Ceng 585 Paper Presentation

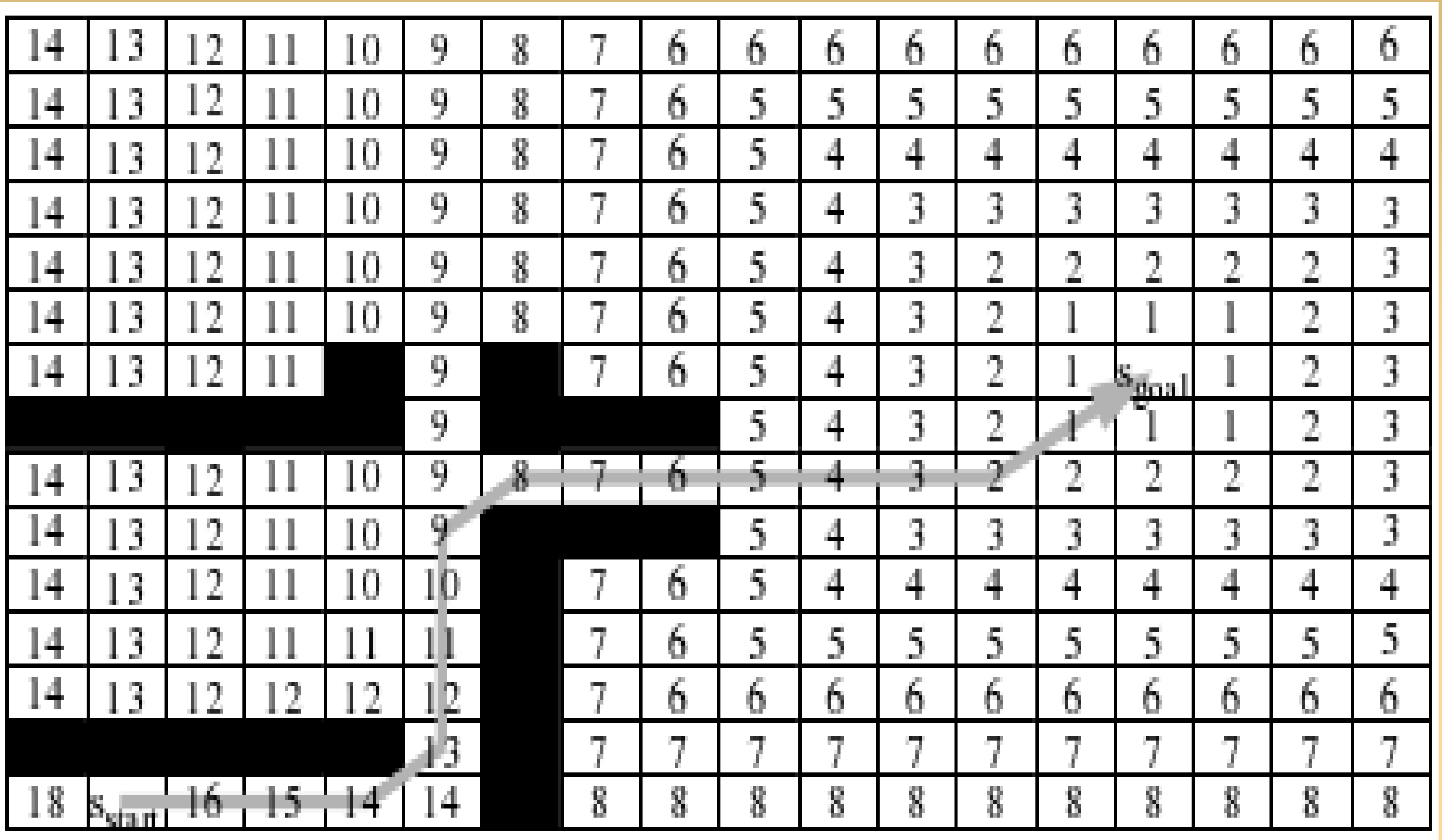
D*-Lite
Path Finding Algorithm and its Variations

Can Eroğul
erogul@ceng.metu.edu.tr

Outline

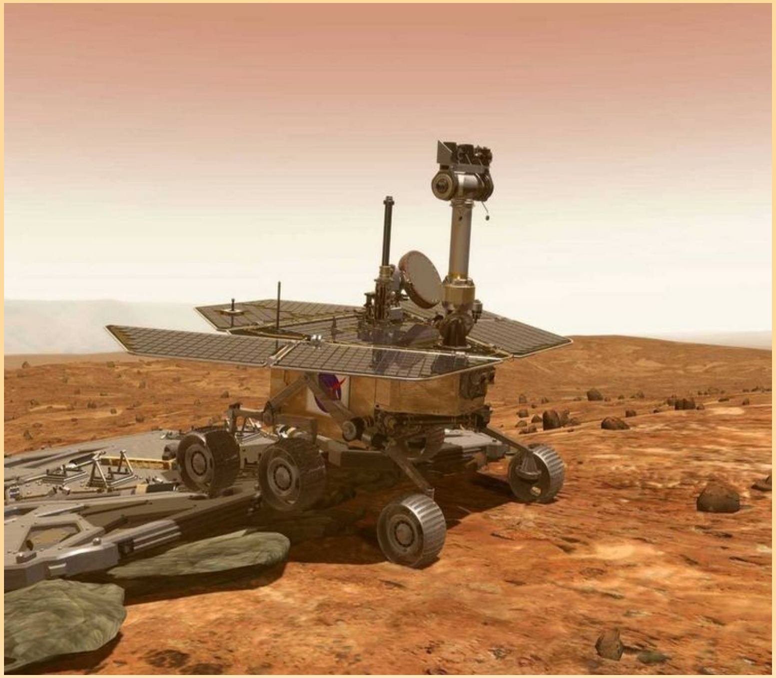
- Motivation
- Environment Properties
- A* (1968), D* (1994)
- D* Lite (2002 & 2005)
- Field D* (2005)
- Multi-resolution Field D* (2006)

Problem Definition: Path Finding



Motivation

- Only A* in lectures
- D* is more useful for robotic domain.
- Used in various robots including Mars rovers "**Spirit**" and "**Opportunity**"




Environment Properties

- Static vs Dynamic
- Complete vs Incomplete (Accesible vs Inaccesible)
- Discrete vs Continuous
- Deterministic vs Non-deterministic
- Stationary Target vs Moving Target

We assume **discrete** & **deterministic** environment with **stationary** target.

Note: All **continuous** domains can be **discretized**.

Why Discretize?

Robotic domain is **continuous**, why discretize? Discretization is a mathematical method.

- Easier calculation: Making data more **suitable** for
  - numerical computation
  - implementation on digital computers

Problems of Grid Based Path Planning

- Path Quality (Limited rotation values ($0, \pi/4, \pi/2$) 4 or 8 neighborhood)
- Memory & computational power requirements

Lack of Smooth Paths

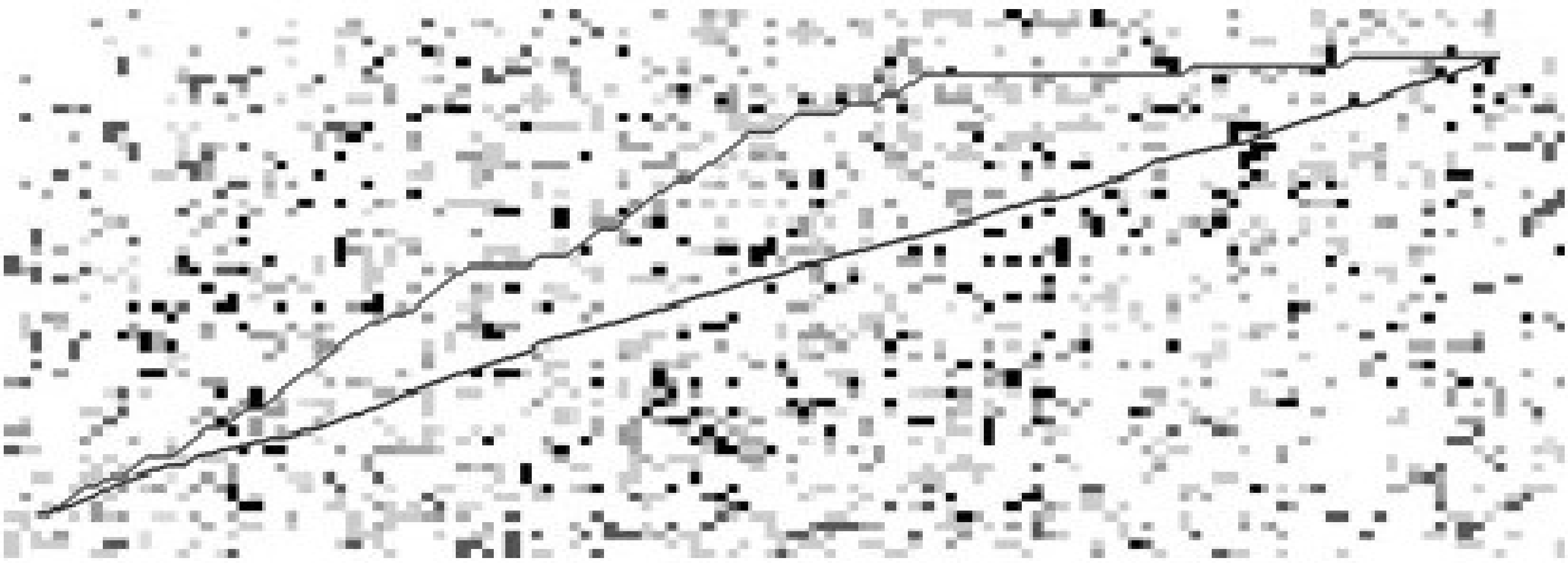


Figure 10. Paths produced by D* Lite (top) and Field D* (bottom) in a 150×60 uniform resolution grid. Again, darker cells have larger traversal costs.

Other Environment Properties

- Stationary Target vs Moving Target (MTS)
- One agent vs Multi-agents
- Fuel constraint? (PHA*)
- Time constraint? (Anytime algorithms)
- Real-Time? (RTA* & LRTA*)
- Shortest path needed?
- Agility or Fatigue?

A* Environment Assumptions

- **Static** world
- **Complete** knowledge of the map
- Freespace Assumption: The robot assumes that the terrain is **clear** unless it knows otherwise.

& also **deterministic** and **discrete** with **stationary** target.

A*

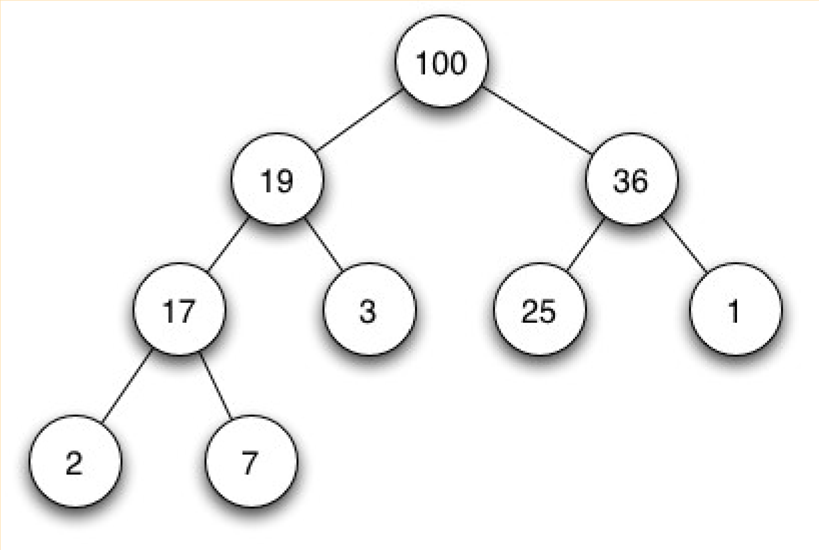
- Covered in 585 lectures.
- Breif reminder: Breadth first search using a **heuristic** function.
- Forward A*: Search starts from the start to goal
- Backward A*: Search starts from the goal to starting point.

A* Details

- f(x) = g(x) + h(x)
- g(x) = Path cost from start to node x
- h(x) = ”Heuristic estimate” of the distance to the goal from node x.
- h(x) should be admisibble (kabul edilebilir). It must **never overestimate** the distance to the goal, so that A* guarantees to find the **shortest path**.
- Implemented with a **priority queue**.

Priority Queue

- **Priority Queue** is an abstract data type
- **Heap** is a data structure.
- Priority Queue can be implemented with heap structure.



Dynamic A*

Capable of planning paths in **unknown**, **partially known** and **changing** environments efficiently.

When the map changes or an unknown obstacle cuts the way, the algorithm replans another path efficiently.

How D* Lite works?

- First run is the same as A*. D* finds the same path with A*.
- When a node changes, D* just recomputes the values of the **inconsistent** nodes, which are necessary to compute, while A* recomputes all of the path.
- D* makes **backward** search. (Starts searching from the goal node)

Inconsistent Nodes

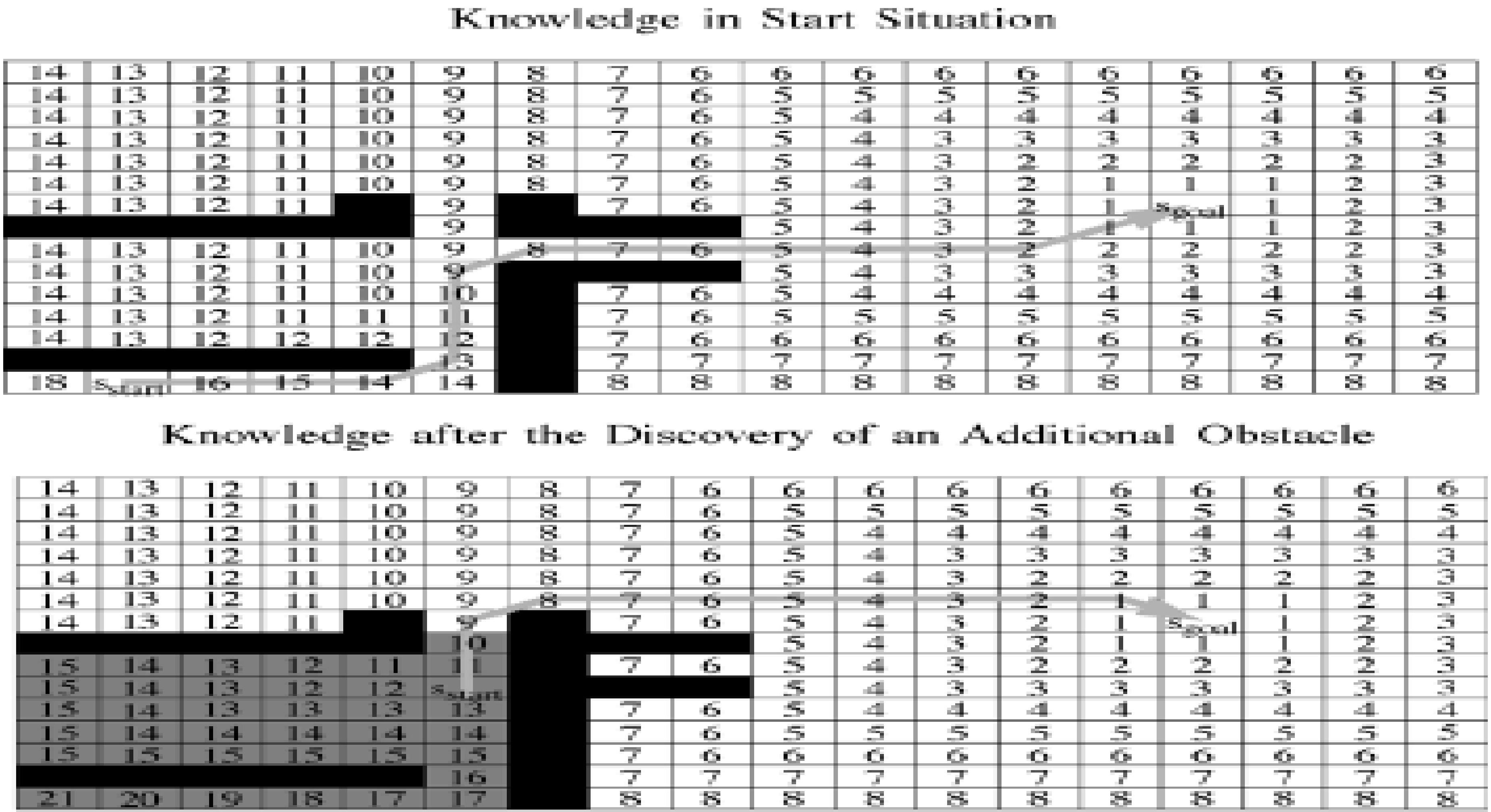


Fig. 2. Simple example (part 1). Gray cells are cells with changed goal distances.

Inconsistent Nodes

- Consistency = (g(x) == rhs(x))
- If a node is inconsistent update all of it's neighbors and itself again. (Updating nodes will try to make them consistent)
- Continue updating while the robots node is inconsistent or there are inconsistent nodes that are closer to the target, which may open a shorter path to the robot.

How D* Lite works?

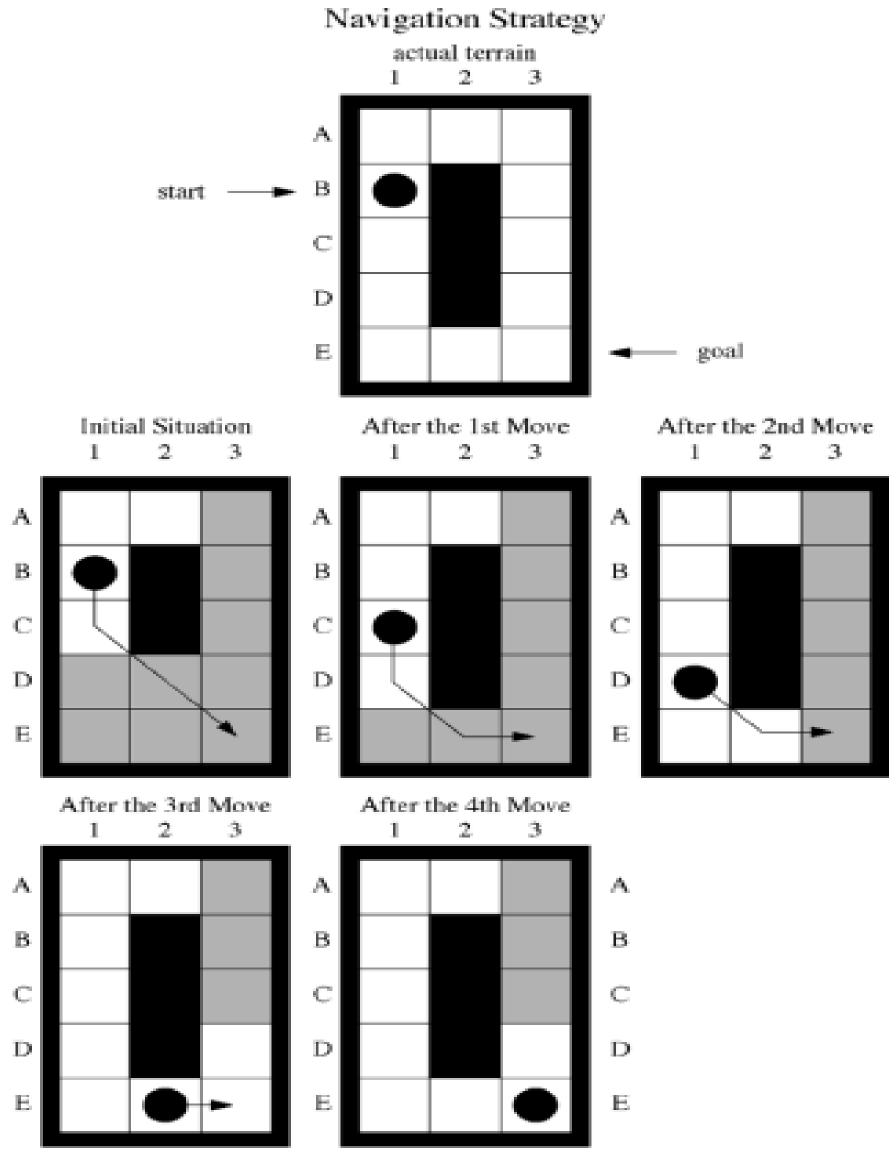


Fig. 1. Illustration of the navigation strategy. Gray cells are cells with unknown blockage status.

Example run of D*

How D* Lite works?

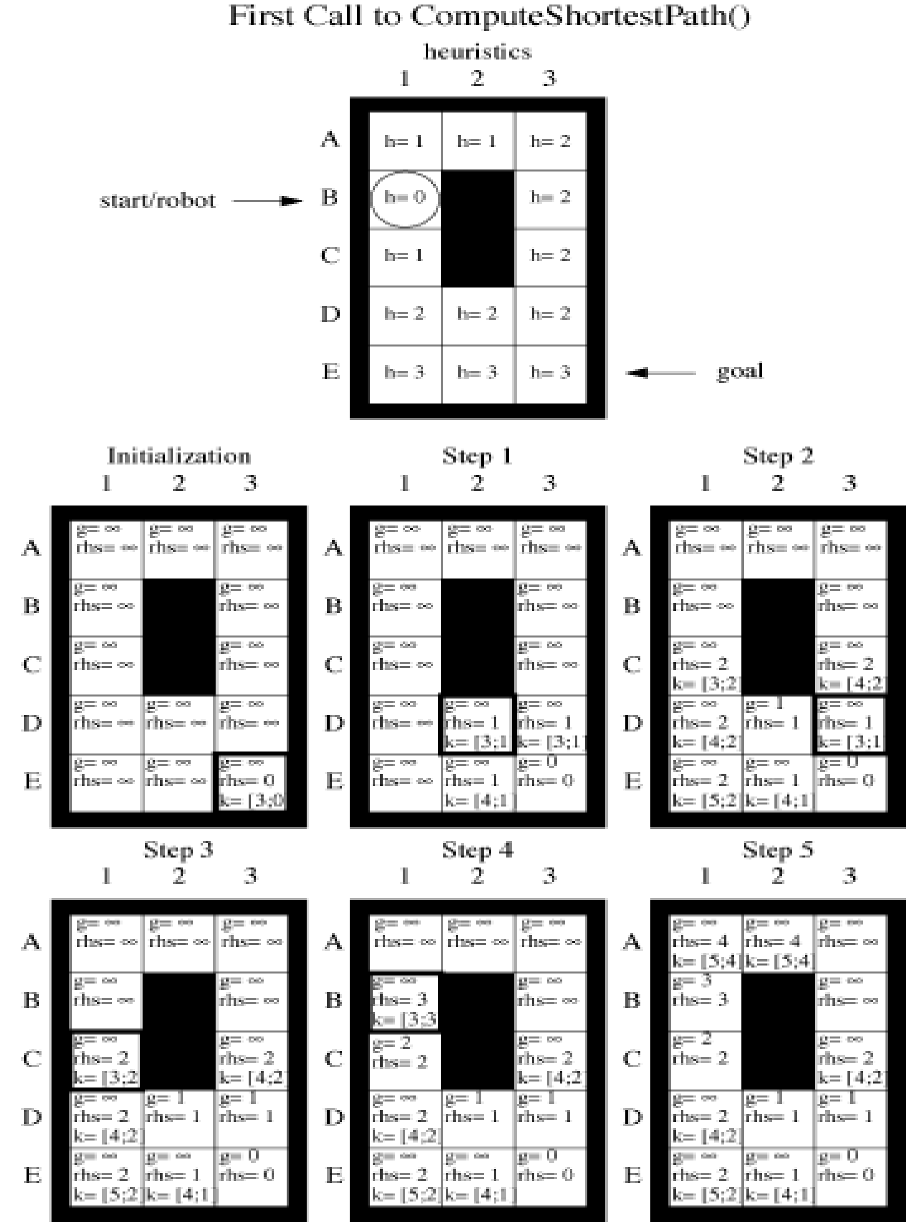


Fig. 6. Operation of the second version of D* Lite (part 1).

First run of D*

How D* Lite works?

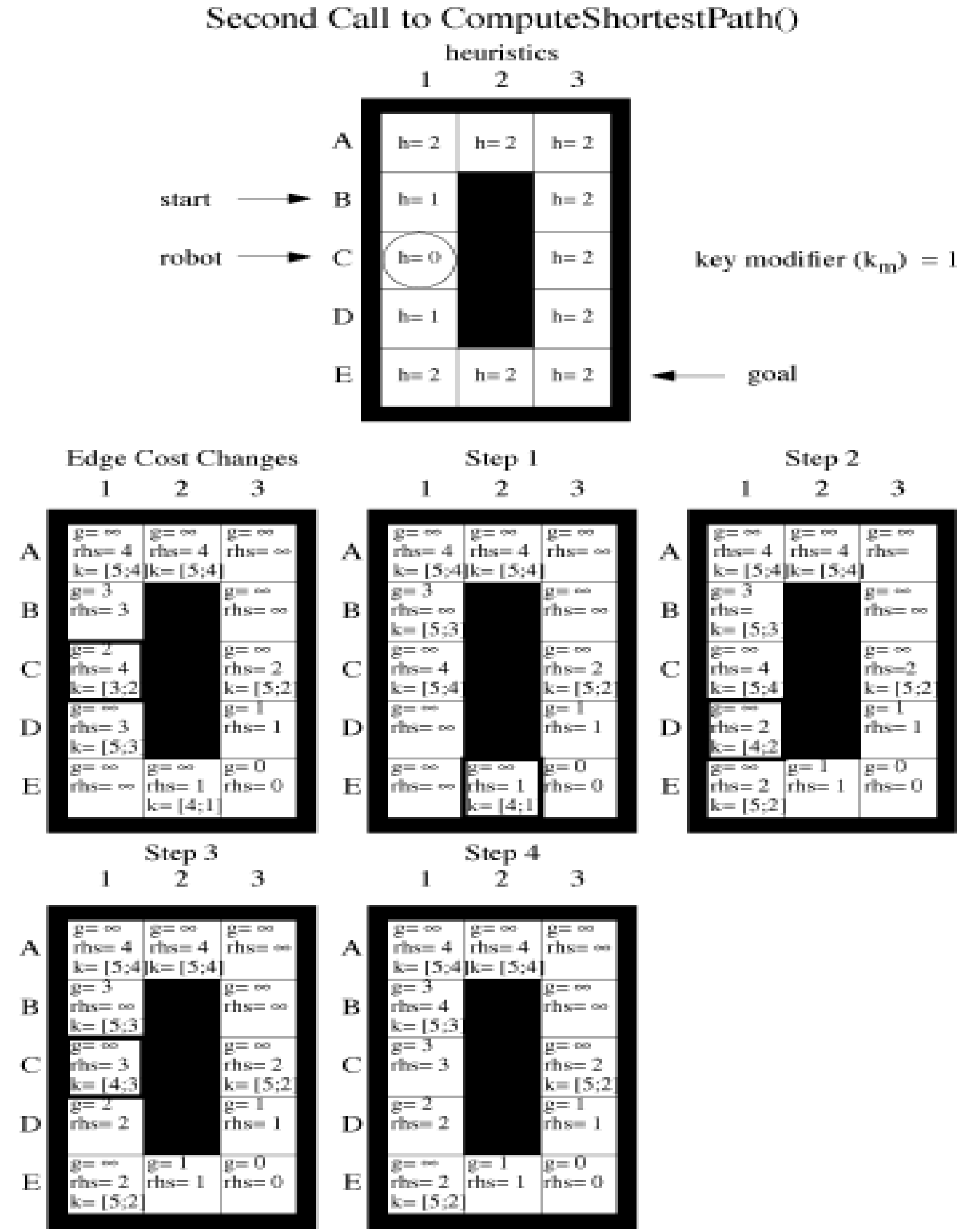


Fig. 7. Operation of the second version of D* Lite (part 2).

Second run of D*

Comparison

TABLE I

EXPERIMENTAL RESULTS—TERRAIN WITH RANDOM OBSTACLES

Search Algorithm	Planning Time	Cell Expansions	Heap Percolates
Breadth-First Search	302.30 msecs	845,433	4,116,516
Backward A*	10.55 msecs	17,096	276,287
Forward A*	7.29 msecs	8,722	177,476
DynamicSWSF-FP	6.41 msecs	13,962	75,738
(Focussed) D*	4.28 msecs	2,138	79,214
D* Lite	2.82 msecs	2,856	32,988

Comparison

TABLE II

EXPERIMENTAL RESULTS—FRACTAL TERRAIN

Search Algorithm	Planning Time	Cell Expansions	Heap Percolates
Breadth-First Search	194.13 msecs	543,408	2,643,916
Backward A*	5.49 msecs	8,680	156,801
Forward A*	4.78 msecs	5,459	124,814
DynamicSWSF-FP	6.26 msecs	13,931	76,703
(Focussed) D*	1.18 msecs	596	19,066
D* Lite	0.97 msecs	393	5,316

A* - D* Comparison

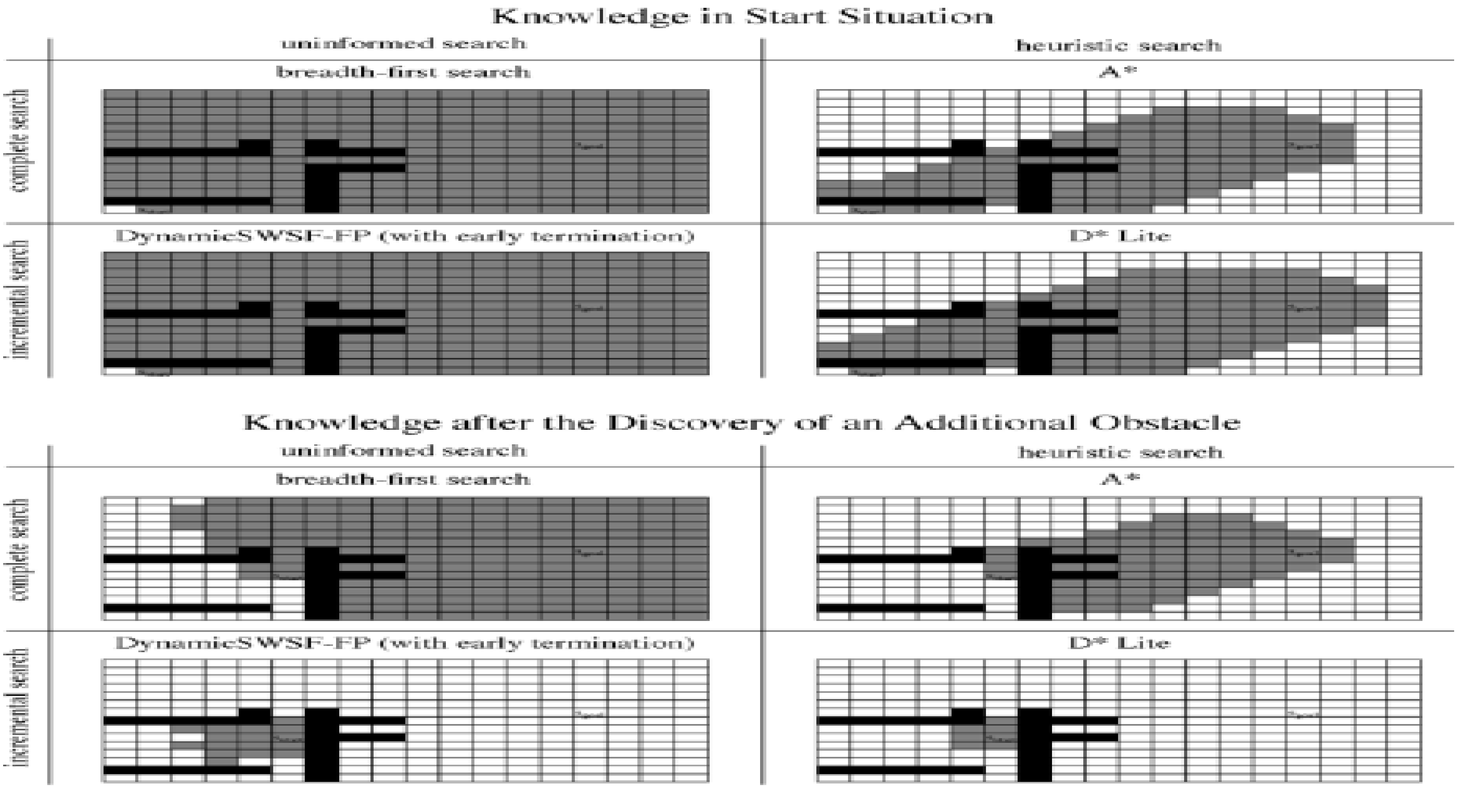


Fig. 8. Simple example (part 2). Gray cells are cells that were expanded.

2 D*-Lite Versions

- D* Lite has 2 versions, which just have different implementations, and D* is also another algorithm. They all find the same path. Second D* Lite algorithm is the fastest.

Problems of Grid Based Path Planning

- Path Quality (Limited rotation values (0,π/4,π/2)) **Solved by Field D***
- Memory & computational power requirements

Field D*

- Operates on continuous domain. After D* computes the path, a post processing function, shortens the path based on interpolation.

Field D*

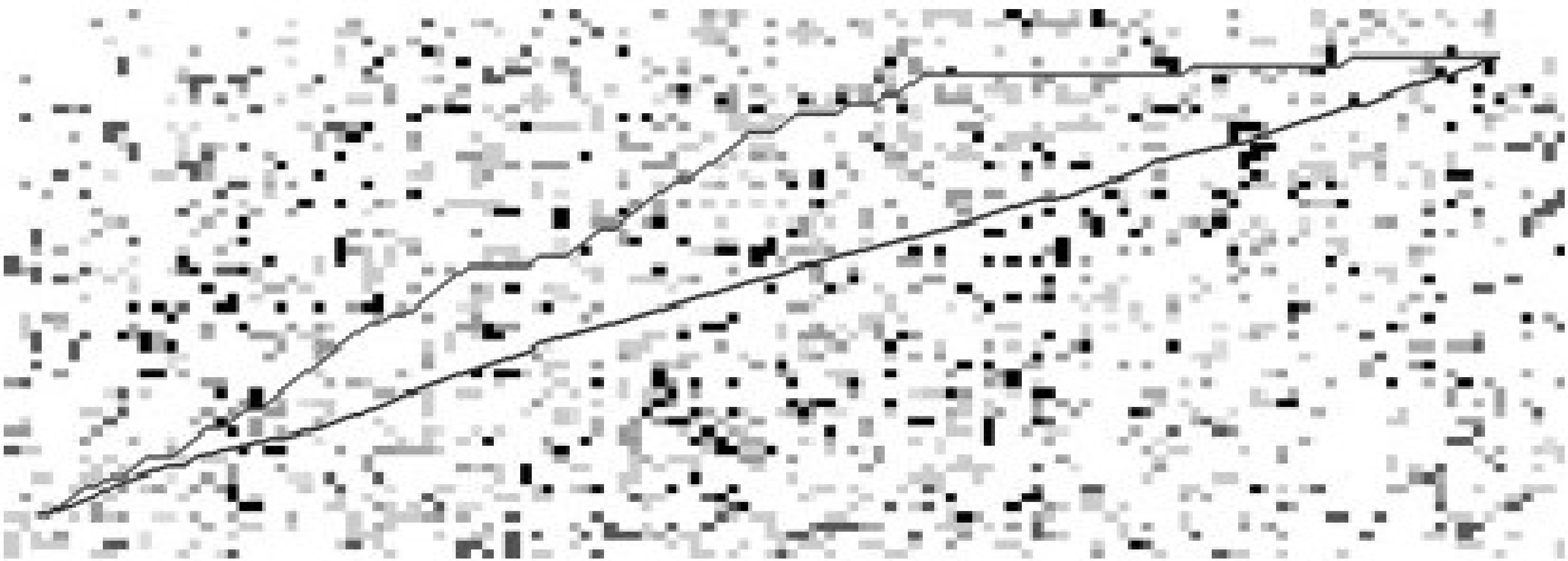


Figure 10. Paths produced by D^* Lite (top) and Field D^* (bottom) in a 150×60 uniform resolution grid. Again, darker cells have larger traversal costs.

Problems of Grid Based Path Planning

- Path Quality (Limited rotation values (0,π/4,π/2)) **Solved by Field D***
- Memory & computational power requirements **Solved by Multi-resolution Field D***

Multi-resolution Field D*

- Computes nearly the same path with Field D* in **shorter time** with **less memory** usage.

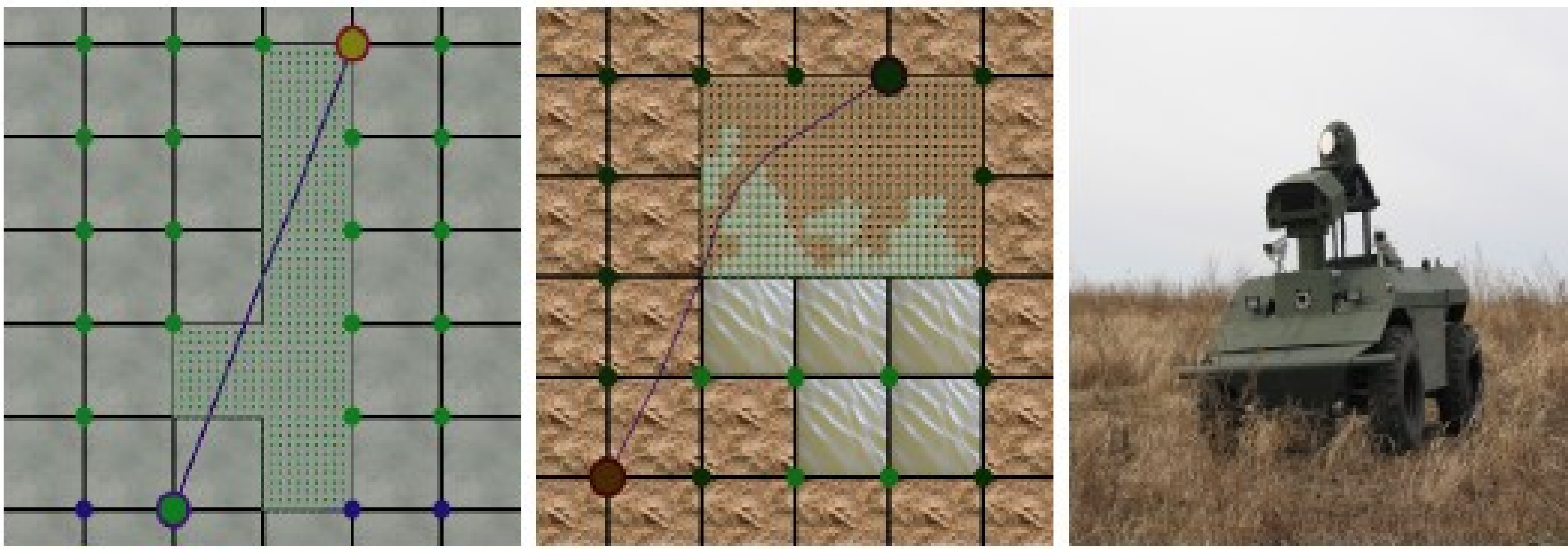
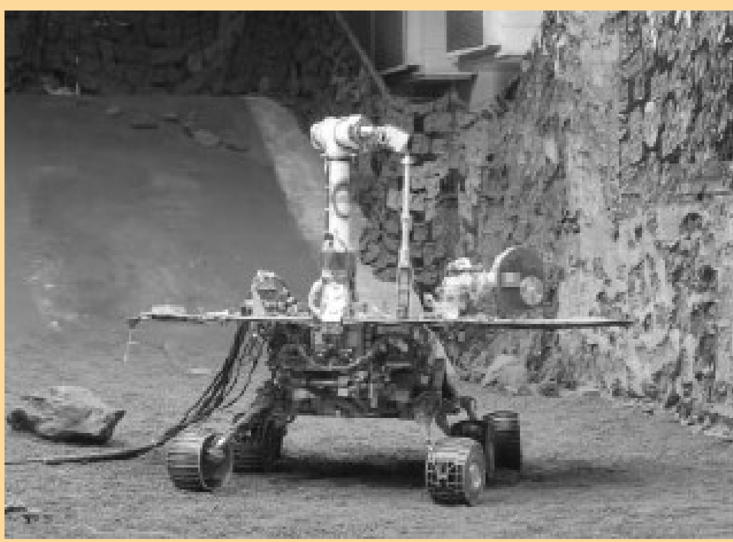


Figure 1. (*left, center*) Multi-resolution Field D* produces direct, low-cost paths (in blue/dark gray) through both high-resolution and low-resolution areas. (*right*) The GDRS XUV robot used for autonomous navigation of outdoor terrain. Multi-resolution Field D* was initially developed in order to extend the effective range of rugged outdoor vehicles such as this by one to two orders of magnitude.

D* on MARS!

- Joseph Carsten and Art Rankin from NASA's Jet Propulsion Laboratory installed a version of **Field D*** using elements of **D*-Lite** on the Mars rovers "**Spirit**" and "**Opportunity**" and first let it control a rover on **Mars** in February 2007 after testing it on a rover on **Mars** in November 2006.



Related Papers

- Optimal and Efficient Path Planning for Partially-Known Environments, Anthony Stenz, ICRA, 94
- Fast Replanning for Navigation in Unknown Terrain 2002 & 2005, Sven Koenig & Maxim Likhachev
- Multi-resolution Field D* IAS 2006 Dave Ferguson, Anthony Stenz

Questions?

Thanks for listening