CS354R

DR SARAH ABRAHAM

DYNAMIC PATH PLANNING

DYNAMIC PATH PLANNING

- When can the environment change after planning?
  - The player does something
  - Other agents get in the way
- Solution strategies are highly dependent on the nature of the game

DISCUSS

- How can we handle dynamic changes to the game environment?

AVOIDING PLAN CHANGES

- Partial planning: Only plan short segments of path
  - Stop A* after a path of some length is found, even if the goal is not reached
    - Use current best estimated path
    - Extreme case: greedy search
    - Common case: hierarchical planning, considering low level when needed
- A short path is less likely to change than a long path
  - Optimality will be sacrificed
  - More even frame times
- Other strategies:
  - Wait for the blockage to pass
  - Lock the path to other agents (implies priorities)

RE-PLANNING

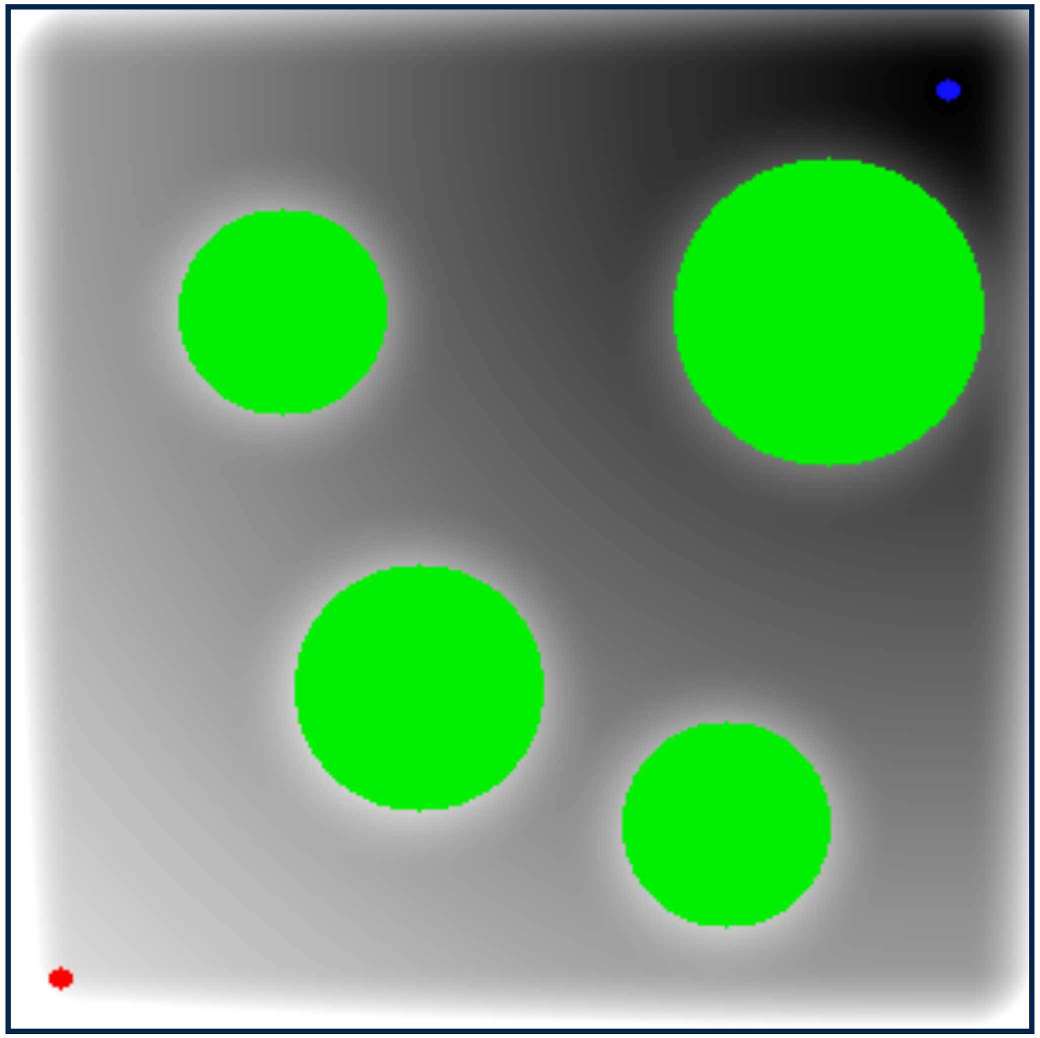
- If you discover the plan has gone wrong, create a new one
- New plan assumes the dynamic changes are permanent
- Used in conjunction with avoidance strategies
    - Re-planning is expensive so avoid doing it
    - Avoid generating a plan that will be re-done (partial planning in conjunction with re-planning)

REACTIVE PLANNING

- Reactive agent plans only its next step using immediately available information
- Potential field planning example:
  - Set up a force field around obstacles (and other agents)
  - Set up a gradient field toward the goal
  - The agent follows the gradient downhill to the goal, while the force field pushes it away from obstacles
  - Can also model velocity and momentum (field applies a force)
- Potential field planning is reactive because the agent just looks at the local gradient at any instant
- Has been used in real robots for navigating things like hallways

POTENTIAL FIELD

- Red is start point, blue is goal
- This used a quadratic field strength around the obstacles
- Note that the boundaries of the world contribute to the field



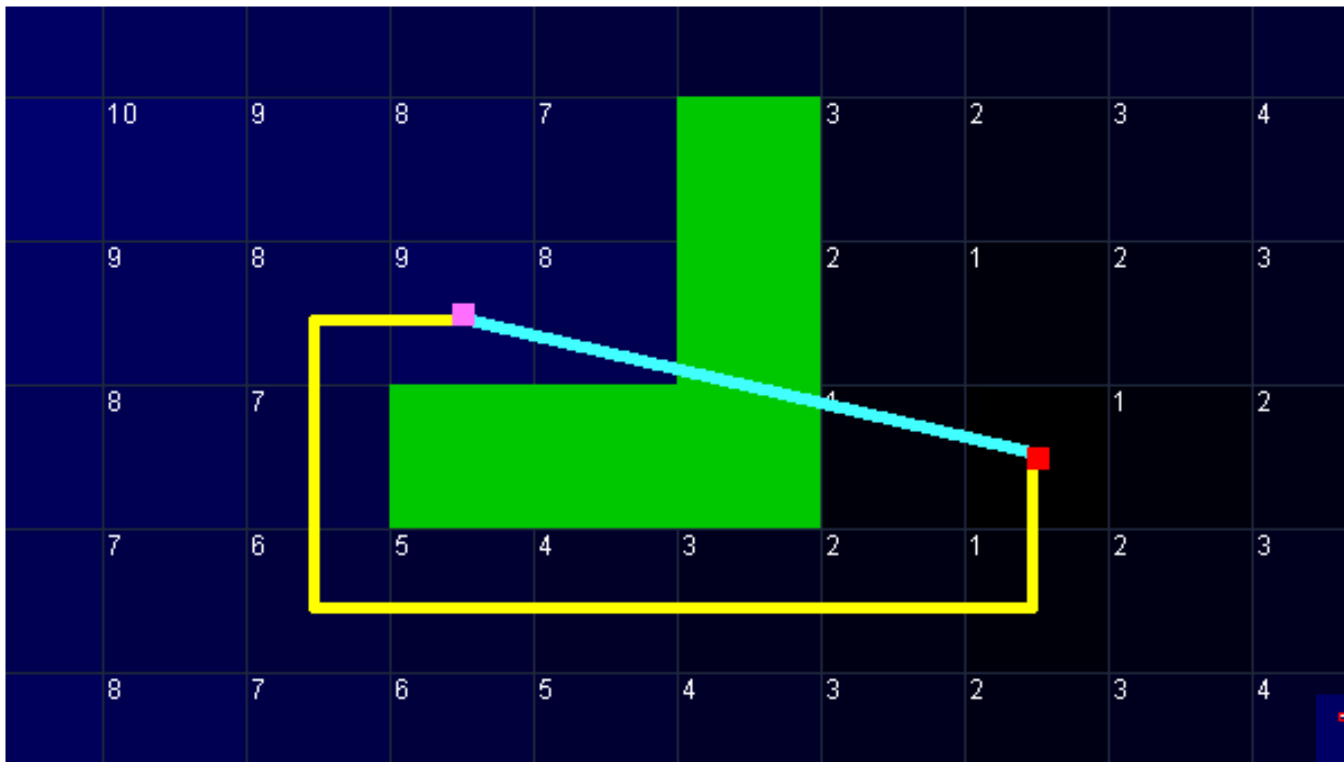
CREATING THE FIELD

- The constant gradient (cost) can be a simple linear gradient based on distance from the goal, d_{goal}: $f_{goal} = k\ d_{goal}$
- The obstacles contribute a field strength based on the distance from their boundary, $f_i(d_i)$
  - Linear, quadratic, exponential, or something else
  - Truncate so that field at some distance is zero
  - Strength determines how likely the agent is to avoid it
- Add all the sub-fields together to get overall field

FOLLOWING THE FIELD

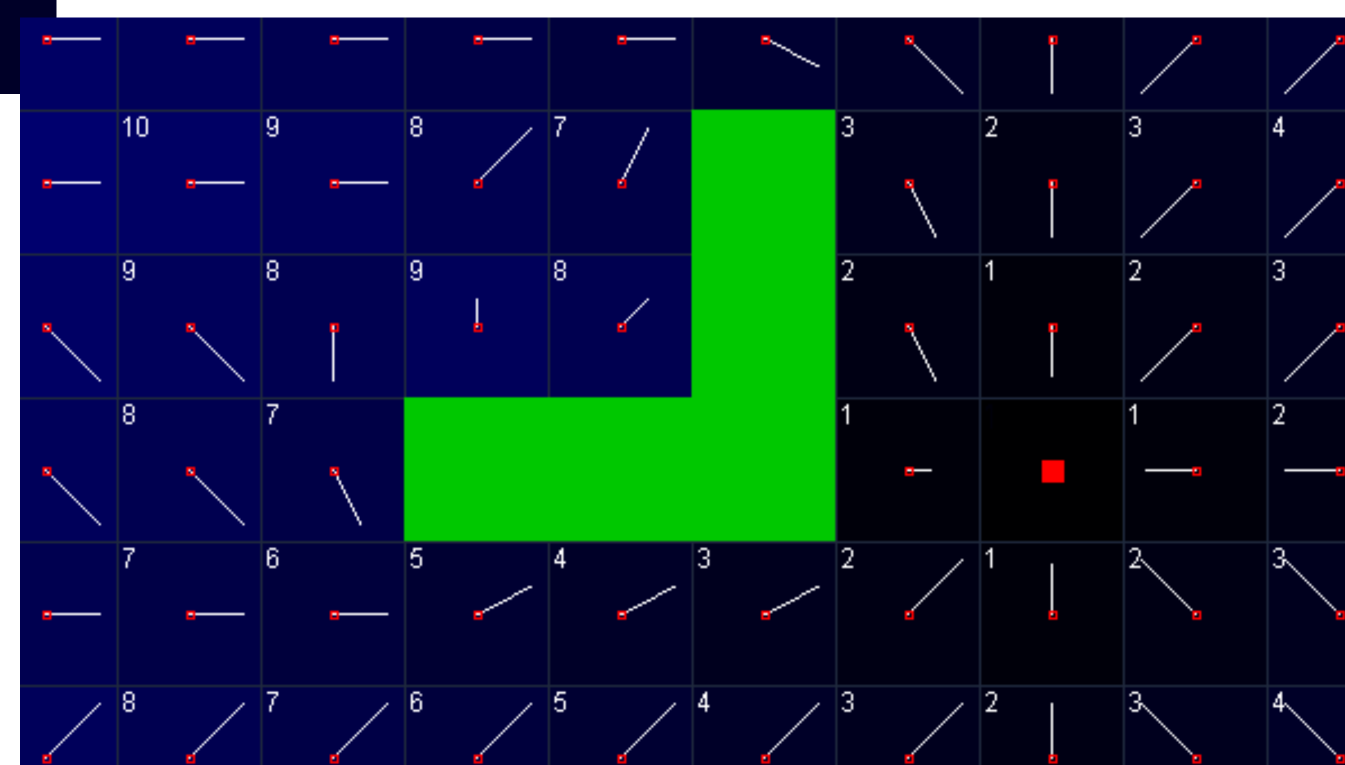
- At each step, the agent needs to know which direction is "downhill"
- From the cost field, compute a vector field indicating direction of flow
  - Compute the gradients of each component and add
  - Need partial derivatives in x and y (for 2D planning)
- Best approach is to consider the gradient as an acceleration
  - Avoids sharp turns and provides smooth motion
  - Higher mass makes large objects turn more slowly
  - Easy to make frame-rate independent
  - High velocities can cause collisions
  - The field is a guide, rather than a true force

COST AND VECTOR FIELDS

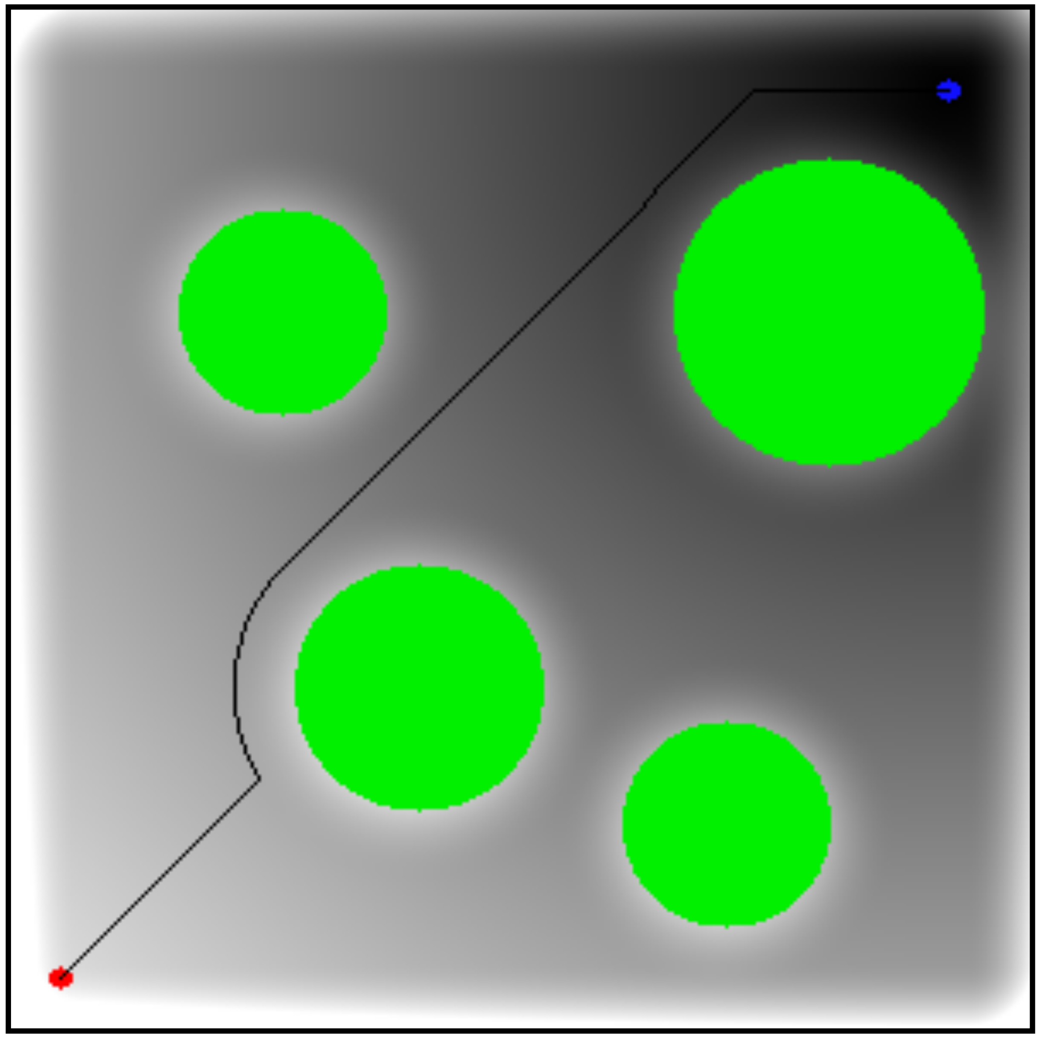


Cost field

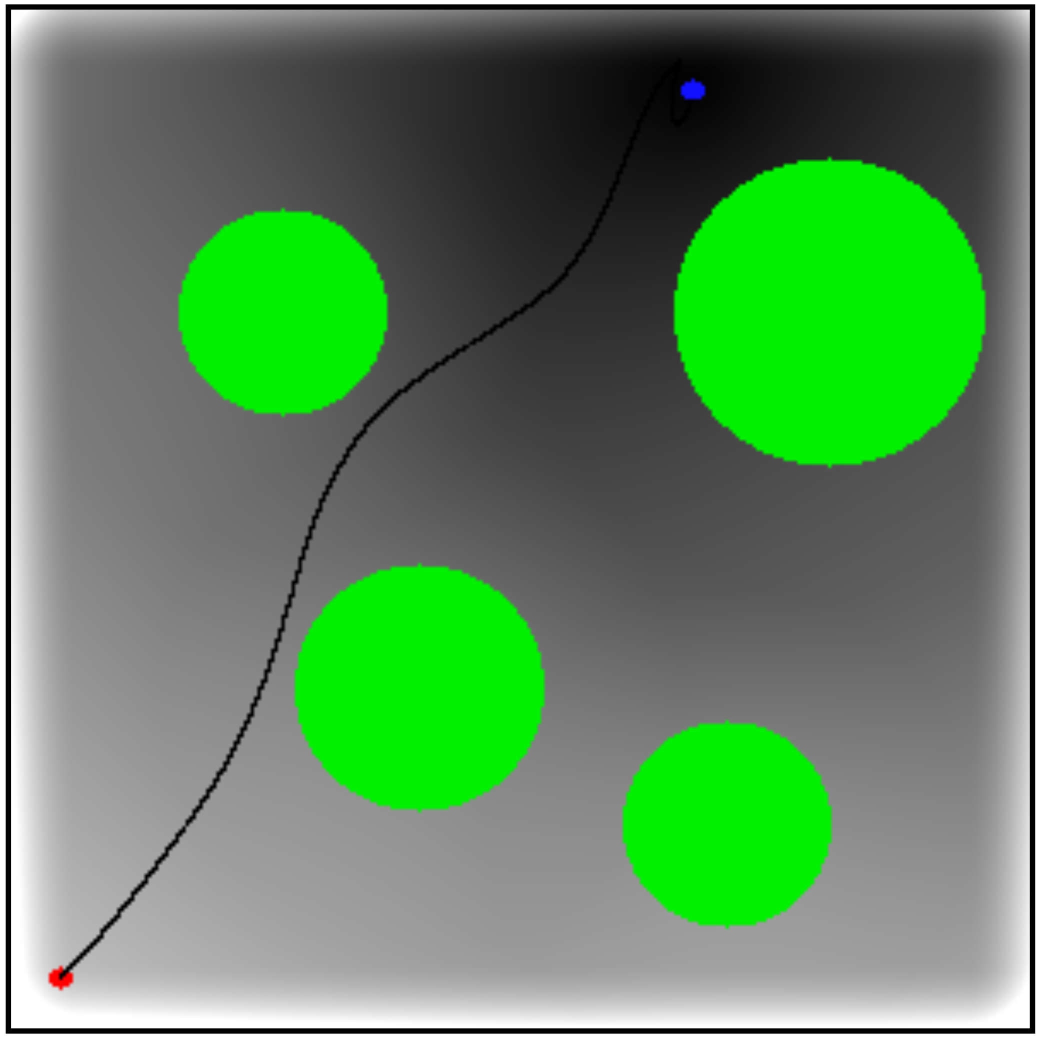
Vector field



FOLLOWING EXAMPLES



No momentum - choose to go to neighbor with lowest field strength



Momentum - but with linear obstacle field strength and moved goal

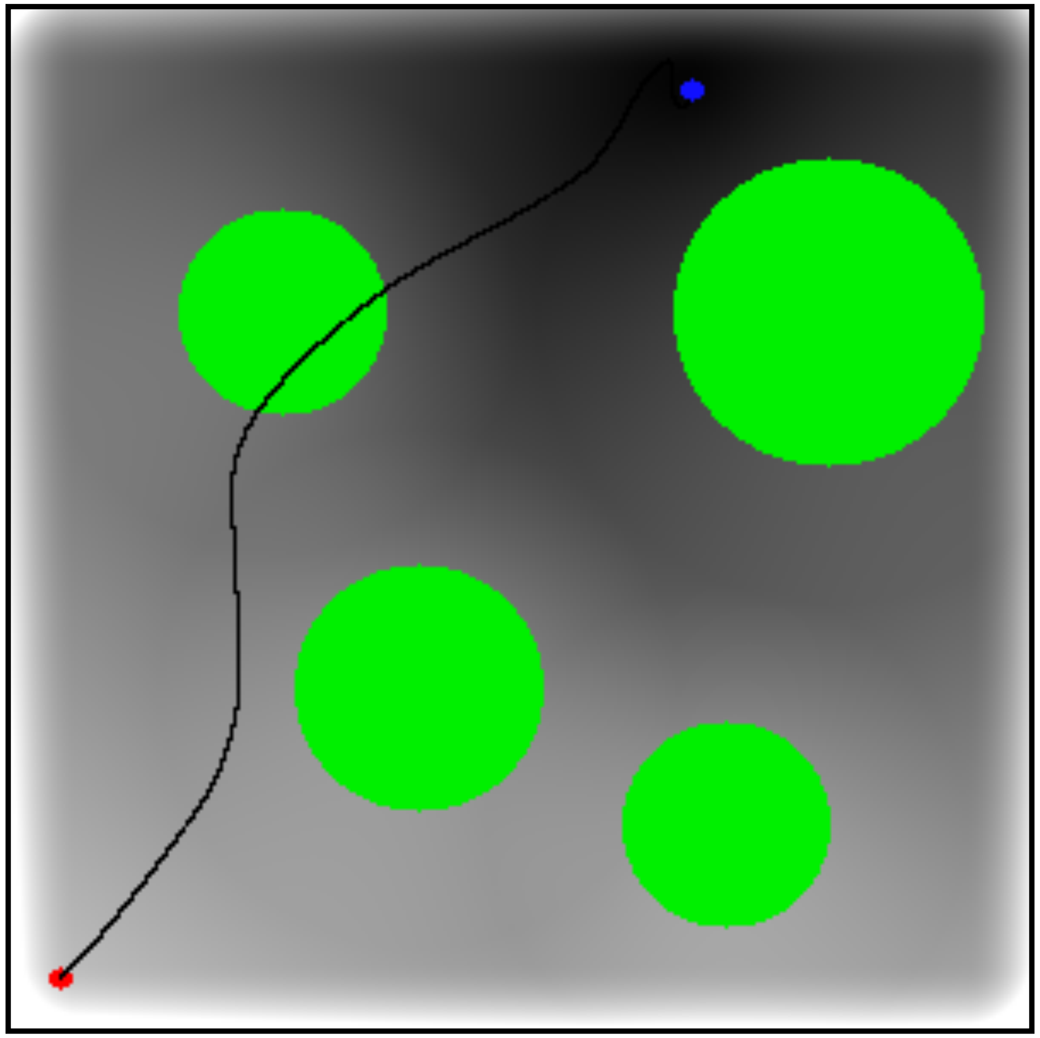
DISCRETE APPROXIMATION

- Compute the field on a grid
  - Allows pre-computation of fields that do not change, such as fixed obstacles
  - Moving obstacles handled as before
- Use discrete gradients
  - Look at neighboring cells
  - Go to neighboring cell with lowest field value
- Advantages: Faster
- Disadvantages: Space cost and approximate

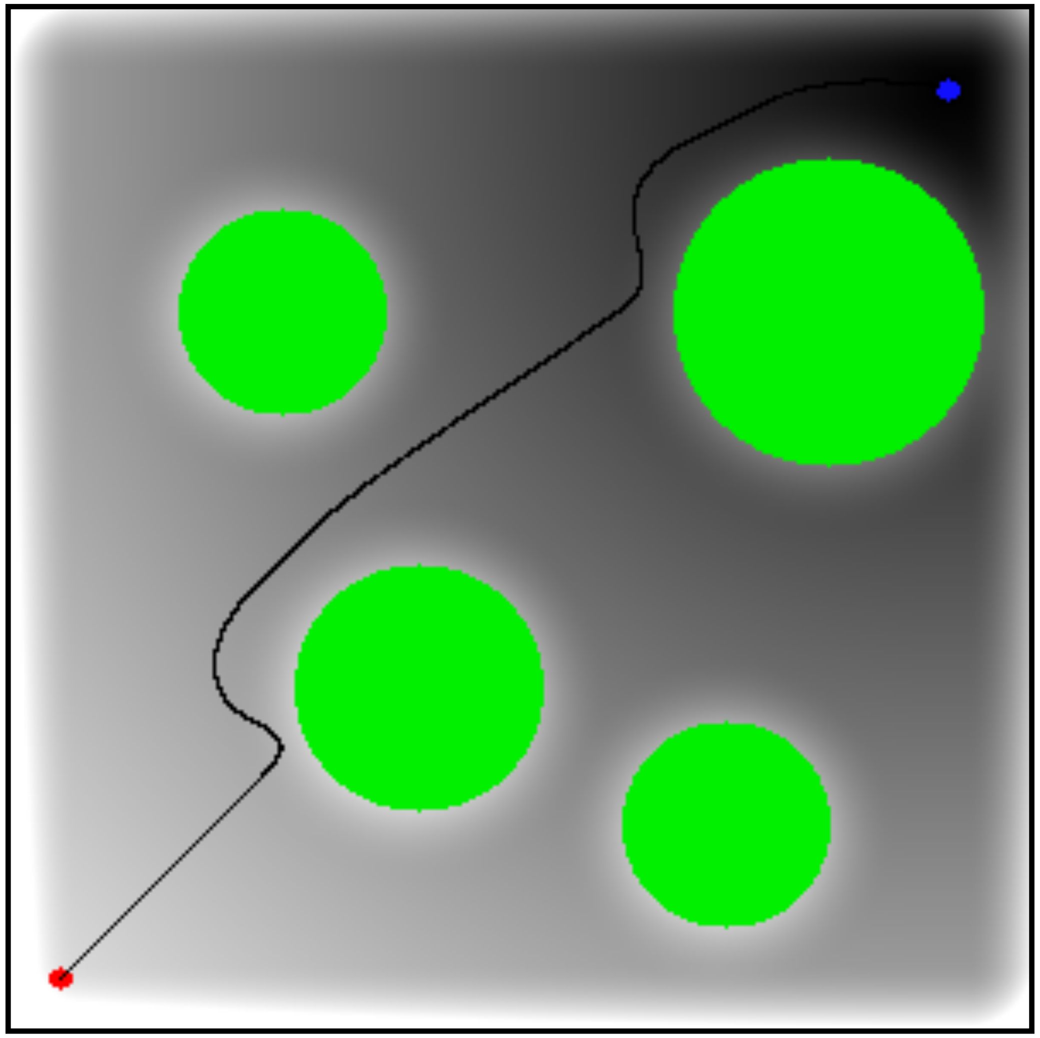
POTENTIAL FIELD PROBLEMS

- There are many parameters to tune
  - Strength of the field around each obstacle
  - Function for field strength around obstacle
  - Steepness of force toward the goal
  - Maximum velocity and mass
- Goals conflict
  - High field strength avoids collisions but produces big forces and unnatural motion
  - Higher mass smoothes paths but increases likelihood of collisions
- Local minima cause problems in general

BLOOPERS

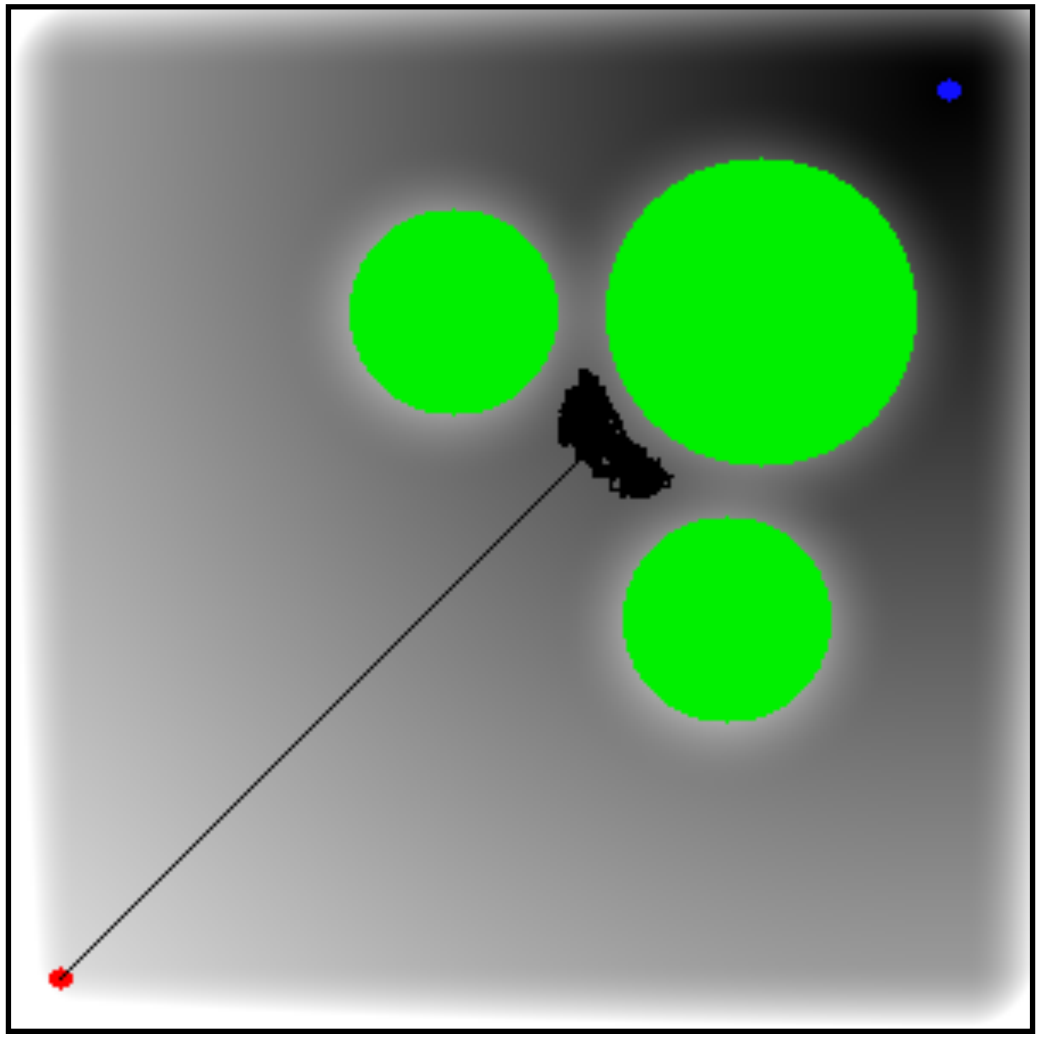


Field too weak



Field too strong

LOCAL MINIMA EXAMPLE



THE LOCAL MINIMA PROBLEM

- Path planning can be viewed as optimization
- Potential field planning is gradient descent optimization
- Gradient descent can get stuck in local minima
- How to work around local minima?
  - Determine if in a local minimum
  - Try another path

OBSTACLE NAVIGATION DEMOS

- Potential fields in robotics:
  - https://www.youtube.com/watch?v=0frsJq36Wpw
- Local minima example:
  - https://www.youtube.com/watch?v=62G78DeQzY8

WHAT ABOUT NAVIGATING IN 3D?

- Does A* work?
  - Sure! But we now need to navigate with volumes rather than meshes

VOLUME NAVIGATION

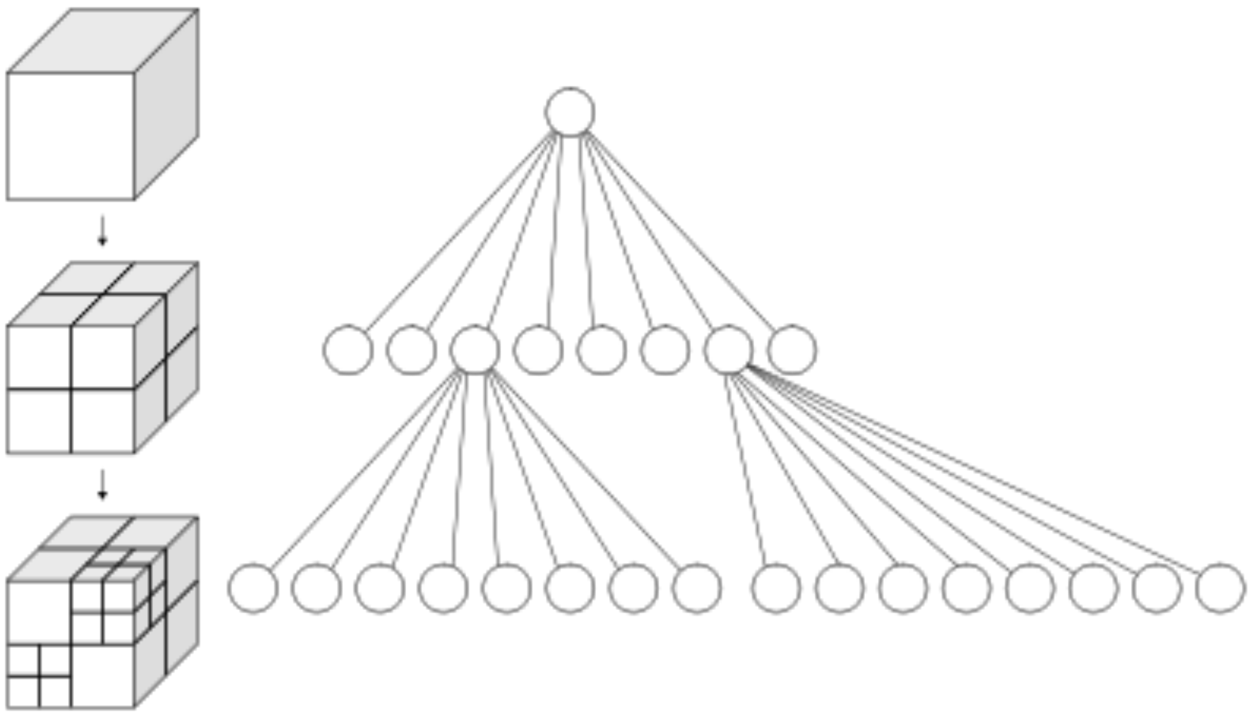
- Nodes are now in volumes rather than along a surface
- Called a voxel grid
  - Cells are evenly sized and spaced
- Voxels are flagged as blocking or non-blocking based on the underlying geometry
- Movements allowed to any of the non-blocking neighboring voxels

ISSUES?

- Path complexity grows exponentially without careful voxel management
  - i.e. Path is in three dimensions but is fixed and relatively constrained
- Voxel granularity tied to both accuracy of pathing and complexity of space and time
  - How can we fix this?

OCTREES

- Spatial data structure for subdividing a space into evenly-sized cubes
- Cubes can be subdivided independent of neighboring cubes
    - Increase subdivision for areas in need of finer granularity



AUTONOMOUS NAVIGATION

- Technique to allow 6 DoF (degrees of freedom) without heavy penalty of space and time overhead
- Use of raycasts or volumes to detect upcoming collisions with obstacles
- Upon detecting an object:
    - Agent applies force to prevent a collision
    - Agent rotates relative to the object to continue on trajectory

COMBINING TECHNIQUES

- Can combine autonomous navigation with previous “path-finding” techniques
- Useful in real-world applications like robotics
- Care must be taken with multiple entities
  - No shared path or information between agents
- Example:
  - https://www.youtube.com/watch?v=ka7Yb_XELAU

DESIGNING FOR SYSTEM LIMITATIONS

- Good design can aid in reducing computation
- Tightly constrained levels are fun but less processing-intensive in 6 DoF games

Descent (1995)

PROJECTION-BASED NAVIGATION

- Combines a 2D navmesh with sensors
  - Agent projected onto the underlying navmesh to perform classic A* path-finding
  - Use of limited sensors along trajectory detect upcoming collisions and adjust position accordingly
- Fewer sensors required as the navmesh is still doing most of the pathfinding work
- Types of sensors can be adjusted based on expected agent behavior

FURTHER READING

- <https://medium.com/ironequal/pathfinding-like-a-king-part-1-3013ea2c099>
- <https://medium.com/ironequal/pathfinding-like-a-king-part-2-4b74588262af>