

# Robot Localization: Historical Context

- Initially, roboticists thought the world could be modeled exactly
- Path planning and control assumed perfect, exact, deterministic world
- Reactive robotics (behavior based, ala bug algorithms) were developed due to imperfect world models
- But Reactive robotics assumes accurate control and sensing to react – also not realistic
- Reality: imperfect world models, imperfect control, imperfect sensing
- Solution: Probabilistic approach, incorporating model, sensor and control uncertainties into localization and planning
- Reality: these methods work empirically!



## Map Learning and High Speed Navigation in RHINO

Sebastian Thrun, Arno Bücken Wolfram  
Burgard Dieter Fox, Thorsten Fröohlinghaus  
Daniel Hennig Thomas Hofmann Michael  
Krell Timo Schmidt

An indoor mobile robot that uses sonar and  
vision to map its environment in real-time

# Case Study: Map Learning and High Speed Navigation in RHINO

- Distributed and decentralized processing. Control is distributed and decentralized. Several onboard and offboard machines are dedicated to several subproblems of modeling and control. All communication between modules is asynchronous. There is no central clock, and no central process controls all other processes.
- Anytime algorithms. Anytime algorithms are able to make decisions regardless of the time spent for computation. Whenever possible, anytime algorithms are employed to ensure that the robot operates in realtime.
- Hybrid architecture. Fast, reactive mechanisms are integrated with computationally intense, deliberative modules.
- Models. Models, such as the two dimensional maps described below, are used at all levels of the architecture.
- Whenever possible, models are learned from data.
- Learning. Machine learning algorithms are employed to increase the flexibility and the robustness of the system. Thus far, learning has proven most useful close to the sensory side of the system, where algorithms such as artificial neural networks interpret the robot's sensors.
- Modularity. The software is modular. A plug and play architecture allows us to quickly reconfigure the system, depending on the particular configuration and application. Sensor fusion. Different sensors have different perceptual characteristics. To maximize the robustness of the approach, most of the techniques described here rely on more than just a single type of sensor.

# Case Study: Map Learning and High Speed Navigation in RHINO

- Distributed and decentralized processing. Control is distributed and decentralized. Several onboard and offboard machines are dedicated to several subproblems of modeling and control. All communication between modules is asynchronous. There is no central clock, and no central process controls all other processes.
- Anytime algorithms. Anytime algorithms are able to make decisions regardless of the time spent for computation. Whenever possible, anytime algorithms are employed to ensure that the robot operates in realtime.
- Hybrid architecture. Fast, reactive mechanisms are integrated with computationally intense, deliberative modules.
- Models. Models, such as the two dimensional maps described below, are used at all levels of the architecture.
- Whenever possible, models are learned from data.
- Learning. Machine learning algorithms are employed to increase the flexibility and the robustness of the system. Thus far, learning has proven most useful close to the sensory side of the system, where algorithms such as artificial neural networks interpret the robot's sensors.
- Modularity. The software is modular. A plug and play architecture allows us to quickly reconfigure the system, depending on the particular configuration and application. Sensor fusion. Different sensors have different perceptual characteristics. To maximize the robustness of the approach, most of the techniques described here rely on more than just a single type of sensor.

# Requirements of a Map Representation for a Mobile Robot

- The precision of the map needs to match the precision with which the robot needs to achieve its goals
- The precision and type of features mapped must match the precision of the robot's sensors
- The complexity of the map has direct impact on computational complexity for localization, navigation and map updating

# Mapping: Occupancy Grids

- 2D metric occupancy grids are used
- Each grid cell has probability of the cell being occupied,  $\text{Prob}(\text{occ}(x,y))$
- Occupancy grid integrates multiple sensors (e.g. sonar and stereo)

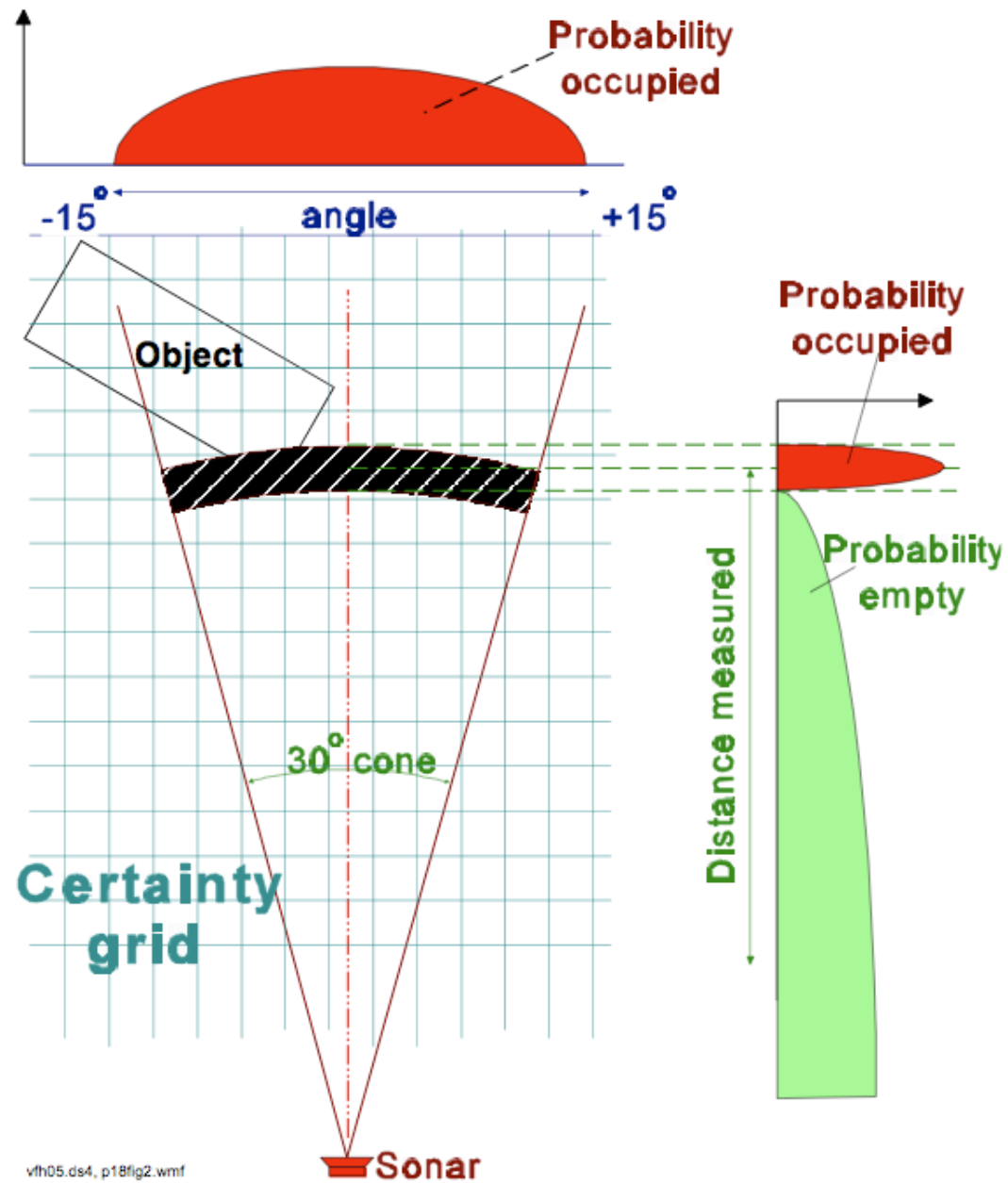


# Sonar Data Interpretation

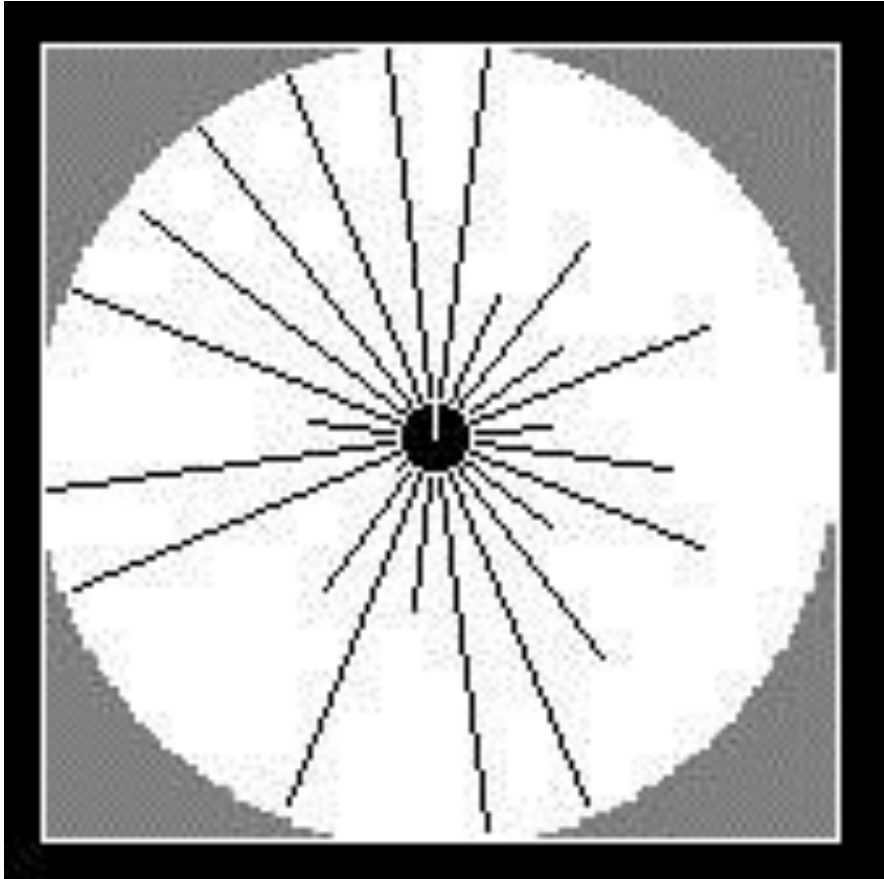
- Need to translate sonar distances into occupancy values:  $\text{Prob}(\text{occ}(x,y))$
- Method: Neural Net, trained on sonar responses
- RHINO uses a  $360^\circ$  ring of sonars
- Input to net: 4 readings nearest  $(x,y)$  – encoded as polar coordinates
- Output:  $\text{Prob}(\text{occ}(x,y))$
- Training data: train with physical robot on real known environments or use robot simulator
- May need to train anew in different environments –wall textures etc.
- Key point: multiple spatial readings needed to overcome noise and sonar effects

# Sonar Sweeps a Wide Cone

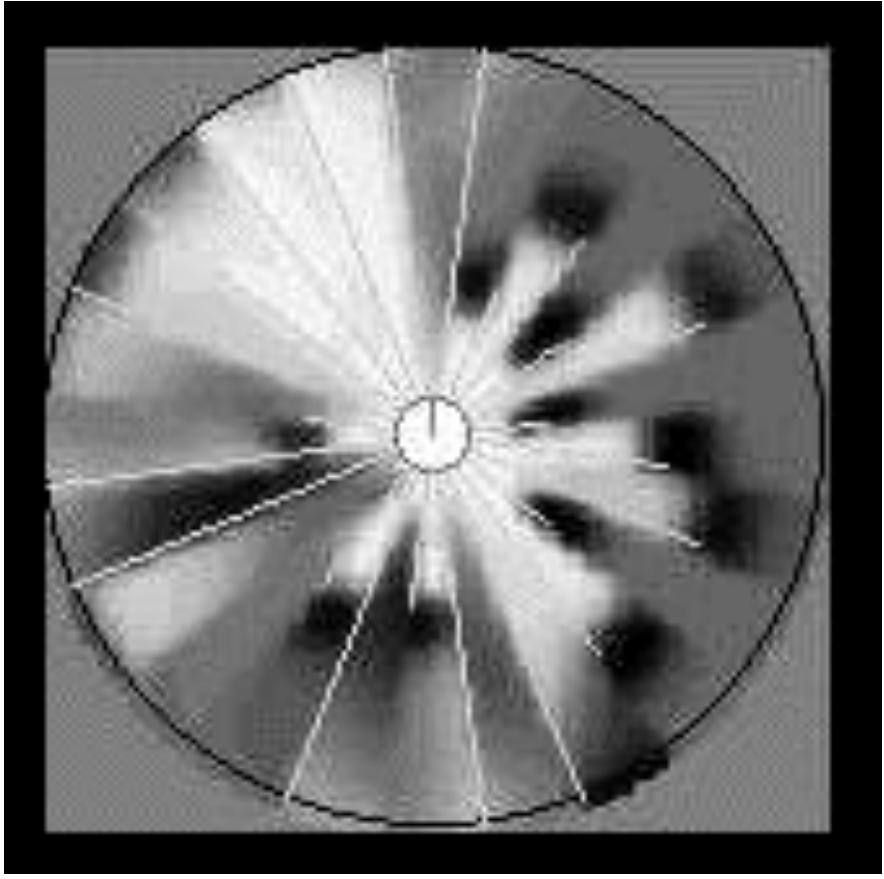
- Obstacle could be anywhere on the arc at distance D.
- The space closer than D is likely to be free.







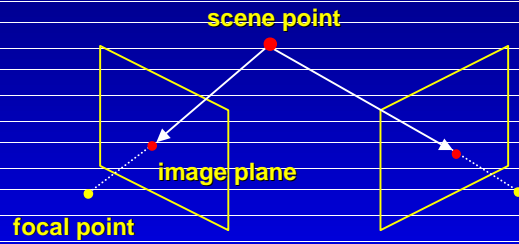
Sonar scan



Probabilistic Occupancy Grid

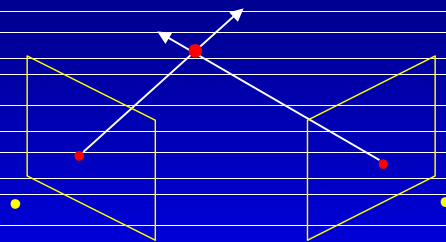
## Stereo

---



## Stereo

---



### *Basic Principle: Triangulation*

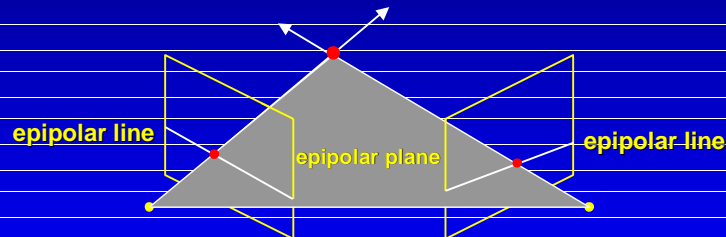
- Gives reconstruction as intersection of two rays
- Requires *point correspondence*

## Stereo Correspondence

---

### *Determine Pixel Correspondence*

- Pairs of points that correspond to same scene point



### *Epipolar Constraint*

- Reduces correspondence problem to 1D search along *conjugate epipolar lines*

## Stereo Matching Algorithms

---

### *Match Pixels in Conjugate Epipolar Lines*

- Assume color of point does not change
- Pitfalls
  - > specularities
  - > low-contrast regions
  - > occlusions
  - > image error
  - > camera calibration error
- Numerous approaches
  - > dynamic programming [Baker 81, Ohta 85]
  - > smoothness functionals
  - > more images (trinocular, N-ocular) [Okutomi 93]
  - > graph-cuts [Boykov 00]

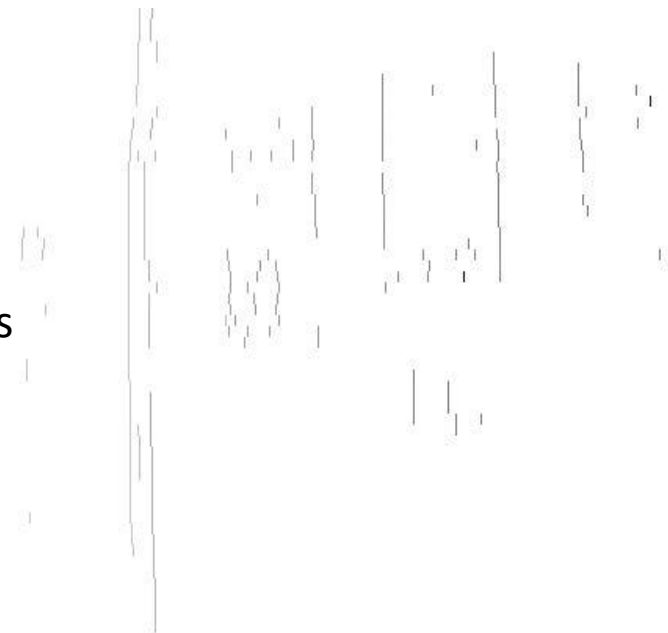
# Stereo Data Interpretation

- Vertical edges (doorways, vertical corners, obstacles) are found in each image and triangulated for 3D depth.
- 3D edge points are projected onto the occupancy grid after being enlarged by the robot radius
- Enlargement allows robot to navigate without hitting corners
- Stereo can miss featureless, homogeneous areas like blank walls
- Integration with sonar can improve mapping accuracy

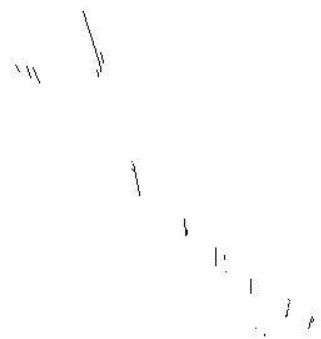
image



Vertical edges



Vertical edge  
projection



Occupancy  
grid

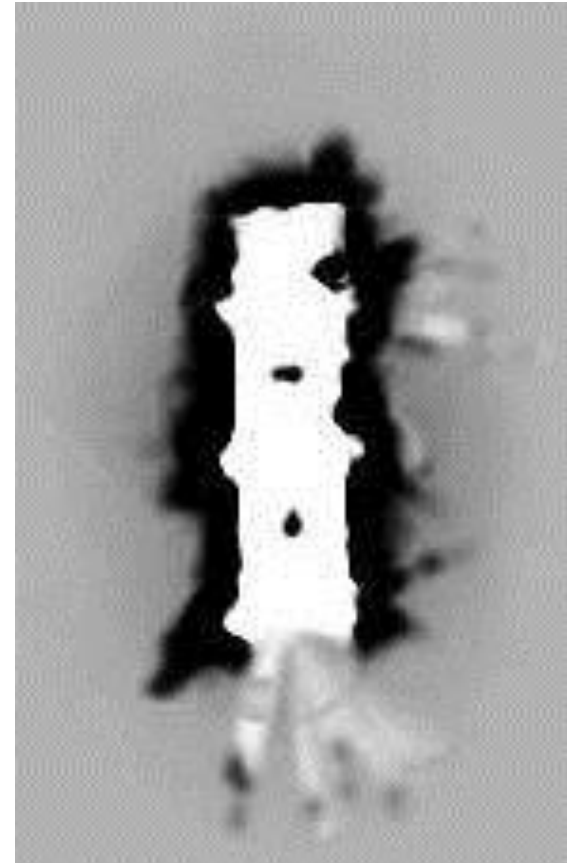
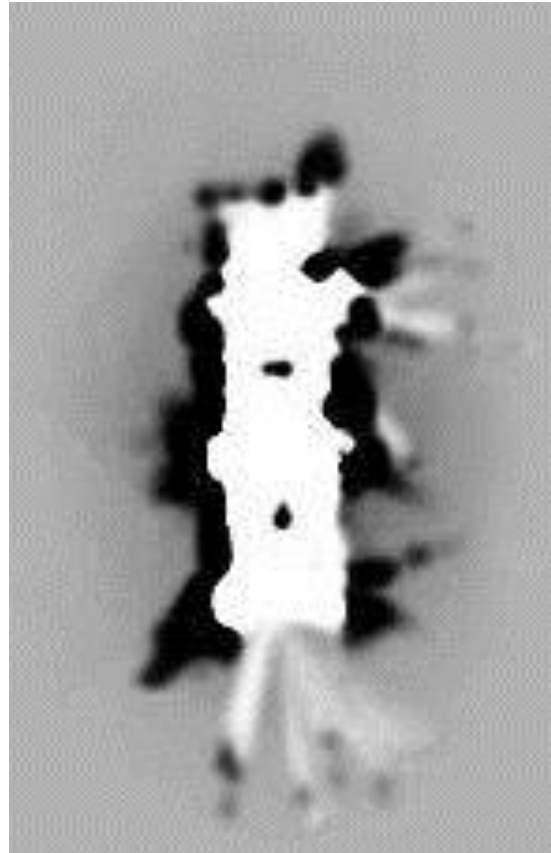
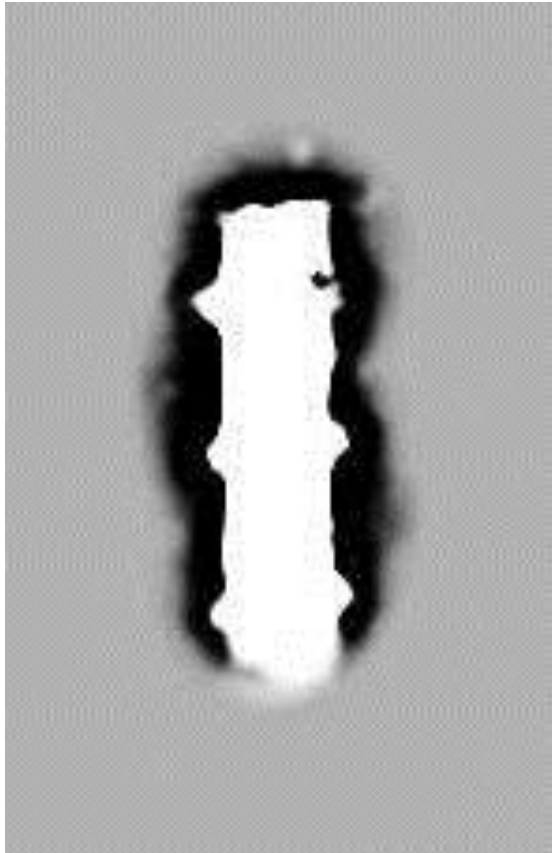


**Results from Stereo matching**

# Updating over time

- Mobile robot is moving and making multiple measurements at each sensing time step
- Need to integrate the new values from the sensors with the current occupancy grid values
- These are probabilistic measures, so a Bayes rule update is used to find new probability of occupancy (more on this later....)

# Integration results



Maps built in a single run (a) using only sonar sensors, (b) using only stereo information, and (c) integrating both. Notice that sonar models the world more consistently, but misses the two sonar absorbing Chairs which are found using stereo vision.

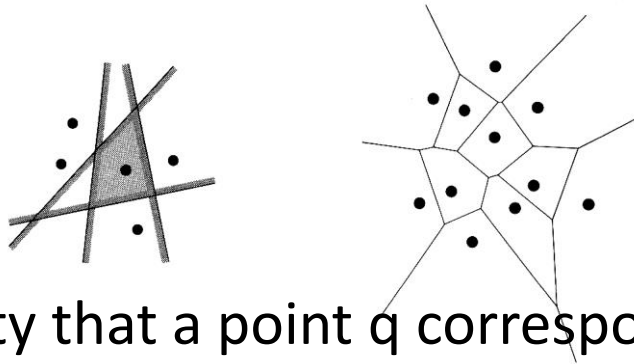
# Topological Maps

- Compact representation, usually as a graph
- Nodes are distinct places, arcs (edges) represent adjacency
- Regions of free-space are nodes, and edges represent connections for adjacency
- Method:
  - Create Voronoi diagram
  - Find critical points – bottlenecks or choke points in the Voronoi diagram
  - Formally, a threshold *epsilon* for minimum distance to obstacle locally
  - Find critical lines: partitions between regions at bottlenecks
  - Partitions are used to form a graph. Nodes are regions, arcs are adjacent regions separated by critical lines



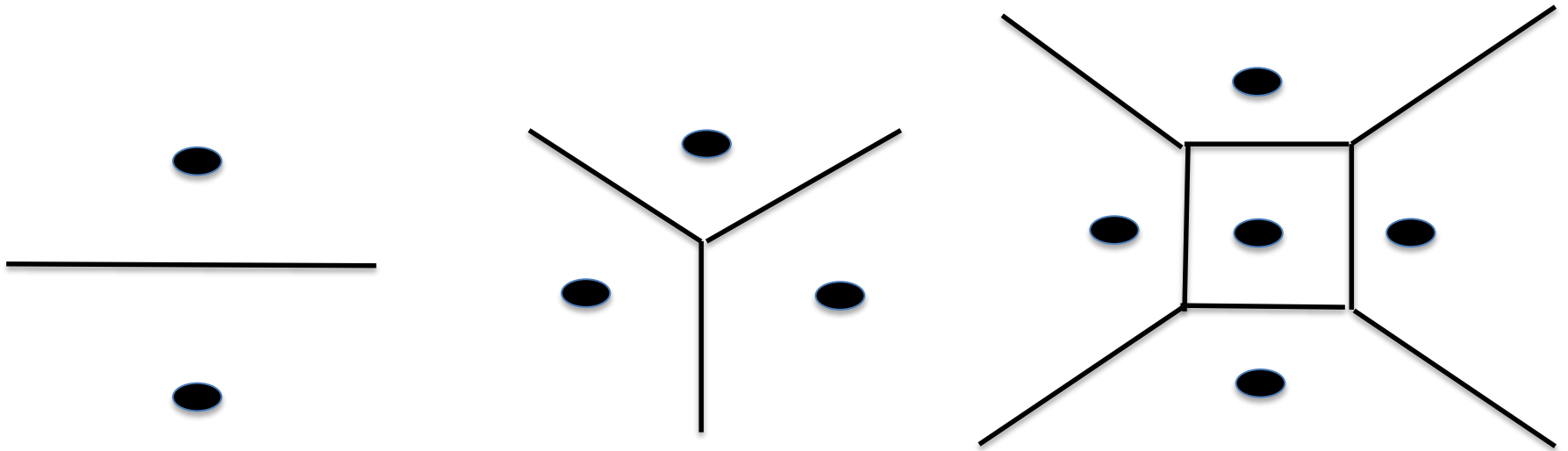
# Voronoi Path Planning

- Find paths that are not close to obstacles, but in fact as far away as possible from obstacles.
- This will create a maximal safe path, in that we never come closer to obstacles than we need.
- Voronoi Diagram in the plane. Let  $P = \{p_i\}$ , set of points in the plane, called *sites*. Voronoi diagram is the subdivision of the plane into  $N$  distinct cells, one for each *site*.



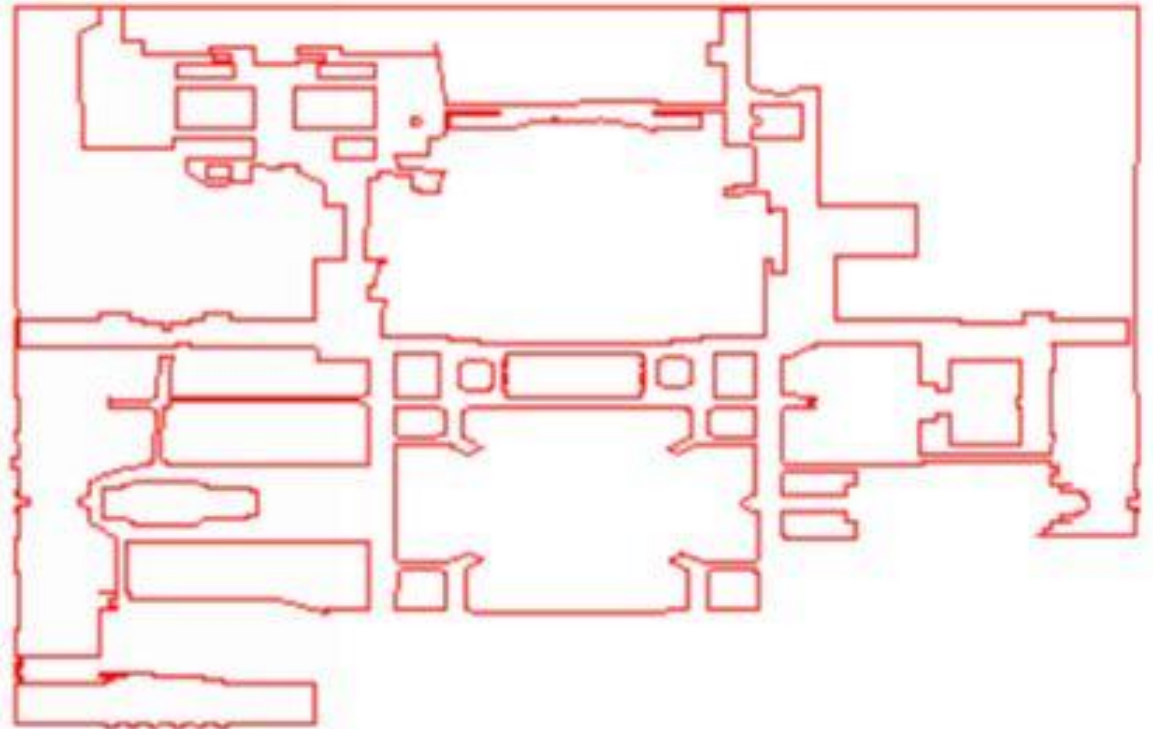
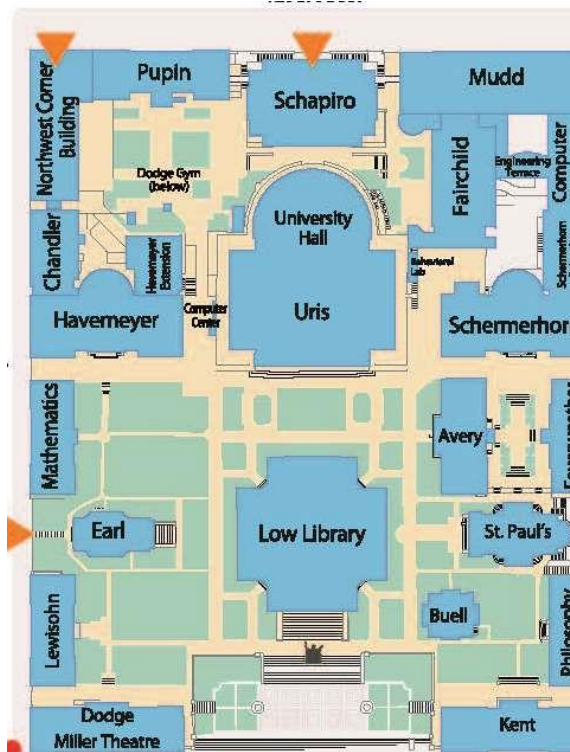
- Cell has property that a point  $q$  corresponds to a site  $p_i$  iff:  
$$\text{dist}(q, p_i) < \text{dist}(q, p_j) \text{ for all } p_j \in P, j \neq i$$

# Voronoi Graph



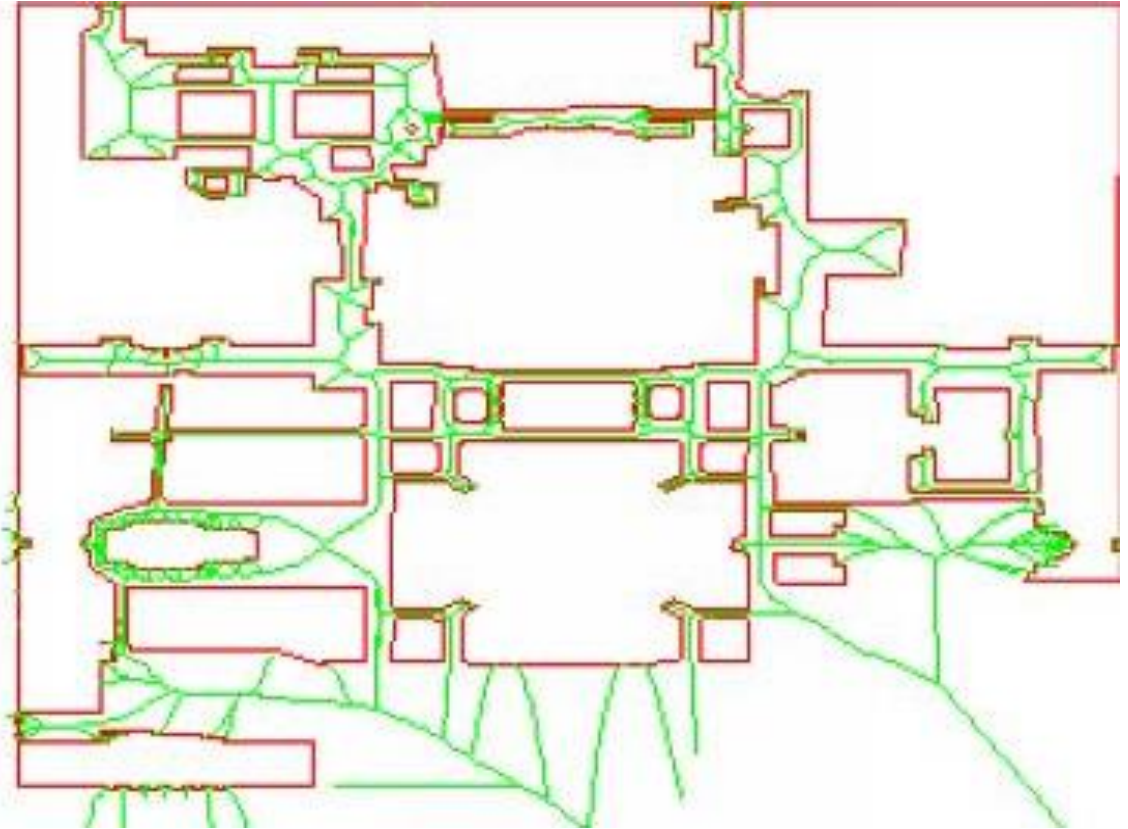
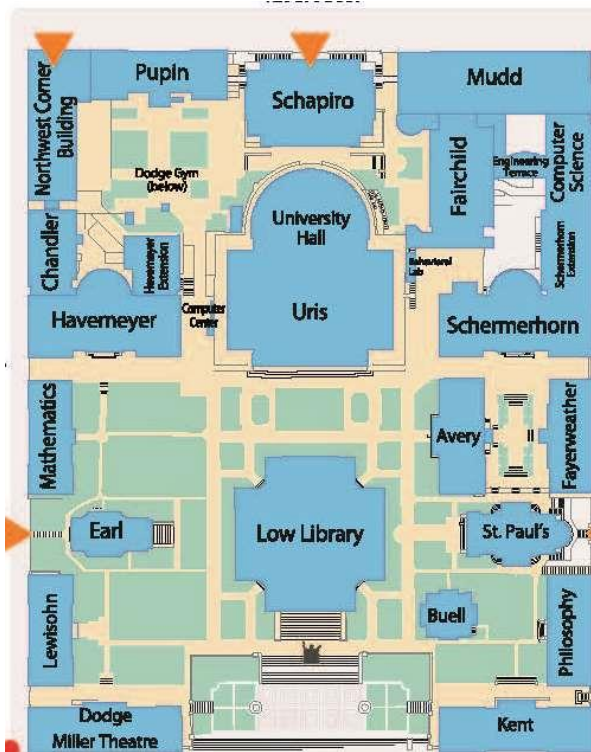
- Intuitively: Edges and vertices are intersections of perpendicular bi-sectors of point-pairs
- Edges are equidistant from 2 points
- Vertices are equidistant from 3 points
- Online demo: <http://alexbeutel.com/webgl/voronoi.html>

# Voronoi Path on Columbia Campus



To find a path:

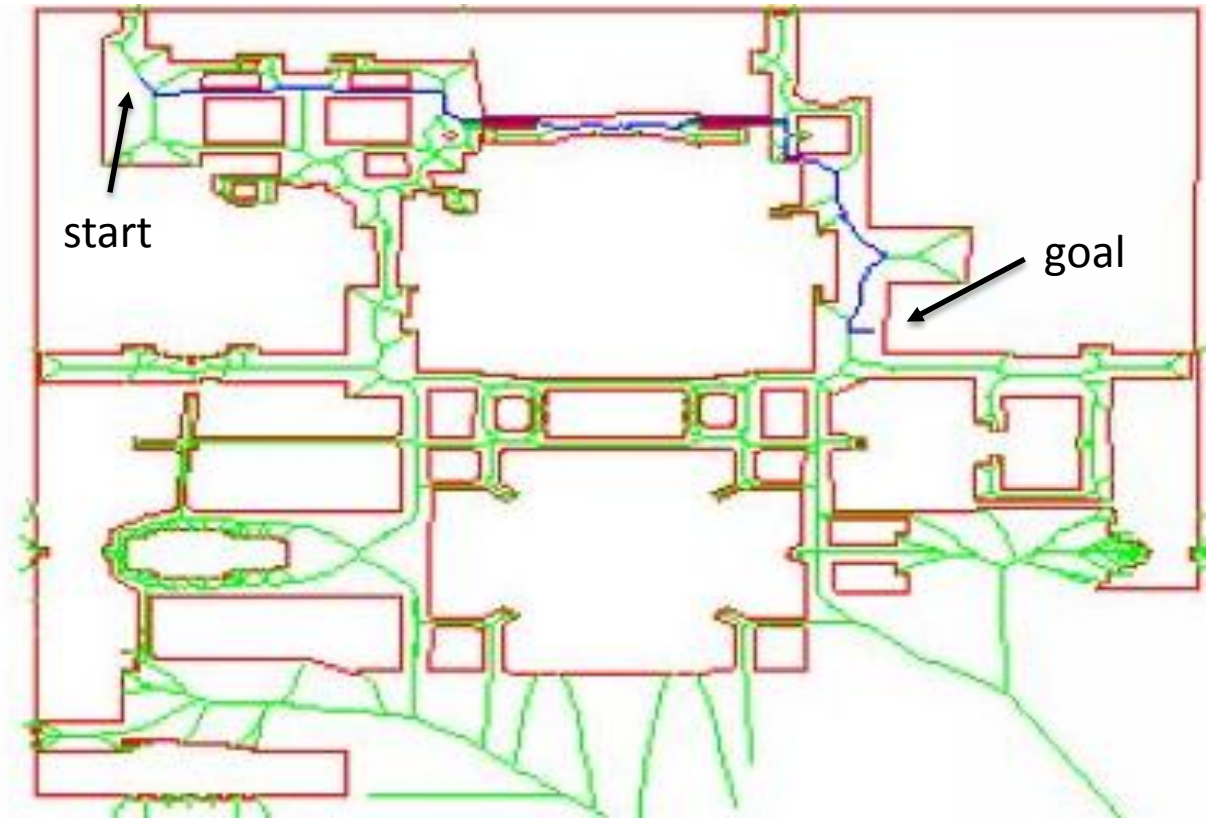
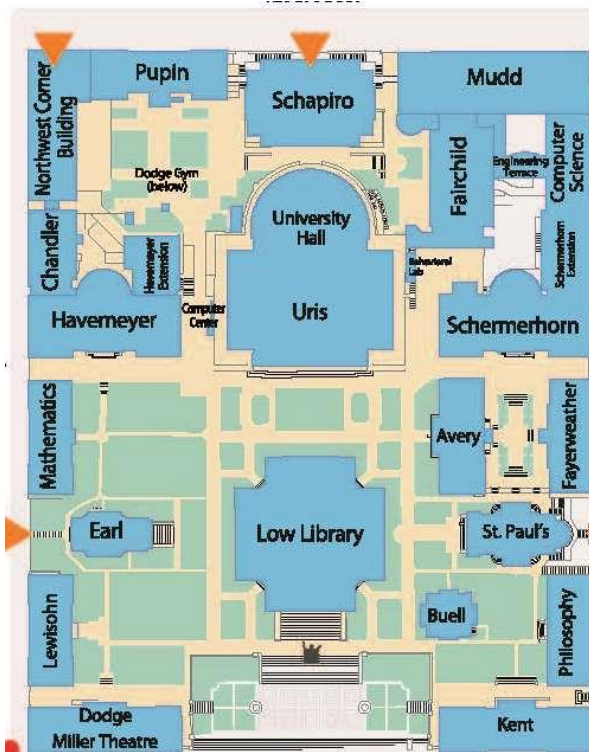
# Voronoi Path on Columbia Campus



To find a path:

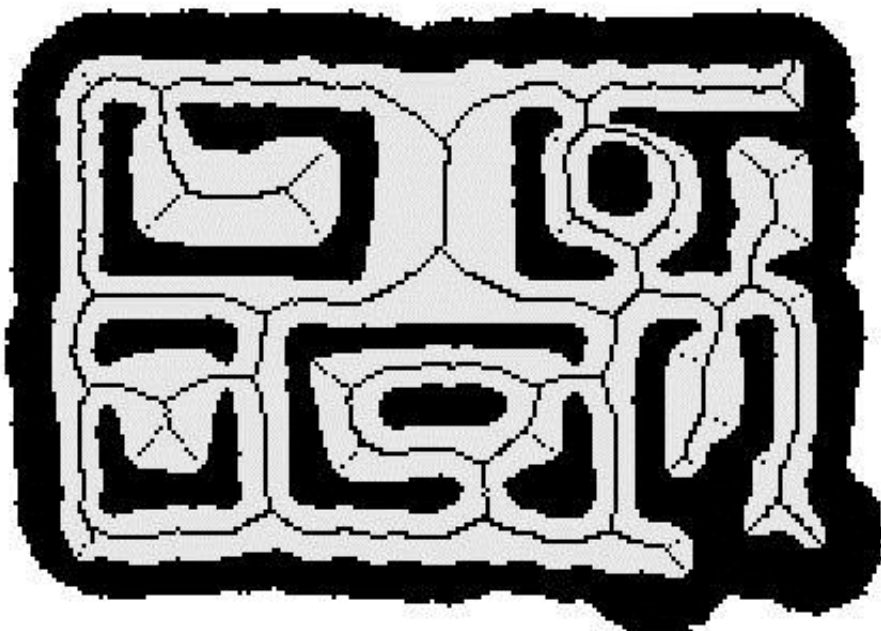
- Create Voronoi graph -  $O(N \log N)$  complexity in the plane
- Connect  $q_{start}$ ,  $q_{goal}$  to graph – local search
- Compute shortest path from  $q_{start}$  to  $q_{goal}$  ( $A^*$  search)

# Voronoi Path on Columbia Campus

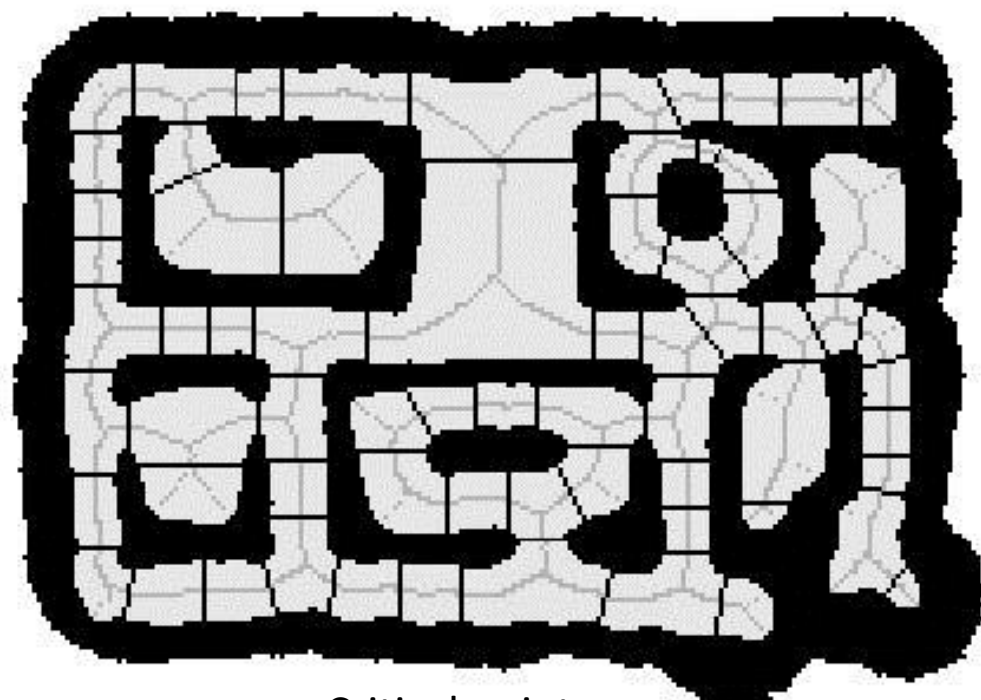


To find a path:

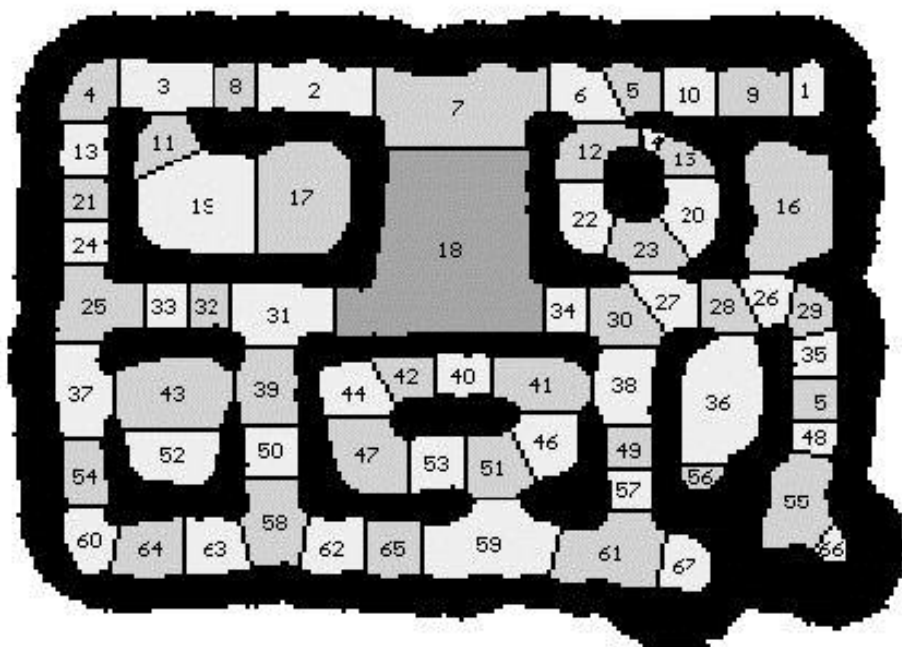
- Create Voronoi graph -  $O(N \log N)$  complexity in the plane
- Connect  $q_{start}$ ,  $q_{goal}$  to graph – local search
- Compute shortest path from  $q_{start}$  to  $q_{goal}$  using  $A^*$



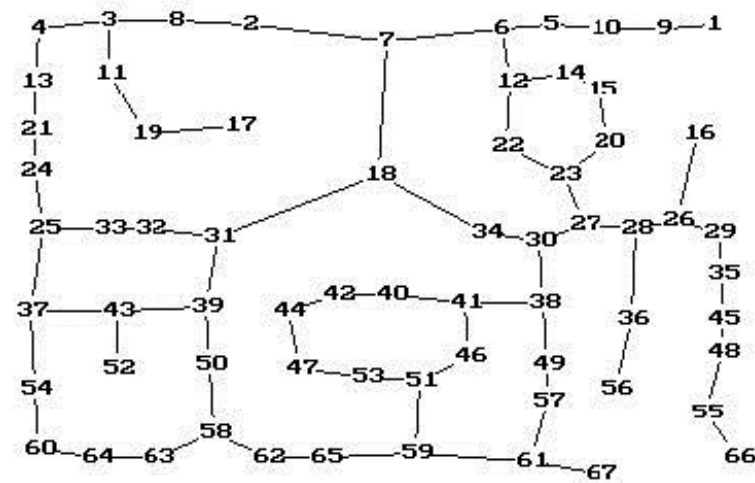
Voronoi Diagram



Critical points





Regions



Topological map

# Path Planning

- More on this later...for now suffice to use a graph search from a start region on the map to a goal region.



# TopBot: Topological Mobile Robot Localization Using Fast Vision Techniques

Paul Blaer and Peter Allen

Dept. of Computer Science, Columbia University

{psb15, allen}@*cs.columbia.edu*





# Autonomous Vehicle for Exploration and Navigation in Urban Environments

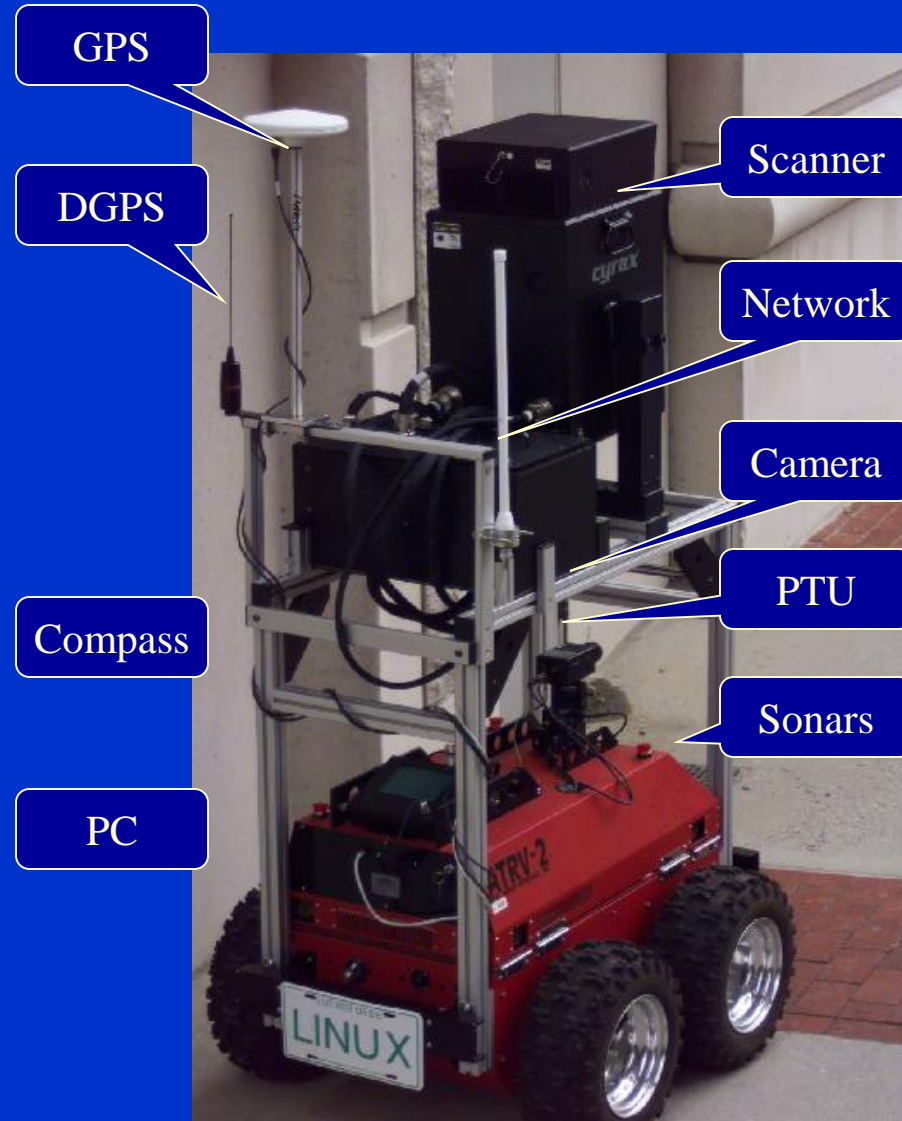
The AVENUE Robot:

- Autonomous.
- Operates in outdoor urban environments.
- Builds accurate 3-D models.

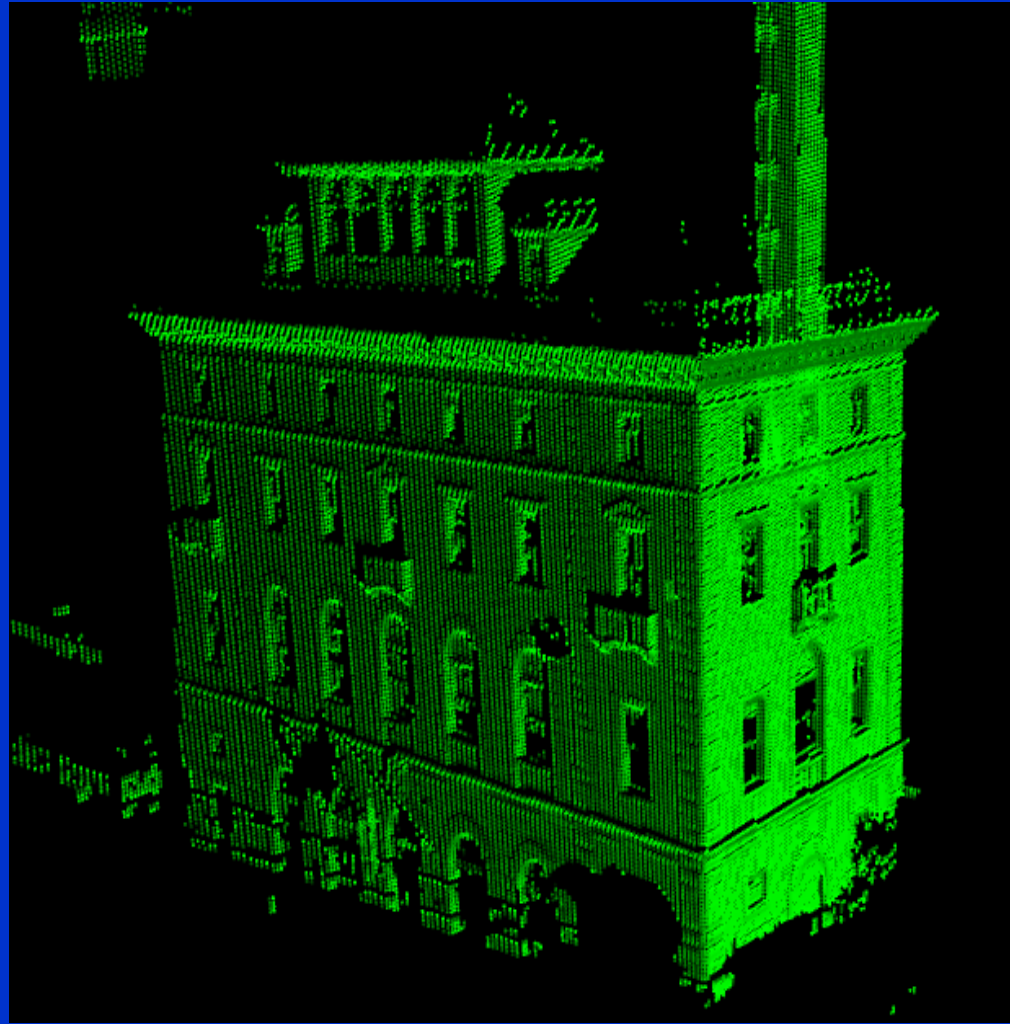
Current Localization

Methods:

- Odometry.
- Differential GPS.
- Vision.



# Range Scanning Outdoor Structures



# Italian House: Textured 3-D Model





# Topological Localization

- Odometry and GPS can fail.
  - Fine vision techniques need an estimate of the robot's current position.
- 

## Histogram Matching of Omnidirectional Images:

- Fast.
- Rotationally-invariant.
- Finds the region of the robot.
- This region serves as a starting estimate for the main vision system.



# Related Work

- Omnidirectional Imaging for Navigation.  
*Ulrich and Nourbakhsh '00 (with histogramming);  
Cassinis et al. '96; Winters et al. '00.*
  - Topological Navigation.  
*Kuipers and Byun '91;  
Radhakrishnan and Nourbakhsh '99.*
  - Other Approaches to Robot Localization.  
*Castellanos et al. '98; Durrant-Whyte et al. '99;  
Leonard and Feder '99; Thrun et al. '98;  
Dellaert et al. '99; Simmons and Koenig '95.*
- 
- AVENUE  
*Allen, Stamos, Georgiev, Gold, Blaer '01.*

# Building the Database

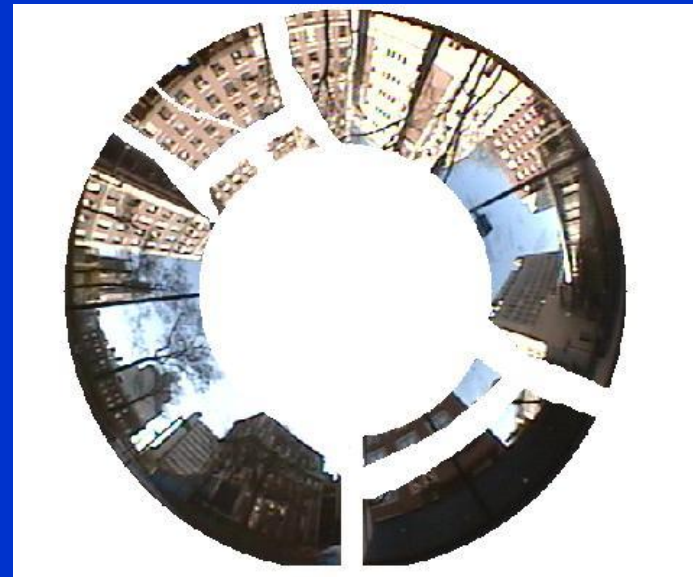
- Divide the environment into a logical set of regions.
- Reference images must be taken in all of these regions.
- The images should be taken in a zig-zag pattern to cover as many different locations as possible.
- Each image is reduced to three 256-bucket histograms, for the red, green, and blue color bands.





# Masking the Image

- The camera points up to get a clear picture of the buildings.
- The camera pointing down would give images of the ground's brick pattern - not useful for histogramming.
- With the camera pointing up, the sun and sky enter the picture and cause major color variations.
- We mask out the center of the image to block out most of the sky.
- We also mask out the superstructure associated with the camera.

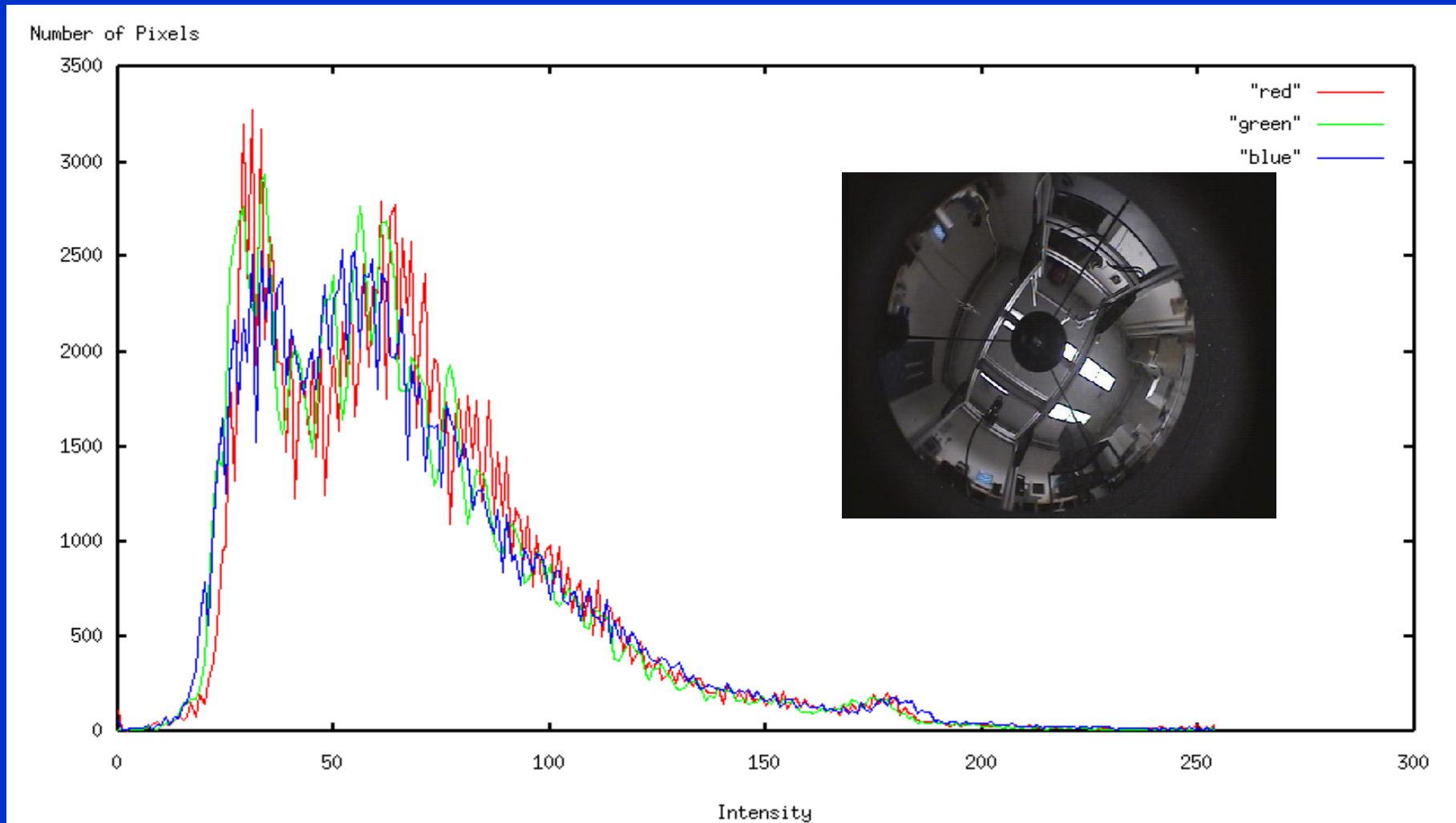


# Environmental Effects

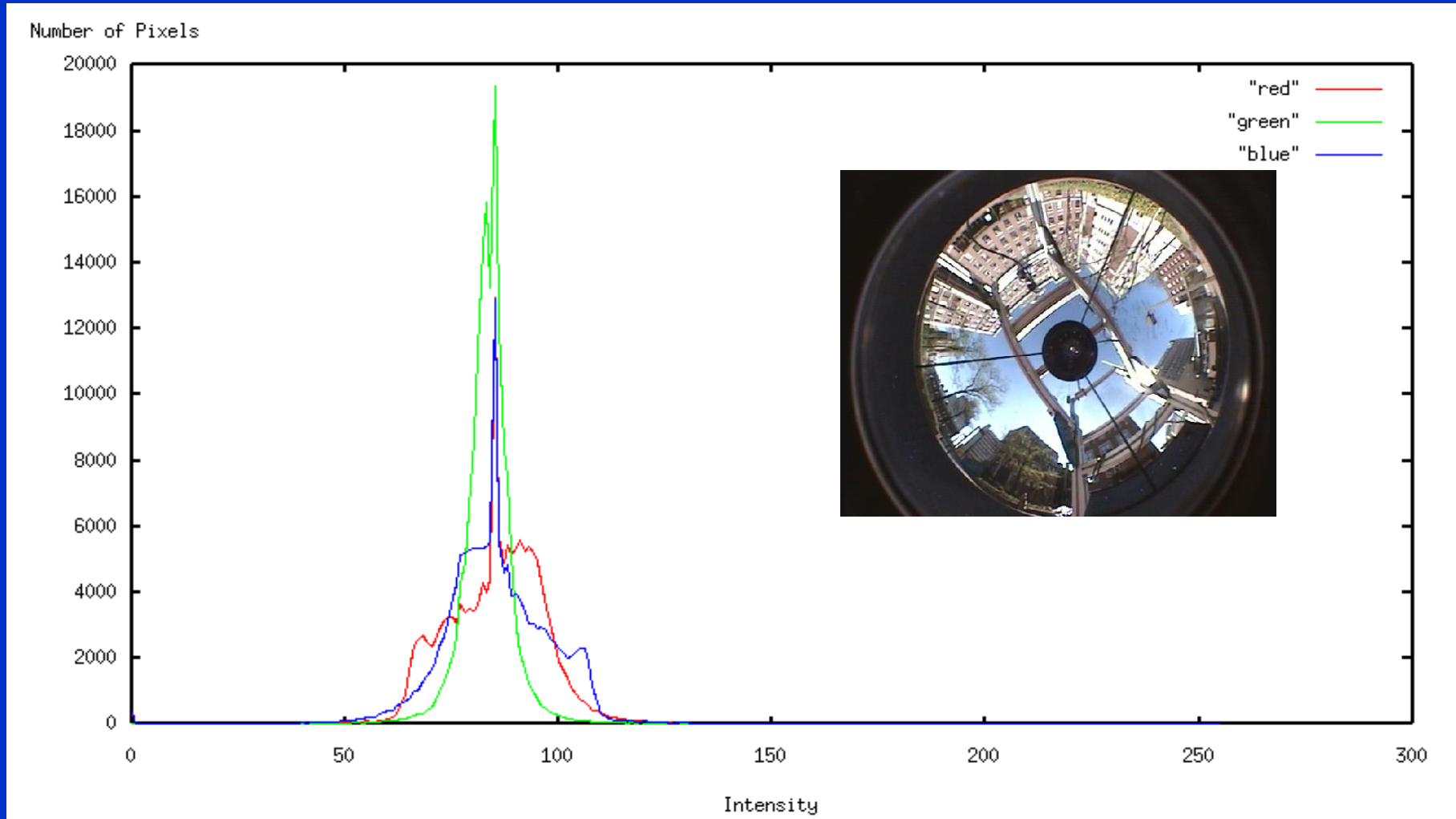
- Indoor environments
  - Controlled lighting conditions
  - No histogram adjustments
- Outdoor environments
  - Large variations in lighting
  - Use a histogram normalization with the percentage of color at each pixel:

$$\frac{R}{R+G+B} , \frac{G}{R+G+B} , \frac{B}{R+G+B}$$

# Indoor Image Non-Normalized Histogram



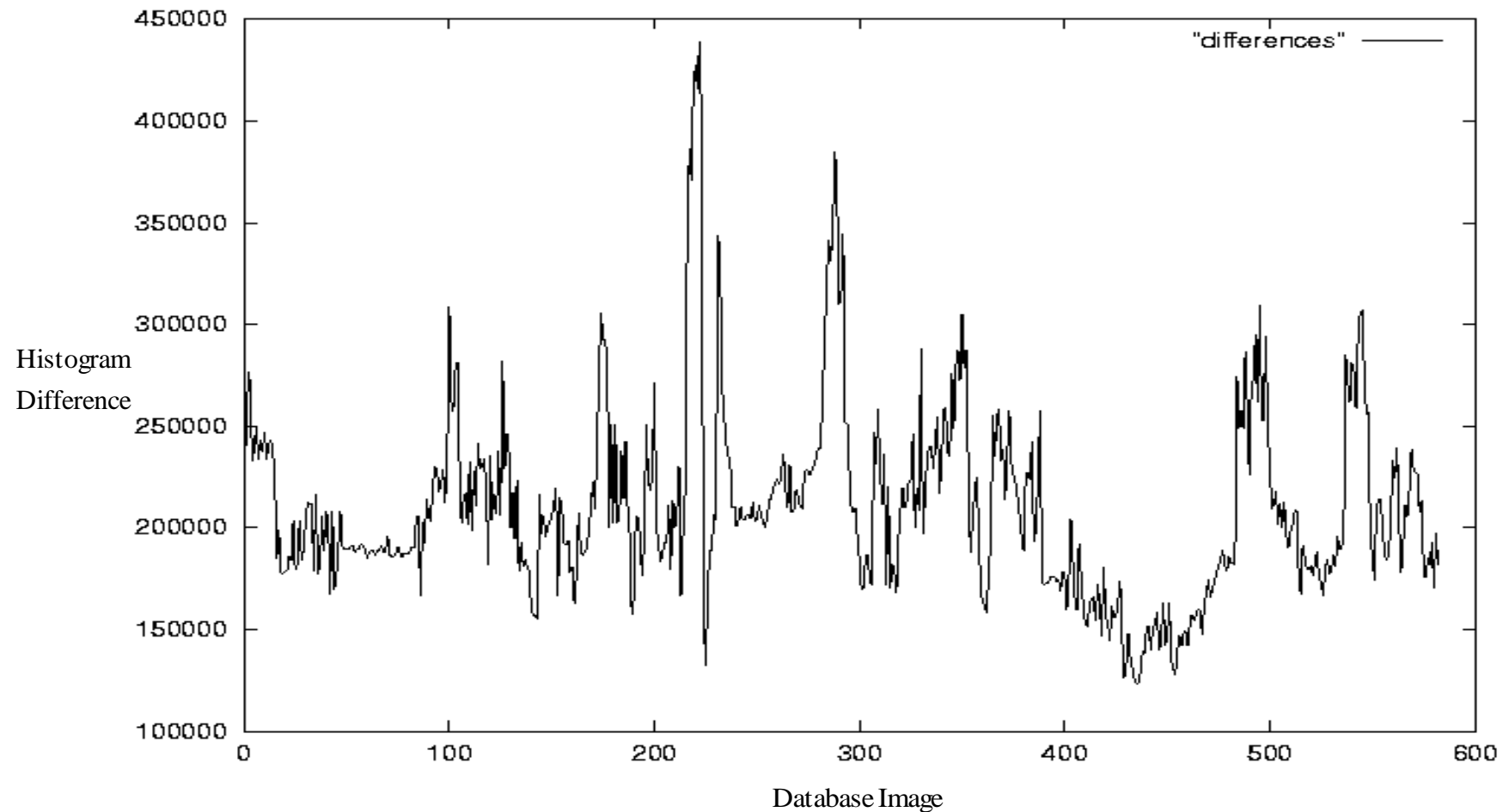
# Outdoor Image Normalized Histogram



# • • • Matching an Unknown Image with the Database

- Compare unknown image to each image in the database.
- Initially we treat each band (r, g, b) separately.
- The difference between two histograms is the sum of the absolute differences of each bucket.
- Better to sum the differences for all three bands into a single metric than to treat each band separately.
- The region of the database image with the smallest difference is the selected region for this unknown.

# Image Matching to the Database



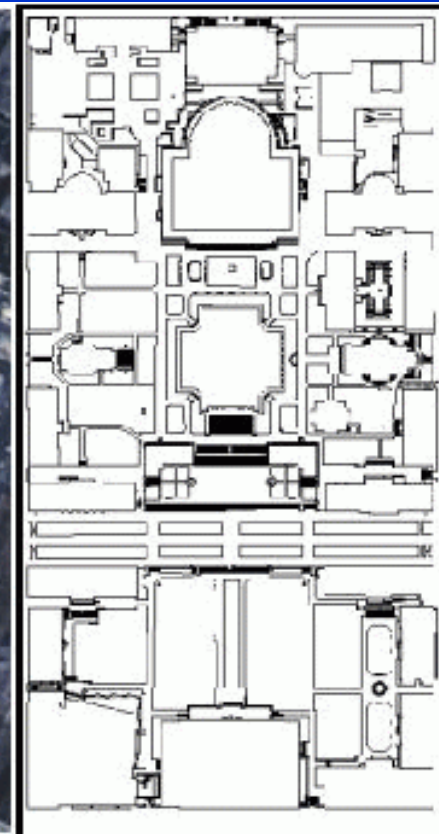
# The Test Environments



Indoor  
(Hallway View)



Outdoor  
(Aerial View)



Outdoor  
(Campus Map)

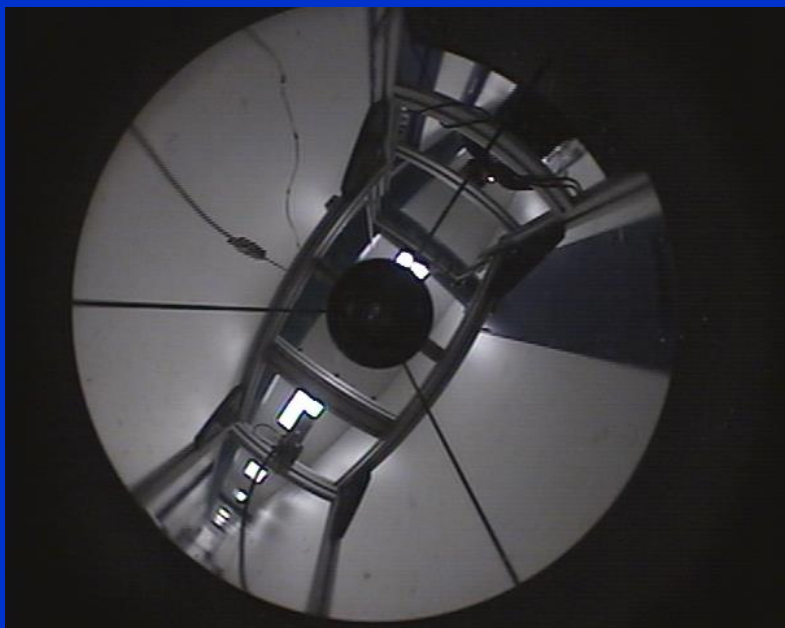
# Indoor Results

Region	Images Tested	Non-Normalized % Correct	Normalized % Correct
1	21	100%	95%
2	12	83%	92%
3	9	77%	89%
4	5	20%	20%
5	23	74%	91%
6	9	89%	78%
7	5	0%	20%
8	5	100%	40%
Total	89	78%	80%

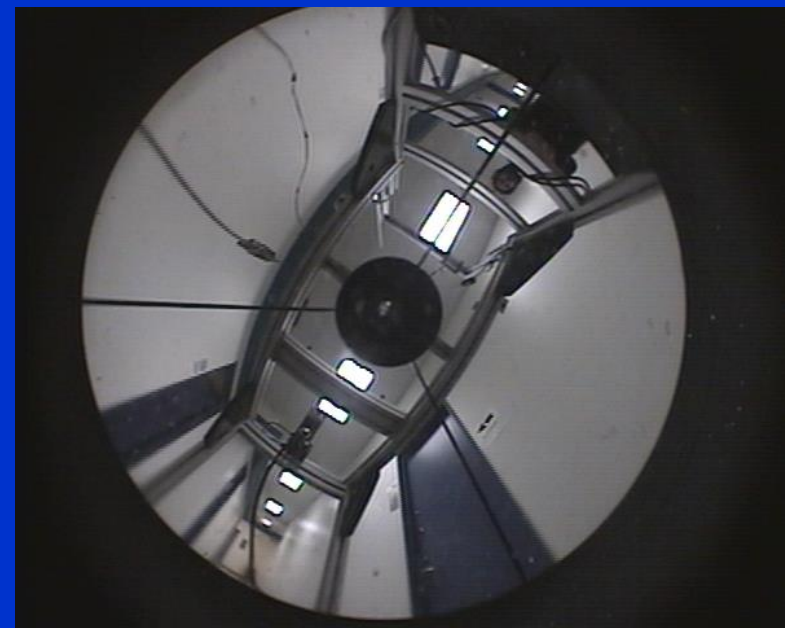


•  
•  
•

# Ambiguous Regions



South  
Hallway



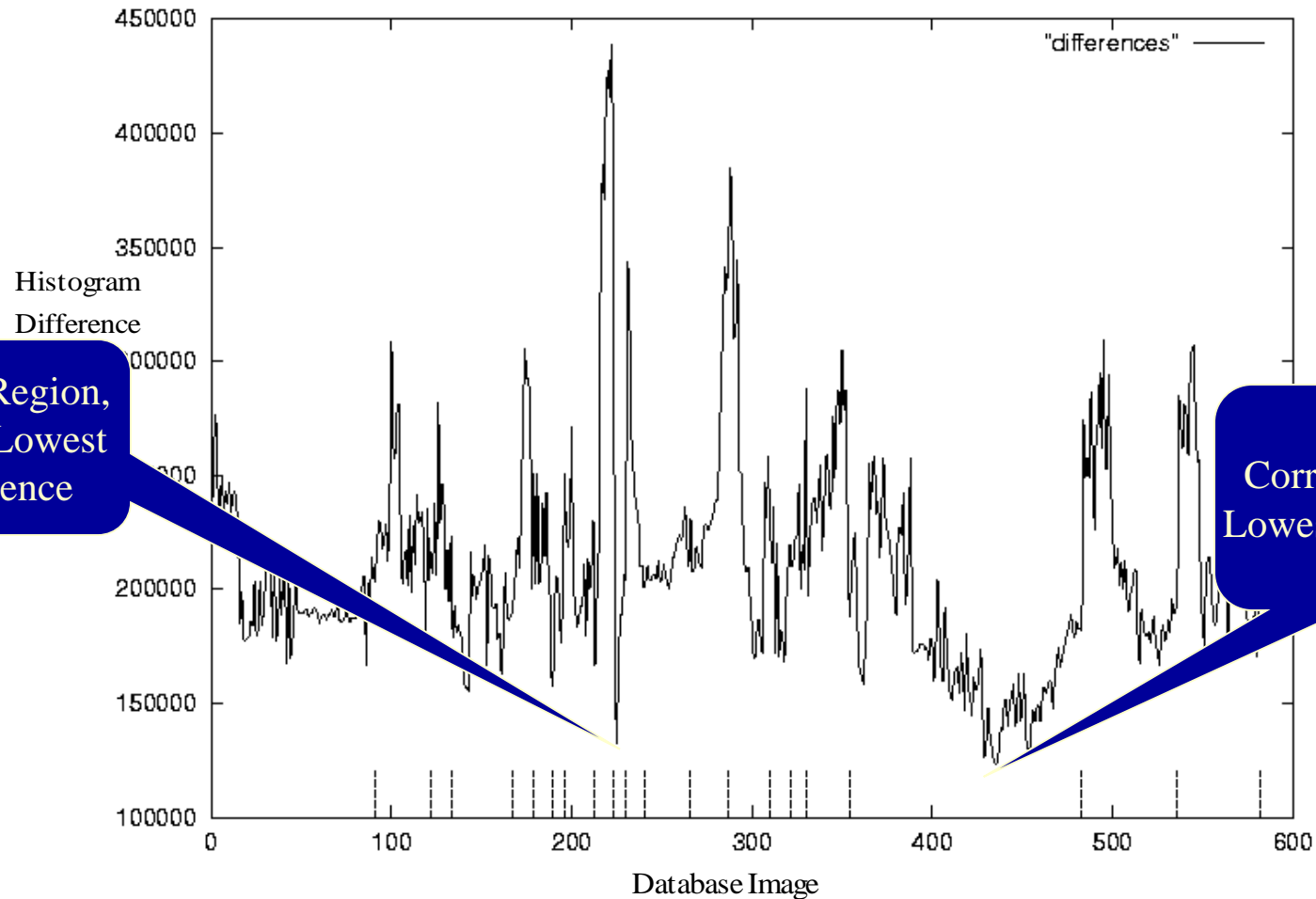
North  
Hallway

# Outdoor Results

Region	Images Tested	Non-Normalized % Correct	Normalized % Correct
1	50	58%	95%
2	50	11%	39%
3	50	29%	71%
4	50	25%	62%
5	50	49%	55%
6	50	30%	57%
7	50	28%	61%
8	50	41%	78%
Total	400	34%	65%

# The Best Candidate Regions

Wrong Region,  
Second-Lowest  
Difference



Correct Region,  
Lowest Difference

# Conclusions

- In 80% of the cases we were able to narrow down the robot's location to only 2 or 3 possible regions without any prior knowledge of the robot's position.
- Our goal was to reduce the number of possible models that the fine-grained visual localization method needed to examine.
- Our method effectively quartered the number of regions that the fine-grained method had to test.

# Future Work

- What is needed is a fast secondary discriminator to distinguish between the 2 or 3 possible regions.
- Histograms are limited in nature because of their total reliance on the color of the scene.
- To counter this we want to incorporate more geometric data into our database, such as edge images.

