Driving and Learning Strategies for Offroad Robots

Raia Hadsell[1,2], Pierre Sermanet[2], Jan Ben[2], Jefferson Y. Han[1], Ayse Naz Erkan[1]

Sumit Chopra[1], Yury Sulsky[1], Beat Flepp[2], Urs Muller[2], Yann LeCun[1]

(1). Courant Institute of Mathematical Sciences, New York University
(2). Net-Scale Technologies, Morganville, NJ 07751

The Robot

- Platform and baseline software developed by CMU/NREC
- DARPA program: LAGR (Learning Applied to Ground Robots)
- Sensors
  - 4 color cameras in 2 stereo pairs (passive vision only: no LADAR)
  - short-range (1m) infrared sensor
  - GPS receiver
  - Bumper switches
- Computing
  - 4 identical computers with 2.0 Ghz Pentium M processors and 1 GB RAM

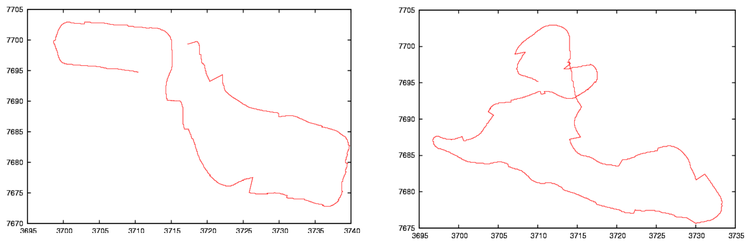
The Task

- Fully autonomous navigation through any terrain; learned adaptability

Driving

the Problem

- Standard approach to robot navigation:
  1) build a global map
  2) use global planner to find route
  3) issue driving commands based on route.
- Problem: Path planning is critically impaired by the modules it depends on.
- Sources of error:
  - Positioning errors caused by GPS jumps and drift cause global map errors.
  - Delays from global planner due to large maps can cause disaster.

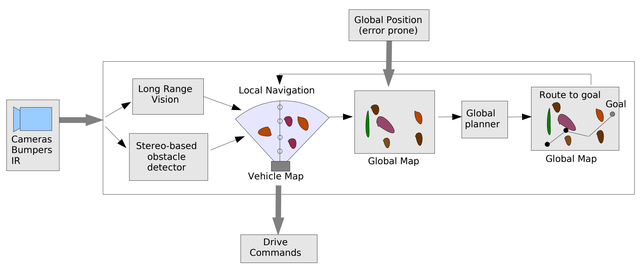


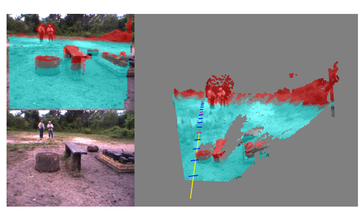
Errors in position estimates: The robot was driven in 2 closed loops, and the XY position given by the robot was plotted. Note the GPS jumps: sharp jags in the loops. Note the overall drift: the loops do not close.

the Solution

Overview: Trusting our eyes

- Separate local, tactical driving from global strategic driving.
  - **Short-range, tactical driving** is done in a local, vehicle-centered map, so it cannot be harmed by positioning errors.
  - **Global, strategic planning** is done in the global map, by a fast path planner similar to A-Star.
- Polar coordinate map is used for local map
  - cell size increases naturally with distance, azimuthal angle is precise.



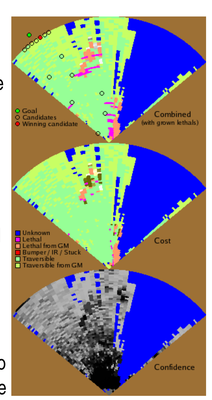


Building a Local, Vehicle-centered Map

1) Robustly fit a ground plane to the disparity point cloud.
2) Fill a polar-coordinate histogram with disparity points:
   - points above the ground plane are obstacle points
   - points on the ground are ground points
3) Determine traversability of each histogram cell
4) Translate polar map to a cartesian vehicle map

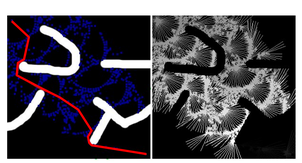
The Vehicle Map

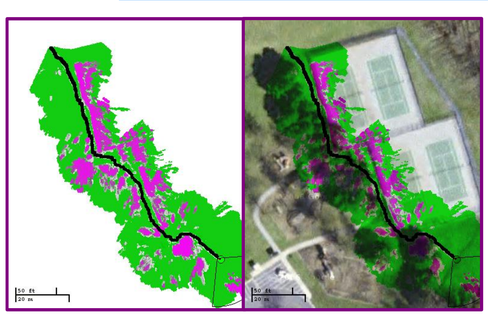
- The vehicle map combines stereo information, long-range vision information, and confidences into a final cost map that is used for local driving.
- Candidate waypoints are found in the vehicle map, and one is chosen based on the current global route. Driving commands towards this waypoint are issued.
- The vehicle map is copied into the global map on every frame



Path Planning in the Global Map

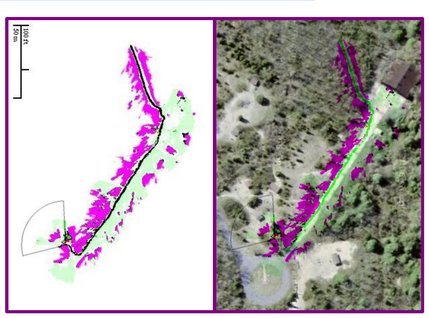
- A-Star is optimal, but very slow due to large "open list".
- RayStar uses rays, not points, and keeps few waypoints- much faster (but not optimal).





Global Map Examples

- Robot's path is black line
- Pink = Lethal (non-traversable with high confidence)
- Green = Traversable
- Course at left was lawn, trees, and fences around tennis court
- Course at right was a dirt path through thick woods (top) followed by a wider concrete path through scrub and tall grass.



Learning

the Problem

- The standard approach for vision-based obstacle detection:
  - stereo-matching algorithm produces *point cloud* of disparity values
  - derive traversability map from point cloud using heuristics
- Problem: stereo range is limited to 10 to 12 meters and grows sparse and inaccurate with distance.
- Robots that navigate using stereo are limited:
  - driving into dead ends
  - missing distant paths
  - driving in a "fog"

- Humans can easily identify paths and obstacles without stereo vision, from monocular images (see examples below)

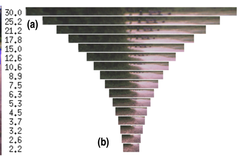
the Solution

- Online Learning for Environmental Adaptability
- Long-Range Obstacle Detection (LROD)
  - Basic approach: *Near-to-Far* or *bootstrap* learning
    1) Reliable module (stereo) labels close-range points
    2) Classifier is trained with these labeled points
    3) Trained classifier is used to classify long-range points
  - Limited! – sparse stereo labels, generalization to far-range difficult because of scaling with distance
- Innovations in LROD
  - Training on large, context-rich windows in the image is better than simple color/texture patches, but generalization is very hard because of scaling.
  - We build a distance-normalized image pyramid to solve scaling problem
  - Label propagation: we use a spatially-indexed quad-tree to identify spatial concurrences between different views of same point, thus expanding training set.
  - Ring buffer holds samples from previous frames; short-term memory that acts as regularization

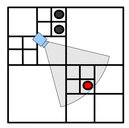
Distance-Normalized Image Pyramid

- Basic idea: build a pyramid of sub-images such that similar objects at different distances are the same size.
- Extract sub-images around imaginary lines on the ground (given by ground plane estimate)
- Resize sub-images to be 12 pixels high and variable width
- Bands in pyramid are from 1 to 30 meters, in a geometric progression

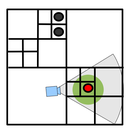
yellow line is base line; blue box is extraction window
left: baseline is at 21m, *right:* baseline is at 2m



the pyramid, with rows (a) and (b) corresponding to sub-images at left.



Object is inserted into quadtree without a label at time t



Query extracts object at time t+1 and assigns label

Spatial Label Propagation

- Expand number and variety of labeled points per frame through semi-supervised label propagation.
- Propagate labels backward in time
- Insert all unlabeled points in a spatially-indexed quadtree
- For each labeled point, query tree and extract all points at the same XYZ world location.
- Label new points with label from querying point
- Train on all points

Ring Buffer = Short term memory

- All training points are put into a fixed-size ring buffer
- Thus training points from previous frames can be trained on again, giving greater robustness and consistency

Training the Classifier On-the-Fly

- Classifier is a logistic regression on a fixed RBF layer
- Radial Basis Function centers are trained with unsupervised learning (K-means)

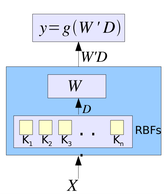
RBF centers:

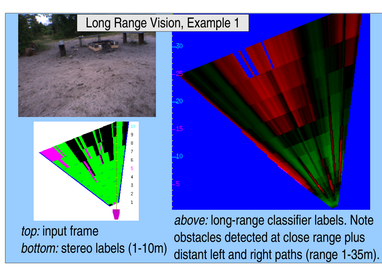
- Output of RBF layer is a vector *D*. For input window *X* and RBFs $[K_1 \dots K_n]$

$$D = \left[\exp\left(-\beta_1 \|X - K_1\|^2\right) \dots \exp\left(-\beta_n \|X - K_n\|^2\right)\right]$$

Loss function: $L = -\sum_{i=0}^{n} \log(g(y_i \cdot W' D_i))$

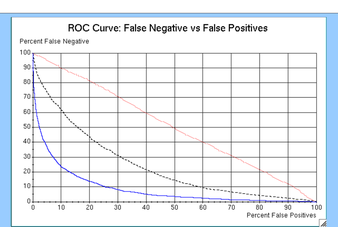
- Gradient of the loss: $\frac{\partial L_i}{\partial W} = -(y_i - g(W' D_i))$
- where $g(z) = \frac{1}{1+e^{-z}}$



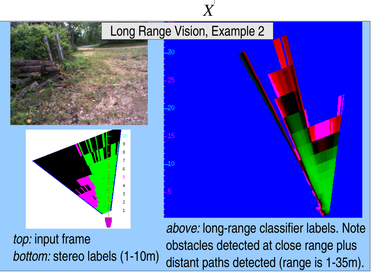


top: input frame
bottom: stereo labels (1-10m)

above: long-range classifier labels. Note obstacles detected at close range plus distant left and right paths (range 1-35m).



red: error rate with random classifier, *black:* trained weights, no online learning
blue: error rate with trained initial weights + online learning



top: input frame
bottom: stereo labels (1-10m)

above: long-range classifier labels. Note obstacles detected at close range plus distant paths detected (range is 1-35m).