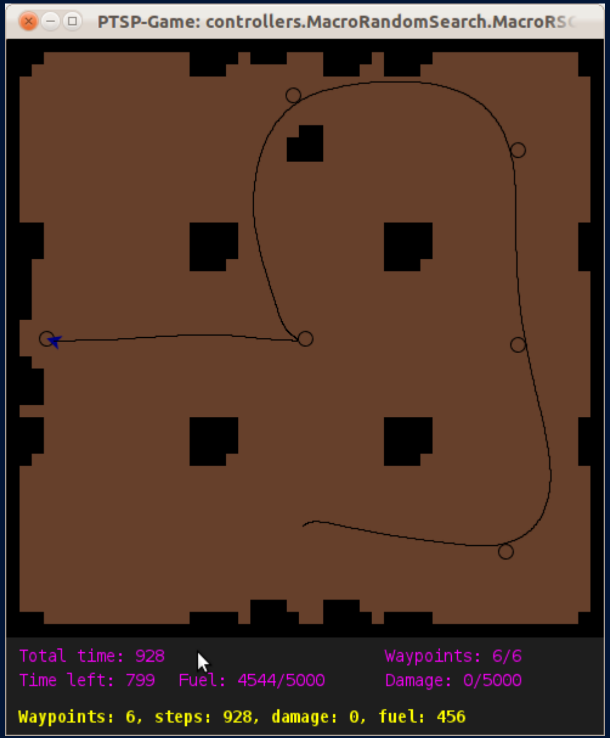
*To reach waypoint locations, the robot often has to avoid numerous obstacles. Planning such motions requires taking into account the robot geometry as well as constraints imposed by its dynamics.*

AI CENTER
FEE CTU

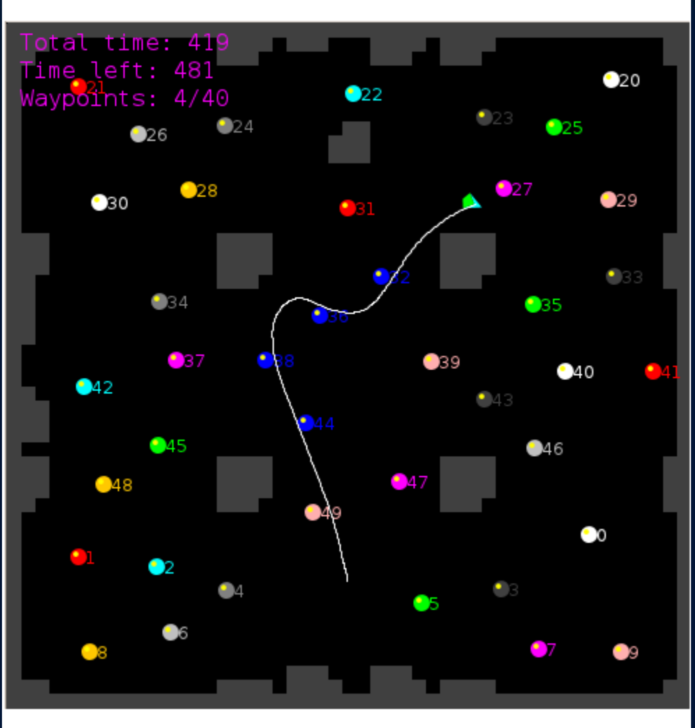
# Introducing Colors

## Clustered TSPs and **Generalized TSPs**

Either only one of each color or all of one color needs to be visited in sequence, minimizing traveling time

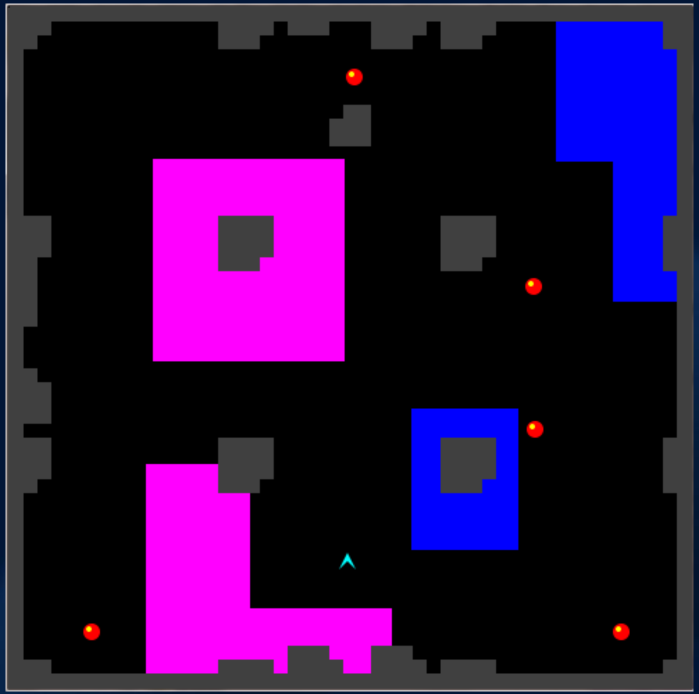


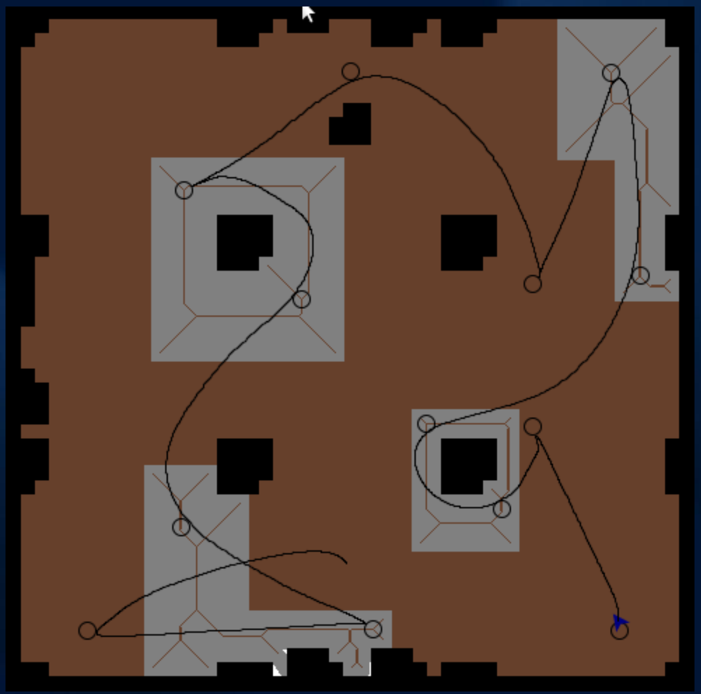


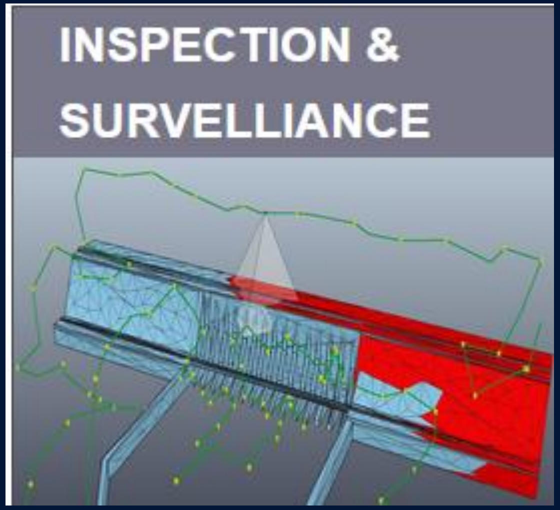


AI CENTER
FEE CTU

# Inspection





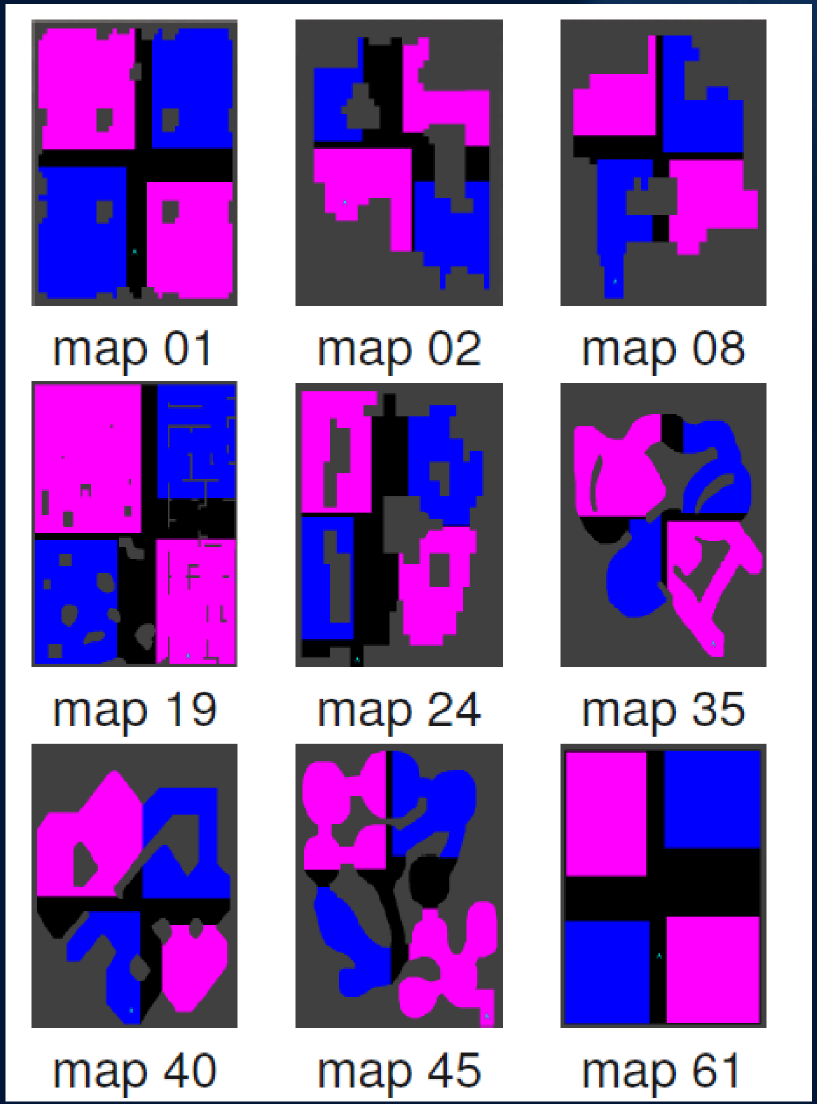


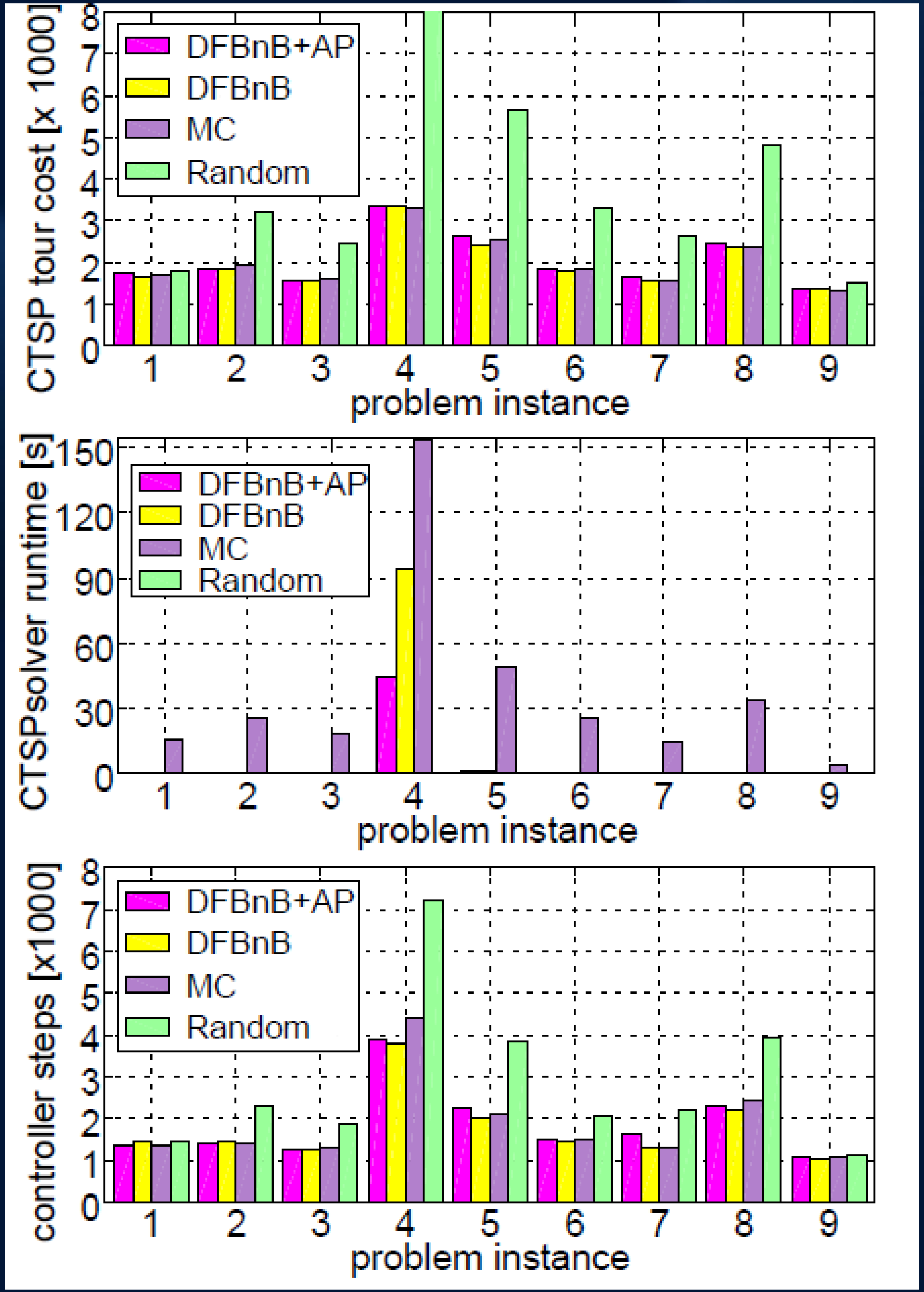

As defects to objects such as pipeline leakage can result in tremendous economical loss, the inspection problem is one of the most important problems in robotics.

Inspection poses significant challenges since the robot needs to determine and reach a set of locations whose combined visibility covers the inspection area.

AI CENTER
FEE CTU

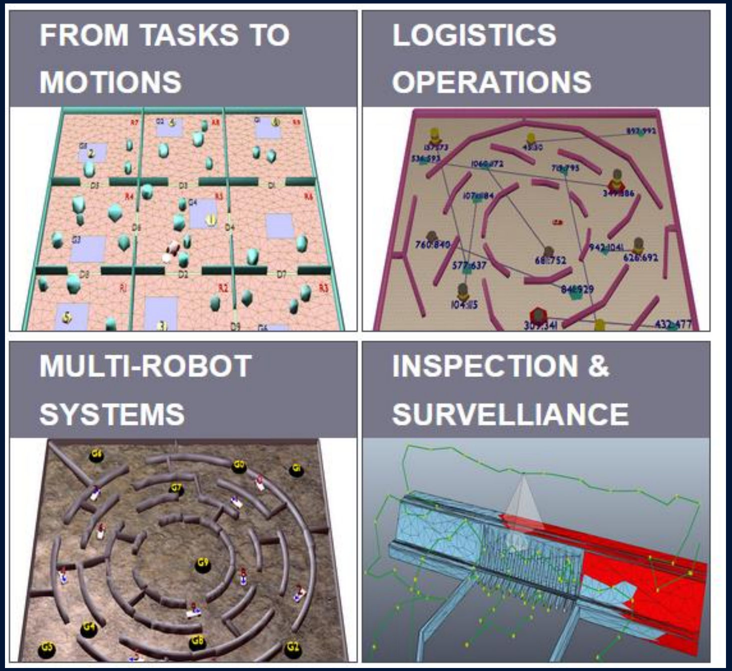
# Results

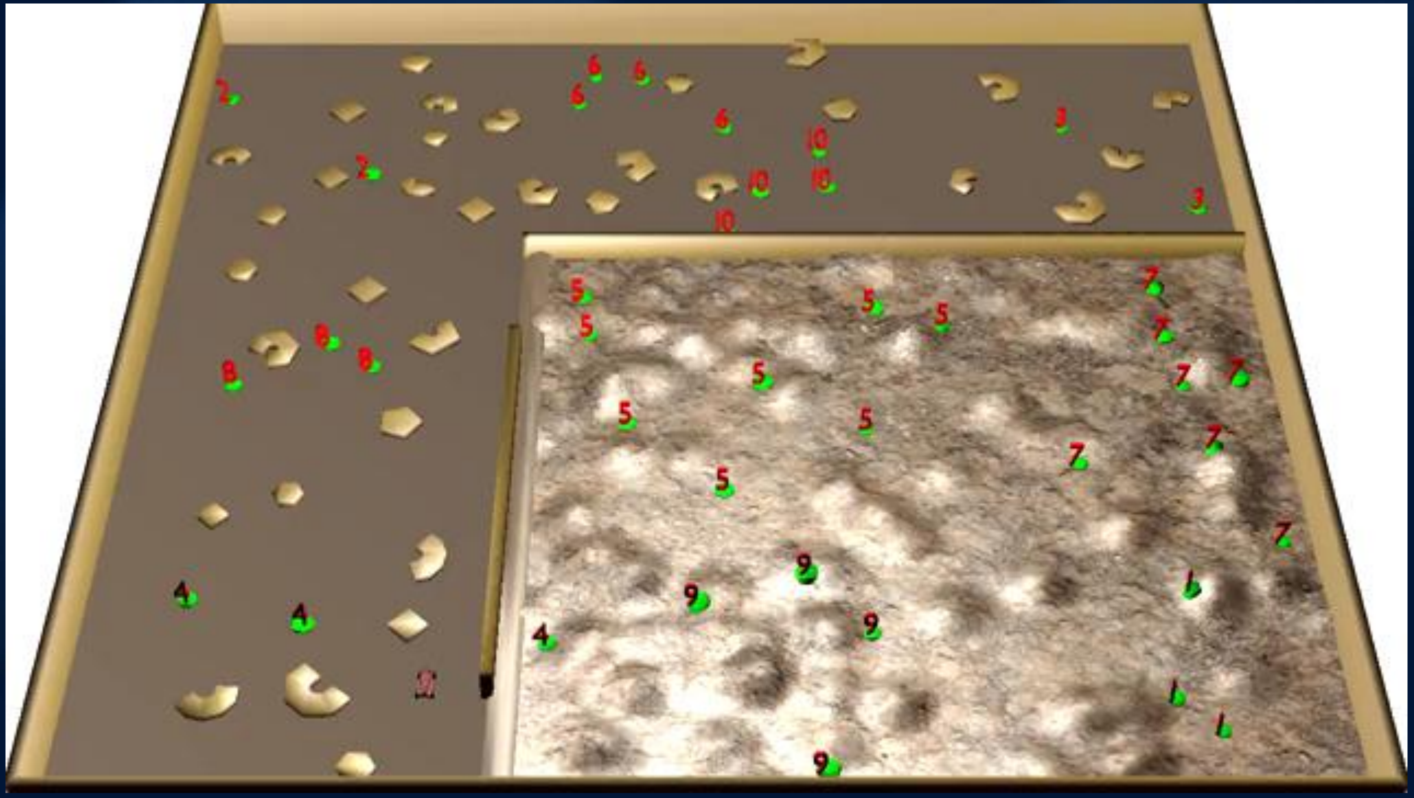


AI CENTER
FEE CTU

# Integration into the „Framework“



AI CENTER
FEE CTU

# Approach

**Sampling-based motion planning**

* generality: dynamics as black-box function snew ← MOTION(s,u,dt)
* continuous state/control spaces: probabilistic sampling to make it feasible
* high-dimensionality: search to find solution

**coupled with discrete abstractions**

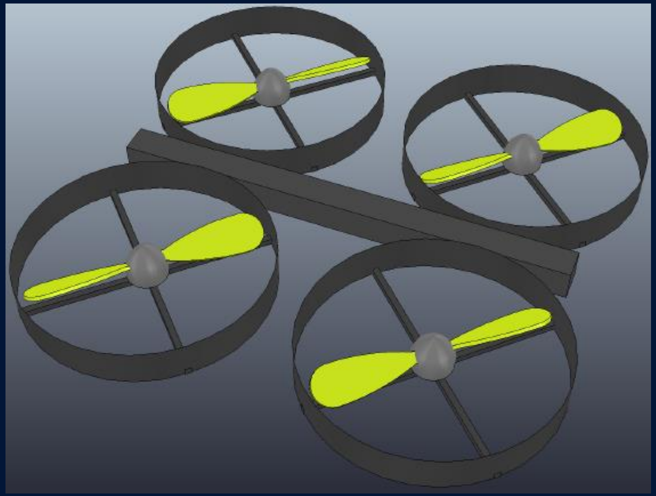
* provide simplified planning layer
* guide search in the continuous state/control spaces

**and motion controllers**

* open up the black-box MOTION function
* facilitate search expansion

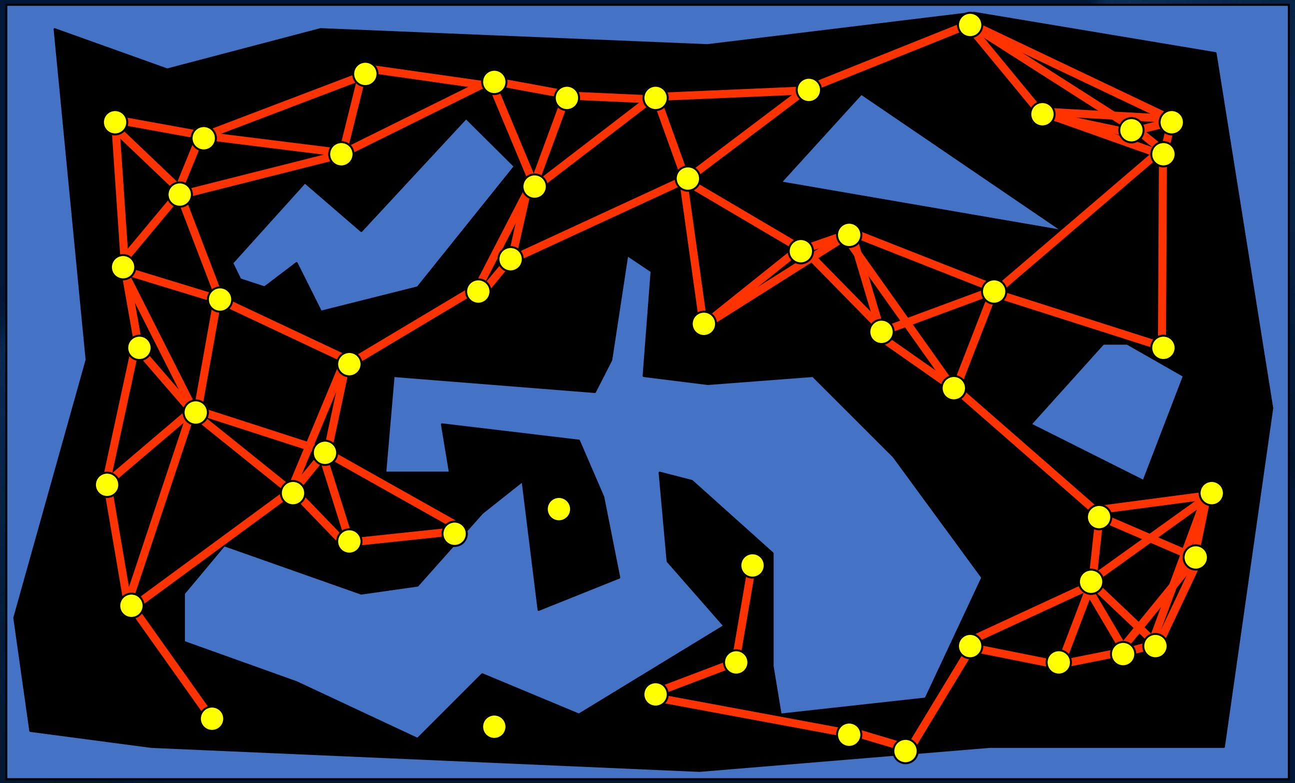
**Formal guarantees**

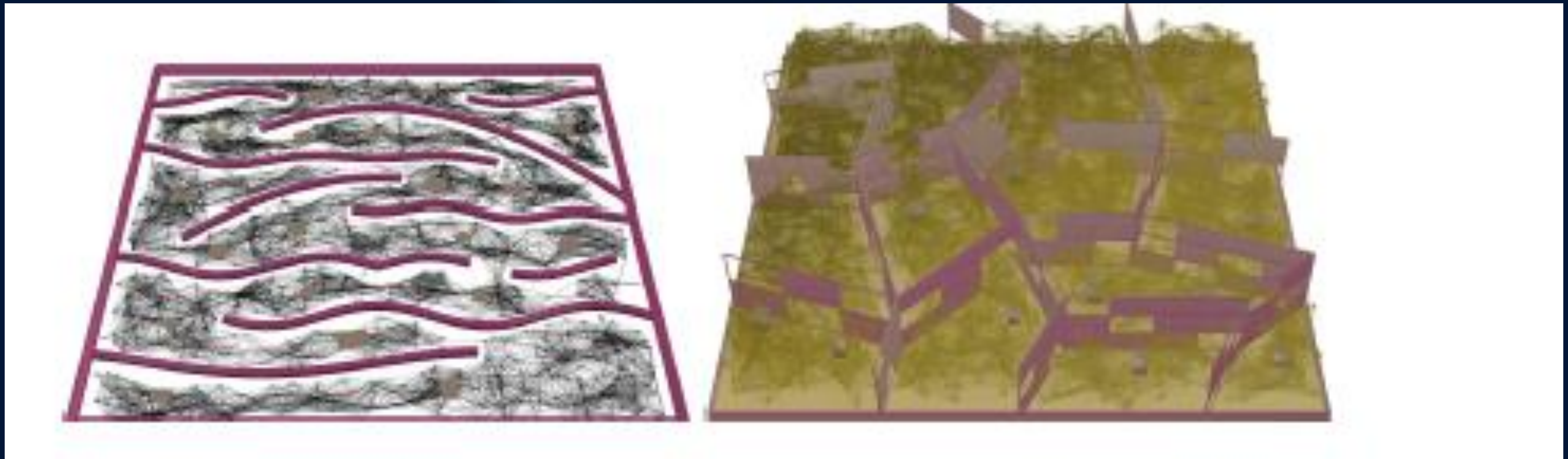
* probabilistic completeness



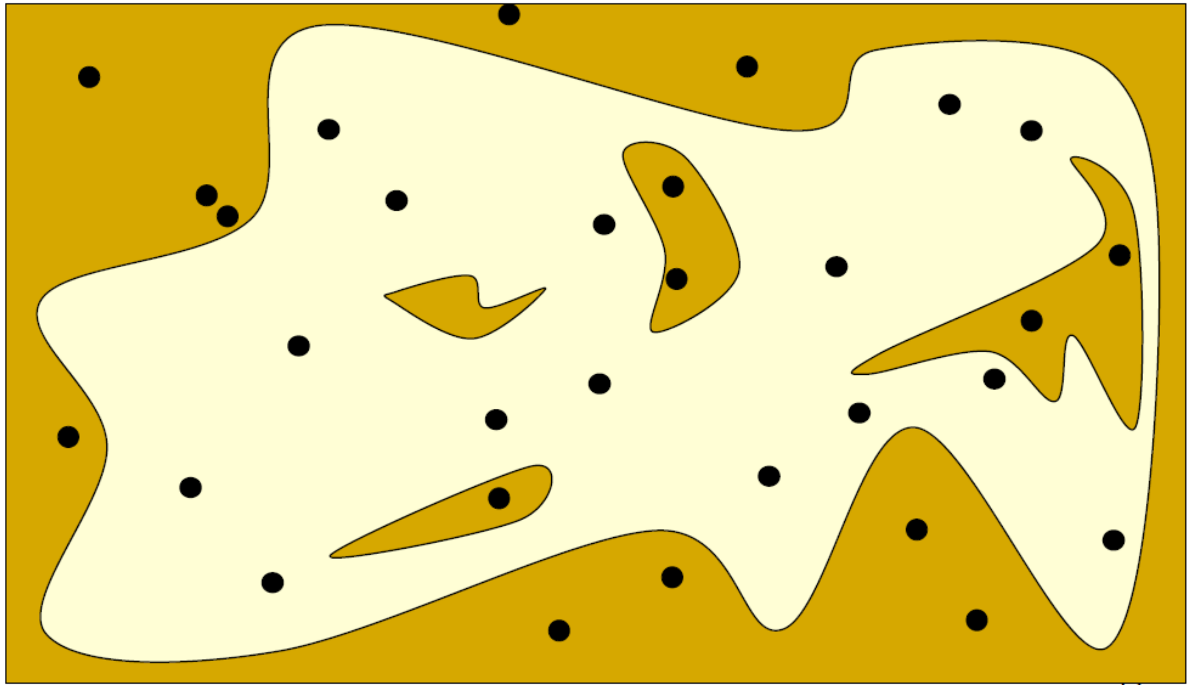
AI CENTER
FEE CTU

# Probabilistic Roadmap

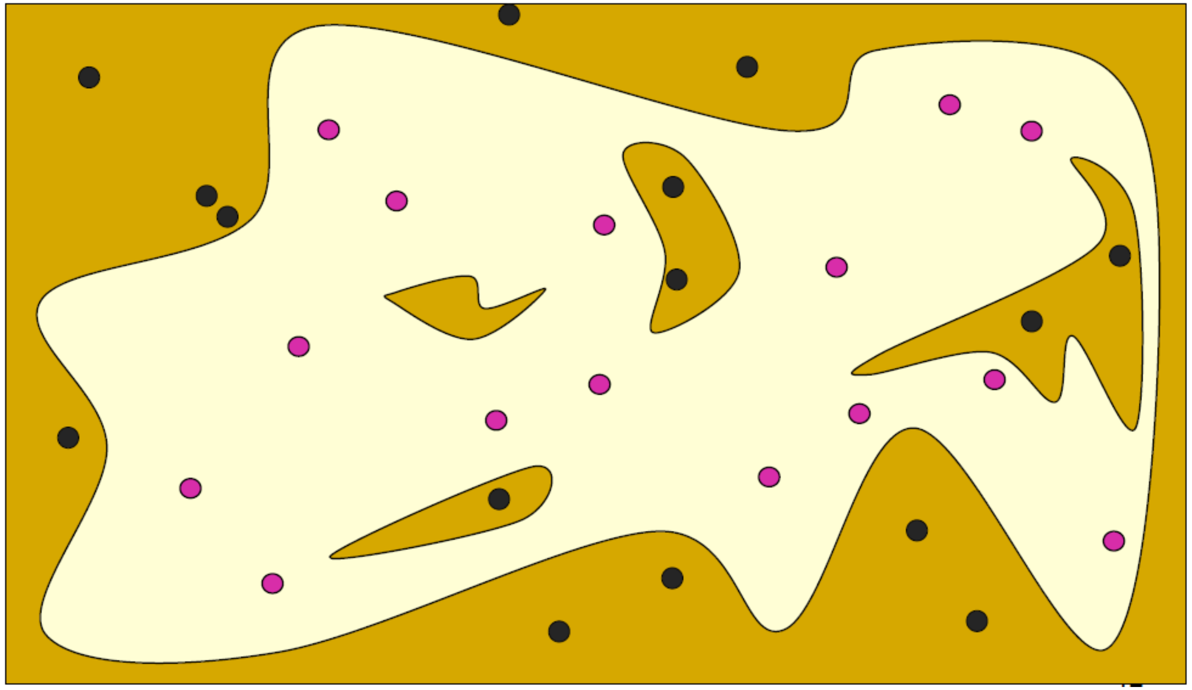




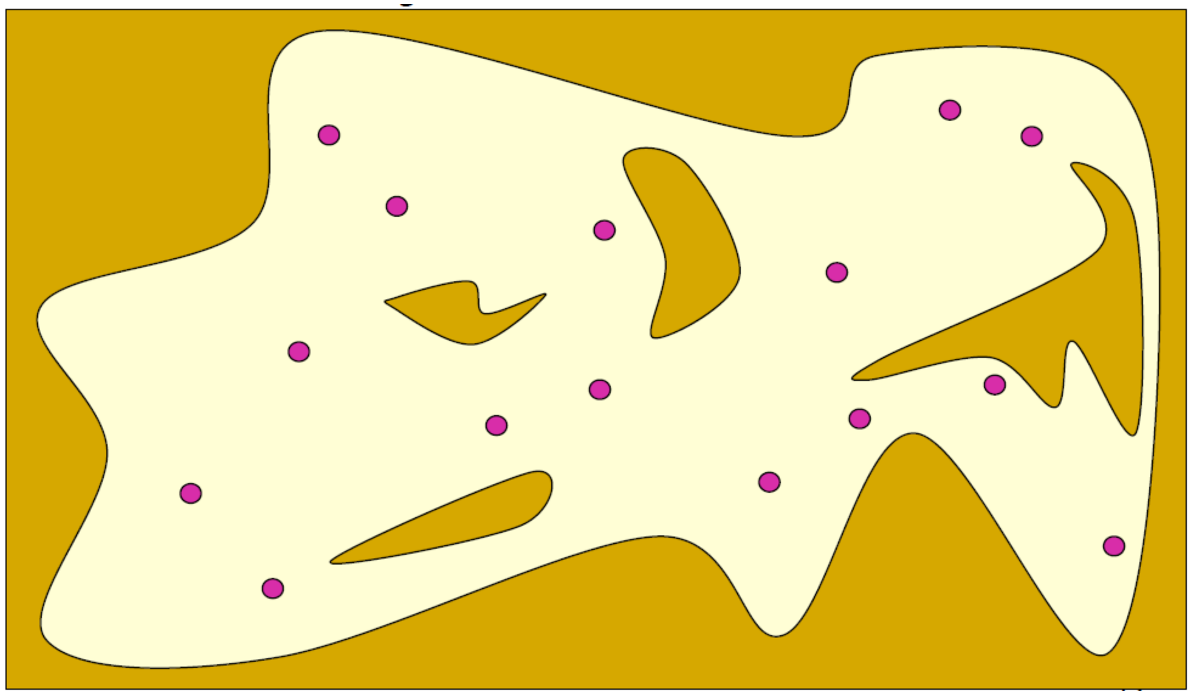
# PRM: (Uniform) Sampling



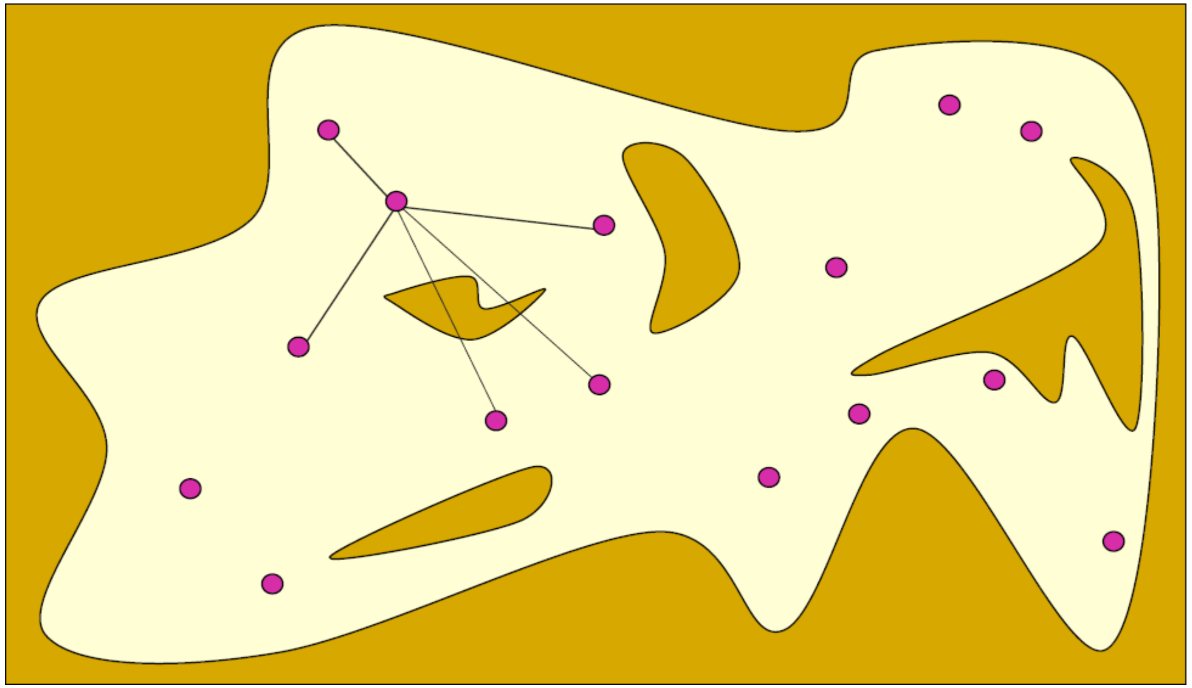
# PRM: Valid Nodes



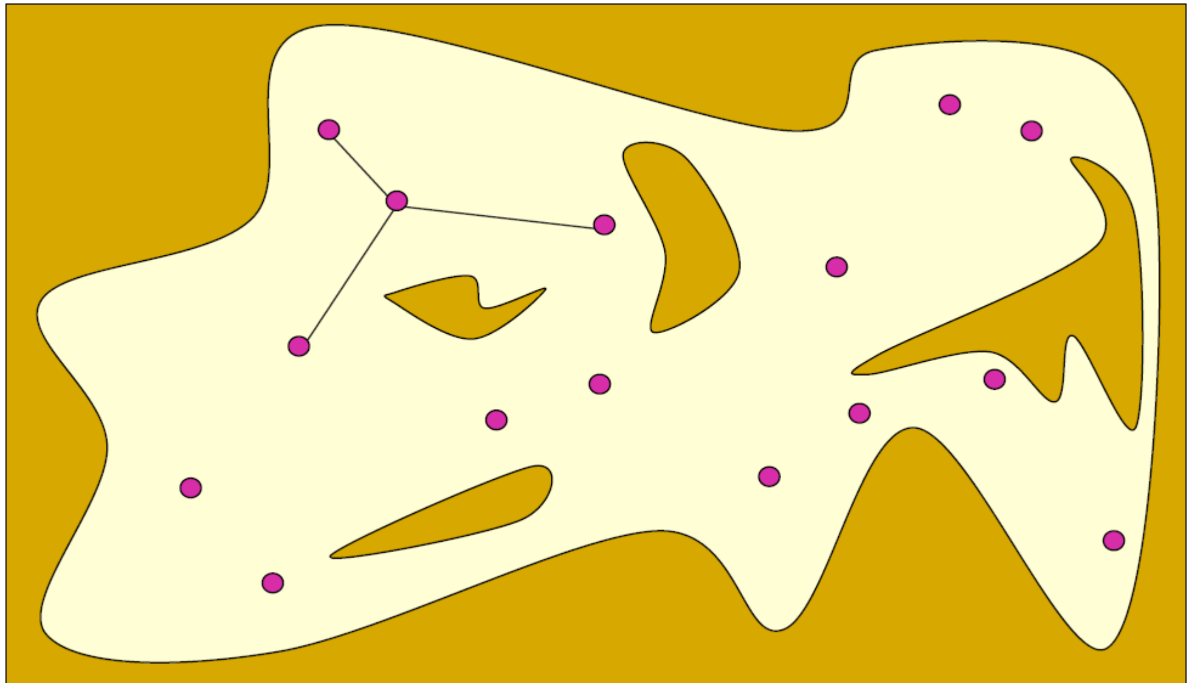
# PRM: Filtering Edges



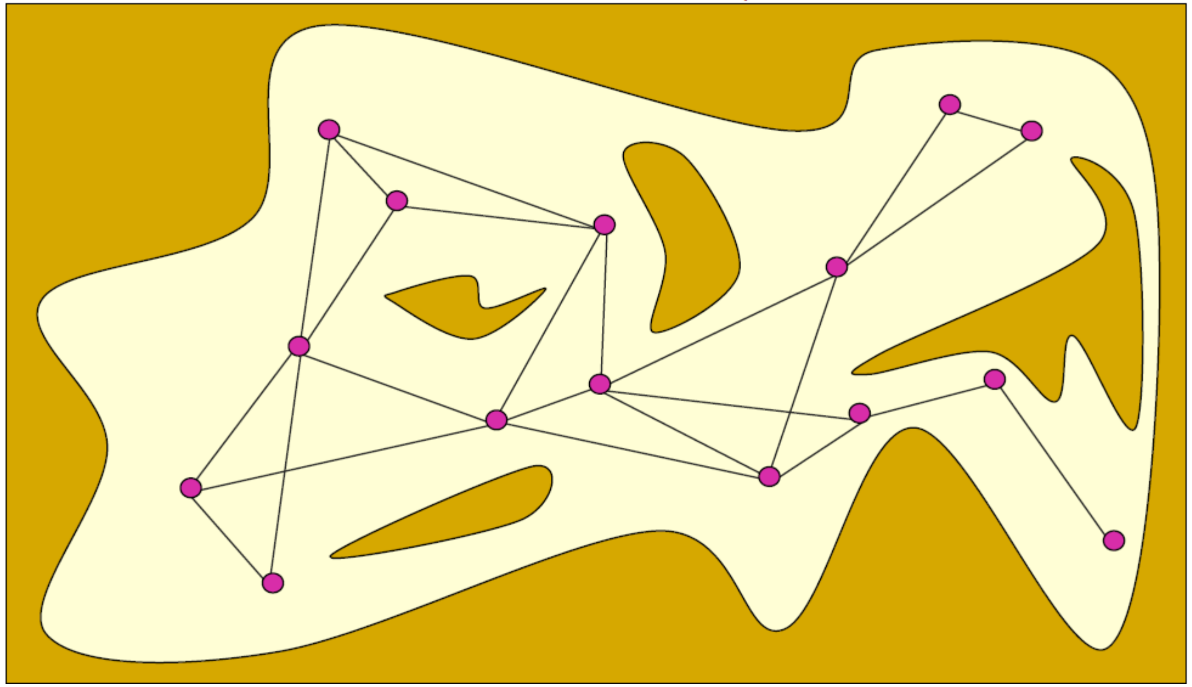
# PRM: Connect Nearest Neighbors



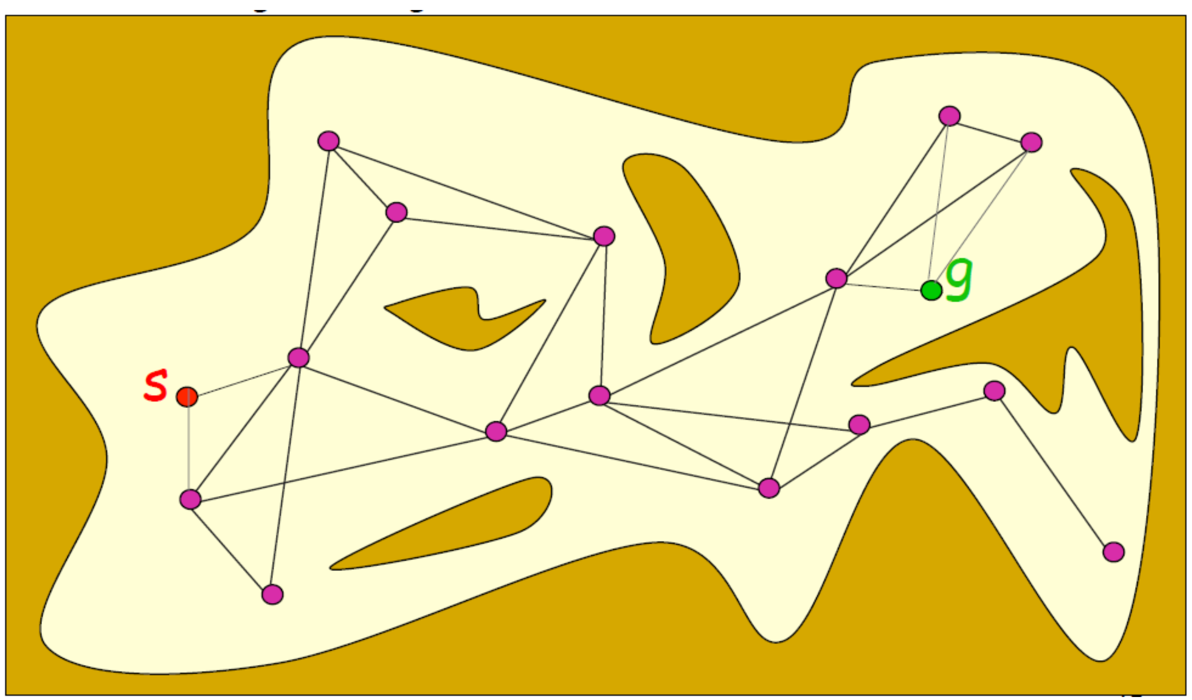
# PRM: Filtering Edges



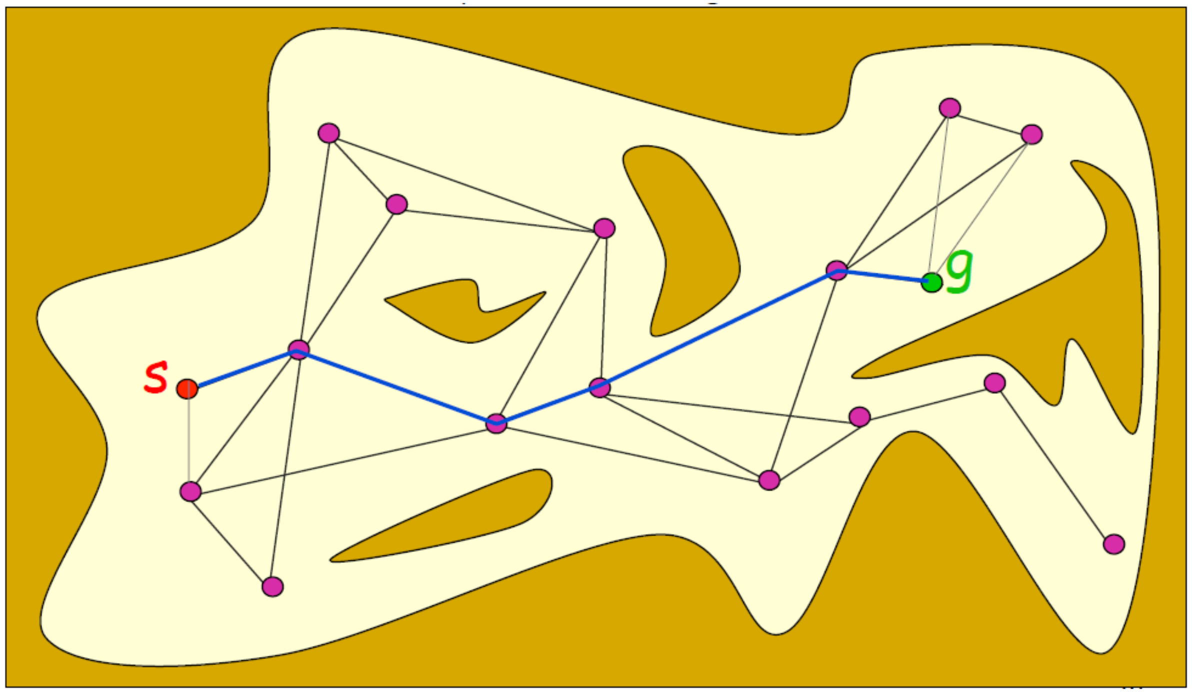
# PRM: Entire Graph



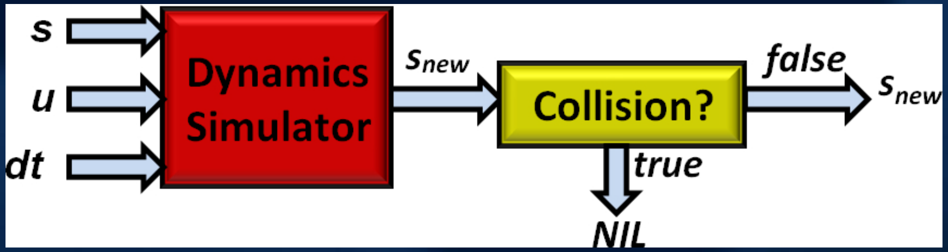
# PRM: Integrate Start and Goal



# PRM: Path Search



AI CENTER
FEE CTU




# Dynamics

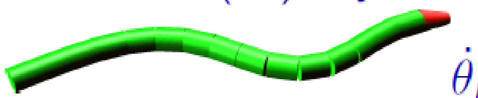
**Express relation between input controls and resulting motions**

**Modeled via physics-based engines**

$$\dot{s} = f(s, u)$$

$$s = (x, y, \theta_0, v, \psi, \theta_1, \ldots, \theta_n) \qquad u = (a, \omega)$$

$$\dot{x} = v\cos(\theta_0) \quad \dot{y} = v\sin(\theta_0) \quad \dot{\theta}_0 = v\tan(\psi) \quad \dot{v} = a \quad \dot{\psi} = \omega$$

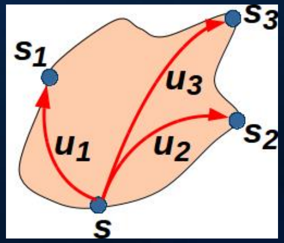


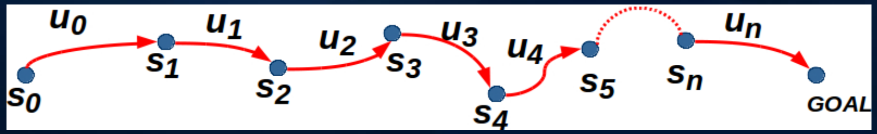
$$\dot{\theta}_i = \frac{v}{d}\left(\prod_{j=1}^{i-1}\cos(\theta_{j-1} - \theta_j)\right)(\sin(\theta_{i-1}) - \sin(\theta))$$

AI CENTER
FEE CTU

# Dynamics

**Necessary to plan motions that can be executed**
**Impose significant challenges**




- ➢ Constrain the feasible motions
- ➢ Often are nonlinear and high-dimensional
- ➢ Give rise to nonholonomic systems
- ➢ State and control spaces are continuous
- ➢ Solution trajectories are often long

**Computational complexity of motion planning with dynamics**

Point with Newtonian dynamics NP-Hard [DXCR 1993]

Polygon Dubin's car Decidable [CPK 2008]

General nonlinear dynamics Undecidable [Branicky 1995]

AI CENTER
FEE CTU

# Introduce Discrete Layer to Guide the Search

Workspace decomposition provides discrete layer as adjacency graph $G = (R,E)$

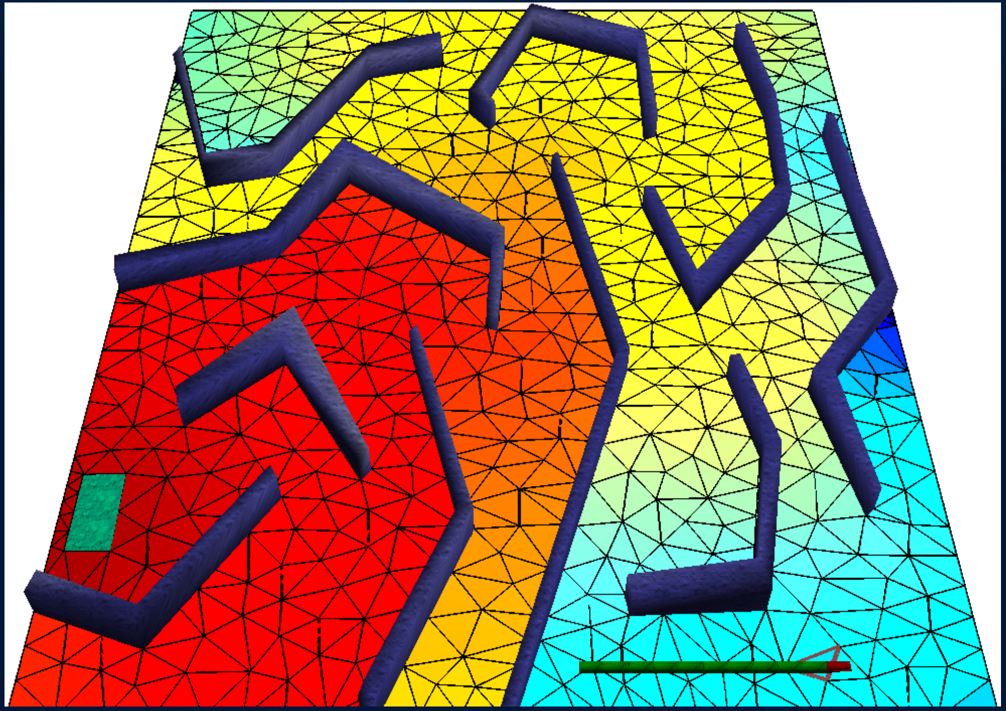
R denotes the regions of the decomposition

$E = \{(r_i,r_j) \mid r_i, r_j \text{ in } R \text{ are physically adjacent}\}$

hcost(r) estimates the difficulty of reaching the goal region from r defined as length of shortest path in $G = (R,E)$ from r to goal

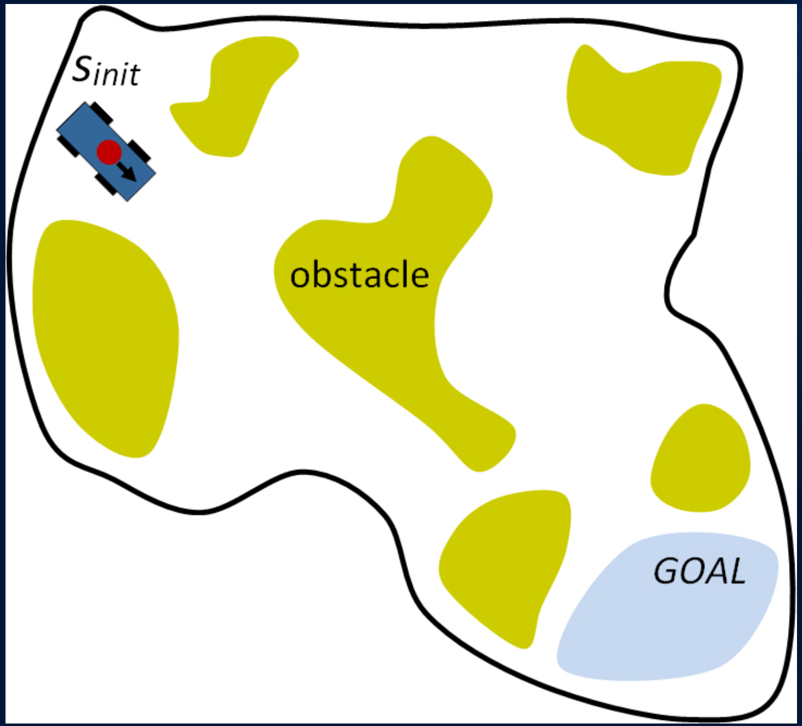
[hcost(r1), hcost(r2),..., hcost(rn)]

computed via BFS/A* on G backwards from goal



AI CENTER
FEE CTU

# Sampling Based Motion Planning

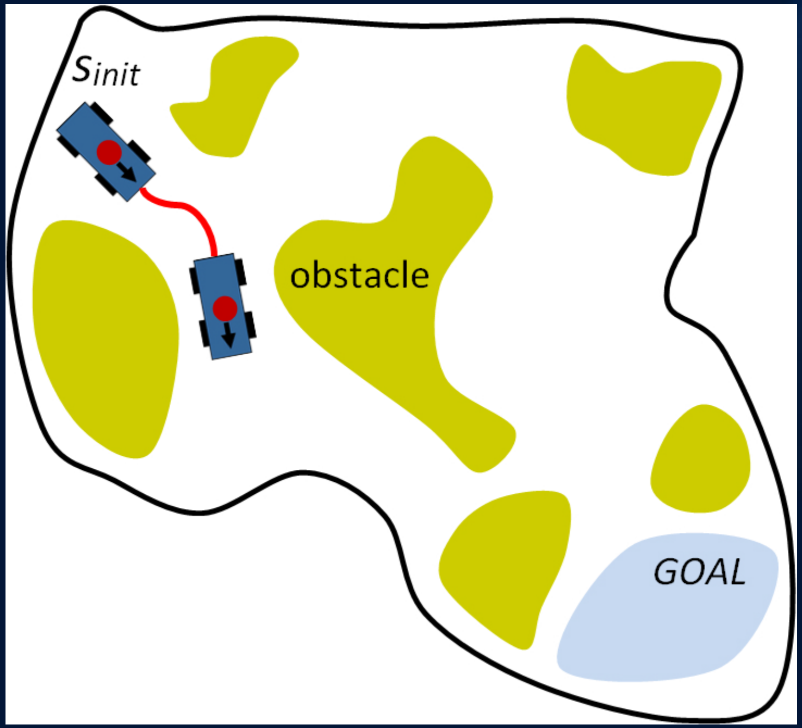



Expand a tree T of collision-free and dynamically-feasible motions

- select a state s from which to expand the tree
- sample control input u
- generate new trajectory by
- applying u to s

AI CENTER
FEE CTU

# Sampling Based Motion Planning



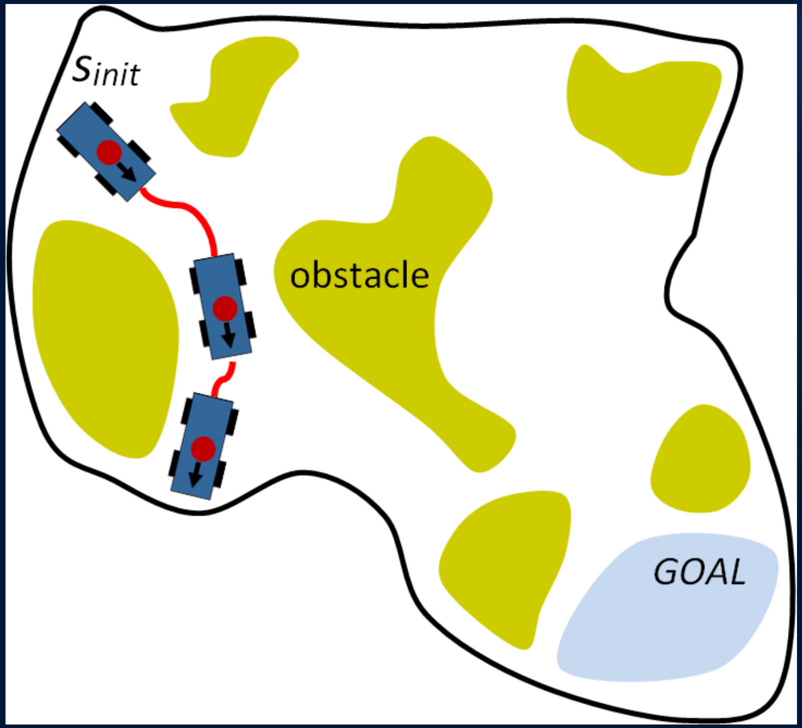

Expand a tree T of collision-free and dynamically-feasible motions

- select a state s from which to expand the tree
- sample control input u
- generate new trajectory by
- applying u to s

delkamp

AI CENTER
FEE CTU

# Sampling Based Motion Planning

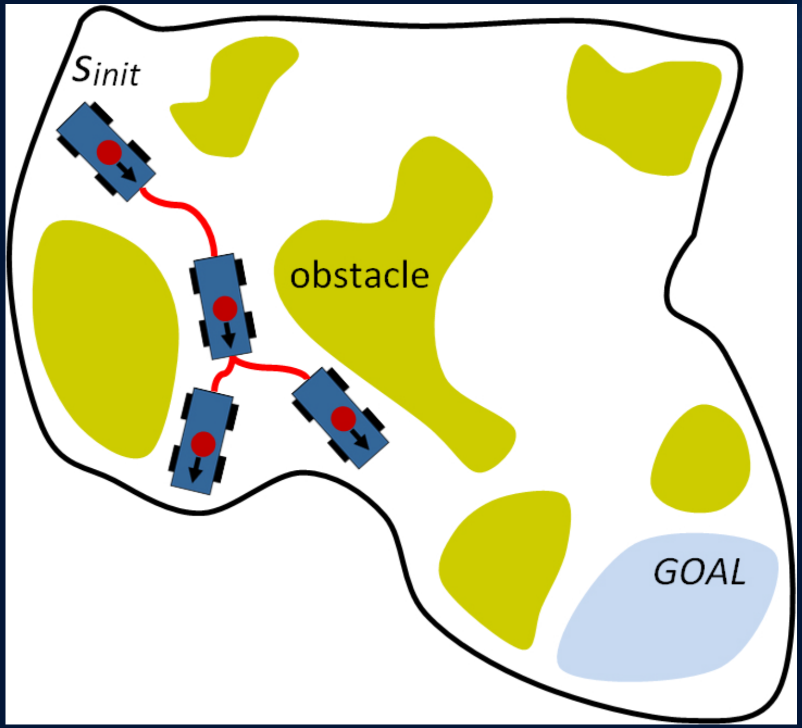



Expand a tree T of collision-free and dynamically-feasible motions

- select a state s from which to expand the tree
- sample control input u
- generate new trajectory by
- applying u to s

AI CENTER
FEE CTU

# Sampling Based Motion Planning



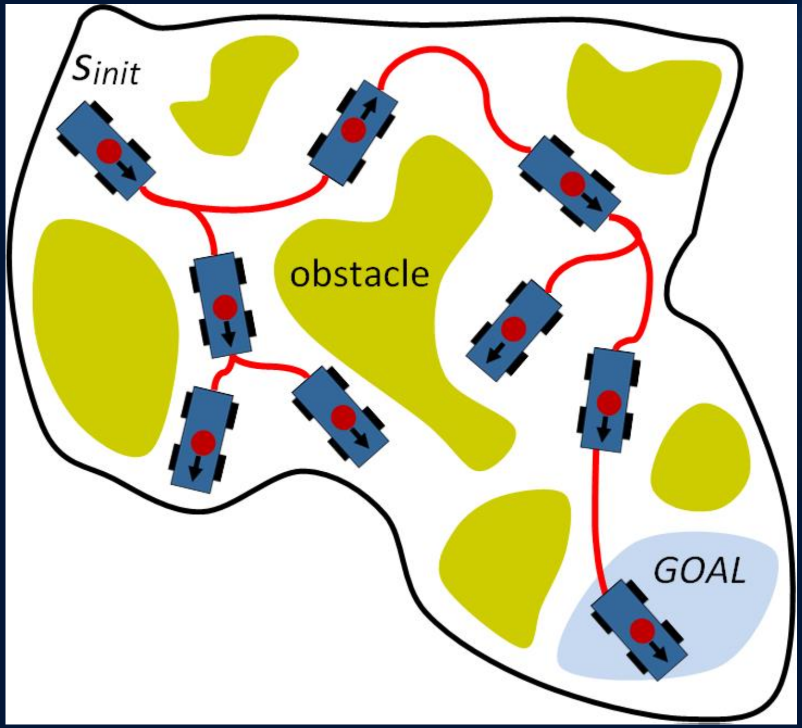

Expand a tree T of collision-free and dynamically-feasible motions

- select a state s from which to expand the tree
- sample control input u
- generate new trajectory by
- applying u to s

7elkamp

AI CENTER
FEE CTU

# Sampling Based Motion Planning




Expand a tree T of collision-free and dynamically-feasible motions

- select a state s from which to expand the tree
- sample control input u
- generate new trajectory by
- applying u to s

AI CENTER
FEE CTU

# Guided Expansion of Motion Tree

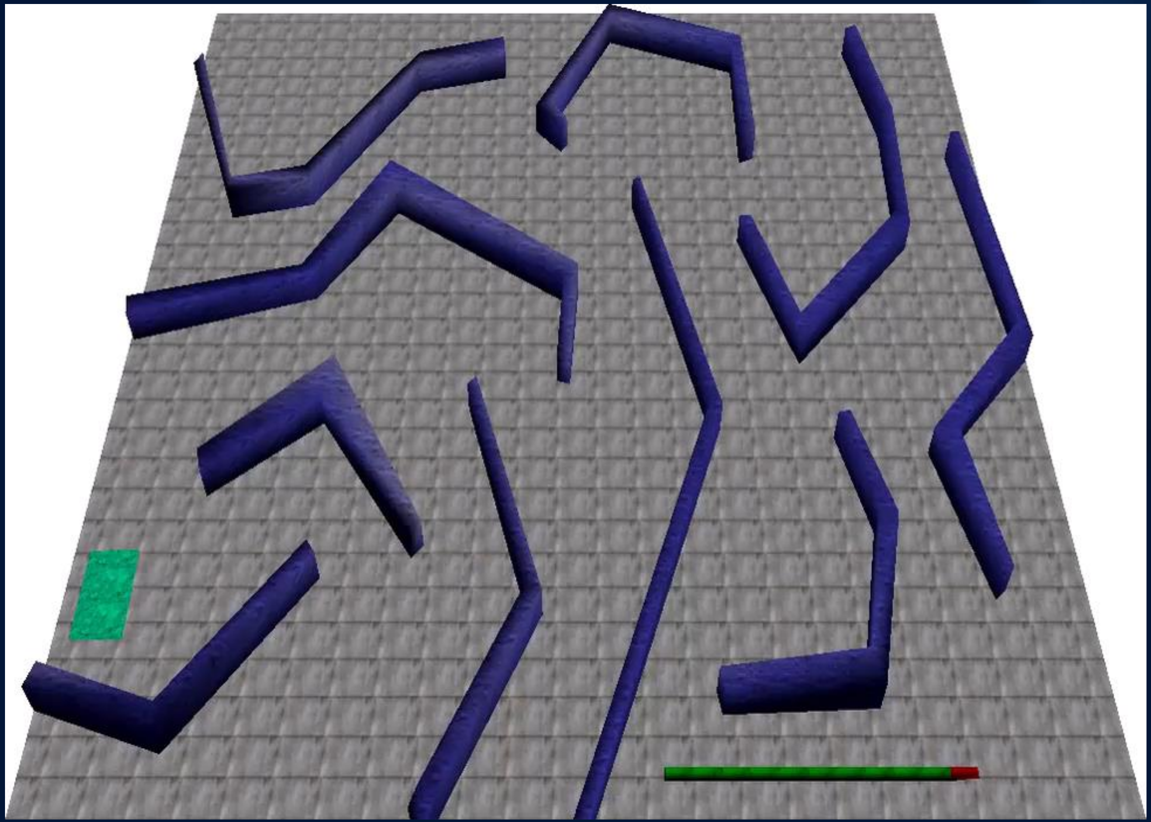
selecting an equivalence class
from which to expand motion tree T

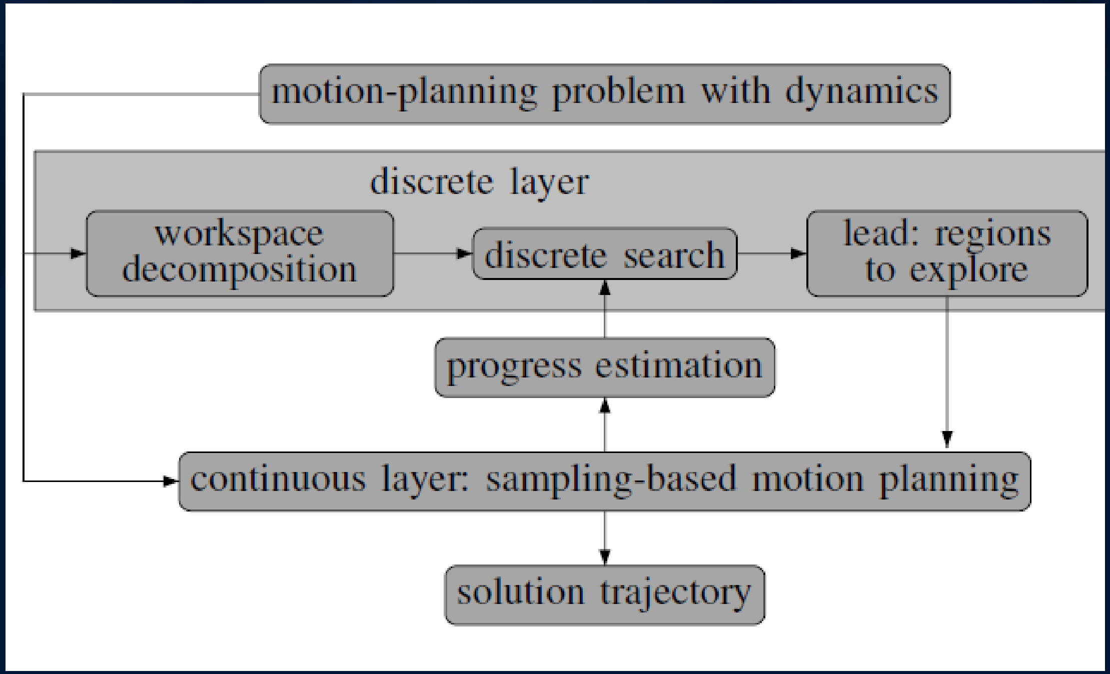
sampling-based motion planning to expand T

delkamp

AI CENTER
FEE CTU

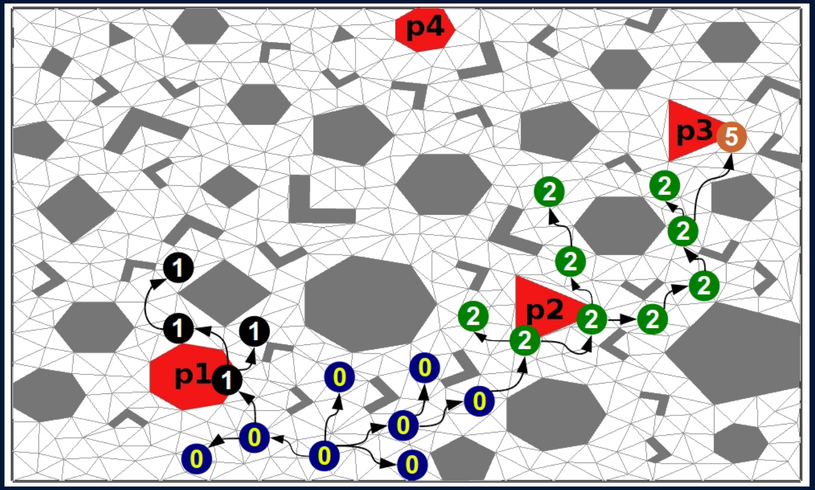
# Architecture





AI CENTER
FEE CTU

# Abstraction




**Used to induce partition of motion tree into equivalence classes**

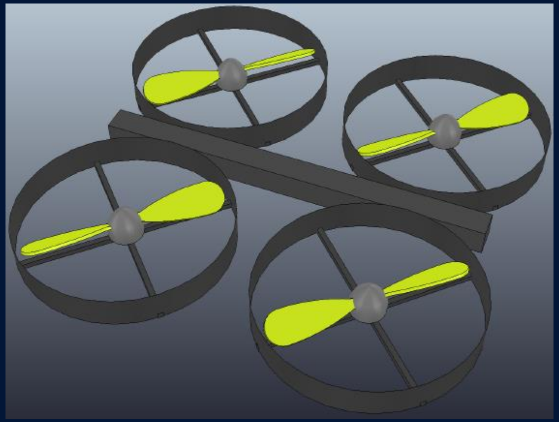
vi = vj
iff
TRAJ(T,vi) provides same abstract information as TRAJ (T,vj)
iff
region(vi) = region(vj)

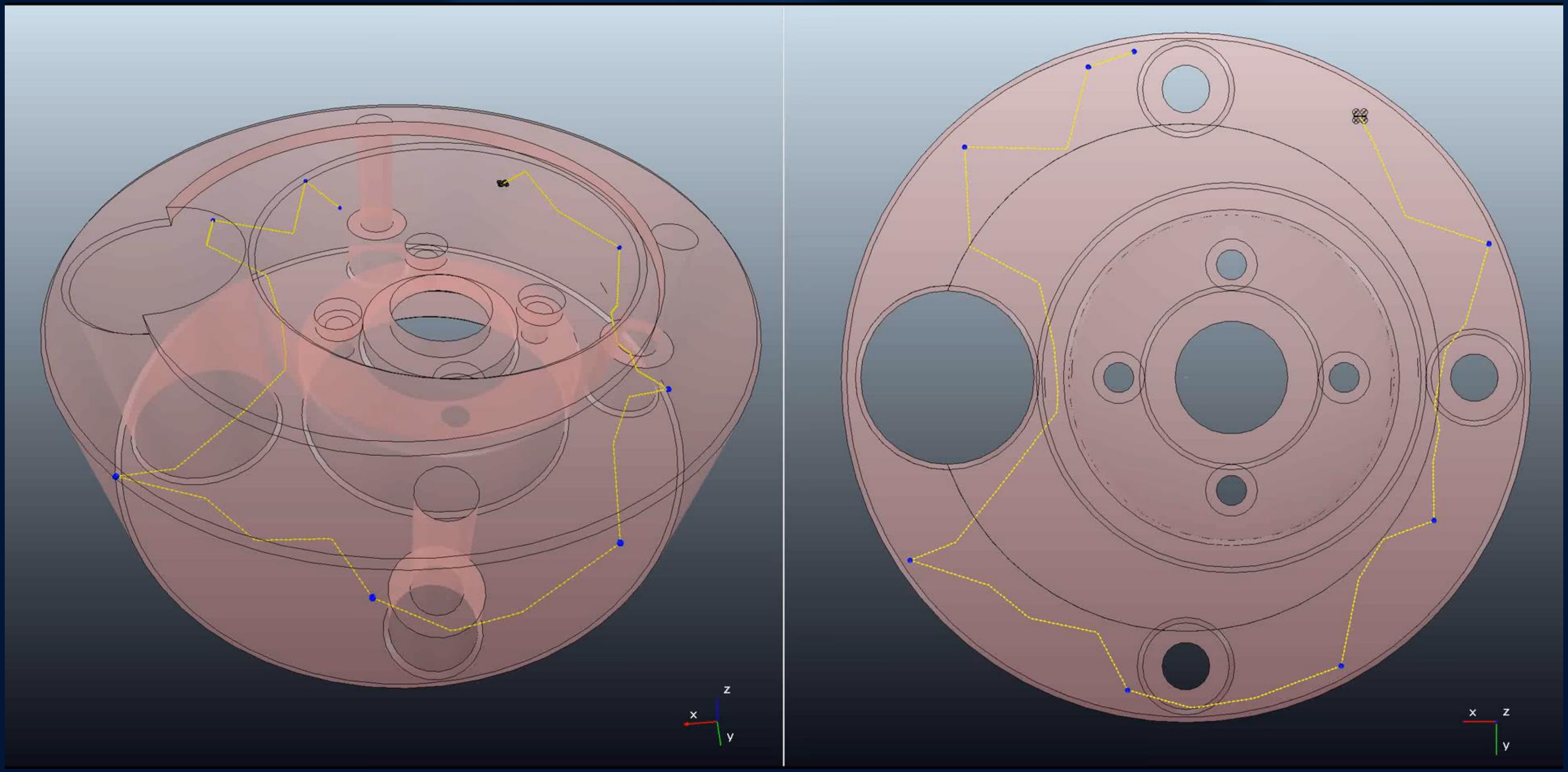
➔ equivalence class corresponding to abstract state <r>
Γ<r> = {v | v in T and region(v) = r}
➔ partition of motion tree T into equivalence classes
Γ = {Γ <r> : Γ <r> > 0}

delkamp

AI CENTER
FEE CTU

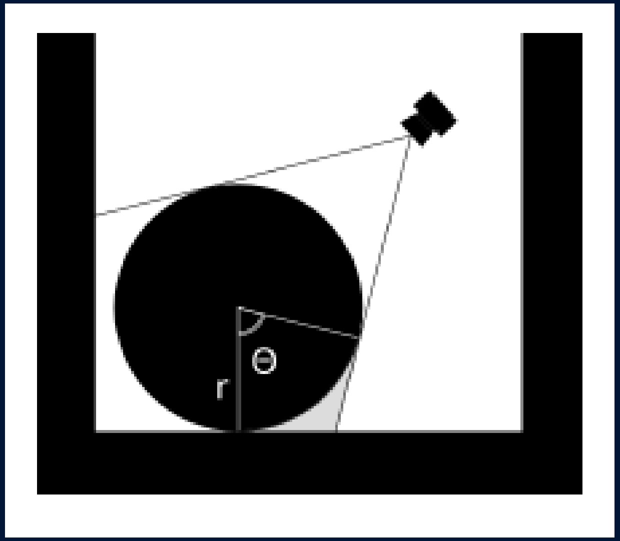
# Inspection

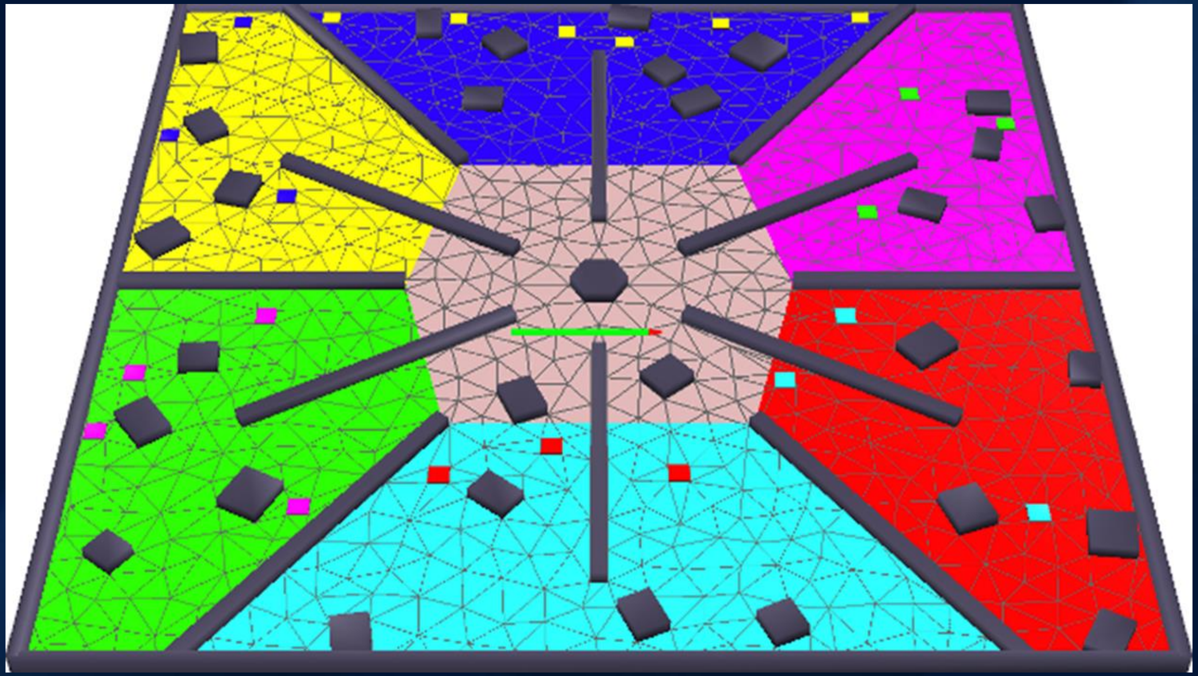


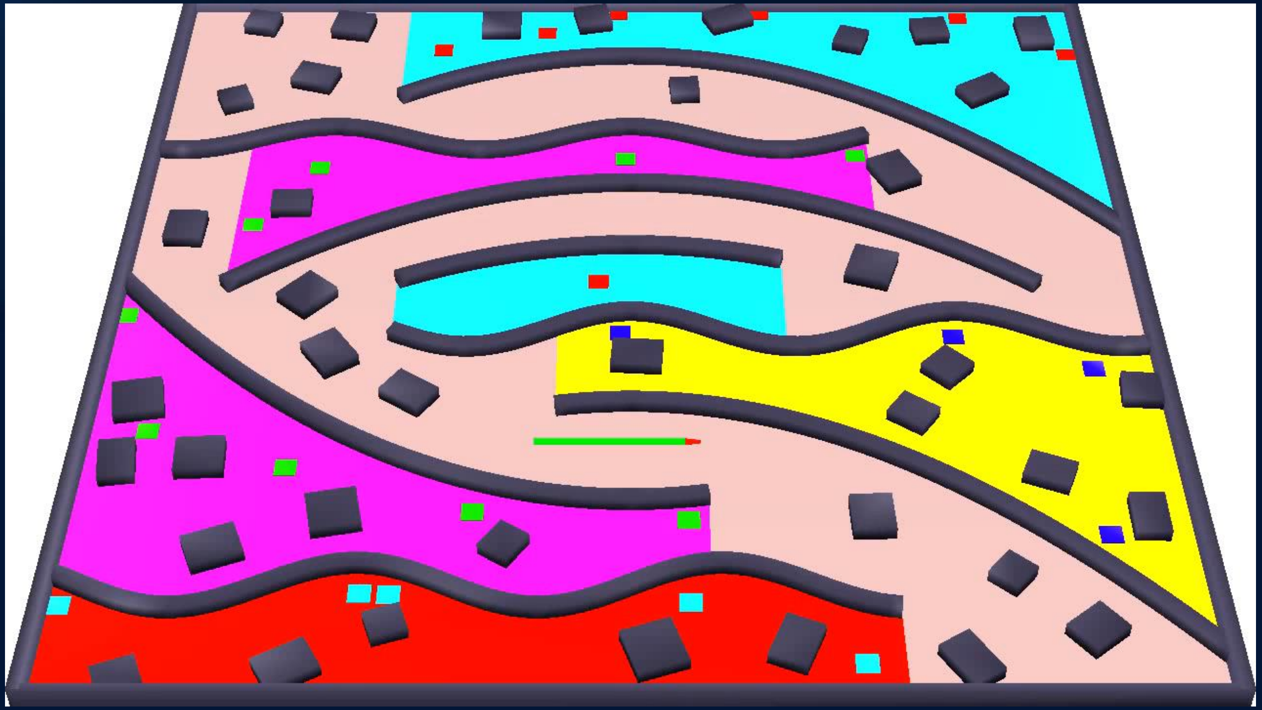



AI CENTER
FEE CTU

# In 2D

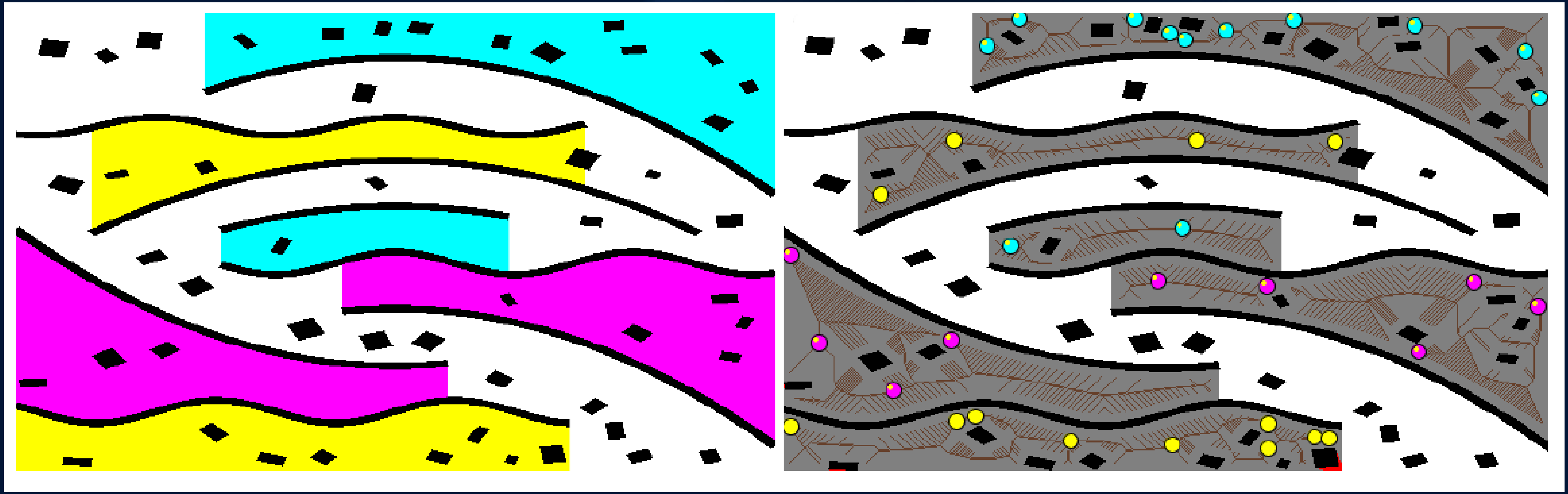




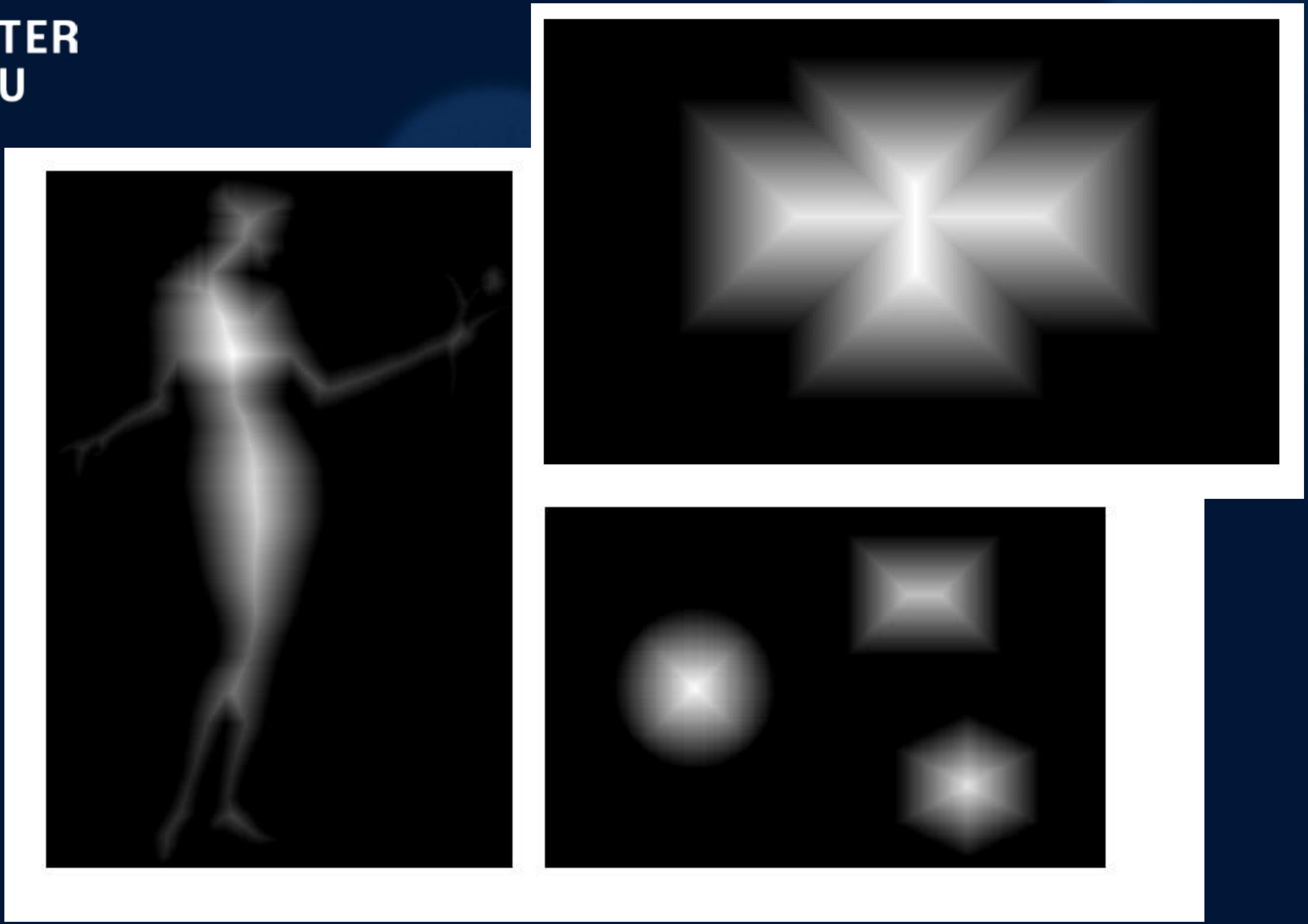
AI CENTER
FEE CTU

# Skeletonization via Grassfiring (Medial Axis) and Filtering



AI CENTER
FEE CTU

# Grassfiring



AI CENTER
FEE CTU

# Generating Inspection Waypoints (1)


**Input:** $\mathcal{I}$: **bitmap image;** $\alpha$: **desired inspection quality,** $0 < \alpha \leq 1$

**Output: a set of inspection points**

---

$h \leftarrow \mathrm{height}(\mathcal{I})$; $w \leftarrow \mathrm{width}(\mathcal{I})$; $\mathcal{B} \leftarrow \mathrm{zeros}(h, w)$

$\diamondsuit$ ***grassfire transformation***

**for** $(i, j) \in \{0, \ldots, h-1\} \times \{0, \ldots, w-1\}$ **do**

  **if** $\mathrm{color}(\mathcal{I}(i,j)) \notin \{\mathtt{black}, \mathtt{gray}\}$ **then**

    $\mathcal{B}(i,j) \leftarrow 1 + \min\{\mathcal{B}(i-1,j), \mathcal{B}(i,j-1)\}$

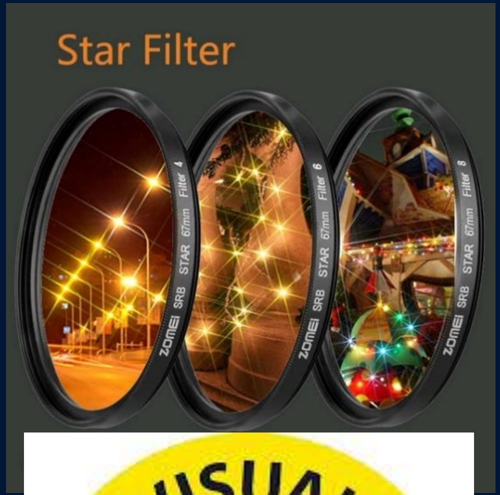
**for** $(i, j) \in \{h-1, \ldots, 0\} \times \{w-1, \ldots, 0\}$ **do**

  **if** $\mathrm{color}(\mathcal{I}(i,j)) \notin \{\mathtt{black}, \mathtt{gray}\}$ **then**

    $\mathcal{B}(i,j) \leftarrow 1 + \min\{\mathcal{B}(i+1,j), \mathcal{B}(i,j+1)\}$

$\mathrm{skeleton} \leftarrow$ **extract pixels making up the most intense lines in the brightness map** $\mathcal{B}$

# Generating Inspection Waypoints (2)





**Input:** $\mathcal{I}$: **bitmap image;** $\alpha$: **desired inspection quality,** $0 < \alpha \leq 1$
**Output: a set of inspection points**

---

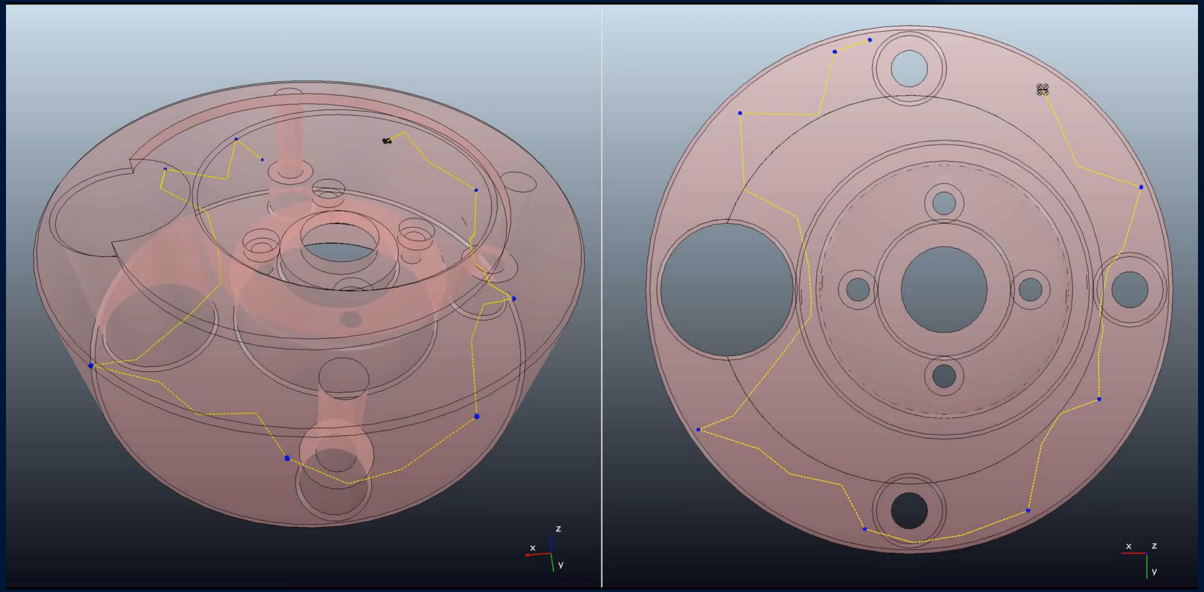
$\diamondsuit$ ***select inspection points***

```
skeleton ← FILTER(skeleton)
inspectionPts ← skeleton
currScore ← VISSCORE(I, inspectionPts)
for p ∈ skeleton do
    newScore ← VISSCORE(I, inspectionPts \ {p})
    if newScore ≥ α ∨ currScore = newScore then
        inspectionPts ← inspectionPts \ {p}
        currScore ← newScore
return inspectionPts
```

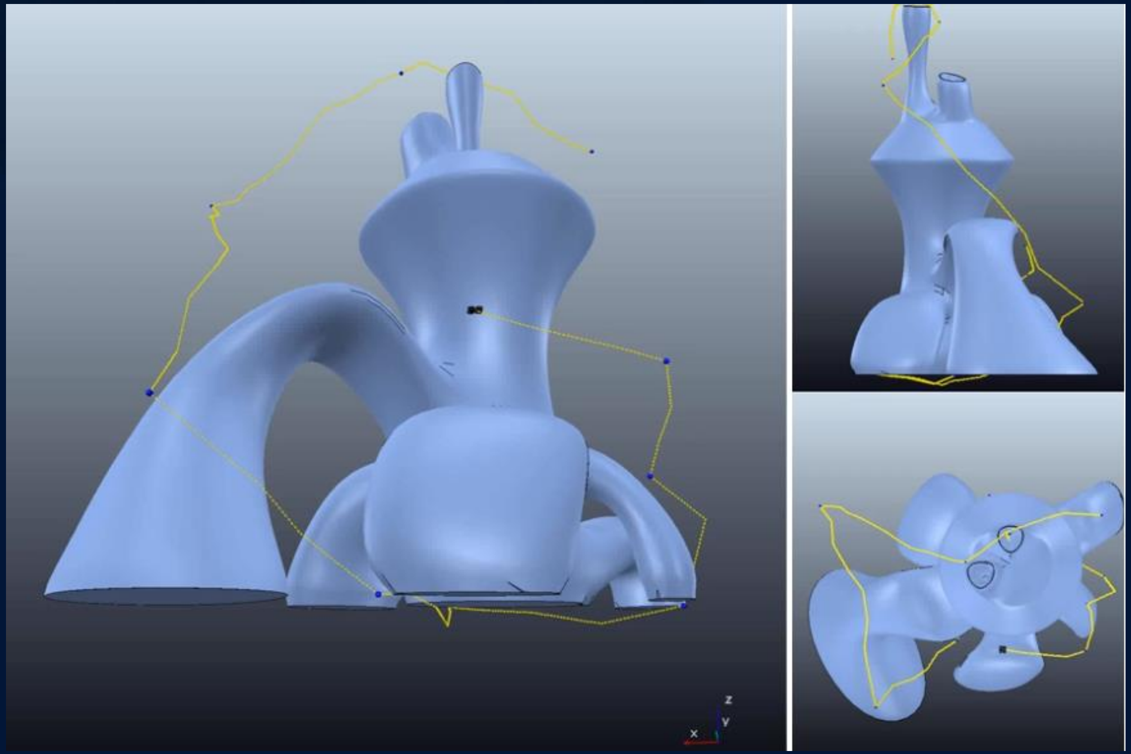
AI CENTER
FEE CTU

# In 3D

## a) Inside

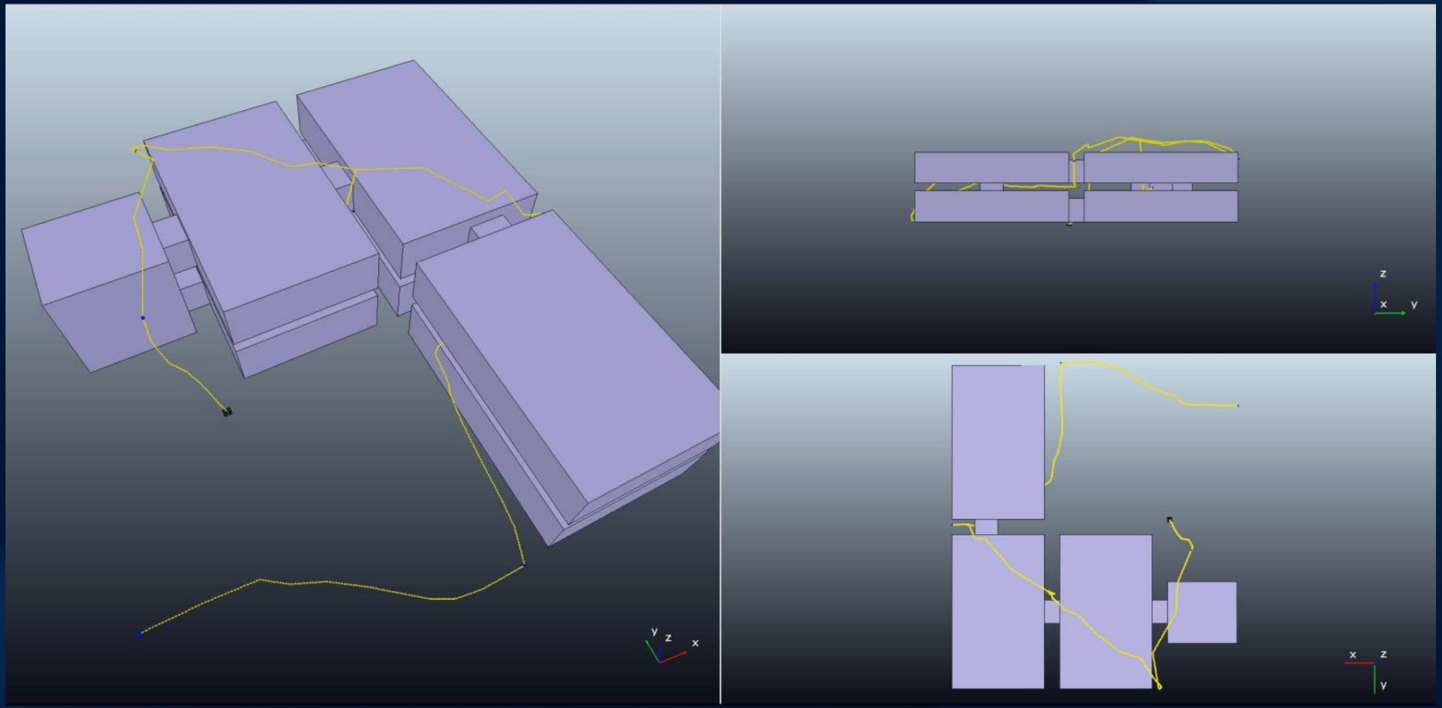


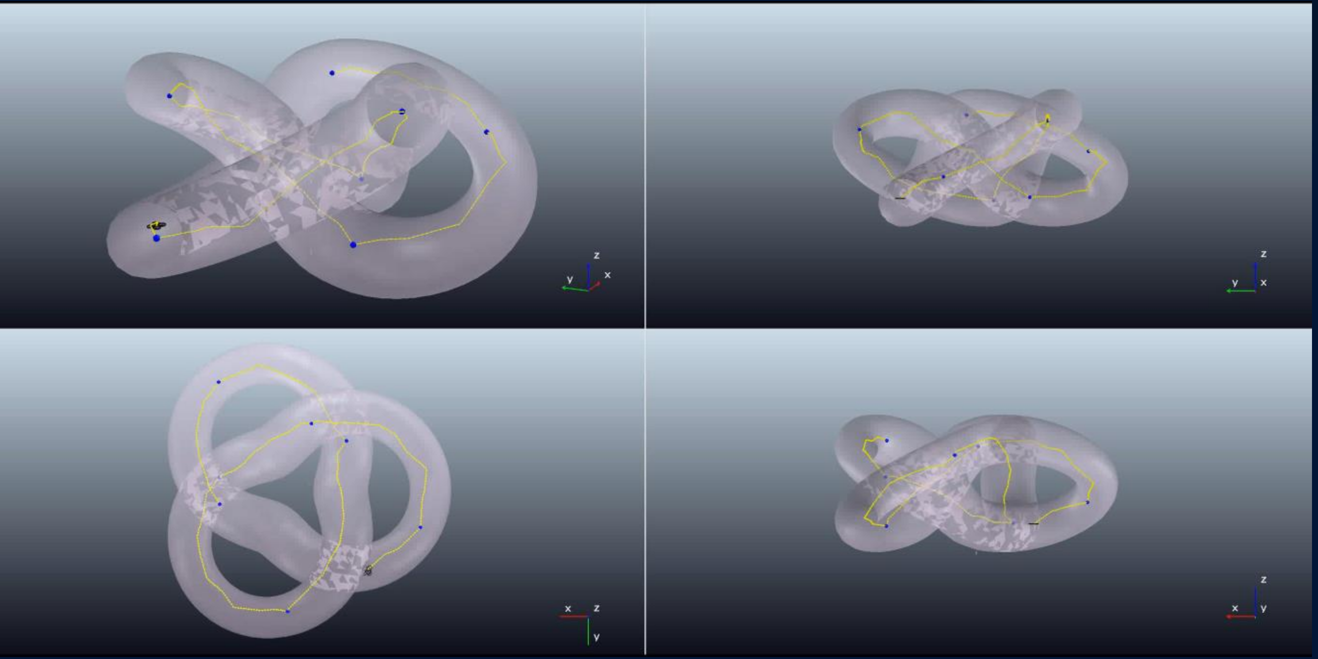
## b) Outside

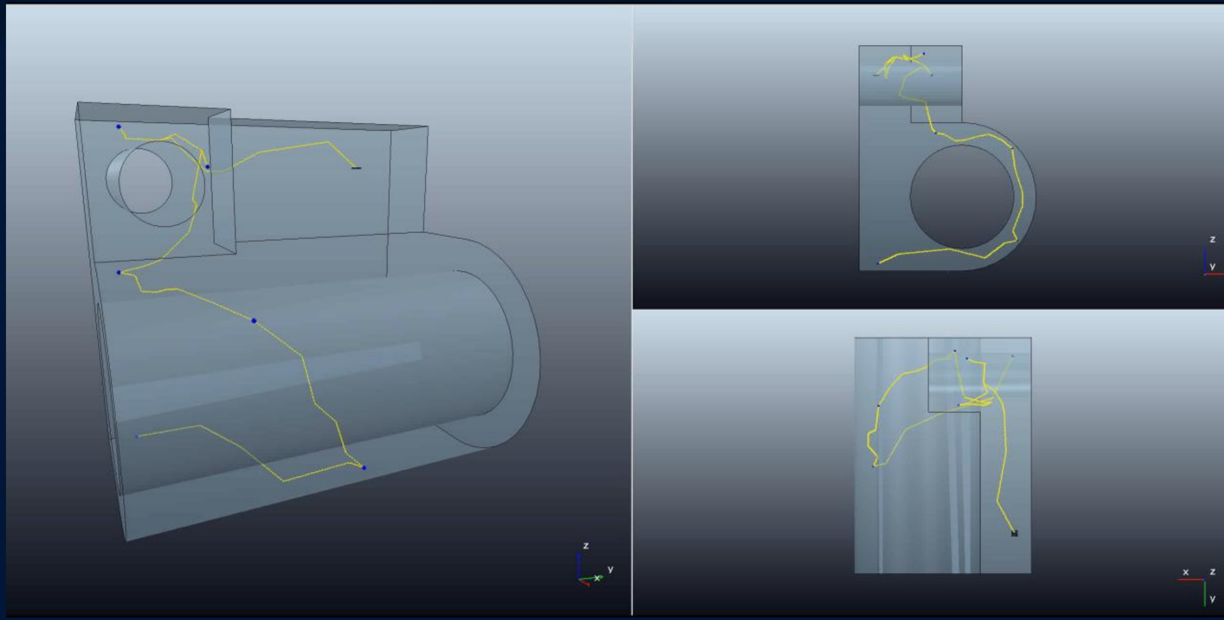


AI CENTER
FEE CTU

# Some more...

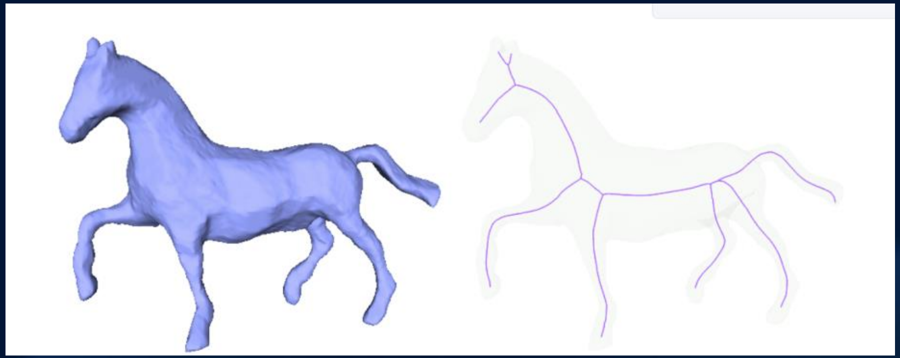






AI CENTER
FEE CTU

# Skeleton, Filtering Algorithm



$$\Sigma(\Omega) = \{x \in \Omega \mid \exists a, b \in \Delta \wedge \ a \neq b \wedge \ ||x - a||_2 = ||x - b||_2\}.$$

**Algorithm 1.** GenerateInspectionPts($\mathcal{W}, \mathcal{O}, \mathcal{R}, \alpha$)

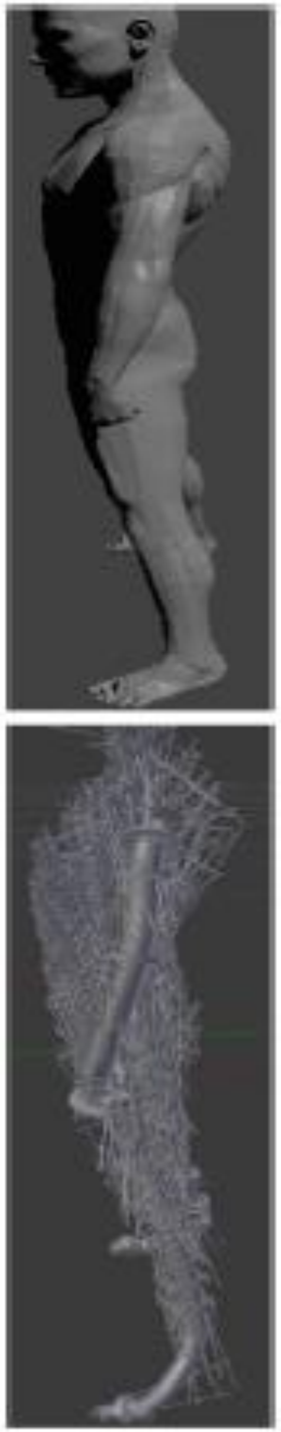
**Input:** $\mathcal{W}, \mathcal{O}, \mathcal{R}$ CAD objects
$\alpha$: desired inspection quality, $0 < \alpha \leq 1$
**Output:** a set of inspection waypoints

$skeleton \leftarrow$ SKELETONIZE($\mathcal{W}$)
$V_1 \leftarrow$ SAMPLE($skeleton$)
$V_2 \leftarrow$ SAMPLE($\mathcal{R}$)
$adjacent \leftarrow$ FilterInspection($V_1, V_2, \alpha$);
**return** HittingSet($adjacent, \alpha$)

**Algorithm 2.** FilterInspectionPts($V_1, V_2, \alpha$)

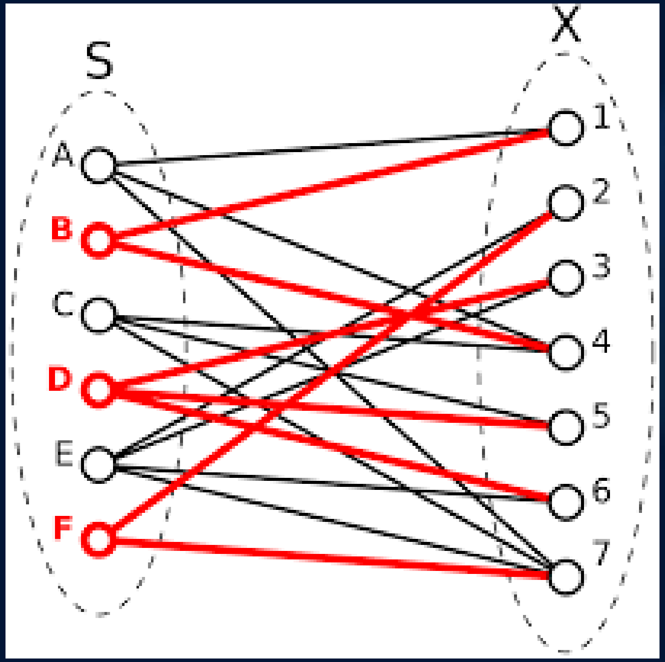
**Input:** $V_1, V_2$, $\alpha$: desired inspection quality, $0 < \alpha \leq 1$
**Output:** a set of inspection points

**for** $(i, j) \in \{0, \ldots, |V_1| - 1\} \times \{0, \ldots, |V_2| - 1\}$ **do**
    $adjacent(i, j) \leftarrow$ VISIBLE($v_i, v_j$)
**return** $adjacent$

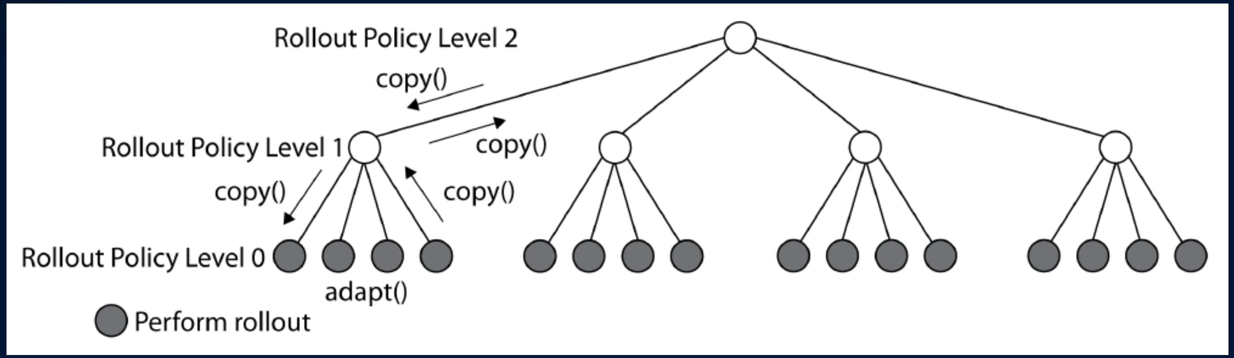


AI CENTER
FEE CTU

# Hitting Set (Exact Cover) Filtering

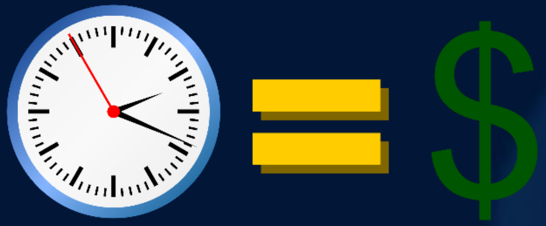



## Using NRPA again



AI CENTER
FEE CTU

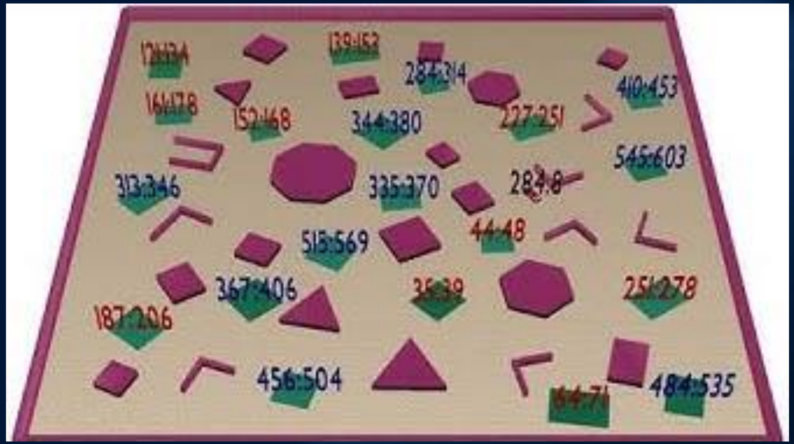
# Temporal Task-Motion Planning

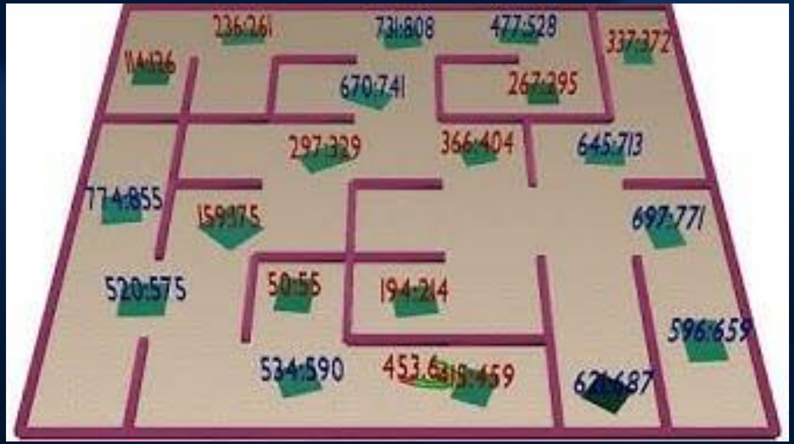



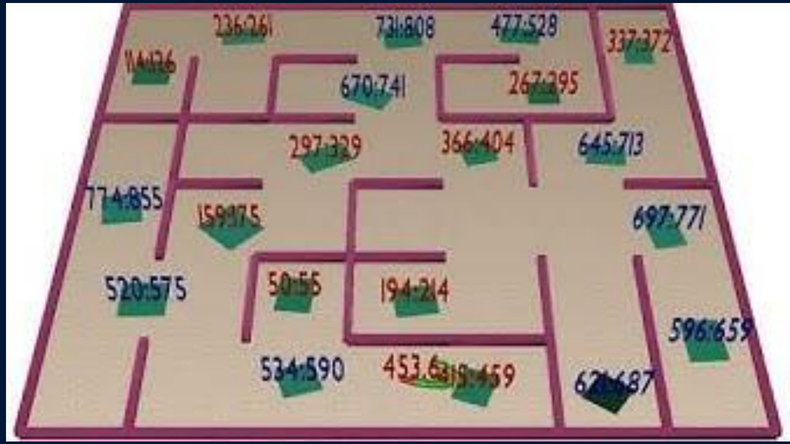
Time is money
Real-world has and needs time constraints
Combining task with motion planning “holy grail”

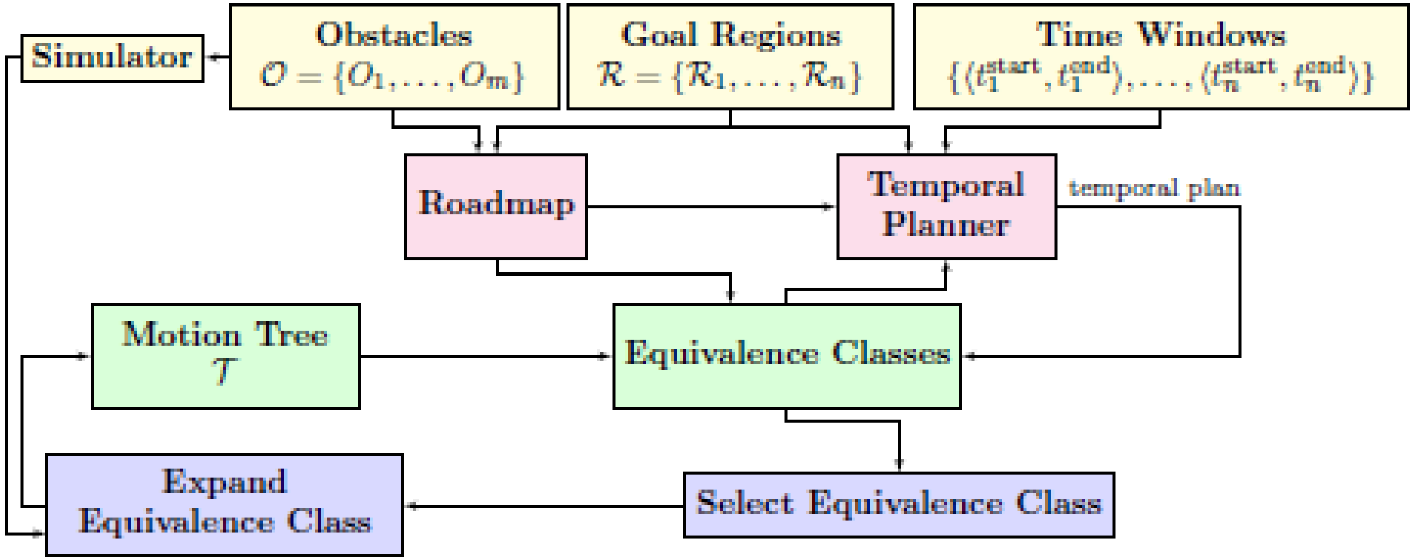
AI CENTER
FEE CTU

# Examples









AI CENTER
FEE CTU

# Integration of Automated Planning

delkamp

AI CENTER
FEE CTU

# Interface with PDDL

TIL = Timed Initial Literal

(at timepoint (fact))
(at timepoint (not fact))

Specified in initial state

➔ actions time windows

```
v1  v3  v5  v4  v2
0.0   1.26   3.22   12.55   21.11
```

```
(at auv v0)
(connected v0 v1)
(connected v0 v2)
(connected v1 v2)
(= (traveltime v0 v1) 0.8)
(= (traveltime v0 v2) 1.5)
(= (traveltime v1 v2) 0.7)
(located task1 v1)
(located task2 v2)
(at 1.1 (tw_open task1))
(at 2.1 (not (tw_open task1)))
(at 2.3 (tw_open task2))
(at 3.3 (not (tw_open task2)))
```

coffee_errors.pddl

```
(define COFFEE

  (requirements
    :typing)

  (:types room - location
          robot human _ agent
          furniture door - (at ?l - location)
          kettle ?coffee cup water - movable
          location agent movable - object)

  (:predicates (at ?l - location ??o - object)
               (have ?m - movable ?a - agent)
               (hot ?m - movable) = true
               (on ?f - furniture ?m - movable))

  (:action boil
    :parameters (?m - movable $k - kettle ?a - agent)
    :preconditions (have ?m ?a)
    :effect (hot ?m))

```

Line 20, Column 22    Spaces: 2    PDDL

AI CENTER
FEE CTU

# PDDL 3 Planning

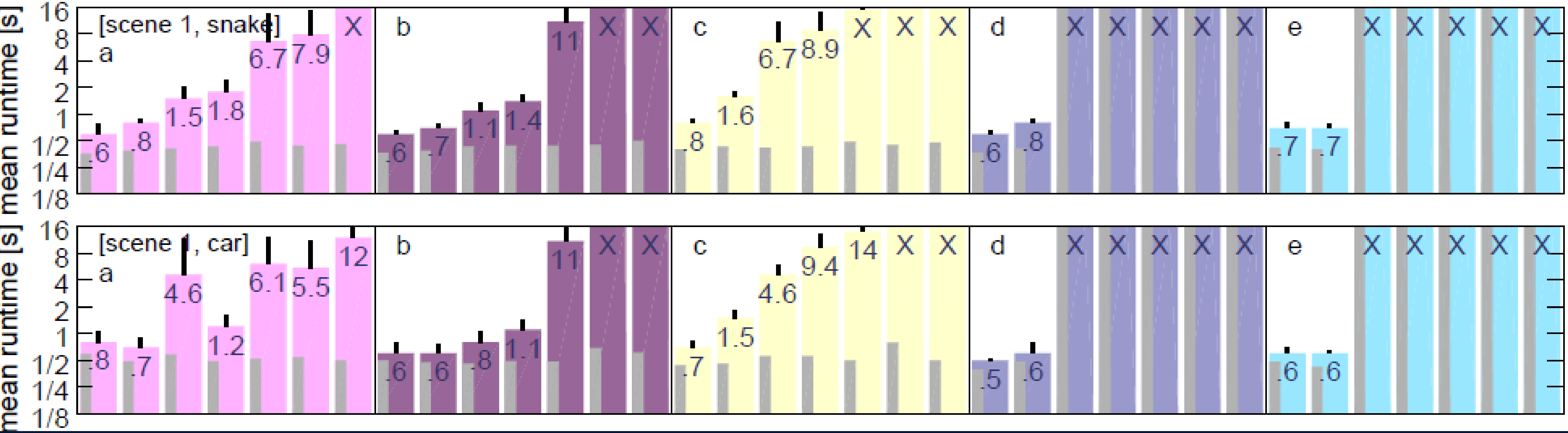


**Monte-Carlo Search for Prize-Collecting Robot Motion Planning with Time Windows, Capacities, Pickups, and Deliveries**

```
(:durative-action execute_task_pickup
 :parameters (?v - vehicle ?wp - waypoint ?t - task)
 :duration ( = ?duration (taskduration ?t))
 :condition (and
   (at start (at ?v ?wp)) (at start (located ?t ?wp)
   (at start (todo ?t))
   (at start (<= (+ (customer ?wp) (cap ?v)) (max_cap ?v)))
   (at start (is-pickup ?wp)) (at start(tw_open ?t)))
 :effect (and
   (at start (not (todo ?t))) (at end (visited ?wp))
   (at end (increase (cap ?v) (customer ?wp)))
   (at end (decrease (profit ?v) (customer ?wp)))
   (at end (completed ?t)))
(:durative-action execute_task_delivery
 :parameters (?v - vehicle ?wp1 ?wp2 - waypoint ?t - task)
 :duration ( = ?duration (taskduration ?t))ov
 :condition (and
   (at start (at ?v ?wp1)) (at start (located ?t ?wp1))
   (at start (todo ?t)) (at start (is-delivery ?wp1))
   (at start (and (visited ?wp2) (link ?wp2 ?wp1)))
   (at start (tw_open ?t)))
 :effect (and
   (at start (not (todo ?t))) (at end (visited ?wp1))
   (at end (increase (cap ?v) (customer ?wp1)))
   (at end (decrease (profit ?v) (customer ?wp1)))
   (at end (completed ?t))))
```

AI CENTER
FEE CTU

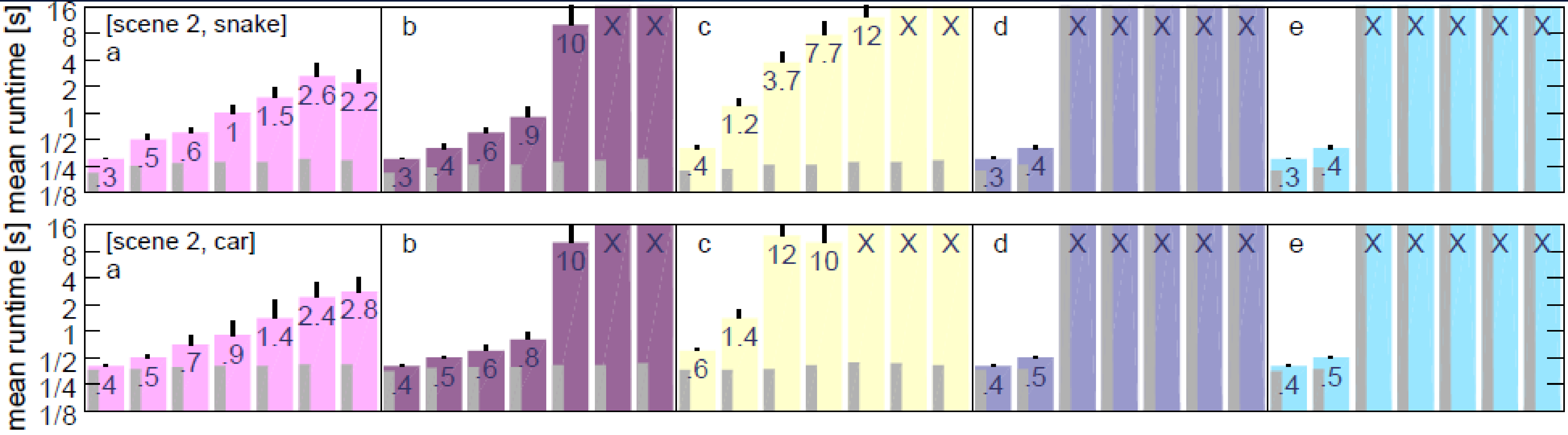

# NRPA BnB Optic OPTIMAL Random



elkamp

AI CENTER
FEE CTU

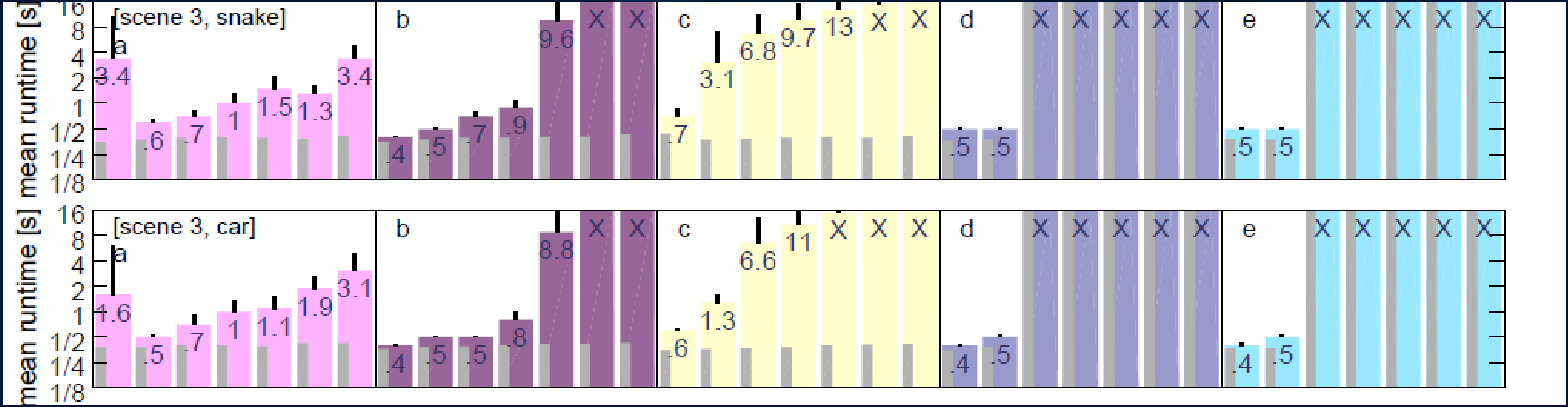

**NRPA** **BnB** **Optic** **OPTIMAL** **Random**



delkamp

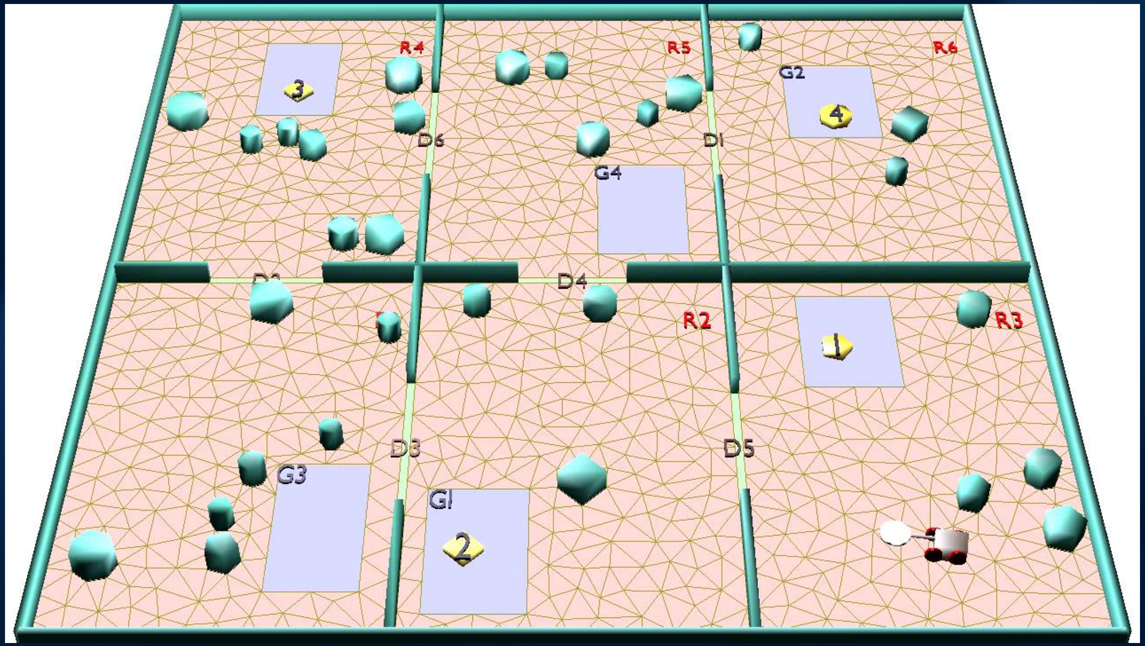
AI CENTER
FEE CTU

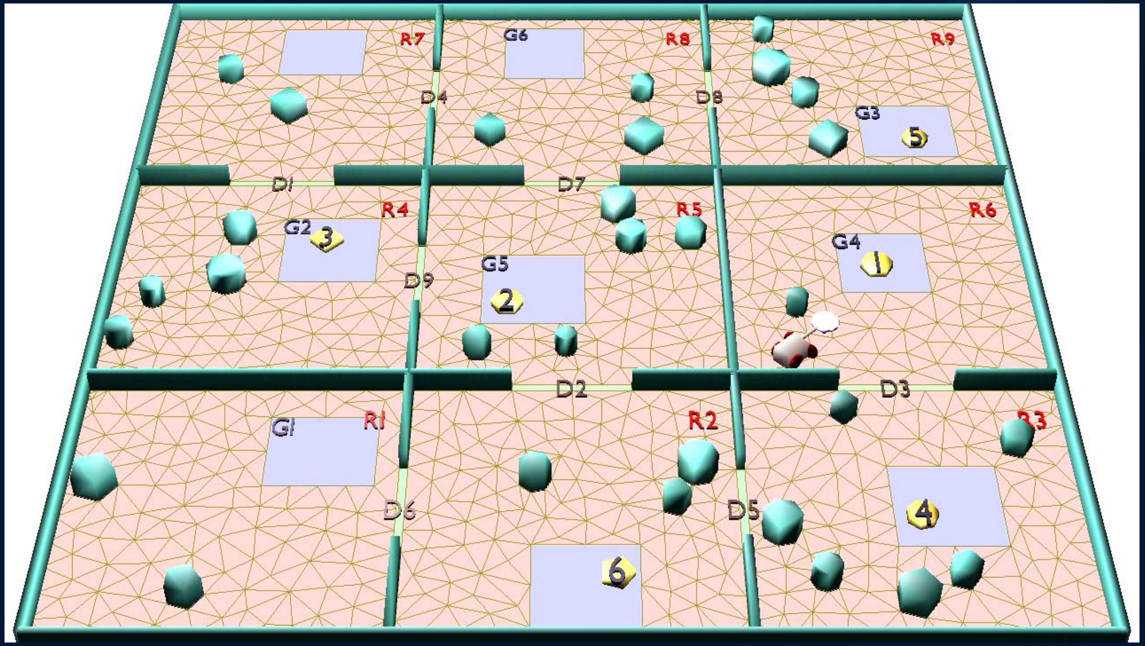

NRPA BnB Optic OPTIMAL Random

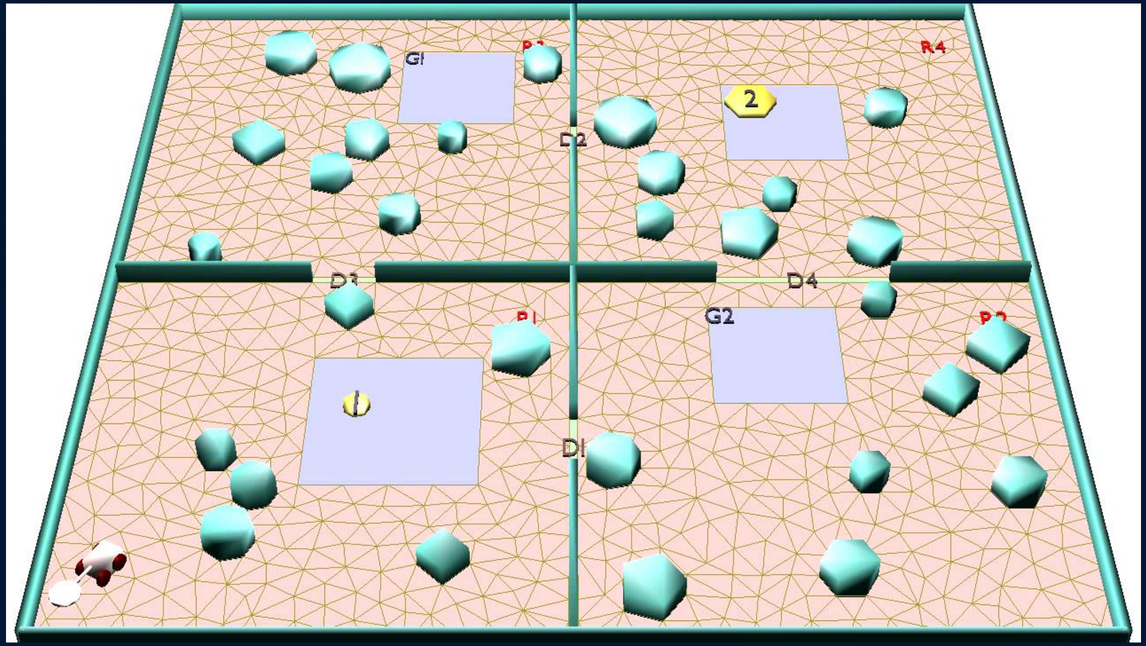


AI CENTER
FEE CTU

# Combined Task and Motion

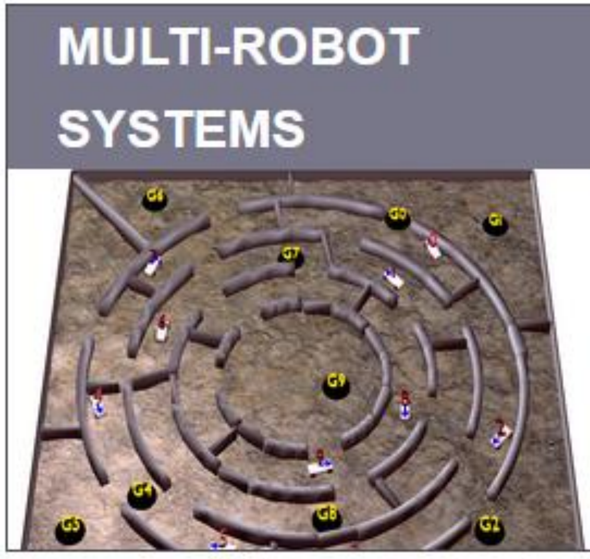


0

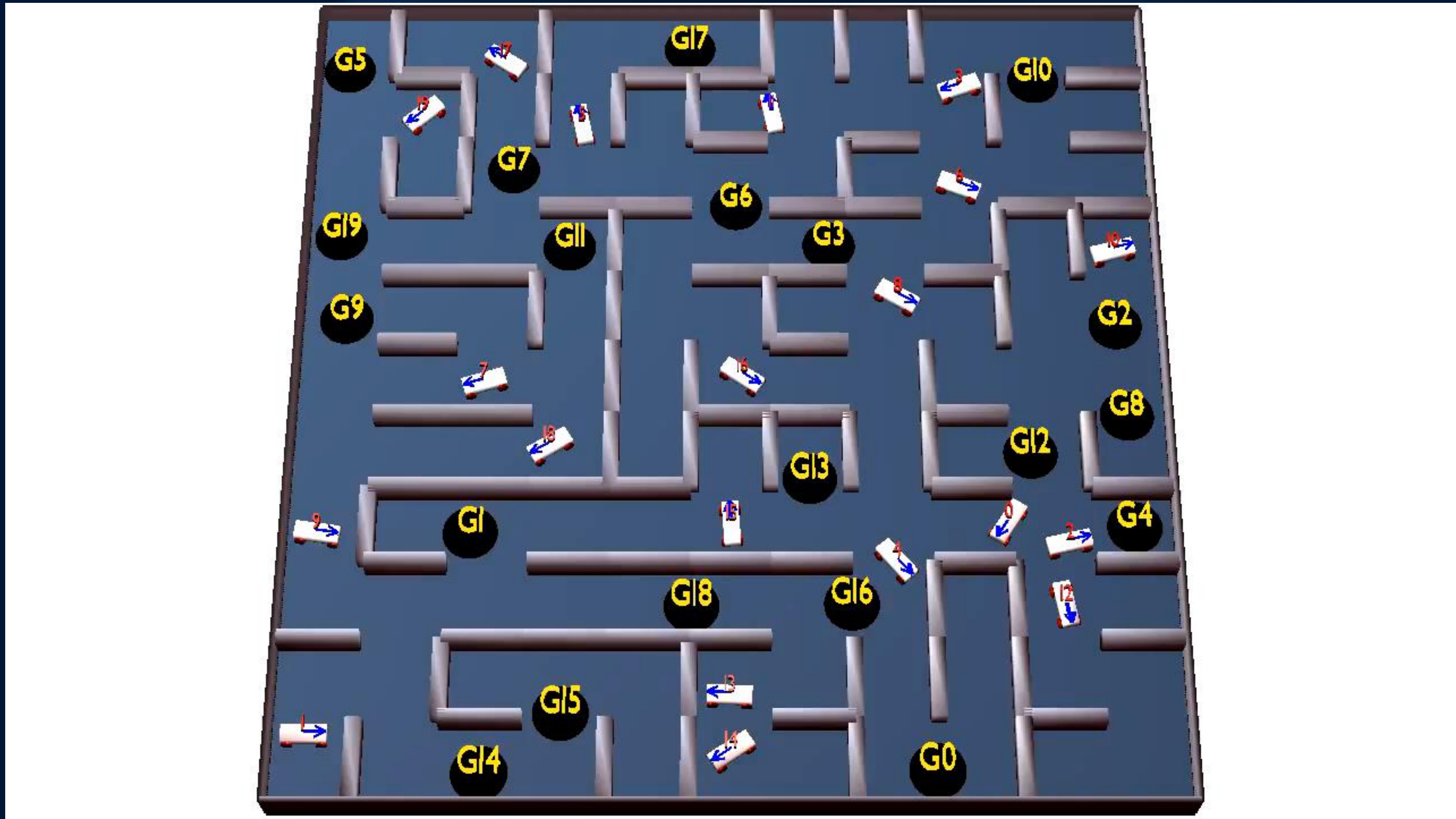
AI CENTER
FEE CTU




# More Robots

→ MAPF as
discrete problem

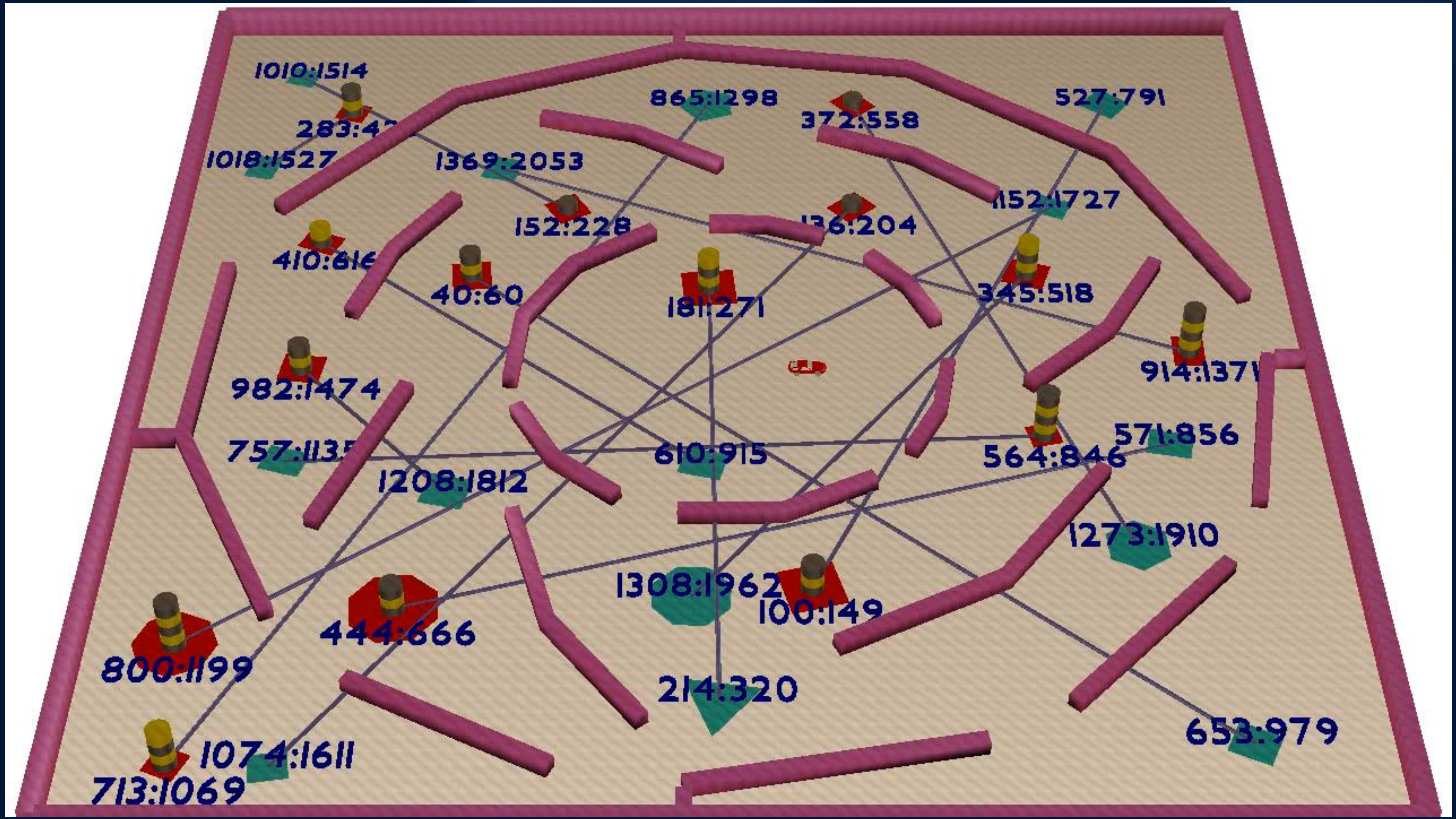
Multigoal Multirobot
Planning?



1

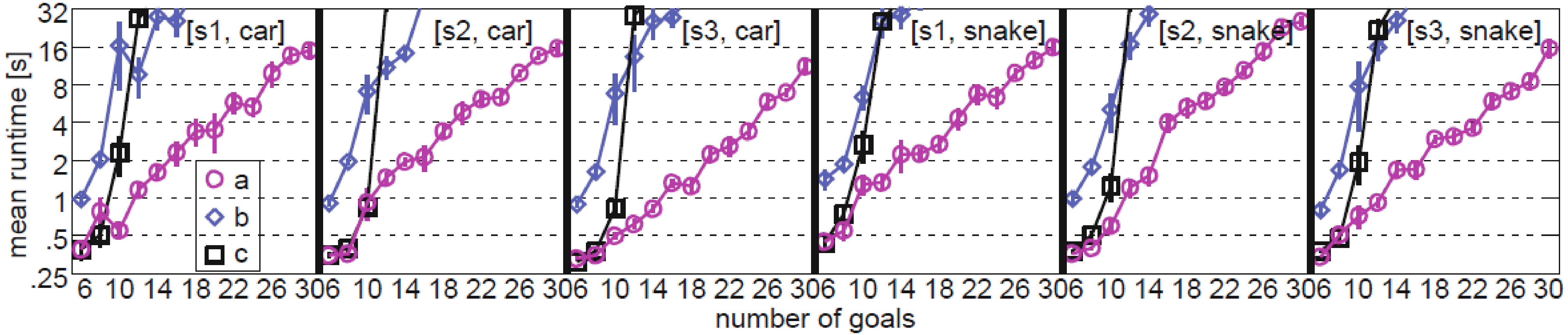
AI CENTER
FEE CTU

# Goods

AI CENTER
FEE CTU


# (a) MC (B) OPTIC (C) Random



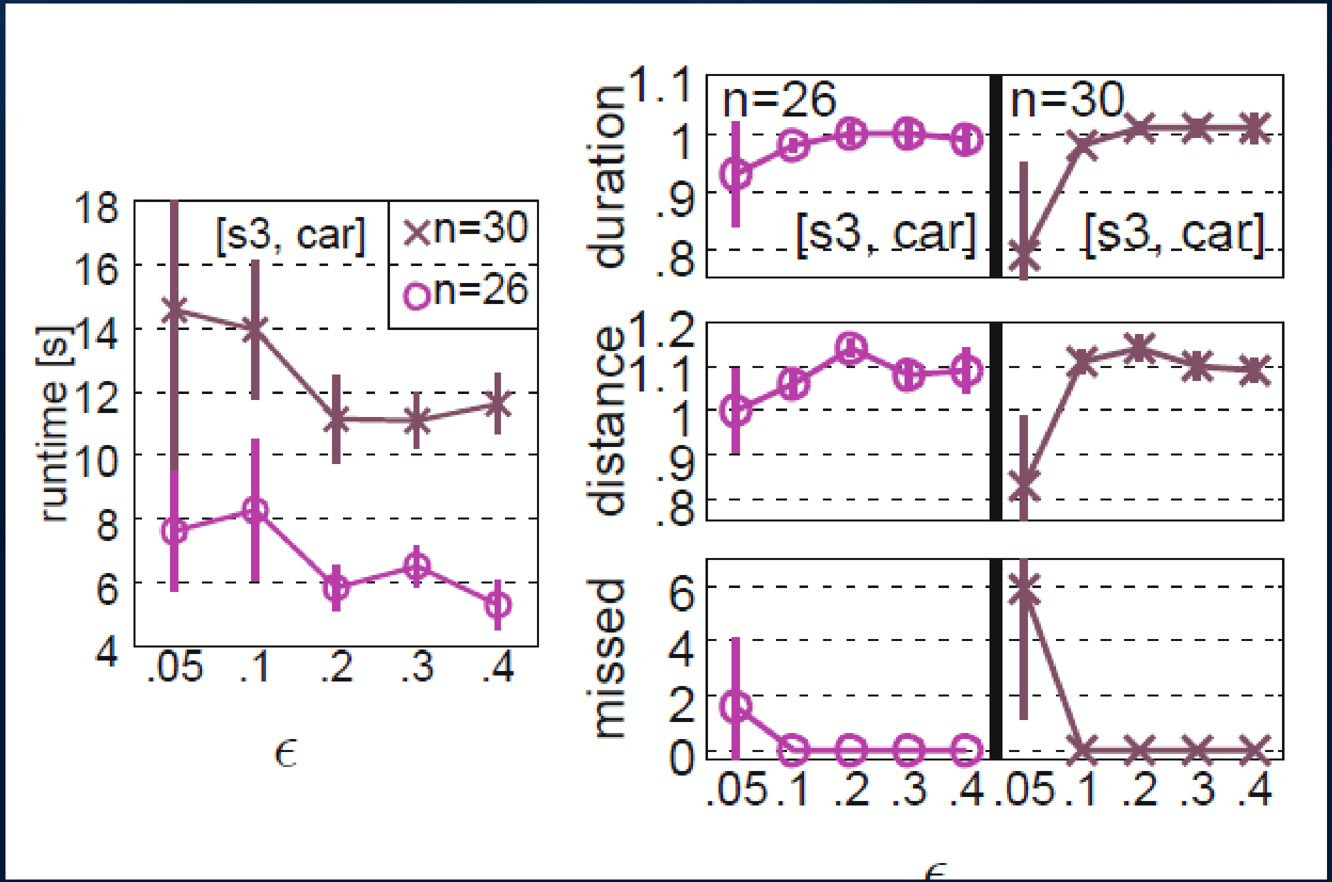
delkamp

AI CENTER
FEE CTU

# Adjusting Time Windows



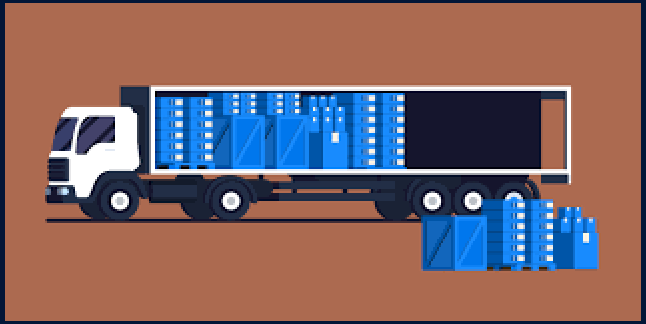
$$[(1-\epsilon)t_i, (1+\epsilon)t_i]$$

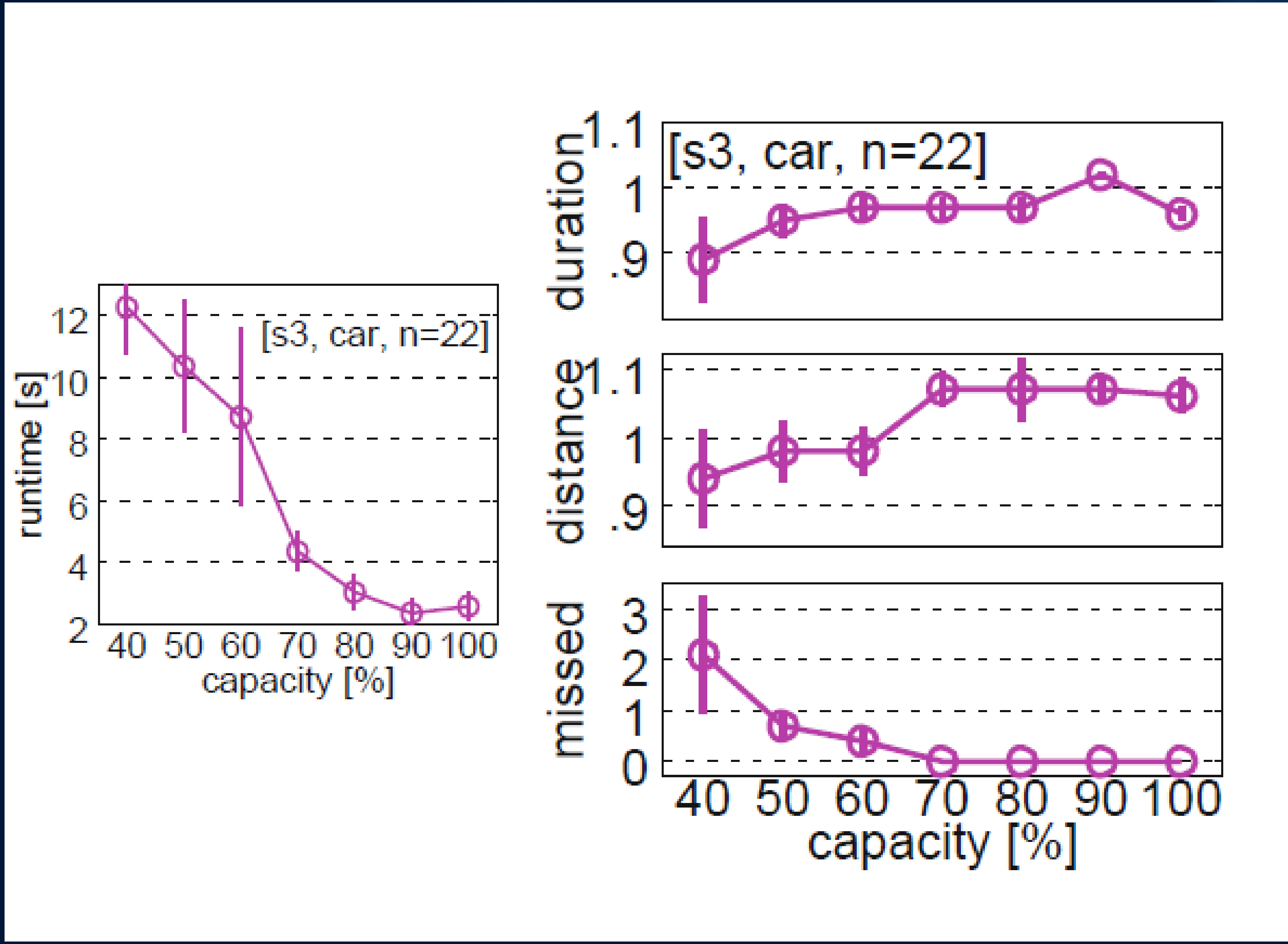


delkamp

AI CENTER
FEE CTU

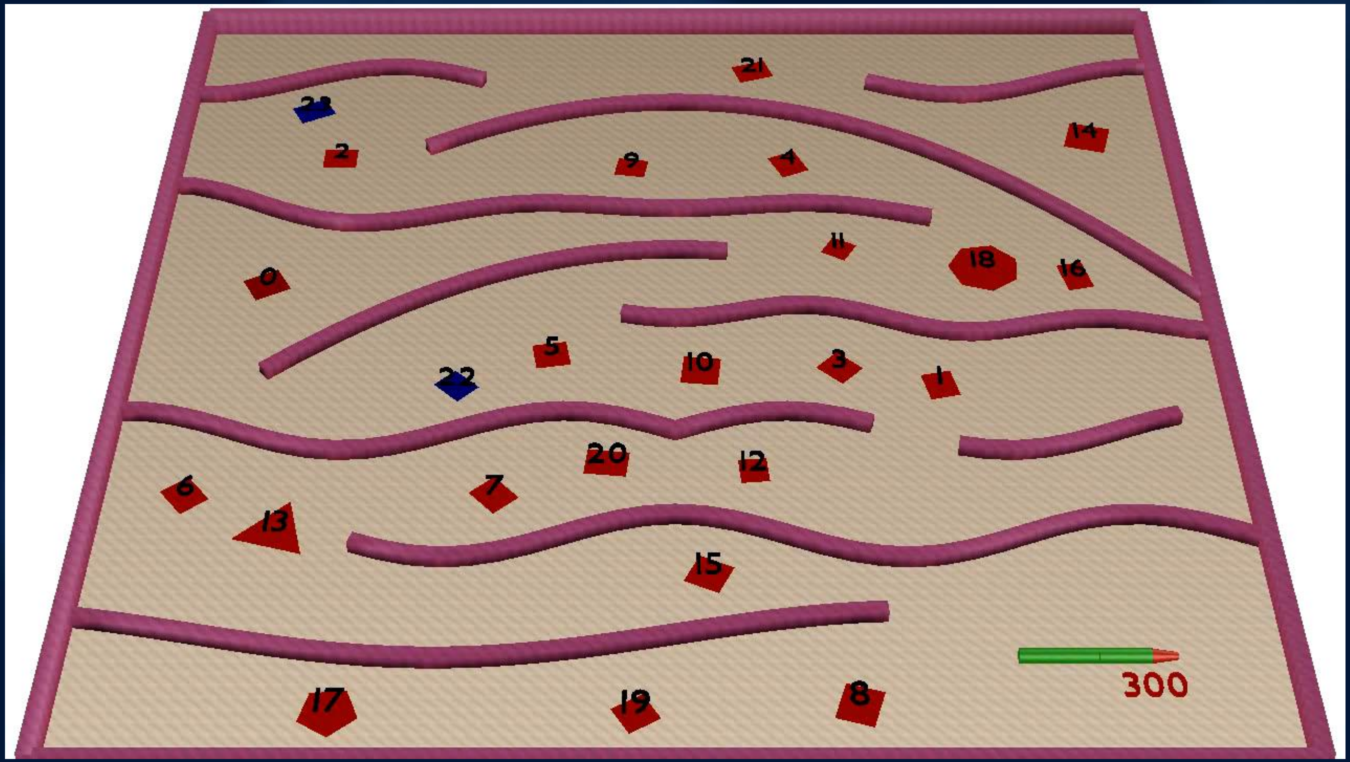
# Adjusting Vehicle Capacity





AI CENTER
FEE CTU

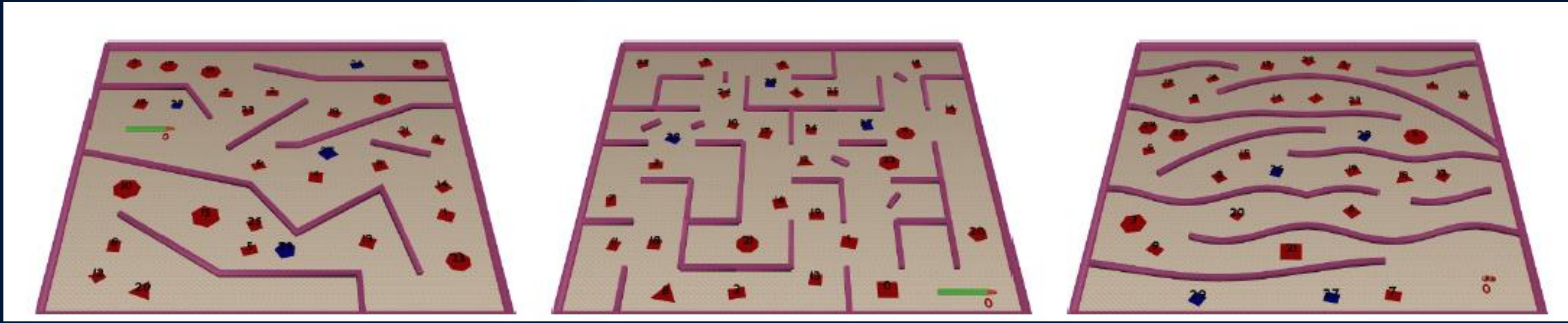
# Energy



AI CENTER
FEE CTU

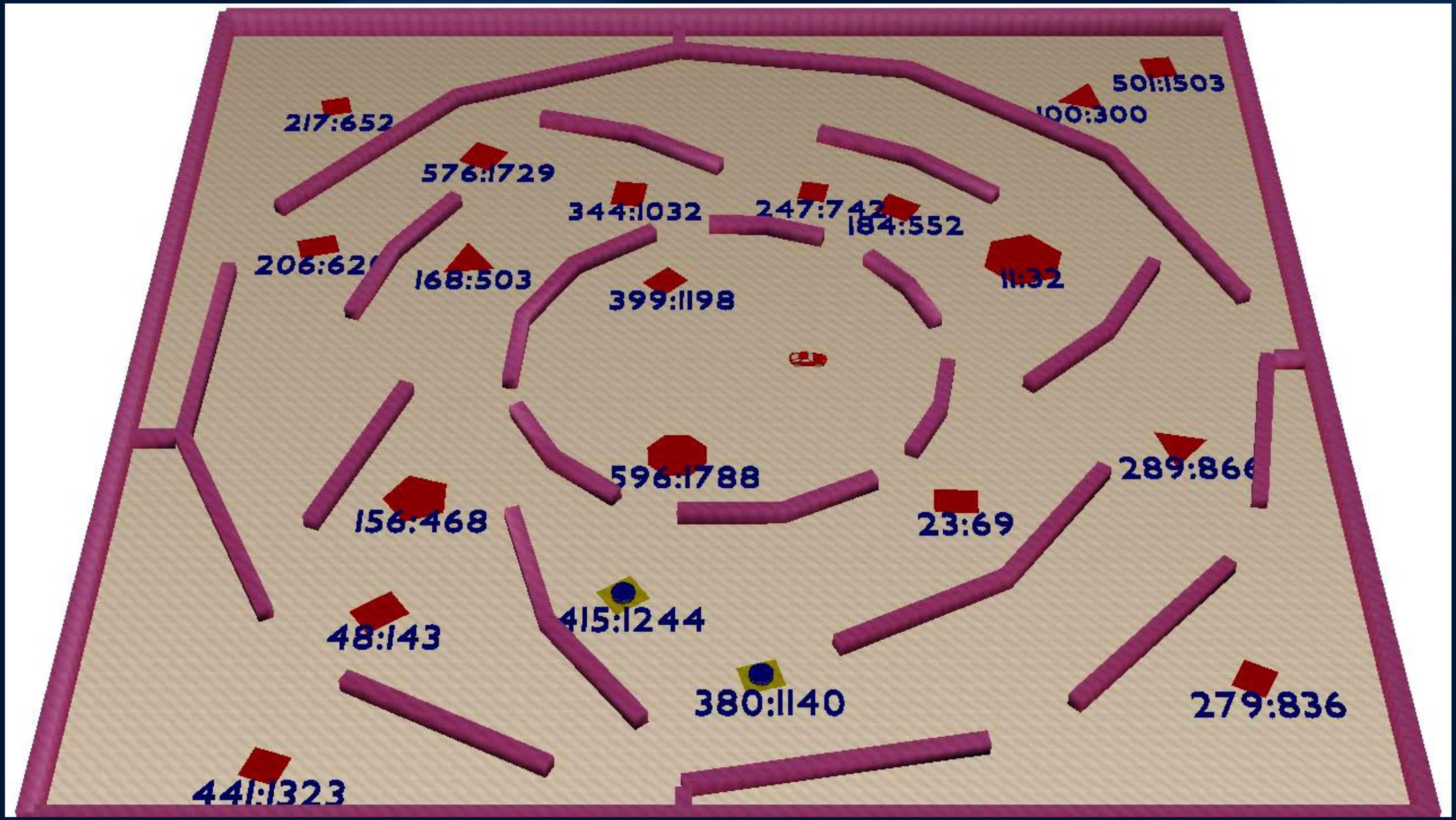
# Recharging

Multi-goal motion planning with multiple recharging stations. Goal regions, recharging stations, and obstacles are shown in red, blue, and magenta, respectively. The robot is required to visit each goal while reducing the distance traveled and the number of recharges

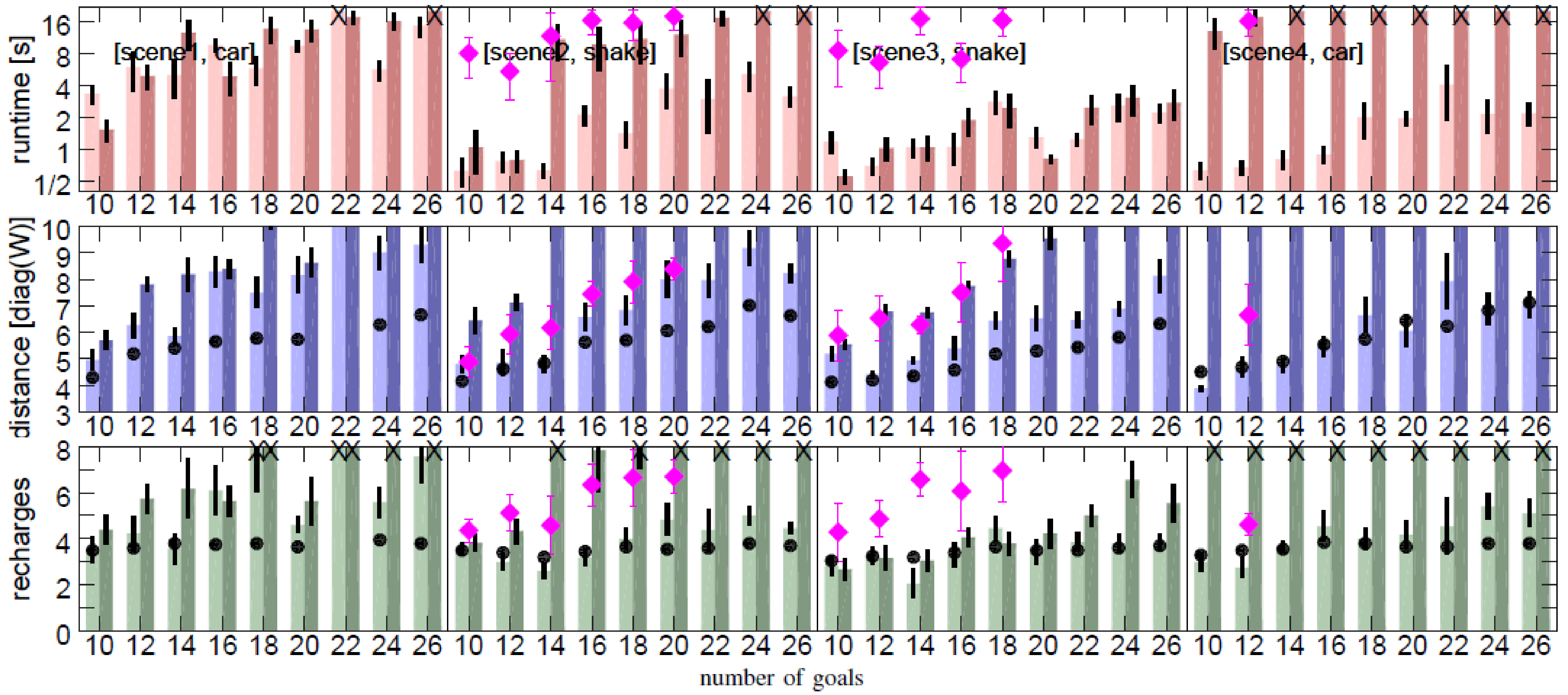


AI CENTER
FEE CTU

# Recharging with Temporal Constraints




# Results



runtime [s]
16
8
4
2
1
1/2
[scene1, car]
[scene2, snake]
[scene3, snake]
[scene4, car]
distance [diag(W)]
10
9
8
7
6
5
4
3
recharges
8
6
4
2
0
10 12 14 16 18 20 22 24 26
number of goals

AI CENTER
FEE CTU

# Conclusion

**Full-Fledged Solution:**

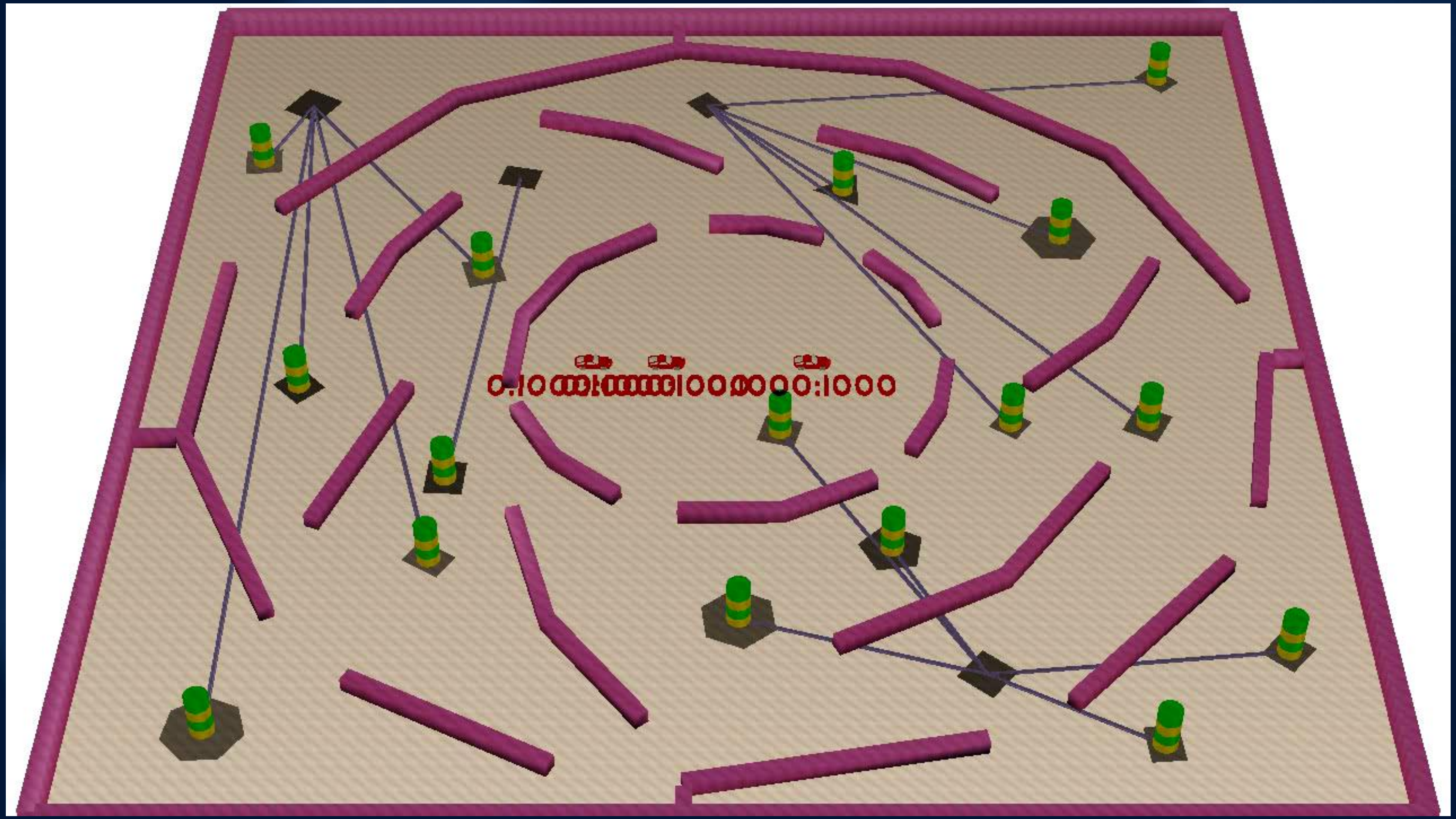
- high-dimensional robotic systems with nonlinear dynamics and
- nonholonomic constraints
- visit all goal regions fast in suitable cost-minimizing order
- unstructured, complex environments

**and efficiently computes**

- collision-free, dynamically-feasible, low-cost trajectories that
- enable the robot to satisfy the task specification

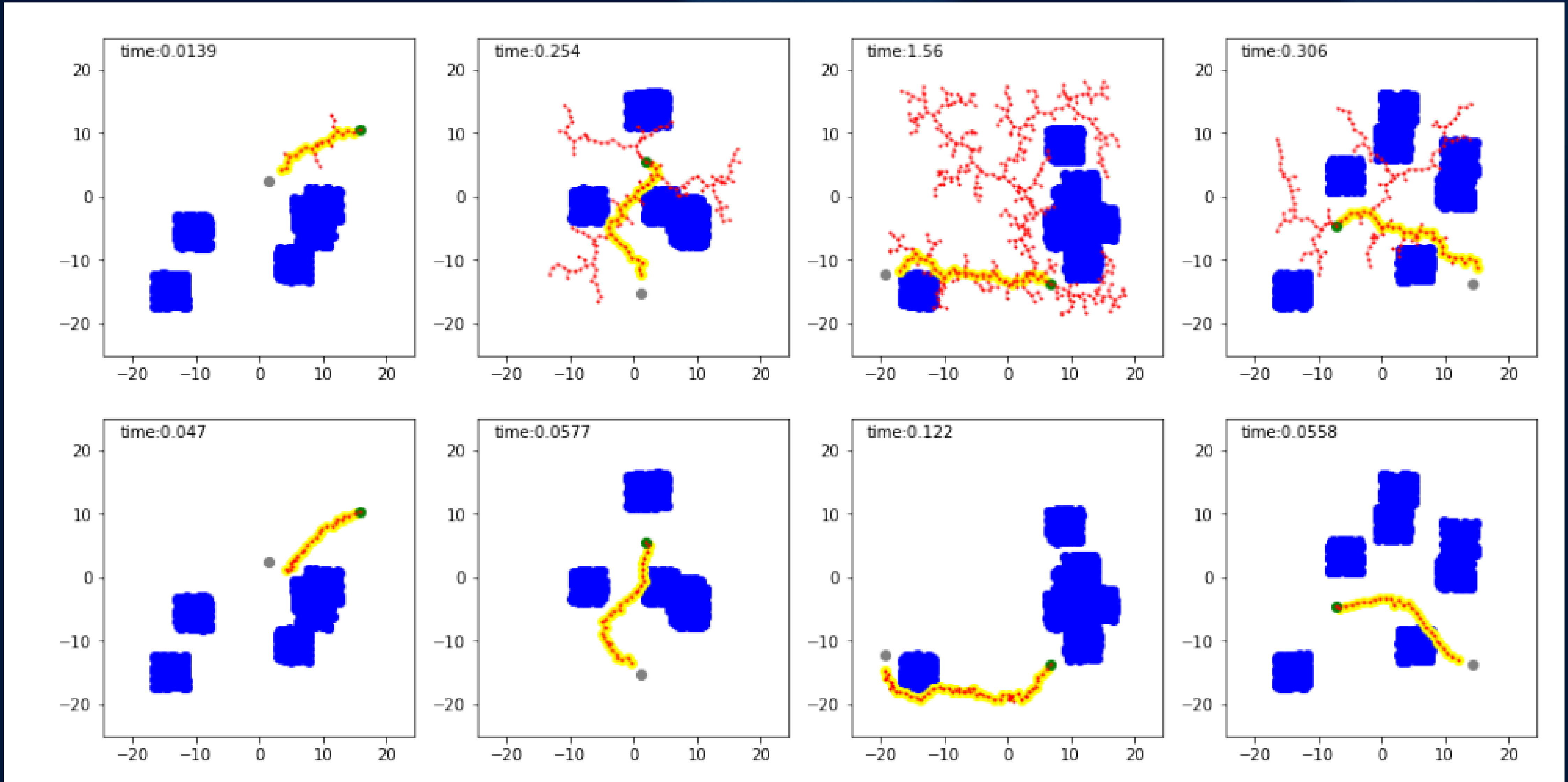
AI CENTER
FEE CTU

# Physical VRP



AI CENTER
FEE CTU

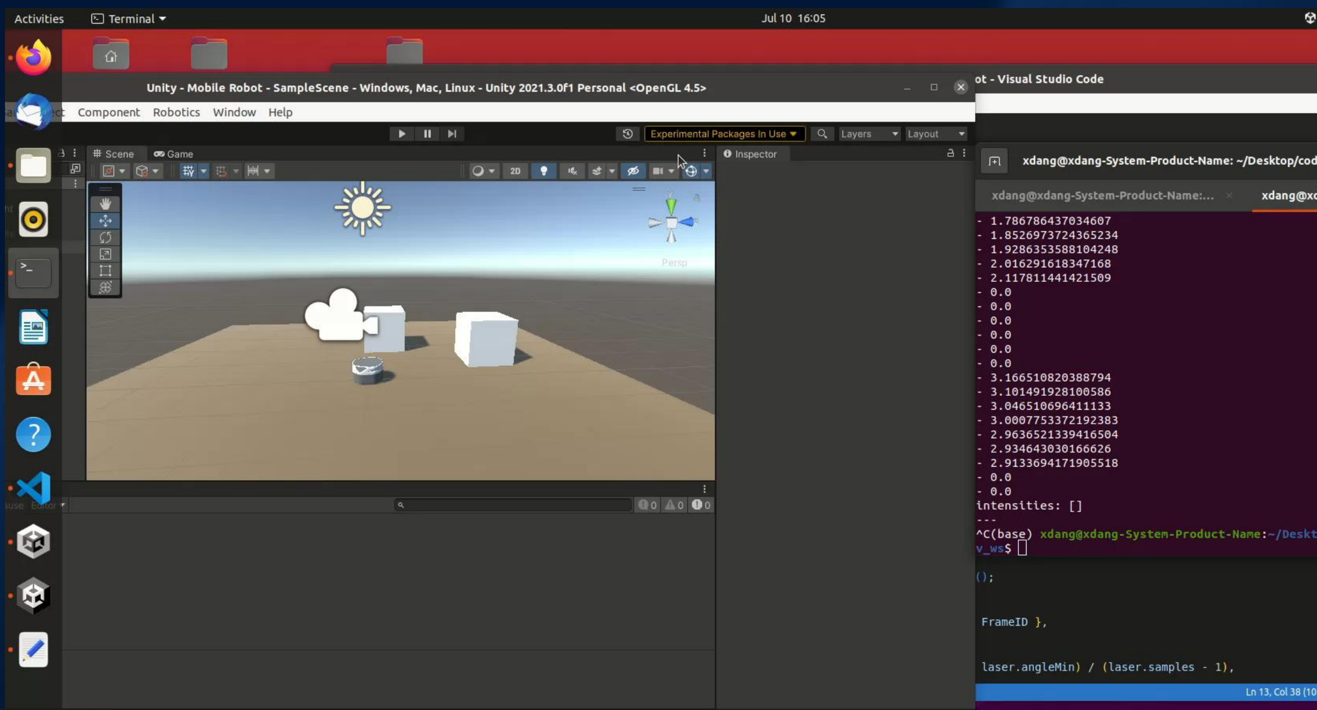
# Deep RRT*

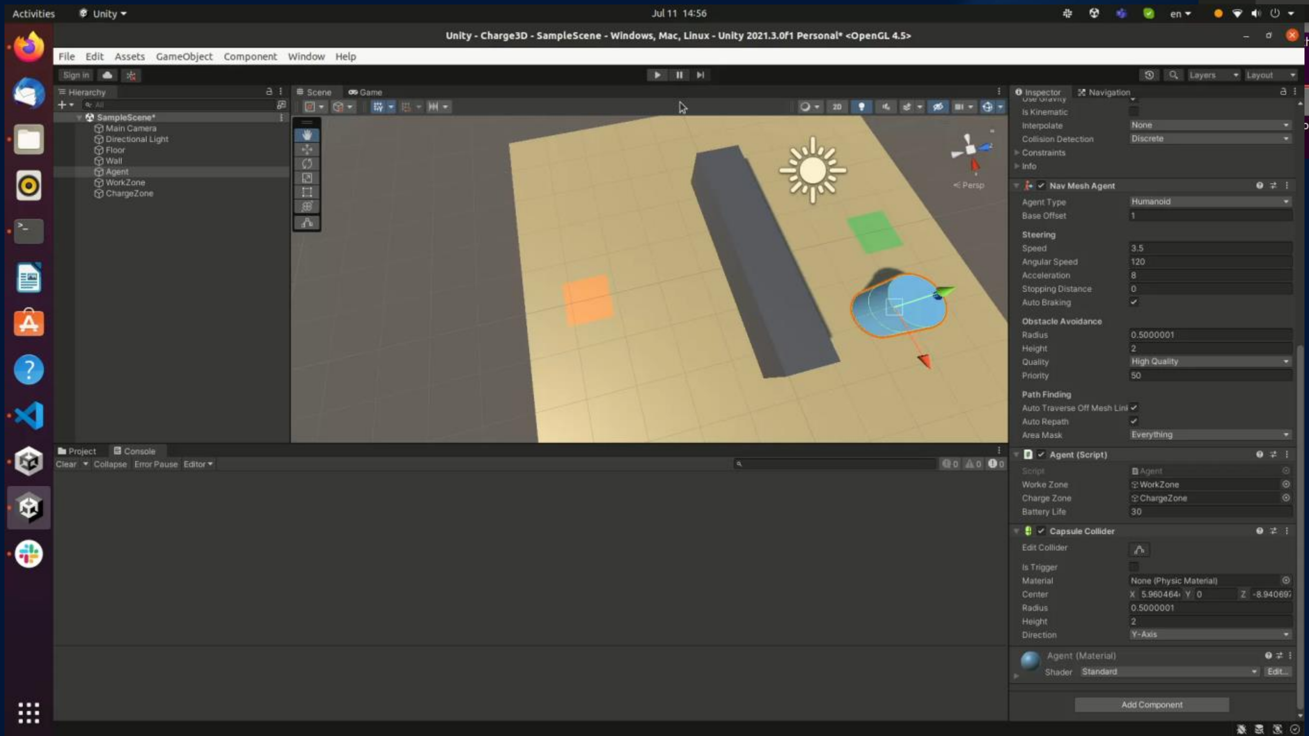


3

AI CENTER
FEE CTU

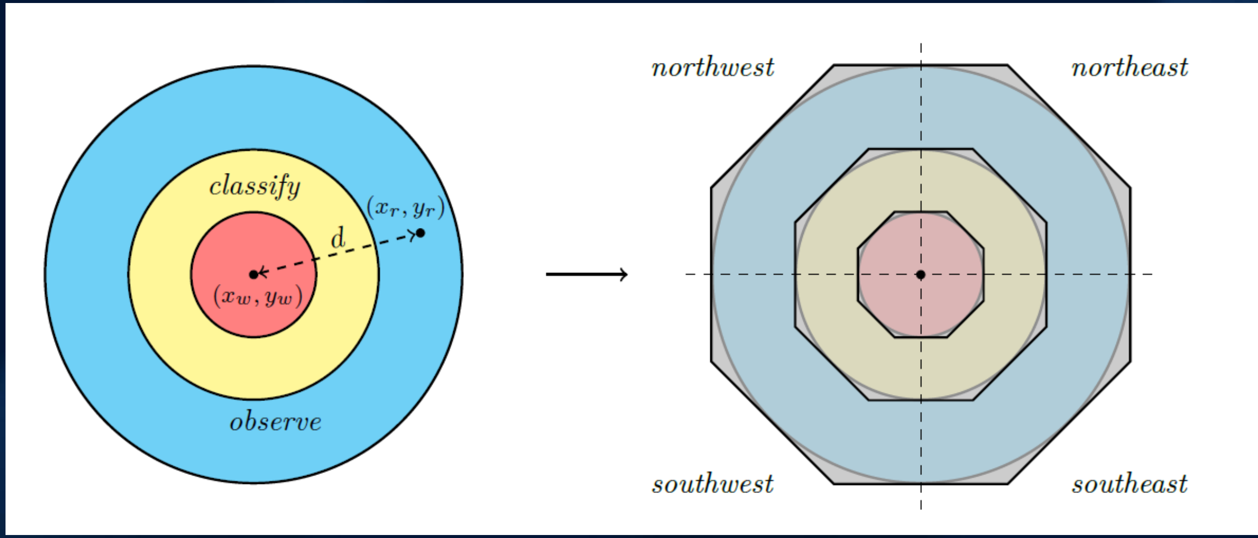
# Unity / ROS [many]

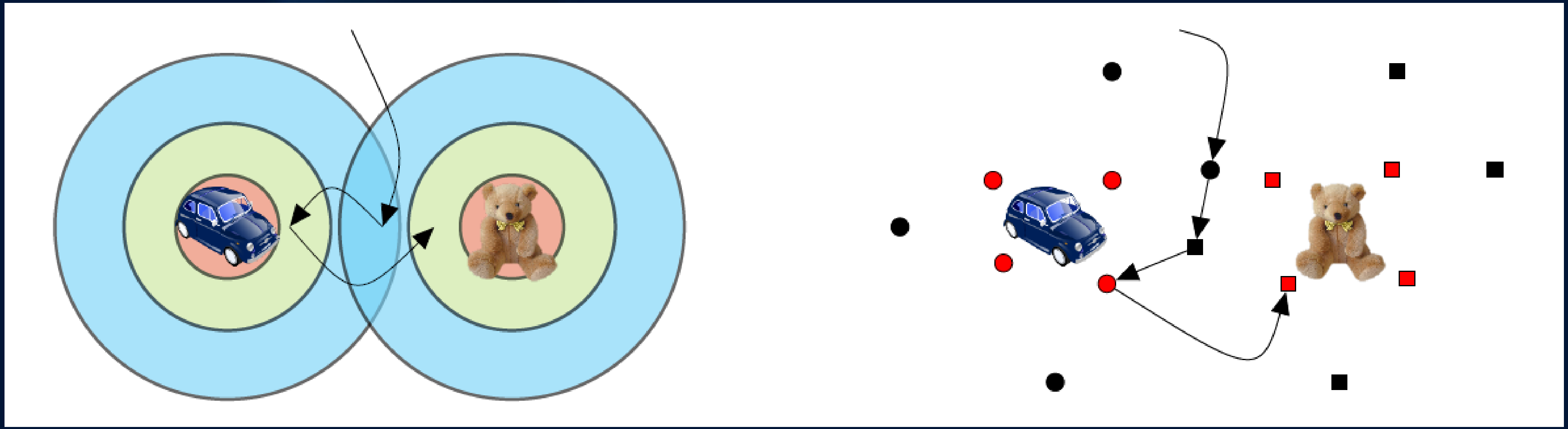


AI CENTER
FEE CTU

# Geometric Reasoning in AI Planning



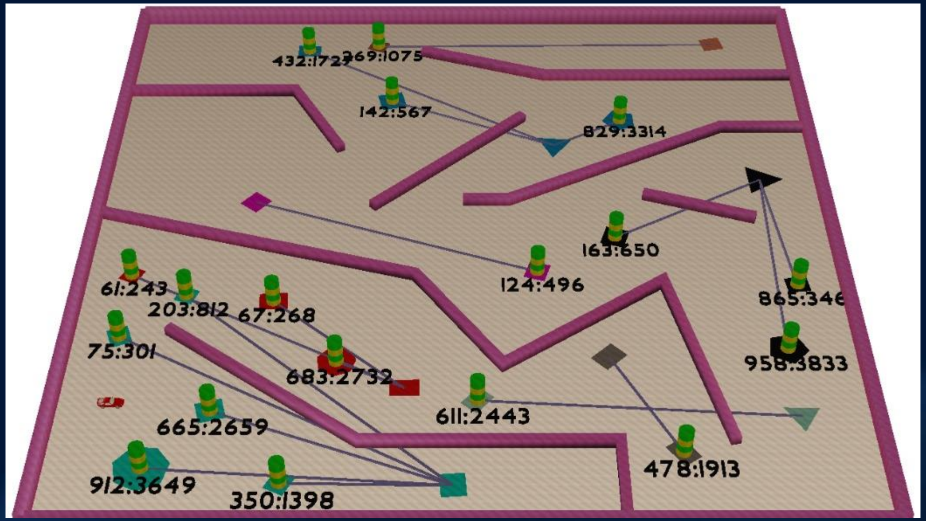
AI CENTER
FEE CTU

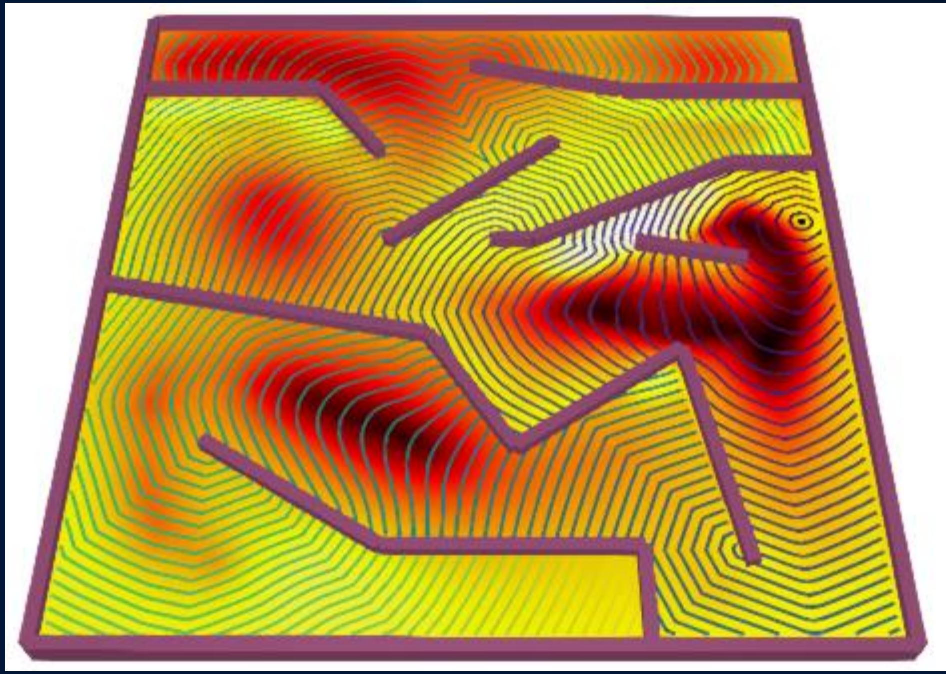
# PDDL can represent a lot

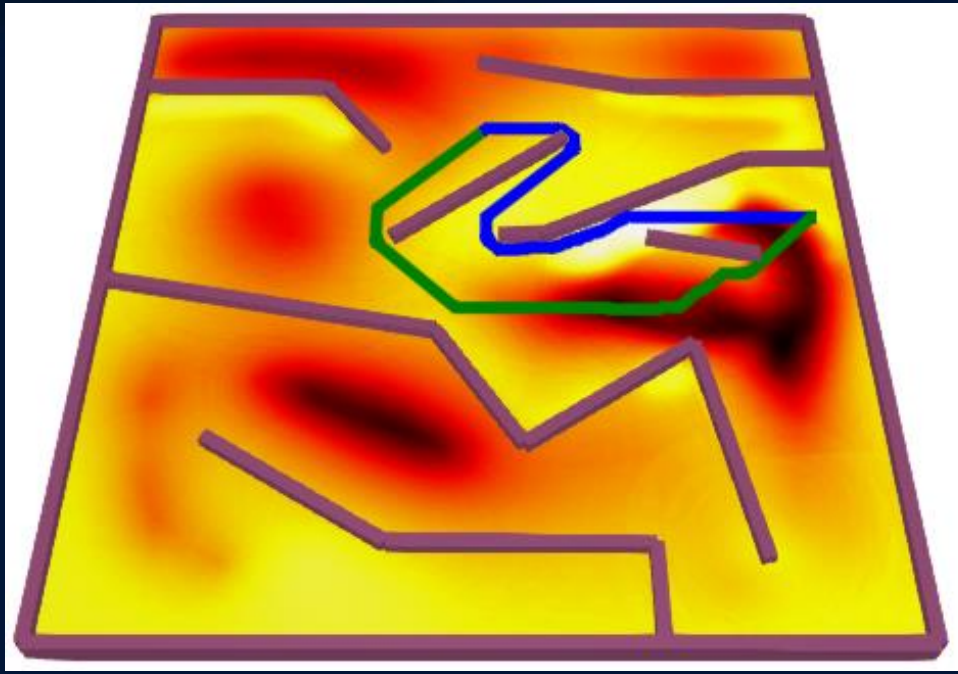
```
(:functions (x_location ?loc - location) (y_location ?loc - location)
    (x_robot) (y_robot) (maxV) (classify_min_distance)
    (observe_min_distance) (classify_min_distance_times_sqrt2)
    (observe_min_distance_times_sqrt2) (observe_max_distance)
    (classify_max_distance_times_sqrt2) (classify_max_distance)
    (observe_max_distance_times_sqrt2))

(:durative-action move_robot
:parameters()
:control (?dx ?dy - number)
:duration (and (>= ?duration (/ ?dx (maxV)))
    (>= ?duration (/ (- ?dx) (maxV)))
    (>= ?duration (/ ?dy (maxV)))
    (>= ?duration (/ (- ?dy) (maxV)))
    (>= ?duration (+ (/ ?dy (maxV)) (/ ?dx (maxV))))
    (>= ?duration (+ (/ (- ?dy) (maxV)) (/ (- ?dx) (maxV))))
    (>= ?duration (+ (/ (- ?dy) (maxV)) (/ ?dx (maxV))))
    (>= ?duration (+ (/ ?dy (maxV)) (/ (- ?dx) (maxV))))
    (<= ?duration 200))
:condition (and (at start (ready))
    ;; the displacement of robot is inside [-100,100], [-100,100] square
    (at start (>= ?dx -100)) (at start (<= ?dx 100))
    (at start (>= ?dy -100)) (at start (<= ?dy 100))
    (at end (>= ?dx -100)) (at end (>= ?dy -100))
    (at end (<= ?dx 100)) (at end (<= ?dy 100))
    ; the robot must stay within [0, 100] [0, 100] square
    (over all (>= (x_robot) 0)) (over all (<= (x_robot) 100))
    (over all (>= (y_robot) 0)) (over all (<= (y_robot) 100)))
:effect (and (at start (not (ready))) (at end (ready))
    (at start (increase (x_robot) ?dx))
    (at start (increase (y_robot) ?dy))))
```

AI CENTER
FEE CTU

# Risk-Aware MGMP
# + Energy
# + Item Load
# + TW
# + PD

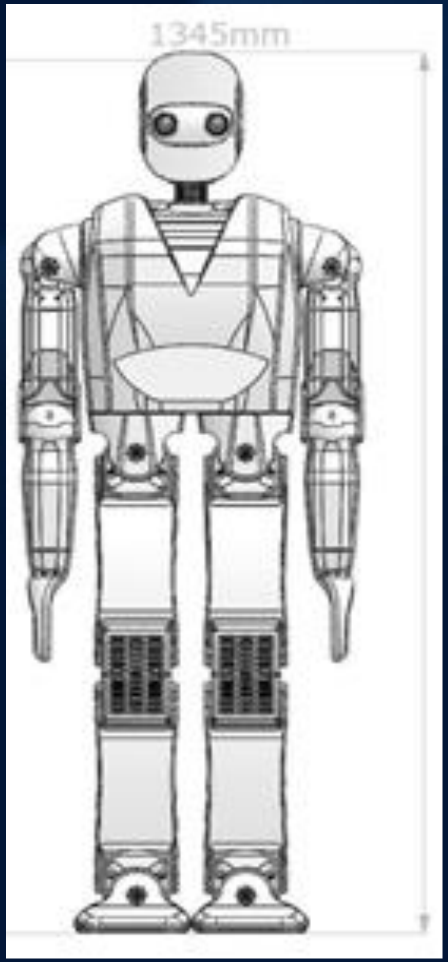




AI CENTER
FEE CTU

# www.aic.fel.cvut.cz




Artificial Intelligence Center

Faculty of Electrical Engineering

Czech Technical University in Prague

AI CENTER
FEE CTU

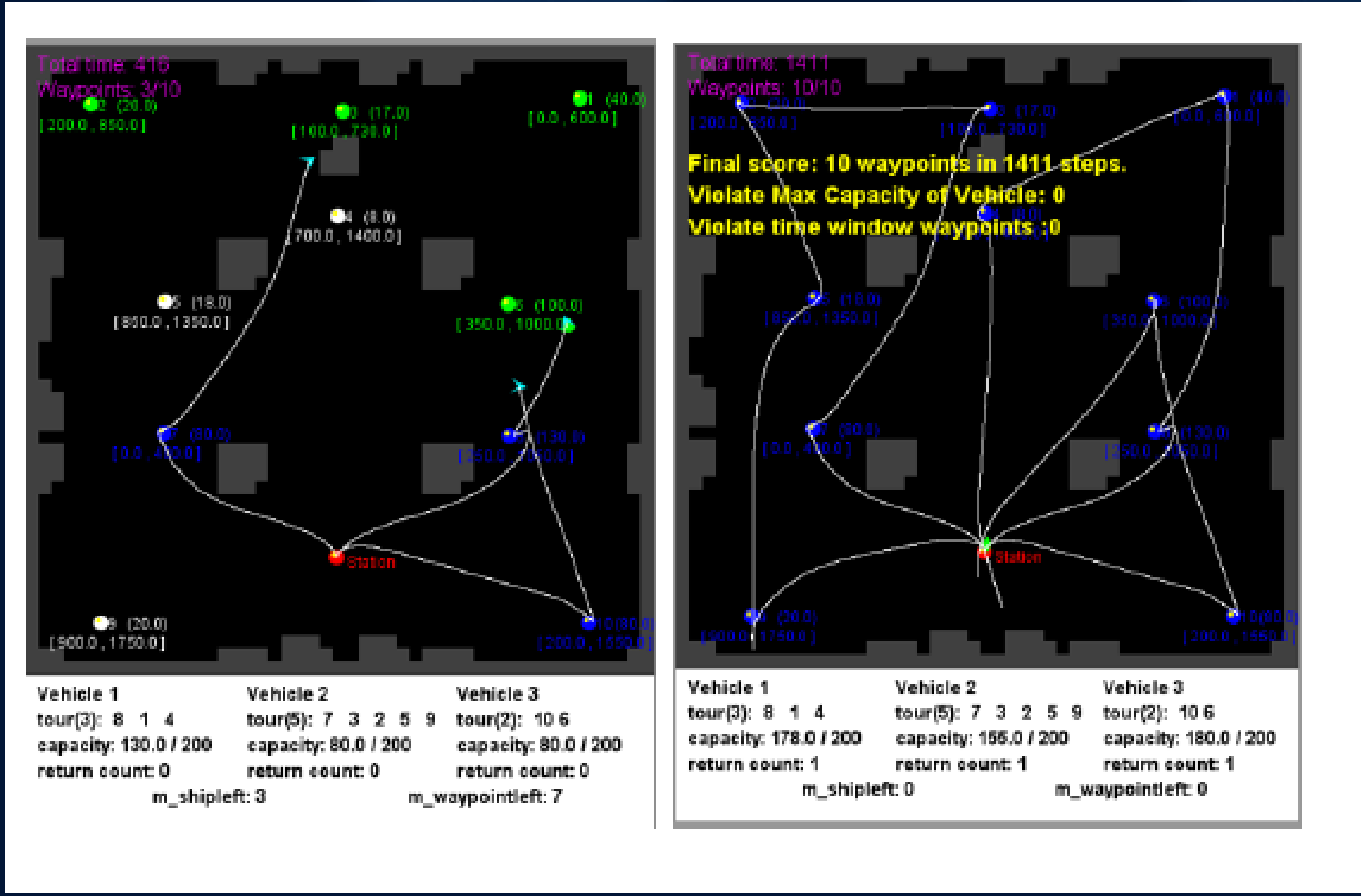
# References

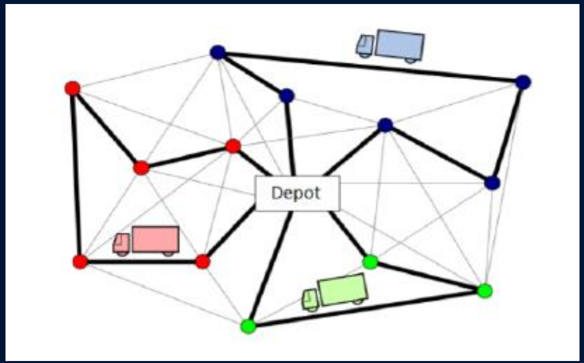

[1] Stefan Edelkamp , Baris Secim, and Erion Plaku . *Surface Inspection via Hitting Sets and Multi-Goal Motion Planning.* Towards Autonomous Robotic Systems (TAROS), 134-149, 2017.
[2] Stefan Edelkamp , Mihai Pomarlan, Erion Plaku . *Multi-Region Inspection by Combining Clustered Traveling Salesman Tours with Sampling-Based Motion Planning* . IEEE RAL. 2(2): 428-435, 2017.
[3] Stefan Edelkamp , Max Gath, Tristan Cazenave, and Fabien Teytaud: Algorithm and knowledge engineering for the TSPTW problem. CISched 2013: 44-51.
[4] Erion Plaku , Sarah Rashidian, and Stefan Edelkamp . *Multi-Group Motion Planning in Virtual Environments.* Computer Animation and Virtual Worlds, 2016.
[5] Sara Rashidian, Erion Plaku , and Stefan Edelkamp . *Motion Planning with Rigid-Body Dynamics for Generalized Traveling Salesman Tours* . Proc. of ACM Conference on Motion in Games, 2014.
[6] Stefan Edelkamp and Christoph Greulich. *Solving Physical Traveling Salesman Problems with policy adaptation* . Proc. of IEEE Conference on Computer in Games (CIG) 2014.
[7] Stefan Edelkamp and Erion Plaku . *Multi-goal motion planning with physics-based game engines* . Proc. of IEEE Conference on Computer in Game (CIG), 2014.
[8] Stefan Edelkamp , Stefan Schroedl. *Heuristic Search, Theory and Practice. Morgan Kaufmann* , 2011.
[9] Stefan Edelkamp , Christoph Greulich, and Denis Golubev . *Solving the Physical Vehicle Routing Problem for Improved Multi-Robot Freespace Navigation.* Proc. of KI, 2016.
[10] Warsame Y, Edelkamp S, and Plaku E (2020): *Energy-Aware Multi-Goal Motion Planning Guided by Monte Carlo Search.* International Conference on Automated Science and Engineering
[11] Edelkamp S, Warsame Y, and Plaku E (2019): *Monte-Carlo Search for Prize-Collecting Robot Motion Planning with Time Windows, Capacities, Pickups, and Deliveries*. KI, vol. 11793, pp. 154-167
[12] Edelkamp S, Lahijanian M, Magazzeni D, and Plaku E (2018): *Integrating Temporal Reasoning and Sampling-Based Motion Planning for Multi-Goal Problems With Dynamics and Time Windows.* IEEE Robotics and Automation Letters, vol. 3, pp. 3473--3480

# Related Work: Physical Vehicle Routing Problem

Multi-Robot Multi-Goal Motion Planning with Time and Resources

Stefan Edelkamp Junho Lee



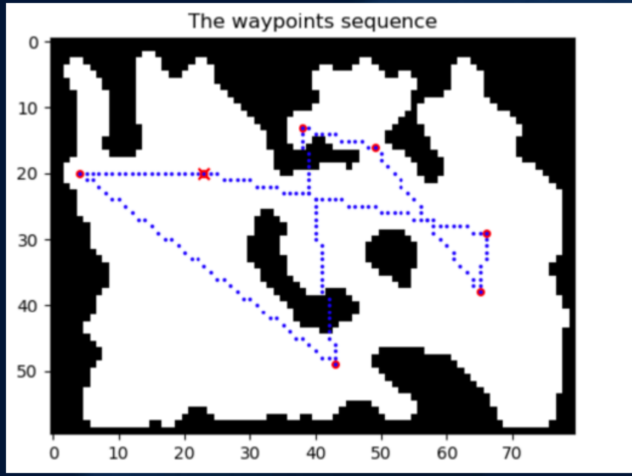

TAROS (1) 2019: 288-299

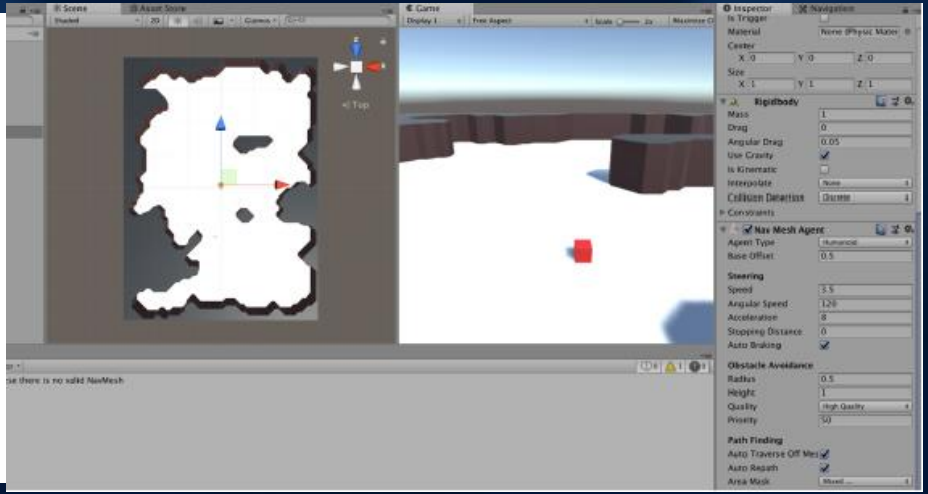
AI CENTER
FEE CTU

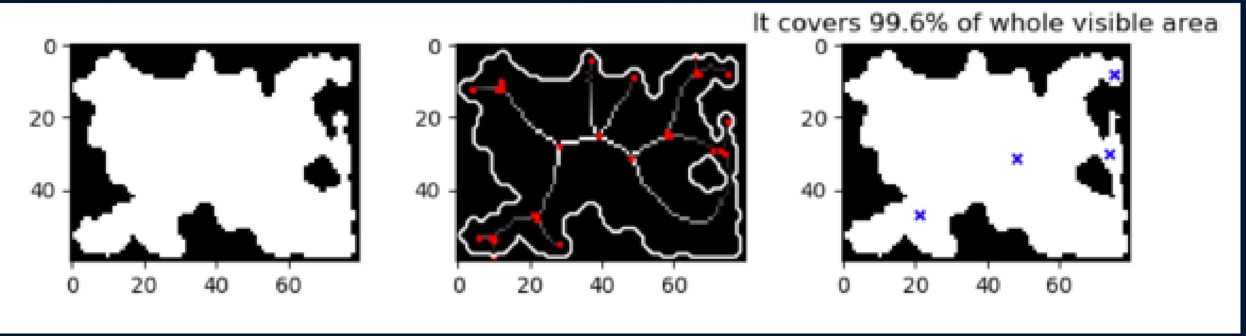
Watchman Routes for Robot Inspection

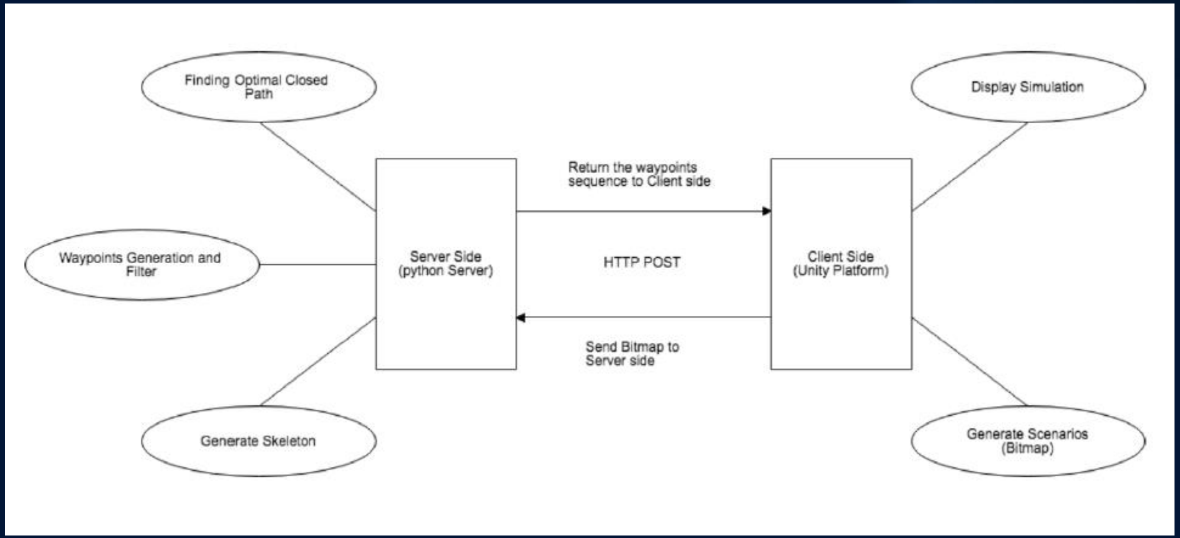
Stefan Edelkamp Zhuowei Yu

# Related Work: Robot Simulation in Unity






TAROS (2) 2019: 179-190

AI CENTER
FEE CTU

# Related Work: Planning Flow Production

## A case study of planning for smart factories

Int. J. Softw. Tools Technol. Transf. 20(5): 515-528 (2018)

Stefan Edelkamp[1] · Christoph Greulich[2]

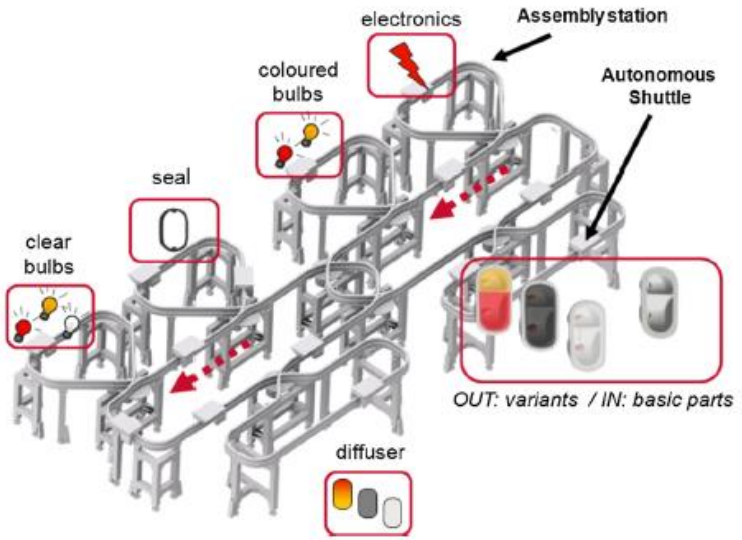



Fig. 1 Assembly scenario for taillights [40]

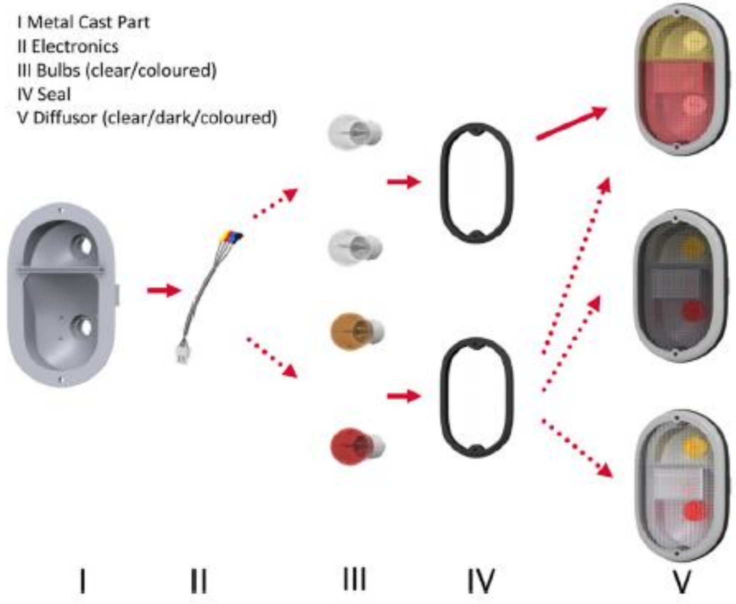
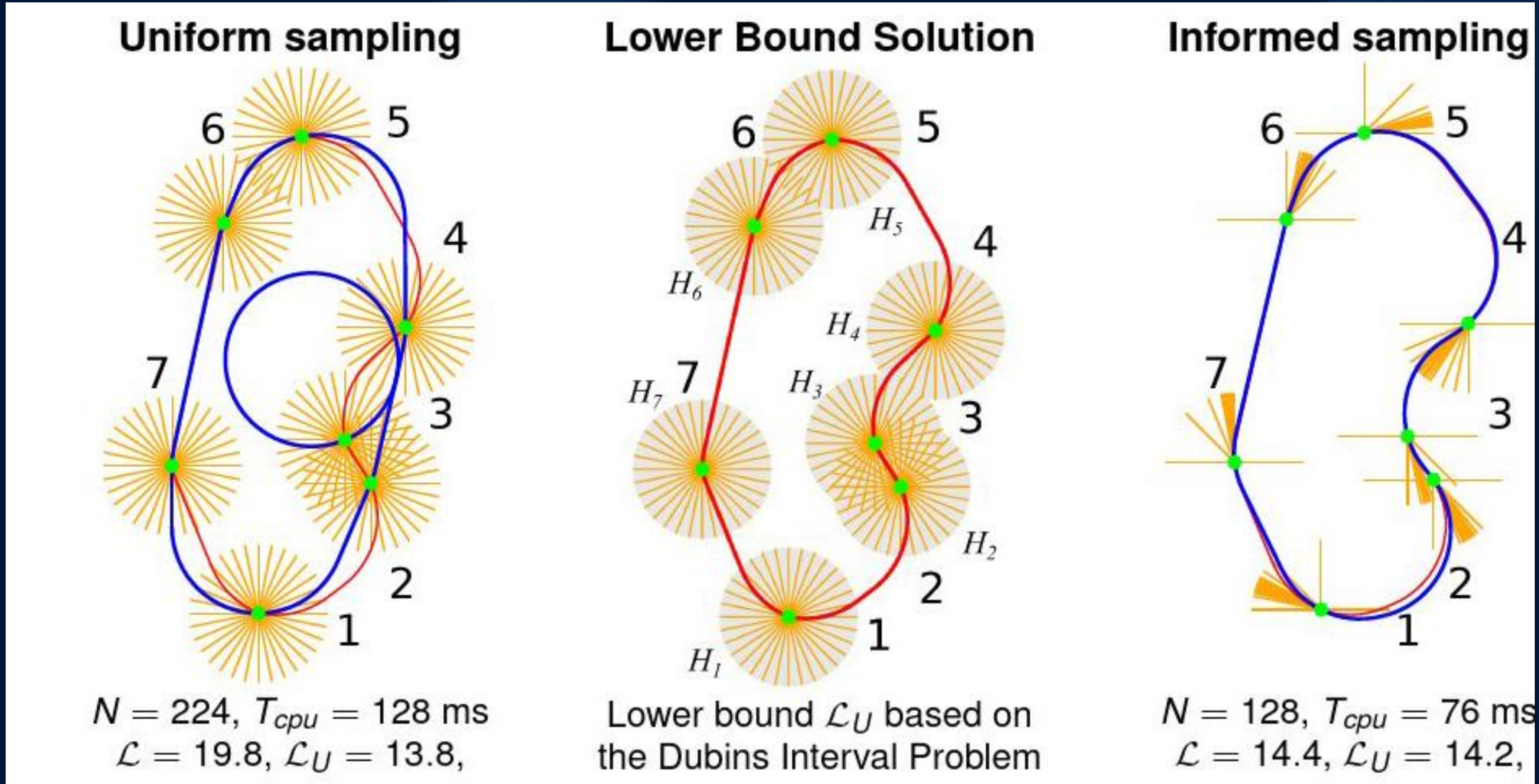



Fig. 2 Assembly states of taillights [21]

AI CENTER
FEE CTU

# Related Work: Dubins Car Touring Problem




- $N$ – the total number of samples (up to 32 samples per waypoint)
- $\mathcal{L}$ is the length of the tour (blue) and $\mathcal{L}_U$ is the lower bound (red) determined as a solution of the **Dubins Interval Problem** (**DIP**)

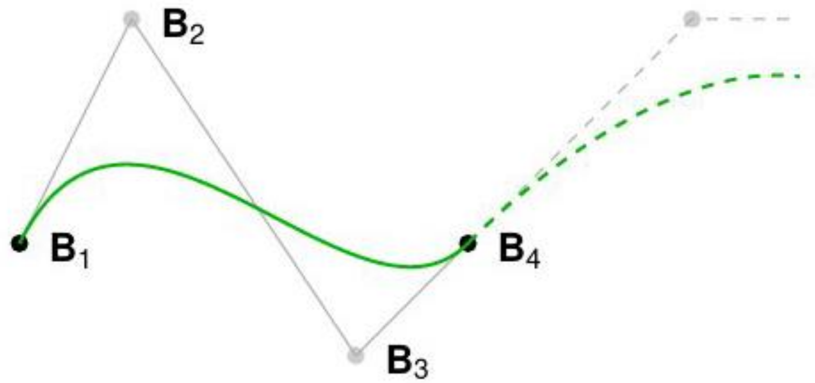
Jan Faigl , Petr Vana , Martin Saska, Tomás Báca, Vojtech Spurný:
On solution of the Dubins touring problem. ECMR 2017: 1-6

AI CENTER
FEE CTU

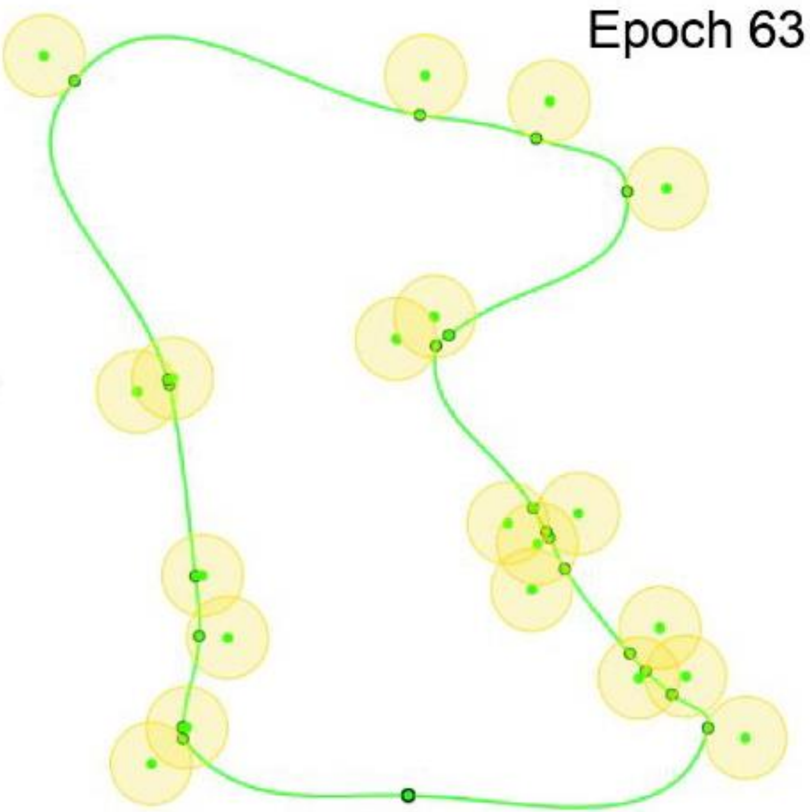
# Related Work: Bezier Curve Touring Problem

## Benefits of Bézier curves

- Flexible and easy to use
- Start/end direction is given by the first/last two control points




Example of a cubic Bézier curve

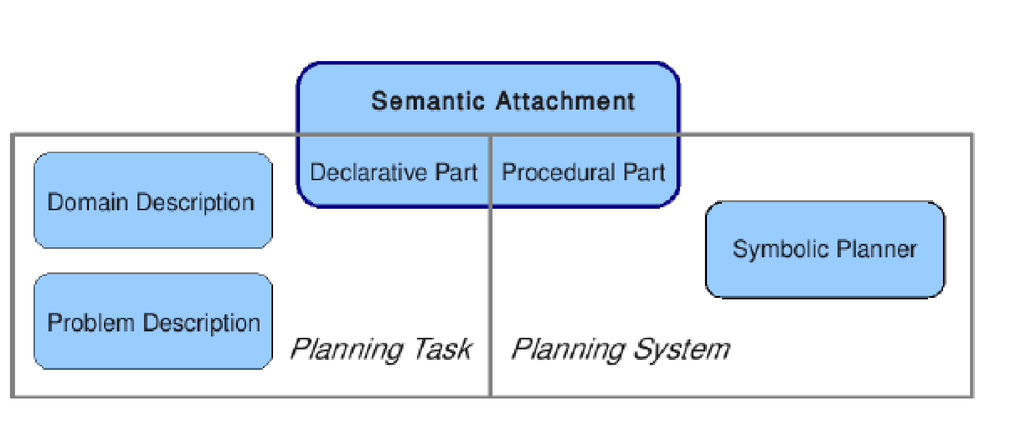


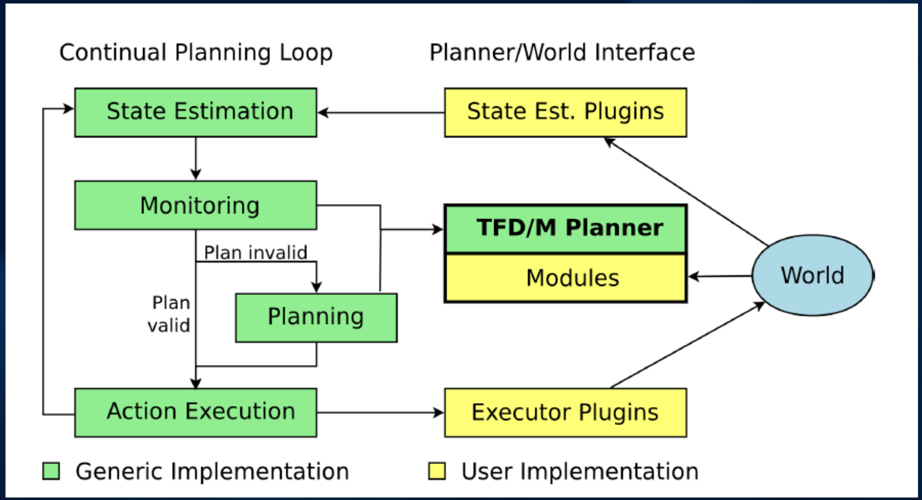

$$\mathbf{X}(\tau) = \mathbf{B}_0(1-\tau)^3 + 3\mathbf{B}_1\tau(1-\tau)^2 + 3\mathbf{B}_2\tau^2(1-\tau) + \mathbf{B}_3\tau^3$$


Jan Faigl, Petr Vána, Robert Penicka :
Multi-Vehicle Close Enough Orienteering Problem with Bézier Curves for Multi-Rotor Aerial Vehicles. ICRA 2019: 3039-3044

AI CENTER
FEE CTU

# Related Work: PDDL/M Planning with Semantic Attachments




Andreas Hertle, Christian Dornhege, Thomas Keller, Bernhard Nebel
Planning with Semantic Attachments:
An Object-Oriented View. ECAI 2012: 402-407

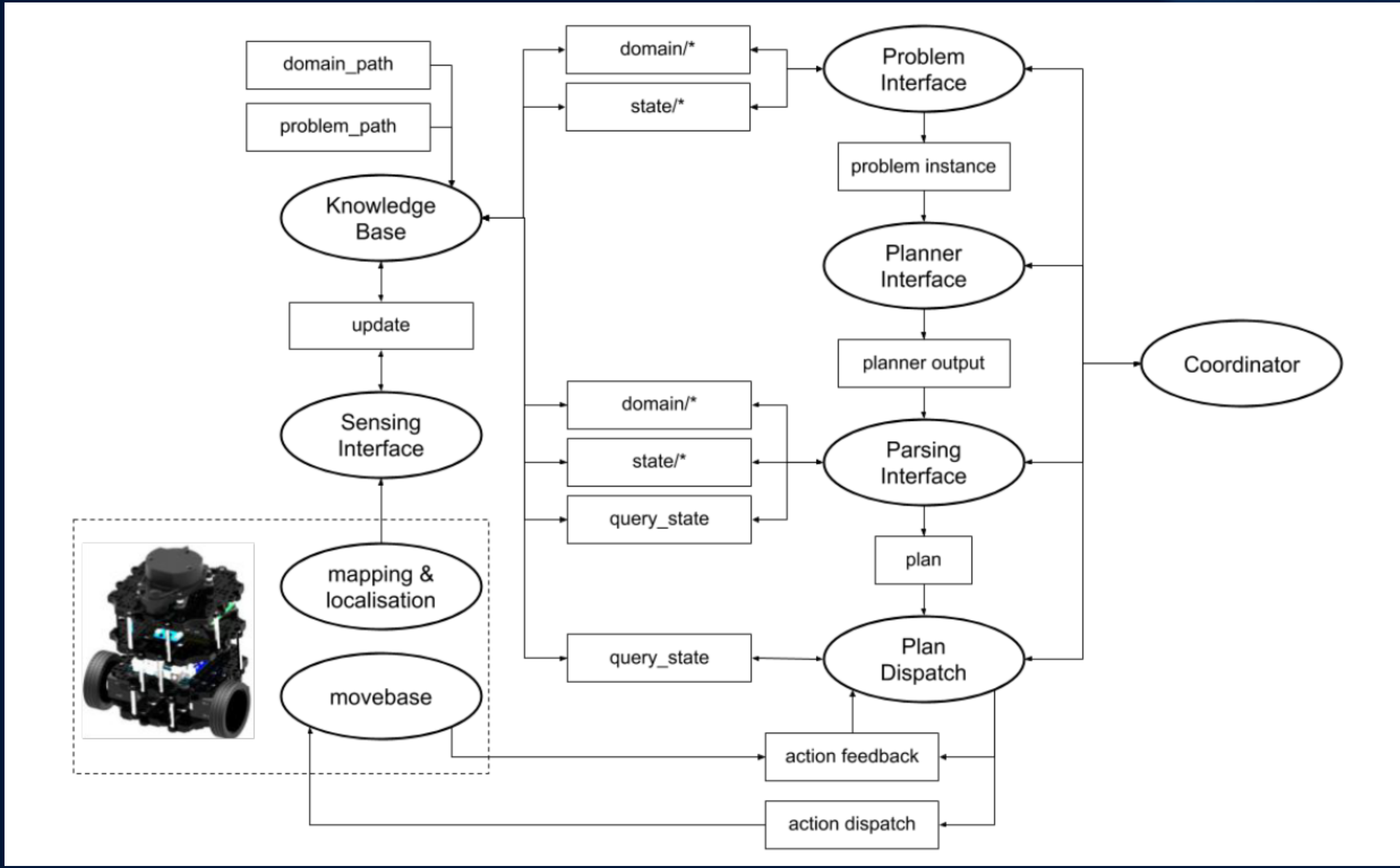
```
(:modules
   (canLoad ?v - vehicle ?p - package
    conditionchecker canLoad@libTrans.so) )
```

```
(:action put-down
 :params (?o - movable ?p - base ?g - grasp)
 :condition (... ([checkPutDown ?o ?p ?g]))
 :effect (and (on ?o ?p) (handempty)
  (not (holding ?o ?g)) ([putDown ?o ?p ?g])
```

```
(:modules
  (putDown ?o - movable ?p - base ?g - grasp
    (q0) (q1) (q2) (q3) (q4) (q5) (q6)
    (x ?o) (y ?o) (z ?o)
    (yaw ?o) (pitch ?o) (roll ?o)
   effect putDown@libTrajectory.so))
```

AI CENTER
FEE CTU

# Related Work: ROS Plan



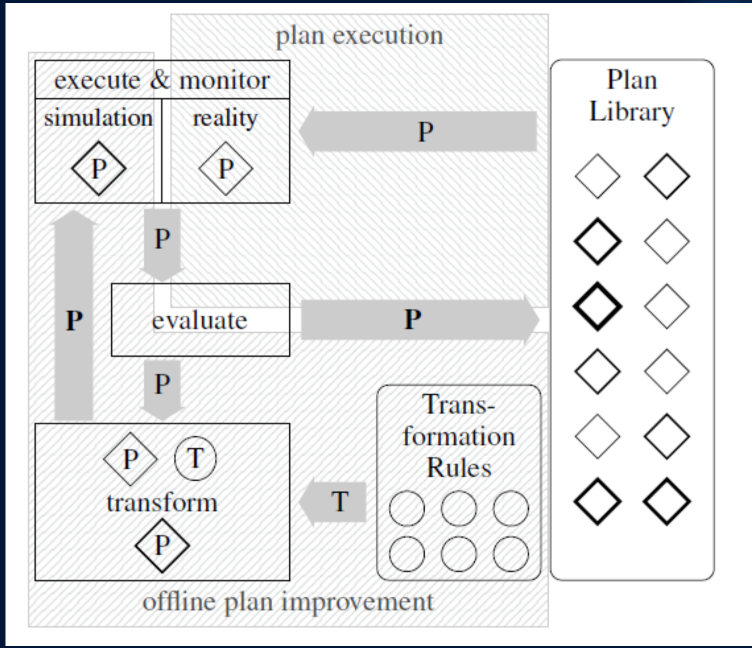


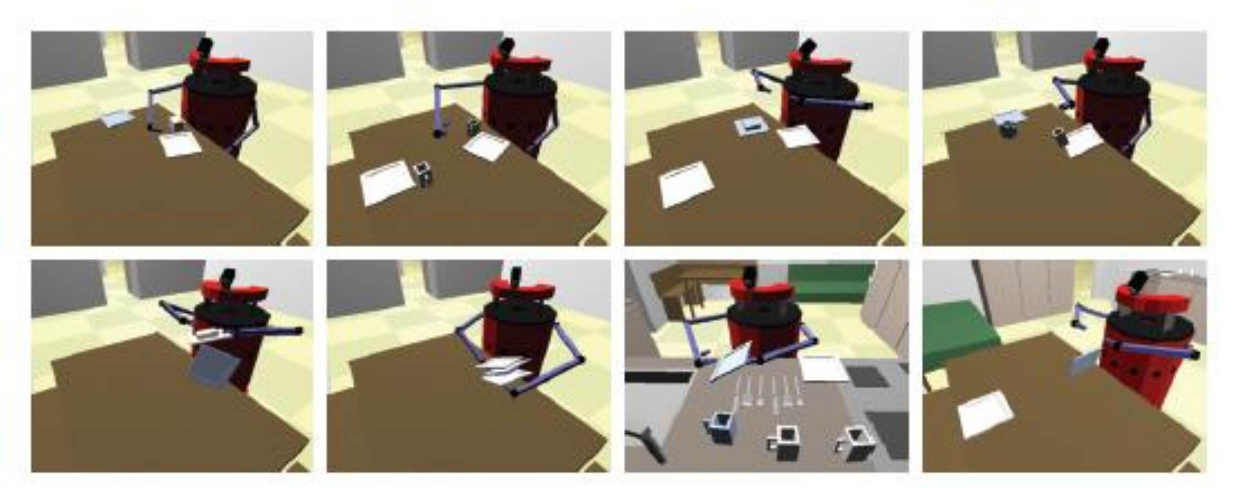
Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcís Palomeras, Natàlia Hurtós, Marc Carreras: ROSPlan: Planning in the Robot Operating System. ICAPS 2015: 333-341

AI CENTER
FEE CTU

# Related Work: Transformational Planning

```
(define-plan (achieve (table-set ?persons))
  ( achieve-for-all
    (lambda (person)
      (with-designators
          ( (table '(the entity (type table)
                                (used-for meals)))
            (seating-location '(the location (at ,table)
                                             (preferred-by ,?person)))
            (plate '(an entity (type plate)
                               (status unused)))
            (cup '(an entity (type cup)
                             (status unused))) )
        (achieve (placed-on
                   plate
                   table
                   '(the location (on ,table)
                                  (matches (entity-location ,plate))
                                  (matches ,seating-location))))
        (achieve (placed-on
                   cup
                   table
                   '(the location (on ,table)
                                  (matches (entity-location ,cup))
                                  (matches ,seating-location))))))
    ?persons))
```



Armin Müller, Alexandra Kirsch, Michael Beetz:

Transformational Planning for Everyday Activity. ICAPS 2007: 248-255

AI CENTER
FEE CTU

# Acknowledgements

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS