A Perception Pipeline for Expeditionary Autonomous Ground Vehicles

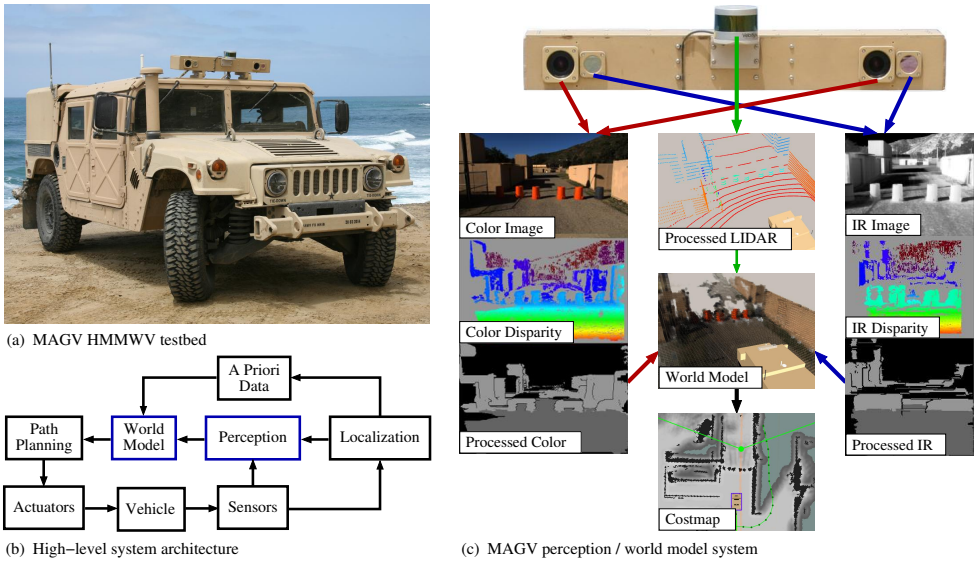
Josh Zapf[a], Gaurav Ahuja[a], Jeremie Papon[b], Daren Lee[b], Jeremy Nash[b], and Arturo Rankin[b]

[a]Space and Naval Warfare Systems Center, Pacific
[b]Jet Propulsion Laboratory, California Institute of Technology

ABSTRACT

Expeditionary environments create special challenges for perception systems in autonomous ground vehicles. To address these challenges, a perception pipeline has been developed that fuses data from multiple sensors (color, thermal, LIDAR) with different sensing modalities and spatial resolutions. The paper begins with in-depth discussion of the multi-sensor calibration procedure. It then follows the flow of data through the perception pipeline, detailing the process by which the sensor data is combined in the world model representation. Topics of interest include stereo filtering, stereo and LIDAR ground segmentation, pixel classification, 3D occupancy grid aggregation, and costmap generation.

Keywords: Autonomous ground vehicle, perception, world modeling



(a) MAGV HMMWV testbed

(b) High–level system architecture

(c) MAGV perception / world model system

Figure 1. Multirole Autonomous Ground Vehicle (MAGV): (a) HMMWV testbed for assessing autonomous technologies, (b) high-level system architecture, (c) perception pipeline, including LIDAR, and color and IR stereo

1. INTRODUCTION

In recent years, the development of autonomous ground vehicles has seen renewed interest, both from the military[1,2] and industry[3,4] sectors. A major push for this innovation came from the challenges organized by the Defense Advanced Research Projects Agency (DARPA),[5–7] 2004 - 2007. Since the DARPA challenges, several companies, including Apple, Google, Tesla, Daimler,[8] and BMW,[4] have been developing autonomous capabilities. Likewise, several government labs have also been developing autonomous capabilities, including the

Further author information: (Send correspondence to J. Zapf)
J. Zapf: E-mail: jjzapf@spawar.navy.mil

Unmanned Systems Technology XIX, edited by Robert E. Karlsen, Douglas W. Gage, Charles M. Shoemaker, Hoa G. Nguyen, Proceedings of SPIE Vol. 10195, 101950F · © 2017 SPIE
CCC code: 0277-786X/17/$18 · doi: 10.1117/12.2266690

US Army Tank Automotive Research Development and Engineering Center (TARDEC), NASA Jet Propulsion Laboratory (JPL), and SPAWAR Systems Center (SSC), Pacific. Government lead initiatives include Cargo UGV,[9] Dismounted Soldier Autonomy Tools (DSAT), and Multirole Autonomous Ground Vehicle (MAGV).

The Multirole Autonomous Ground Vehicle (MAGV) program is sponsored by the Office of Naval Research (ONR) Code 30 to develop an autonomy kit that can be installed on an expeditionary ground vehicle to support partial or fully autonomous missions, including route reconnaissance and clearance, remote overwatch, and logistics connector. Figure 1 depicts the MAGV system. Subfigure (a) depicts the system installed on a HMMWV. Subfigure (b) depicts the high-level system architecture. Subfigure (c) depicts a simplified breakdown of the perception and world model systems. Onboard sensors (color stereo, IR stereo, LIDAR) provide data which is processed by the perception system. The ***perception*** system extracts information concerning the vehicle's environment and uses the pose estimate from the ***localization*** system to store this information in the world model representation. The ***world model*** uses a combination of the perceived data along with a priori knowledge to generate cost structures to be used by the path planning system. Finally, the ***path planning*** system utilizes this information to generate reference trajectories for the actuator controls.

This paper focuses on the interaction between the perception and world model systems. We refer to this interaction as the ***perception pipeline***. The perception pipeline is composed of several components, including stereo filtering, pixel segmentation and classification, sensor fusion, and costmap generation. In Section 2, we begin with a detailed look at sensor calibration. In Section 3, we next discuss several algorithms dealing with sensor processing. In Section 4, we describe the sensor fusion process in which multiple sensor streams are combined in the world model representation. In Section 5, we then discuss several real world experiments and results. Finally, in Section 6 we end with some concluding remarks and future work.

2. SENSOR CALIBRATION

We now discuss the routines developed for calibrating the perception head depicted in Figure 1. The perception head includes two 75 cm baseline stereo rigs, color and IR, as well as a 16-line scanning LIDAR. Table 1 gives a complete list of the primary components and their associated costs at the time of purchase.

Table 1. MAGV perception head

Component	Quantity	Price per Unit
Velodyne Puck (VLP-16) LIDAR	1	$8K
Flir A65 thermal imaging camera	2	$8K
Point Grey GS3-PGE-23S6C-C color GigE camera	2	$1K
Kowa LM8HC-V 1" 8mm F1.4/F2.8/F4/F8 lens	2	$0.5K
Highland Technology T340 function generator	1	$2.5K
Total		$29.5K

2.1 Intrinsic Camera Calibration

Intrinsic camera calibration for the color stereo cameras uses a standard checkerboard or dot pattern and the detection and calibration routines provided by the OpenCV library,[10] which implements the method proposed by Zhang.[11] Using a custom calibration GUI, seen in Figure 2, trained operators are typically able to achieve reprojection errors on the order of 0.1 pixels. A stereo range accuracy curve for a typical calibration can be seen in Figure 3. The GUI also provides functionality for checking reprojection and epipolar error.

While visual-spectrum calibration is well understood, calibration of long-wave IR cameras, such as the FLIR A65, which operates in the 7.5 - 13 μm spectral range, has not been thoroughly explored. This range is generally thought of as "thermal", as it is the range in which the black-body radiation curve peaks for typical temperatures (0 - 100 C). As such, when attempting to capture observations of a calibration target, the simplest way to generate local responses is with a temperature gradient. To this end, we developed and tested a range of calibration targets with a temperature differential, see Figure 4. The first, baseline target, was a standard black and white target heated by direct sunlight. Sunlight is effective at generating a large temperature difference (>30 C in our experiments) between black and white surfaces. Unfortunately, this requires near perfect weather conditions and

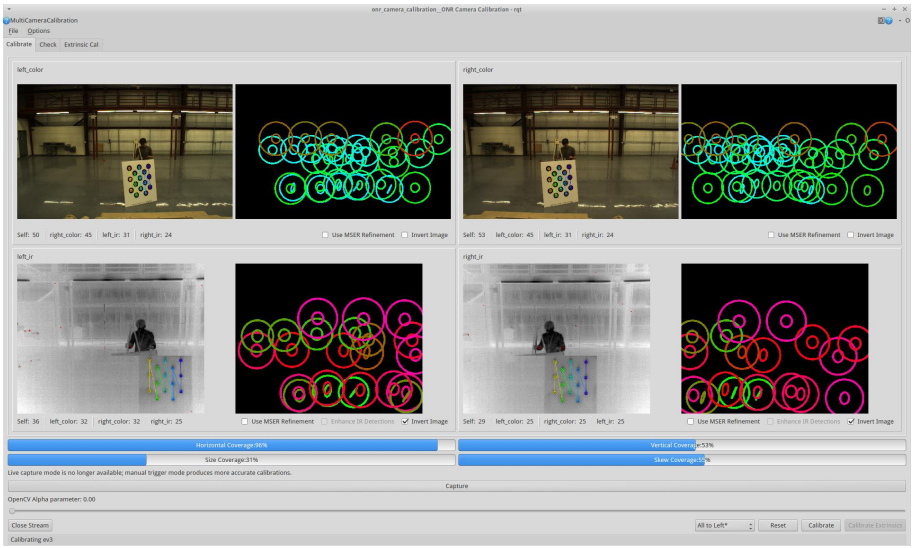


Figure 2. Stereo calibration GUI: Operators can modify all of the parameters governing calibration and are provided with visual feedback and progress bars that estimate coverage completeness. The GUI also contains tabs for checking the quality of an existing calibration and for calibrating stereo-to-LIDAR extrinsics.

cannot be performed indoors. Furthermore, the amount of thermal radiation can vary drastically. At times, this results in over-heating the board, which results in blurring of the target image.

Moving away from a passive target, we developed two "active" calibration targets. The first used a matrix of standard white LEDs, which we ran at an overvoltage of 8 V to generate heat. While effective at generating a heat signature, the need to run the LEDs above their standard operating range resulted in decreased lifespan. The second target used low-resistance resistors across a small voltage source to produce heat and did not suffer from any component failures. Experiments using this target showed that trained operators could consistently achieve reprojection errors similar to those produced by the color stereo.

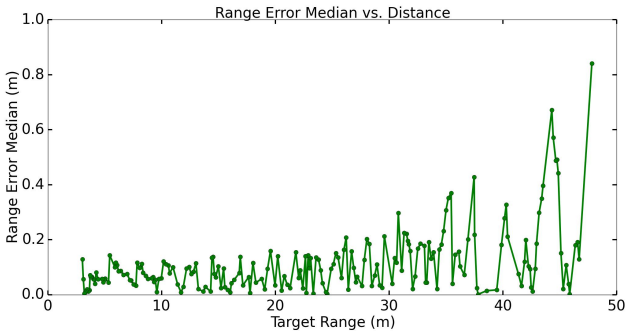


Figure 3. Stereo range accuracy curve

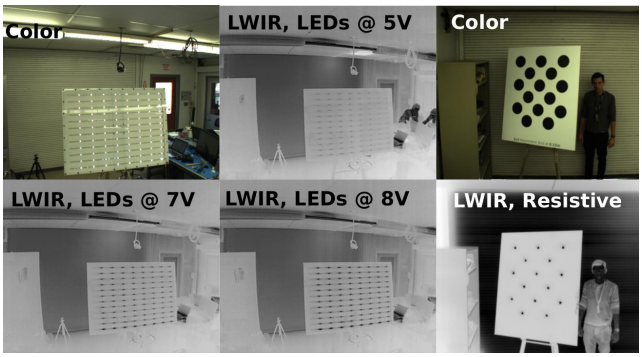


Figure 4. Examples of calibration targets in LWIR

2.2 Extrinsic Stereo-to-LIDAR Calibration

Extrinsic stereo-to-LIDAR calibration is more difficult than camera-to-camera extrinsic calibration, since it requires finding correspondences between different modalities. In this work, we match stereo points to LIDAR points in two steps. First, we use the calibration board detections from the stereo calibration to determine a rough initial alignment. Second, we use a global registration method to refine the initial alignment.

To align the calibration board detections, we must determine the locations of corresponding features in both the stereo and LIDAR point clouds. For stereo, we detect the calibration target dots in the left and right images

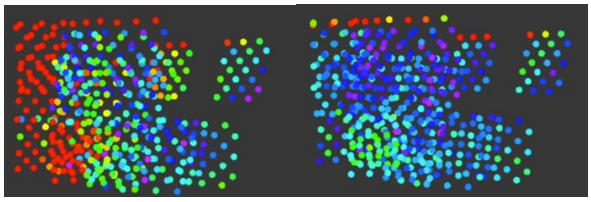


Figure 5. Errors for detected dots before (left) and after (right) alignment. Red (blue) denotes high (low) error.

and use triangulation to determine their locations in 3D space. Localizing the target in the LIDAR point cloud requires detecting and segmenting out the target using purely geometric features, since the pattern is not visible. To do this, all planes in the point cloud are first segmented and a stereo-to-LIDAR transformation is calculated for each plane that matches the size of the actual target. Once this is done for each stereo-LIDAR point cloud pair, the transformations are clustered and the one with the most correspondences is used as the initial guess for the extrinsic transformation, see Figure 5. This initial guess is used as the initial condition for a Generalized Iterative Closest Point (ICP)[12] point-to-plane alignment. This alignment can be performed on any scene, but we found it worked best on large flat textured surfaces, e.g., the corner of a brick building.

3. SENSOR PROCESSING

In this section, we describe the sensor processing chain in which raw sensor data is transformed into various knowledge products characterizing the state of the robot environment. The primary perception sensors are LIDAR, color stereo, and IR stereo. The data streams produced by these sensors are processed individually and their resulting knowledge products are combined in the world model. Figure 6 depicts the basic processing chain for each of these sensors. The architecture for the IR stereo processing chain is identical to the color stereo processing chain. However, the negative obstacle detector, roughness detector, and terrain classification nodes are currently only used on the color data.

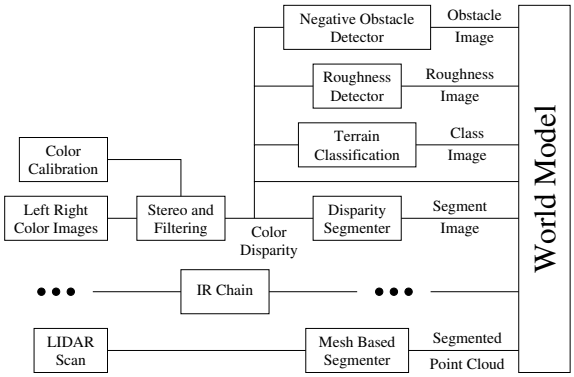


Figure 6. Color stereo, IR stereo, and LIDAR processing chains

3.1 Stereo Correlation and Filtering

Stereo pairs of color and long wavelength IR (LWIR) cameras are used for passive perception in the MAGV system. The LWIR cameras provide stealthy night-time capabilities as well as fill in holes left by the color cameras due to over and underexposure. For both imaging modalities, the stereo pipeline described in our prior work[13] is leveraged to reduce noisy range data from the stereo correlator. As shown in Figure 7, a series of pre- and post-filters are applied to help increase the disparity density of the stereo correlator and remove spurious false positive range data. A local, window-based stereo correlator is employed for improved mixed pixel disparity and efficient computational performance at 10 Hz.

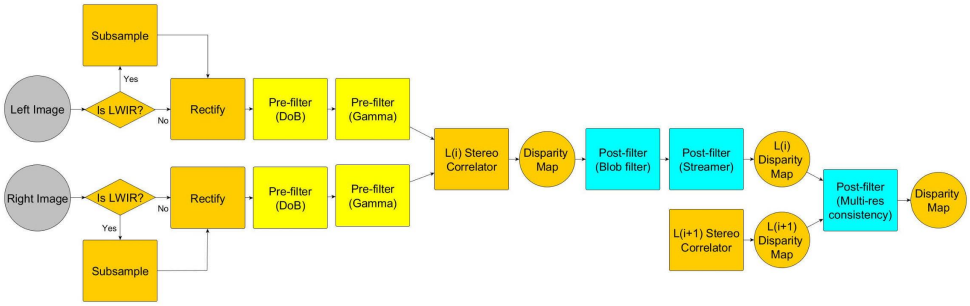


Figure 7. Stereo pipeline for color and LWIR imagery

3.1.1 Pre-filters

For both image modalities, we apply band-pass pre-filtering to reduce the image pair intensity differences and a Laplacian of Gaussian convolution to enhance the underlying image texture through a difference of boxes (DoB) approximation. The normalization of the image intensities is required since approaches that compare local matches based on intensity differences, such as the sum of absolute differences (SAD), are sensitive to contrast changes. Finally, a gamma correction filter is applied to enhance image contrast before matching. Due to the lower signal-to-noise ratio of the LWIR data, the LWIR pipeline has an additional pre-filter step that downsamples the input imagery to reduce the inherent noise from the sensor using 2x2 block averaging.

3.1.2 Stereo correlator

As in our previous work,[13,14] we leverage the computational efficiency of the SAD similarity metric with SAD5 overlapped correlation windows. The constraints of using only CPU-based processors for stereo processing at 10 Hz prohibited the use of algorithms with greater computational complexity, such as semi-global matching. The neighboring windows improve the correlator along disparity discontinuities by reducing the ambiguity between foreground and background objects. The overlapped windows also create a larger correlation area that helps increase correspondence. To further increase the disparity density, we use a rectangular 11x7 window that is more suitable for outdoor scenes with a receding ground plane. Due to the computational demand of performing both color and LWIR simultaneously at multiple resolutions, the SAD5 algorithm was reimplemented using Intel Advanced Vectorized Extensions (AVX2) to leverage 256-bit registers. The improved implementation yielded a performance increase of 35%.

3.1.3 Post-filters

Similar to previous work,[13,14] the spurious false positive range data generated by the stereo correlator is removed using a blob filter, streamer filter, and a multi-resolution consistency check. The blob filter examines the gradient of disparity data to find regions of similar disparity. The blob filter helps remove speckles of spurious range data. The streamer filter removes artifacts from mixed pixels which overlap both foreground and background depth values. This ambiguity results in a stream of false positive depth values emanating from the foreground object to the background. The streamer filter removes these disparities by examining a trio of range pixels and testing for receding depths above a given threshold.

Repeated textures cause ambiguity in window-based correlators resulting in noisy range data. To mitigate this behavior, the multi-resolution consistency check compares the range values at different image pyramid levels to ensure they agree within some threshold. We leverage the coarser and typically denser range values of the lower image resolution to identify false positives in the higher resolution. As shown in Figure 8, our algorithm uses the next level in the image pyramid $L(i+1)$ to check the input image level $L(i)$ range data. Only the $L(i)$ pixels that have valid disparity values in $L(i+1)$ are checked for consistency. All other $L(i)$ pixels are invalidated. For the color pipeline, L0 and L1 levels are used. For LWIR, L1 and L2 levels are used.

The consistency threshold is based on the expected range error Δr given by

$$\Delta r = \frac{r^2}{fb}\Delta d, \tag{1}$$

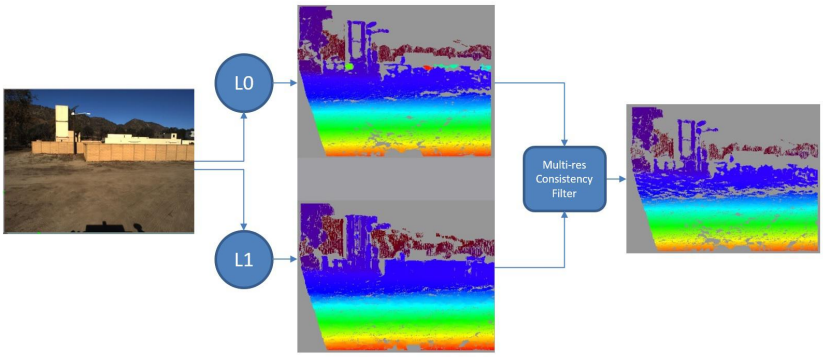


Figure 8. Multi-resolution consistency filtering

where r is the target range, f is the focal length of the camera in pixels, b is the baseline between cameras, and Δd is the pixel correlation uncertainty. Through this formulation, the multi-resolution consistency filter adaptively selects the consistency threshold, maximizes the preservation of ranges in the more accurate near and mid ranges, and can easily be applied to both image modalities.

3.2 Ground and Object Segmentation

To safely navigate an environment, it is convenient to segment the sensor point clouds into ground and object groups. As depicted in Figure 6, ground segmentation is performed independently for each sensor. Performing the segmentation in the sensor space provides the highest level of resolution, while segmenting on individual sensor streams simplifies the system architecture.

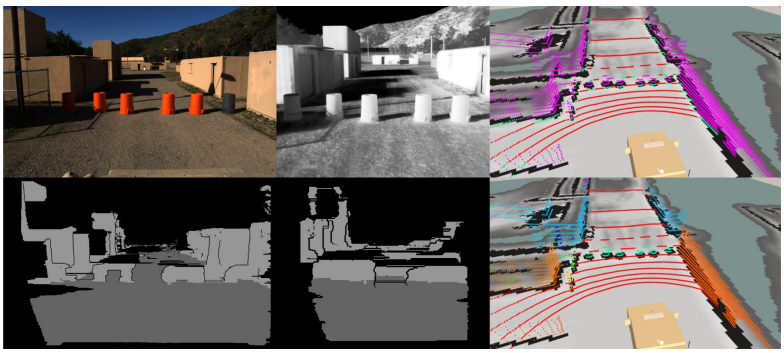


Figure 9. Ground and object segmentation: (left) color stereo ground segmentation, (middle) IR stereo ground segmentation, (top-right) LIDAR ground segmentation, (bottom-right) LIDAR object clustering

3.2.1 Stereo Segmenter

The stereo ground segmentation used by MAGV was developed by the Southwest Research Institute (SwRI) under the Small Unit Mobility Enhancement Technology (SUMET) program. The algorithm provides ground/not-ground classification of each pixel in a stereo image. The steps are provided in Algorithm 1 and are discussed here. Example segmentations for color and IR are provided in Figure 9.

In Step 1, a "v-disparity"[15] image $I_{v\Delta}$ is constructed from the original disparity image I_Δ by accumulating the pixels of same disparity along the $\boldsymbol{v}$-axis, where (u, v) denotes the coordinates of a pixel in I_Δ. That is, for a given point $(u', v') \in I_{v\Delta}$, the intensity is given by the number of points in the corresponding row $v \in I_\Delta$ with disparity equal to v'. In Step 2, a ground plane is extracted in the vicinity of the vehicle using $I_{v\Delta}$. Note that a ground plane maps to a line in $I_{v\Delta}$. Thus, a Hough transform is used to robustly extract the plane.

In Step 3, the original disparity image I_Δ is filtered by scanning each row and filling gaps less than a certain distance with the lesser of the two end disparities. Sobel operators are then applied in Step 4 to obtain the x and y gradient images. Next, a flood filling algorithm is used in Step 5 to separate ground and object pixels further from the vehicle. The flood fill is constrained both by the gradient and the space, i.e., disparity gap, between neighboring pixels. The remaining non-ground pixels represent object pixels. In Step 6, object heights are computed using the flood-filled ground for all object points within a neighborhood of a ground point. Short objects are then removed in Step 7 and the flood fill is performed again in Step 8 to refine the ground map.

Algorithm 1 Stereo ground segmentation

1: Construct v-disparity image
2: Perform ground plane detection
3: Filter disparity image
4: Compute disparity gradient images
5: Apply flood fill to extract ground
6: Create object and height maps
7: Remove short objects
8: Reapply flood fill to refine ground

Algorithm 2 Mesh based LIDAR segmentation

1: Construct terrain mesh from LIDAR scan
2: Compute gradient field from terrain mesh
3: Extract ground points using gradient field
4: Find transition points
5: Cluster non-ground points

3.2.2 LIDAR Segmenter

The raw LIDAR scan is segmented into ground and non-ground points by applying local adjacency constraints.[16] The steps are provided in Algorithm 2 and discussed here. An example segmentation is provided in Figure 9.

In Step 1, the terrain mesh is constructed from the LIDAR scan by creating an undirected graph, with connections defined for each node that point to its LEFT, RIGHT, UP, and DOWN neighbors. In Step 2, the gradient field is then computed by estimating a gradient value at each node according to

$$\nabla f = \max_i \left(\frac{dz_i}{\sqrt{dx_i^2 + dy_i^2}} \right), \tag{2}$$

where $(dx_i, dy_i, dz_i)^\top$ denotes the vector pointing from the source node to one of it's four (LEFT, RIGHT, UP, or DOWN) neighbors. To avoid noisy gradient estimates, a minimum distance threshold must be satisfied when selecting the neighboring node. Next, an agglomerative clustering algorithm is used in Step 3 to extract the ground points. The algorithm is seeded with a set of points in the near vicinity of the vehicle, and then a recursive search is performed to find neighboring nodes that satisfy the max gradient threshold. To avoid excessively large jumps, thresholds are also placed on the link distances.

After the ground points have been extracted, transition points are identified in Step 4 by traversing the rings and looking for non-ground points that are adjacent to ground or transition points and satisfy one of the following conditions: (1) the gradient value without the up link computed is below the max gradient threshold and the change in height is less than the max height threshold or (2) the link distance is less than a threshold. Finally, in Step 5 all points that were not labeled ground or transition are clustered into groups using an agglomerative clustering approach based only on the link distances. In a future work, these object groups will be fed to a multi-object tracking system.

3.3 Terrain Classification

Terrain classification is needed for autonomous navigation to provide situational awareness of the types of objects and surfaces within the robot's environment. This ability is particularly important for expeditionary vehicles, where the terrain class provides critical information for determining traversability. In the following sections, we describe two terrain classification methods. The first uses a decision tree approach and is currently part of the MAGV system. The second uses a deep convolutional neural network approach and is currently under development for future releases of the MAGV system.

3.3.1 Decision Tree Classifier

The current terrain classifier used by MAGV was developed by the Southwest Research Institute (SwRI) under the Small Unit Mobility Enhancement Technology (SUMET) program.[17] The approach uses a standard decision tree to classify pixels based on a number of available features. These features are organized into plug-ins with configurable parameters. Feature selection is accomplished via a comprehensive search over a large subset of reasonable feature configurations using a standard cross-validation approach. Available features include: (1) stereo disparity, (2) ground segmentation, i.e., a binary classification of disparity pixels into ground and non-ground, (3) texton features,[18] (4) local binary pattern features,[19] and (5) statistical features, e.g., skewness, kurtosis, maximum peak height, developed interfacial area ratio, and average roughness. Typical class labels include grass, foliage, wood, dirt, rock, concrete, and sky.

The implemented decision tree incorporates a confidence measure based on the strength and purity of the leaf. The strength is defined by the number of training samples sorted into the leaf, whereas the purity is defined by the ratio of correctly labeled samples to the number of training samples sorted into the leaf. Specifically, the confidence is calculated as

$$confidence = \frac{N_{\text{leaf}} - N_{\text{risk}}}{N_{\text{leaf}}} \cdot \frac{\min(N_{\text{leaf}}, c_{\text{saturation}})}{c_{\text{saturation}}}, \tag{3}$$

where N_{leaf} is the number of samples in the leaf, N_{risk} is the number of samples in the leaf that do not have the leaf label, and $c_{\text{saturation}}$ is the number of samples a leaf must contain to be considered densely populated. By giving every classification a confidence, suspect samples can be handled with more caution when integrating samples in the world model.

To train and test the classifier, a custom graphical user interface was designed. The process for constructing the training set involves labeling data, retraining, and then comparing the classified result with the original image. Errors in the classified image are labeled and used to retrain the classifier. The test set is created using a similar process. For a typical dataset consisting of labeled images taken from a wooded trail road, the average classification rate is 85.7%. Figure 10 depicts a typical classified scene.

Figure 10. Decision tree classification example: (left) projected RGB corresponding to valid stereo points, (middle) class labeled image, (right) confidence image

3.3.2 Convolutional Neural Network Classifier

Estimation of terrain classes directly from single RGB images requires high level understanding beyond what is needed for the decision tree approach, as there is no explicit geometric encoding to help detect surfaces such as roads. Deep Convolutional Neural Networks (CNNs) are well suited to this, as their deep representation permits them to model the higher order relationships that must be inferred without explicit geometric knowledge. For this work, we selected the MultiNet architecture proposed by Teichmann et al.[20] as our starting point, due to its efficient run-time performance and ability to adapt on relatively small training sets. MultiNet is a derivative of two of the most popular CNN networks; the Visual Geometry Group (VGG) network,[21] and the Fully Convolutional Network (FCN).[22]

VGG is useful as it provides a set of weights that were pre-trained on the ImageNet[23] dataset. While the VGG weights were trained on an object classification task, their use has been shown to provide a significant boost over random initialization across a wide variety of tasks. For this work, we used VGG weights as our initial condition for all layers, with the exception of the final layers at the top of the network. These top layers

are randomly initialized and trained from scratch, as our output classes bear no similarity to those of ImageNet. The FCN architecture is important as it takes baseline VGG and removes the fully connected layers used for classification. These layers are replaced with deconvolution layers, which make the network a fully convolutional network; a network which maps directly from input pixels to class-labeled output pixels.

Our dataset consisted of 1492 images which were hand-annotated with 10 classes. Labelers did not attempt to densely paint all pixels, but rather simply made marks over general areas, as seen in Figure 11. Training took approximately 2 hours on a Titan X GPU, and consisted of fine-tuning the pre-trained VGG weights to our particular task. A full quantitative evaluation of classifier performance is beyond the scope of this work, but qualitative results from a hold-out validation set are shown in Figure 11. From these, it is clear that the classifier generalizes well and is not affected by the sparse labeling of the training set. Classification times on the Titan-X GPU were below 100 ms, meaning that the classifier could meet the vehicle's 10 FPS requirement.

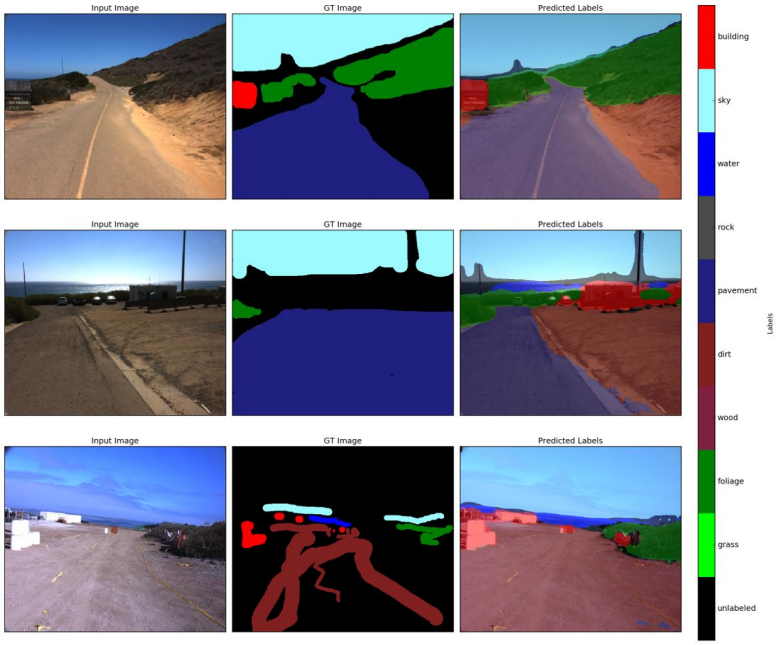


Figure 11. Examples of results from the CNN classifier: The left column shows input RGB images. The middle shows ground truth annotated labels. The right shows the predictions from our CNN. The CNN is able to learn a dense labeling, even though the ground truth annotations were sparse. Furthermore, observe that the CNN is actually far more accurate than the ground truth for the image in the middle row; it correctly labels the dirt on the right side, where the ground truth has pavement. These images were from the hold-out set, and were not seen during training.

3.4 Roughness Detection

Small objects like rocks and trunks may present traversability hazards to the platform. However, their size may be less than the resolution of the world model (0.2 m), which is limited by memory and computation. Nonetheless, the stereo cameras have sufficient native resolution to detect these small hazards. The objective of the roughness detector is therefor to quantify the coarse nature of these objects by estimating their local height variances and providing these as attributes to the world model.

For every disparity image, we transform each disparity pixel into gravity-aligned Cartesian coordinates and aggregate the points into a grid of 3D cells (voxels). The geometry estimated by the stereo system contains orders of magnitude difference in resolution (0.05 m) and range (>30 m) and is sparse with respect to the resolution. Given the sparsity of the data and the pseudo real-time requirement ($\leq$ 10 Hz), we use a spatial hashing scheme to achieve the update speed of a regular voxel grid with a fraction of the required memory. Each voxel maintains the number of points inserted into the voxel and a rolling sum of the first and second moments

of the gravity-aligned z-axis value of all the points inserted into the voxel. This scheme makes the computation of the variance at each voxel invariant to the number of inserted disparities for a given neighborhood and allows for the reversible insertion and removal of points. Since individual range images are noisy and incomplete, we maintain a sliding window of the past N frames. We then compute the local height variance of each voxel by summing the moments of its immediate voxel neighbors and project this into a costmap. See Figure 12 for an example scene with corresponding roughness estimate and projected costmap.

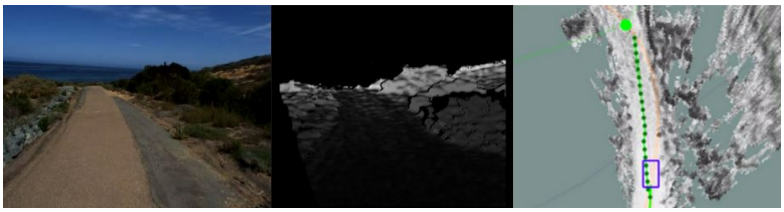


Figure 12. Roughness detection: (left) left stereo camera image, (middle) estimated roughness image, i.e., variance, (right) projected roughness costmap

3.5 Negative Obstacle Detection

Negative obstacles are a particularly dangerous class of obstacles that can cause catastrophic damage to the platform. During this program, we focused on the two most hazardous negative obstacles encountered on dirt trails: steep declines to the sides and abrupt drop-offs. In regions void of perception data, the world model assigns a cost that represents unknown terrain. Because the planner allows some level of exploration in unknown terrain, it is crucial to explicitly detect if the leading edge of an unknown region is the start of a drop-off or a steep decline. The drop-off and side-slope detectors output detections that are inserted into the costmap at 10 Hz via an attribute plugin to the world model. Figures 13 and 14 show the clear benefits of the negative obstacle detector. To avoid potential false positives, current and previous detections are differenced using a distance threshold of 0.5 meters and any detection not found in the new frame is discarded.

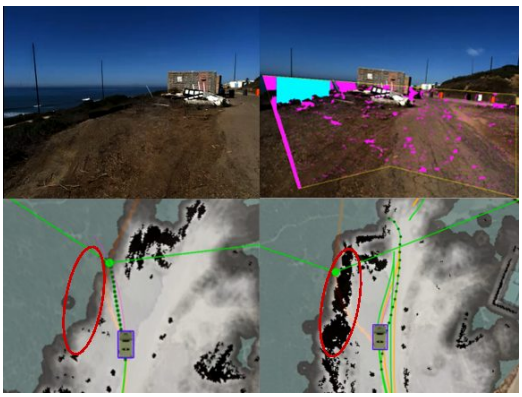


Figure 13. Negative obstacle detector - side drop-off: (top-left) original image, (top-right) detection image (cyan = negative obstacle, magenta = disparity void), (bottom-left) costmap without detector, (bottom-right) costmap with detector

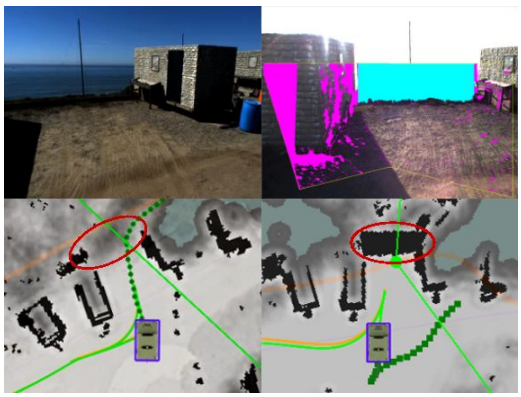


Figure 14. Negative obstacle detector - front drop-off: (top-left) original image, (top-right) detection image (cyan = negative obstacle, magenta = disparity void), (bottom-left) costmap without detector, (bottom-right) costmap with detector

The side-slope and drop-off detectors operate on a range image generated by the color stereo cameras. To constrain the search, the image coordinates of the horizon are estimated for each column in the range image using the left rectified camera model transformed into the vehicle coordinate frame. A workspace polygon is then defined such that the top line is the horizon and lower left and lower right corners of the range image are not searched. The drop-off detector steps down each column of the stereo range image starting from the horizon and labels each no data pixel as a candidate drop-off pixel until it reaches a pixel with range data. N meters of range

data just below the bottom candidate drop-off pixel of each column are evaluated; if the data density is high and the 3D geometry is fairly planar with low slope, the candidate drop-off pixels in that column are transferred to a drop-off detection image. Once the search is complete, a connected components algorithm is run on the drop-off detection image and close detection regions with similar alignment are connected. The components that have a small 3D length, a range beyond a maximum range, or an extreme aspect ratio are discarded.

The side-slope detector has vertical and horizontal detectors. The vertical side-slope detector steps up each column of the range image within the workspace polygon. During the search, if (1) there is a range void above a pixel, (2) the distance across the range void exceeds a minimum depth, and (3) there is a down slope across the range void smaller than a maximum slope, the pixels in the range void of that column are labeled candidate side-slope pixels in a side-slope detection image. The horizontal side-slope detector performs a similar search across each row starting at the center and working its way out to the polygon extents. As with the drop-off detector, a connected components algorithm is again applied to filter out potential false positives.

One complication we encountered is image saturation due to under and overexposure, which can cause voids in the stereo range image with 3D signatures similar to those of drop-offs. Thresholding image intensity and variance of a region of interest inside candidate drop-offs has been shown to reduce false positives. LIDAR data could also be used to determine if a solid surface exists where there is image saturation.

4. SENSOR FUSION

We now describe the process by which the sensor streams and knowledge products described above are fused in the world model. The world model system was originally developed by the Southwest Research Institute (SwRI) under the Small Unit Mobility Enhancement Technology (SUMET) program. Since the close of SUMET, several new capabilities and enhancements have been added to this system under MAGV.

The world model is responsible for maintaining a model of the robot environment as well as creating costmaps to be digested by the path planning system. In addition to processing the onboard sensor data, the world model also maintains several databases of a priori information that can be used to enhance and complement the observed data. These include the OpenStreetMap[24] route network, elevation maps, previously observed obstacle maps, and user supplied no-go regions. Although very important, the use of a priori data is outside the scope of this paper. Instead, we focus solely on the fusion of the onboard sensor data, which is especially critical for operating in expeditionary environments where a priori data may not be available.

4.1 Multi-sensor Integration

Before discussing the details of our sensor fusion process, we first motivate the problem by discussing the need for multi-sensor integration and provide some initial results illustrating the benefits of our approach. When operating in complex environments, it is nearly impossible to create a complete map of the robot's surroundings using a single sensing modality. All sensors have limitations and are susceptible to environmental conditions that degrade performance. For this reason, it is important to chose a sensor suite that is complementary, leveraging the strengths of other sensors when one or more sensors becomes unreliable.

A color stereo rig is an inexpensive sensor that can produce fairly accurate point clouds by distinguishing fine variations in image texture. However, a color stereo rig is susceptible to poor lighting conditions including overexposure from sun glare and underexposure from deep shadows. To alleviate these shortcomings, an infrared (IR) stereo rig can be used in conjunction with a color stereo rig. An IR camera forms an image using infrared radiation rather than visible light. As such, it is less susceptible to changing lighting conditions. The downside to IR is that the cost to image quality ratio is much higher than for color. Figure 15 illustrates the benefit of our approach for fusing color and IR stereo.

LIDAR is another sensing modality that is useful for autonomous navigation which works by illuminating the target with a laser light. The point clouds produced by LIDAR are generally not very susceptible to lighting conditions and tend to be very accurate. Nonetheless, LIDAR tends to be more susceptible to small particulate matter, i.e., dust, and does not have the same detection and classification power as color imagery. Furthermore, LIDAR is an active sensor and thus easier to detect, which may be a concern for military operations.

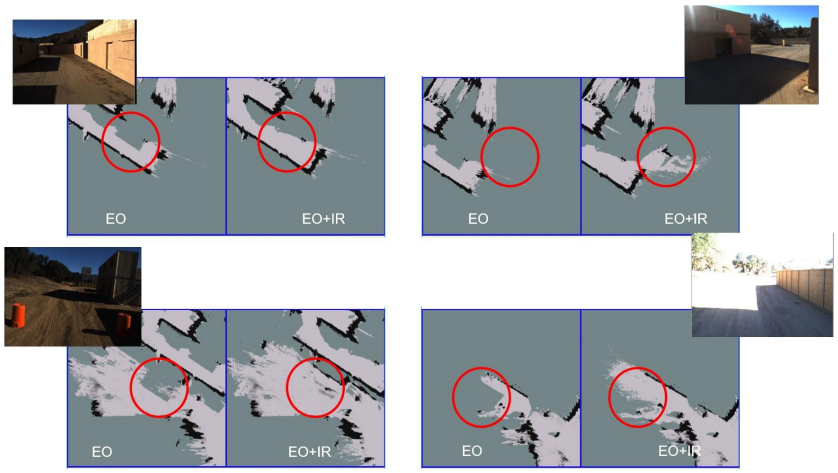


Figure 15. Integration of color and IR stereo: (top and bottom-left) IR is used to fill in holes left by color stereo in areas of underexposure. (bottom-right) IR is used to fill in holes left by color stereo in areas of overexposure.

4.2 Voxel Grid

A popular approach to modeling environments in 3D is to use a grid of cubic volumes of equal size (voxels) to discretize the mapped area. The MAGV world model uses an octree representation to manage this grid, similar to that used in OctoMap.[25] The grid is defined with respect to a fixed global frame such that the origin and orientation are aligned with the initial vehicle pose. As the robot moves through the environment, the localization solution is used to place sensor hits within their appropriate voxels. With each sensor hit, the voxel attributes are updated, including the occupancy probability, color, roughness, and material classification. Along with insertion operations, the world model also handles deletion operations, whereby a voxel previously identified as occupied is cleared. Deletion operations are necessary due to the presence of false positives from noisy sensor measurements and the time-dependent nature of dynamic objects. Voxel clearing is handled by the disparity and LIDAR ray tracing algorithms discussed in Section 4.3.

4.2.1 Log Odds Update

An occupancy probability or confidence is maintained in each voxel which encodes our belief about whether or not the voxel is occupied by an object. A common approach for integrating various sensor measurements is based on Baye's theorem.[26] The objective is to estimate the posterior probability of occupancy $P(m|x_{1:t}, z_{1:t})$, where m is the grid index and $z_{1:t}$ and $x_{1:t}$ are the sensor measurements and poses up to time t. Applying Baye's theorem, the conditional probability may be expressed as

$$P(m|x_{1:t}, z_{1:t}) = \frac{P(z_t|x_{1:t}, z_{1:t-1}, m)P(m|x_{1:t}, z_{1:t-1})}{P(z_t|x_{1:t}, z_{1:t-1})}. \tag{4}$$

Assuming that the current measurement z_t is independent from $x_{1:t-1}$ and $z_{1:t-1}$ given that we know m, i.e., $P(z_t|x_{1:t}, z_{1:t-1}, m) = P(z_t|x_t, m)$, it follows that

$$P(m|x_{1:t}, z_{1:t}) = \frac{P(m|x_t, z_t)P(z_t|x_t)P(m|x_{1:t}, z_{1:t-1})}{P(m)P(z_t|x_{1:t}, z_{1:t-1})}. \tag{5}$$

By analogy, a similar expression can be derived for the complement $P(\bar{m}|x_{1:t}, z_{1:t})$. Writing the ratio of the probability that the voxel is occupied to the probability that the voxel is free, we arrive at

$$\frac{P(m|x_{1:t}, z_{1:t})}{P(\bar{m}|x_{1:t}, z_{1:t})} = \frac{P(m|x_t, z_t)}{P(\bar{m}|x_t, z_t)} \cdot \frac{P(\bar{m})}{P(m)} \cdot \frac{P(m|x_{1:t-1}, z_{1:t-1})}{P(\bar{m}|x_{1:t-1}, z_{1:t-1})}. \tag{6}$$

The above function is an example of an odds function which has the general form

$$\text{Odds}(x) = \frac{P(x)}{P(\bar{x})} = \frac{P(x)}{1 - P(x)}. \tag{7}$$

Consider now that $P(m)$ is simply the prior probability that the voxel is occupied. With no prior knowledge regarding the voxel's state of occupancy, $P(m) = P(\bar{m}) = 0.5$, and Equation 6 can be expressed as

$$\text{Odds}(m|x_{1:t}, z_{1:t}) = \text{Odds}(m|x_t, z_t) \cdot \text{Odds}(m|x_{1:t-1}, z_{1:t-1}). \tag{8}$$

Finally, taking the log of both sides, we obtain the log odds ratio

$$\text{L}(m|x_{1:t}, z_{1:t}) = \text{L}(m|z_t, x_t) + \text{L}(m|x_{1:t-1}, z_{1:t-1}). \tag{9}$$

The above equation implies that multiple sensor measurements can be fused using a simple additive update to produce a probability of occupancy, which is essentially a weighted count of the sensor hits. According to the definition of the odds function, Equation 7, a voxel is considered occupied if $\text{L}(m|x_{1:t}, z_{1:t}) > 0$. Note, however, that the above update function may become problematic in dynamic environments where the map must adapt quickly, since a large number of updates may be required to change the sign of $\text{L}(m|x_{1:t}, z_{1:t})$. To overcome this, the following clamping policy is implemented to bound the log odds ratio

$$L(m|z_{1:t}) = \max(\min(L(m|z_{1:t-1}) + L(m|z_t)), l_{max}), l_{min}). \tag{10}$$

4.2.2 Exponential Inverse Sensor Model

The probability $P(m|x_t, z_t)$ is referred to as the "inverse sensor model", because it maps the sensor measurement z_t back to its causes, i.e., the occupancy of m. A simplified version of the inverse sensor model developed by Heng[27] was adapted for MAGV and is defined as

$$p(r) = 0.5\left(1 + ke^{-\Delta r}\right) \qquad \Delta r = \frac{r^2}{fb}\Delta d, \tag{11}$$

where r is the measured depth, f is the camera focal length in pixels, b is the stereo baseline, and k and Δd are tunable parameters denoting the stereo significance and the disparity uncertainty. Note that the probability of occupancy is expressed as a decaying exponential of the distance from the camera. A plot of this function is provided in Figure 16 for $k = 1$ and $\Delta d = 0.5$. To gain insight about this model, consider that the depth of a point with disparity d is given by $r = fb/d$. Differentiating with respect to d, we obtain

$$\Delta r = \frac{fb}{d^2}\Delta d, \tag{12}$$

where Δr and Δd represent the uncertainty of the range and disparity measurements. Thus, the smaller the disparity, the larger the uncertainty in the range measurement.

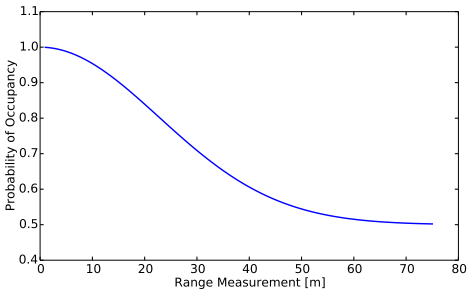


Figure 16. Exponential inverse sensor model

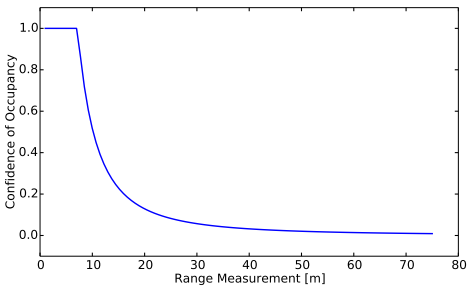


Figure 17. Pixel error inverse sensor model

4.2.3 Pixel Error Inverse Sensor Model

A second inverse sensor model was tested for MAGV which we now describe. Assuming a ± 1 pixel error in disparity, the possible depth range for a given point is

$$r_{min} = \frac{fb}{fb/r + 1} \qquad r_{max} = \frac{fb}{fb/r - 1}, \tag{13}$$

where f is the camera focal length in pixels, b is the stereo baseline, and r is the measured depth. The confidence associated with the point actually falling within the voxel can thus be approximated as

$$p(m|x_t, z_t) = k \cdot min(1, max(0, res/(r_{max} - r_{min}))), \tag{14}$$

where res denotes the map resolution (0.2 m). A plot of this function is provided in Figure 17.

4.2.4 Alternative Update Functions and Sensor Models

The MAGV system was written such that it can be configured to use different update functions and sensor models for different sensor types, i.e., LIDAR, and color and IR stereo, and different operations, i.e., insertion and deletion. An example of an alternative update function is the max update function

$$P(m|x_{1:t}, z_{1:t}) = \max\left(P(m|x_{1:t-1}, z_{1:t-1}), P(m|x_t, z_t)\right), \tag{15}$$

which simply keeps the maximum observation probability. Conversely, a min update function could be defined for deletion operations. Obviously, this update function loses the eloquence of the log odds theory derived above. Nonetheless, in certain scenarios where the map must adapt quickly, it does appear to have at least empirical merit. In addition to the inverse sensor models discussed above, simple fixed probability models were also tested. In general, these tended to perform noticeably poorer for stereo, but were sufficient for LIDAR.

4.3 Ray Tracing

From time to time, it is necessary to clear occupied voxels from the voxel grid. These voxels could represent false positives produced by noisy sensor measurements or simply the remains of a dynamic object that has moved. The mechanism for clearing these voxels is referred to as ray tracing and works by tracing the ray emanating from the sensor origin to the observed 3D point, as shown in Figure 18. In the context of the world model, the probabilistic interpretation of the approach is a beam-based inverse sensor model, where the occupancy value of the voxel containing the point return is incremented and the occupancy values of the voxels in the line of sight between the sensor and the point return are decremented.

A common approach to ray tracing is to use a Bresenham algorithm to iterate through all the voxels leading to a sensor observation.[25] However, this can be computationally expensive, since there may be upwards of a million sensor hits per frame with hundreds of voxels between the sensor and each observation. An alternative approach was taken for MAGV based on projecting world model voxels onto range images. This approach can achieve real-time rates on a single thread even for a large number of voxels and sensor observations.

4.3.1 Disparity Ray Tracer

The disparity ray tracer works by first extracting voxels from the voxel grid that correspond to the viewing frustum of the stereo rig. Only voxels previously touched by a sensor hit are returned in this process. Likewise, voxels that already have a minimal occupancy value are skipped. All other voxels are projected into the disparity image. For each voxel projected into the disparity image, we count the number of consistent and inconsistent disparity points that intersect the projection. The term "consistent" here implies that the depth associated with a disparity point is less than the depth of the voxel in question, i.e., the point is in front of the voxel. If the inconsistent points, i.e, the points lying behind the voxel, outnumber the consistent points, i.e., the points lying in front of the voxel, the voxel should be cleared. In practice, however, we skew this ratio to favor the consistent points in order to avoid accidentally clearing a true object.

Note that the inverse sensor models derived above do not apply in this case as they were derived for insertions and not deletions. In theory, a probability model could be designed based on a range dependent weighted average of all the observed disparity points. However, for our purposes it proved sufficient to use either a fixed measurement update value or a value proportional to the ratio of consistent to inconsistent points.

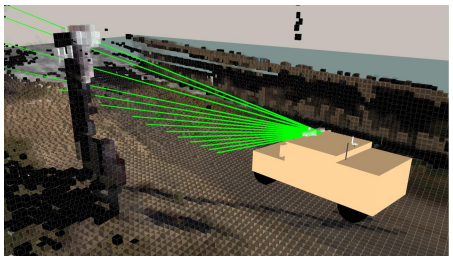


Figure 18. RViz (ROS visualization) of LIDAR ray tracer

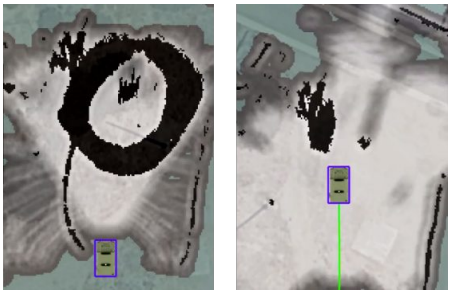


Figure 19. Clearing artifacts with the LIDAR ray tracer: (left) artifacts caused by a dynamic object, (right) cleared costmap after engaging LIDAR tracer and driving forward

4.3.2 LIDAR Ray Tracer

Objects like trees and bushes at the edges of the road induce stereo occlusion boundary artifacts that accumulate in the world model and artificially block turnoffs. With a forward-looking, forward-moving vehicle, these artifacts are never reobserved and therefore do not have the opportunity to be removed by the stereo disparity tracer. The LIDAR sensor has a wide field-of-view that is capable of reobserving and rejecting these artifacts by observing points that lie behind the artifacts in the line of sight.

For every full LIDAR rotation, we construct an artificial spherical range image with a horizontal resolution based on the LIDAR horizontal angular resolution (3.49 mrad) and a vertical resolution based on the beam divergence (3 mrad).[28] Since the vertical resolution of the sensor (34.9 mrad) is an order of magnitude greater than the beam divergence, the resulting range image is sparse. Within the world model, we query all voxels within a 30 m x 30 m area centered around the vehicle. For each voxel, we project the center of the voxel into the spherical range image and compare the voxel's center range to the range indexed in the image. Similar to the disparity tracer, we count the number of consistent and inconsistent points within a radius of the projected voxel center, where the radius is determined by the distance to the voxel. If the inconsistent points outnumber the consistent points, the voxel's occupancy value is decremented.

The method achieves our system's 10 Hz pseudo real-time requirement in single threaded operation. The rate can be increased by evaluating voxels based on a Bernoulli sampling scheme and adjusting the probability that a voxel will be evaluated for tracing. In practice, the LIDAR tracer ran much faster than the required 10 Hz and therefore the Bernoulli sampling scheme was unnecessary, but system-level functional performance was observed to be insensitive to the sampling scheme. Figure 19 shows the effectiveness of the LIDAR ray tracer. Despite the LIDAR's limited angular resolution, the tracer is able to rapidly clear out artifacts through forward motion in a push broom fashion.

4.4 Costmap Generation

The data contained in the voxel grid is used to generate a 2D costmap for the path planning system. The path planning system in turn uses this costmap to generate an obstacle free reference trajectory for the control system to follow. The costmap is aligned with the voxel grid such that the cost for each cell in the costmap is determined by summing the costs associated with each voxel in the corresponding voxel grid column.

4.4.1 Ground Cost

The cost contribution of a ground cell is computed as the sum of the material cost and the normal cost,

$$c_{\text{ground}} = c_{\text{ground\_material}} + c_{\text{normal}}. \tag{16}$$

An occupied voxel may poses one or more material attributes, with defined percentages of composition as determined by the terrain classifier. See Section 3.3 for further details. The material cost associated with these attributes is calculated as the weighted sum of the material percentages,

$$c_{\text{ground\_material}} = \sum_{i=1}^{N} w_i m_i, \tag{17}$$

where N denotes the number of material classes, m_i denotes the i^{th} material percentage, and w_i denotes the weight associated with material i. The normal cost is calculated from the angle formed between the ground normal and absolute vertical. Since the normal vector is a unit vector, this angle is easily calculated as

$$\alpha = \arccos\left(\boldsymbol{n} \cdot [0,0,1]^\top\right). \tag{18}$$

The ground normal cost is then computed by applying a piecewise linear function to the ground normal angle. This function is defined by a set of tunable control points (x_i, y_i), for $i = 1, \ldots, n$, as

$$(x) = \begin{cases} y_1, & \text{if } x <= x_1 \\ \dfrac{(y_i - y_{i-1})}{(x_i - x_{i-1})}(x - x_{i-1}) + y_{i-1}, & \text{if } x_{i-1} < x <= x_i \text{ for } i = 2, \ldots, n \\ y_n & \text{if } x > x_n \end{cases}. \tag{19}$$

4.4.2 Object Cost

The cost contribution of an object voxel is computed as the product of a height function and 3 multipliers,

$$c_{\text{object}} = f_{\text{height}}(h) \cdot k_{\text{object\_material}} \cdot k_{\text{density}} \cdot k_{\text{confidence}}, \tag{20}$$

where $k_{\text{object\_material}}$ is a scaling factor associated with the voxel's material attributes, k_{density} is a scaling factor associated with the density of the observed sensor hits, and $k_{\text{confidence}}$ is a scaling factor associated with the voxel's estimated probability or confidence of occupancy. The height function is a piecewise linear function of the form in Equation 19, with input given by the height of the voxel from the ground. The object material factor $k_{\text{object\_material}}$ is calculated in an analogous fashion to the ground material cost, Equation 17, as the weighted sum of the material percentages. Likewise, the confidence factor is calculated by applying a piecewise linear function to the estimated occupancy probability or confidence.

The density factor k_{density} is set to 1 for all insertion operations, i.e., sensor hits, and set to 0 for most deletion operations, i.e., ray tracing. The only exception is when the pixel error inverse sensor model is used for disparity ray tracing. See Section 4.2.3 for details. In this case, the density value is set to a scaled ratio of the number of consistent disparity points to the total number of disparity points intersecting the voxel projection. The density factor k_{density} is then calculated by applying a piecewise linear function of the form in Equation 19 to the computed density value.

5. EXPERIMENTS AND RESULTS

We now provide some system-wide analysis and results regarding the current state of performance of the perception pipeline. System-wide testing differs form elemental testing in terms of the scope and intent of the experiment. An elemental test seeks to isolate a given capability to quantify only the performance of a subsystem, whereas a system test seeks to evaluate the performance of the system as a whole. Examples of elemental tests include stereo range accuracy, Section 2.1, and cross-validation for terrain classification, Section 3.3. In what follows, we describe the results of several system-level experiments designed to evaluate the overall effectiveness of the developed perception pipeline.

All experiments were performed on the MAGV HMMWV testbed pictured in Figure 1. The primary components of the perception head are listed in Table 1. All computation is handled by a locally networked configuration of 4 desktop-grade computers housed within the vehicle and powered by a bank of batteries tied to the vehicle alternator. Furthermore, A pseudo real-time requirement was placed on all components of the perception pipeline to operate at 10 Hz in order to minimize various lag effects and remain responsive to dynamic environments.

5.1 Ground Truth Analysis

One issue with building a large perception system like the one described in this paper is that it can be difficult to quantify performance. This is especially true when attempting to compare different test configurations or hyperparameter settings. The chief difficulty lies in the fact that while all configurations might successfully complete test runs, the underlying world models they generated can vary greatly in completeness and quality.

Under typical conditions, LIDAR is the most accurate perception sensor and can be used to generate a pseudo ground truth for the stereo systems. Recall that stereo is useful for a number of processes including terrain classification, roughness detection, negative obstacle detection, and stealthy passive-only operations. To quantify the quality of the stereo pipeline, we therefor developed a tool which compares costmaps generated using different stereo configurations against a pseudo ground truth costmap generated using only LIDAR.

The first step in this evaluation is to take a recording of all of the sensors while the vehicle traverses the test scenario. It is important to run the evaluation strictly from a recording, as we need the sensor inputs to be identical across different configurations. The ground truth costmap is then generated from the recording using a special system configuration which only uses the Velodyne sensor and persists data across the entire world model. An example of a typical ground truth costmap generated in this way is given in Figure 20.

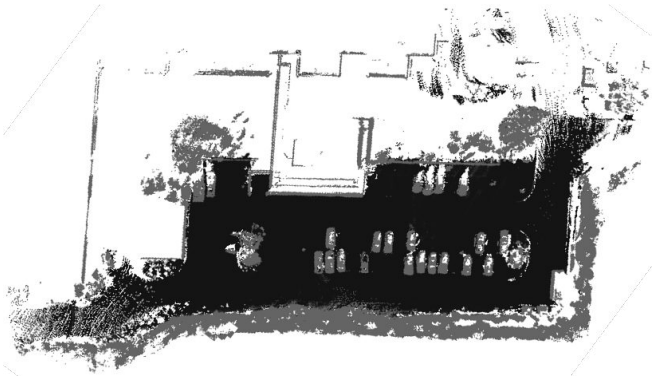


Figure 20. Example of a ground truth costmap constructed by driving the vehicle around a parking lot several times. This allows the composition of a dense map using the highly accurate (but sparse) returns from the Velodyne-16. In this map, black corresponds to drivable ground, gray is obstacles (undrivable areas), and white is unobserved space.

Once the ground truth costmap has been created, different test configurations can be run, for example, with and without LWIR cameras, or using different weightings for adding data into the world model. As these tests are run, metadata for the run is recorded which is later used for comparison. Once all test configurations have been run, an evaluation movie is created (a frame of which is shown in Figure 21) which allows comparison of the accuracy of the costmap over time. This evaluation presents a graphical view of the accuracy of the local costmap, overlaying correctly/incorrectly detected obstacles and ground in different colors. Additionally, it also presents quantitative measures: confusion matrices, precision-recall curves, as well as a frame-by-frame running plot of precision, recall, and F1 scores.

5.2 System Tests

A testing schedule was devised around a 6 month development cycle, with the end of each cycle resulting in a week long integration test. From a software point of view, integration refers to the act of merging a capability previously under development into the baseline repositories. An integration test is designed to test a capability's benefit to the overall system and assess its readiness for integration.

Leading up to an integration test, a capability must first prove its readiness by passing a series of smaller system tests in an integration exercise. Examples of these tests are depicted in Figures 22, 23, and 24. Although seemingly straight forward, these tests provide a good indication of the overall performance of the perception pipeline. In particular, these tests tend to exacerbate stereo occlusion artifacts which often result in the creation of false positives around depth discontinuities. These false positives can create artificial barriers, causing the vehicle to veer off course. Likewise, repeated patterns in the tarps used to construct the congested L tend to confuse the stereo correlation algorithms, which may also result in the creation of false positives. Finally, these tests are conducted at various times of the day under various lighting conditions such that obstacles may be hidden by deep shadows or overexposure caused by sun glare. The current baseline system is able pass each of these tests more than 90% of the time.

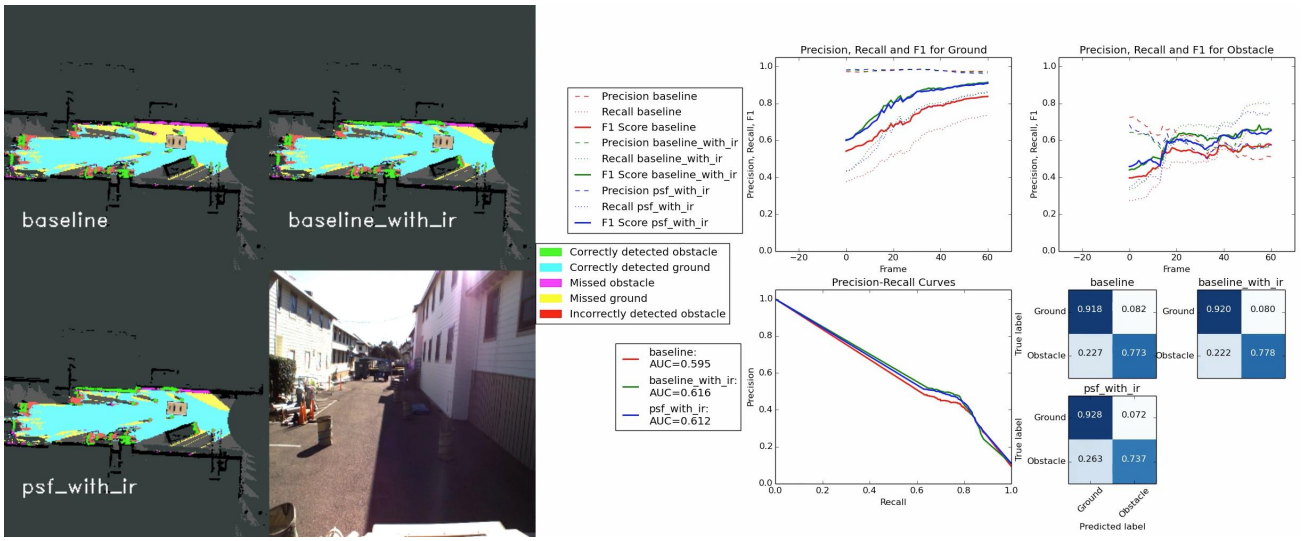


Figure 21. Ground truth analysis tool: (left) Costmap evaluation results from different test configurations as well as an example video frame. This test shows that adding in the IR cameras allows the ground missed in the baseline, color-only configuration to be filled in correctly. Note the yellow areas in front and behind on the right side of the vehicle. (top-right) Frame-by-frame running plot of the current precision, recall, and F1 scores. (bottom-right) Confusion matrices and precision-recall curves for the current map.

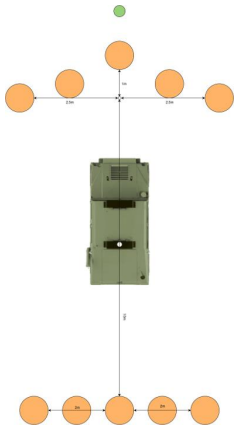


Figure 22. The congested escape tests the ability of the vehicle to perform multi-point turns to drive out of a congested environment composed of 2-3 ft high barrels.

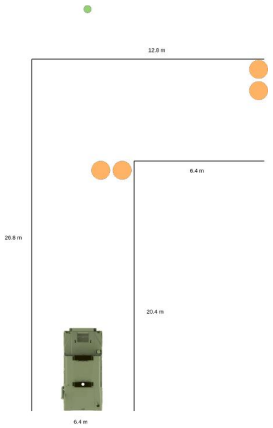


Figure 23. The congested L tests the ability of the vehicle to navigate sharp turns in a congested environment composed of textured tarps and barrels.

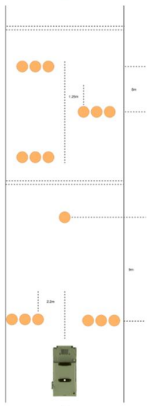


Figure 24. The maneuverability course tests the ability of the vehicle to avoid objects, i.e., barrels, within a constrained corridor.

The integration test incorporates the system tests described above into two large scale nominal autonomy courses. The first course forces the vehicle to navigate between closely spaced buildings while avoiding obstacles as it follows a path through a mock village. The second course forces the vehicle to navigate a 1.5 mile unimproved dirt road while navigating both artificial and natural obstacles. Natural obstacles include ground vegetation, small boulders, low hanging tree branches, and negative obstacles such as sharp drop-offs. To measure the performance of the vehicle on these courses, an automated test tool was created to label and count forced manual overrides, including safety stops and planner timeouts, as well as measure various statistics, including autonomous drive times and distances. The tool is also able to store relevant sensor data at the time of intervention such that it can be analyzed to determine the point of failure. The current state of performance is such that the vehicle can safely navigate both courses with on average 1-2 safety stops or planner timeouts.

6. CONCLUSION AND FUTURE WORK

In this paper, we have described a perception pipeline designed for autonomous ground vehicles operating in expeditionary environments. These environments create unique challenges for autonomy and perception due to the unpredictable and unstructured nature of the terrain and obstacles. In these scenarios, the autonomous system must rely almost entirely on sensor data, since detailed a priori data may not exist. In order to accurately model the environment, it thus becomes important to fuse multiple sensor modalities to mitigate the potential weakness in a single sensor. In this paper, we specifically describe the fusion of color stereo, IR stereo, and LIDAR. Starting with calibration in Section 2, we discussed relevant methods for intrinsic and extrinsic calibration, including a novel approach to IR calibration using active calibration boards. In Section 3, we then discussed various sensor processing techniques including methods for stereo filtering and various algorithms for the segmentation and classification of pixels and point clouds. Next, in Section 4 we discussed our fusion process whereby the individual sensor streams are aggregated in the world model to generate costmaps for producing actionable vehicle trajectories. Finally, in Section 5 we provided results from of our system-level testing indicating the current state of performance.

A variety of capabilities and enhancements are either conceived or currently underway to further push the performance of the MAGV system. Multi-object tracking will allow the vehicle to predict the movement of dynamic objects and thus allow for safe operation among vehicles and pedestrians. Enhanced semantic segmentation based on advances in deep learning will provide greater situational awareness of the terrain and objects within a scene. Finally, advanced planning techniques that leverage multiple data streams and require minimal tuning will produce human like reference trajectories.

REFERENCES

[1] Canepari, Z., Cooper, D., and Cott, E., "Navy robots test the limits of autonomy." New York Times, 6 May 2015 https://www.nytimes.com/2015/05/07/technology/robotica-navy-tests-limits-autonomy.html?_r=0. (Accessed: 14 February 2017).

[2] Zych, N., Silver, D., Stager, D., Green, C., Pilarski, T., Fischer, J., Kuntz, N., Anderson, D., Costa, A., Gannon, J., et al., "Achieving integrated convoys: cargo unmanned ground vehicle development and experimentation," in [*SPIE Defense, Security, and Sensing*], 87410Y–87410Y, International Society for Optics and Photonics (2013).

[3] Urmson, C., "How a driverless car sees the road." TED, March 2015 https://www.ted.com/talks/chris_urmson_how_a_driverless_car_sees_the_road. (Accessed: 14 February 2017).

[4] Aeberhard, M., Rauch, S., Bahram, M., Tanzmeister, G., Thomas, J., Pilat, Y., Homm, F., Huber, W., and Kaempchen, N., "Experience, results and lessons learned from automated driving on germany's highways," *IEEE Intelligent Transportation Systems Magazine* **7**(1), 42–57 (2015).

[5] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al., "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics* **23**(9), 661–692 (2006).

[6] Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., et al., "A perception-driven autonomous urban vehicle," *Journal of Field Robotics* **25**(10), 727–774 (2008).

[7] Urmson, C., Baker, C., Dolan, J., Rybski, P., Salesky, B., Whittaker, W., Ferguson, D., and Darms, M., "Autonomous driving in traffic: Boss and the urban challenge," *AI magazine* **30**(2), 17 (2009).

[8] Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C. G., et al., "Making bertha drivean autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine* **6**(2), 8–20 (2014).

[9] Braid, D., Broggi, A., and Schmiedel, G., "The terramax autonomous vehicle," *Journal of Field Robotics* **23**(9), 693–708 (2006).

[10] Bradski, G. *Dr. Dobb's Journal of Software Tools* .

[11] Zhang, Z., "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence* **22**(11), 1330–1334 (2000).

[12] Segal, A., Haehnel, D., and Thrun, S., "Generalized-icp.," in [*Robotics: science and systems*], **2**(4) (2009).
[13] Lee, D., Rankin, A., Huertas, A., Nash, J., Ahuja, G., and Matthies, L., "Lwir passive perception system for stealthy unmanned ground vehicle night operations," in [*SPIE Defense+ Security*], 98370D–98370D, International Society for Optics and Photonics (2016).
[14] Rankin, A., Bajracharya, M., Huertas, A., Howard, A., Moghaddam, B., Brennan, S., Ansar, A., Tang, B., Turmon, M., and Matthies, L., "Stereo-vision based perception capabilities developed during the robotics collaborative technology alliances program," in [*Proceedings of SPIE*], **7692**, 76920C1–76920C15 (2010).
[15] Labayrade, R., Aubert, D., and Tarel, J.-P., "Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation," in [*Intelligent Vehicle Symposium, 2002. IEEE*], **2**, 646–651, IEEE (2002).
[16] Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A., "On the segmentation of 3d lidar point clouds," in [*Robotics and Automation (ICRA), 2011 IEEE International Conference on*], 2798–2805, IEEE (2011).
[17] Chambers, D. R., Gassaway, J., Goodin, C., and Durst, P. J., "Simulation of a multispectral, multicamera, off-road autonomous vehicle perception system with virtual autonomous navigation environment (vane)," in [*SPIE Security+ Defence*], 964802–964802, International Society for Optics and Photonics (2015).
[18] Konolige, K., Agrawal, M., Blas, M. R., Bolles, R. C., Gerkey, B., Sola, J., and Sundaresan, A., "Mapping, navigation, and learning for off-road traversal," *Journal of Field Robotics* **26**(1), 88–113 (2009).
[19] Ojala, T., Pietikainen, M., and Harwood, D., "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in [*Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*], **1**, 582–585, IEEE (1994).
[20] Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., and Urtasun, R., "Multinet: Real-time joint semantic reasoning for autonomous driving," *arXiv preprint arXiv:1612.07695* (2016).
[21] Simonyan, K. and Zisserman, A., "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556* (2014).
[22] Long, J., Shelhamer, E., and Darrell, T., "Fully convolutional networks for semantic segmentation," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 3431–3440 (2015).
[23] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "ImageNet: A Large-Scale Hierarchical Image Database," in [*CVPR09*], (2009).
[24] OpenStreetMap contributors, "Planet dump retrieved from http://planet.osm.org ." http://www.openstreetmap.org (2017).
[25] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W., "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots* **34**(3), 189–206 (2013).
[26] Moravec, H. and Elfes, A., "High resolution maps from wide angle sonar," in [*Robotics and Automation. Proceedings. 1985 IEEE International Conference on*], **2**, 116–121, IEEE (1985).
[27] Heng, L., Honegger, D., Lee, G. H., Meier, L., Tanskanen, P., Fraundorfer, F., and Pollefeys, M., "Autonomous visual mapping and exploration with a micro aerial vehicle," *Journal of Field Robotics* **31**(4), 654–675 (2014).
[28] Velodyne LiDAR, *VLP-16 Velodyne LiDAR Puck user's manual and programming guide*. (Accessed: 27 Febrary 2017.