Augmenting Cartographic Resources for Autonomous Driving

Young-Woo Seo
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
ywseo@ri.cmu.edu

Chris Urmson
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
curmson@ri.cmu.edu

David Wettergreen
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
dsw@ri.cmu.edu

Jin-Woo Lee
Electrical and Controls
Integration Lab
R&D General Motors
30500 Mound Rd
Warren, MI 48090
jin-woo.lee@gm.com

ABSTRACT

In this paper we present algorithms for automatically generating a road network description from aerial imagery. The road network inforamtion (RNI) produced by our algorithm includes a composite topoloigical and spatial representation of the roads visible in an aerial image. We generate this data for use by autonomous vehicles operating on-road in urban environments. This information is used by the vehicles to both route plan and determine appropriate tactical behaviors. RNI can provide important contextual cues that influence driving behaviors, such as the curvature of the road ahead, the location of traffic signals, or pedestrian dense areas. The value of RNI was demonstrated compellingly in the DARPA Urban Challenge [1], where the vehicles relied on this information to drive quickly, safely and efficiently.

The current best methods for generating RNI are manual, labor intensive and error prone. Automation of this process could thus provide an important capability. As a step toward this goal, we present algorithms that automatically build the skeleton of drivable regions in a parking lot from a single orthoimage. As a first step in extracting structure, our algorithm detects the parking spots visible in an image. It then combines this information with the detected parking lot boundary and information from other detected road-markings to extract a skeleton of the the drivable regions within the lot.

[1]The Urban Challenge (or the DARPA Urban Challenge) was an autonomous vehicle competition in which competitors had to build vehicles capable of auotnomously driving 60 miles amongst moving traffic in an urban environment. Visit http://www.darpa.mil/grandchallenge for more information.

Categories and Subject Descriptors

H.4 [**Computer Vision Applications in GIS**]: Spatial Information Acquisition, Image and Video Understanding

1. INTRODUCTION

In general, for reliable autonomous driving, a robotic vehicle must model both the parts of the environment that move and the parts that don't. In this paper, we focus on providing a strong prior for the former by developing algorithms to extract a description of the road network a vehicle will operate on. We term this information Road Netowrk Information (RNI). RNI provides a topological representation of geospatial elements (e.g., where intersections and road lanes are and how they are connected). This information can be used to inform a vehicle where it can drive, model what can be expected, and provide contextual cues that influence driving behaviors. This information is static and may be provided a priori.

In contrast, moving obstacle information is dynamic (e.g., the location and speed of other vehicles) and obviously cannot be generated a priori. Onboard sensors such as laser range finders and vision sensors are used to generate this information in realtime [18]. Even so, the algorithms used to generate these models can exploit available road network information to help interpret noisy or sporadic sensor data, and focus attention on relevant portions of the sensor fields of view.

The value of RNI was demonstrated during the Urban Challenge competition [24, 25]. Figure 1 shows a small set of road network information describing the starting chutes of the Urban Challenge site. This information was used by our vehicle to anticipate upcoming intersections and other fixed rules of the road. In particular, RNI informs that the speed limit is 30 miles per hour and that the first intersection (labeled as "I14135") after leaving the starting chute is a yield-intersection, which allows our vehicle to anticipate the

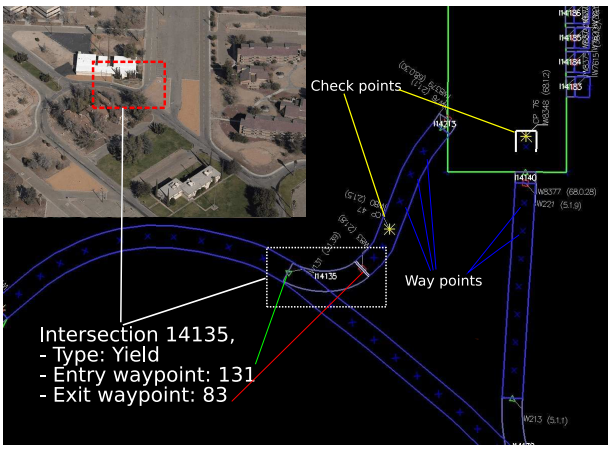


Figure 1: An example of the road network information used in the Urban Challenge. RNI is an internal representation of a robotic vehicle's driving environment and represented topologically. In this example it is comprised of a set of vertices (i.e., waypoints marked by blue "x" and checkpoints marked by yellow "*") and their connections. An intersection is a region that includes a subset of waypoints between entry and exit points [18]. Viewed best in color.

need to execute a yielding behavior. Without prior knowledge of the road geometry and associated required behaviors, achieving the level of performance demonstrated during the challenge would be very difficult.

Currently road network descriptions are generated manually using a combination of GPS survey and aerial imagery. These techniques for converting digital imagery into road network information are labor intensive, reducing the potential benefit provided by the broad availability of digital aerial imagery. To fully exploit the benefits of digital imagery, these processes should be automated.

As a step toward this goal, we present a set of aerial image analysis algorithms that extract the structure of parking lot and build the skeleton of their drivable regions. The extraction of parking lot skeletons are motivated by our local motion planner which heavily relies on structures (e.g., road-lanes) of urban environments to generate a smooth path to its goal. Because of this dependence, for unstructured regions such as parking lots, our planner has to superimpose virtual structures onto parking lots and evaluate all the feasible paths over this structure [10]. This two step process (create virtual structure and apply nominal motion planning algorithms) is computationally very expensive, and could be avoided if the parking lot structure were known in advance.

Our algorithm to extract the structure of parking lots consists of a hierarchical approach of generating and filtering candidate parking spot hypotheses. To minimize human intervention in the use of aerial imagery, we devise a self-supervised learning algorithm that automatically generates a set of parking spot templates to learn the appearance of a parking lot and estimates the structure of the parking lot from the learned model. The low-level layer, which extracts and compiles geometrical meta-information for easy-to-find parking spots, is highly accurate and serves as a prime source of examples for self-supervised training. The high-level layer uses outputs from the low-level layer to predict plausible candidate hypotheses for more difficult parking spot locations and then filters some of erroneous hypotheses using self-trained learners. The result of parking spot detection is then combined with a parking lot boundary detector and other detected lane markings to extract the skeleton of drivable regions with in the parking lot. Our method is described in detail in Section 3.

2. RELATED WORK

The GIS community has a broad focus including building maps for human consumption in various applications and contexts; theoretical studies of road network structure for developing faster algorithms for the handling geospatial data [8], extracting connectivity of roads from raster maps [4], localizing moving objects on the known road networks [27], and many other topics. Among these, research on raster map analysis shares the most commonality with our approach in that it involves extracting interesting features from a raster images of various maps [3, 4, 13]. For example, in [4], the authors present image processing algorithms that obtain locations and orientations of road intersections from raster images and estimate the connectivity of the region under investigation. They utilize combinations of mathematical morphology operators (e.g., thining and thickening) and heuristics. The distinction between the raster image analysis and ours lies in the characteristics of images. Extracting data from rasterized maps has a different challenges (e.g., aliasing or distored color vs. variations in illumination and appearance, occlusions) than analyzing overhead aerial imagery. In orthophoto analysis, Chen and his colleagues propose a conflation algorithm that integrates two different geospatial data such as vectorized road map and orthoimages [2]. Their approach is similar to ours in that they use a classification algorithm (i.e., a naive Bayes classifier) to estimate boundaries of roads and generate and filter out hypotheses on interesting points for the conflation. But their filtering is guided by information in vectorized road maps whereas ours is based on learning of distributions on parking spot appearance. Although there is a number of research works on extracting road network structures from overhead aerial imagery, to the best of our knowledge, our work is the first attempt that builds the the map of a parking lot for autonomous driving.

Overhead aerial images have been utilized to provide prior information about environments for outdoor robot navigation. Despite being potentially out of date, aerial image analysis provides an important source of information which enables robots to plan globally to achieve their goals. In combinations with other onboard sensors such as vision sensors and laser range finders, aerial images have been used in the generation of long-range paths [20, 26], localization [6], maintenance of robots' world model [17, 25], mapping [16], and prediction of terrain traversability [21]. While overhead imagery has been used as a complement to other onboard sensors' outputs to guide outdoor robot navigations, to the best of our knowledge, there is no work for automatically generating road network information from overhead aerial images.

There are two similar works in the realm of parking structure analysis. Wang and Hanson propose an algorithm that uses multiple aerial images to extract the structure of a parking lot for simulation and visualization of parking lot activities [28]. Multiple images from different angles are used

to build a 2.5 dimensional elevation map of the parking lot. This usage of multiple images makes it difficult to generalize their method because it is not easy to obtain such images for the same geographic location from publicly available imagery. Dolgov and Thrun devise algorithms that builds a lane-network graph of a parking lot from sensor readings [7]. They first build a grid map of static obstacles from range measurements about a parking lot and then use a Markov Random Fields (MRFs) to infer a topological graph that most likely fits the grid map. They define a series of potentials to incorporate their prior on a road network. However, instead of directly minimizing these potentials imposed on road segments, a Voronoi diagram is used as a subset of the topological road network. This work is the most similar work to ours in that they building a road network for robotic vehicles, but different in that they need to drive the robot to collect range measurements, which are the sole input to their mapping algorithm.

Most prior work in parking lot image analysis [9, 11, 29] focused primarily on detecting empty parking spots in surveillance footage when the overall geometrical structure of the parking lot is known. Our work addresses the more general problem of extracting the entire parking lot structure from overhead imagery. A similarity between our work and these works on empty parking spot detection lies in the fact that we utilize coherent structural patterns over entire image region.

The field of self-supervised learning has recently attracting attention from the robot learning community since it requires no (or substantially less) human involvement for carrying out learning tasks. This framework is highly desirable for robot learning because it is usually hard to collect large quantities of high-quality human-labeled data from any real world robotic application domain.

Self-supervised learning frameworks typically utilize the most precise data source to label other data sources that are complementary, but unlabeled. For example, conventional laser range finders provide accurate distance estimates between the robot and surrounding objects, but have limited range. Sofman and his colleagues use local range estimates as self-labeled examples to learn relations between the characteristics of local terrain and corresponding regions in aerial images [21]. These learned relations are used to map aerial images to long range estimates of traversability over regions that a robot is going to explore. Similarly, Stavens and Thrun utilize laser range measurements to predict terrain roughness [22]. They first analyze the associations between inertial data and laser readings on the same terrain and use the learned rules to predict possible high shock areas of upcoming terrains. Lieb and his colleagues devised a self-supervised approach to road following that analyzes image characteristics of previously traversed roads and extracts templates for detecting boundaries of upcoming roads [15]. In our algorithms, the low-level analysis phase extracts lines forming parking lot lane markings, resulting in a collection of canonical parking spot image patches which can be used as training examples. We additionally use these initial parking spots to guide a random selection of negative examples, parking lot boundary segmentation, and road-marking detection.

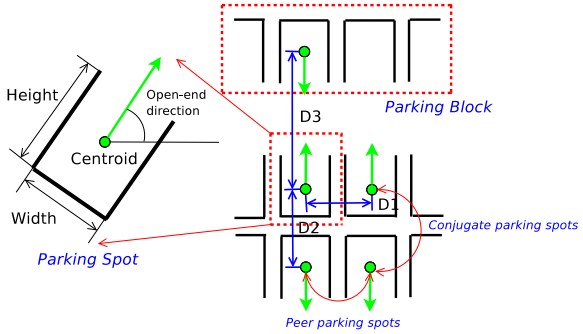


Figure 2: A model of parking lot is illustrated.

3. AUGMENTING CARTOGRAPHIC RESOURCES FOR AUTONOMOUS DRIVING

In this section, we describe how the skeleton of a parking lot is automatically built from an overhead image. We formulate parking lot structure analysis as parking spot detection because the structure may be easily recovered if the image coordinates of all the visible parking spots are identified. Section 3.1 details the parking spot detection.

We define the skeleton of a parking lot as a depiction of the drivable regions in the parking lot. To accurately build a skeleton, the boundary of a parking lot with an image should be identified since the image may also contain non-parking-lot regions. Once the boundary is identified, drivable regions can be determined by superimposing the parking lot structure over the estimated parking lot boundary. A normalized safety of traversability value is assigned to drivable regions to ensure reliable autonomous driving. Section 3.2 describes this skeletonization process in detail.

3.1 Parking Spot Detection

Figure 2 illustrates how a parking lot is represented in this paper. Our algorithm parameterizes each individual *parking spot* by its height, width, orientation, and centroid location in image coordinates. We define a *parking block* as a row of parking spots of which open-end directions are the same. Each parking block is characterized by the distance between neighboring parking spots in the block (i.e., "D1" in figure 2). Parking blocks are related to each other by two distance measures: the distance between conjugate parking spots (i.e., "D2") and the distance between blocks (i.e., "D3" in the figure 2).

If the image coordinates of all visible parking spots are known, it would be trivial to estimate parameters shown in the figure 2. However, in practice we must estimate these parameters from the given image to determine the parking lot structure. In what follows, we describe in detail our hierarchical approach to detecting parking spots. Section 3.1.1 presents the multiple image processing steps involved in the low-level image analysis layer. This layer accurately extracts a set of easily found parking spots from the image. Section 3.1.2 details the high-level processing layer which then extrapolates and interpolates the spots found by the low-level analysis to hypothesize the locations of the remaining parking spots. We then discuss our self-supervised hypothesis filtering approach, which removes erroneous hypotheses from

the collection.

3.1.1 Low-Level Analysis: Detecting Canonical Parking Spots

Geometrical and image characteristics differ between parking lots. Most overhead aerial parking lot images contain a number of well-illuminated empty parking spots. Our low-level analysis extracts these easy-to-find spots to be used by the high-level analysis as "seeds" for additional hypothesis generation and by the final filtering stage as canonical self-supervised training examples to adapt the filter to this particular image. The low-level layer carries out multiple image processing steps: line extraction, line clustering, and (parking) block prediction.

Straight lines are important to understanding the shape of a parking lot. We extract most of the available straight lines using the approach proposed by [12]. The approach computes image derivatives to obtain intensity gradients at each pixel and quantizes the gradient directions using predefined ranges. A connected component algorithm is then used to group pixels assigned the same direction to form line supporting regions. The first principal eigenvector of a line supporting region determines the direction of the line.

Although a majority of extracted lines may align with lane markings of the underlying parking lot, some of them come from other image regions such as road lanes or contours of other adjacent buildings. Since we only need the lines aligned with the line-markings of the parking lot, it is necessary to remove lines that do not belong to parking lot structure. To this end, we first group the extracted lines into clusters based on their orientations and then remove lines that are either too short or too long from each of the line clusters. The remaining lines are used for estimating parameters of a parking block. A line cluster corresponds to (at least) one of the parking blocks. [2] We repeat this process (the removal of some of the extracted lines and estimation of parameters of a parking block) to individual line clusters.

For the parameter estimation, we first estimate the nominal height of parking spot by computing the mode of each line in the selected cluster. We next build a Euclidean distance matrix across all possible line pairs, quantize the distances and compute the mode to obtain the nominal width of parking spots within a lot. Finally, we quantize the orientations of lines and compute the mode again to estimate the orientation of parking spots' open-end.

The completion of these image processing steps results in generating few, but highly accurate initial estimate of true parking spots. Figure 3 shows the rectangular patches around the image locations. Although most of these self-supervised parking spot templates are in fact true parking spots, some of them are not since the line analysis algorithm is imperfect. To filter out these incorrect self-supervised parking spot templates, we train a SVM with parking spot examples, which are previously obtained [19], and conduct a binary classification.

This low-level analysis is then extended to additionally identify entire parking blocks. We project the centroids of all the initial parking spots onto a virtual line whose orientation is the mean of the initial parking spots' orientation. This projection returns distances of centroids from the origin, $\rho_i = c_{i,x}\cos(\theta_i) + c_{i,y}\sin(\theta_i)$, where $c_{i,x}$ and $c_{i,y}$ are image coordinates of parking spot centroid and θ_i is the open-end orientation of the ith initial parking spot (See [19] for the details). After projection, boundaries between parking blocks are clearly visible and the distance between peer parking spots (i.e. $D1$ in the Figure 2) is used to determine boundaries between parking blocks. From the discovered parking blocks, we finish the parameter estimation by computing three distances between parking blocks (i.e. $D1$, $D2$, and $D3$ in the Figure 2).

[2] For most of the testing images used in this paper, this is true because individual images have a large portion or a whole part of a parking lot. However, when a non-parking lot image is given, our parameter estimation based on line detection might not work.

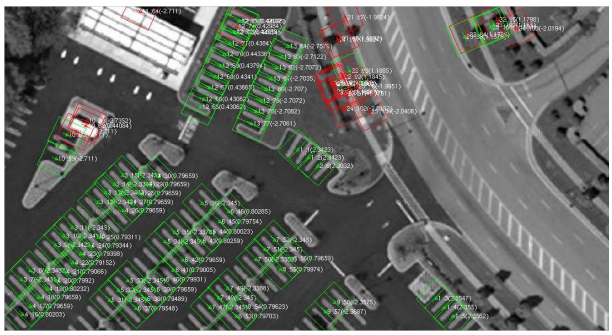


Figure 3: An illustrative example image is shown. The low-level analysis produces a set of self-supervised parking spots that are depicted by rectangular patches around their centroids. After filtering out some of the patches (i.e., red patches), the remaining patches (i.e., green patches) are used as positive example to train our hypothesis filters. Viewed best in color.

3.1.2 High-Level Analysis: Interpolation, Extrapolation, Block Prediction, and Filtering

The high-level layer is intended to detect all the visible parking spots in an image. It first hypothesizes the parking spot locations based on the parameters estimated by the low-level layer. It then filters these hypotheses by classifying the rectangular image patches around these hypotheses using self-supervised classifiers.

Parking Spot Interpolation and Extrapolation A parking spot hypothesis represents an image coordinate that indicates the centroid of a potential parking spot. A rectangular image patch around the hypothesis is evaluated to determine if a local characteristic of the image is similar to that of a true parking spot. To cover the image regions that possibly contain true parking spots, we use the image coordinates of the centroids of each self-supervised parking spot as the starting points in each of the discovered parking blocks. We then generate parking spot hypotheses by selecting image locations through three processes: interpolation, extrapolation, and block prediction. The hypothesis generation aims to cover the image regions, which potentially contain true parking spots, that were initially undiscovered, by using the estimated coordinates of parking spots at each end of an estimated parking block. The interpolation procedure chooses image coordinates between two end parking spots in a parking block, whereas the extrapolation procedure extends hypotheses beyond the two boundary parking

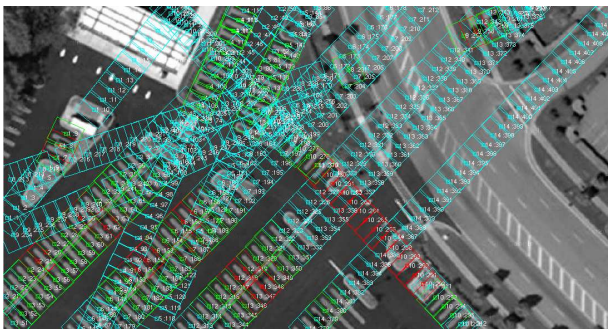


Figure 4: A set of the generated parking spot hypotheses is shown. Parking spot hypotheses are rectangular image patches. Different rectangle colors indicate results of different hypothesis generation processes (red patches by the interpolation, cyan ones by extrapolation, and green ones by the low-level analysis). In this example image, there are 114 true parking spots and 411 parking spot hypotheses. Viewed best in color.

spots. The estimated parking spot width is used as the spatial interval between parking spot hypotheses. Finally, block prediction aims at discovering any missing parking blocks. We use the estimated distances between parking blocks and select image regions to test for the existences of the missing parking blocks.

Self-supervised Hypothesis Filtering The hypothesis generation process produces n parking spot hypotheses represented by the corresponding number of rectangular image patches, $\mathbf{g}_1, ..., \mathbf{g}_n$. Figure 4 shows a complete set of the generated parking spot hypotheses where individual parking spot hypotheses are represented as rectangles. Each parking spot hypothesis is evaluated to determine if it is a true parking spot. We formulate this decision problem as binary classification for assigning a label, $y_i \in \{-1, +1\}$, to a given patch vector, $\mathbf{g}_i$, where $\mathbf{g}_i$ is an m (= height $\times$ width)-dimensional column vector. Because raw intensity values of a grayscale image patch might not be consistent even in the same class, we use three different pieces of information to inject invariance into our parking spot patch representation: intensity statistics (such as mean, variance, smoothness, skewness, uniformity, and entropy); responses of the Radon transform; local histograms of oriented gradients (HOG) [5]. We compare the performance of hypothesis filters trained by patches represented by these information and ones in raw-intensity patches in the section 3.1.3.

Our experiments compare four machine learning techniques as hypothesis filters for this binary classification task: Support Vector Machines (SVMs), Eigenspots, Markov Random Fields (MRFs), and Bayesian Linear Regression (BLR).

Support Vector Machines SVMs are the *de facto* supervised learning algorithm for binary classification. They seek to find the hyperplane that is maximizing a notion of margin between each class [1]. Linear SVMs are fast, have publicly available implementations, and handle high-dimensional feature spaces well.

Eigenspots Since processing these high-dimensional image patches is computationally expensive, we reduce the dimensionality of our vector space by using principal component analysis (PCA) [1] to find the principal subspace of the self-supervised parking spots obtained by the low-level analysis; we retain the top $k \ll m$ dimensions of the original vector space. In homage to Turk and Pentland [23], we call the eigenvectors of the parking spot space extracted by this method the "Eigenspots" of the space.

We use this new space in two ways. Our first technique simply measures the distance from a candidate patch to the center of the space (i.e. the mean canonical parking spot, $\mathbf{\Psi}$). Given a new image patch $\mathbf{g}$, we compute, $T(\mathbf{g}) = \|\mathbf{D}^{-1/2}\mathbf{E}^T(\mathbf{g} - \mathbf{\Psi})\|^2$ where $\mathbf{\Psi} = \frac{1}{\text{number of positives}} \sum_i \mathbf{g_i}$, $\mathbf{D}$ is a diagonal matrix containing eigenvalues $\lambda_1, ...\lambda_k$, and $\mathbf{E}$ is a matrix whose columns are the eigenvectors of the covariance matrix used in the PCA computation. $T(\mathbf{g})$ is also known as the Mahalanobis distance [1] from the origin of the Eigenspot space. If this distance is less than a threshold, we classify the new image patch as a parking spot. Our second usage simply pushes the examples through the PCA transformation before training a SVM classifier and learning a mixture of multivariate Gaussian distributions. Specifically, we transform each example as $\widetilde{\mathbf{g}} = \mathbf{D}^{-1/2}\mathbf{E}^T(\mathbf{g} - \mathbf{\Psi})$.

Pairwise Markov Random Fields. Because SVMs and Eigenspots only consider the local characteristics of an image patch to perform the binary classification, their performances are limited by the distribution of the training data. Thus it is useful to investigate neighboring image patches around the patch of interest as well as to look at the local characteristics of the image patch. An image patch is highly likely a parking spot when the majority of neighboring patches are parking spots, even the local characteristics of the patch is unlikely classified as a parking spot.

To implement this idea, we use a pairwise Markov Random Fields (MRFs) [14]. A pairwise MRF, $\mathcal{H}$, is an undirected graphical model that factorizes the underlying joint probability distribution $P(Y, G)$ by a set of pairwise cliques. [3] $\mathcal{H}$ is comprised of a set of nodes and their edges where a node models a random variable and an edge between nodes represents dependence between them.

In this work, there are two different types of nodes: observed and unobserved nodes. An observed node corresponds to an image patch whereas an unobserved node is the true label of the observed node. Although we observe the value of a node ($G_k = \mathbf{g}_k$), the true label of the node ($Y_k = y_k \in \{-1, +1\}$) is not observed. The task is then to compute the most likely values of Y (i.e. whether a hypothesis ($\mathbf{g}_i$) is parking spot ($y_i = 1$) or not) given the structure of the undirected graph, $\mathcal{H}$, and characteristics of image patches, G. The joint probability distribution is factorized as

$$P(Y, G) = \frac{1}{Z} \prod_{i=1}^{N} \Phi(G_i, Y_i) \prod_{j \in N(i)} \Psi(Y_i, Y_j)$$

where $\Phi(G_i, Y_i)$ is a node potential, $\Psi(Y_i, Y_j)$ is an edge potential, Z is the partition function that ensures a probability density of this model, $N(i)$ is the set of nodes in the neighborhood of the ith node. Our implementation of MRFs considers first-order neighbors.

As we assume that candidate parking spots are generated from a mixture of multivariate Gaussian distributions, we estimate the node potentials using a Gaussian Mixture model (GMM) [1]. Due to the possibility of two class labels,

[3] There may be bigger cliques in the graph, but the pairwise MRF only consider pairwise cliques.

each node has two potentials: a potential being a parking spot, $\Phi(G_i, Y_{j=+1})$ and the other potential being not a parking spot, $\Phi(G_i, Y_{j=-1})$. The edge potential is computed by Potts model [14].

$$\Psi(Y_i, Y_j) = \psi(Y_i, Y_j) = \exp\left\{-\beta(Y_i - Y_j)^2\right\}$$

where β is a penalty factor for label disagreement between nodes. In particular, if $\beta = 0$, edge potentials are identical regardless of the label disagreement and only node potentials are used. On the contrary, if $\beta = \infty$, only the edge potentials are meaningful and the node potentials are ignored. For inferencing the most likely labels of individual parking spot hypotheses in a given aerial image, we use loopy belief propagation because it is easy to implement [30].

Bayesian Linear Regression Our self-supervised canonical parking spots are highly accurate, but their number is often too few to generalize over unseen image patches. To remedy this insufficient number of positive examples, we use canonical parking spots previously obtained from other aerial images. As we will show its benefit in the experimental results, this certainly helps our hypothesis filters improve their performances. However, naively consuming all the available data might result in a solution overfitted to a given data. To effectively utilize data, we employ a Bayesian linear regression (BLR). BLR provides a theoretical way of incorporating previously obtained parking spot templates as a prior information for the optimal weight vector learning. The optimal weight vector, $\mathbf{w}^*$, is obtained by

$$\begin{aligned} p(\mathbf{w}^*|\mathbf{G}) &\propto \arg\max_{\mathbf{w}} p(\mathbf{G}|\mathbf{w})p(\mathbf{w}) \\ p(\mathbf{G}|\mathbf{w}) &= \prod_{i=1}^{N} p((\mathbf{g}_i, y_i)|\mathbf{w}) \propto \exp\left\{\frac{1}{2\sigma^2}\sum_i (y_i - \mathbf{w}^T\mathbf{g}_i)^2\right\} \\ p(\mathbf{w}) &\propto \exp\left\{-\frac{1}{2}\mathbf{w}\mathbf{\Sigma}^{-1}\mathbf{w} + \mu^T\mathbf{\Sigma}^{-1}\mathbf{w}\right\} \end{aligned}$$

where $p(\mathbf{G}|\mathbf{w})$ is the likelihood function and $p(\mathbf{w})$ is the prior distribution that is a zero-mean Gaussian. The final form of BLR is a regularized linear regression where the parameters of the resulting conditional Gaussian distribution of $\mathbf{w}^*$ given data D is

$$\begin{aligned} \mathbf{\Sigma}_{\mathbf{w}|\mathbf{D}} &= \left(\mathbf{G}\mathbf{G}^T + \lambda\mathbf{I}\right)^{-1} \\ \mu_{\mathbf{w}|\mathbf{D}} &= \left(\mathbf{G}\mathbf{G}^T + (\sigma^2\lambda)\mathbf{I}\right)^{-1}\mathbf{Y}\mathbf{G} \end{aligned}$$

where λ is a regularizing term that controls contributions of the weight prior. We classify an image patch as positive if the regression value is greater than the predefined threshold,

$$h(\mathbf{g}_i) = 2I\left[y(\mathbf{g}_i) \geq \delta\right] - 1, \delta \in \mathbb{R}.$$

where $y(\mathbf{g}_i) = \mathbf{g}_i^T\mathbf{w}^*$ is the output of BLR and $I\left[y(\mathbf{g}_i) \geq \delta\right]$ is an indicator function that returns 1 if $y(\mathbf{g}_i)$ is greater than δ, otherwise 0.

3.1.3 Experimental Results

The goal of our task is to extract the structure of a parking lot that is visible in an aerial image. The knowledge of the image coordinates of parking spots facilitates estimation of parameters that describe the structure of the parking lot. Thus the purpose of our experiments is to verify how well our filtering methods perform in detecting all the visible parking spots in an aerial image.

We use twenty aerial images collected from *Google* [4] map service. There are on average 116 visible parking spots in each individual image in different shapes and under different illumination conditions and a total of 2,324 parking spots across all aerial images.

	fn	*fp*	*acc*
Self-supervised Parking Spots	0.5512	0.0471	0.7008
Generated Hypotheses	0.3719	0.9382	0.2311

Table 1: Performance comparison of parking spot hypotheses generated by the low-level and high-level analysis layers is measured by three different performance metrics. These metrics include "false negative (*fn*)," "false positive (*fp*)," and "accuracy (*acc*)."

Table 1 shows the micro-averaged performance of the hypothesis generation by the low-level and the high-level analysis where the accuracy is defined as a ratio of the number of correctly classified parking spots to the total number of parking spots used in evaluation. This micro-averaged performance is computed by merging contingency tables across the twenty different images and then using the merged table to compute performance measures. Since the self-supervised examples are highly accurate (a low false positive rate (4.71%)), their parking spot templates are used as positive examples for training all filtering methods. An equal number of negative examples are randomly generated.

A false positive is a non-parking-spot example that is classified as a parking spot. A false positive output is quite risky for autonomous robot driving; in the worst case, a false positive output might make a robotic vehicle drive somewhere that the robot should not drive. Despite having nearly zero false positives, the self-supervised parking spots recover only 43.55% of the true parking spots (1,012 out of 2,324 true parking spots over 20 images.) This high false negative rate [5] may cause problems for autonomous driving: for example, an autonomous robotic vehicle might not be able to park itself even if there are plenty of parking spots available. By using information provided by the low-level analysis, the high-level hypothesis generation analysis reduces the false negative rate from 55.12% to 37.19%. However, it increases the false positive rate to 93.82% as well. The filtering stage then corrects this shift in false positive rate by removing erroneous hypotheses. Importantly, as we will see in the results, this technique cannot recover from false negatives in the hypothesis generation.

Table 2 compares the performance of self-trained filtering methods. The parking spot hypotheses generated by the high-level layer were labeled by hand for the evaluation. Hyper-parameters of SVMs were determined by 10-fold cross validation.[6] Eigenspots are computed only using positive examples. For the MRF inference, we build a mesh from the estimated layout of parking spot hypotheses where a node in the grid corresponds to an image patch. We again use positive and negative examples to obtain GMM and use the obtained GMM to estimate node potentials. We observe the

[4] http://map.google.com

[5] A false negative is a parking-spot example that is classified as a non-parking-spot example.

[6] For SVM implementation, we use libsvm which is publicly available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

	false negative	*false positive*	*accuracy*
SVMs	0.3880 ± 0.0012 (0.0188)	0.3136 ± 0.0106 (-0.0230)	0.6627 ± 0.0012 (0.0073)
Eigenspots	0.3074 (0.0090)	0.8004 (0.1085)	0.3013 (-0.0880)
SVMs w/ Eigenspots	0.3826 ± 0.0221 (-0.0116)	0.3227 ± 0.0201 (-0.0256)	0.6603 ± 0.0109 (0.0200)
MRFs w/ GMM	0.3929 ± 0.0301 (-0.0147)	0.3644 ± 0.0041 (-0.0098)	0.6280 ± 0.0074 (0.0101)
BLR	0.3270 ± 0.0009 (0.0184)	0.6611 ± 0.0129 (-0.1007)	0.4091 ± 0.0070 (0.1238)
SVMs	0.4271 ± 0.0350 (-0.0186)	**0.0429 ± 0.0112 (-0.0086)**	0.9189 ± 0.0012 (0.0095)
Eigenspots	**0.2765 (0.0000)**	0.3969 (0.0196)	0.6151 (-0.0176)
SVMs w/ Eigenspots	0.4320 ± 0.0111 (-0.1142)	0.0450 ± 0.0276 (-0.0143)	0.9165 ± 0.0012 (0.0242)
MRFs w/ GMM	0.3466 ± 0.0786 (-0.1846)	0.0798 ± 0.0145 (0.0110)	0.8937 ± 0.0243 (0.0085)
BLR	0.4136 ± 0.0313 (-0.0099)	0.2827 ± 0.0241 (0.0151)	0.7043 ± 0.0232 (-0.0121)
SVMs	0.3951 ± 0.0345 (0.0113)	0.0457 ± 0.0012 (-0.0105)	**0.9213 ± 0.0111 (0.0085)**
Eigenspots	0.2765 (0.0000)	0.3759 (0.0144)	0.6335 (-0.0130)
SVM w/ Eigenspots	0.3880 ± 0.0011 (-0.0567)	0.0486 ± 0.0012 (-0.0165)	0.9194 ± 0.0042 (0.0204)
MRFs w/ GMM	0.3342 ± 0.0188 (-0.1318)	0.0817 ± 0.0012 (0.0126)	0.8945 ± 0.0174 (0.0011)
BLR	0.3970 ± 0.0114 (-0.0105)	0.2712 ± 0.0005 (0.0098)	0.7169 ± 0.0011 (-0.0079)

Table 2: Results comparing different filtering methods.

results by varying β in the range 0 to 10 with steps of size 2.[7] We empirically set 2 as β for MRFs, 5 as λ and .5 as a threshold for binary classification for BLR implementations.

In the table 2, there are three blocks of rows describing three different experimental scenarios. In the first scenario, we trained the filtering methods using a parking spot templates from the image under analysis consisting of the self-supervised parking templates as positive examples and randomly generated negative examples. In the second scenario, we trained these methods using self-supervised examples from all other images not including the target image. Finally, in the last scenario we trained the methods using self-supervised examples from all images. The randomly generated negative examples were sampled while running each of these scenarios. Due to this randomness in negative examples, we averaged our results over 5 separate runs for each scenario. Each cell in the table displays the mean and standard deviation.

In addition, we wanted to measure the usefulness of our feature representation over raw-intensity parking spot patches. We re-ran the above experiments using the same parking patches in raw-intensity values. The numbers in parentheses indicates the performance difference between different parking spot patch representations. Positive values in the accuracy indicate improvements of our feature representation over raw-intensity whereas negative values in false positive and false negative columns indicate improvements. Overally, the performance difference is negligible, but our feature representation method enables our algorithms to reduce the dimension (m) of parking spot patches' from 240 to 93, resulting in computationally more efficient solution (i.e., faster training with less memory).

Ideally, the method with the lowest false positive and negative rates would be the best, but in practice it is hard to achieve both of them simultaneously. For our autonomous driving application, we prefer the method with the lowest false positive to one with lowest false negative because a false positive is more risky than a false negative. In general, the performances of hypothesis filters are improved as the amount of training data is increased. Linear SVMs performed surprisingly well, particularly in terms of false positives and accuracy. Additionally, training an SVM using the subspace generated by the Eigenspots analysis performs only marginally better than simply using the Eigenspot distant measure computation. This performance difference can potentially be decreased by statistically fitting the threshold value used during distance measure classification. As discussed in Section 3.1.2, MRFs utilize higher-level interactions to improve prediction accuracy. However, estimating the GMM requires a substantial amount of data; the performance degradation in the first row of the table indicates that the canonical parking spots extracted by the low-level analysis alone were too few to accurately this fit this model.

[7]We fit our Gaussian Mixture model using the publicly available GMMBayes from http://www.it.lut.fi/project/gmmbayes/

3.2 Skeletonization

In this paper, *skeletonization* refers to a process of extracting the framework of drivable regions in a parking lot image. To accurately build a skeleton, we need to know the structure of a parking lot. This is done by estimating boundaries of parking blocks that are obtained from parking spot detection. In parallel, we segment a given aerial image into two regions: "parking lot" and "non-parking lot" regions. Then the drivable regions in a parking lot are recovered by superimposing the constructed structure over the segmented parking lot image. In this step, we use self-supervised examples to find cues for parking lot for the boundary segmentation and road-marking classification.

The following sections detail how self-supervised examples are used in segmenting the parking lot boundary and in detecting road-markings.

3.2.1 Parking Lot Boundary Segmentation

The floodfill algorithm is a painting method that fills connected regions within an image with a constant value. We assume that the magnitude of image gradient in drivable regions is similar to those of parking spots. After computing magnitudes of the image gradient, we randomly select some of the self-supervised parking spots and use them to obtain a threshold value. The centroids of those selected parking spots are used as starting points. Despite its simplicity, our modified floodfill algorithm works reasonably well in that it detects all the visible parking lot regions in our test images.

Figure 5(a) shows a binary image of the segmented drivable region where autonomous vehicle can only drive on white image regions.

3.2.2 Road-Markings Detection

Road-markings are important parts of drivable regions and differentiated from other drivable regions by average intensity and their color histograms. To detect road-markings in a given parking lot image, we train a binary road-marking classifier that assigns a pixel with one of the binary decisions: road-marking (+1) or non-road-marking (−1). To obtain a training set, we utilize road-markings that are parts of the self-supervised parking spot templates. A set of the randomly selected road-markings are used to learn characteristics of road-marking in a particular parking lot. We use Bresenham's line algorithm to select pixels along the selected lines and learn a multivariate Gaussian distribution of two different color spaces: Hue-Saturation-Intensity (HSI) and RGB, in which individual pixels are represented by six-dimensional vectors, $\mathbf{x}_i \in \mathbb{R}^6$. [8]

$$p(\mathbf{x}_i|C_k) = \frac{1}{(2\pi)^{d/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mu_k)^T \mathbf{\Sigma}_k^{-1}(\mathbf{x}_i - \mu_k)\right\}$$

where $p(\mathbf{x}_i|C_k)$ is a conditional probability of $\mathbf{x}$ given $C_k, k \in \{-1, 1\}$, d is the dimension of a pixel vector, $\mathbf{\Sigma} = d \times d$ is kth class' covariance matrix, and $\mu = d \times 1$ is kth class' mean vector. We learn another Gaussian distribution for non-road-marking class. The road-marking detection is done by investigating the likelihood ratio between two classes:

$$y(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \log\left(\frac{p(\mathbf{x}_i|C_1)}{p(\mathbf{x}_i|C_{-1})}\right) > 0 \\ -1 & \text{otherwise} \end{cases}$$

A result of the road-marking classification shown in Figure 5(b) which has a number of false positives along road lanes outside of the parking lot. These errors occur because the magnitudes of road lane are similar to those of parking spots. However, since the detection result is used in conjunction with other results (i.e., parking lot boundary segmentation and parking spot detection) to build the skeleton of drivable regions, it is acceptable to include some of the non-parking lot regions.

3.2.3 Drivable Region Identification

There are three inputs for identifying drivable regions: results of parking spot detection, results of road-marking detection, and results of parking lot boundary segmentation. Although none of these inputs is perfect, a combination of these imperfect inputs works reasonably because they are complementary to each other. For example, our road-marking detection method produces a number of false positives on road lanes (See Figure 5(a)). During the drivable region identification phase, these high false-positive regions are disregarded because they are located outside of the parking lot based on the result of parking lot boundary segmentation result (See Figure 5(b)).

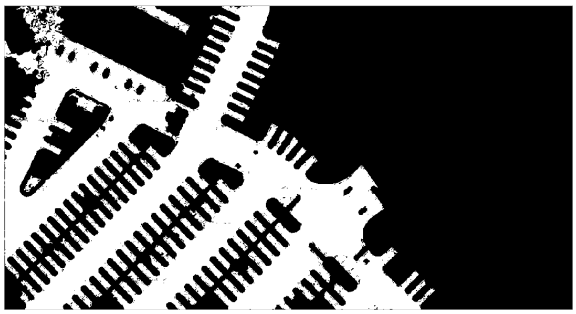
Based on the best result of parking spot detection, the structure of parking lot is uncovered by computing boundaries of parking blocks. This structure can roughly tell us how the geometric shape of a parking lot looks like, but cannot tell where exactly an autonomous vehicle should drive. To define drivable regions of a parking lot, the parking lot structure is superimposed over the segmented parking block boundaries. Then drivable regions of a parking lot become clearly visible to the vehicle. However, the binary image of drivable regions shown in 5(a) still has some errors. Although these black speckles do not look significant in the image, they may cause serious problems when used for autonomous driving as they may be regarded as obstalces. To remove these errors, we apply a mathematical morphology operation (i.e., "close") to smooth the segmentation binary image. Since the smoothing can only remove small-size speckles, we implement an heuristics to remove islands in the drivable regions. These islands are in fact road-markings (e.g., stop-lines, driving direction marking) on the drivable region in the original image. While these features represent important contextual information, we want to remove them from the description of the drivable regions. To remove these islands, we utilize the results of road-marking detection. That is, for each of the islands in drivable regions, it can be removed if an island does not belong to road-markings that are parts of parking blocks. Figure 5(c) shows the binary image of drivable regions after removing speckles and islands. Finally to accurately depict boundaries of drivable regions, we apply a modified "brushfire" algorithm that incrementally propagates distance values from non-drivable regions (e.g., parking blocks). Figure 5(d) shows the final result of the skeletonization that depicts image regions where a robotic vehicle may drive.

4. CONCLUSIONS

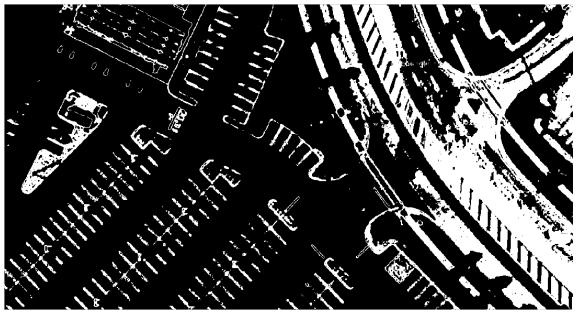
This work proposes orthoimage analysis methods that automatically build a parking lot map for autonomous driving. In a hierarchical scheme, our algorithms analyze the structure and build the skeleton of drivable regions in a parking lot from an orthoimage. For parking spot detection, a low-level analysis layer extracts a set of easily detected canonical parking spots and estimates parking blocks using line detection and clustering techniques. A high-level analysis then extends those spots using geometrical characteristics of typical parking lot structures to interpolate and extrapolate new hypotheses and uses self-supervised machine learning techniques to filter out false positives in the proposed hypotheses. Our experiments show that training the classifiers using a self-supervised set of canonical parking spots extracted by the low-level analysis successfully adapts the filter stage to the particular characteristics of the image under analysis. Self-supervised examples are also effectively utilized to train a road-marking detector and a parking lot boundary segment. A composite of this information is then used to extract the structure of a parking lot.

Future work will complete the automatic building of RNI by first fitting a mesh of roadlanes and by then annotating contextual information (e.g., location of stoplines, speed limits) to the obtained skeleton. Although we have extensively evaluated our filtering methods in various metrics, there are no quantitative performance evaluations on boundary segmentation, road-marking detection, and skeleton building. We will evaluate these methods with more complex parking lot images. We will also continue to work on improving the performance of parking spot detection. Particularly, a high false negative rate might be acceptable in terms of safe-autonomous driving in any parking lots. However, it might cause a serious problem when actually generating the

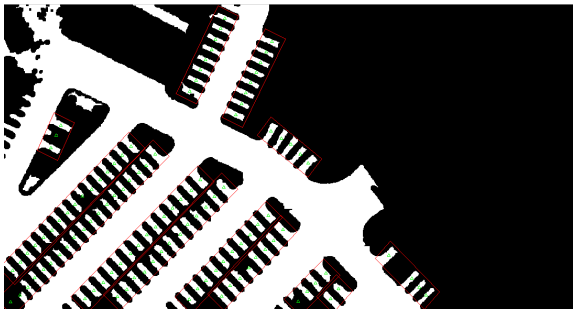
[8] We utilized other color spaces such as Lab and YCbCr and found that a combination of HSI and RGB works best.



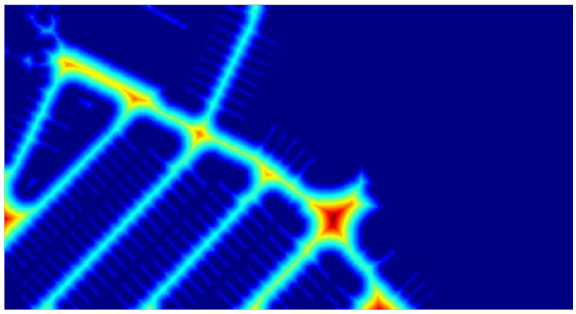
(a) The detected boundary of a parking lot.



(b) The results of the road-marking detection.



(c) The structure of a parking lot is superimposed over parking lot boundary segmentation. The red rectangles represents boundaries of parking blocks whereas the green triangles are detected parking spots. Viewed best in color.



(d) The final results of skeletonization. Safety of traversability is color-scaled for visualization purpose. Red corresponds to highest safety of traversability whereas blue is lowest safety of traversability. Viewed best in color.

Figure 5: The figure at bottom right shows the final result of the skeletonization and all other figures are inputs for the skeletonization process.

skeleton of a parking lot because a high false negative output underestimate the actual area of the parking lot. We will also consider generalizing our methods to more complex parking configurations.

5. ACKNOWLEDGMENTS

This work is funded by the GM-Carnegie Mellon Autonomous Driving Collaborative Research Laboratory (CRL).

6. REFERENCES

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] C.-C. Chen, C. A. Knoblock, and C. Shahabi. Automatically conflating road vector data with orthoimagery. *GeoInformation*, 10:495–530, 2006.

[3] Y. Chen, R. Wang, and J. Qian. Extracting contour lines from common-conditioned topographic maps. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):1048–1057, 2006.

[4] Y.-Y. Chiang and C. A. Knoblock. Automatic extraction of road intersection position, connectivity and orientations from raster maps. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2008.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[6] C. Dogruer, B. Koku, and M. Dolen. Global urban localization of outdoor mobile robots using satellite images. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3927–3932, 2008.

[7] D. Dolgov and S. Thrun. Autonomous driving in semi-structured environments: Mapping and planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3407–3414, 2009.

[8] D. Eppstein and M. T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2008.

[9] T. Fabian. An algorithm for parking lot occupation detection. In *Proceedings of IEEE Computer Information Systems and Industrial Management*

Applications, 2008.

[10] D. Ferguson, T. M. Howard, and M. Likhachev. Motion planning in urban environments: Part ii. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1070–1076, 2008.

[11] C.-C. Huang, S.-J. Wang, Y.-J. Chang, and T. Chen. A bayesian hierarchical detection framework for parking space detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2097–2100, 2008.

[12] P. Kahn, L. Kitchen, and E. Riseman. A fast line finder for vision-guided robot navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1098–1102, 1990.

[13] A. Khotanzad and E. Zink. Contour line and geographic feature extraction from usgs color topological paper maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):18–31, 2003.

[14] S. Z. Li. *Markov Random Fields Modeling in Computer Vision*. Springer-Verlag, 2000.

[15] D. Lieb, A. Lookingbill, and S. Thrun. Adaptive road following using self-supervised learning and reverse optical flow. In *Proceedings of Robotics Science and Systems*, 2005.

[16] M. Persson, T. Duckett, and A. Lilienthal. Improved mapping and image segmentation by using semantic information to link aerial images and ground-level information. In *Recent Progress in Robotics*, 2008.

[17] C. Scrapper, A. Takeuchi, T. Chang, T. Hong, and M. Shneier. Using a priori data for prediction and object recognition in an autonomous mobile vehicle. In *Proceedings of the SPIE Aerosense Conference*, 2003.

[18] Y.-W. Seo and C. Urmson. A perception mechanism for supporting autonomous intersection handling in urban driving. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1830–1835, 2008.

[19] Y.-W. Seo and C. Urmson. A hierarchical image analysis for extracting parking lot structures from aerial image. Technical Report CMU-RI-TR-09-03, Robotics Institute, Carnegie Mellon University, 2009.

[20] D. Silver, B. Sofman, N. Vandapel, J. A. Bagnell, and A. Stentz. Experimental analysis of overhead data processing to support long range navigation. In *Proceedings of IEEE/IRJ International Conference on Intelligent Robots and Systems*, pages 2443–2450, 2006.

[21] B. Sofman, E. Lin, J. A. Bagnell, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learing. In *Proceedings of Robotics Science and Systems*, 2006.

[22] D. Stavens and S. Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proceedings of Conference in Uncertainty in Artificial Intelligence*, 2006.

[23] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[24] C. Urmson, C. Baker, J. Dolan, P. Rybski, and B. Salesky. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 30(2):17–28, 2009.

[25] C. Urmson and et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics: Special Issues on the 2007 DARPA Urban Challenge*, pages 425–466, 2008.

[26] N. Vandapel, R. Donamukkala, and M. Hebert. Experimental results in using aerial ladar data for mobile robot navigation. In *Proceedings of International Conference on Field and Service Robotics*, pages 103–112, 2003.

[27] H. Wang and R. Zimmermann. Snapshot location-based query processing on moving objects in road networks. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2008.

[28] X. Wang and A. R. Hanson. Parking lot analysis and visualization from aerial images. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 36–41, 1998.

[29] Q. Wu, C. Huang, S. yu Wang, W. chen Chiu, and T. Chen. Robust parking space detection considering inter-space correlation. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 659–662, 2007.

[30] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report TR2001-022, Mitsubishi Electric Research Laboratories, 2002.