

Multi-Modal Obstacle Detection in Unstructured Environments with Conditional Random Fields

Mikkel Kragh¹ and James Underwood²

Abstract

Reliable obstacle detection and classification in rough and unstructured terrain such as agricultural fields or orchards remains a challenging problem. These environments involve large variations in both geometry and appearance, challenging perception systems that rely on only a single sensor modality. Geometrically, tall grass, fallen leaves, or terrain roughness can mistakenly be perceived as non-traversable or might even obscure actual obstacles. Likewise, traversable grass or dirt roads and obstacles such as trees and bushes might be visually ambiguous.

In this paper, we combine appearance- and geometry-based detection methods by probabilistically fusing lidar and camera sensing using a conditional random field. We apply a state-of-the-art multi-modal fusion algorithm from the scene analysis domain and adjust it for obstacle detection in agriculture with moving ground vehicles. This involves explicitly handling sparse point cloud data and exploiting both spatial, temporal, and multi-modal links between corresponding 2D and 3D regions.

The proposed method is evaluated on a diverse dataset, comprising a dairy paddock and a number of different orchards gathered with a perception research robot in Australia. Results show that for a two-class classification problem (ground and non-ground), only the camera leverages from information provided by the other modality. However, as more classes are introduced (*ground, sky, vegetation, and object*), both modalities complement each other and improve the mean classification score. Further improvement is achieved by introducing recursive inference with temporal links between successive frames.

Keywords

Obstacle Detection, Sensor Fusion, Field Robots, Agriculture

1 Introduction

In recent years, automation in the automotive industry has expanded rapidly with products ranging from assisted-driving features to semi-autonomous cars that are fully self-driven in certain restricted

circumstances. Currently, the technology is limited to handle only very structured environments in clear conditions. However, frontiers are constantly pushed, and in the near future, fully autonomous cars will emerge that both detect and differentiate between objects and structures in their surroundings at all times.

In agriculture, automated steering systems have existed for around two decades (Abidine et al. 2004). Farmland is an explicitly constructed environment, which permits recurring driving patterns. Therefore, exact route plans can be generated and followed to centimeter precision using accurate global navigation

¹ Department of Engineering, Aarhus University, Denmark

² Australian Centre for Field Robotics, The University of Sydney

Corresponding author:

Mikkel Kragh, Department of Engineering, Aarhus University, Denmark

Email: mkha@eng.au.dk

systems. A few R&D projects exist within companies that try to incorporate obstacle detection and avoidance (Case IH 2016; ATC 2016; ASI 2016). However, as of today, no commercial products exist on the market. In order to fully eliminate the need for a human driver, the vehicles need to perceive the environment and automatically detect and avoid obstacles under all operating conditions. Unlike self-driving cars, farming vehicles further need to handle unknown and unstructured terrain and need to distinguish traversable vegetation such as crops and high grass from actual obstacles, although both protrude from the ground. These strict requirements are often addressed by introducing multiple sensing modalities and sensor fusion, thus increasing detection performance, solving ambiguities, and adding redundancy. Typical sensors are monocular and stereo color cameras, thermal camera, radar, and lidar. Due to the difference in their physical sensing, the detection capabilities of these modalities both complement and overlap each other (Peynot et al. 2010; Brunner et al. 2013).

A number of approaches have been made to combine multiple modalities for obstacle detection in agriculture. Self-supervised systems have been proposed for stereo-radar (Reina et al. 2016a), rgb-radar (Milella et al. 2015, 2014), and rgb-lidar (Zhou et al. 2012). Here, one modality is used to continuously supervise and improve the detection results of the other. In contrast, actual sensor fusion provides reduced uncertainty when combining multiple sensors as opposed to applying each sensor individually. A distinction is often made between low-level (early) fusion, combining raw data from different sensors, and high-level (late) fusion, integrating information at decision level. At low-level, lidar has been fused with other range-based sensors (lidar and radar) using a joint calibration procedure (Underwood et al. 2010). Additionally, lidar has been fused with cameras (monocular, stereo, and thermal) by projecting 3D lidar points onto corresponding images and concatenating either their raw outputs (Dima et al. 2004; Wellington et al. 2005) or pre-calculated features (Häselich et al. 2013). This approach potentially leverages the full potential of all sensors, but suffers from the fact that

only regions covered by all modalities are defined. Furthermore, it assumes perfect extrinsic calibration between the sensors involved. At high-level, lidar and camera have been fused for ground/non-ground classification, where the idea is to simply weight the a posteriori outputs of individual classifiers by their prior classification performances (Reina et al. 2016b). Another approach combines lidar and camera in grid-based fusion for terrain classification into four classes, where again a weighting factor is used for calculating a combined probability for each cell (Laible et al. 2013). A similar approach uses occupancy grid mapping to combine lidar, radar, and camera by probabilistically fusing their equally weighted classifier outputs (Kragh et al. 2016). However, none of the above methods truly leverage the potentially complex connections between sensor technologies and their detection capabilities across object classes. One sensor may recognize class A but confuse B and C, whereas another sensor may recognize C but confuse A and B. By learning this relationship, the sensors can be fused to effectively distinguish all three classes.

Within the domain of scene analysis, lidar and camera have recently been combined with success to learn these relationships and improve classification accuracy of semantic segmentation. In these approaches, a common setup is to acquire synchronized camera and lidar data from a side-looking ground vehicle passing by a scene. A camera takes images with a certain frequency, and a single-beam vertically-scanning laser is used in a push-broom setting, allowing subsequent accumulation of points into a combined point cloud. By looking at an area covered by both modalities, a scene consisting of a high number of 3D points and corresponding images is then post-processed, either by directly concatenating features of both modalities at low-level (Namin et al. 2014; Posner et al. 2009; Douillard et al. 2010; Cadena and Košecká 2016), or by fusing intermediate classification results provided by both modalities individually at high-level (Namin et al. 2015; Xiao et al. 2015; Zhang et al. 2015; Munoz et al. 2012). For this purpose, conditional random fields (CRFs) are often used, as they provide an efficient

and flexible framework for including both spatial, temporal, and multi-modal relationships.

In this paper, we adapt the offline fusion algorithm of Namin et al. (2015) and adjust it for online applicable obstacle detection in agriculture with a moving ground vehicle. This involves explicitly handling sparse point cloud data and exploiting both spatial, temporal, and multi-modal links between corresponding 2D and 3D regions. We combine appearance- and geometry-based detection methods by probabilistically fusing lidar and camera sensing using a CRF. Visual information from a color camera serves to classify visually distinctive regions, whereas geometric information from a lidar serves to distinguish flat, traversable ground areas from protruding elements. We further investigate a traditional computer vision pipeline and deep learning, comparing the influence on sensor fusion performance. The proposed method is evaluated on a diverse dataset of agricultural orchards (mangoes, lychees, apples, and almonds) and a dairy paddock gathered with a perception research robot.

The main contributions of the paper are threefold:

- Adaptation of an offline sensor fusion method used for scene analysis to an online applicable method used for obstacle detection. This involves extending the framework with temporal links between successive frames, utilizing the localization system of the robot.
- Comparison of sensor fusion performance when using traditional computer vision and deep learning.
- Evaluation of multi-modal obstacle detection in various agricultural environments. This involves looking at how features and classifiers generalize across different environments, and how the number of classes (binary vs. multiclass) affects the improvement when fusing multiple modalities.

The paper is divided into 5 sections. Section 2 presents the proposed approach including initial classifiers for the camera and the lidar, individually, and a CRF for fusing the two modalities. Section 3 presents the experimental platform and datasets, followed by experimental results in section 4.

Ultimately, section 5 presents a conclusion and future work.

2 Approach

Our method works by jointly inferring optimal class labels of 2D segments in images and 3D segments in corresponding point clouds. By first training individual, initial classifiers for the two modalities, we use a CRF for combining the information using the perspective projection of 3D points onto 2D images. This provides pairwise edges between 2D and 3D segments, thus allowing one modality to correct the initial classification result of the other. Clustering of 2D pixels into 2D segments and 3D points into 3D segments is necessary in order to reduce the number of nodes in the CRF graph structure.

A schematic overview of the algorithm is shown in Figure 1. A synchronized image and point cloud are fed into a pipeline, where feature extraction, segmentation and an initial classification are performed for each modality. 3D segments from the point cloud are then projected onto the 2D image, and a CRF is trained to fuse the two modalities. Finally, recursive inference is introduced to the CRF by adding temporal links to the previous frame, utilizing the localization system of the robot.

In the following subsections, the 2D and 3D classifiers are first described individually. The CRF fusion algorithm is then explained in detail.

2.1 2D Classifier

Most approaches combining lidar and camera use traditional computer vision with hand-crafted image features for the initial 2D classification (Douillard et al. 2010; Cadena and Košecká 2016; Namin et al. 2015; Xiao et al. 2015; Zhang et al. 2015; Munoz et al. 2012). However, recent advances with self-learned features using deep learning have outperformed the traditional approach for many applications. In this paper, we therefore compare the two approaches and evaluate their influence when fusing image and lidar data.

The **traditional computer vision** pipeline consists of three steps: the image is first segmented,

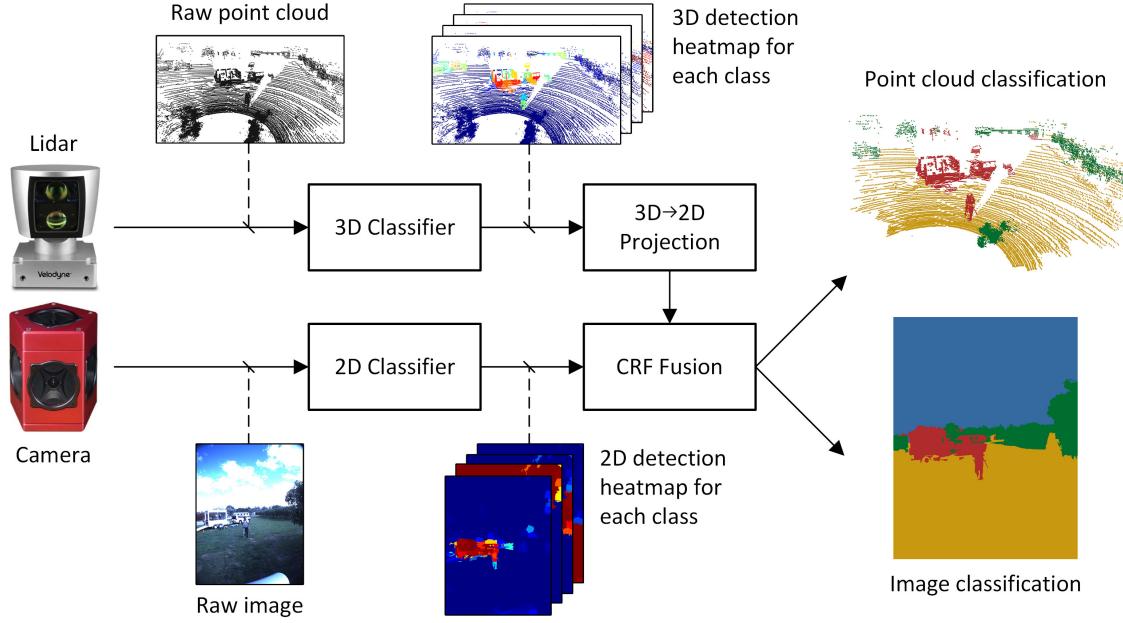


Figure 1. Schematic overview of fusion algorithm.

features are then extracted for each segment, and a classifier is finally trained to distinguish a number of classes based on the features. In our case, we segment the image into superpixels using SLIC (Achanta et al. 2012). Figure 2a shows an example of this segmentation. For each superpixel, average RGB values, GLCM features (energy, homogeneity and contrast) (Haralick et al. 1973) and a histogram of SIFT features (Lowe 2004) are extracted. The histogram of SIFT features uses a bag-of-words (BoW) representation built using all images in the training set. Dense SIFT features are calculated over the image, and a histogram of word occurrences is generated for each superpixel. All features are then normalized by subtracting the mean and dividing by the standard deviation across the training set. Finally, they are used to train a support vector machine (SVM) (Wu et al. 2004) classifier with probability estimates using a one-against-one approach with the libsvm library (Chang and Lin 2011). This provides probability estimates $P_{\text{initial}}(x_i^{2D} | \mathbf{z}_i^{2D})$ of class label x_i^{2D} , given the

features \mathbf{z}_i^{2D} of superpixel i . An example heatmap of an *object* class is visualized in Figure 2b.

In recent years, **deep learning** has been used extensively for various machine learning problems. Especially for image classification and semantic segmentation, convolutional neural networks (CNNs) have outperformed traditional image recognition methods and are today considered state-of-the-art (Krizhevsky et al. 2012; He et al. 2015; Long et al. 2015). In this paper, we use a CNN for semantic segmentation (per-pixel classification) proposed by Long et al. (2015). As we have a very limited amount of training data available, we use a model pre-trained on the PASCAL-Context dataset (Mottaghi et al. 2014). This includes 59 general classes, of which only a few map directly to the 9 classes present in our dataset (see section 3.2). For the remaining classes, we remap such that all objects (bottle, table, chair, computer, etc.) map to a common *object* class, and all traversable surfaces (grass, ground, floor, road, etc.) map to a common *ground* class. We then maintain the 59 classes of the

pre-trained model, and finetune on the overlapping class labels from our annotated dataset. In this way, we preserve the ability of the pre-trained network to recognize general object classes (humans, buildings, vehicles, etc.), but use our own data for optimizing the weights towards the specific camera, illumination conditions, and agricultural environment used in our setup.

The softmax layer of the CNN provides per-pixel probability estimates for each object class. However, in this paper, class probability estimates are needed for each superpixel. We therefore use the same superpixel segmentation as for the traditional vision pipeline, and average and normalize per-pixel estimates within each superpixel. An example heatmap of an *object* class is visualized in Figure 2c.

2.2 3D Classifier

When classifying individual points in a point cloud, the point density and distribution influence the attainable classification accuracy, but also the method of choice for feature extraction. Point features are calculated using a local neighborhood around each point. Traditionally, this is accomplished with a constant neighborhood size (Wellington et al. 2005; Hebert and V 2003; Lalonde et al. 2006; Quadros et al. 2012). For a single-beam laser accumulating points in a push-broom setting, this procedure works fine, as the point distribution is roughly constant, resulting in a dense point cloud. For a rotating, multi-beam lidar generating a single scan, however, the point density varies with distance, resulting in a sparse point cloud. Using a constant neighborhood size in this case, results in either a low resolution close to the sensor or noisy features at far distance. Therefore, in this paper, we use an adaptive neighborhood size depending on the distance between each point and the sensor. This ensures high resolution at short distance and prevents noisy features at far distance. We use the method from Kragh et al. (2015) where the neighborhood size scales linearly with the sensor distance. The intuition behind this relationship assumes a flat ground surface beneath the sensor, such that points from a single, rotating beam pointing towards the ground

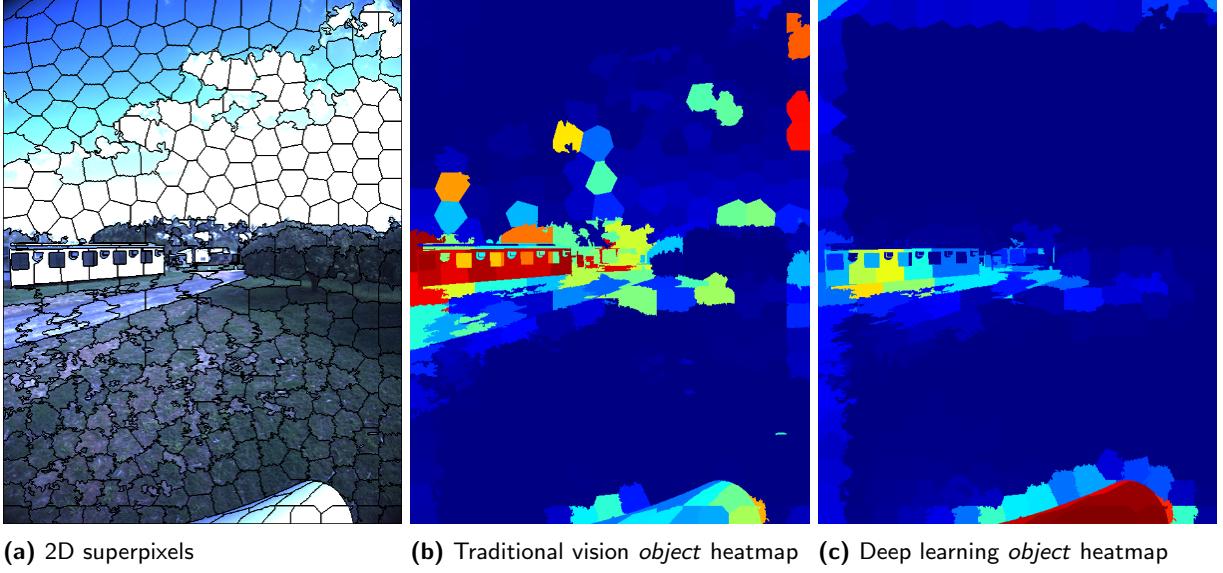
are distributed equally along a circle. Figure 3a illustrates this circle along with a top-down view in Figure 3b. The radius $\|\mathbf{p}\|_{xy}$ corresponds to the distance in the ground plane between the sensor and a point \mathbf{p} . The distance between any two neighboring points on the circle is thus $2\|\mathbf{p}\|_{xy} \sin \frac{\theta_H}{2}$ where θ_H is the horizontal angle difference (angular resolution). In order to achieve a neighborhood (gray area) with M points on a single beam, the neighbourhood radius must be:

$$r = 2\|\mathbf{p}\|_{xy} \sin \frac{M\theta_H}{4} \quad (1)$$

which scales linearly with $\|\mathbf{p}\|_{xy}$. This relationship holds only for single laser beams. However, since the angular resolution for a multi-beam lidar is normally much higher horizontally than vertically, the relationship still serves as a good approximation.

The point cloud is first preprocessed by aligning the xy -plane with a globally estimated plane using the RANSAC algorithm (Fischler and Bolles 1981). This transformation makes the resulting point cloud have an approximately vertically oriented z -axis. Using the adaptive neighborhood, 9 features related to height, shape, and orientation are then calculated for each point (Kragh et al. 2015). f_1-f_4 are height features. f_1 is simply the z -coordinate of the evaluated point, whereas f_2 , f_3 , and f_4 denote the minimum, mean and variance of all z -coordinates within the neighborhood, respectively. f_5-f_7 are shape features calculated with principal component analysis. As eigenvalues of the 3×3 covariance matrix, they describe the distribution of the neighborhood points (Lalonde et al. 2006). f_8 is the orientation of the eigenvector corresponding to the largest eigenvalue. It serves to distinguish horizontal and vertical structures (e.g. a ground plane and building). Finally, f_9 denotes the reflectance intensity of the evaluated point, provided directly by the lidar sensor utilized in the experiments. Since the size of the neighborhood varies with distance, all features are made scale-invariant.

As for the 2D features, an SVM classifier with probability estimates is trained to provide per-point class probabilities. A segmentation procedure then clusters points into supervoxels by minimizing both spatial distance and class probability difference



(a) 2D superpixels (b) Traditional vision *object* heatmap (c) Deep learning *object* heatmap

Figure 2. Example of 2D segmentation, and probability estimates for traditional vision and deep learning.

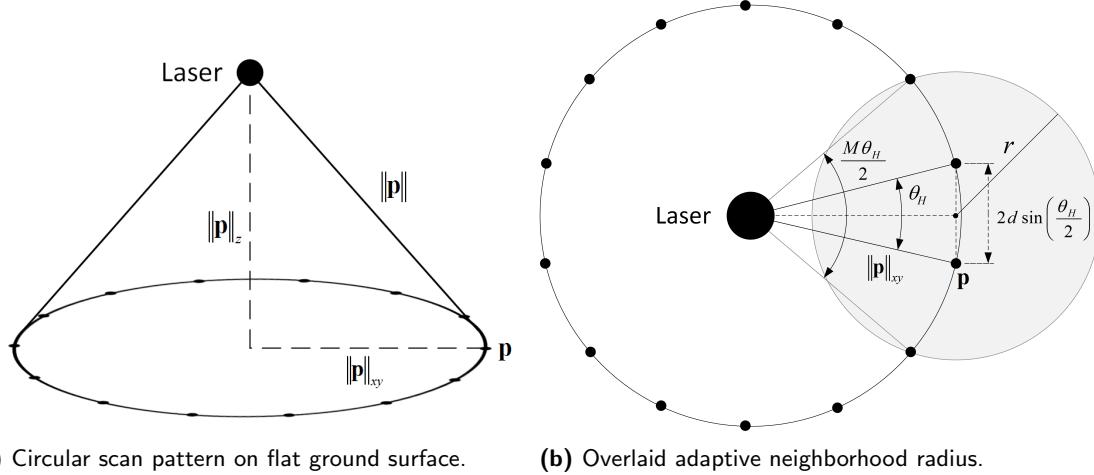


Figure 3. Example of adaptive neighborhood radius for single-beam lidar with $M = 4$.

between segments. Our method uses the approach by Papon et al. (2013) where voxels are clustered iteratively. However, we modify the feature distance measure D between neighboring segments:

$$D = \lambda D_s + \chi^2 \quad (2)$$

where D_s is the spatial euclidean distance between two segments, χ^2 is the Chi-Squared histogram distance (Pele and Werman 2010) between their mean histograms of probability estimates, and $\lambda > 0$ is a weighting factor. By minimizing this measure during the clustering procedure, points are grouped together based on their spatial distance and initial probability

estimates. Each segment i is then given a probability estimate $P_{\text{initial}}(x_i^{3D} | \mathbf{z}_i^{3D})$ by averaging the class probabilities of all points within the segment. Finally, edges between adjacent segments are stored. Figure 4 shows a probability output example of a single class (*object*), the segmented point cloud and its supervoxel edges connecting the segment centers.

Using the extrinsic parameters defining the pose of the lidar and the camera, the point cloud can be projected onto the image using a perspective projection. The extrinsic parameters are given by the solid CAD model of the platform including sensors and refined using an unsupervised calibration method for cameras and lasers (Levinson and Thrun 2013). For computational purposes, the projected points are distorted according to the intrinsics of the camera instead of undistorting the image. Figure 5 illustrates the projected point cloud, pseudo-coloring points by their associated 3D segments. Edges between 2D and 3D segments are then defined by the overlap (see section 2.3.2), resulting in single 2D segments mapping to multiple 3D segments and vice versa.

2.3 Conditional Random Field

Once initial probability estimates of all 2D and 3D segments have been found and their edges defined, an undirected graphical model similar to the one visualized in Figure 6 can be constructed. Each 2D and 3D segment (superpixel and supervoxel) is assigned a node in the graph, and edges between the nodes are defined as described in the sections above. In Figure 6, additional recursive edges are shown between frame f and $f - 1$. These serve as temporal links between 3D nodes in subsequent frames.

A CRF directly models the conditional probability distribution $p(\mathbf{x} | \mathbf{z})$, where the hidden variables \mathbf{x} represent the class labels of nodes and \mathbf{z} represent the observations/features. The conditional distribution can be written as:

$$p(\mathbf{x} | \mathbf{z}) = \frac{1}{Z(\mathbf{z})} \exp(-E(\mathbf{x} | \mathbf{z})) \quad (3)$$

where $Z(\mathbf{z})$ is the partition (normalization) function and $E(\mathbf{x} | \mathbf{z})$ is the Gibbs energy. Considering a pairwise CRF for the above graph structure, this

energy can be written as:

$$\begin{aligned} E(\mathbf{x} | \mathbf{z}) = & \sum_{i=1}^{N^{2D}} \phi_i^{2D} + \sum_{i=1}^{N^{3D}} \phi_i^{3D} + \sum_{i,j \in E^{2D}} \psi_{ij}^{2D} \\ & + \sum_{i,j \in E^{3D}} \psi_{ij}^{3D} + \sum_{i,j \in E^{2D-3D}} \psi_{ij}^{2D-3D} + \sum_{i,j \in E^{\text{Time}}} \psi_{ij}^{\text{Time}} \end{aligned} \quad (4)$$

where ϕ_i^{2D} and ϕ_i^{3D} are unary potentials, N^{2D} and N^{3D} are the number of 2D and 3D nodes, ψ_{ij}^{2D} , ψ_{ij}^{3D} , ψ_{ij}^{2D-3D} and ψ_{ij}^{Time} are pairwise potentials, and E^{2D} , E^{3D} , E^{2D-3D} and E^{Time} are edges. For simplicity, function variables and weights for the unary and pairwise potentials are left out but explained in more detail in the following sections.

2.3.1 Unary Potentials The unary potentials for 2D and 3D segments are defined by the negative logarithm of their initial class probabilities. This ensures that the conditional probability distribution in equation 3 will correspond exactly to the probability distribution of the initial classifiers if no pairwise potentials are present:

$$\phi_i^{2D}(x_i^{2D}, \mathbf{z}_i^{2D}) = -\log(P_{\text{initial}}(x_i^{2D} | \mathbf{z}_i^{2D})) \quad (5)$$

$$\phi_i^{3D}(x_i^{3D}, \mathbf{z}_i^{3D}) = -\log(P_{\text{initial}}(x_i^{3D} | \mathbf{z}_i^{3D})) \quad (6)$$

where \mathbf{z}_i^{2D} and \mathbf{z}_i^{3D} are the 2D and 3D features described above, and x_i^{2D} and x_i^{3D} are the class labels. The potentials describe the cost of assigning label x to the i 'th 2D or 3D segment. If the probability estimate of the initial classifier is close to 1, the cost is low, whereas if the probability is close to 0, the cost is high.

For unary potentials, no CRF weights are included, since we assume class imbalance to be handled by the initial classifiers.

2.3.2 Pairwise Potentials In equation 4, three different types of pairwise potentials and edges appear. These are 2D edges between neighboring 2D superpixel nodes, 3D edges between neighboring 3D supervoxel nodes, 2D-3D edges connecting 2D and 3D nodes through the perspective projection, and recursive edges connecting subsequent frames.

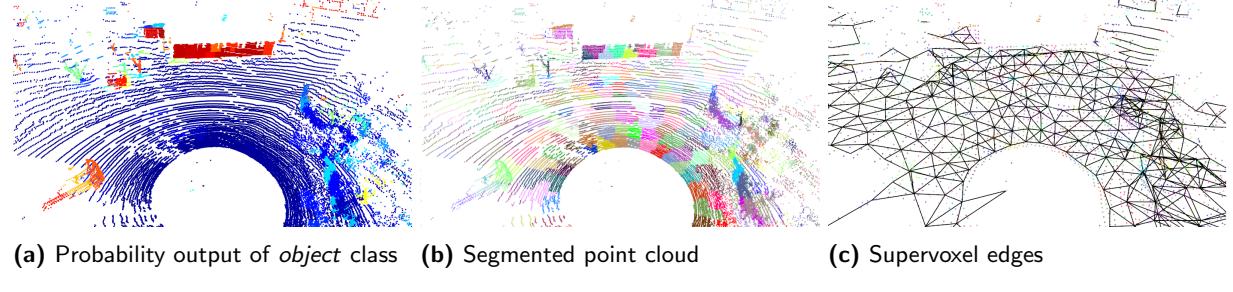


Figure 4. Example of 3D classification, segmentation, and edge construction.



Figure 5. Projection of 3D segments onto 2D superpixels.

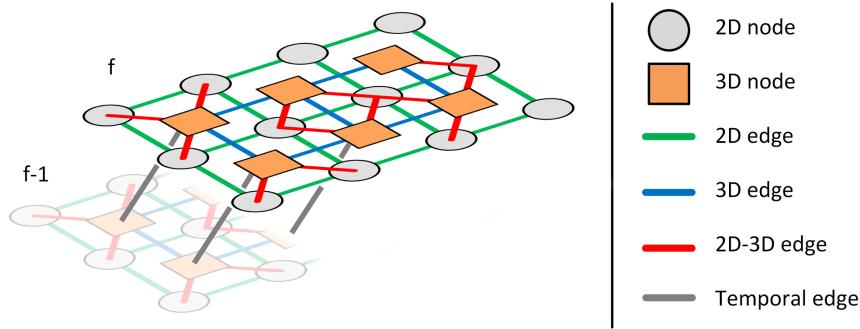


Figure 6. CRF graph with 2D nodes (superpixels), 3D nodes (supervoxels), and edges between them both spatially and temporally.

2D and 3D edges The pairwise potentials for neighboring 2D or 3D segments act as smoothing terms by introducing costs for assigning different labels. As is common for 2D segmentation and classification, the cost depends on the exponentiated distance between the two neighbors, such that a small distance will incur a high cost and vice versa (Boykov and Jolly 2001; Krähenbühl and Koltun 2012). In 2D,

the distance is in RGB-space:

$$\psi_{ij}^{2D}(x_i^{2D}, x_j^{2D}, \mathbf{z}_i^{2D}, \mathbf{z}_j^{2D}) = w_p^{2D}(x_i^{2D}, x_j^{2D}) \cdot \delta(x_i^{2D} \neq x_j^{2D}) \exp\left(-\frac{|I_i - I_j|^2}{2\sigma_{2D}^2}\right) \quad (7)$$

where I_i is the RGB-vector for superpixel i and σ_{2D} is a weighting factor trained with cross-validation. w_p^{2D} is a weight matrix. It is learned during training and represents the importance of the pairwise potentials. The matrix is symmetric and class-dependent, such that interactions between classes are taken into

account. As is common for pairwise potentials, an indicator function (delta function) ensures that the potential is zero for neighboring segments that are assigned the same label.

In 3D, the cost depends on the difference between plane normals (Hermans et al. 2014; Namin et al. 2015):

$$\psi_{ij}^{3D}(x_i^{3D}, x_j^{3D}, \mathbf{z}_i^{3D}, \mathbf{z}_j^{3D}) = w_p^{3D}(x_i^{3D}, x_j^{3D}) \cdot \delta(x_i^{3D} \neq x_j^{3D}) \exp\left(-\frac{|\theta_i - \theta_j|^2}{2\sigma_{3D}^2}\right) \quad (8)$$

where θ_i is the angle between the vertical z-axis and the locally estimated plane normal for supervoxel i and σ_{3D} is a weighting factor trained with cross-validation. The angle is calculated as $\theta = \cos^{-1}(f_8)$ (see section 2.2). Similar to 2D, the weight matrix \mathbf{w}_p^{3D} is symmetric and class-dependent.

2D-3D edges The pairwise potential for 2D and 3D segments connected through the perspective projection is defined by their area of overlap as in Namin et al. (2015). Let S_i^{2D} denote the set of pixels in 2D segment i , and let $S_j^{3D \rightarrow 2D}$ denote the set of pixels intersected by the projection of 3D segment j onto the image. Then, we first define a weight $\omega(S_i^{2D}, S_j^{3D})$ as the cardinality (number of elements) of the intersection of the two sets:

$$\omega(S_i^{2D}, S_j^{3D}) = |S_i^{2D} \cap S_j^{3D \rightarrow 2D}| \quad (9)$$

Effectively, this describes the area of overlap between a 2D segment i and a projected 3D segment j . The pairwise potential is then calculated by normalizing this weight by the maximum weight across all 2D segments that are overlapped by the projected 3D segment j :

$$\psi_{ij}^{2D-3D}(x_i^{2D}, x_j^{3D}, \mathbf{z}_i^{2D}, \mathbf{z}_j^{3D}) = w_p^{2D-3D}(x_i^{2D}, x_j^{3D}) \cdot \delta(x_i^{2D} \neq x_j^{3D}) \frac{\omega(S_i^{2D}, S_j^{3D})}{\max_{k \in E_j^{2D-3D}} \omega(S_k^{2D}, S_j^{3D})} \quad (10)$$

where k denotes a 2D segment in the set of all edges E_j^{2D-3D} generated during the projection of 3D segment j onto the image. Using this definition of

the pairwise potential between 2D and 3D segments, we introduce a cost of assigning corresponding 2D and 3D nodes with different class labels. The cost depends on the overlap between the segments, such that a large overlap will result in a high cost, and vice versa. The normalization in equation 10 ensures that the weights for associating a 3D node to multiple 2D nodes sums to 1. However, it does not guarantee the opposite. The sum of weights for associating a 2D node to multiple 3D nodes can thus in theory take any positive value.

Similar to 2D and 3D edges, the weight matrix for 2D-3D edges \mathbf{w}_p^{2D-3D} is class-dependent. However, since the potential concerns different domains (2D and 3D), the weights are made asymmetric as in Winn and Shotton (2006). That is, the cost of assigning x_i^{2D} to class A and x_i^{3D} to class B might not be the same as the other way around. This allows for interactions that depend on both class label and sensor technology.

Recursive edges Recursive inference adds a temporal link from the current frame to a previous frame. By utilizing the localization system of the robot, the location of 3D nodes in a previous frame f_p are transformed from the sensor frame into the world frame. From here, they are then transformed into the current frame f_c where they will likely overlap with the same observed structures. Effectively, this adds another view point to the sensors and can thus help solve potential ambiguities. The extrinsic parameters defining the transformation from the navigation frame (localization system) to the sensor frame (lidar) are given by the CAD model of the platform and refined using an extrinsic calibration method for range-based sensors (Underwood et al. 2010). In the CRF, recursive inference introduces another pairwise potential:

$$\begin{aligned} \psi_{ij}^{\text{Time}}(x_{i,f_c}^{3D}, x_{j,f_p}^{3D}, \mathbf{p}_{i,f_c}^{3D}, \mathbf{p}_{j,f_p}^{3D}) &= \\ w_p^{\text{Time}}(x_{i,f_c}^{3D}, x_{j,f_p}^{3D}) \delta(x_{i,f_c}^{3D} \neq x_{j,f_p}^{3D}) & \\ \cdot \exp\left(-\frac{|\mathbf{p}_{i,f_c}^{3D} - T_{f_p}^{f_c}(\mathbf{p}_{j,f_p}^{3D})|^2}{2\sigma_{\text{Time}}^2}\right) & \end{aligned} \quad (11)$$

where x_{i,f_c}^{3D} is the label and \mathbf{p}_{i,f_c}^{3D} is the 3D position of 3D node i in the current frame f_c , $T_{f_p}^{f_c}$ is the transformation from the previous frame f_p to the current f_c , and σ_{Time} is a weighting factor trained with cross-validation. The transformation is provided by the localization system of the robot. The potential thus depends on the euclidean distance between a 3D node in the current frame and a transformed 3D node in a previous frame, such that a cost is introduced for assigning different labels at the same 3D location. Only 3D nodes can be transformed, as 2D nodes do not have a 3D position. However, since 3D nodes in a previous frame are connected with corresponding 2D nodes, 2D information is indirectly carried on to subsequent frames as well. Similar to 2D and 3D edges, the weight matrix for temporal edges $\mathbf{w}_p^{\text{Time}}$ is symmetric and class-dependent.

During training, only frame f_c is annotated. All nodes (2D and 3D) from the previous frame f_p therefore have unknown labels. We therefore marginalize out these unobserved nodes and only optimize based on nodes from frame f_c .

2.3.3 Training and Inference During training, the CRF weights $\mathbf{w} = [\mathbf{w}_p^{2D}, \mathbf{w}_p^{3D}, \mathbf{w}_p^{2D-3D}, \mathbf{w}_p^{\text{Time}}]$ are estimated with maximum likelihood estimation. Additionally, bias weights are introduced for all pairwise terms to account for tendencies independent of the features. To avoid overfitting, we use L_2 -regularization for all non-bias weights. Since the graph is cyclic, exact inference is intractable and loopy belief propagation is therefore used for approximate inference. The same applies at test time for decoding. The decoding procedure seeks to determine the most likely configuration of class labels by minimizing the energy $E(\mathbf{x} | \mathbf{z})$. The energy can thus be seen as a cost for choosing the label sequence \mathbf{x} given all measurements \mathbf{z} .

3 Experimental Platform and Datasets

3.1 Platform

The experimental research platform in Figure 7 has been used to collect data from various locations in Australia. The robotic platform is based on a Segway RMP 400 module and has a localization system

consisting of a Novatel SPAN OEM3 RTK-GPS/INS with a Honeywell HG1700 IMU, providing accurate 6-DOF position and orientation estimates. A Point Grey Ladybug 3 panospheric camera system with 6 cameras and a Velodyne HDL-64E lidar both cover a 360° horizontal view around the vehicle recording synchronized images and point clouds.

Since this paper focuses on obstacle detection, only the forward-facing camera and the corresponding overlapping part of the point clouds are used for the evaluation.

3.2 Datasets

From May to December 2013, data were collected across different locations in Australia. The diverse datasets include recordings from both a dairy paddock and orchards with mangoes, lychees, apples, and almonds. Figure 8 illustrates a few examples from the forward-facing Ladybug camera during the recordings. Various objects/obstacles such as humans, cows, buildings, vehicles, trees, and hills are present in the datasets. A total of 103 frames have been manually annotated per-pixel in 2D images and per-point in 3D point clouds. By annotating both modalities separately, we can evaluate non-overlapping regions and get reliable ground truth data even if there is a slight calibration error between the two modalities. 9 categories are defined (*ground*, *sky*, *vegetation*, *building*, *vehicle*, *human*, *animal*, *pole*, and *other*). Due to the physics of the lidar, *sky* is only present in the images. Table 1 presents an overview of the datasets.

4 Experimental Results

To evaluate the proposed algorithm, a number of experiments were carried out on the datasets presented in Table 1. First, the overall results are presented by evaluating the improvement in classification when introducing the fusion algorithm. Then, we specifically address binary and multiclass scenarios, compare traditional vision with deep learning, and evaluate the transferability of features and classifiers across domains (*mangoes*, *lychees*, *apples*, *almonds*, and *dairy*).



Figure 7. Robotic platform “Shrimp” with lidar, panospheric camera, and navigation system.

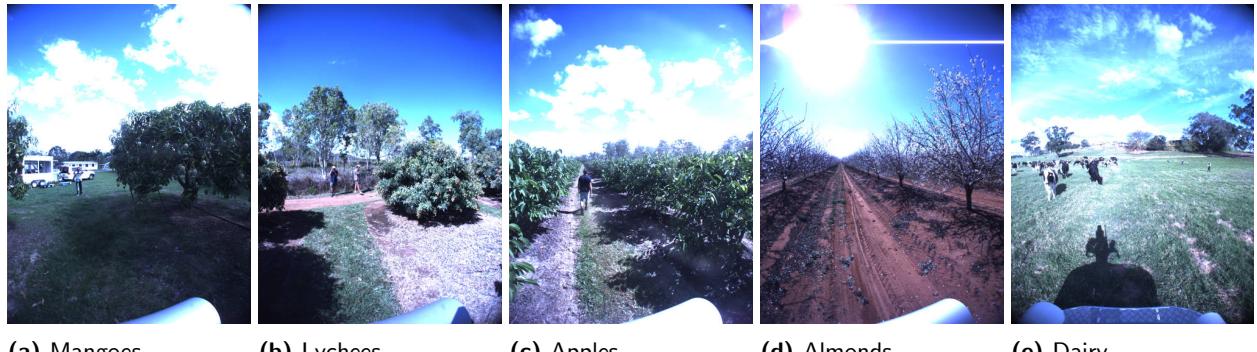


Figure 8. Example images from datasets.

Table 1. Dataset overview.

Dataset	Environment	Season	Frames	Obstacles*
Mangoes	Orchard	Summer	36	Buildings, trailer, cars, tractor, boxes, humans
Lychees	Orchard	Summer	15	Buildings, trailers, cars, humans, iron bars
Apples	Orchard	Summer	23	Trailer, car, humans, poles
Almonds	Orchard	Spring	14	Buildings, cars, humans, dirt pile, plate
Dairy	Field	Winter	15	Humans, hills, poles, cows

* All frames contain ground and vegetation (trees).

To obtain sufficient training examples for each class, the categories *building*, *vehicle*, *human*, *animal*, *pole* and *other* were all mapped to a

common *object* class. A total of four classes were thus used for the following experiments, $x_i = \{\text{ground}, \text{sky}, \text{vegetation}, \text{object}\}$. For all experiments,

5-fold cross-validation was used corresponding to the 5 different datasets in Table 1. That is, for each dataset, data from the remaining four datasets were used for training initial classifiers and CRF weights. This was done to test the system in the more challenging but realistic scenario, where training data is not available for the identical conditions as where the system would be deployed.

For image classification and CRF training and decoding, we used MATLAB along with the computer vision library VLFeat (Vedaldi and Fulkerson 2008), and the undirected graphical models toolbox UGM (Schmidt 2007). For point cloud classification, we used C++ and Point Cloud Library (PCL) (Rusu and Cousins 2011). A list of parameter settings for all algorithms is available in Appendix A.

4.1 Results Overview

Table 2 presents the results for applying the CRF with the three different types of pairwise potentials enabled. *Initial*, CRF_{2D} , and CRF_{3D} thus refer to single-modality results obtained with the direct output of the initial 2D or 3D classifier and the “smoothed” version of the CRF, respectively. CRF_{2D-3D} additionally introduces sensor fusion by adding edges across the two modalities, while $CRF_{2D-3D, Time}$ further adds temporal links across subsequent frames. The results are presented in terms of intersection over union (IoU) and accuracy. Both measures were evaluated per-pixel in 2D and per-point in 3D. Results were obtained with the traditional vision classifier (instead of the deep learning variant) for 2D as it provided the better fusion results. A detailed comparison of traditional vision and deep learning is described in section 4.3.

From Table 2, we see a gradual improvement in classification performance when introducing more terms in the CRF. First, the initial classifiers for 2D and 3D were improved separately by adding spatial links between neighboring segments. This caused an increase in mean IoU of 6.9% in 2D and 6.6% in 3D. Then, by introducing multi-modal links between 2D and 3D, the performance was further increased. In 2D, the increase in mean IoU was only 2.0%, whereas in 3D it amounted to 6.9%.

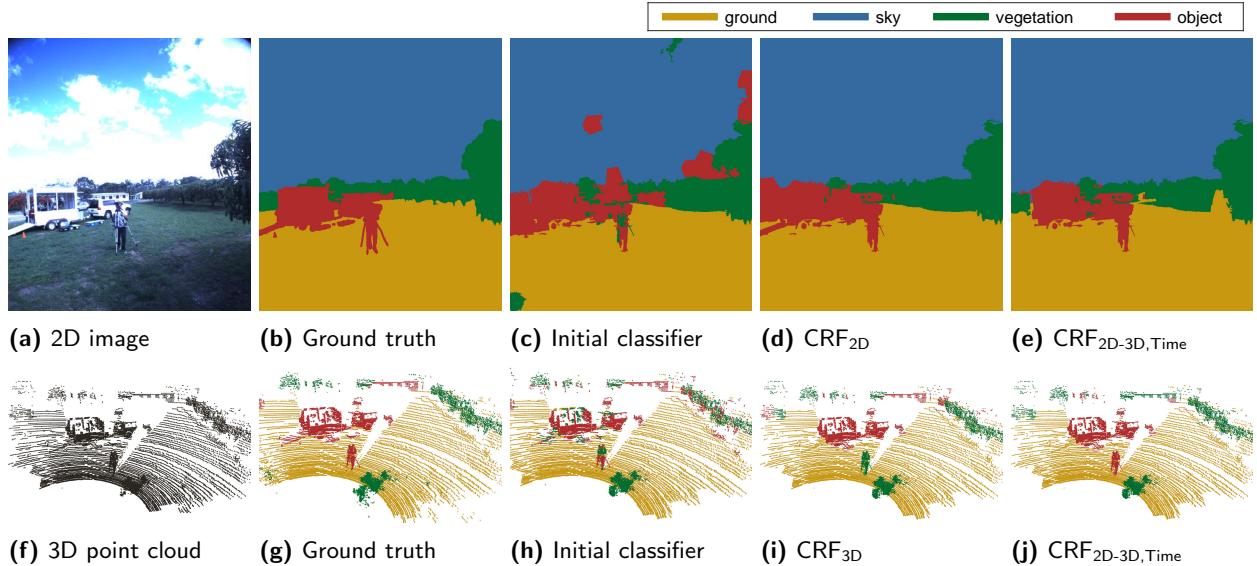
The most prominent increases belonged to the *object* class, where appearance or geometric clues from one modality significantly helped recognize the class in the other modality. Ultimately, adding recursive inference provided the best overall performance. In 2D, an increase in mean IoU of 0.7% was achieved, whereas in 3D recursive inference caused an increase of 0.2%. The most significant increase was for the *object* class in 2D with an increase in IoU of 2.1%.

Figure 9 illustrates an example of a corresponding image and point cloud classified with the initial classifiers and with the CRF. From (c), it is clear that the initial classification of the image was noisy and affected by saturation problems in the raw image. When introducing 2D edges in the CRF (d), most of these mistakes were corrected. Finally, when combined with information from 3D, the CRF was able to correct *vegetation* and *ground* pixels around the trailer (e). For 3D, some confusion between *vegetation* and *object* occurred in the initial 3D estimate (h), but was mostly solved by introducing 3D edges in the CRF (i). The person in the front of the scene was mistakenly classified as vegetation when using 3D edges, but this was corrected after fusing with information from 2D (j). In some cases, misclassifications in one domain also affected the other. In 2D, sensor fusion introduced a misclassification of the trailer ramp (e), which was seen as *ground* by the initial 3D classifier. Most likely, this happened because the ramp was flat and essentially served the purpose of connecting the ground and the trailer. In 3D, sensor fusion introduced a misclassification of the building in the background (j). This could be caused by rotational calibration errors between the two sensors, causing a constant pixel-wise misalignment. The influence on sensor fusion of this misalignment depends on distance, since the pixel areas of 3D segments decrease with distance under perspective projection. Therefore, the percentage alignment error due to calibration increases with distance.

For the same example section of the dataset presented in Figure 9, Figure 10 illustrates the accumulated classification results in 3D, of a trajectory along the end of a row. This section was chosen as a compact area with many examples of

Table 2. Classification results for 2D and 3D.

	ground	sky	IoU vegetation	object	mean	accuracy
2D, Initial	0.842	0.921	0.698	0.185	0.662	0.892
2D, CRF _{2D}	0.896	0.968	0.768	0.291	0.731	0.938
2D, CRF _{2D-3D}	0.914	0.968	0.779	0.343	0.751	0.945
2D, CRF _{2D-3D,Time}	0.916	0.970	0.780	0.364	0.758	0.947
3D, Initial	0.931	-	0.750	0.294	0.658	0.880
3D, CRF _{3D}	0.929	-	0.851	0.393	0.724	0.924
3D, CRF _{2D-3D}	0.929	-	0.884	0.565	0.793	0.941
3D, CRF _{2D-3D,Time}	0.932	-	0.891	0.562	0.795	0.944

**Figure 9.** Example results. The two rows show 2D and 3D results, respectively.

the different classes. The accumulated point cloud was generated by applying the $CRF_{2D-3D,Time}$ fusion method to each frame and then transforming all 3D points from the sensor frame into the world frame. To generate the figure, the most recent class prediction within any $0.5m$ radius is chosen to represent the region. That is, if a point \mathbf{p}_1 was given class label c_1 at time t_1 , then this inherited class label c_2 of point \mathbf{p}_2 at time t_2 if $|\mathbf{p}_2 - \mathbf{p}_1| \leq 0.5m$ and $t_2 > t_1$.

Effectively, this corresponds to always trusting the most recent prediction of the algorithm.

Figure 11 visualizes the learned CRF weights averaged over the 5 cross-validation folds. As explained in section 2.3.2, (a), (b), and (d) are symmetric, whereas (c) is asymmetric. For visualization purposes, we trained the CRF without bias weights, as these would introduce another matrix for each potential and thus make the interpretation

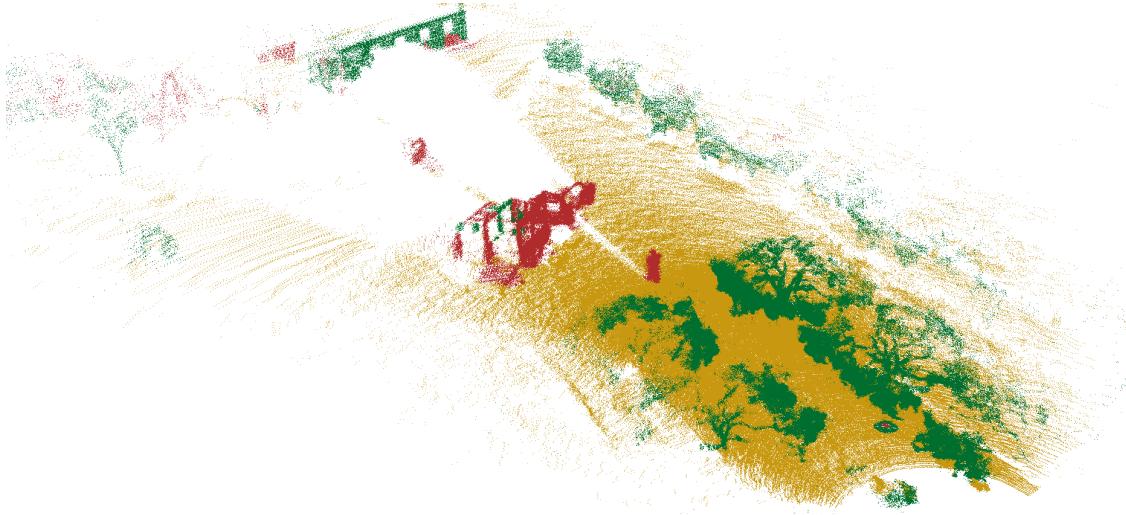


Figure 10. Example of accumulated classification results in 3D of a trajectory.

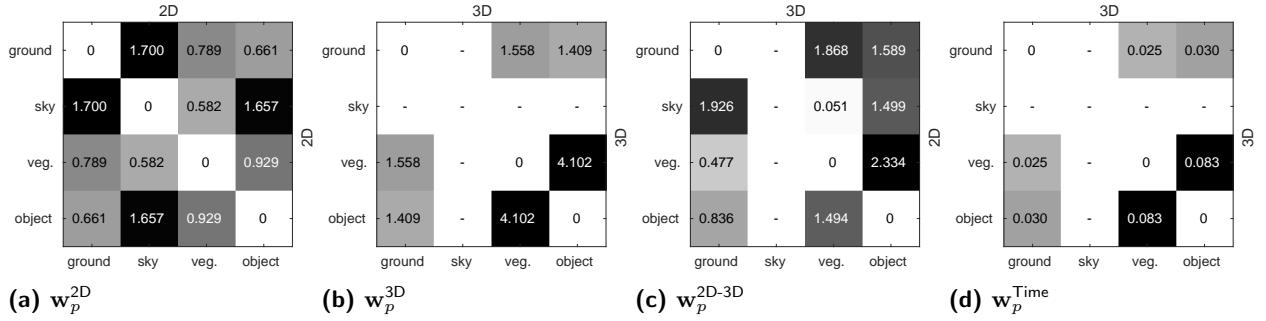


Figure 11. Learned CRF weight matrices averaged over cross-validation folds. High weights correspond to rare occurrences and vice versa.

of the weights more difficult. Figure 11a shows the weight matrix for neighboring 2D segments. The weights depend on the certainty of the initial classifier and how often adjacent superpixels with different labels appeared in the training set. *ground-object* and *vegetation-sky* appeared often and thus had low weights, whereas *ground-sky* and *object-sky* were rare and therefore were penalized with high weights. Intuitively, this makes sense, as vegetation often separates the ground from the sky in agricultural fields. Figure 9 illustrates how *object* superpixels in the middle of the sky in (c) were corrected by the CRF to *sky* in (d). This was directly caused by a high

value of w_p^{2D} (*ground, sky*) and multiple adjacent *sky* neighbors.

Figure 11b shows the weight matrix for neighboring 3D segments. Here, the highest weight was for *object-vegetation*. Structurally, these classes were difficult to distinguish with the initial classifier as seen in Figure 9 (g). However, when introducing spatial links in the CRF, most ambiguities were solved as seen in (h).

Figure 11c shows the weight matrix for the 2D-3D fusion. As mentioned in section 2.3.2, the matrix is asymmetric, as we allow different interactions between the 2D and 3D domain. The interpretation

of these weights is considerably more complex than \mathbf{w}_p^{2D} and \mathbf{w}_p^{3D} , since the weights incorporate calibration and synchronization errors between the lidar and the camera, and since overlapping 2D and 3D segments intuitively cannot have different class labels. However, a notable outlier was the weight for *sky-vegetation* which was remarkably low. The only apparent explanation for this is a calibration error between the two modalities. Physically, a 2D segment cannot be *sky* if an overlapping 3D segment has observed it. Therefore, label inconsistencies near border regions of *vegetation* and *sky* will cause the CRF weight to decrease.

Figure 11d shows the weight matrix for recursive inference. The weights were all small and thus matched the small increase in classification performance when introducing recursive inference. As the weights describe the cost of assigning different labels at the approximate same 3D location, we see the same trend as for neighboring 3D segments in Figure 11b.

4.2 Binary and Multiclass Classification

Due to the physics of the camera and the lidar, the two modalities perceive significantly different characteristics of the environment. The lidar is ideal for distinguishing elements that are geometrically unique, whereas the camera is ideal for distinguishing visual uniqueness. The choice of classes therefore highly affects the resulting improvement with the CRF fusion stage.

In this section, we compare binary and multiclass classification scenarios. The first scenario maps all annotated labels except *ground* to a common *non-ground* class, such that $x_i = \{\text{ground}, \text{non-ground}\}$. The second scenario is the same 4-class scenario as presented above. For convenience, the results from Table 9 are replicated in this section.

Table 3 presents the results for the 2D and 3D domains separately. For 2-class classification, the CRF fusion only improved 2D performance, whereas 3D performance actually declined. This is because the geometric classifier (lidar) is good at detecting ground points, and thus can single-handedly distinguish *ground* and *non-ground*. For 4-class classification, however, the CRF fusion

introduced improvements in both 2D and 3D. This was caused by the geometric classifier being less discriminative for *vegetation* and *object*, since both classes were represented by obstacles protruding from the ground. Therefore, color and texture cues from the visual classifier could help separate the classes.

4.3 2D Classifiers

As described in section 2.1, a traditional vision pipeline with hand-crafted features was compared to a deep learning approach with self-learned features. Figure 12 compares the two approaches before and after applying the CRF fusion. (a) and (b) show 2D and 3D results for each class, respectively. Filled bars denote initial classification results, whereas hatched bars show classification results after sensor fusion (CRF_{2D-3D}). In Figure 12a, we see that the initial classification results for deep learning were significantly better than for traditional vision with a mean IoU of 73.5% vs. 66.2%. The most significant difference was for the *object* class. Here, deep learning had a clear advantage, since the CNN was pre-trained on an extensive dataset with a wide collection of object categories. When fused with 3D data, however, both methods reached a mean IoU of 75.1%. The improvement in classification performance was thus much higher for the traditional vision pipeline than for deep learning. And interestingly, if we look at 3D classification in Figure 12b, the best mean IoU was obtained when fusing with the traditional vision pipeline. Here, a mean IoU of 79.3% was achieved, compared to 76.6% for deep learning. A possible explanation for this is that deep learning is extremely good at recognition, since it uses a hierarchical feature representation and thus incorporates contextual information around each pixel. However, in doing this, a large receptive field (spatial neighborhood) is utilized, which along with multiple max-pooling layers reduces the classification accuracy near object boundaries (Chen et al. 2014). And since the fusion stage of the CRF assumes exact localization in both 2D and 3D, we actually experience a smaller improvement when fusing with deep learning.

Table 3. Classification results for binary and multiclass scenarios.

	2-class scenario		4-class scenario	
	mean IoU	accuracy	mean IoU	accuracy
2D, initial	0.916	0.958	0.662	0.892
2D, CRF _{2D}	0.933	0.967	0.731	0.938
2D, CRF _{2D-3D}	0.939	0.970	0.751	0.945
2D, CRF _{2D-3D,Time}	0.940	0.970	0.758	0.947
3D, initial	0.926	0.962	0.658	0.880
3D, CRF _{3D}	0.930	0.964	0.724	0.924
3D, CRF _{2D-3D}	0.911	0.954	0.793	0.941
3D, CRF _{2D-3D,Time}	0.917	0.957	0.795	0.944

Figure 12 (c) and (d) show 2D and 3D results for each dataset, respectively. Here, we see the same tendency that deep learning was superior in 2D in its initial classification for all datasets except *lychees*. However, when fused with 3D data, the two methods basically performed equally well. Traditional vision was better for *mangoes* and *lychees*, deep learning was better for *apples*, and they were almost equal for *almonds* and *dairy*.

To summarize, when evaluating individual performance, deep learning was better than traditional vision. However, when applying a CRF and fusing with lidar, the two methods gave similar results. The CRF was thus able to compensate for the shortcomings in the traditional vision approach.

4.4 Transferability

In section 4.1, we evaluated the combined classification results over all datasets. In this section, we revisit and break apart these results into separate datasets. In this way, we can evaluate the transferability of features and classifiers across datasets and across classes. This will allow us to answer a question like: how well do the features and classifiers trained on the combined imagery from *mangoes*, *lychees*, *apples* and *almonds* generalize to recognize a new scenario, such as *vegetation* in the *dairy* dataset? Figure 13 compares the classification performances in 2D and 3D separately across object classes and datasets. Filled

bars denote initial classification results, whereas hatched bars show classification results after sensor fusion (CRF_{2D-3D}).

Figure 13a shows that for 2D, features and classifiers transferred quite well for *ground* and *sky*, possibly due to a combination of limited variation in visual appearance and an extensive amount of training data. However, a larger variation was observed across datasets for *vegetation* and *object*. For the *vegetation* class, the *dairy* dataset had the lowest 2D classification performance. This might be because the mean distance to the tree line was much higher for the dairy paddock than for the orchards, as seen in Figure 8. The visual appearance varies with distance, and especially features describing texture are affected by associated changes in scale and resolution. For the *object* class, a large variation in 2D performance was seen across all datasets. This is most likely due to the large variation in *object* appearances, as the class covered humans, vehicles, buildings, and animals. Also, as listed in Table 1, not all datasets included examples of buildings and animals.

Figure 13b shows that for 3D, the features and classifiers transferred well for *ground*, but experienced the same tendencies in variation for *vegetation* and *object* as seen in 2D. For the *vegetation* class, the *dairy* dataset had an IoU close to 0%. This is likely due to the mean distance to the tree line which was outside the range of the lidar. Only a few 3D points

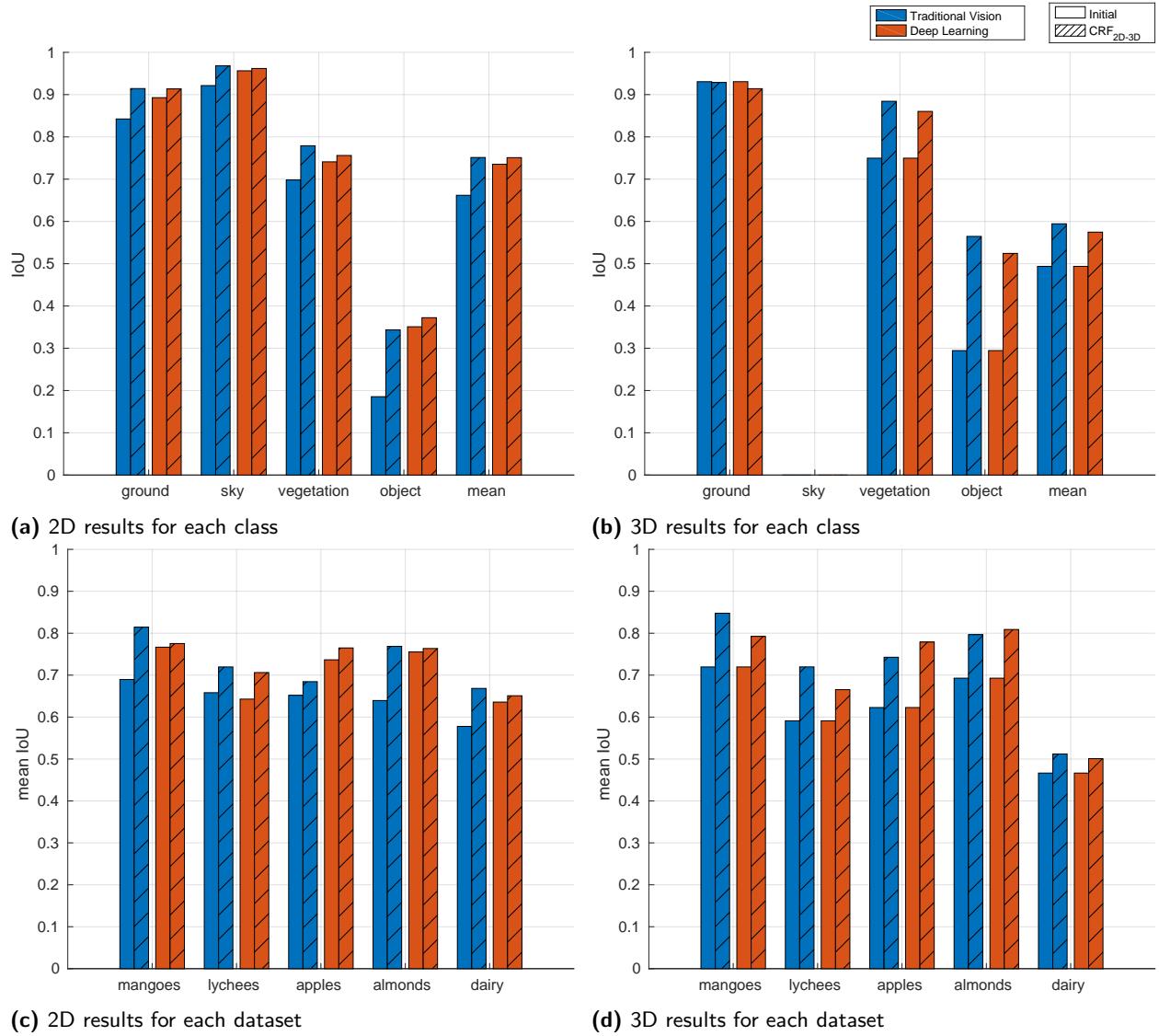


Figure 12. Evaluation of traditional vision vs. deep learning before and after sensor fusion.

within range were labeled *vegetation*, and since the classification performance decreases with distance, most of these were misclassified. For the *object* class, a large variation in 3D performance was seen across all datasets, similar to 2D. However, the initial 3D classifier performed better than 2D, suggesting

slightly better transferability for 3D features and classifiers.

Evaluating the transferability of CRF weights, we compared the increase in classification performance across the different datasets (the difference between filled and hatched bars of the same colour in Figure 13). Generally, the CRF weights transferred

well across all datasets in both 2D and 3D. However, in 2D, a decrease in IoU for the *object* class was observed for *apples*, while in 3D, the *ground* class experienced both increases and decreases. For the latter, difference in terrain roughness could possibly explain the mixed results.

To summarize, with minor exceptions, features and classifiers transferred well across the *ground*, *sky*, and *vegetation* classes for all datasets in both 2D and 3D. For these classes, the CRF framework is able to deliver performance increases even when training data is supplied from different environments, which is reasonable given that the appearance of these classes to some degree is independent of the specific site. For the *object* class, however, features and classifiers transferred poorly in both 2D and 3D, resulting in considerable performance variations across datasets. This was likely caused by limited training data covering the large variation in geometry and appearance within the *object* class, as cows were only present in the *dairy* dataset, tractors in *mangoes*, iron bars in *lychees*, etc.

5 Conclusion

This paper has presented a method for multi-modal obstacle detection by fusing camera and lidar sensing with a conditional random field. Initial 2D (camera) and 3D (lidar) classifiers have been combined probabilistically, exploiting both spatial, temporal, and multi-modal links between corresponding 2D and 3D regions. The method has been evaluated on data gathered in various agricultural environments with a moving ground vehicle.

Results have shown that for a two-class classification problem (ground and non-ground), only the camera leveraged from information provided by the lidar. In this case, the geometric classifier (lidar) could single-handedly distinguish ground and non-ground structures. However, as more classes were introduced (*ground*, *sky*, *vegetation*, and *object*), both modalities complemented each other and improved the mean classification score.

The introduction of spatial, multi-modal, and temporal links in the CRF fusion algorithm showed gradual improvements in the mean intersection

over union classification score. Adding spatial links between neighboring segments in 2D and 3D separately, first improved the initial and individual classification results with 6.9% in 2D and 6.6% in 3D. Then, adding multi-modal links between 2D and 3D caused a further improvement of 2.0% in 2D and 6.9% in 3D. And finally, adding temporal links between successive frames caused an increase of 0.7% in 2D and 0.2% in 3D. The performance gains may not justify the added complexity of the method for all applications of obstacle avoidance in agricultural contexts, however, the method proves that it is possible to reduce uncertainty when probabilistically fusing lidar and camera as opposed to applying each sensor individually.

A traditional computer vision pipeline was compared to a deep learning approach for the 2D classifier. It was shown that deep learning outperformed traditional vision when evaluating their individual performances. However, when applying a CRF and fusing with lidar, the two methods gave similar results.

Finally, transferability was evaluated across agricultural domains (*mangoes*, *lychees*, *apples*, *almonds*, and *dairy*) and classes (*ground*, *sky*, *vegetation*, and *object*). Results showed that features and classifiers transferred well across domains for the *ground* and *sky* classes, whereas *vegetation* and *object* were less transferable due to a larger inter-domain variation in appearance and geometry.

In situations where scene parsing can benefit from input from different sensor modalities, the paper provides a flexible, probabilistically consistent framework for fusing multi-modal spatio-temporal data. The approach is flexible and may be extended to include additional heterogeneous data sources in future work, including radar, stereo or thermal vision, all of which are directly applicable within the framework.

Funding

This work is sponsored by the Innovation Fund Denmark as part of the project SAFE - Safer Autonomous Farming Equipment (project no. 16-2014-0) and supported by the Australian Centre for Field

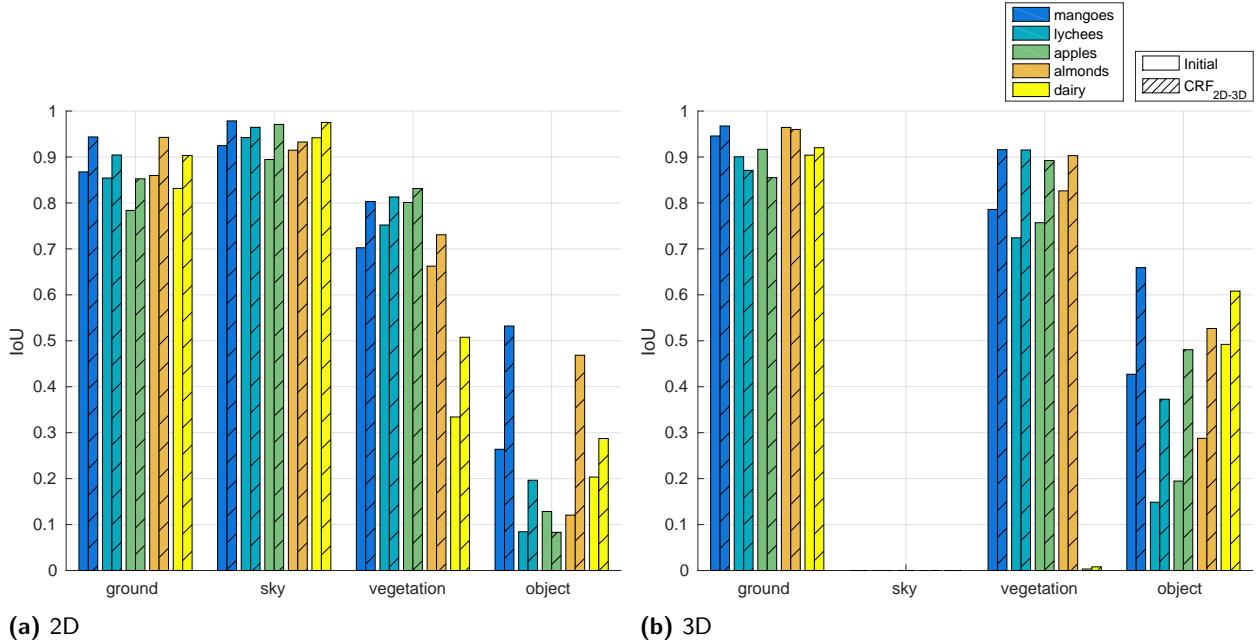


Figure 13. Classification results across object classes and datasets before and after sensor fusion.

Robotics at The University of Sydney and Horticulture Innovation Australia Limited through project AH11009 Autonomous Perception Systems for Horticulture Tree Crops. Further information and videos available at: <http://sydney.edu.au/acfr/agriculture>.

ATC (2016). Autonomous Tractor Corporation. <http://www.autonomoustractor.com/>. Accessed: 2016-09-28.

References

- Abidine, A. Z., Heidman, B. C., Upadhyaya, S. K., and Hills, D. J. (2004). Autoguidance system operated at high speed causes almost no tomato damage. *California Agriculture*, 58(1):44–47.
- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.
- ASI (2016). Autonomous Solutions. <https://www.asirobots.com/farming/>. Accessed: 2016-09-28.
- Boykov, Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 105–112. IEEE Comput. Soc.
- Brunner, C., Peynot, T., Vidal-Calleja, T., and Underwood, J. (2013). Selective Combination of Visual and Thermal Imaging for Resilient Localization in Adverse Conditions: Day and Night, Smoke and Fire. *Journal of Field Robotics*, 30(4):641–666.
- Cadena, C. and Košeká, J. (2016). Recursive Inference for Prediction of Objects in Urban Environments. In *International Symposium on Robotics Research*, pages 539–555.

-
- Case IH (2016). Case IH Autonomous Concept Vehicle. <http://www.caseih.com/apac/en-in/news/pages/2016-case-ih-premieres-concept-vehicle-at-farm-progress-show.aspx>. Accessed: 2016-09-28.
- Chang, C.-c. and Lin, C.-j. (2011). LIBSVM. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27.
- Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *International Conference on Learning Representations*, pages 1–14.
- Dima, C., Vandapel, N., and Hebert, M. (2004). Classifier fusion for outdoor obstacle detection. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 665–671 Vol.1. IEEE.
- Douillard, B., Fox, D., and Ramos, F. (2010). A Spatio-Temporal Probabilistic Model for Multi-Sensor Multi-Class Object Recognition. In *Springer Tracts in Advanced Robotics*, volume 66, pages 123–134.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Haralick, R. M., Shanmugam, K., and Dinstein, I. (1973). Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621.
- Häselich, M., Arends, M., Wojke, N., Neuhaus, F., and Paulus, D. (2013). Probabilistic terrain classification in unstructured environments. *Robotics and Autonomous Systems*, 61(10):1051–1059.
- He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 7, pages 171–180.
- Hebert, M. and V, N. (2003). Terrain Classification Techniques From Ladar Data For Autonomous Navigation. In *In Collaborative Technology Alliances Conference*.
- Hermans, A., Floros, G., and Leibe, B. (2014). Dense 3D semantic mapping of indoor scenes from RGB-D images. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2631–2638. IEEE.
- Kragh, M., Christiansen, P., Korthals, T., Jungeblut, T., Karstoft, H., and Nyholm Jørgensen, R. (2016). Multi-Modal Obstacle Detection and Evaluation of Occupancy Grid Mapping in Agriculture. In *Proceedings of the International Conference on Agricultural Engineering, Aarhus, Denmark*, pages 1–8.
- Kragh, M., Jørgensen, R. N., and Pedersen, H. (2015). Object Detection and Terrain Classification in Agricultural Fields Using 3D Lidar Data. In *Computer Vision Systems : 10th International Conference, ICVS 2015, Proceedings*, volume 9163, pages 188–197.
- Krähenbühl, P. and Koltun, V. (2012). Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. *Advances in Neural Information Processing Systems 24*, (4):109–117.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.
- Laible, S., Khan, Y. N., and Zell, A. (2013). Terrain classification with conditional random fields on fused 3D LIDAR and camera data. In *2013 European Conference on Mobile Robots*, pages 172–177. IEEE.

- Lalonde, J.-F., Vandapel, N., Huber, D. F., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839–861.
- Levinson, J. and Thrun, S. (2013). Automatic Online Calibration of Cameras and Lasers. In *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation.
- Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440. IEEE.
- Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.
- Milella, A., Reina, G., and Underwood, J. (2015). A Self-learning Framework for Statistical Ground Classification using Radar and Monocular Vision. *Journal of Field Robotics*, 32(1):20–41.
- Milella, A., Reina, G., Underwood, J., and Douillard, B. (2014). Visual ground segmentation by radar supervision. *Robotics and Autonomous Systems*, 62(5):696–706.
- Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., and Yuille, A. (2014). The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898. IEEE.
- Munoz, D., Bagnell, J. A., and Hebert, M. (2012). Co-inference for multi-modal scene analysis. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV’12, pages 668–681, Berlin, Heidelberg. Springer-Verlag.
- Namin, S. T., Najafi, M., and Petersson, L. (2014). Multi-view terrain classification using panoramic imagery and LIDAR. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, number Iros, pages 4936–4943. IEEE.
- Namin, S. T., Najafi, M., Salzmann, M., and Petersson, L. (2015). A Multi-modal Graphical Model for Scene Analysis. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 1006–1013. IEEE.
- Papon, J., Abramov, A., Schoeler, M., and Worgotter, F. (2013). Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034. IEEE.
- Pele, O. and Werman, M. (2010). The Quadratic-Chi Histogram Distance Family. In *Lecture Notes in Computer Science*, volume 6312 LNCS, pages 749–762.
- Peynot, T., Underwood, J., and Kassir, A. (2010). Sensor Data Consistency Monitoring for the Prevention of Perceptual Failures in Outdoor Robotics. In *Seventh IARP Workshop on Technical Challenges for Dependable Robots in Human Environments Proceedings*, pages 145–152, Toulouse, France.
- Posner, I., Cummins, M., and Newman, P. (2009). A generative framework for fast urban labeling using spatial and temporal context. *Autonomous Robots*, 26(2-3):153–170.
- Quadros, A., Underwood, J., and Douillard, B. (2012). An occlusion-aware feature for range images. In *2012 IEEE International Conference on Robotics and Automation*, pages 4428–4435. IEEE.
- Reina, G., Milella, A., Rouveure, R., Nielsen, M., Worst, R., and Blas, M. R. (2016a). Ambient awareness for agricultural robotic vehicles. *Biosystems Engineering*, 146:114–132.
- Reina, G., Milella, A., and Worst, R. (2016b). LIDAR and stereo combination for traversability assessment of off-road robotic vehicles. *Robotica*, 34(12):2823–2841.

-
- Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1850–1857. IEEE.
- Schmidt, M. (2007). UGM: A Matlab toolbox for probabilistic undirected graphical models. <http://www.cs.ubc.ca/~schmidtm/Software/UGM.html>.
- Underwood, J. P., Hill, A., Peynot, T., and Scheding, S. J. (2010). Error modeling and calibration of exteroceptive sensors for accurate mapping applications. *Journal of Field Robotics*, 27(1):2–20.
- Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms. <http://www.vlfeat.org/>.
- Wellington , C., Courville, A., and Stentz , A. T. (2005). Interacting Markov Random Fields for Simultaneous Terrain Modeling and Obstacle Detection. In *Proceedings of Robotics: Science and Systems*.
- Winn, J. and Shotton, J. (2006). The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, volume 1, pages 37–44. IEEE.
- Wu, T.-F., Lin, C.-J., and Weng, R. C. (2004). Probability Estimates for Multi-class Classification by Pairwise Coupling. *Journal of Machine Learning*, 5:975–1005.
- Xiao, L., Dai, B., Liu, D., Hu, T., and Wu, T. (2015). CRF based road detection with multi-sensor fusion. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, number Iv, pages 192–198. IEEE.
- Zhang, R., Candra, S. A., Vetter, K., and Zakhor, A. (2015). Sensor fusion for semantic segmentation of urban scenes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1850–1857. IEEE.

Appendix A: Parameter List

A list of all parameter settings for 2D and 3D classifiers and the CRF fusion framework is available in Table 4.

Table 4. Algorithm parameters used for initial classifiers (2D and 3D) and CRF fusion.

2D classifiers		3D classifier	CRF fusion	
Image		Point cloud		Pairwise potentials
width	616	beams	64	σ_{2D} 1
height	808	θ_H	0.08°	σ_{3D} 1
SLIC		Feature extraction		σ_{Time} 1
region size	40	M	60	
regularization factor	3000	Supervoxels		
SIFT		seed resolution	0.1	
bin size	3	voxel resolution	0.2	
magnification factor	4.8	λ	1	
BoW		iterations	10	
vocabulary size	50			
fraction of strongest features	0.5			
SVM		SVM		
examples	100000	examples	40000	
kernel	RBF	kernel	RBF	
γ	1/57	γ	1/9	
C	1	C	1	
CNN				
learning rate	10^{-12}			
momentum	0.99			
epochs	10			
data augmentation	horizontal flip			