Online Learning for Offroad Robots: Using Spatial Label Propagation to Learn Long-Range Traversability

Raia Hadsell¹, Pierre Sermanet¹,², Jan Ben², Ayse Naz Erkan¹, Jefferson Han¹, Beat Flepp², Urs Muller², Yann LeCun¹

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

- Vision-based Navigation for Mobile Robots
  - Structured indoor environments
  - Unstructured indoor environments
  - Structured outdoor environments (road-following)
  - **Unstructured outdoor environments**
    - goal vs. non-goal driven
Problem Definition

Vision-based Navigation for Mobile Robots

- **Unstructured outdoor environments**

Primary tasks (for goal-driven systems):

- Obstacle avoidance
- Landmark detection
- Path planning
- Position estimation
- Map building
Motivation: stereo shortcomings

- Vision-based navigation often done using **stereo**
  - Find **disparities** between 2 calibrated camera images
  - Ground/obstacle points identified using estimated ground plane

- Stereo maps are short-range, noisy, and prone to error (tall grass)
- Navigating in fog
Motivation: Beyond Stereo

Beyond stereo: **Long-range obstacle detection**
- Obstacles, cul-de-sacs, dead ends
- promising paths
- manmade structures
- collapsible vs. non-collapsible vegetation

**Robustness**
- adaptation to new terrain
- memory of old terrain
Previous Work

**Supervised Learning** applied to Autonomous Navigation

- Learning steering angles directly from images:
  - **ALVINN** [Pomerlau, Robot Learning, 1993]; **MANIAC** [Jochem et al., IROS, 1995]
  - **DAVE** [LeCun et al., NIPS, 2005]
  - [Gaussier et al., IROS 1997], [Jones et al., IROS 1997]

- Learning obstacles from images:
  - **NEURO-NAV** [Meng and Kak, ICRA 1993]
  - [Manduchi et al., Autonomous Robot 2003]
  - [Huertas et al., Workshop of Applications of Computer Vision 2005]
  - **Demo III** [Hong et al., Aeroscience Conference 2002]
  - [Hong et al., ICRA 2002]
  - [Rasmussen, ICRA 2002]
Previous Work

**Self-Supervised Learning** applied to Autonomous Navigation

- **Near-to-Far** Learning: A reliable (but limited scope) module provides labels to train another module (with wider scope).
  - LADAR module ---> satellite image pixels (traversability)
    [Sofman et al. Improving robot navigation through self-supervised online learning. RSS 2006.]
  - Wheel data ---> LADAR features (terrain roughness)
    [Stavens, Thrun. A self-supervised terrain roughness estimator for offroad autonomous driving. UAI 2006.]
  - Wheel data ---> LADAR features (load-bearing surface)
    [Wellington, Stentz. Online adaptive rough-terrain navigation in vegetation. ICRA 2004.]
  - Vehicle location ---> color camera patches (road detection)
    [Dahlkamp et al. Self-supervised monocular road detection in desert terrain. RSS, June 2006.]
  - Bumper hits, wheel current --> color camera features (traversability)
    [Kim et al. Traversibility classification using unsupervised on-line visual learning for outdoor robot navigation. ICRA 2006]
LAGR Robotic Platform

**LAGR** (Learning Applied to Ground Robots)

- DARPA program 2005-2008
- 8 competing research labs develop navigation software for single platform
- Periodic testing in unfamiliar terrain
- CMU/NREC designed platform and baseline software

**Platform:**

- 4 color cameras (2 stereo pairs, 640x480)
- GPS receiver
- 2 front bumper switches
- Onboard IMU (inertial measurement unit)
- 4 onboard 1.2Ghz computers
LAGR Robotic Platform

Cameras
Bumpers
IR

Long Range Vision
Stereo-based obstacle detector

Vehicle Map

Local Navigation

Global Position or POSE (error prone)

Global Map

Drive Commands

Route to goal

Global Map

Goal
LAGR Robotic Platform

- Cameras
- Bumpers
- IR

Local Navigation
- Long Range Vision
- Stereo-based obstacle detector

Vehicle Map

Global Position or POSE (error prone)

Global Map

Global planner

Route to goal

Global Map

Drive Commands

Goal
Learning Framework

Paradigm: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

**Paradigm**: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

Paradigm: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

**Paradigm:** Near-to-Far Self-Supervised Learning

- **Inputs:** large windows from image
- **Labels:** Stereo module
- **Classifier:** convolutional neural network (feature extraction) + online logistic regression
Learning Framework

Paradigm: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

Paradigm: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

**Paradigm:** Near-to-Far Self-Supervised Learning

- **Inputs:** large windows from image
- **Labels:** Stereo module
- **Classifier:** convolutional neural network (feature extraction) + online logistic regression
Learning Framework

**Paradigm**: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

Paradigm: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

Paradigm: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

Paradigm: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

**Paradigm**: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Learning Framework

**Paradigm**: Near-to-Far Self-Supervised Learning

- **Inputs**: large windows from image
- **Labels**: Stereo module
- **Classifier**: convolutional neural network (feature extraction) + online logistic regression
Control Loop Overview

I. Pre-Processing
II. Labeling and Feature Extraction
III. Label Propagation
IV. Online Training and Classification
Control Loop: Preprocessing

I. Pre-Processing

1) Image rectification and stereo algorithm -> image + point cloud
2) Ground plane estimation -> image + point cloud + plane equation
3) Convert to YUV, normalization -> YUV image + point cloud + plane eq.
4) Horizon leveling and distance/scale normalization -> image pyramid
Control Loop: Preprocessing

I. Pre-Processing

1) Image rectification and stereo algorithm -> image + point cloud
2) Ground plane estimation -> image + point cloud + plane equation
3) Convert to YUV, normalization -> YUV image + point cloud + plane eq.
4) Horizon leveling and distance/scale normalization -> image pyramid

\[ P = [p_0, p_1, p_2, p_3] \]
Control Loop: Preprocessing

I. Pre-Processing

1) Image rectification and stereo algorithm -> image + point cloud
2) Ground plane estimation -> image + point cloud + plane equation
3) Convert to YUV, normalization -> YUV image + point cloud + plane
4) Horizon leveling and distance/scale normalization -> image pyramid

\[ P = [p_0, p_1, p_2, p_3] \]
Control Loop: Preprocessing

I. Pre-Processing

1) Image rectification and stereo algorithm -> image + point cloud
2) Ground plane estimation -> image + point cloud + plane equation
3) Convert to YUV, normalization -> YUV image + point cloud + plane eq.
4) Horizon leveling, distance/scale normalization -> image pyramid

\[ P = [p_0, p_1, p_2, p_3] \]
Preprocessing

Distance-normalized image pyramid

- Better learning with large windows rather than simple patches-context, transitions, feet
- But, aspect varies severely, makes generalization from near to far impossible
- Solution: normalize so that height of an object is independent of distance from camera.
Preprocessing

Distance-normalized image pyramid

- 20 bands from 1 to 35 meters
- Uniform height (16 pixels), variable width

(a). sub-image extracted from far range. (21.2 m from robot).
(b). sub-image extracted at close range. (2.2 m from robot).
(c). the pyramid, with rows (a) and (b) corresponding to sub-images at left.
Control Loop Overview

I. Pre-Processing

II. Labeling and Feature Extraction

III. Label Propagation

IV. Online Training and Classification
II. Labeling and Feature Extraction

1) Histogram of stereo point cloud; heuristics to determine cost.  
   \[ \text{-> training labels} \]

2) Feature extraction using convolutional neural network  
   \[ \text{-> pyramid of feature vectors} \]

\[
\begin{align*}
F_W(X)^{1..p}, Y^{1..p}
\end{align*}
\]

samples | labels

Stereo-based obstacle detector

Convolutional neural network (2 layers) $F_W$
Feature Extraction

Convolutional Neural Network:
- Output of convolutional network: feature vector (120 components)
- Trained offline using 150 diverse logfiles (1.2 million samples)
- 48 7x6 filters (first layer), 5x3 filters (second layer), 80 outputs
- Low-level features are pooled
- Learns highly discriminative features
- Naturally shift and scale invariant
Convolutional Neural Network:

- YUV image band
- 16 pixels tall

80 features per 3x16x11 input window

Logistic regression 80 features -> 3 classes

Convolutions with 7x6 kernels
- 6x10x6 input window

Pooling/subsampling with 2x2 kernels
- 6x5x3 input window

Convolutions with 5x3 kernels
- 80x1x1 input window

80 features per 3x16x11 input window
Control Loop Overview

I. Pre-Processing

II. Labeling and Feature Extraction

III. Label Propagation

IV. Online Training and Classification
Control Loop: Training Set

III. Label Propagation

1) Insert current samples (feature vectors) into QuadTree
2) Query QuadTree for concurrent samples
3) Compute probabilistic labels

-> “soft” labeled pyramid

Convolutional neural network (2 layers)

Stereo-based obstacle detector
Label Propagation with spatially indexed Quad-tree

- **Time t**: $X^i$ has coords (x,y) and label $?$
- **Time t**: Add $X^i$ to quad-tree at position (x,y)
- **Time t+n**: Stereo gives label $Y^i$ to coords (x,y)
- **Time t+n**: Extract $X^i$ and train with label $Y^i$
View-Invariant Training Samples

- Query at coords (x,y) yields samples $X^i$ and (possibly) labels $Y^i$: 
  $\{(X^1,Y^1),(X^2,Y^2),\ldots,(X^m,Y^m)\}$

- Compute probabilistic, **soft** label across $Y$:
  
  $$P_{ob} = \frac{|Y^i = ob|}{|Y^i = ob| + |Y^i = gr|} \quad \quad P_{gr} = \frac{|Y^i = gr|}{|Y^i = ob| + |Y^i = gr|}$$

- And propagate over all samples: $\{(X^1,Y^s),(X^2,Y^s),\ldots,(X^m,Y^s)\}$
View-Invariant Training Samples

- Query at coords \( (x,y) \) yields samples \( X^i \) and (possibly) labels \( Y^i \):

\[
\{(X^1, Y^1), (X^2, Y^2), \ldots (X^m, Y^m)\}
\]

- Compute probabilistic, **soft** label across \( Y \):

\[
P_{ob} = \frac{|Y^i = ob|}{|Y^i = ob| + |Y^i = gr|}
\]

\[
P_{gr} = \frac{|Y^i = gr|}{|Y^i = ob| + |Y^i = gr|}
\]

- And propagate over all samples: \( \{(X^1, Y^s), (X^2, Y^s), \ldots (X^m, Y^s)\} \)

\[
P = 0.8
\]

\[
P = 0.8
\]

\[
P = 0.8
\]
Control Loop Overview

I. Pre-Processing
II. Labeling and Feature Extraction
III. Label Propagation
IV. Online Training and Classification
Online Learning

**Classifier:**
- trained online on each frame using gradient descent
- Output of convolutional network: feature vector D (120 components)
- Weights W are a logistic regression
- Regularization: decay to default weights, L2 regularization

\[ y = g(W' D) \]

\[ D = F_W(X) \]

\[ X \text{ (yuv: 16x11x3)} \]

**Loss:**
\[ L = -\sum_{i=1}^{n} \log(g(y \cdot W' D)) - \alpha R(W) \]

**Learning:**
\[ \frac{\partial L}{\partial W} = y \cdot g(-y \cdot W' D) D \]

**Inference:**
\[ y = g(W' D) \]

\[ g(z) = \frac{1}{1 + e^{-z}} \]
Online Learning Comparison

- **Red**: uninitialized weights
- **Black**: default weights, no online learning
- **Blue**: default weights, online learning

ROC Curve: False Negative vs False Positives
LAGR
long-range vision
Results
LAGR
long-range
vision
Results
Failure mode: saturated area is misclassified

Failure mode: bad ground plane estimate

Failure mode: poor short term memory leads to erratic classification
In Summary

Long Range Vision on LAGR platform:

- Near-to-Far **online** learning
- we use **context-rich image windows**, not color histograms
- Distance-normalized image pyramid
- Spatial label propagation
- **Convolutional network** for feature extraction
- Runs at 4-5 hz
- Smooth, accurate maps **5 to 35 meters**

Caveats

- Range estimates can be very poor (ground plane estimation)
- Glare, close obstacles, tall grass/scrub (poor stereo labels)
- Desperate need for visual odometry to correct positioning errors