# COMPUTER VISION FOR ROBOT NAVIGATION

Sanketh Shetty
Computer Vision and Robotics Laboratory
University of Illinois Urbana-Champaign

# MEET THE ROBOTS



# WHAT DOES A ROBOT CARE ABOUT WHEN NAVIGATING?

- Current Location and Destination (possibly intermediate tasks & goals)
- Minimizing damage to self
- Minimizing damage to environment
- Maximizing efficiency (energy spent vs. distance travelled)
- Knowing what objects it can interact with.
- Knowing how it can interact with its environment
- "Learning" about its operating environment (e.g. Map building)

# WHERE CAN VISION HELP?

- Robot Localization
- Obstacle avoidance
- Mapping (determining navigable terrain)
- Recognizing people and objects
- Learn how to interact (e.g. grasp) with objects

# CASE STUDY: DARPA GRAND CHALLENGE


- Stanley, VW Touareg
  - Learned discriminative machine learning models on Laser range-finding data to determine navigable vs. rough-obstacle-filled terrain.
  - Terrain classifier => speed bottleneck.
  - Used vision to extend the path-planning horizon (video)

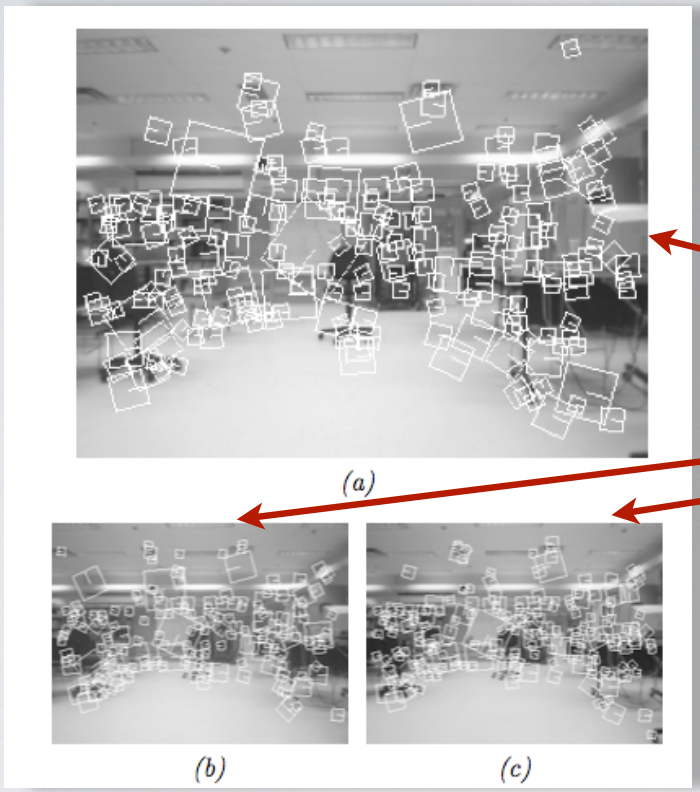
# TODAY'S PAPERS

- Vision based Robot Localization and Mapping using Scale Invariant Features, S. Se et al. (ICRA 2001)
- High Speed Obstacle Avoidance Using Monocular Vision and Reinforcement Learning, Michels et al. (ICML 2005)
- Opportunistic Use of Vision to Push Back the Path Planning Horizon, Nabbe et al. (IROS 2006)

# Vision based Robot Localization and Mapping using Scale Invariant Features

- Goal: Simultaneous Localization and Map Building using stable visual features.
  - Evaluated in an indoor environment.
- Prior Work: Used laser scanners and range finders for SLAM
  - Limited range.
  - Unsatisfactory description of the environment.
- Why do we care about building maps?

# SYSTEM DESCRIPTION



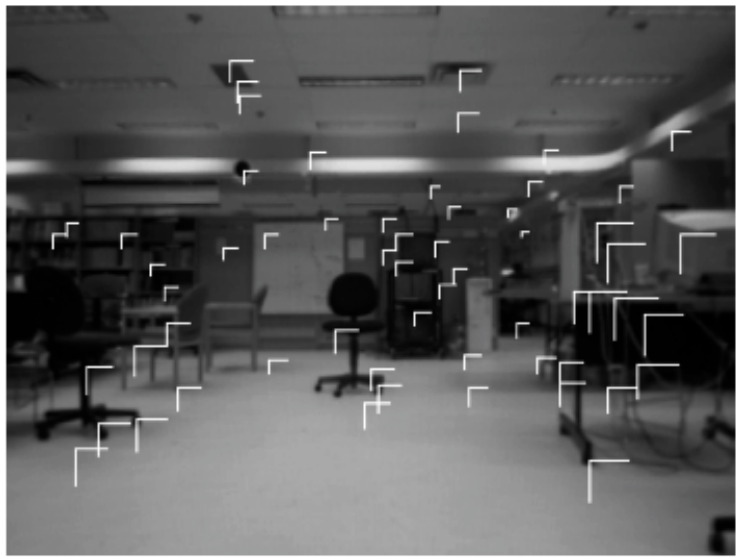


Obtain 3 images from Triclops camera.

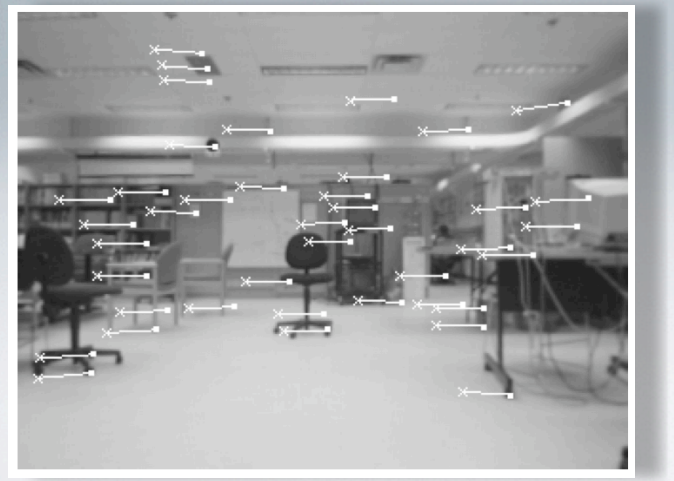
Detect SIFT Key-points.

# STEREO MATCHING OF KEY-POINTS

- Match key-points across 3 images using following criterion:
  - Epipolar Constraints
  - Disparity
  - Scale and Orientation
- Prune ambiguous matches.
- Calculate (X,Y,Z) world coordinates from disparity and camera parameters, for stable points.



# EGO-MOTION ESTIMATION



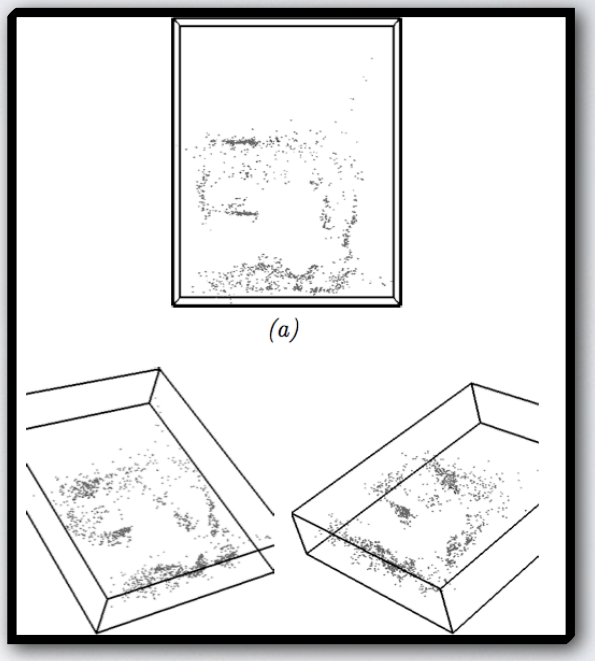
- Initialize solution of transformation between two frames from odometry data.
- Use least squares to find a correction to transformation matrix that better projects points in one frame to the next.

# LANDMARK TRACKING

- Each tracked SIFT landmark indexed by(X, Y, Z, scale, orientation, L)
- 4 Types of Landmark Points identified
- Details:
  - Track Initiation
  - Track Termination
  - Field of View
    - Only points with Z>0 and within 60 degree viewing angle of the Triclops are considered.

# RESULTS & DISCUSSION

- Authors add heuristics to make SLAM robust.
  - Viewing angle heuristic.
  - Determining permanent landmarks
- Improve tracking of 3D position of key-points using Kalman Filtering.



# DISCUSSION

- "This is the bootstrapping problem where two models both require the other to be known before either can proceed. In this case, they present the interplay between localization over time and obtaining a map of the environment." - Ian
- "As noted by the author, the processing time is mainly used by SIFT feature extraction. Probably in such environment, we don't need to use such a strong feature as the original SIFT feature. We could reduce the bins or cells to make it quicker." -Gang
  - I think finding the interest points are most time-consuming. SIFT descriptor extraction is much fast, and we can even learn a quick approximation mapping for that. - Jianchao
  - Processing time for SIFT feature extraction is always considerable. However I guess maintaining a database containing SIFT landmarks before robot navigation could be really useful. Even if you think in disaster management scenarios, first responders use maps to get into a building. So why not facilitating navigation, by precompiling a database of SIFT databases for critical infrastructures? - Mani

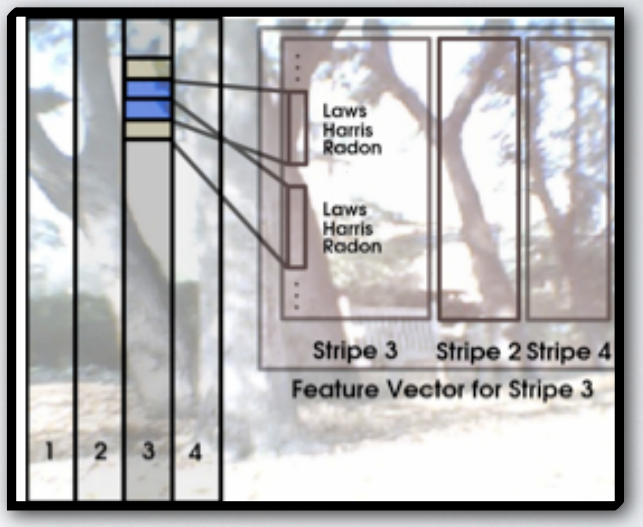
# High Speed Obstacle Avoidance Using Monocular Vision and Reinforcement Learning

- Goal: Control a remote control car driven at high-speeds through an unstructured environment. Visual features are used as inputs to the control algorithm (RL).
- Novelty:
  - Use monocular visual cues to figure out object depth.
    - Similar ideas as Make3d (Saxena et al. 2008).
  - Use computer graphics to generate copious amounts of data to train algorithm on.
- Stereo gave them poorer results
  - Limited range
  - Holes in the scene with no depth estimates
- "I am just wondering why researchers are interested in recovering the 3D information from a single image. Even human cannot estimate the depth well using single eye. Maybe use more calibrated cameras with SIFT mapping? " - Jianchao

# POSSIBLE MONOCULAR CUES

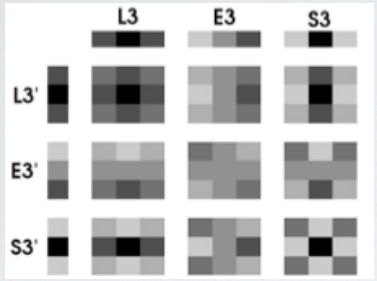
- Texture & Color (Gini & Marchi 2002, Shao et al. 1988)
- Texture gradient
- Linear Perspective
- Occlusion
- Haze
- Defocus

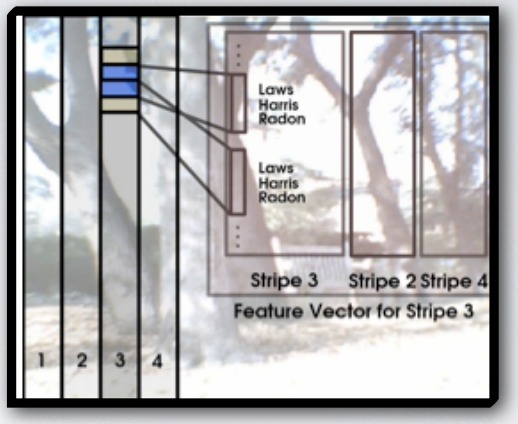
# SYSTEM DESCRIPTION




Linear regression on features to predict depth of nearest obstacle in each stripe.

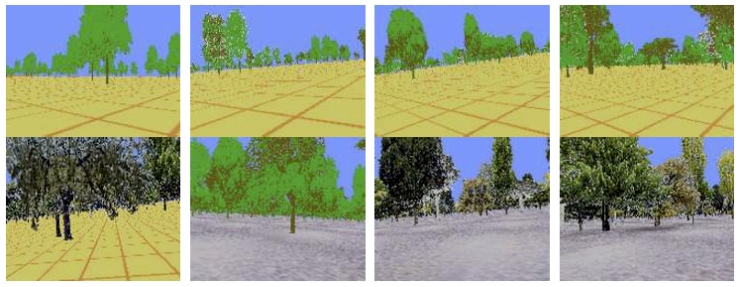
# COMPOSITION OF FEATURE VECTOR




- Divide each scene into 16 stripes
  - Each stripe is labeled with (log) depth of nearest obstacle.
  - Divide each stripe into 11 overlapping windows over which:
    - Texture energies and Texture gradients are computed.
  - Augment each stripe feature with features from adjacent stripes.

# TRAINING



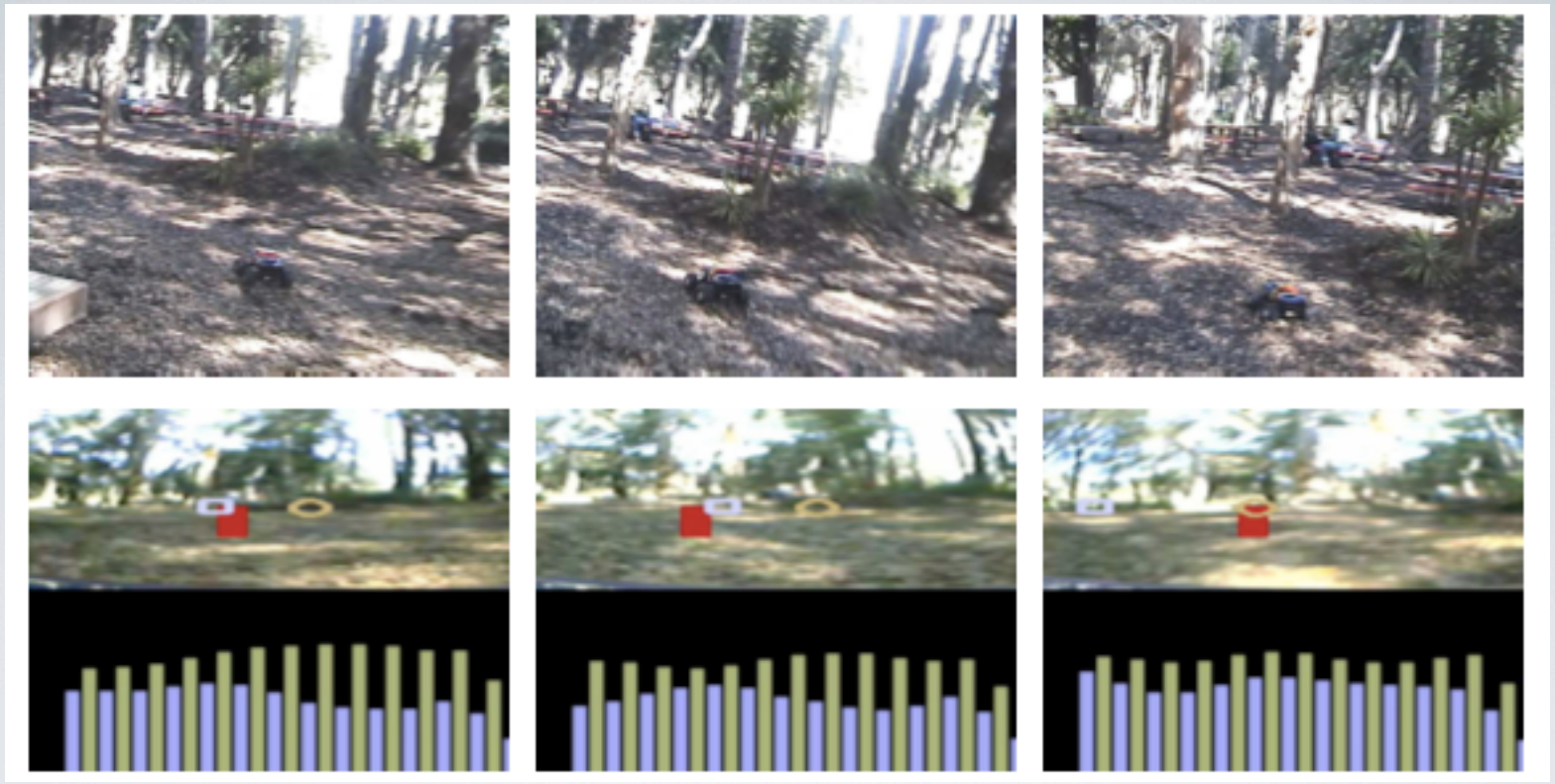
- Synthetic graphics data is used to train the system on a number of plausible environments.
- Real Images + Laser scan data is also used.
- Linear regression/ SVR / Robust Regression models learned

$$w = \arg\min_{w} \sum_{i=1}^{N} \sum_{s=1}^{S} \left( w^T x_{is} - \ln(d_i(s)) \right)^2$$

# CONTROL ALGORITHM (RL)

$$R(s) = -|v_{desired} - v_{actual}| - K \cdot \text{Crashed}$$

$\theta_1$: $\sigma$ of the Gaussian used for spatial smoothing of the predicted distances
$\theta_2$: if $\hat{d}_i(\alpha_{chosen}) < \theta_2$, take evasive action rather than steering towards $\alpha_{chosen}$
$\theta_3$: the maximum change in steering angle at any given time step
$\theta_4$, $\theta_5$: parameters used to choose which direction to turn if no location in the image is a good steering direction (using the current steering direction and the predicted distances of the left-most and right-most stripes of the image).
$\theta_6$: the percent of max throttle to use during an evasive turn



SAMPLE VIDEO

# EXPERIMENTAL RESULTS

| Train | $E_{depth}$ | Rel Depth | $E_\theta$ | Hazard Rate |
|---|---|---|---|---|
| None | .900 | - | 1.36 | 23.8% |
| Laser Best | .604 | .508 | .546 | 2.69% |
| Graphics-8 | .925 | .702 | 1.23 | 15.6% |
| Graphics-7 | .998 | .736 | 1.10 | 12.8% |
| Graphics-6 | .944 | .714 | 1.04 | 14.7% |
| Graphics-5 | .880 | .673 | .984 | 11.0% |
| Graphics-4 | 1.85 | 1.33 | 1.63 | 34.4% |
| Graphics-3 | .900 | .694 | 1.78 | 38.2% |
| Graphics-2 | .927 | .731 | 1.72 | 36.4% |
| Graphics-1 | 1.27 | 1.00 | 1.56 | 30.3% |
| G-5 (L+Harris) | .929 | .713 | 1.11 | 14.5% |

| Feature | $E_{depth}$ | Rel Depth | $E_\theta$ | Hazard Rate |
|---|---|---|---|---|
| None | .900 | - | 1.36 | 23.8% |
| Y (intensity) | .748 | .578 | .792 | 9.58% |
| Laws only | .648 | .527 | .630 | 2.82% |
| Laws | .640 | .520 | .594 | 2.75% |
| Radon | .785 | .617 | .830 | 6.47% |
| Harris | .687 | .553 | .713 | 4.55% |
| Law+Harris | .612 | .508 | .566 | 2.69% |
| Laws+Radon | .626 | .519 | .581 | 2.88% |
| Harris+Radon | .672 | .549 | .649 | 3.20% |
| Law+Har+Rad | .604 | .508 | .546 | 2.69% |

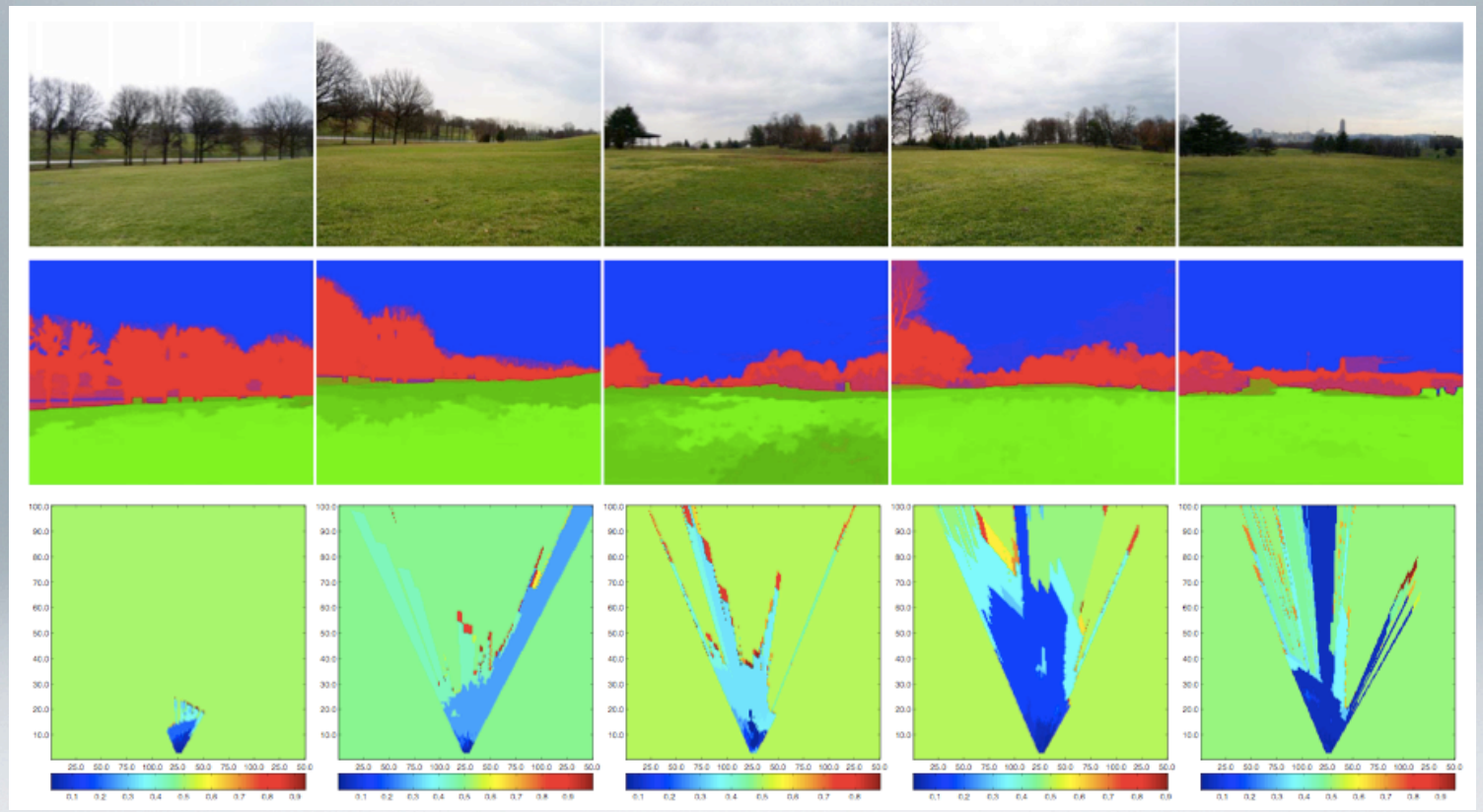
- Texture energy + Texture gradients work better together
- Harris and Radon features gave comparable performance
- Performance improved with increasing complexity of graphics
  - Except when shadows and haze were added

# QUESTIONS & COMMENTS

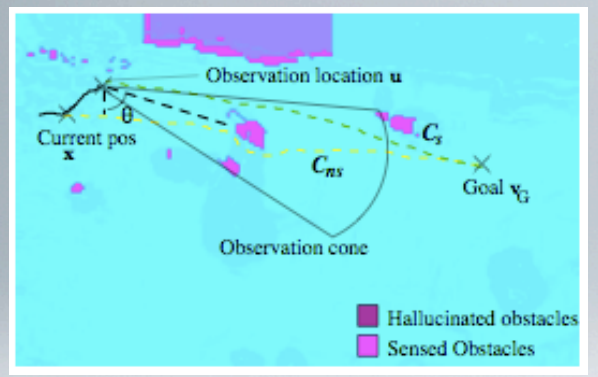
- ~~Why do they use log (depth) here when they regress on depth for the Make3D paper?~~
- Why does a linear predictor on log(depth) work so well using texture features?
- The layout information recovered by Michels et al. (2005) seems like it would be incredibly useful for the map generation required by Se and colleagues (2001). By attaching SIFT features to approximated 3D locations, map creation can begin before ego-motion estimation. - Eamon

# Opportunistic Use of Vision to Push Back the Path Planning Horizon

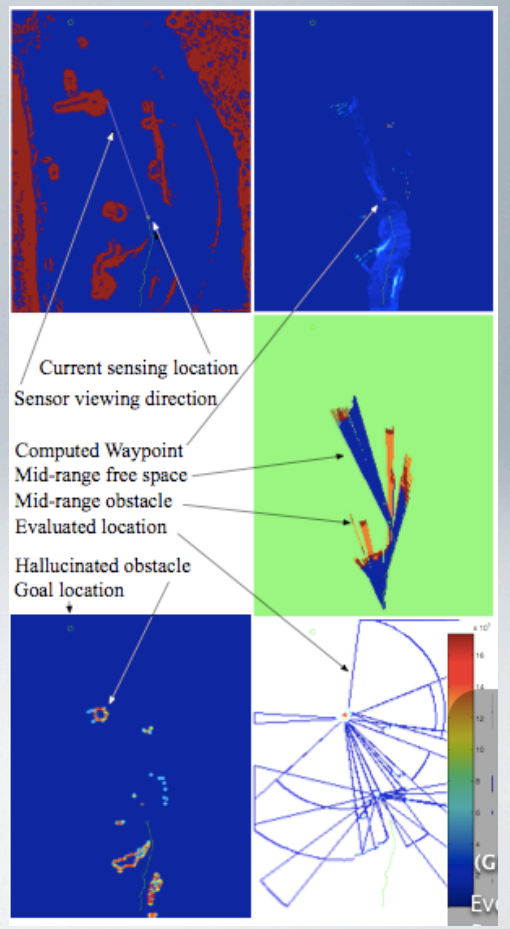
- Goal: Overcome myopic planning effect by early detection of faraway obstacles and determination of navigable terrain.
  - Use rough geometric estimates of environment from monocular data for path-planning.
- Applications: Outdoor robotic navigation



# PLANNING VIEWPOINTS

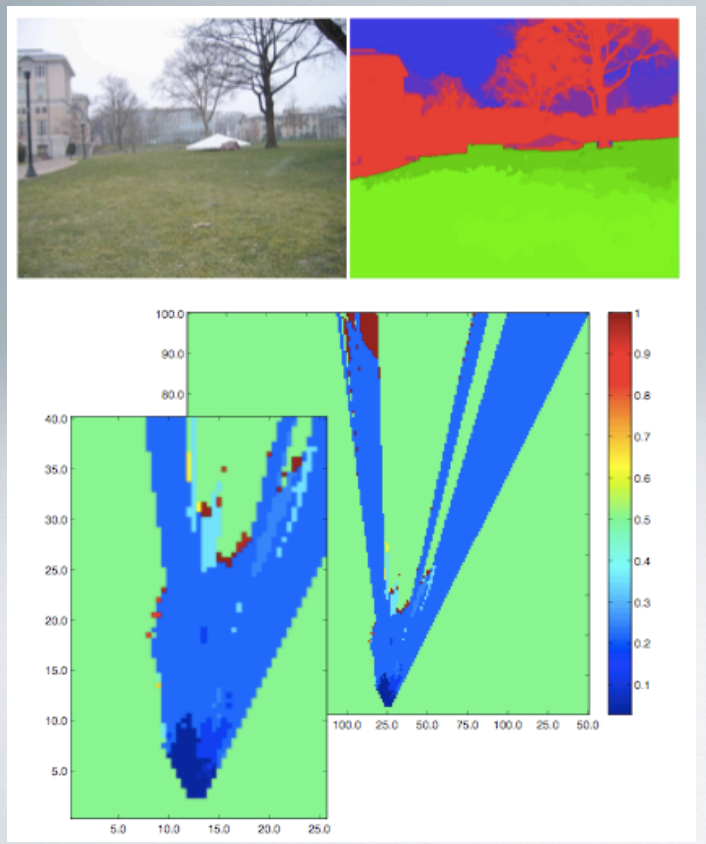


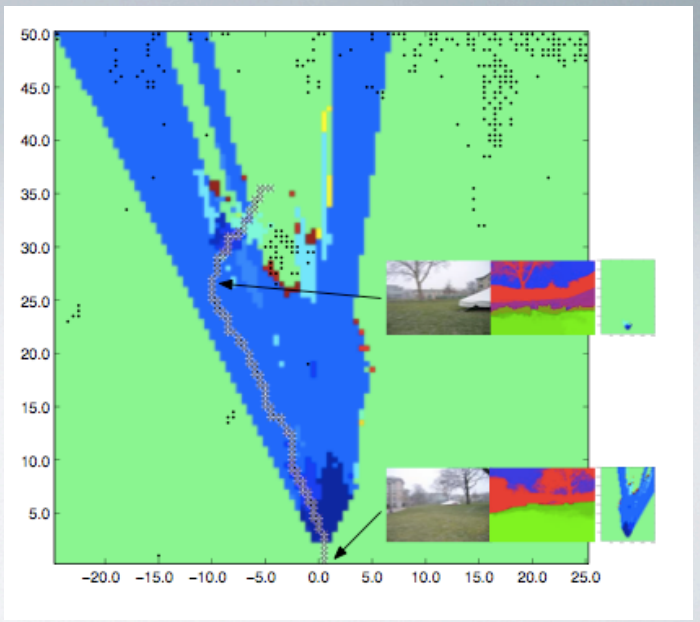

$$\psi_{\mathbf{u}}(\mathbf{x}, \mathbf{v}_G) = 1 - max_\theta \left( \sum_j \frac{P(L_j)(C_{ns}(L_j,\mathbf{x},\mathbf{v}_G) - C_s(L_j,\mathbf{x},\mathbf{u},\theta,\mathbf{v}_G))}{C_{ns}(L_j,\mathbf{x},\mathbf{v}_G)} \right)$$



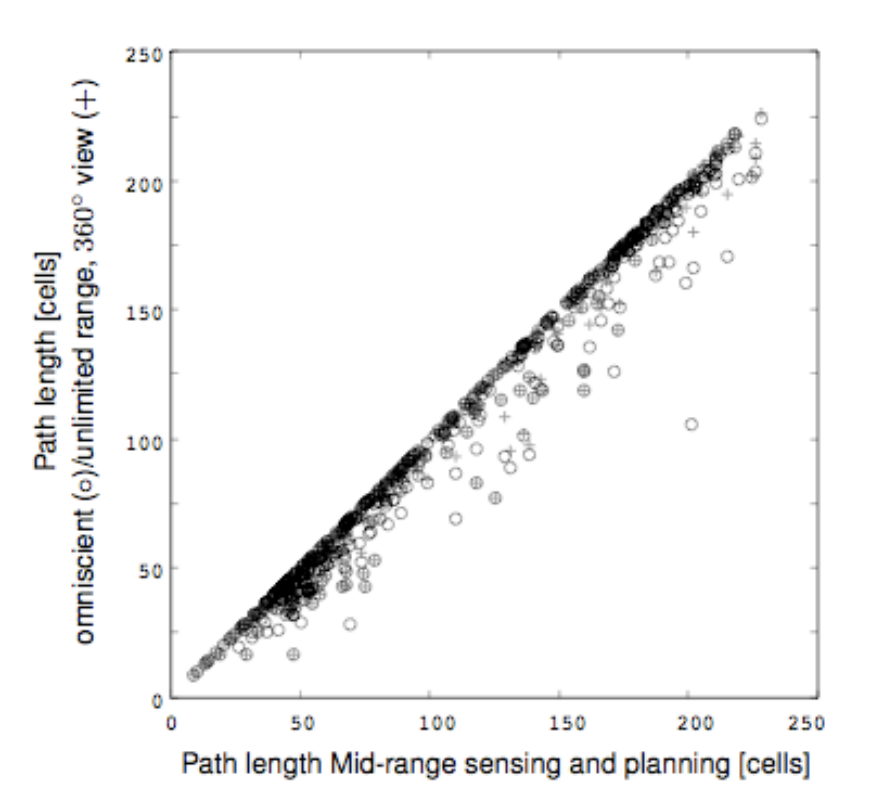
# SAMPLE RUN

ATRV-JR + Firewire Camera
+SICK Laser Range finder



# COMPARISON OF DIFFERENT SENSING STRATEGIES



# DISCUSSION

- Knowledge of the ground plane is important (Laser scanning data can be used here)
- Performance should improve with system trained on images the robot is likely to see (e.g. the data used by Michels et al.)
- Training task specific categories (e.g. road vs. rough vs. grass vs. trees) should improve navigation performance.

# FINAL QUESTIONS

- Can stereo be totally ignored? How can stereo cues be integrated to improve planning?
  - Hadsell, et al. (IROS 2008): Train Deep Belief Networks on Image data with stereo supervision.
- Do we have a satisfactory explanation for why linear predictors of depth based on texture features work?
- What are effective strategies for data collection to train these robots?

DONE!