Scene Recognition for Mobile Robots in Plant-rich Environments Considering Traversable Plants

(植物繁茂環境における移動ロボットのための
通過可能植物を考慮した環境認識)

January 2023

Doctor of Philosophy (Engineering)

Shigemichi Matsuzaki
松崎 成道

Toyohashi University of Technology

Abstract

Autonomous mobile robots can help human by automating laborious tasks that are traditionally done manually. The techniques of mobile robots have been developed in a last few decades, and have come to maturity in some environments in the recent years. For example, service robots are commercialized and used in public places such as restaurants and airports. Such technologies are expected to be applied to a wider variety of fields. Agriculture is one of the fields where automation is highly demanded due to labor shortage and aging. Forests are also a type of environment where the automation of the tasks is expected but yet to be fully realized. Applying autonomous mobile robots in such fields will bring about positive impact in many ways such as mitigating physically demanding tasks, and improving productivity.

In agricultural fields and forests, there may be plants growing out to the paths. However, majority of conventional mobile robots rely on scene recognition methods that consider only the geometric information of the environment. Those methods, therefore, cannot recognize paths covered by flexible plants as traversable. This problem hinders application of mobile robots in unstructured plant-rich environments. To realize successful navigation in such environments, the robots need to have an ability to distinguish traversable plants from other obstacles.

In this thesis, we describe a framework of scene recognition for robot navigation in plant-rich environments that explicitly considers traversable plants. The proposed method employs a image-based deep neural network (DNN) for scene recognition. To mitigate the need of laborious manual data annotation which is used in a usual practice of training DNNs, we propose some methods to train the network without manual annotation along with the scene recognition framework. This thesis consists of three key proposals.

First, we propose a method to train a DNN for semantic segmentation that does not require manual annotation on target images. We work around the need for manual annotation by utilizing multiple publicly available datasets as source datasets from which to transfer knowledge about appearances of objects. Specifically, we exploit segmentation models pre-trained on each source dataset to generate pseudo-labels for the target images based on agreement of all the pre-trained models on each pixel. The proposed method allows for effectively transferring the knowledge from multiple sources rather than relying on a single dataset and realizes precise training of semantic segmentation model.

Second, we propose a method to estimate the traversability of plant parts covering a path and navigating through them. We employ an image-based DNN model with two decoder branches to estimate on each pixel the general object classes, and *traversability* indicating how likely the object can be hit by the robot while moving. We train the traversability estimation branch utilizing the robot's traversal experience during the data acquisition phase, and thus the training procedure is free from manual annotation. A real-world navigation experiment was conducted using the proposed scene recognition method.

Third, we propose a method of online refinement of the scene recognition model to deal with misclassification that occurs during robot operation. In our system, misclassification may lead the robot to getting stuck due to the traversable plants recognized as obstacles. Yet, misclassification is inevitable in any estimation methods. To deal with the problem, we propose a framework that allows for refining a semantic segmentation model on the fly during the robot's operation utilizing observation of a human's interaction with the traversable plant parts.

The proposed framework enables robot navigation in plant-rich environments by recognizing traversability plants. It also allows for easy deployment of the mobile robots in such environments by providing manual annotation-free training methods, and practical online refinement of the scene recognition model to easily deal with misclassification problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Techniques for mobile robotics have been rapidly developed in last few decades. The theory of probabilistic robotics was established as a fundamental component to build autonomous robots that are robust to various kinds of uncertainty in systems, environments, and observations [1]. Recently, some mobile robot systems have been commercialized. An example is service robots operating in public places such as airports and stores [2, 3]. Autonomous vehicles are also tested and even launched in public areas [4, 5]. Those applications mainly target structured environments such as the inside of buildings and urban areas, which are dominated by drivable ground and rigid obstacles.

Compared to the structured scenes, there are far fewer examples of real world applications of fully autonomous mobile robots in unstructured scenes. Despite the situation, there is a high demand for mobile robots in such environments. Agricultural mobile robots are one of the examples of highly demanded robotic applications. Due to population growth throughout the world and population aging of agricultural workers, increasing productivity is an urgent challenge [6]. Moreover, the tasks in agriculture are often repetitive and labor-intensive. Automating such tasks by means of robotic techniques will make better and safer working environments. The robotic systems including mobile robots are thus deemed to be an enabler of transformation of food production [6]. Japan is promoting "Smart agriculture" [7], where robotics and ICT (information and communication technologies) are employed to various agricultural tasks such as planting, monitoring, and harvesting crops etc.

Forests are another example of plant-rich environments. The mobile robot applications in forest environments are even fewer than the agricultural robots, as the forest environments are more complex than an agricultural environment [8] due to grown vegetation on the ground, highly unstructured and steep slope terrains, etc. The tasks of the forest robots include forest preservation and monitoring, wildfire fighting, forest inventory, forest sensing etc. [8, 9, 10, 11]. Forest sensing is a task to measure the attributes of forest trees using sensors such as LiDARs for forest management [9]. Mobile Laser Scanning (MLS) is attracting attention for faster data acquisition than traditional Terrestrial Laser Scanning (TLS), and better precision of scanning than Aerial Laser Scanning (ALS). In current MLS, the robot is controlled manually. Automating the robot navigation will be beneficial for the application.

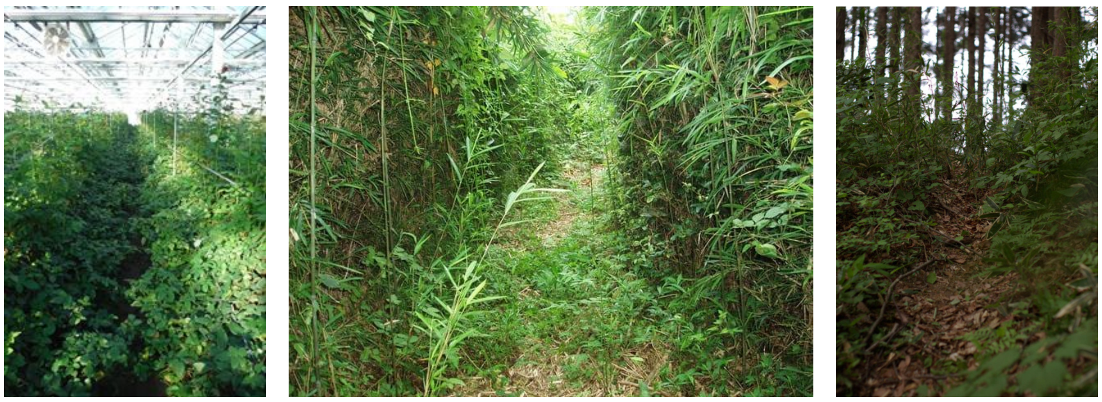


Figure 1.1: Examples of plant-rich environments. In such environments, the traversable paths may be covered by plant parts such as foliage and branches. They may be detected as obstacle by conventional mobile robots with range sensors though they are flexible, and a robot can move through them.

1.2 Limitations of the existing work

Despite the demands on mobile robots in plant-rich unstructured environments, there is still a gap between current research work and real-world applications. Navigation in such environments is hindered by multiple factors. One of them is presence of plant parts such as foliage and branches growing out to the paths. Fig. 1.1 shows examples of the plant-rich environments. Human can recognize a traversable region covered by plants and walk through it by pushing the plants aside under such situations. Most of the conventional mobile robots, however, are not able to deal with such plant parts because they rely on range sensors to detect the obstacles, and thus consider only presence of objects. It is crucial to overcome this limitation to apply mobile robot systems to unstructured environments.

Traditionally in agricultural fields, traversable path detection relies on crop row detection [12, 13, 14, 15, 16]. Those methods assume that paths are straight and open. They thus do not consider the traversable plant parts appearing in the middle of the paths, although Mandow et al. [17] mentioned in their work that such plant may disturb range sensing, and specialized detection is required.

Some researchers have developed path planning and/or autonomous navigation methods in forest environments [18, 19]. However, they typically use only tree trunks as obstacles, and ignore traversable plants by filtering scattered points induced by vegetation. Such methods assume clear distinction between tree trunks and vegetation, which does not hold in not well-maintained environments.

Recent approaches to robot navigation in unstructured environment with vegetation prefer to train a deep neural network (DNN) using self-labeled training data to predict future events (successful traversal or collision) given an image and a sequence of control signals [20], or directly estimate traversability [21, 22] to output the optimal control signals. They showed an ability to successfully navigate in some unstructured scenes with obstacles hard to recognize with only geometric information such as tall grass. However, they require both successful and failure examples of traversal experiences to train the network. It limits the applicability of the methods to environments with objects that the robot must not collide with.

1.3 Research goal

The primary purpose of our research is to develop a scene recognition method for navigation of mobile robots considering traversable plants covering their path. For this, we claim that estimation of semantic information is necessary. Specifically, we consider two types of information: 1. General object classes such as *plants*, *artificial object*, and *ground*, and 2. *Traversability* indicating how likely an object can be hit by the robot while moving. Existing traversability estimation methods only consider either one of them. In reality, they both serve as important information to estimate object traversability. By considering object classes, we can utilize our prior knowledge that plants are more likely to be traversable than rigid artificial objects. However, plants also have part that should not be hit by robots while traversing such as stems, and inflexible branches. Here, traversability can serve as information that is more directly relevant to the physical property of the objects allowing robots to traverse through. Traversability estimation can, however, have some misclassifications which may lead to fatal problems such as crushing into obstacles. In such case, object class information can refine the estimation by the prior knowledge about the objects. Those two types of information are thus complementary to each other.

In addition, we also claim that the training method of the scene recognition should be as easy as possible. Data-driven estimation methods such as deep neural networks (DNNs) have two problems. First, they usually require a large amount of data, which are hard to acquire manually. Second, the performance of a model that is trained on a specific environment usually degrades on different environments due to gaps of data distributions between the environments, which stems from difference in appearance in image data, for example. Due to the second problem, a model should be trained on a dedicated dataset of the target environment for practical use. However, the first problem hinders specialized model training on the target dataset. Considering deployment of the robotic system to end-users, the system should not involve such a cumbersome training process. To deal with the problem, we also develop a method for scene recognition model training that does not require manual labeling. It allows for easily adapting the model to new environments.

To sum up, our goal is to develop a scene recognition method to recognize traversable plants for mobile robot navigation in plant-rich environments, and its training method that does not require manual annotation for easy deployment.

1.4 Proposed framework

To achieve our goal, we propose a framework of scene recognition considering semantic information of the objects, as well as geometric information. Fig. 1.2 outlines the proposed framework. There are three phases in the system, namely *data collection*, *model training*, and *robot operation* phases.

In the data collection phase, a robot with an RGB-D sensor is manually controlled in the target environment while taking a sequence of RGB-D data and optionally the robot's wheel odometry, so that 3D map can be built in the following data labeling on traversable regions in images. During the data collection, the human operator make the robot traverse through traversable plant parts. The information of the traversals are

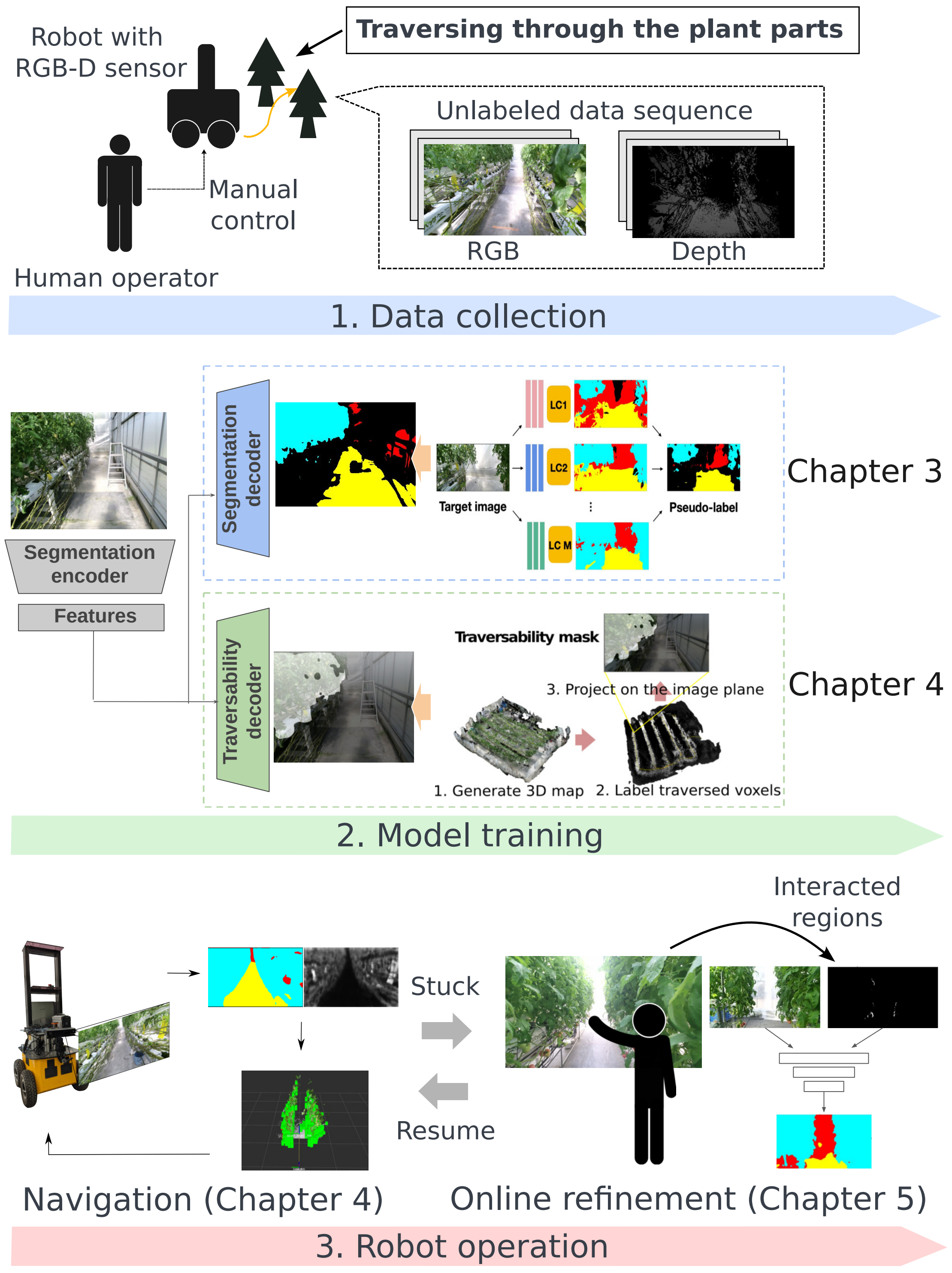


Figure 1.2: Overview of the proposed framework. It consists of three phases. 1. *Data collection phase*: a robot is controlled by a human operator and moves through the target environment while collecting RGB-D data. 2. *Model training phase*: a DNN model is trained using the data taken in the data collection phase. To mitigate the necessity of manual annotation, we incorporate publicly available datasets and the robot's experience of traversals during the data collection phase. 3. *Robot operation phase*: The robot autonomously moves using the trained scene recognition model. In case of getting stuck due to misclassification, the model is refined online by observing interaction of a human operator with the environment.

then utilized to generate label images indicating regions that the robot has traversed, coined *traversability masks.*

In the model training phase, a DNN-based scene recognition model is trained using the data taken in the data collection phase. Specifically, we propose a DNN model with two decoder branches sharing an encoder, one for semantic segmentation with general object classes coined *Semantic Segmentation Module (SSM)*, and another for class-agnostic traversability estimation coined *Traversability Estimation Module (TEM).* We train the whole model without any manual annotation on the target images. The SSM is trained utilizing multiple publicly available datasets as source datasets. The TEM is trained utilizing the traversability masks.

Finally, in the robot operation, the trained model is used for scene recognition. The recognized object classes and traversability are projected to the 3D space by the depth information, and 3D voxel map with the semantic information is built. The robot navigate through the traversable plants by treating as free spaces the voxels with *plant* class and traversability greater than a certain threshold.

During the robot operation, the robot may get stuck due to misclassification of traversable objects. To refine the misclassification, we introduce an online model refinement method so that the robot will correctly classify previously misclassified regions and successfully navigate through them. Specifically, we propose a data acquisition method by observing human interacting with regions that the robot should recognize as plants, and few-shot learning-based model refinement using the data.

1.5 Contributions of the thesis

The main contributions of the thesis are as follows:

1. **A novel scene recognition framework for navigation in plant-rich environments**

   We propose a framework that employs a deep neural network (DNN) for image-based semantic segmentation on general object classes, and pixel-wise estimation of traversability. Navigation through the traversable plants is realized by treating as free spaces the regions that are classified as *plant* and *traversable* by the model.

2. **Manual annotation-free training of the scene recognition model**

   We propose methods to train the DNN-based scene recognition model without any manual annotation on the target dataset.

   a) For the training of semantic segmentation, we utilize publicly available image datasets. We propose a method to effectively transfer knowledge from those source datasets to the target dataset without manually annotated labels.

   b) The traversability estimation is trained using the robot's experience of traversals. While recent existing methods [20, 21, 22, 23] require negative experience of navigation, which can be dangerous to acquire, we train the traversability estimation model only with positive traversability labels using Positive and Unlabeled (PU) learning framework [24].

c) We propose a method to refine the model on the fly during the robot's operation utilizing observation of a human's interaction with the environment to deal with misclassification.

1.6 Thesis outline

The remainder of this thesis is as follows. In Chapter 2, we review the literature of the related work. Chapter 3 describes a method to train a semantic segmentation model utilizing publicly available datasets with full pixel-wise labels which are not relevant to the target environments. Chapter 4 describes a method of scene recognition considering both general object classes and class agnostic traversability, and its manual annotation-free training. Chapter 5 describes a method of online refinement of the scene recognition model utilizing label data collected by observing a human's interaction with the environment. Finally, we conclude the thesis and discuss a future direction of the research in Chapter 6.

Chapter 2

Literature Review

2.1 Scene understanding for mobile robots

2.1.1 Obstacle detection and terrain analysis

In structured environments, the structure around the robot is measured by sensors such as sonar sensors, laser range finders, stereo cameras and so on [25, 26, 27]. Those methods assume only rigid obstacles in the environments, which is a reasonable assumption in structural environments such as indoor scenes.

In unstructured environments, however, there exist much more variety of obstacles including tall vegetation, slippery terrains such as sand and snow, negative obstacles like steps and ditches, and so on. It is difficult for the geometry-based obstacle detection to deal with those types of obstacles and terrains. To estimate the complex traversability information of the scenes, various types of sensors are used such as *exteroceptive* sensors to measure the state of the external world like a camera and a LiDAR, and proprioceptive *exteroceptive* sensors to measure the state of the robot itself like a vibration sensor and an inertial sensor [28].

Geometry-based terrain analysis

In geometry-based traversability estimation methods, a robot estimates obstacles, terrain elevation, slopes, and roughness.

Obstacles are classified into *positive* and *negative* ones. The former means obstacles above the ground, and the latter indicates terrestrial spots below the ground that hinder robots' navigation such as ditches and cliffs [28].

In earlier work of obstacle detection simply set criteria of the height and size to distinguish obstacles from the ground plane [29]. Nordin et al. [30] proposed a method to build traversability map using 2D LiDARs and conducted robot navigation in a forested environment with relatively flat terrain.

Santamaria-Navarro et al. [31] employed a 3D LiDAR for terrain classification considering positive and negative obstacles, and roughness of the terrain. Suger et al. [32] also used a 3D LiDAR to extract structural features. For classification, they compared Naive Bayes and PU learning by Elkan and Noto [24]. Frey et al. [33] utilized a DNN to estimate traversability on a 3D voxel map with traing data automatically collected in a simulator.

Vision-based terrain analysis

As visual information has rich information about the object texture and semantics, cameras are often used in traversability estimation. Kim et al. [34] proposed a machine learning-based traversability classification on superpixels using hand-crafted features using color and texture information. Hadsell et al. [35] used a pre-trained Deep belief network (DBN) [36] for estimating traversability from an image. Khan et al. [37] proposed a terrain classification method based on multiple texture features and random forest (RF) classifier [38]. Lee et al. [39] applied Bayesian clustering-based classification on color and texture features per superpixel.

Recent progress of semantic segmentation techniques driven by DNNs has enabled fine-grained terrain classification on each image pixel. Chavez-Garcia et al. [40] employed an image-based CNN to estimate traversability in pixel-wise binary classification. Matunara et al. [41] proposed a semantic segmentation-based terrain classification for estimating preferable regions for driving. Suryamurthy [42] employed DNN on RGB images for terrain segmentation as well as roughness estimation. Guan et al. [43] proposed a terrain classification method based on semantic segmentation by DNN for controlling an excavator.

Proprioceptive terrain analysis

Brooks et al. [44] proposed a terrain analysis method using a vibration sensor. Sebastian et al. [45] proposed a unique method to classify the terrain type based on the state transition information coupled with a support vector machine (SVM). This method uses only global positioning and a wheel encoder to estimate odometry, and do not require any further dedicated external sensors. Haddeler et al. [46] proposed a method to build a traversability map by estimating *collapsibility* metric, which is the risk of terrain collapsing when the robot steps over it, with a tactile sensor.

Terrain analysis using sensors of multiple modalities

Using sensors with multiple modalities provides richer information about terrain characteristics. Rasmussen [47] proposed a method to traversable road regions using structural and color features and a neural network. Kelly et al. [48] employed a terrain classification using two separated terrain classifier trained with geometric features and multi-spectral images, respectively. Kim et al. [23] used appearance and geometric features acquired by a stereo camera for traversability classification. Happold et al. [49] trained a neural network on geometric features, and propagate the estimated traversability to further regions using a camera image. Sock et al. [50] proposed a method to build a probabilistic traversability map by fusing classification results from image-based and LiDAR-based traversability classifiers.

Miki et al. [51] recently proposed a method for control a legged robot adaptively to varying terrain features by integrating 3D LiDAR-based terrain analysis and proprioceptive observations such as body orientation, body velocity etc.

Self-supervised learning for terrain analysis

Self-supervised learning in this context means a type of methods where the system is able to collect labels for training an estimation model by itself, and does not involve human in data annotation.

Especially in the 2000s, many studies have adopted an approach called *near-to-far learning*, which uses a reliable sensor readings in a limited scope for automatic labeling, and propagate the learned information to a wider range. The approach is effective to broaden the range of perception beyond that of range sensors such as stereo cameras and LiDARs. In near-to-far learning, various combinations of sensors are used, such as from a LiDAR or a stereo camera to RGB camera [49, 52, 53], from RGB camera (near range) to RGB camera (far range), proprioceptive sensors to exteroceptive sensors, and hybrid use of multiple sensors for labeling [54].

Lieb et al. [55] proposed a method of segmenting the traversable road region in a self-supervised manner, under an assumption that the terrain in front of the vehicle is traversable. Wellhausen et al. [56] proposed to automatically label images utilizing the footprints of legged robots. To automatically determine the class of the terrain, they use signals from a force-torque sensor. Some studies such as Onozuka et al. [57] and Matsuzaki et al. [58] train a recognition model for traversable area segmentation using a limited amount of self-labeled data. Schmid et al. [59] proposed a similar method as [57] which uses projections of the vehicle trajectory on images as labels of traversable regions.

2.1.2 Traversability estimation considering vegetation

In unstructured scenes, deformable vegetation can obstruct robot navigation [28]. Such vegetation can be traversed by the robots, but is hard to distinguish by geometric information. Foroutan et al. [60] investigated the influence of vegetation density on obstacle detection in the context of off-road vehicle navigation.

Some studies attempted to detect vegetation using a LiDAR sensor [61, 62]. Macedo et al. [61] statistically analyzed 2D LiDAR scans to distinguish vegetation and solid obstacles. Hebert et al. [62] proposed a classification method based on local shape features. Lalonde et al. [63] used 3D LiDAR to classify natural terrains into three classes, namely *surfaces*, *linear structures*, and *porous volumes* (foliage, grass). Wurm et al. [64] proposed vegetation classification based on the remission values of a laser sensor. Bradley et al. [65] used a multi-spectral camera to distinguish vegetation from other types of objects. Bradley et al. [66] also proposed vegetation detection by measuring chlorophyll content using a near-infrared (NIR) sensor.

Some studies estimate terrain elevation under occlusion by vegetation cover. Wellington et al. [67] proposed a method to adaptively predict the ground height in rough-terrain with vegetation. They used multiple sensors such as a stereo camera, an NIR camera, and two LiDARs to online learn the mapping from the sensor readings to the ground surface through experience.

Kim et al. [23] proposed scene recognition for mobile robots that considers traversable plants using visual and geometric information. In [23], a feature on an image patch is formed by color, texture, and structural information taken from a stereo camera. At the same time, the image patches are automatically labeled through the robot's experience of success or failure of navigation. The pairs of a feature and a label associated with the image patches are then used to train a classifier online. Sivakumar et al. [68] proposed a monocular camera-based visual navigation method for agricultural mobile robots. This method employs a convolutional neural network (CNN) to estimate the state of the robot relative to the crop rows, and navigate a robot by Model Predictive Control (MPC).

Kahn et al. [20] introduced a DNN model that takes sequence of images and action commands to predict future events (successful navigation or collision). Polevoy et al. [22] employed a similar model, but directory estimated terrain traversability. Gasparino et al. [21] proposed a method to estimate a robot's waypoints considering traversability induced by deformable vegetation, snow, sand, etc. In [21], images are self-labeled based on a pre-defined kinodynamic model of the robot. Baquero Velasquez et al. [69] proposed LiDAR-based crop row detection and robot state estimation targeting highly unstructured agricultural fields where the paths are partially covered by plant parts.

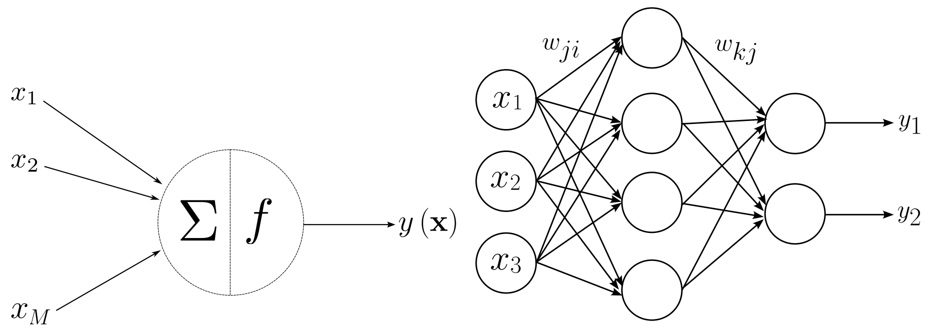
As in the aforementioned terrain analysis methods, self-supervised learning based on a robot's experience is also popular among traversability estimation considering deformable plants and other complex terrains [20, 21, 22, 23], as it allows them for easier data collection without tedious manual annotation. Moreover, since it is based on actual experience of success or failure of traversals, it does not require heuristic criteria on traversability. For its efficiency and effectiveness in data collection, Self-supervised learning is considered to be one of the most promising approaches to traversability estimation [70].

Limitations of current plant-induced traversability estimation

Some earlier attempts to detect vegetation [61, 62] use structural features from 2D LiDAR sensors. These methods consider vegetation on the ground surface growing vertically. In contrast, our work aims at recognition of plants that could be hanging over the path, or growing from the side of the paths, which can appear in some greenhouses and forest paths. Wurm et al. [64] used characteristics of laser reflection to distinguish vegetation from flat terrains. Some work use color information to detect vegetation [65, 66]. We argue that such detection is insufficient for scene recognition considering traversable plants, since plants have both traversable parts such as foliage and thin branches, and non-traversable parts such as stems and thick branches.

Kim et al. [23] proposed a pioneering work that explicitly recognizes object traversability. This method classifies image grids with a fixed size, and thus precise recognition is not possible. Moreover, the classification is based on a naive clustering-based method with hand-crafted features. In our method, we employ a DNN model for pixel-wise estimation of object classes and traversability for more precise recognition of traversable plants.

In the method by Kahn et al. [20], data are collected through a robot's autonomous operation and involves failures such as falling to ditches and crashing into obstacles. The methods by Polevoy et al. [22] and Gasparino et al. [21] require GNSS for robot state estimation in data collection and self-labeling. This limits the application of the method to open fields without obstacles that cause the multipath effect. In addition, [22] and [21] estimates only traversability on image pixels. We argue that object class information also plays an important role in traversability estimation. In fact, in Chapter 4, we show that jointly estimating object class and traversability can improve the accuracy of traversability estimation. Moreover, the aforementioned methods require both positive and negative experience of navigation to get binary labels to train an estimation model. Negative experience such as bumping into obstacles is hard and dangerous to get. In Chapter 4, we seek a possibility of learning the traversability with only positive experience employing Positive and Unlabeled learning (see Sec. 2.3.3).



(a) Perceptron (b) A multi-layer perceptron (MLP)

Figure 2.1: Network diagram for the ANN models

2.2 Deep Neural Networks (DNNs)

In a last decade, DNN techniques have brought technical breakthroughs in a broad range of research fields including computer vision, natural language processing, etc. We employ a DNN on RGB images to estimate object classes and traversability throughout the present work. In this section, we briefly review key techniques of DNN that are relevant to our methods.

2.2.1 Artificial Neural Networks (ANNs)

An ANN is a mathematical model of nonlinear function inspired from the network of neurons in mammalian brains [71]. It consists of interconnected unit neuron models shown in Figure 2.1(a). A neuron takes multiple inputs and calculates a weighted sum of the inputs, and finally apply a nonlinear activation function to yield a single output. Formally, given an M-dimensional input vector $\mathbf{x} = \{x_i\}_{i=1}^{M}$, the neuron model is formulated as follows:

$$y(\mathbf{x}) = f\left(\sum_{i=1}^{M} w_i x_i + w_0\right) \tag{2.1}$$

where w_i denotes a *weight* corresponding to x_i, w_0 denotes a *bias*, respectively. The model is called the *Perceptron* [72] $f(\cdot)$ denotes a nonlinear activation function. Originally, $f(\cdot)$ is given in the form of step function

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0. \end{cases} \tag{2.2}$$

A single perceptron is only applicable to cases where positive and negative data are linearly separable.

By stacking the perceptrons, a multi-layer perceptron (MLP) is built. Here, we consider an MLP with one hidden layer as shown in Fig. 2.1(b). From D input variables $x_1, \cdots, x_D$, M linear combinations are constructed as follows [73]:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \tag{2.3}$$

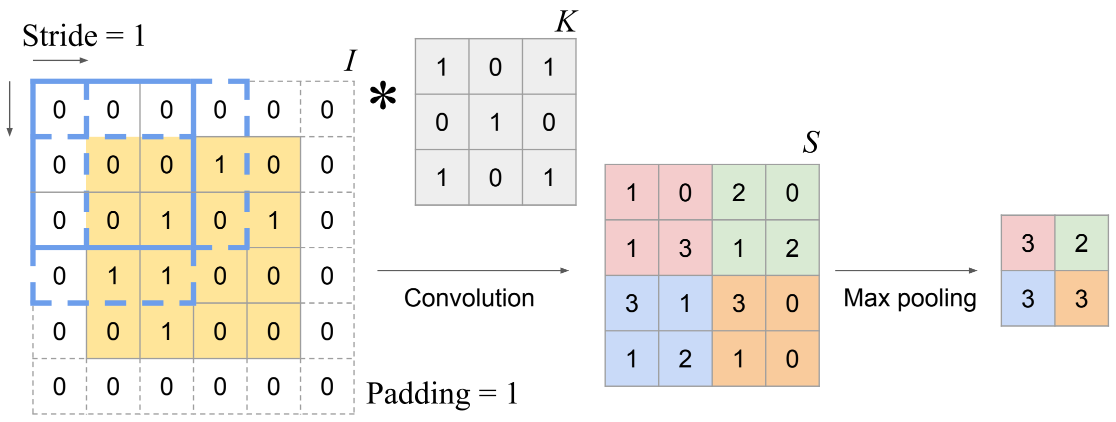


Figure 2.2: Basic operations in CNN. This example shows operations when stride=1, padding=1, and using max pooling with the receptive field of 2×2.

where $j = 1, \cdots, M$, and w_{ji} indicates a weight on i-th input element to calculate j-th linear combination. The value a_j is transformed through a nonlinear function $h(\cdot)$:

$$z_j = h(a_j). \tag{2.4}$$

These values are then linearly combined to give output values:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \tag{2.5}$$

where $k = 1, \cdots, K$, and K denotes the number of outputs. The output values are then activated by a nonlinear function $\sigma(\cdot)$. The choice of $\sigma(\cdot)$ depends on the assumed distribution of target values.

Overall, the network function can be written as follows:

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right), \tag{2.6}$$

where $\mathbf{x}$ denotes a vector of the group of all weight and bias parameters.

2.2.2 Convolutional Neural Networks (CNNs)

CNN is a specialized type of neural network that uses convolution operation with grid-like kernels [74]. It played a significant role in the drastic improvement in computer vision tasks in the last decade.

Basic operations in CNN

The overview of the operations in CNN is shown in Fig. 2.2.

Convolution Suppose we have two-dimensional input $I \in \mathbb{R}^{H_i \times W_i}$, and a two-dimensional kernel $K \in \mathbb{R}^{H_f \times W_f}$. The convolution operation defined between the input I and the kernel K is as follows:

$$S(i,j) = (K * I)(i,j) = \sum_{m=0}^{H_f} \sum_{n=0}^{W_f} I(i+m, j+n) K(m,n), \tag{2.7}$$

where i and j denotes the indices of a pixel in the row and the column axes, and m and n denotes the indices of the kernel in the row and the column axes, respectively.

Unlike MLPs where a neuron is densely connected with all downstream neurons, CNN has sparse connections where each neuron is connected to only a limited number of next neurons. It contributes to drastically reducing the number of learnable parameters compared to MLPs and thus reduces the model complexity. In addition, convolution also enables parameter sharing, that is, the same parameter is applied to for more than one input values. In contrast, in MLPs, one parameter is assigned to an input value and never reused elsewhere. It reduces the memory requirement to store the model [74]. Moreover, the shared weights allow for extracting local features with the same filters over a feature map. Therefore, a convolution kernel can be interpreted as a learnable image filter. This characteristic is suitable for tasks using images.

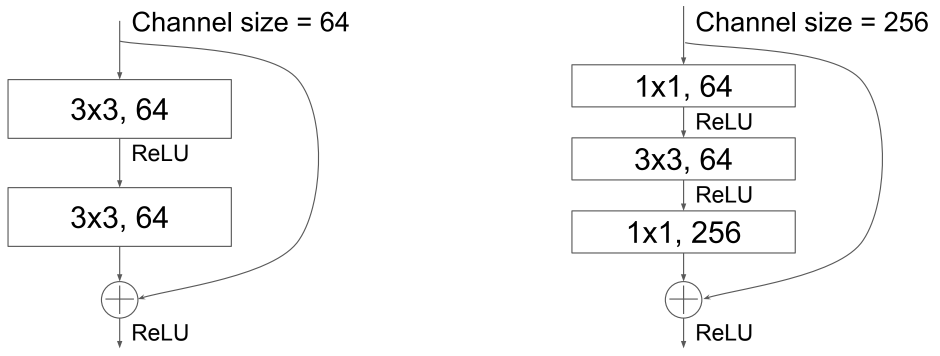
Pooling Pooling is an operation to summarize values in a specific local receptive field. At the same time, the pooling operation downsamples the input feature map. Among several types of pooling, the most used ones are *max pooling* and *average pooling.* Max pooling returns the maximum value, and average pooling returns the mean value of the values within the receptive field.

Padding When convolution is applied to the input with the original resolution, the size is reduced after the operation. To maintain the output size, we often append pixels with an arbitrary value around the input. Most general way is zero-padding which sets the values of padded pixels to 0.

History of CNNs

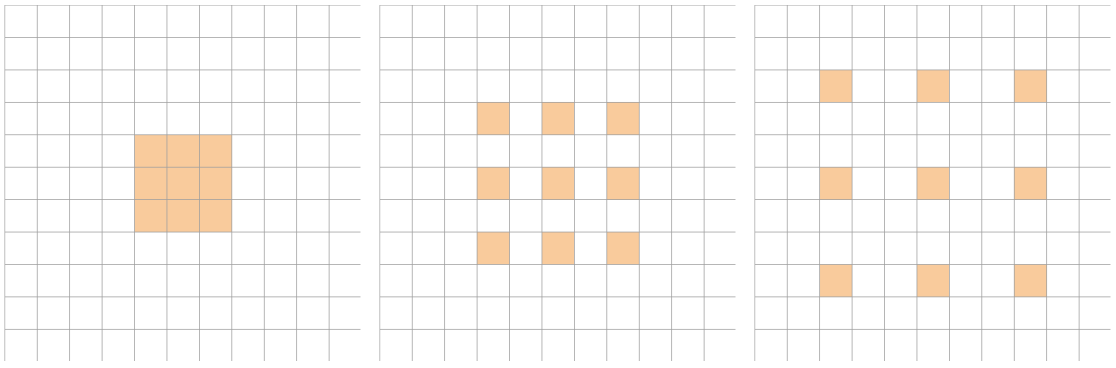
In 1980, Fukushima [75] proposed *Neocognitron*, which is today considered as the original deep CNN architecture. The design of Neocognitron was inspired by the structure of mammalian visual system. It consists of two different cell models, namely *S-cell* and *C-cell*, stacked alternately. The former takes inputs in a fixed size of receptive field from the previous layer, calculates the weighted sum of them, and outputs a value activated by a nonlinear function that is now called ReLU. The latter takes inputs from a group of S-cells, and responds strongly when at least one of the S-cells yields a large value of input. They are equivalent to convolution and pooling operation used in today's CNNs, respectively. After a decade, LeCun et al. [76] proposed a convolutional network with gradient-based training for a handwritten zip code recognition task. LeCun et al. also proposed an architecture called LeNet [77] for document recognition. In the 2010s, CNN-based methods such as AlexNet [78], GoogLeNet [79], VGGNet [80], and ResNet [81] have brought breakthroughs in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [82].

ResNet [81] is one of the most influential techniques in the deep learning research. In very deep neural networks, *vanishing gradient* is a critical problem, where error gradient diminishes during propagation to lower layers as the depth of the network becomes deeper. This problem hindered development of deeper networks, although it was known that making deeper networks improve the performance. To resolve this problem, ResNet introduced residual blocks with a skip connection from the input to the output to improve error propagation and thus enabled to build deeper networks. ResNet consists of multiple residual blocks stacked together. The visual illustration of a residual block is shown in Fig. 2.3. There are two types of residual blocks: *regular* and *bottleneck*, shown in Fig. 2.3(a) and 2.3(b), respectively. The regular residual block maintains the channel



(a) Regular residual block (b) Bottleneck residual block

Figure 2.3: Network diagram of a residual block [81]



(a) $l = 1$ (b) $l = 2$ (c) $l = 3$

Figure 2.4: Dilated convolution with 3×3 kernel and different dilation rate l

size during the convolutions. The bottleneck one first reduces the number of channels by 1×1 convolution, applies 3×3 convolution, and expand the channels back to the original size by 1×1 convolution. The bottleneck block is used to reduce the number of computations.

Types of convolution operations

Various types of convolution have been proposed to improve the network's expressive power and/or computational efficiency.

Dilated convolution or atrous convolution [83, 84] is used to aggregate multi-scale contextual information without losing resolution of the feature maps. In dilated convolution, given a parameter called the *dilation rate* l, $l-1$ spaces are inserted between kernel elements. The concept of dilated convolution is shown in Fig. 2.4.

Depthwise separable convolution [85] was proposed as a building block of MobileNet, an efficient CNN model intended for mobile and embedded vision applications. The depthwise separable convolution factorize a standard $D_K \times D_K \times C$ convolution on a feature map with the chanel size C into a composition of depthwise (i.e., per-channel) $D_K \times D_K \times 1$ convolution and a pointwise $1 \times 1 \times C$ convolution. Such factorization can reduce computation by $\left(\frac{1}{N} + \frac{1}{D_K^2}\right)$, where N denotes the number of output channels.

Factorized convolution [86, 87], similarly to the depthwise separable convolution, factorizes a standard $D_K \times D_K$ convolution into a $D_K \times 1$ convolution and a $1 \times D_K$ convolution. This factorization reduces the required parameters with insignificant performance degradation.

Group convolution divides a convolution layer into G groups of small filters along the channel axis. Group convolution was first applied to AlexNet [78, 88] to distribute training in multiple GPUs. Xie et al. [89] then applied group convolution in their model ResNeXt and showed that it does not only reduce the parameters but also improve the accuracy.

2.2.3 DNNs for semantic segmentation

Semantic segmentation is a task to predict an object class on each pixel of an image. It is one of the most important tasks in the field of computer vision and its application includes scene recognition for the navigation of self-driving cars and autonomous mobile robots.

Types of architecture

Long et al. [90] proposed the idea of converting classification networks into a fully convolutional network to produce a probability map for the input of arbitrary size. Most of the semantic segmentation models have been developed based on it [91, 92]. Ronneberger et al. [92] proposed the UNet architecture. It consists of symmetric encoder and decoder, and

Many recent high accuracy models use ResNet proposed by He et al. [81] as a backbone network to build very deep networks. Chen et al. [84] proposed DeepLab network where atrous convolution is used to enlarge the receptive field while maintaining the resolution of the feature maps. Atrous Spatial Pyramid Pooling (ASPP) used in DeepLab applies atrous convolution with multiple dilation rate to extract multi-scale features. Variants of DeepLab have then been developed [93, 94]. Zhao et al. [95] proposed Pyramid Scene Parsing Network (PSPNet). PSPNet shares a similar idea with [84] to aggregate multi-scale information, but uses multi-scale pooling operations instead. HRNet [96, 97] extract features in multiple resolutions in parallel while exchanging the information among the resolutions. It showed superior performance to the DeepLab family and PSPNet, which had been considered as de facto standards of semantic segmentation networks.

While it is proven to be effective to stack many layers to get a high accuracy, the performance comes at the high computational cost. To tackle this problem, computationally efficient models have also been studied, targeting the applications such as robotics and autonomous driving cars where real time recognition is required for the operation. Badrinarayanan et al. [91] proposed SegNet, which is one of the earliest semantic segmentation model with an encoder-decoder architecture. Based on a similar encoder-decoder structure, Paszke et al. [98] proposed ENet. ENet significantly reduces required operations and memory consumption by aggressively downsampling the input image in earlier stages, and employing a decoder smaller than the encoder. Although ENet is still one of the fastest model today, it largely sacrifices preciseness of segmentation results. ERFNet by Romera et al. [99] adopted replaced convolution operations in ResNet [81] with factorized convolution [86, 87] to explore a good trade-off between efficiency and cost. ESPNet [100] and ESPNetv2 [101] proposed by Mehta et al. employ an efficient spatial pyramid (ESP) modules to enlarge the receptive field while maintaining computational efficiency.

Applications

Image segmentation techniques have been applied in various fields. The aforementioned efficient models for semantic segmentation [91, 98, 99, 100, 101] are mainly intended for scene recognition of autonomous vehicles.

Some studies applied semantic segmentation in navigation tasks of mobile robots. Lulio et al. [102] proposed an image segmentation method for robot navigation in orchards with a combination of color-based region segmentation and an artificial neural network. Sharifi et al. [103], Aghi et al. [104], and Lin et al. [105] proposed DNN-based segmentation methods for the purpose of robot navigation.

Metrics

There are some metrics to evaluate the result of semantic segmentation. The most often used one is intersection over union (IoU), which compares the similarity between two arbitrary shapes [106]. Let the number of true positive, false positive, true negative, and false negative predictions as TP, FP, TN, and FN, respectively. IoU on the predictions is then calculated as follows:

$$IoU = \frac{TP}{TP + FP + FN}. \tag{2.8}$$

It is equivalent to calculating the proportion of the area of true positive predictions, which is the intersection of the ground truth positive regions and the predicted positive regions, over the union of the ground truth and the prediction, as the name suggests. In multi-class semantic segmentation tasks, we evaluate IoU on each class, as well as mean IoU, which is a simple average of the class-wise IoUs. We use IoU as the most basic metric of semantic segmentation throughout the thesis.

Other than IoU, metrics such as accuracy, precision, and recall are also used. These are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \tag{2.9}$$

$$Precision = \frac{TP}{TP + FP}, \tag{2.10}$$

$$Recall = \frac{TP}{TP + FN}. \tag{2.11}$$

Accuracy is the proportion of correct predictions on both positive and negative over all predictions. Precision indicates how much of the positive predictions were correct. Recall, on the other hand, indicates how much of ground truth positive regions were predicted correctly. Precision and recall are in a trade-off relationship because when a model predicts conservatively, precision will increase while recall will decrease, and vice versa.

2.3 Machine learning with limited data

In conventional machine learning including DNNs, a model is trained with a dedicated dataset with a large amount of input data associated with ground truth labels. The labels are usually given through manual annotation by human annotators, which is an extremely tedious and time-consuming task. For example, annotation of The Cityscapes dataset [107], an image dataset of urban scenes, required more than 1.5 hours on average per image. It is unrealistic to prepare such a dataset for every target task. To mitigate the need for manual labeling, researchers have proposed various approaches to train a model with a limited amount and quality of data. Here, we review the approaches that are relevant to the methods presented in the thesis.

2.3.1 Domain Adaptation (DA)

Domain adaptation (DA) is a type of machine learning task where labeled data in one or more *source* domains, to learn a classifier for unseen or unlabeled data in *target* domain in the same task [108]. DA is a special case of transfer learning (TL), where knowledge learned in a source domain in a task is exploited in a target task which is possibly different from the source task [109, 110]. DA has been attracting attention as a method to overcome the burden of annotation in ordinary machine learning and deep learning tasks. Since the range of applicable fields of DA is broad, in the following subsections, we summarize the existing methods of unsupervised domain adaptation (UDA), an actively studied subfield of DA where a labeled source dataset and an unlabeled target dataset are available for training, and especially UDA for image semantic segmentation.

Unsupervised domain adaptation

Semantic segmentation based on deep learning usually requires a large amount of training data labeled on each pixel and manual labeling is especially time-consuming. One promising approach is to use a large amount of image-label pairs automatically synthesized by computer graphics. Since the appearance of the synthetic images is different from real-world images, the model trained on the synthetic data does not perform well in the real world. Therefore, we need UDA to re-train the model in a different domain i.e., the real environment. There are two major approaches to UDA methods [111, 112]. One is called “domain alignment” where a model is trained so that the distribution discrepancy of domains is minimized. The alignment is done in different levels such as the input space [113, 114], feature space [115, 116], and output label space [117, 118]. Those methods aim to learn domain-invariant features of both domains. Recently, such training is often realized by adversarial training [119, 116].

The other approach of UDA is pseudo-label-based methods. This type of method uses outputs from a model pre-trained with a source dataset as true labels. By directly training the model on the target images with the pseudo-labels, it can fully learn domain-specific information. However, the pseudo-labels for the target images inevitably contain misclassification and it may affect the training as noise. To deal with the noise, different approaches have been proposed, such as thresholding based on the confidence of the prediction [120, 121], the adaptive weighting of loss values based on the uncertainty of the prediction [122], and prototype-based pseudo-label denoising [123].

Multi-source domain adaptation

Apart from the aforementioned single-source adaptation methods, there are several studies of multi-source domain adaptation (MDA). Mansour et al. stated that the target distribution can be represented as a weighted mixture of multiple source distributions. Following this analysis, Xu et al. [124] implemented an MDA training with deep networks. For a detailed survey of MDA, readers are referred to Zhao et al. [125]. Zhao et al. [126] proposed multi-source Domain Adaptation for Semantic Segmentation (MADAN) by extending CyCADA [113] to MDA for semantic segmentation. He et al. [127] proposed an unsupervised MDA method for semantic segmentation. In [127], a simple image translation in LAB color space is used to images from the multiple source datasets to minimize the domain gap, instead of GAN-based methods. In training, they train a model for each source dataset by collaborative learning where output distributions of object class probabilities for an image from the models are matched via Kullback-Leibler Divergence (KLD) to let them learn domain-invariant semantic contexts across different domains. One of the major contributions of [127] is that it outperformed existing methods like MADAN without any adversarial learning or any other sophisticated tricks, and thus easy to train. Park et al. [128] extended the work of He et al. by introducing a pseudo-label rectification method using multiple source models.

A work of Nigam et al. [129] utilizes multiple models trained with different source datasets as feature extractors, and adapt the segmentation layers to a target dataset of overhead images from a drone. Although their method is similar to ours in a way that multiple pre-trained networks are exploited, their task is supervised domain adaptation where they prepare their own labeled dataset, while our method utilizes publicly available datasets that are not specifically built for greenhouse scenes.

Benchmarks

As a benchmark of semantic segmentation, a task considered de-facto standard is adaptation from synthetic datasets such as the GTA5 dataset [130] and the SYNTHIA dataset [131] to a real dataset, most often Cityscapes dataset [107]. In this task, scenes are limited to urban scenes in both the source datasets and the target datasets. Besides, the set of object classes is identical in all the datasets. In Chapter 3, we further investigate the methodology of transferring knowledge in cases where the structure of the environments is different, and the source and the target datasets do not share their label space.

2.3.2 Few-shot Learning (FSL)

Few-shot learning is one of machine learning paradigms where limited new samples with ground truth labels (*support set*) are given for reference to make inference on data with object classes unseen in pre-training (*query set*). For the pre-training of the network, an external dataset with abundant labeled data is usually used.

The idea of few-shot image classification was first extended to image segmentation by Shaban et al. [132]. The first one-shot segmentation approach was proposed by Shaban et al. [132]. In [132], the features of the target objects in support images are aggregated via masked average pooling within given ground truth masks, and the pooled features are used to produce classification weights for segmentation of a query image. A major approach to few-shot segmentation is to aggregate features of support images within masks and taking cosine similarity with features of query images [133, 134]. In few-shot segmentation, a support set with ground truth mask is exploited to aggregate features and the aggregated features are used to produce classification weights for segmentation of a query image [132, 133, 134].

Qi et al. [135] proposed a simple method for few-shot image classification coined weight imprinting, where normalized features of a target object are directly used as weights of the classifier. They claim that weight imprinting performs well even without fine-tuning. Based on the weight imprinting approach, Siam et al. [136] proposed the Adaptive Masked Proxies (AMP) for effective feature aggregation specifically designed for few-shot image segmentation. Our online learning is inspired by [135] and [136] for their simplicity.

2.3.3 Positive and Unlabeled Learning (PU Learning)

PU learning is a problem where part of the data is labeled as positive, and the rest is unlabeled and could be either positive or negative. Unlike the setting where positive and negative data are assumed, PU setting cannot be solved by just applying a simple classification of positive and unlabeled data since the unlabeled data include positive ones. Elkan and Noto [24] showed that it is possible to estimate the label probabilities by modeling the probability that given samples are labeled under *selected completely at random* assumption. Detailed review can be found in Bekker et al. [137].

As a practical application, Yang et al. [138] applied PU learning framework to object detection task where some objects are not labeled. Suger et al. [32] also introduced PU learning in traversability estimation based on hand-crafted structural features.

Chapter 3

Multi-source Pseudo-label Learning of Semantic Segmentation

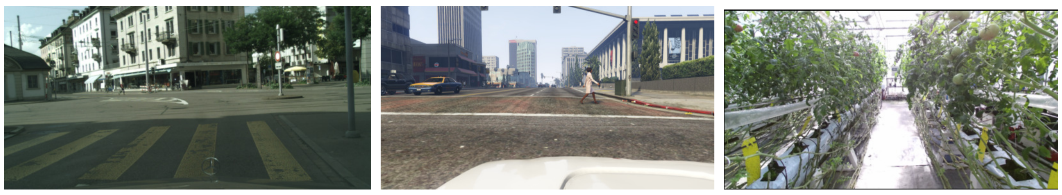
3.1 Introduction

Semantic segmentation based on Deep Neural Network (DNN) is widely used in the scene recognition system of self-driving vehicles and autonomous mobile robots. We also employ DNN-based semantic segmentation to estimate general object classes on each image pixel. Usually, DNNs are trained using a large amount of data with hand-annotated labels to ensure high accuracy and generalization. However, hand-annotation is time-consuming and physically demanding. In the case of introducing a mobile robot in new environments, it is not realistic to prepare such a fully labeled dataset for every environment even though a model should be trained specifically for the environment.

To address this problem, one might consider applying a model trained with existing publicly available image datasets with ground truth labels. The performance of pre-trained models, however, often deteriorates on different data domains due to the difference in data distributions between the training dataset and the target scenes, known as "domain shifts" [108]. Domain adaptation (DA) is a task to adapt a model trained on one data domain to another domain with minimal degradation of performance. Unsupervised domain adaptation (UDA) is a type of DA where labeled source datasets and unlabeled target datasets are available. UDA can mitigate the burden of hand-annotation by leveraging information in the source domain and transferring it to the target.

A major task of UDA is adaptation from labeled synthetic images automatically acquired in a simulator to unlabeled real images. The adaptation task from GTA 5 [130] or SYNTHIA [131] to Cityscapes [107] is one of the most widely used benchmarks and many studies have verified their effectiveness on these tasks. However, it is often difficult to prepare appropriate source datasets with full segmentation labels for target environments in the wild. In the case of greenhouses, there is no real image dataset of greenhouses with segmentation labels, and rich simulation environments applicable to our task.

Instead, we consider utilizing publicly available rich image datasets of the real world not specifically designed for greenhouses. Fig. 3.1 shows real and synthetic urban scenes



(a) Cityscapes [107] (b) GTA5 [130] (c) Greenhouse

Figure 3.1: Difference of appearance between an urban scene and a greenhouse. The image of a real urban scene (a) and the image of a simulated urban scene (b) share similar distribution of objects, and the difference mainly stems from the texture. On the other hand, (a) and the image of a real greenhouse (c) have a large discrepancy of the structure of the environments.

as well as a greenhouse scene for comparing the difference of appearance. The difference between real and synthetic urban scenes (Figs. 3.1(a) and 3.1(b)) mainly stems from the discrepancy of their characteristics of appearance such as textures. However, they share a similar distribution of the objects in the images, e.g., the majority of areas are occupied by buildings and roads. On the other hand, the images of real urban scenes and the greenhouses (Figs. 3.1(a) and 3.1(c)) are both real, but there are instead structural differences such as the distribution and scales of the objects in the images, etc. These structural differences of the datasets makes the existing UDA methods less effective. In fact, as later shown in Sec. 3.3.3, state-of-the-art UDA methods result in poor performance.

While the outputs of the source models contain a lot of noise, we also observed that all the models provide accurate classification on large areas of the images (more details are in Sec. 3.2.3). Based on the observation, we propose to exploit multiple datasets as source datasets for pseudo-label generation, rather than relying on a single dataset dissimilar to the target dataset. Our key contribution is the method to generate pseudo-labels using models trained with multiple publicly available datasets to effectively transfer knowledge to structurally different target images without ground truth labels. Pseudo-labels for the target images are generated by merging outputs from multiple models, each of which is pre-trained with a labeled source dataset. In particular, a pseudo-label is assigned on a pixel only when all the source models predict the pixel as the same object class. This method effectively filters out misclassified labels and leads to a better performance of the trained models. To further suppress the effect of noise in the pseudo-labels, we also introduce a loss weighting based on the uncertainty of estimation and pseudo-label denoising inspired by Zheng et al. [122] and Zhang et al. [123], respectively.

We demonstrate the effectiveness of the method in the experiments of adaptation from three source datasets of urban or outdoor scenes, to multiple greenhouse datasets as the target.

3.2 Proposed Method

3.2.1 Overview

The purpose of this work is to train a semantic segmentation model that achieves low error on greenhouses images without hand-annotated labels. To this end, we consider using multiple publicly available labeled image datasets as source datasets to transfer

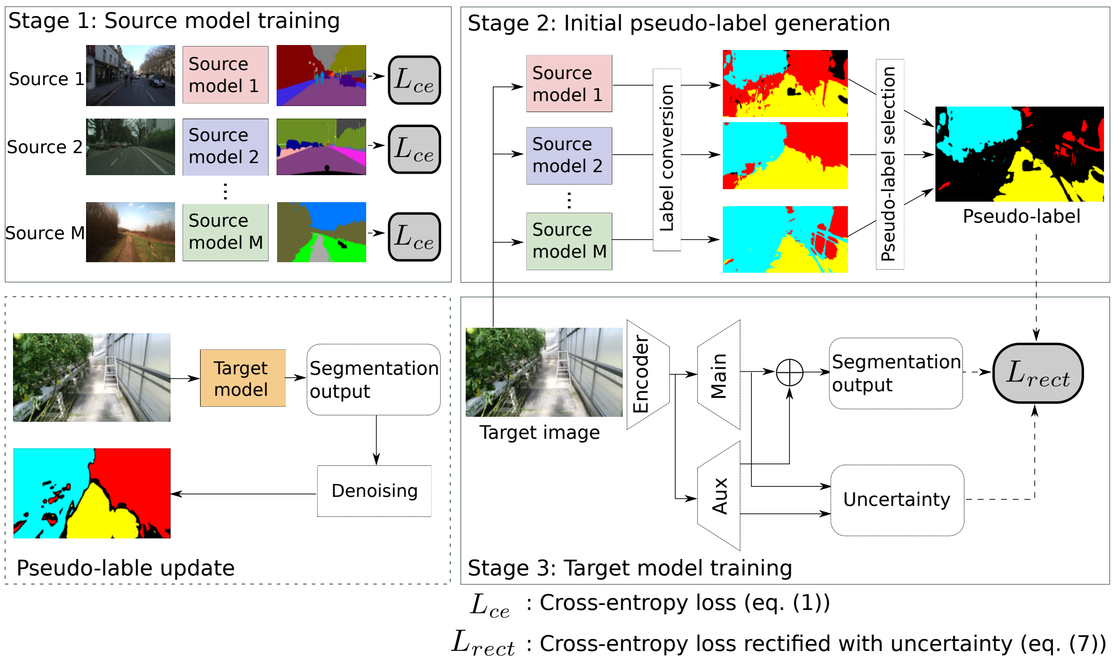


Figure 3.2: Overview of the proposed pseudo-label learning for semantic segmentation. Stage 1: Train models with the source datasets in supervised training. Stage 2: Pseudo-labels are generated for the target images using the output from the source models. Stage 3: Train a target model using the generated pseudo-labels. (Best viewed in color)

knowledge to the target images. Specifically, we use datasets of urban scenes and outdoor scenes, namely, CamVid dataset [139], Cityscapes dataset [107], and Freiburg Forest dataset [140], as source datasets. Each source dataset has a different set of object classes. The target datasets are greenhouse images with the following classes: *plant*, *artificial object*, and *ground*. The target datasets do not have any label data for supervision.

Formally, we assume M source datasets $S_1, \ldots, S_M$ and an unlabeled dataset S_T. A source dataset S_i consists of an image set $X_i = \{x_{i,j}\}_{j=1}^{N_i}$ and a set of corresponding segmentation label maps $Y_i = \{y_{i,j}\}_{j=1}^{N_i}$, where N_i denotes the number of images in S_i. The source datasets can have a different number of object classes. The target dataset S_T consists of only an image set $X_T = \{x_{T,j}\}_{j=1}^{N_T}$, where N_T denotes the number of images in S_T.

Fig. 3.2 shows the overview of the proposed method. It consists of three stages. The first stage is the training of source models. For each source dataset, a segmentation model is trained in a supervised manner. In the second stage, the target images are fed into the source models and pseudo-labels are generated by merging the outputs from the source models. An output from each source model is first converted to the target labels using a label conversion function, which is heuristically defined. The pseudo-labels are then generated by selecting the labels that the models unanimously predict as the same label. Finally in the third stage, the target model is trained with the pairs of the target images and corresponding pseudo-labels. The classification loss is adaptively weighted by the uncertainty of the prediction as proposed in [122]. During the training, the pseudo-labels are updated using the outputs from the currently trained model for further leveraging the knowledge learned from the initial pseudo-labels. In the pseudo-label update, a denoising method inspired by [123] is used.

In our method, we employ ESPNetv2 [101], a light-weight semantic segmentation model, as base network architecture for its suitability to real-time scene recognition of mobile robots. The proposed method, however, is not dependent on a specific network architecture.

3.2.2 Model pre-training

We first train a segmentation model on each source in a fully supervised manner. In this step, each model is trained with the original classes in the corresponding source dataset. A standard cross entropy loss is used as a loss function:

$$L_{ce}(x, y; \theta) = -\sum_{h,w} \sum_{c \in C} y^{(h,w,c)} \log F\left(x; \theta\right)^{(h,w,c)}, \tag{3.1}$$

where $F\left(x; \theta\right)^{(h,w,c)}$ denotes the probability of class c at the location (h, w) for an image x predicted by the segmentation model with weights θ, and $y^{(h,w)}$ denotes a one-hot label at pixel (h, w). A source model for dataset S_i is initialized as follows:

$$\theta_i = \arg\min_{\theta} \sum_{(x,y) \in S_i} L_{ce}(x, y; \theta). \tag{3.2}$$

Usually, in loss calculation, greater weight is set for the loss of rarer classes in order to deal with class imbalance. For our task, however, we empirically found that a model provides more accurate pseudo-labels when the model is trained with a weight proportional to the frequency of the label, which is contrary to the usual practice. We suppose the reason is as follows. The usual loss weighting is mainly for capturing relatively small objects such as road signs and poles. In our target environment, i.e., greenhouses, such a fine-grained perception is not necessary because the space of the environment is fixed and thus the variation of the scale of the objects is limited. Moreover, the object classes in the target datasets are coarse. Therefore, the usual practice of training may lead to a model that is overly focused on small objects which is unnecessary for the pseudo-label generation.

3.2.3 Pseudo-label generation using multiple pre-trained models

Underlying idea

Our intuition behind this method of multi-source pseudo-label generation is that while the prediction from each source model contains a lot of misclassifications, the models correctly predict many of the pixels in common. Fig. 3.3 shows outputs for a target image from different source models trained with the CamVid dataset [139], Cityscapes dataset [107] and the Freiburg Forest dataset [140] as well as the target image, hand-labeled ground truth, and the pseudo-label generated from the three outputs. As can be seen in Fig. 3.3, many of the true regions of “plant” class and “ground” class are correctly classified in all the models. This implies that getting a consensus of the models is effective for generating precise pseudo-labels as shown in Fig. 3.3(c). Another thing worth noting is that each model has its own characteristics of prediction. For example, the prediction of the model trained with Freiburg Forest dataset on the ground regions

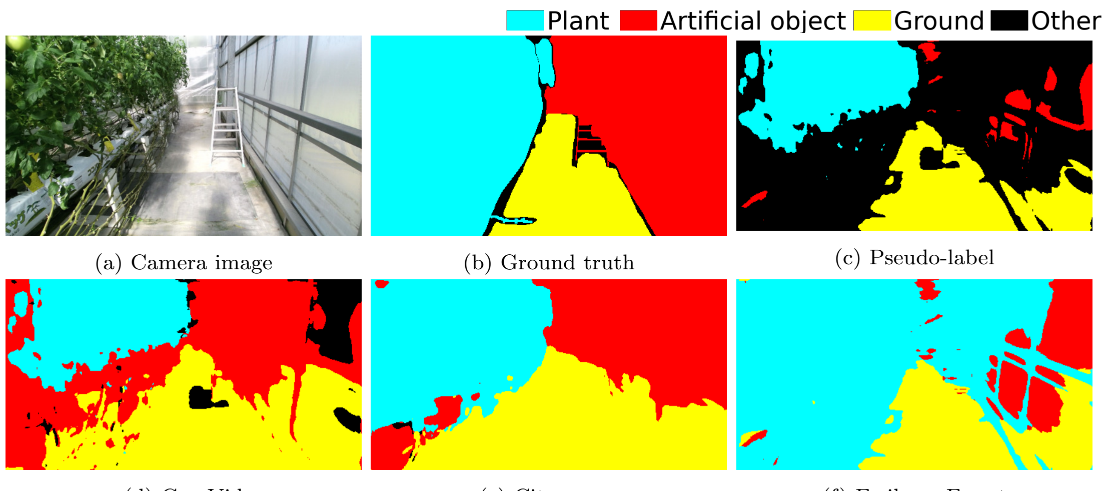


Figure 3.3: Examples of output from each source model and resulting pseudo-label. The predictions by the source models (d)-(f) are correct on many of the pixels, though they are not perfect. By taking consensus of the models on each pixel, we can get fairly precise pseudo-labels as shown in (c).

better captures the shape of true ground regions than the other two models. However, the prediction on the plants and artificial objects is not accurate. On the other hand, in the model trained with Cityscapes, the prediction on the ground region includes a lot of false positives, while that on the other objects is more accurate than the Freiburg Forest model. These differences stem from the structural differences of the source datasets. The proposed method avoids biased training to the structural features of a specific source dataset dissimilar to the target by getting agreements of multiple source models.

Pseudo-label generation method

In this step, pseudo-labels for the target images are generated using outputs from the models pre-trained with the source datasets. First, the target images are fed in all the pre-trained models to yield predicted labels in the label space of each source domain. The predicted label from the model of source i for the jth target image is given as follows:

$$p_{i,j}^{(h,w)} = \arg\max_{c} F\left(x_{T,j};\theta_i\right)^{(h,w,c)}. \tag{3.3}$$

The predicted labels are then mapped to the target label space using a label conversion function $\xi_i : \mathcal{Y}_i \rightarrow \mathcal{Y}_T$, where $\mathcal{Y}_i$ and $\mathcal{Y}_T$ denote a set of labels in source S_i and target S_T, respectively:

$$p'^{(h,w)}_{i,j} = \xi_i(p_{i,j}^{(h,w)}). \tag{3.4}$$

Here, we heuristically define the mapping from the source classes to the target classes for each source domain as shown in Table 3.1. After the label conversion, for each pixel of the target images, a predicted label is assigned to the pixel if the prediction is the same among all the source models. Otherwise, “other” label is assigned and not used for training. Formally, a pseudo-label for the pixel (h, w) of the jth target image $x_{T,j}$ is generated as follows:

$$\hat{y}_j^{(h,w)} = \psi\left({p'_{1,j}}^{(h,w)}, \cdots, {p'_{M,j}}^{(h,w)}\right) = \begin{cases} c_k & \text{if } \forall i \in \{1, ..., M\}, p'_{i,j} = c_k \\ c_\phi & \text{otherwise,} \end{cases} \tag{3.5}$$

where $c_k \in \mathcal{Y}_T$ is a class category in the target dataset and c_ϕ is a class label "other" that is not used for loss calculation during the training and thus does not affect the training.

3.2.4 Model training on the target data

We train a segmentation model with the target images and the corresponding pseudo-labels. Although the valid labels in the pseudo-labels generated in Sec. 3.2.3 are unanimously predicted by the source models and thus highly likely correct, we introduce uncertainty-based loss weighting and prototype-based pseudo-label denoising for further suppressing the effect of noise in the pseudo-labels.

Loss weighting based on the uncertainty of estimation

In standard training of semantic segmentation models, the cross entropy loss is widely used. In the training with pseudo-labels, however, equally treating all the pseudo-labels may lead to poor performance due to noise in the pseudo-labels. Zheng et al. [122] proposed an adaptive loss weighting that decays as the uncertainty of the estimation increases so that the losses on the pixels with high uncertainty do not affect the training much. Their method exploits an auxiliary segmentation branch to predict pixel-wise uncertainty of the label prediction. We attach an auxiliary segmentation branch to ESPNetv2 for uncertainty estimation. The entire segmentation network is shown in Fig. A.1 in Appendix A.

Given the output of primary branch $F(x;\theta)$ and that of the auxiliary branch $F_{aux}(x;\theta)$, the uncertainty of the prediction is defined as Kullback-Leibler Divergence (KLD) of the distributions of class probabilities from the two branches:

$$D_{kl}\left(x;\theta\right)^{(h,w)} = \sum_{c \in C} F(x;\theta)^{(h,w,c)} \log \frac{F(x;\theta)^{(h,w,c)}}{F_{aux}(x;\theta)^{(h,w,c)}}. \tag{3.6}$$

Using eq. (3.6), the loss function is defined as follows:

$$L_{rect}\left(x, \hat{y}; \theta\right) = \sum_{h,w} \left(\exp\{-D_{kl}\left(x;\theta\right)^{(h,w)}\} L_{ce}\left(x, \hat{y}; \theta\right)^{(h,w)} + D_{kl}\left(x;\theta\right)^{(h,w)}\right). \tag{3.7}$$

Prototype-based pseudo-label denoising

In the middle of the training, we update the pseudo-labels with the outputs from the current model to further leverage the knowledge learned from the initial pseudo-labels. We introduce pseudo-label updating with prototype-based denoising inspired by Zhang et al. [123].

At first, the prototypes are calculated. The prototype of class c is a mean of the features that belong to the class c and is calculated using the current pseudo-labels as follows:

$$\eta^{(c)} = \frac{\sum_{(x,\hat{y})\in S_T}\sum_{h,w} f^{(h,w)} * \mathbb{1}\left(\hat{y}^{(h,w)} == c\right)}{\sum_{(x,\hat{y})\in S_T}\sum_{h,w} \mathbb{1}\left(\hat{y}^{(h,w)} == c\right)}, \tag{3.8}$$

where $f^{(h,w)}$ is an intermediate feature provided by the currently trained model $F(\cdot;\theta)$ at pixel (h,w), and $\mathbb{1}$ is the indicator function that returns 1 when its argument is true and 0 otherwise.

Pseudo-labels for image $x_{T,j}$ are then updated as follows:

$$\hat{y}_j^{(h,w)} = \begin{cases} \arg\max_c \left(\omega^{(h,w,c)} F\left(x_{T,j};\theta\right)^{(h,w,c)}\right) & \text{if } \max_c \left(\frac{\omega^{(h,w,c)} p_j^{(h,w,c)}}{\mu}\right) > \alpha \\ c_\phi & \text{otherwise,} \end{cases} \tag{3.9}$$

where $\mu = \sum_{c'} \omega^{(h,w,c')} p_j^{(h,w,c')}$ is a normalization factor and α is a threshold for pseudo-label selection, which is set to 0.9 in the experiments. ω is a vector of weight values to modulate the class probabilities, calculated using the distances between the feature and each prototype. For each pixel, the weight ω is calculated as follows:

$$\omega^{(h,w,c)} = \frac{\exp\left(-\left\|f^{(h,w)} - \eta^{(c)}\right\|/\tau\right)}{\sum_{c'} \exp\left(-\left\|f^{(h,w)} - \eta^{(c')}\right\|/\tau\right)}, \tag{3.10}$$

where τ is a temperature parameter that controls the degree of bias of the distribution, which is set to 1 in [123]. The values of ω indicate the class likelihood of the feature based on the distance from the feature to each prototype, and the probabilities predicted by $F(\cdot;\theta)$ are rectified by ω.

3.2.5 The overall algorithm

The overall algorithm is shown in Algorithm 1. As already mentioned, our training pipeline consists of three stages. A segmentation model for each source dataset is trained in standard supervised learning in the first stage and initial pseudo-labels for the target images are generated using the pre-trained models in the second stage. In the third stage, the target model is trained using the target images and the pseudo-labels. The training is done in several "rounds", each of which consists of several epochs. Here, the number of epochs per round is set to 5, and the maximum number of rounds is set to 10.

During training, the pseudo-labels are updated using the target model currently being trained. In this work, we update the pseudo-labels once after the third round of training with the initial pseudo-labels because the training with the initial pseudo-labels converges by the third round. We compare the different pseudo-label update strategies in Sec. 3.3.5.

3.3 Experiments

3.3.1 Experimental setup

Training conditions

We use PyTorch implementation of ESPNetv2 [101]. For estimating uncertainty, an auxiliary segmentation branch is attached. The architecture is shown in Fig. A.1 in

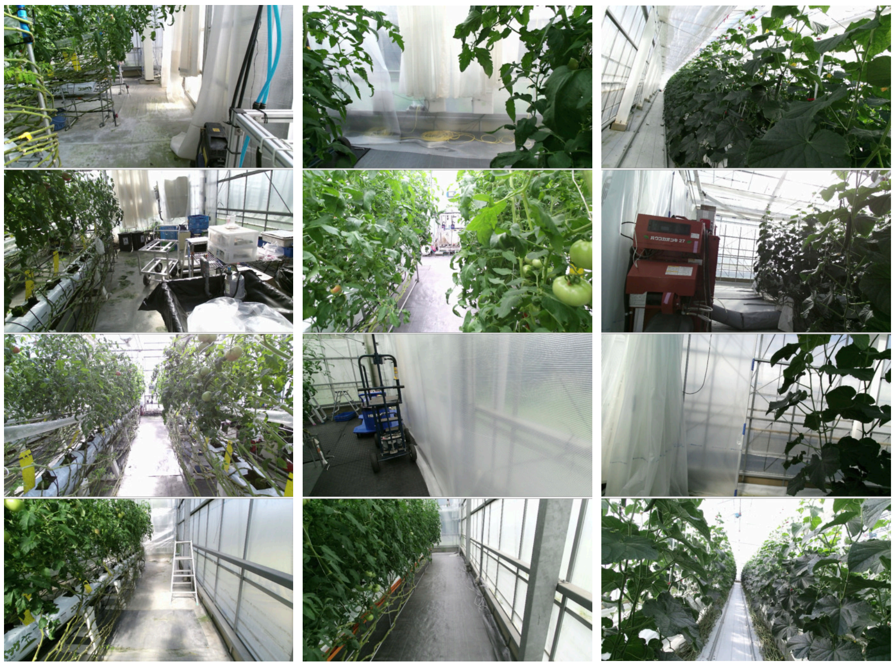
Algorithm 1 Multi-source pseudo-label learning

```
Input: Source dataset $S_i$, label conversion functions $\xi_i$ with $i \in \{1, \cdots, M\}$, Target
    dataset $S_T = \{X_T\}$
Output: The target model $F(\cdot; \theta_T)$
 1: # Step 1: Source model pre-training
 2: Train the source model $F(\cdot; \theta_i)$ with $S_i = \{X_i, Y_i\}$ and $L_{ce}$
 3:
 4: # Step 2: Generating initial pseudo-labels
 5: Get outputs for the target images: $p_i \leftarrow F(x_T; \theta_i)$
 6: Label conversion: $p'_i \leftarrow \xi_i(p_i)$
 7: Generate pseudo-labels: $\hat{y} \leftarrow \psi(p'_1, \cdots, p'_M)$
 8:
 9: # Step 3: Target model training
10: for $round \leftarrow 0$ to $max\_round$ do
11:     if $is\_round\_to\_update\_pseudo\_labels$ then
12:         Calculate the prototypes by eq. (3.8)
13:         Update the pseudo-labels by eq. (3.9)
14:     end if
15:
16:     for $k \leftarrow 0$ to $epoch\_per\_round$ do
17:         for $j \leftarrow 0$ to $len(X_T)$ do
18:             Train the model with $\{X_T, \hat{Y}\}$ and $L_{rect}$
19:         end for
20:     end for
21: end for
```

Appendix A. The processing time was about 44 [msec/image], or 23 [fps]. The number of floating point operations (FLOPs) is 0.79 [GFLOPs] while that of the original ESPNetv2 for semantic segmentation is 0.76 [GFLOPs] [101], indicating that attatching the

Table 3.1: Label conversion from the source datasets to the target sets

CamVid	**Cityscapes**	**Forest**	**Greenhouse A, B, C**
Tree	Vegetation	Grass, Tree	Plant
Building, Pole, SignSymbol, Fence, Car, Road_marking	Building, Wall Fence, Pole, Traffic light, Traffic sign, Car, Truck, Bus, Train, Motorcycle, Bicycle	Obstacle	Artificial object
Road, Pavement	Road, Sidewalk, Terrain	Road	Ground
Sky, Pedestrian Bicyclist, Unlabeled (Not used in the training)	Sky, Person, Rider, Background	Sky	Other



(a) Greenhouse A (b) Greenhouse B (c) Greenhouse C

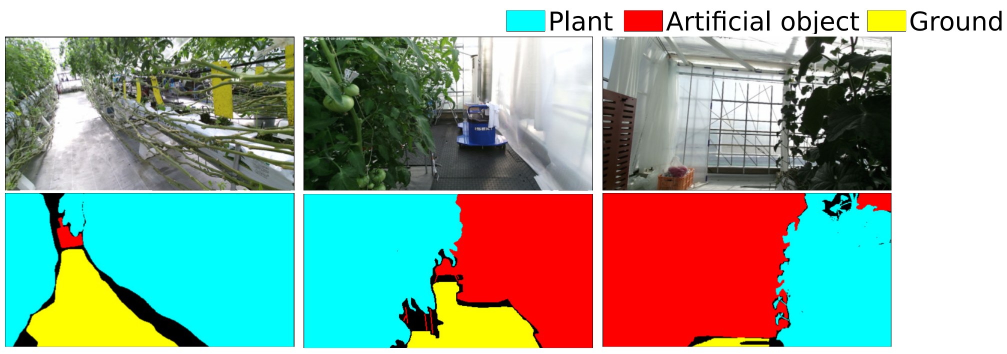
Figure 3.4: Example of images of the target datasets. Greenhouse A and B are the same greenhouse that grows tomatoes, but taken in different times and thus the lighting condition and growth of leaves differ. Greenhouse C is a greenhouse of cucumber that has different appearance from tomato plants.

auxiliary branch did not affect the computational efficiency. All training of the proposed method is performed on one NVIDIA Quadro RTX 8000 with 48GB RAM. The numbers of training rounds and epochs per round are set to 10 and 5, respectively. The learning rate is fixed to 5×10^{-4} and the batch size is 48. Adam [141] is used as an optimizer. Weights of the target models are initialized by the supervised training of semantic segmentation on the CamVid dataset [139], except for the last classification layer initialized with random values. We measure segmentation performance with the Intersection over Union (IoU) metric.

Datasets

We use CamVid [139], Cityscapes [107] and Freiburg Forest dataset [140] (hereafter referred to as CV, CS, and FR, respectively) for the training of the source models. CV, CS, and FR have 367, 2975, and 230 labeled training images, respectively. Label conversions from the source datasets to the target datasets (ξ_i) are heuristically defined as shown in Table 3.1 . A more detailed analysis of the label mapping is in Appendix B.

As target greenhouse datasets for adaptation, we acquired images from three greenhouses; Greenhouse A that grows tomatoes, Greenhouse B that is the same greenhouse as Greenhouse A but acquired in different date and time, and Greenhouse C that grows cucumbers. Each target dataset has 6689 unlabeled images. Example images of the target datasets are shown in Fig. 3.4. For each target dataset, we prepared test images with manually annotated labels for evaluation. The numbers of the test images are 100 for Greenhouse A, and 50 for B and C. The labels on the test images include the "other"



(a) Greenhouse A (b) Greenhouse B (c) Greenhouse C

Figure 3.5: Example of the test images. Regions of each object class are roughly labeled, rather than labeled with pixel-level precision. For example, the plant rows in Greenhouse A are uniformly labeled as "plant"

class and the pixels with the class are not considered in the calculation of the metric (See Fig. 3.5). More detailed annotation policy is described in Appendix C.

3.3.2 Comparison to single-source baselines

We conducted an experiment of adaptation from the source outdoor datasets to each of the target greenhouse datasets. In the single source baselines (CV, CS, and FR), the initial pseudo-labels were generated via the pseudo-label update procedure described in Sec. 3.2.4 in the source label space $\mathcal{Y}_\rangle$, followed by the label conversion by ξ_i to map the labels to the target label space. All the models were trained with the initial pseudo-labels for the first three rounds. After that, the pseudo-labels were updated using the hard pseudo-labels with the prototype-based denoising and used in the rest of the training.

Table 3.2 shows the result of the training. The results of "no adapt" were provided by directly feeding the target images to the source models followed by the label conversion. Before the adaptation, the IoUs of the source models were low and not applicable to real applications. The performance of single-source training depended on the source dataset. In some cases, the adaptation resulted in improvement from the model without adaptation, while in some others the performance deteriorated after the adaptation. The cause of the degradation is that the training progressed in the wrong direction due to the noisy pseudo-labels. This indicates that the training with a single source may not be reliable enough when the source and target datasets have a large domain shift.

With our multi-source adaptation, on the other hand, the accuracy was significantly improved even though each source model is not reliable on the target data. The results show the ability of our method to transfer useful common knowledge about the appearance of objects from the source datasets to the target. The two-source training resulted in comparative or even better performance than the three-source training (CV+CS+FR). It is worth noting that the combination of different types of environments results in especially high accuracy, i.e., FR+CV and CS+FR. CV (CamVid) and CS (Cityscapes) are datasets of urban scenes, while FR (Freiburg Forest) consists of images of outdoor scenes with rough terrains and vegetation, This implies that we should use datasets with a high variety of environments as sources.

The most suitable combination, however, depends on the target dataset in the two-

Table 3.2: Result of the adaptation in IoU (CV: CamVid, CS: Cityscapes, FR: Freiburg Forest. The best result for each target is shown in **bold** and the second best result is underlined.)

Target	**Source**	**Class IoU**			**mIoU**
		Plant	**Artificial object**	**Ground**	
Greenhouse A	CV (no adapt)	64.72	56.19	46.34	55.75
	CS (no adapt)	63.71	68.71	36.37	56.26
	FR (no adapt)	60.24	45.27	30.49	45.33
	CV	70.21	68.66	57.10	65.32
	CS	60.17	71.47	31.02	54.22
	FR	52.73	50.68	40.84	48.11
	CV+CS	73.81	73.49	47.55	64.94
	CS+FR	79.50	76.35	70.32	75.39
	FR+CV	77.40	73.00	63.20	71.20
	CV+CS+FR	**80.71**	**78.21**	**72.68**	**77.20**
Greenhouse B	CV (no adapt)	65.57	65.30	57.65	62.84
	CS (no adapt)	83.33	80.09	49.68	71.03
	FR (no adapt)	48.47	54.34	19.17	40.66
	CV	71.37	77.15	62.73	70.42
	CS	83.49	77.39	44.53	68.47
	FR	47.04	58.60	42.87	49.50
	CV+CS	**87.19**	87.00	60.96	78.38
	CS+FR	82.56	**89.44**	70.60	80.87
	FR+CV	75.99	85.70	60.48	74.06
	CV+CS+FR	83.56	89.08	**73.37**	**82.33**
Greenhouse C	CV (no adapt)	58.89	58.29	19.82	45.66
	CS (no adapt)	79.09	81.10	18.11	59.43
	FR (no adapt)	61.65	49.24	25.56	45.48
	CV	69.96	70.30	22.52	54.26
	CS	85.51	85.30	21.86	64.23
	FR	66.00	65.52	0.6549	44.06
	CV+CS	88.76	87.20	28.61	68.19
	CS+FR	84.99	82.26	29.56	65.61
	FR+CV	91.12	**90.00**	**45.12**	**75.41**
	CV+CS+FR	**91.16**	88.84	40.67	73.56

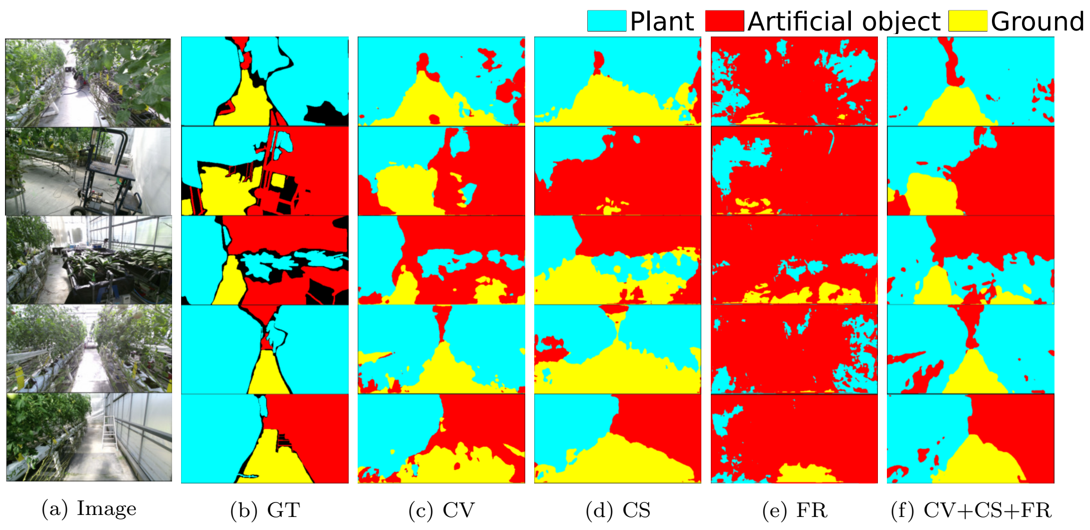


Figure 3.6: Result of the adaptation on Greenhouse A. The prediction by the multi-source model (f) is more precise than those of the single-source models (c)-(e).

source setting. For example, CS+FR resulted in the second-best performance on Greenhouse A and B, while it was the worst among the multi-source settings on Greenhouse C. On the other hand, the three-source training consistently performed the best or the second-best on all the target datasets. We, therefore, suggest exploiting the three source datasets for the segmentation task in greenhouse environments.

Fig. 3.6 is the result of the adaptation on Greenhouse A. In Fig. 3.6, while the results of the single-source training are relatively noisy, the multi-source ones provide smoother outputs. In particular, the plant regions are classified more accurately. Qualitatively, we suppose that this level of accuracy is sufficient for the operation of mobile robots. For Greenhouse B and C, the results are shown in Fig. D.1 and D.2 in Appendix D. From Figs. D.1 and D.2, we can see that accurate segmentation can be achieved in other environments, and thus the knowledge that the proposed method transfers from the source datasets is applicable in a variety of greenhouse environments.

3.3.3 Comparison to existing methods

Baseline methods

To evaluate the relative performance of our proposed method, we conducted training with existing methods of supervised learning as well as single-source and multi-source UDA for semantic segmentation. Implementation details of the baseline methods are described in Appendix E.1.

Supervised learning methods We introduce a supervised support vector machine (SVM) on superpixel features and a supervised DNN as supervised baselines. The SVM method is equivalent to some existing segmentation methods such as Sharifi et al. [103] and Lulio et al. [102], where an image is partitioned into small segments and then classified based on their hand-crafted features. As a DNN model, we use the architecture that is the same as the proposed method. We refer to the SVM method as *SP-SVM* and the DNN method as *SP-DNN*. Implementation details of SP-SVM are in Sec. E.1.1.

For evaluation of the supervised baselines, we conducted 10-fold cross-validation with the 100 labeled images of Greenhouse A, which are used as the test data in the experiments of the UDA methods. The reported results are averages of IoUs on the test fold of each of the 10 training trials. Note that the training setting and the evaluation condition are different from the UDA training. This is due to the limited amount of manually labeled greenhouse images available. Nevertheless, we report the results of the supervised baselines to demonstrate baseline performances that we can expect from supervised learning.

UDA As existing UDA methods for semantic segmentation, we introduce representative UDA methods in different generations, i.e., confidence regularized self-training (CRST) [121] proposed in 2019, Seg-Uncertainty [122] and ProDA [123] proposed in 2021. They employ self-supervised learning with pseudo-labels, which is a popular approach in current UDA methods for semantic segmentation. Seg-Uncertainty and ProDA also involve GAN-based training based on [117] to further enhance the adaptation. Seg-Uncertainty and ProDA are the original works of the uncertainty-based loss weighting and the pseudo-label denoising in our method, respectively.

We used the source code distributed by the authors with modifications to apply them to our experiments, such as adding the new datasets and converting the source labels to the target label set defined in Table 3.1. In terms of ProDA, we report the result of stage 1 proposed in [123]. Although the original method employs knowledge distillation after stage 1 to boost the performance, they reported the state-of-the-art performance at the time with only the training of stage 1. We, therefore, suppose the result of stage 1 on our setting is sufficient as a baseline. The rest of the training process follows the algorithm of each method. Implementation details are in Appendix E.1.2.

Although there are also multi-source UDA methods such as Multi-source Domain Adaptation for Semantic Segmentation (MADAN) [126], we could not train it due to limitations of computational resources. In Appendix E.2.2, we show results of image style transfer between the source dataset and the target greenhouse dataset, which is the first step of MADAN.

Source datasets

In the comparative studies of single-source UDA methods, we report the results on Cityscapes (CS) since it provided best results than CV and FR) in all of the single-source baseline methods. We suppose this may be because CS has a sufficient amount of images (about 3000) for training models with a large capacity such as DeepLab v2, while the other source datasets (CV and FR) have only a few hundreds. On CV and FR, the model may have overfit to the training sets due to the scarcity of the data, and thus resulted in poorer adaptation performances. As there was no significant difference of adaptation performance between the source datasets in the single-source setting shown in Sec. 3.3.2 (see Table 3.2), we believe it is reasonable to use CS to demonstrate the performance of the baselines on behalf of the three source datasets.

Results

Table 3.3 shows the results. Overall, the proposed method outperformed the baseline methods except for SP-DNN, and thus exhibited the ability to transfer knowledge from the datasets dissimilar to the target dataset. Although the performance of the baseline

Table 3.3: Results of the baseline methods. CS is used in the single-source UDA baselines. **Bold** denotes the best performance among the UDA methods

Method	**Class IoU**			**mIoU**
	Plant	**Artificial object**	**Ground**	
SP-SVM*	74.99	54.66	53.54	61.06
SP-DNN*	86.27	78.64	88.49	81.68
CRST [121]	78.89	76.85	65.10	73.62
Seg-Uncertainty [122]	52.17	70.68	27.29	50.04
ProDA [123]	56.15	60.81	28.53	48.50
CV+CS+FR (Proposed)	**80.71**	**78.21**	**72.68**	**77.20**

*For SP-SVM and SP-DNN, we report averages of 10-fold cross-validation using 100 labeled images.

methods might be improved with extensive hyper-parameter search, the results indicate that relying on a single source is not effective enough even when using state-of-the-art UDA methods in our task where the source and target datasets do not share their structure of the scenes. Qualitative results are shown in Appendix E.2. While SP-DNN resulted in the best IoUs, it also showed a clear tendency of overfitting due to the scarcity of training data. Although increasing the training data will resolve the problem, it requires laborious manual annotation. On the other hand, the proposed method enables training with a large amount of unlabeled images and provides a comparative performance.

Among the baseline UDA methods, CRST resulted in better performance than others, although they are reported to be better than CRST [122, 123]. We conjecture that adversarial learning negatively contributed to these results. ProDA and Seg-Uncertainty involve adversarial learning in the training processes to make the spatial structure in the output space as close as possible between the source and the target datasets [117]. Adversarial learning, including GANs [119], is effective when the difference between the source and the target stems mainly from the styles such as texture and color and the datasets share similar structure. For instance, GANs can be used to transfer the image style of source images closer to the target images without significantly changing the image contents, i.e., semantic class of corresponding image regions, so that a segmenter for the target images can be trained on the transferred source images in the target's style with the ground truth labels. However, when the structure of the scenes is different, adversarial learning does not necessarily convey meaningful information. In fact, as shown in Appendix E.2.2, the image contents are not preserved between the original and the translated images after CycleGAN-based image style transfer between CS/FR and Greenhouse A. For example, some ground regions in CS are transferred to green plant-like stuff, and some plant regions are transferred to regions with sky-like appearance (See Fig. E.2). In [126], the authors also pointed out that the structural differences of objects between the source and the target datasets led the training to poor performances on those object classes (e.g., the source in target images are much taller than those in target images). Those facts imply adversarial learning is not as effective for adaptation under such structural differences between the source and the target scenes.

Table 3.4: Comparison of multi-source merging strategies

Majority	Unanimity (proposed)
70.03	**77.20**

3.3.4 Comparison of strategies to merge multi-source information

To see the validity of the proposed pseudo-label generation based on unanimity of the source models, we conducted training with a different strategy, namely majority-based pseudo-label generation. In the majority-based strategy, a label is assigned to a pixel when more than two source models of the three predict the pixel as the same object class.

The results are shown in Table 3.4. The training with the majority-based pseudo-labels resulted in a significant drop in performance compared to the unanimity-based method. We suppose that since the majority-based method more aggressively involves pixels in the pseudo-labels, more noise was included and it affected the training. We can conclude that the pseudo-labels should be chosen carefully through the strict unanimity criteria for better training. Although the pseudo-labels generated in such a way become coarse, the model trained with them provides reasonable predictions as shown in the aforementioned experiments. The validity of training with coarse labels is also reported in [142].

3.3.5 Effect of updating the pseudo-labels

We evaluate the effect of pseudo-label update strategies. We compare the strategies as follows.

- **Initial** the initial pseudo-labels are used throughout the training.
- **Update once** the pseudo-labels are updated after three rounds (i.e., 15 epochs).
- **Update every round** the pseudo-labels are updated every round after the first three rounds.

The results are shown in Table 3.5. *Update once* outperformed the other strategies. We suppose the reason is as follows. The initial pseudo-labels are sparse and many pixels are ignored in training. Although it plays a crucial role to transfer primary knowledge from multiple source datasets, using such labels throughout the training leads to a suboptimal result. Updating the pseudo-labels with *Update once* strategy enables to exploit the knowledge learned through the training with the initial pseudo-labels and involve more pixels in the training to further improve the performance.

Table 3.5: Results of the training with and without pseudo-label update

Initial	Update once	Update every round
75.44	**77.20**	74.82

Table 3.6: Ablation on the pseudo-label noise suppression strategies

Confidence threshold	Loss weighting [122]	Prototype-based denoising [123]	mIoU
			73.95
✓			74.66
✓	✓		75.89
✓		✓	75.53
✓	✓	✓	**77.20**

Table 3.7: Results of the training with different values of threshold α

α	0.7	0.8	0.9	0.95
mIoU	76.87	77.16	**77.20**	76.63

While the label update improved the performance, updating the labels every round resulted in the worst IoU. This is an expected result because the pseudo-labels inevitably include wrong ones due to misclassifications and the errors are accumulated as the pseudo-labels are updated by the model trained with them. This problem is also pointed out in [123]. We, therefore, conclude that the pseudo-labels should be updated once in the early stage of the training.

3.3.6 Ablation on pseudo-label noise suppression strategies

We conducted an ablation study on the methods for noise suppression. We examined the methods as follows: confidence thresholding, loss weighting, and prototype-based denoising. The confidence thresholding is simply removing outputs with a confidence value below a predefined threshold from the pseudo-labels. Here, the confidence threshold is set to 0.9 in all the settings. The loss weighting and the prototype-based denoising are the methods described in Sec. 3.2.4 and 3.2.4, respectively.

We report the training results on Greenhouse A. The results are shown in Table 3.6. The strategies for suppressing the effect of noise in pseudo-labels effectively improved the training performance. The combination of the loss weighting and the prototype-based pseudo-label denoising on top of the simple confidence thresholding resulted in the best performance. We, therefore, adopted this setting to all the other training we report in the following sections.

3.3.7 Parameter sensitivity analysis

We analyze the effect of the confidence threshold in the pseudo-label generation. We trained models with different thresholds on Greenhouse A dataset. The results are shown in Table 3.7. $\alpha = 0.9$ resulted in the best performance. However, the difference among the parameters is less than one point and thus the method did not show significant sensitivity to the parameter choice.

3.4 Discussion and future work

3.4.1 Limitations of the *hard* pseudo-label generation strategy

In the proposed method, pseudo-labels are selected for training only when all the source models unanimously make predictions on the corresponding pixels. Although such a conservative strategy contributed to removing noise as shown in Sec. 3.3.4, it also has some drawbacks. Here, we list the limitations of the proposed method.

1. **The limitation of valid labels**

   The conservative strategy naturally leads to a limited amount of valid labels, which affects effective training. This problem will be prominent especially when one of the source model performs badly and does not agree with others. This also hinders increasing the number of source datasets since the more the source models are, the harder it becomes to get agreement of all the source models and thus the produced pseudo-labels become too sparse.

2. **The limitation of applicable environments**

   The current method assumes that all source datasets are close enough to the target dataset to produce reliable pseudo-labels because, as mentioned above, valid pseudo-labels are not produced if one of the source models performs badly. Practically, the pseudo-label generation should adaptavely change the weight on each source dataset depending on the similarity between each source and the target dataset to be applicable to a variety of targets, which is not possible in the current method.

3. **The limitation of capability to learn novel object classes**

   The current method assumes that the label conversion ξ_i on all source datasets are surjective. This does not hold in cases where, for example, the target has *grass* class and *tree* class, while a source dataset only has *plant* class. In such a case, the current method cannot involve both of those classes in training. There may also be cases where the target dataset has classes that do not appear in the source datasets, e.g., objects specific to the environment. The proposed method is not capable of learning such novel objects.

3.4.2 Attempts to improve the pseudo-label generation and preliminary results

To overcome the aforementioned limitations, we are working on improving the pseudo-label generation method. Here, we briefely summarize the possible solution and some preliminary results.

Basic idea

The aforementioned limitations stem from the conservative nature of the pseudo-label selection in the current method. On the one hand, it contributes to removing possibly wrong labels by taking agreement of all source models. On the other hand, however, it completely ignores information of the pixels that are excluded from the training.

Moreover, it does not consider the similarity between the source datasets and the target, and treats their outputs equally. Intuitively, the predictions of a model trained on a dataset more similar to the target should be prioritized more than others.

Based on the consideration above, we consider exploiting the predicted probability distribution as well as the one-hot labels. In addition, we also take into account the information of the domain similarity between each source dataset and the target dataset. By using the soft pseudo-labels generation considering the domain similarity, we are expecting improvements as follows:

1. Exploiting information about the prediction confidence, which is ignored in the hard method
2. Adjusting the contribution weight of each source dataset depending on the similarity to the target
3. Involving object classes that appear in only some of the source datasets in training

Overview of the method

First, a semantic segmentation model $F(\cdot;\theta_i)$ is trained on each training dataset S_i in the same way as the current method (see Sec. 3.2.3).

Second, soft pseudo-labels are generated using the pre-trained source models $F(\cdot;\theta_i)$. Specifically, the outputs from the source models are weighed with the domain similarity between the target dataset and the corresponding source dataset, summed up, and normalized to an object probability distribution on each pixel (see Fig. 3.7). To calculate the domain similarity, we employ a method by Liu et al. [143] which evaluates the domain gap based on the entropy of the prediction of the source models on the target data. In [143], the domain gap G_i between source dataset S_i and target dataset S_T is calculated as follows:

$$G_i = \frac{1}{N_T \log C_i} \sum_{j}^{N_T} E\left(p_{i,j}\right), \tag{3.11}$$

where $E(\cdot)$ denotes entropy of the prediction defined as follows:

$$E\left(p_{i,j}\right) = -\sum_{h,w} \sum_{c=1}^{C_i} p_{i,j}^{(h,w,c)} \log\left(p_{i,j}^{(h,w,c)}\right). \tag{3.12}$$

The domain similarity is calculated as inverse of the domain gap. Using the domain similarity, soft pseudo-labels are generated as follows:

$$\hat{y}_{T,j}^{(h,w)} = \mathrm{Norm}\left(\sum_{i=1}^{M} \frac{1}{G_i} \psi_i\left(p_{i,j}^{(h,w)}\right)\right), \tag{3.13}$$

where $\mathrm{Norm}(\cdot)$ denotes a function to normalize the values so that $\sum_c \hat{y}_{T,j}^{(h,w,c)} = 1$, and $\psi_i : \mathbb{R}^{C_i} \to \mathbb{R}^{C_T}$ denotes a function to convert a probability distribution in the source label space to the target label space.

Finally, the target model $F(\cdot;\theta_T)$ is trained using the soft pseudo-labels. Instead of using the ordinary cross entropy loss (eq. (3.1)) ignoring the background class c_ϕ, we

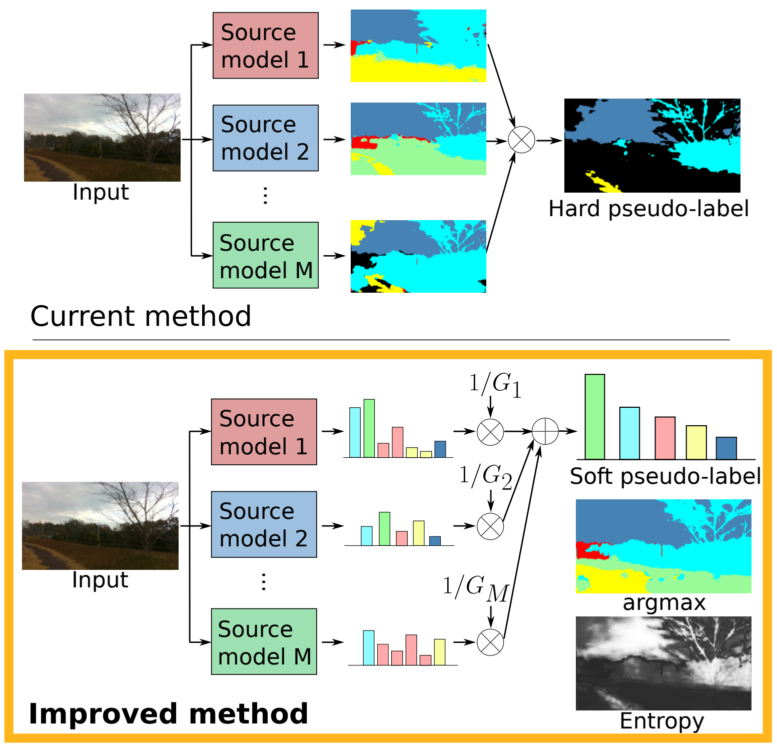


Figure 3.7: Overview of the improved pseudo-label generation. **Top**: Current method to generate pseudo-labels using multiple source segmentation models. A pixel is assigned with a label only if all models agree with each other to generate precise labels. If a class is not included in one dataset, the class never appears in the resulting pseudo-labels. **Bottom**: Proposed method. Instead of generating one-hot labels, the proposed method merges predicted object probabilities by weighing them with the domain similarity, which is calculated as inverse of the domain gap. It emphasize the prediction of model that is more certain. It can also involve labels that are not present in part of the source datasets. By using the argmax of the soft pseudo-labels as training labels and weighing the loss on each pixel with the inverse entropy of the soft pseudo-labels, we can approximate the training using the hard pseudo-labels.

weigh the cross entropy with the value that decays as the entropy of the soft pseudo-label increases:

$$L_{w\_ce}^{(h,w)} = W^{(h,w)} \log F\left(x;\theta_T\right)^{(h,w,\hat{l}^{(h,w)})}, \tag{3.14}$$

where $W^{(h,w)} = \exp\left(-\lambda_{scale} \cdot E\left(\hat{y}^{(h,w)}\right)\right)$ is the pixel-wise weight based on the entropy of the soft pseudo-label, and $\hat{l}^{(h,w)} = \arg\max_c \hat{y}^{(h,w,c)}$ is the label of the class with the highest score. We replace L_{ce} in eq. (3.7) with eq. (3.14).

Analysis of the soft pseudo-label

Fig. 3.8(b), 3.8(c), and 3.8(d) show an example of hard pseudo-label generated by our previous method, argmax $\hat{l}$, and the entropy-based certainty weight W_{ent} of the soft pseudo-label $\hat{y}_{T,j}$, respectively. The hard pseudo-labels in our previous method are generated based on unanimous prediction by multiple source models in one-hot label format. Although this conservative strategy contributes to maintaining the accuracy of the labels, it still has some wrong ones. In Fig. 3.8(b), for example, part of the wall is

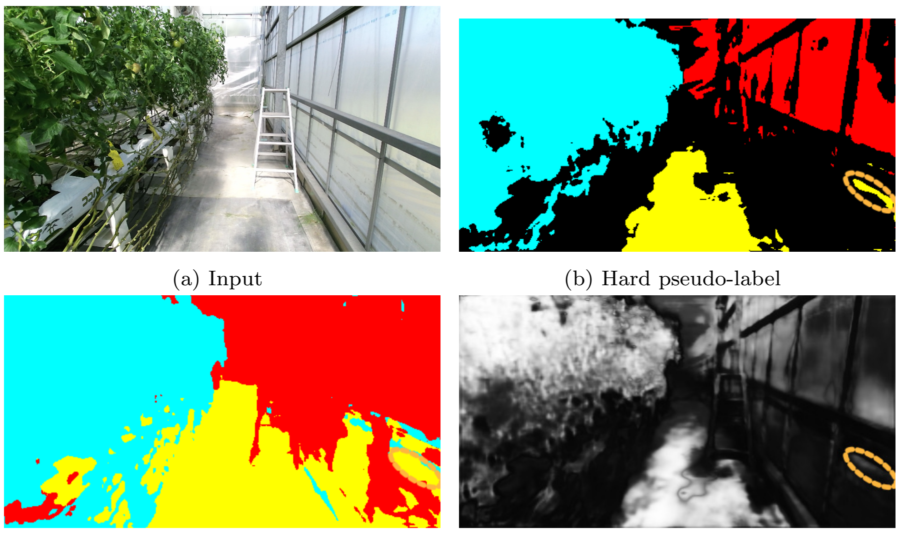

(a) Input (b) Hard pseudo-label (c) argmax (d) Certainty

Figure 3.8: Example of hard and soft pseudo-label. (a) Input. (b) Hard pseudo-label generated by the current method. Precise labels are extracted by merging the outputs from the source models based on unanimity. Some parts are misclassified (circled in orange). Those misclassified pixels and correct pixels are treated equally, which will affect training. (c) argmax, and (d) certainty (inverse of entropy) of the soft pseudo-label. The misclassified pixels in hard pseudo-label is assigned with a lower weight than the correctly classified pixels. It can suppress the effect of the misclassified pixels.

misclassified as *ground* (circled in orange). Once selected as a valid label, the correct ones and wrong ones are treated equally, which may lead to worse performance.

The advantage of the soft pseudo-label over the hard one is that it can utilize the information of the prediction certainty of the source models. In Fig. 3.8(d), weight on the misclassified part is less than correctly classified part. Unlike the hard method that treats all labels equally, we can expect the soft method to mitigate the effect of such wrong assignment of the label as misclassification often occurs on pixels with high prediction uncertainty [123]. We can also see that pixels with a high certainty weight have a valid pseudo-label. This indicates that weighing the loss with the certainty weight can be thought as a soft alternative of the previous unanimity-based hard pseudo-label generation.

In addition, as described in Fig. 3.7, the new method allows for involving object classes that are present in only some of the source datasets because it does not immediately exclude labels on which the models do not agree with each other, and rather accumulates predicted probabilities with certainty weights. If some models predict the class with high certainty, that prediction may be used as the label.

Preliminary results

We conducted training on two target datasets, namely Greenhouse A dataset used in the experiments of the current method, and *TUT Campus* dataset, consisting of images inside the campus of Toyohashi Univerisity of Technology including paved pedestrian roads and unpaved paths with ground vegetation. As source datasets, we use CamVid (CV), Cityscapes (CS), and Freiburg Forest (FR) dataset as in the previous experiments.

Table 3.8: Comparison of training with the hard and soft pseudo-labels

Hard (majority)	Hard (unanimity)	Soft
70.03	**77.20**	74.08

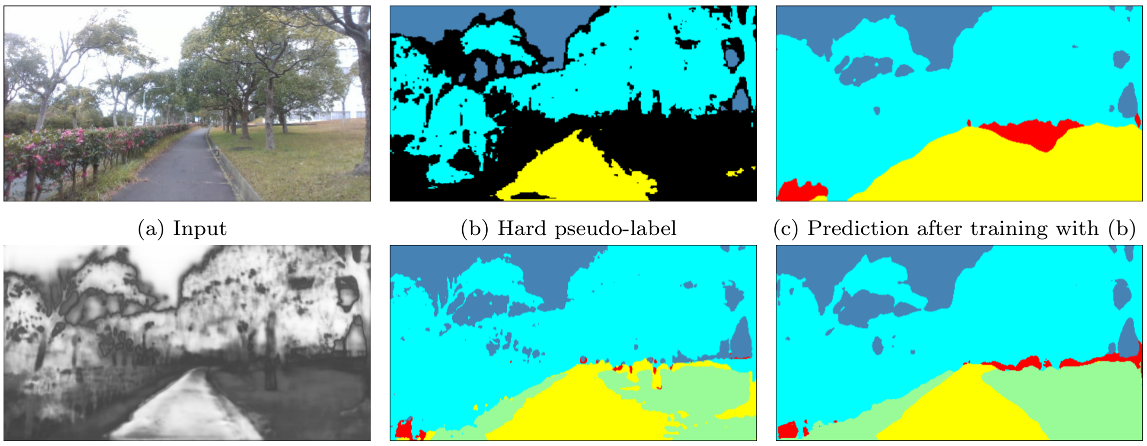

(a) Input (b) Hard pseudo-label (c) Prediction after training with (b)
(d) Certainty of soft pseudo-label (e) argmax of soft pseudo-label (f) Prediction after training with (d)(e)

Figure 3.9: Results of training with the soft pseudo-labels on *TUT Campus* dataset

Greenhouse A dataset We trained a model with the three classes used in the previous experiments. Table 3.8 shows the comparison with the hard pseudo-label generation method. The soft pseudo-label generation method did not perform as well as the current hard method based on unanimity. However, it surpassed the majority-based hard pseudo-labels, showing a better potential as a way to exploit information beyond the unanimously classified labels. Further improvement on training strategy will potentially provide even better performance.

TUT Campus dataset In this target dataset, we use five object classes that dominate the scenes, namely *plant* (trees etc.), *grass* (ground vegetation etc.), *artificial object*, *road*, and *sky*. CS and FR have a distinction between *plant* (*vegetation* in CS, and *tree* in FR) and *grass* (*terrain* in CS, and *grass* in FR). CamVid, however, only has *tree* class that corresponds to *plant* in the target dataset (see Table 3.2).

We trained a model with 859 unlabeled images. Fig. 3.9 shows examples of the hard and soft pseudo-labels, and resulting predictions after training using the two types of labels. From the figure, we can see a clear limitation of the current method that the generated pseudo-labels do not include *grass* class. This is because CV does not have a corresponding class, and thus the source models can never get agreement with each other on the class. As a result, most of the grass regions were excluded from the training, and those regions were misclassified as ground. On the other hand, the soft pseudo-label involved *grass* as well, and resulted in more reasonable prediction. The result exhibit a potential of the method on applicability to a more variety of target scenes.

Future prospect and chellenges

The preliminary results showed a potential of the soft pseudo-label generation method. Especially the results on TUT Campus dataset showed an advantage of the new method,

Table 3.9: Quantitative results on TUT Campus dataset

Method	**Class IoU**					**mIoU**
	Plant	**Grass**	**Artificial**	**Ground**	**Sky**	
Hard	**83.97**	0.0	51.57	16.39	80.96	46.58
Soft	82.96	**44.11**	**67.68**	**27.72**	**81.31**	**60.76**

that is a capability of training a model with object classes not included in some of the source datasets. It will improve the applicability of the method to a wider variety of target scenes. However, the training method is still not optimal, and has a large room for improvement. Particularly, we need to explore a way to exploit the soft pseudo-labels more effectively. Quantitative evaluation on manually labeled data is also necessary. In addition, we also need to further investigate how the domain similarity influence the pseudo-label generation on different kinds of scenes to see the adaptability of the method to various environments.

3.5 Summary

In this chapter, we tackled a task of unsupervised domain adaptation from multiple outdoor datasets to greenhouse datasets, to train a model for scene recognition of agricultural mobile robots. We considered using multiple publicly available datasets of outdoor images as source datasets since it is difficult to prepare synthetic datasets of greenhouses as done in conventional work. To effectively exploit the knowledge from the source datasets whose appearance and structure is very different from the target environment, we proposed a pseudo-label generation method that takes outputs from each source model and selects only the labels that are unanimously predicted by all the source models. By the combination of our pseudo-label generation method and conventional methods, training of semantic segmentation on the greenhouse datasets is enabled without manual labeling of images. This contributes to reducing the burden of applying a visual scene recognition system of mobile robots in greenhouses.

We also summarized the recent progress on improving the method of pseudo-label generation to overcome some limitations.

In the next chapter, we describe a method of scene recognition considering traversable plants. We train a model for predicting general object classes using the method described here.

Chapter 4

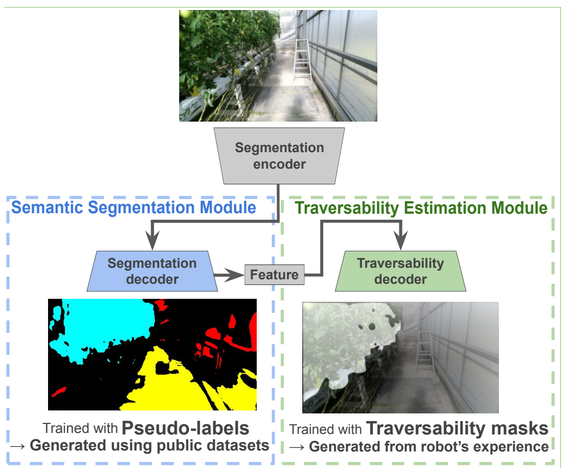
Image-Based Scene Recognition for Robot Navigation Considering Traversable Plants

4.1 Introduction

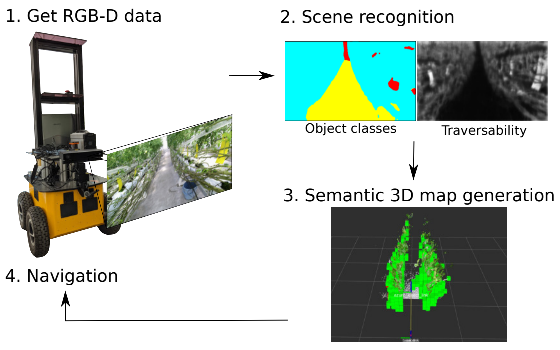
In this chapter, we present a novel framework of scene recognition and robot navigation that takes into account traversable plants covering the paths. To recognize objects as traversable, it is not sufficient to just recognize plants because plants have both traversable parts such as branches and leaves, and non-traversable parts such as stems. Datasets with such fine-grained classes are not publicly available to the best of our knowledge. To provide information of *traversability* of objects, which cannot be acquired from existing datasets, we use label data indicating the regions with objects that the robot has traversed during data acquisition phases, coined *traversability masks*. Since the traversability masks are generated based on a limited number of the robot's traversal experiences, all traversable regions are not labeled but some of them are left unlabeled. Directly using such masks as labels of traversable plants will confuse the training. To take full advantage of the incomplete information of the traversability masks, we present a deep neural network (DNN) architecture and its training method inspired by PU (positive and unlabeled) learning [24], where a model is trained with labeled positive examples and unlabeled examples including both positive and negative ones.

The overview of the proposed method is shown in Fig. 4.1. The proposed DNN architecture consists of Semantic Segmentation Module (SSM) for pixel-wise general object classification and Traversability Estimation Module (TEM) for estimating traversability which indicates how likely that each pixel can be traversed by robots. SSM is trained using the method described in Chapter 3. TEM is then trained using the features from SSM as inputs and the traversability masks. For the training of TEM, we introduce a training method inspired by PU learning [24] in order to take full advantage of the information given by the traversability masks. The overall network is therefore trained without manual annotation of the training images.

The prediction results (the object class and the traversability of each pixel) are then projected to the 3D space to build a semantic 3D voxel map used in navigation. For each data frame, the prediction results (observations) are temporally fused with the prior probabilities of the object classes and the traversability in each voxel using


Segmentation encoder
Semantic Segmentation Module
Traversability Estimation Module
Segmentation decoder
Feature
Traversability decoder
Trained with Pseudo-labels
→ Generated using public datasets
Trained with Traversability masks
→ Generated from robot's experience

(a) Network architecture



(b) Navigation pipeline

Figure 4.1: Overview of the proposed method. (a) Our image-based recognition module consists of two modules: Semantic Segmentation Module (SSM) for pixel-wise general object classification, and Traversability Estimation Module (TEM) for pixel-wise traversability estimation. SSM is trained by a pseudo-label learning method. TEM is trained with the *traversability masks* generated based on the robot's experience. (b) During the robot navigation, object classes and the traversability are predicted, fused with depth data and projected in the 3D space to build a semantic 3D map. The voxels predicted as plants and with high traversability are not considered as obstacles and thus the robot is able to traverse paths covered by traversable plant parts.

Bayes' update. By treating as free spaces voxels where the probability of plant is the highest and the traversability is greater than a threshold, the robot is able to navigate through traversable plant parts covering the paths. In this work, we conduct navigation experiments in a real-world greenhouse as an example of plant-rich environments.

The contributions of this work are as follows:

1. A novel framework of scene recognition considering both general object classes and the traversability of objects for robot navigation in plant-rich environments.
2. A manual annotation-free training method for the scene recognition model.
3. A PU learning-based training for traversability estimation which requires only positive labels given to a part of traversable regions in images.
4. Applying the scene recognition model to the navigation tasks in a real greenhouse with plants partially covering the paths.

4.2 Proposed Method

Fig. 4.1 shows the overview of the proposed framework. We employ two network modules for the scene recognition: Semantic Segmentation Module (SSM) for semantic segmentation, and Traversability Estimation Module (TEM) for traversability estimation. SSM is trained by an unsupervised domain adaptation method. For the training of TEM, we generate traversability masks based on the robot's traversal experience. The network is then trained with the generated traversability masks (Fig. 4.1(a)). Predictions of the trained network are projected to the 3D space to build semantic local 3D map that indicates the object classes and traversability of the regions around the robot and the navigation is operated based on the map (Fig. 4.1(b)).

4.2.1 Traversability mask

The traversability masks are label data that indicate the image regions traversed by the robot or a human during the data acquisition phase. Using the traversability masks, we automatically label the regions of images dominated by traversable plant parts to train a network so that it can distinguish traversable and non-traversable plants based on the robot's experience.

We first acquire RGB-D data by manually operating a robot or walking inside the target greenhouse with an RGB-D sensor moving through the plants. We then build a 3D voxel map using RTAB-Map [144] from the RGB-D data. After that, by utilizing the robot's path estimated in the mapping process, the voxels traversed by the robot or the human are labeled. Here, we approximate the shape of the robot or the human with a rectangular, and from each observation point of the 3D voxel map, voxels within the rectangular are labeled as traversed. Finally, traversability masks are generated by projecting the traversability labels of the voxels on an image plane from each sensor pose on the map.

Examples of the traversability masks are shown in Fig. 4.2. One may notice that while much of traversable plant parts such as leaves and branches are labeled, there is also quite a lot of unlabeled regions of traversable plants. This is because the traversability masks are based on the limited number of traverses during the data acquisition phase. This nature of the traversability masks makes it inappropriate to simply treat them as positive and negative examples. By our training method inspired by PU learning [24] described in 4.2.2, the traversability masks are effectively exploited to distinguish traversable and non-traversable plants.

4.2.2 Network architecture

In this section, we describe a network architecture for pixel-wise semantic segmentation and traversability estimation. The overview of our network is shown in Fig. 4.1(a). The network consists of Semantic Segmentation Module (SSM) for pixel-wise object label assignment and Traversability Estimation Module (TEM) for estimating how likely that each pixel can be traversed by robots. SSM is first trained independently. TEM is then trained with intermediate features of SSM as inputs and traversability masks as training labels. The reason for not training them simultaneously is to avoid overfitting of the entire network to the traversability masks which include unlabeled traversable regions.

Our network is based on ESPNetv2 [101] and we make two modifications. First, an auxiliary segmentation branch is attached to the middle of the main network of ESPNetv2. We refer to the segmentation network including the auxiliary branch as SSM (Semantic Segmentation Module). The auxiliary branch is used to estimate pixel-wise uncertainty for the training of SSM proposed in Chapter 3. Second, a branch for estimating traversability, i.e., TEM is attached. The intermediate feature from the SSM is fed into a 3×3 convolution and the sigmoid function to produce pixel-wise probability predictions. For the complete network architecture, see Fig. A.1 in the Appendix. The

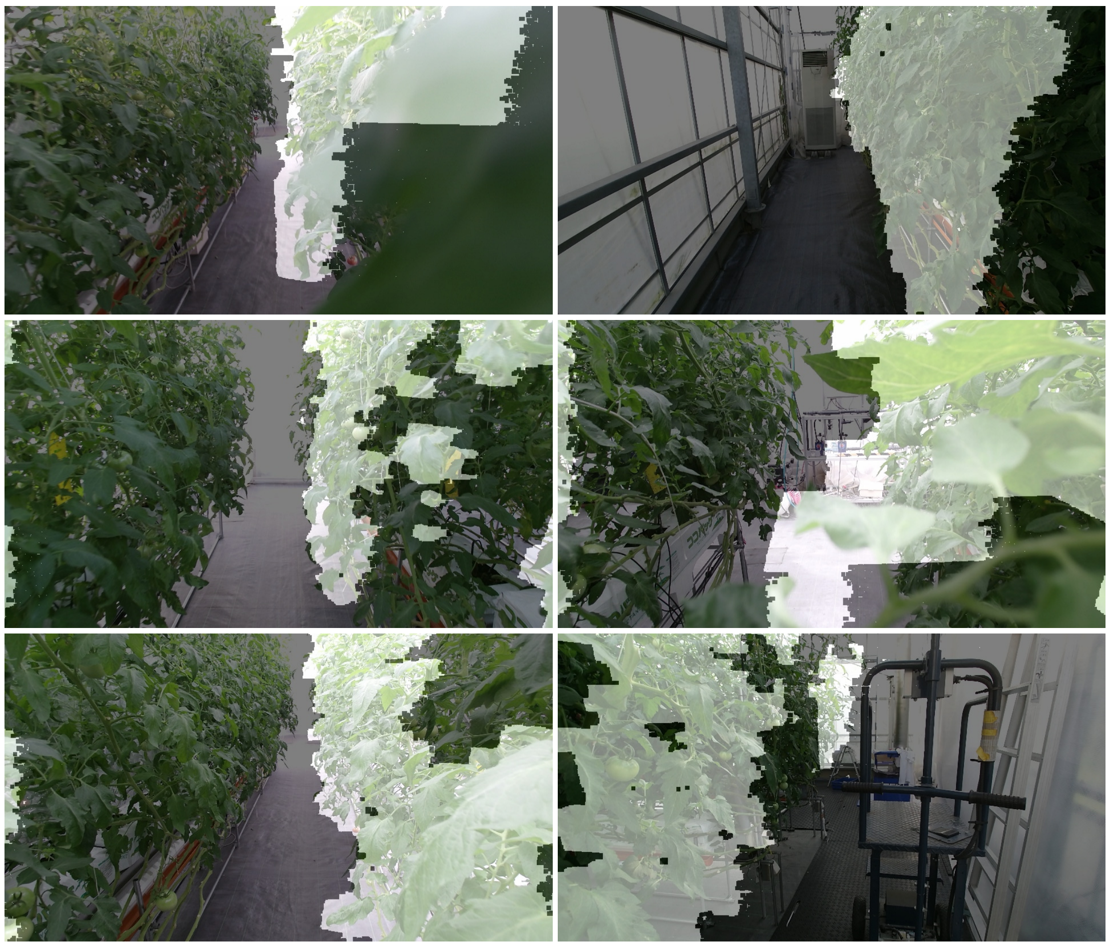


Figure 4.2: Examples of traversability masks. Regions traversed by the robot are superimposed on the corresponding camera images

additional layers do not affect the computational efficiency and still allows for prediction at sufficient speed as explained in 4.3.2.

The advantages of our architecture are three-fold. Firstly, the model structure allows for training of TEM using traversability masks via PU learning. As already mentioned, ordinary binary classification relying only on traversability masks will fail because the unlabeled regions also include traversable plant regions, and treating those unlabeled traversable regions will confuse the training. Instead, we utilize discriminative features learned through the semantic segmentation task for PU learning. By this way, we can effectively exploit the incomplete and positive-only traversability masks.

Secondly, by separating the tasks into semantic segmentation and traversability estimation, the branches for each task can be trained with the maximum possible number of data. Because the traversability masks are generated from a map, the number of available training data for TEM is less than that for SSM, which only requires RGB images. Thanks to the network structure, the SSM can fully exploit available training images for a better performance of semantic segmentation individually from the training of TEM. In fact, as shown later in 4.3.1, the SSM was trained with 6689 images while the TEM was trained with 889 images with traversability masks in our experiment. Thirdly, the two modules provide complementary information. Misclassifications of TEM can be refined by a prediction of SSM by utilizing prior knowledge that, e.g., artificial objects are not traversable. We evaluate this effect in 4.3.1.

Semantic Segmentation Module

Semantic Segmentation Module (SSM) is responsible for pixel-wise general object classification. Here we use three general object classes: plants, artificial objects, and the ground. Besides the object classes, the SSM also provides a discriminative feature on each pixel for the training of the TEM. For training SSM, we use a method described in Chapter 3.

Traversability Estimation Module

Traversability Estimation Module (TEM) estimates the probabilities that the pixels are traversable. In the training of TEM with the traversability masks, we introduce the Positive and Unlabeled (PU) learning framework. The purpose of PU learning is to model a probability function $p(y = 1|x)$, where x is an input data and $y \in \{0, 1\}$ is a binary label of the data x, in a situation where a part of the positive data ($y = 1$) is labeled as positive, and the rest is unlabeled and can be either positive or negative. Elkan and Noto [24] state that under the "selected completely at random" assumption, meaning that the labels are given completely at random, PU learning can be solved by modeling a classifier $g(x)$ such that $g(x) = p(s = 1|x)$, where s denotes a random variable indicating that data x is labeled if $s = 1$. Here, note that $y = 1$ if $s = 1$ but not vice versa. The equation can then be transformed as follows, given the "selected completely at random" assumption, i.e., $p(s = 1|y = 1, x) = p(s = 1|y = 1)$:

$$\begin{aligned} g(x) &= p(s = 1|x) \\ &= p(y = 1 \wedge s = 1|x) \\ &= p(y = 1|x)p(s = 1|y = 1, x) \\ &= p(y = 1|x)p(s = 1|y = 1). \end{aligned} \tag{4.1}$$

Therefore, the traversable probability of a pixel $p(y = 1|x)$ can be calculated as follows:

$$p(y = 1|x) = \frac{g(x)}{c}, \tag{4.2}$$

where $c = p(s = 1|y = 1)$ which denotes the conditional probability that a positive data is labeled. c is approximated by feeding the training data to the estimated $g(x)$ and average the probabilities of the labeled features, i.e.,

$$c = p(s = 1|y = 1) \approx \frac{1}{n} \sum_{x \in P} g(x), \tag{4.3}$$

where P denotes a set of all labeled pixels in the training dataset and n is the number of elements in P.

In TEM, we first model the label probability $g(x) = p(s = 1|x)$ by a convolution and a sigmoid activation. The output is then divided by c calculated by eq. (4.3) to produce $p(y = 1|x)$. The feature map from the SSM is fed in TEM. The size of the feature map is (B, C, H, W) where B denotes the batch size, C denotes the channel size, H and W

denote the height and the width, respectively. Here we consider a vector of size C as an input x for the corresponding pixel. 3×3 convolution is applied to the features, followed by a sigmoid function to scale the output in the range of $[0, 1]$. We use 3×3 convolution instead of 1×1 convolution to take information of adjacent pixels into consideration in the estimation.

Note that the "selected completely at random" assumption does not strictly hold in our task. In practice, however, we confirm that this formulation is effective to estimate pixel-wise traversability.

4.2.3 3D semantic voxel map

Using the results of the recognition explained in section 4.2.2, we build 3D semantic voxel map around the robot. At first, an RGB image acquired from the RGB-D sensor is passed to the segmentation network. The predictions from SSM and TEM are then mapped into the 3D space using the corresponding depth image. The 3D space around the robot is divided into voxels with a side of 0.1 [m], and the predictions mapped into the 3D space are assigned to a corresponding voxel that they fall into. For the object class information, a histogram of the object classes is constructed in each voxel and the class label with highest frequency within the voxel is used as an observation. For the traversability information, the traversability values of the points within a voxel are averaged and treated as an observation.

In each time step, those observations are temporally fused by Bayesian update. The observation of the object class z_t^o at time t is fused by the following equation:

$$P(l_t|z_t^o) = \eta P(z_t^o|l_{t-1})P(l_{t-1}), \tag{4.4}$$

where l_t denotes an object label of the voxel at time t, η is a normalization term and $P(l_{t-1})$ is the prior of label l_{t-1} at time $t-1$. $P(z_t^o|l)$ is the likelihood of label l with observation z_t^o. The likelihood is calculated using greenhouse images not used in the training of SSM as follows. Firstly, pseudo-labels are generated for each image. The same images are then fed in SSM and outputs of pixel-wise class label are produced. For each object class l, a histogram of predicted object classes over all pixels of the pseudo-labels with the label l. A conditional probability $P(z_o|l)$ is calculated by normalizing the histogram and used as a likelihood function. The object class with the highest probability is assigned to the voxel as its label.

Similarly, the observation of traversability is fused by the following equation:

$$P(\tau_t|z_t^\tau) = \eta P(z_t^\tau|\tau_{t-1})P(\tau_{t-1}), \tag{4.5}$$

where $\tau_t \in \{0, 1\}$ denotes an event that the voxel is traversable (1) or not traversable (0) at time t. The likelihood $P(z_t^\tau|\tau)$ is calculated using the training images with traversability masks as follows. The input images are fed in TEM and outputs of traversability values are yielded. For each label $\tau \in \{0, 1\}$ in the traversability masks, a histogram of estimated traversability values is calculated over all pixels with the traversability label τ. The histograms are then normalized to calculate a conditional probability $P(z_t^\tau|\tau)$ and it is used as a likelihood.

After the integration of observed points to the semantic voxel map, obstacle point cloud is generated by treating the voxels whose object class is "plant" and the traversability is higher than the threshold as free spaces and the others as obstacles. The location

Table 4.1: Greenhouse datasets used in the training. The type of each set is shown in the brackets

	Train	Test	Date
A	6684 (unlabeled)	33 (true trav. labels)	May 25, 2018
B	899 (w/ trav. masks)	-	July 12, 2019

of a point for a voxel is the centroid of the points that have been accumulated in the voxel over the frames. By feeding the obstacle point cloud to a conventional navigation module, the navigation in plant-rich environments is realized. For mitigating the degradation of the processing speed as the number of voxels increases, the voxels where no point has been observed for a certain number of consecutive frames are removed from the map. In the experiment below, the number of frames is set to 10.

4.3 Experiments

4.3.1 Evaluation of TEM

Used datasets

For the training of SSM by the UDA method, we used three source datasets: CamVid [139], Cityscapes [107], and Freiburg Forest [140], for generating pseudo-labels. As target data, we used 6684 unlabeled greenhouse images from Greenhouse A dataset, taken in a greenhouse growing tomatoes.

For the training of TEM, we used 899 pairs of an RGB image and the corresponding traversability mask from Greenhouse B dataset, taken in the same greenhouse as Greenhouse A on a different date. The data were collected through a human operator's control of the robot for about 20 minutes in total. During the data collection, the robot traversed each of the paths with plant rows on both sides only once. The labels are therefore imbalanced as can be seen in Fig. 4.2. Approximately 40 to 50 % of the traversable plant regions were actually labeled. Greenhouse A and B have a different appearance due to a different level of growth of the plants. For the evaluation, we manually gave true labels of traversable regions on 33 images from Greenhouse A. Overview of the greenhouse datasets is shown in Table 4.1.

Training conditions and hyperparameters

The DNN model is implemented with PyTorch [145] and all of the training is performed on one NVIDIA Quadro RTX 8000 with 48GB of memory. In the training of SSM, the training epoch is set to 200 with a fixed learning rate of 5×10^{-5}, and the batch size of 64. TEM is then trained with weights of SSM fixed. The number of epoch is set to 200 and the batch size is 64. We use cyclical learning rate scheduling [146]. The initial learning rate is 5×10^{-5} and linearly increases by a factor of 10 in 10 epochs and then decreases to the original value in 20 epochs.

Baselines for traversability estimation

As baseline methods of image-based traversability estimation, we use the following methods.

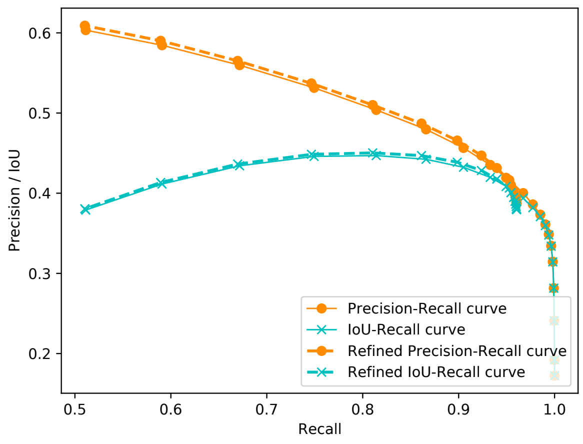


Figure 4.3: Precision-Recall curve and IoU-Recall curve of traversability estimation

Semantic segmentation We train a semantic segmentation model with "traversable plant" class as well as the classes used in the training of SSM. A model identical to SSM is used as a semantic segmentation model. For training, we used the same dataset as the one used in the training of TEM (899 RGB images with traversability masks). Pseudo-labels are generated for each image with the three classes. Labels of the pixels with the "plant" class are then changed to "traversable plant" if the value of the corresponding traversability mask is 1 (traversable). As a result, the object classes used in the training are as follows: *traversable plant, other plant, artificial object*, and *ground*. We initialize the network via pre-training with Greenhouse A dataset. A constant learning rate of 1×10^{-4} is used and an epoch size is 50.

Kim et al. Out of few studies that explicitly predict the traversability of objects such as plants, we use Kim et al. [23, 34] as another baseline. They addressed the problem of training a classifier based on a robot's experience of successes/failures of traversals, which shares the research target with ours. To evaluate the baseline method, we conducted 10-fold cross validation using the 33 test images with ground truth labels. We use those data instead of the training images with traversability masks for two reasons: 1. the baseline method assumes complete labels of traversable/non-traversable while the traversability masks are not complete and include unlabeled traversable regions. 2. the baseline method is for incremental online learning with several image frames instead of hundreds of training data. In the original work [34], ten consecutive frames of images are used for training.

Results

We evaluate the binary traversability images generated from the predicted traversability with different thresholds. Here we compare two types of binary images: *raw* and *refined*. The *raw* binary images are generated by just binarizing the predicted traversability with a threshold. The *refined* binary images are generated by additionally setting all the pixels predicted as an artificial object or the ground to non-traversable. This refinement provides the filtering effect of false positive traversability predictions even in the case of high predicted traversability, which is equivalent to the calculation in the voxel described

Table 4.2: Performance of TEM

	IoU (%)	Accuracy (%)	Precision (%)	Recall (%)
Raw	44.69	82.73	50.39	**81.41**
Refined	**45.03** (+0.34)	**83.05** (+0.32)	**51.00** (+0.61)	81.04 (-0.37)
Segmentation	2.97	82.76	45.45	3.10
Kim et al. [34]	30.95	80.42	35.87	72.12

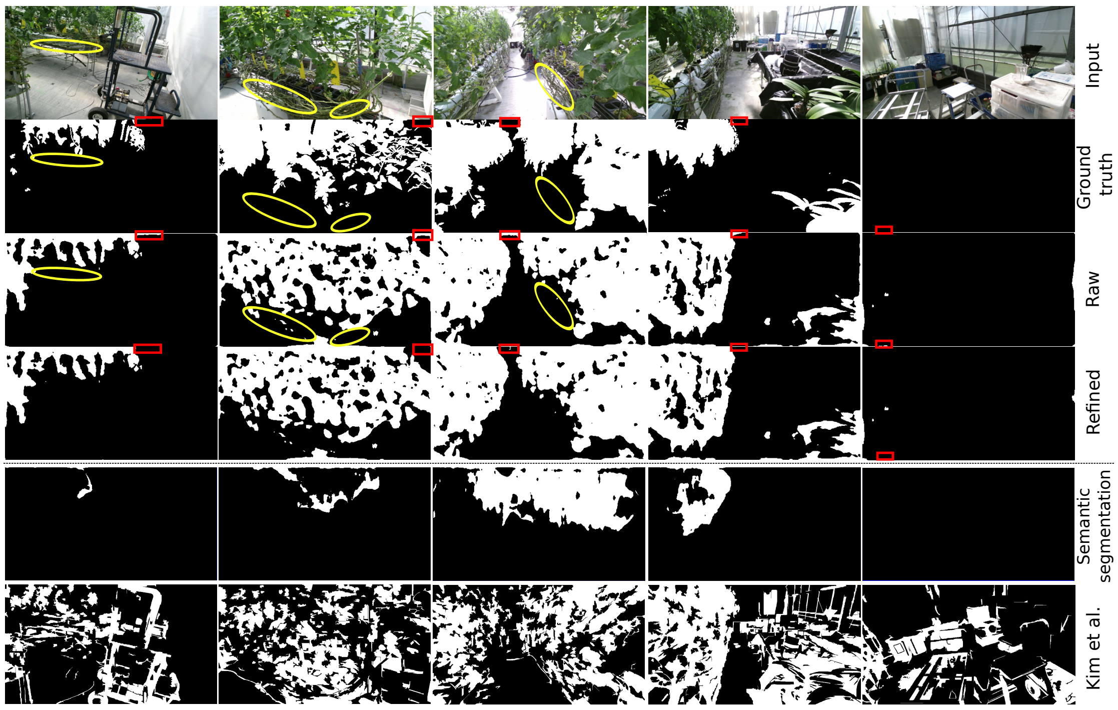


Figure 4.4: Prediction results of TEM. From the top: Camera images, Ground truth, predicted binary images with threshold 0.75, refined binary images, binary images generated from the ordinary semantic segmentation method ("traversable plant" pixels are shown in white), and binary images generated from the probability maps predicted by the baseline method [34]. Note that the traversable regions are not identical to the plant regions, but plant parts such as stems are not traversable. TEM is capable of distinguishing the traversable plants from the non-traversable ones (highlighted in yellow ellipses). The refinement by predicted object classes mainly worked as noise suppression (highlighted in red rectangles).

in 4.2.3 where the final decision of traversability is given considering both the object class and the predicted traversability. Fig. 4.3 shows a Precision-Recall curve as well as an IoU-Recall curve to analyze the performance of TEM. The best IoU was achieved when the threshold was 0.75. Therefore, we use this traversability threshold in the rest of this chapter. Table 4.2 shows the accuracy, precision, and recall of the model on the test set with the threshold of 0.75.

From Fig. 4.3 and Table 4.2, we can see high recall and relatively low precision. The refined predictions resulted in better precision and IoU. This result indicates that fusing the predictions of object classes and traversability decreased the false positive rate and thus is suitable for safe robot operation compared to only predicting the traversability.

Fig. 4.4 is the visualization of the prediction by TEM. Much of the traversable regions are classified as positive. In addition, some non-traversable plants are predicted correctly. This result shows that our model can learn the differences between traversable and non-traversable plants. In terms of the refined results, we cannot qualitatively see obvious improvements, but we confirmed that small prediction noise was suppressed. Note that these segmentation results are yielded from automatically generated traversability masks, which are incomplete and noisy. Even from such data, our PU learning-based training framework effectively learns the features of traversable objects, and provides reasonable estimation.

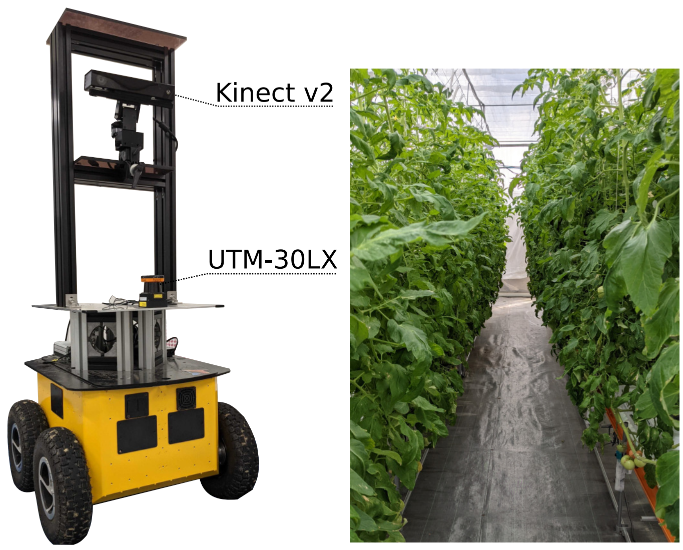
Comparison to the semantic segmentation TEM provided better performance than the segmentation-only method in all the metrics. Especially, the IoU and the recall were significantly low in the ordinary semantic segmentation. In Fig. 4.4, we can see that the semantic segmentation method could not capture the traversable plant regions well. We suppose it is because the labeling of "traversable plant" regions does not cover all the traversable plant regions as mentioned in 4.2.1 and results in inconsistent labels with traversable plant regions labeled as non-traversable plant. Such labels confuse the training. Therefore, the segmentation-only method does not get along with the sparse traversable masks. On the other hand, our proposed method enables more accurate and effective training of the model to estimate traversability of image regions.

Comparison to Kim et al. We report the average of the metrics over the 10 different partitions for the cross-validation in Table 4.2. For each data partition, the threshold of the probability is determined so that it yields the best mean IoU for the test images. As a result, our proposed method outperformed the baseline in all the metrics even though the baseline model is trained with ground truth labels while our method is trained with noisy and incomplete traversability masks. As can be seen in Fig. 4.4, although the plant regions are mostly classified correctly, other regions of artificial objects and the ground are also classified as traversable. Moreover, the traversable leaves and branches and non-traversable stem parts are not distinguished and almost equally classified as traversable.

Our method is advantageous over the baseline by Kim et al. in three aspects. First, our PU learning-based method enables the training without negative labels that are difficult to acquire. In [23], negative labels are collected using the information of the robot actually bumping into an obstacle, which is not appropriate in environments such as greenhouses. Our learning framework allows for safer and more practical way of training the scene recognition model by utilizing only positive labels acquired via the control by a human operator. Second, our model provides semantics about object classes to complement the estimation of traversability. As can be seen in Fig. 4.4, there are a lot of regions wrongly classified as traversable in the prediction of the baseline method. There is no way to correct the misclassifications in the baseline, while our method can amend them using the prediction by SSM as mentioned above. Third, our method can utilize more discriminative features thanks to the DNN model trained in the task of semantic segmentation, which led to the better performance of the proposed method over the baseline even by the raw predictions of TEM.

4.3.2 Navigation in a greenhouse

Finally, we conducted a real-world navigation experiment in a greenhouse. All the software is processed on a laptop with an Intel Core i7-6700HQ, an NVIDIA GeForce 960M, and 32 GB RAM. Revast Mercury is used as a robot platform (See Fig. 4.5(a)).



(a) Robot configuration (b) Inside the greenhouse

Figure 4.5: Experiment environment

As sensors, the robot is equipped with a Kinect v2, an RGB-D sensor, and Hokuyo UTM-30LX, a laser range finder.

An image of a path in the greenhouse is shown in Fig. 4.5(b). The length of the paths is approximately 7.5 [m]. The software for the DNN prediction is implemented on Robot Operating System (ROS) [147] with C++ and Libtorch, a C++ library for PyTorch. The frame rate of the prediction of the network is approximately 8 [fps] and the 3D mapping runs at approximately 5 [fps]. The frame rate of the original ESPNetv2 is also approximately 8 [fps] on the same computer and thus our modification to the network does not affect the computational efficiency.

As a baseline for navigation, we also test a system that considers all the voxels as obstacles. It corresponds to using only geometric information, which is a major approach in most of the current mobile robots.

Obstacle detection in a simple forward motion

As a primary experiment, we conducted an experiment of obstacle detection using the proposed 3D semantic map during the robot's act of moving forward. We assume that the path is straight and constant control signals of linear velocity are given to the robot while the robot recognizes the traversability of objects in front. The linear velocity is set to 0.1 [m/s]. An obstacle point cloud is generated from the 3D semantic voxel map by outputting only voxels other than "traversable plant" as obstacle points. When an obstacle point is detected within a range in front of the robot, the control signal is stopped.

We carried out 5 trials on the same path. The baseline navigation method consistently failed the task due to the part of plants grown out to the path. In contrast, our system was able to recognize the traversable plants and navigate through the path between the plant rows in all the trials. This result shows the ability of our scene recognition method to recognize the traversable plants covering the path. There were, however,

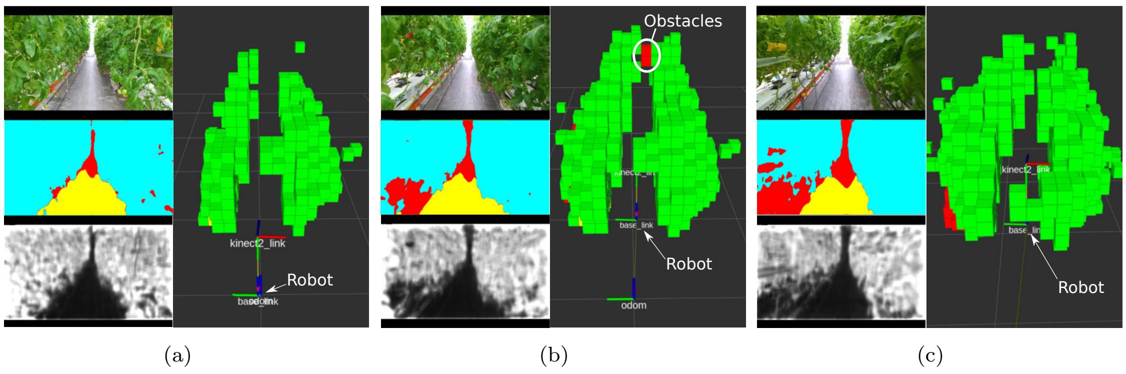


Figure 4.6: Generated 3D maps during the experiment. An input image, a predicted segmentation image, and a predicted pixel-wise traversability are shown on the left side of each figure. In the segmentation images, *blue* is the plant, *red* is the artificial object and *yellow* is the ground. In the figures of 3D map, *green* indicates traversable voxels with the plant class and the traversability higher than the threshold. The rest of the colors indicates the same object classes as in the segmentation images.

some cases where the robot stops for a while and then restarts even though there was no obstacle. This was due to misclassification of voxels in front of the robots as obstacles. The misclassification of voxels stems either from the error in the image-based semantic segmentation or from the registration error between the predicted segmentation images and the corresponding depth images, which RGB-D sensors inevitably have. Errors temporarily occurring on a few pixels are suppressed by the voxelization and the Bayes' update. When the error consistently occurs on a large area, however, the corresponding voxel can be misclassified.

Fig. 4.6 is the visualization of the semantic 3D maps generated during the experiment at different times in chronological order. There were a lot of plant parts partially covering the path. Our proposed method was able to recognize the traversable plants and to traverse them. Fig. 4.6(b) shows the case where the system wrongly recognized the regions in a far front of the robot as obstacles shown in the red voxels. The robot stopped moving in response to the obstacles. After observing a few data frames while stopping, the obstacle voxels were removed from the map and the robot resumed the navigation. We suppose the cause of this phenomenon as follows. When the misclassified regions are far from the robot, the segmentation results and the depth readings can be more noisy and it results in the misclassification of the voxels or creating voxels on wrong locations. In the case in Fig. 4.6(b), it is more likely that the voxels were spawned on wrong locations due to the depth noise. As the robot approaches the voxels, both the segmentation and the depth values become more accurate and the voxels are updated with those observations. As a result, in the case in Fig. 4.6(c), the misclassified voxels were deleted from the voxel map because points in those voxels were not observed for the pre-defined number of frames (10 frames) when the robot was stopping near them.

Integration with move_base

We also conducted an experiment of applying our 3D semantic voxel map to navigation using *move_base*[1], a de facto standard navigation software in ROS [147].

[1]`http://wiki.ros.org/move_base`

Table 4.3: Result of the navigation experiment using *move_base*

Trials	Traversed (intervened)	Stuck
12	8 (5)	4

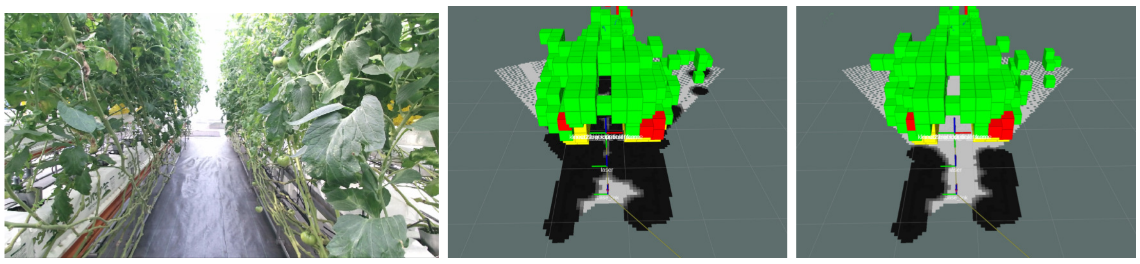

(a) Input image (b) Baseline for navigation (c) Ours

Figure 4.7: Costmaps in the baseline and the proposed method. (a) An input image. (b) Semantic 3D voxel map and the costmap of the baseline. All the voxels are considered as obstacles and thus the path is recognized as blocked. (c) Semantic 3D voxel map and the costmap generated by the proposed method. The voxels of traversable plant are not considered as obstacles and the path is recognized as traversable.

The obstacle point cloud is fed in move_base as a sensor reading. In addition to the obstacle point cloud, a 2D laser range finder (LRF) is used to complement the blind spot of the RGB-D sensor and stabilize the navigation. Because of this purpose, the obstacle range of the LRF is limited to 1.0 [m]. The configuration of the LRF is the same in the baseline method. Sub-goals are given manually via Rviz, an interactive GUI tool for visualization.

Here, we define criteria for judging the success or failure of the trials. In all the trials, we observed that the accuracy of the SSM consistently degrades about 2.5 [m] before the end of the paths and the robot fails to detect the traversable path (We discuss this problem in 4.3.3). We therefore consider a trial in which the robot reached about 2.5 [m] before the end of the path as "traversed". When the robot stops due to misclassification, we allow a human's intervention by manually restarting move_base to clear the costmap at most once. Trials where the robot gets stuck even after the human intervention are marked as "stuck".

The baseline navigation method consistently failed to find a path on three paths before entering them. An example of a costmap generated with the baseline method is shown in Fig. 4.7(b). We carried out 12 trials of navigation on the three paths (4 each) with the proposed method. The result is shown in Table 4.3. The robot with the proposed method succeeded to traverse through the path partially covered by plants in 8 trials, 5 of which involved a human's intervention. The costmaps generated with the proposed method are compared in Fig. 4.7(c). From these results, we can see that our proposed method enables the robot to recognize the traversable plant parts, which would otherwise be recognized as obstacles, and traverse through them. However, misclassifications affected the robot's motion as can be seen from the number of required interventions and failures.

4.3.3 Discussion

1. **Cause of the failures of navigation**

   We showed that our method enabled navigation through the paths covered by traversable plant parts in the greenhouse, while the baseline method constantly failed.

   In all the trials, however, the robot with our proposed method consistently failed to navigate about 2.5 [m] before the end of the paths. In addition, in some of the successful trials, it required a human's intervention due to misclassifications. On the other hand, in the experiment in 4.3.2, all the 5 trials were successful without any interventions.

   We suppose that the difference of the degrees of the success rates stems from the difference of the density of leaves. In such a case, majority of the regions are dominated by a single class of "plant" and a few misclassifications are filtered out in the probability calculation in each voxel. In the experiment in 4.3.2, however, leaves were sparse and artificial objects behind the plant rows were also visible through the leaves. This was especially obvious near the end of the path. Since the SSM is trained with pseudo-labels, which are generated by merging the outputs from multiple pre-trained models, the model lacks pixel-level accuracy. This is the current limitation of our method.

   For more reliable navigation, the accuracy of the recognition model needs to be improved. In addition, to deal with the prediction noise in DNN models, a better filtering and/or refinement methods should be implemented.

2. **Adaptability of the proposed system**

   The success of the training of our scene recognition model depends on the UDA of semantic segmentation, since TEM takes the intermediate features of the semantic segmentation network (SSM) for estimation. In terms of the segmentation tasks in greenhouses, we confirmed in Chapter 3 that our UDA method is applicable to multiple greenhouses and multiple seasons. The proposed method will thus also be applicable to various greenhouse environments. When applying the proposed system in a new environment, in general, the segmentation model will need to be newly trained to adapt to the environment. Once the segmentation model is trained, TEM can be trained in the same manner as described in this chapter. Note that our proposed model and the training method provide advantages of the training with a little images and incomplete positive labels, which lead to a minimum burden in deployment.

3. **Influence of the sensor setting**

   In this work, we employed a Kinect v2 for simplicity of implementation of the sensor system to acquire calibrated RGB and depth images. Although it is sensitive to disturbance from strong light, we did not observe any severe effect of the sunlight on the depth readings. in our experiments. We suppose it is because the sunlight was cut by the translucent roof and walls of the greenhouse and the leaves, and thus the sensor was able to work in moderate lighting conditions. When we apply the system to outdoor environments, however, the choice of sensors will influence the accuracy and reliability of the system. Note that our proposed method is compatible with any sensors that can provide registered RGB images and depth readings and the sensors can be selected based on the lighting condition of the target environment. Applicable sensor settings include a stereo sensor, and well-calibrated 3D LiDAR and a monocular camera.

In terms of sensor calibration, Kinect v2 was used out-of-the-box in our experiments. In multi-sensor setting, it will indeed require a precise calibration, although the Bayes' update and the voxelization process can to some extent deal with a little noise due to the misalignment of the sensors.

4.4 Summary

In this chapter, we described a method of estimating the traversability of plants covering a path and navigating through them in plant-rich environments for mobile robots. We proposed the following novel methods. 1. A scene recognition model that estimates general object classes and the traversability of the objects. 2. A manual annotation-free training of the model using an unsupervised domain adaptation method for the semantic segmentation module (SSM) and the information of the robot's traversals, named *Traverasability masks* for the traversability estimation module (TEM). 3. A PU learning-based training method to effectively train TEM with the traversability masks, which include unlabeled traversable regions. Our proposed method enables easy and safe deployment of mobile robots with the capability of recognizing traversable plants with a minimum burden. In the comparative analysis, we confirmed that the proposed method is able to estimate the traversability of image regions more accurately than a conventional semantic segmentation and an existing work of image-based traversability estimation. Moreover, we applied the scene recognition model to the navigation tasks in a real greenhouse with plants covering the paths and confirmed its ability to recognize traversable plants and navigate through them.

Chapter 5

Online Refinement of the Scene Recognition Model by Observing Human's Interaction with the Environment

5.1 Introduction

Although we demonstrated the effectiveness of the scene recognition method through navigation experiments in a plant-rich greenhouse in the previous chapter, misclassifications remain to be a critical problem. Fig. 5.1 shows an example of a case where some voxels of traversable plants are misclassified as other obstacles and the path is considered blocked. When such a misclassification often occurs, the robot easily gets stuck.

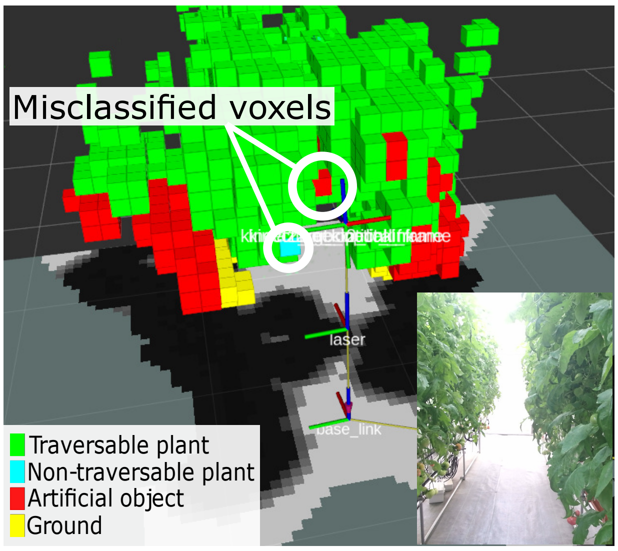


Figure 5.1: Misclassification in the conventional system. Since voxels other than *traversable plant* class are treated as obstacles in our system, the path is considered as blocked when voxels of traversable plants on the path are misclassified, which obstructs the robot.

Such misclassifications are inevitable in scene recognition methods including DNNs. A method to deal with them on the fly is, therefore, needed for practical uses of such a navigation system.

To this end, we develop a method for online refinement of the recognition model exploiting a human's interaction with the misclassified traversable plant regions during the actual operation of mobile robots. Ideally, the robot should indicate possibly misclassified regions to the human with some interfaces to acquire data efficiently. In this work, however, we consider a situation where the human randomly touches the plant parts. We propose a framework for refining the semantic segmentation model making use of the information acquired through such interaction. The proposed method will be a building block of more advanced system for traversable plant recognition with a function for online refinement of the recognition model.

In the proposed method, a human first touches regions of plants in the 3D space to provide the robot with information to correct the recognition of plants. The robot observes the interaction using a human pose detector such as OpenPose [148] to identify plant regions indicated by the human. Masks of the interacted regions in images are generated for the model refinement. The scene recognition model is then refined using the mask so that next time the robot encounters similar scenes, the robot recognizes the scenes correctly. More specifically, we introduce a few-shot learning method based on weight imprinting [135]. Since the masks of interacted regions are generated using depth images from an RGB-D sensor, we add a modification to the process of weight imprinting to deal with noise included in the masks.

The contributions of this work are as follows:

1. A novel framework of online refinement of a scene recognition model exploiting data acquired during the robot's operation using a human's interaction with the environment.

2. A novel weight imprinting-based few-shot segmentation that is robust to noise in the masks.

5.2 Proposed Method

Our aim is to refine the scene recognition model on the fly with data collected during the robot's operation. General fine-tuning is not suitable due to limitations of computational resources and of the variety of image data acquired during the operation. We, therefore, adopt a few-shot segmentation method based on weight imprinting [135], which does not require fine-tuning with costly backpropagation.

5.2.1 Preliminary: Weight imprinting

Weight imprinting [135] was first proposed as a method for few-shot image classification. Given an embedding extractor $\phi : \mathbb{R}^N \to \mathbb{R}^D$ which maps an input $x \in \mathbb{R}^N$ to a D-dimensional vector $\phi(x) \in \mathbb{R}^D$, and a softmax classifier $f : \mathbb{R}^D \to \mathbb{R}^C$ which produces probability values for C classes, a probability for class j is given as follows:

$$f_j(\phi(x)) = \frac{\exp\left(\mathbf{W}_j^T \phi(x)\right)}{\sum_c^C \exp\left(w_c^T \phi(x)\right)}, \tag{5.1}$$

where $\mathbf{W}_j$ denotes the weight vector for class j. Here, both the embedding $\phi(x)$ and the weight vector w_j are assumed to be L2-normalized. Then, the main computation in eq. (5.1) is taking the inner product, or equivalently the cosine similarity between $\phi(x)$ and w_j. Viewing each weight as a template vector of the corresponding class, the authors argued that the prediction is equivalent to finding the template closest to the embedding $\phi(x)$ based on the cosine similarity in the embedding space. Based on the observation, they proposed to use the embedding for input data of a novel class as the classification weight of the class.

Siam et al. [136] applied weight imprinting to few-shot semantic segmentation. Given the intermediate feature map $\mathbf{x}_i \in \mathbb{R}^{H \times W \times D}$, with the height H and the width W, and a corresponding binary mask $M_i \in \{0,1\}^{H \times W}$, they aggregate embeddings over a given mask via simple averaging followed by L2 normalization (masked average pooling: MAP):

$$\begin{aligned} \mathbf{x}_{MAP} &= \frac{\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} M^{(h,w)}\mathbf{x}_i^{(h,w)}}{\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} M^{(h,w)}} \\ \hat{\mathbf{x}}_{MAP} &= \frac{\mathbf{x}_{MAP}}{\|\mathbf{x}_{MAP}\|_2}, \end{aligned} \tag{5.2}$$

where (h,w) denotes the coordinate of the image pixel. The normalized average embedding $\hat{\mathbf{x}}_{MAP}$ is used as classification weights of the novel class. In [136], they proposed to apply it to multiple layers to exploit multi-resolution information, which is, in general, considered important for semantic segmentation. In our method, however, we apply weight imprinting only on the last classification layer for simplicity of implementation.

5.2.2 Problem setting

We assume that the scene recognition model $f(I)$ is pre-trained with a large amount of data before deploying the robot in a target environment as described in Chapter 4 with C classes. We then assume to acquire a few pairs of an RGB image and a corresponding binary mask $S = \{(I_i, M_i)\}_{i=1}^{N}$, where N denotes the number of the pairs. The value 1 in the binary mask $M_i \in \{0,1\}^{H \times W}$ indicates a pixel that should be learned as *plant* and the mask is acquired through interaction by the human with the environment during the robot operation. The process of generating the mask is described in the next subsection.

Our purpose is to update the model on the fly with the image-mask pairs S so that the model predicts the plant parts more accurately. As mentioned above, we formulate the problem of the online learning as a few-shot learning problem where the objects of the masked regions are learned as a novel $C + 1$th class unseen in the pre-training.

5.2.3 Data collection

For few-shot learning, a pair of an input image I and a corresponding mask M is generated based on the human's interaction. The interaction mask indicates image regions that should be classified as *plant*.

Labeling plant regions via a human's interaction

First, a human interacts with the regions of plant parts as shown in Fig. 5.2. A robot observes the human's interaction using an RGB-D camera. The human joints are recognized by OpenPose [148]. The coordinate of the right hand is projected into the 3D space

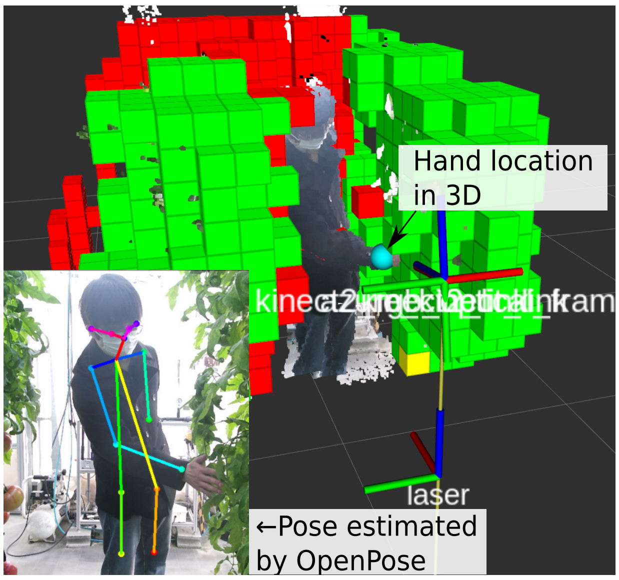


Figure 5.2: A human interacting with the plant parts. The location of the hand is estimated on the image by a human pose estimator like OpenPose [148], and projected to the 3D space using the depth from the RGB-D sensor. After that, voxels that are touched by the human are estimated.

using the depth. Voxels with which the human interacted are identified by searching for voxels within a sphere with a radius of 5 [cm] centered at the projected hand coordinate.

Generating a training mask

After the map labeling, a mask M for training is generated by the process as follows.

1. Get a pair of RGB and depth images.

2. For each pixel of the depth image, project it in the 3D space and search for a voxel that the pixel belongs to.

3. Label the pixel as 1 if the corresponding voxel has been interacted, and otherwise 0.

4. Repeat i) to iii) for five consecutive frames and take pixel-wise OR of them to deal with the depth noise. We call the resulting mask an *interaction mask* denoted as M'.

5. Predict object labels on the RGB image I using the network and set pixels of M' to 0 if the predicted object label of the corresponding pixels are other than *plant* (see Fig. 5.3).

We call the resulting mask M a *training mask*. Here we assume that the interacted regions in M' are dominated by plants. The training mask M, therefore, indicates false negative plant regions.

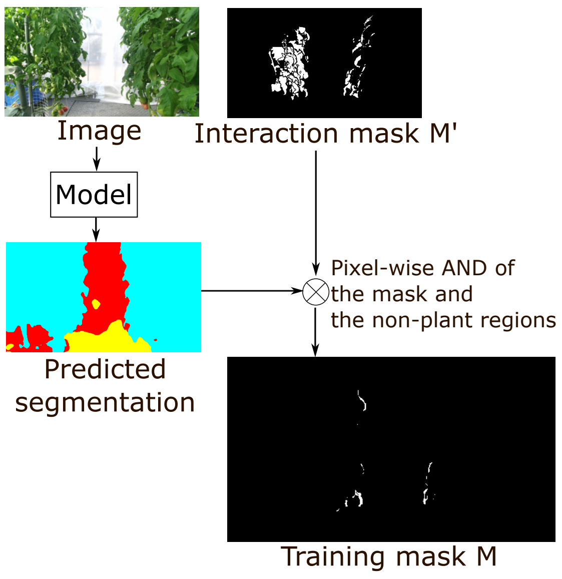


Figure 5.3: The process of generating a training mask from a pair of a camera image and an interaction mask. The interaction mask indicates the plant regions touched by the human operator. Since the purpose here is to estimate the plant regions that are misclassified as other object classes, we take pixel-wise AND operation of the interaction mask and the non-plant regions.

5.2.4 Network architecture and the loss function

Based on the architecture described in Chapter 4, we make two modifications on the network architecture for semantic segmentation and the loss function to apply weight imprinting-based few-shot learning,

First, we apply L2 normalization on the weight vectors of the final classification layer $\{\mathbf{W}_j\}_{j=1}^{C}$, which is a 1×1 convolution, and the features $\{\mathbf{x}_i^{(h,w)}\}_{i=1}^{N}$, which is a feature vector in pixel (h, w) of an intermediate feature map yielded by forwarding an image I_i through the model f. The score of the features for each class is then evaluated as a cosine similarity between the features and the weight vector of the class [135].

Second, to formulate the softmax loss on the normalized data in a more natural way and also to encourage the model to learn discriminative feature representations, we introduce the Additive Angular Margin Loss (ArcFace) [149] defined as follows:

$$L_{arc} = -\sum_{i=1}^{N} \log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1,j\neq y_i}^{C} e^{s\cos\theta_j}}, \tag{5.3}$$

where $\cos\theta_j = \mathbf{W}_j^T\mathbf{x}_i$, m denotes an angular margin parameter, y_i denotes an object label of $\mathbf{x}_i$, and s denotes a scaling factor that controls smoothness of the predicted probability distribution. By setting the angular margin m for the score of the correct class, the network is trained so that the intra-class distances of the feature distributions become large and thus discriminative features are learned.

Although we adopt the architecture described in Chapter 4 in this work, note that the proposed method is not dependent on a specific network architecture.

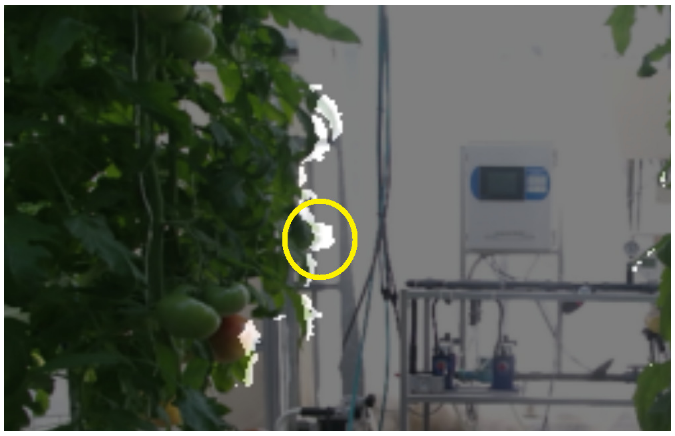


Figure 5.4: Training mask with error due to noise in the depth sensor. Although the assumption here is that the mask indicates the region of plants, the mask includes non-plant pixels due to the sensor noise and registration error. This may lead to poor performance of weight imprinting with the conventional masked average pooling (MAP).

5.2.5 Online learning by the robust weight imprinting

For online learning of segmentation, we adopt a few-shot learning based on weight imprinting [135, 136]. Suppose we have N pairs of an input image and a training mask $\{(I_i, M_i)\}_{i=1}^{N}$ and an intermediate feature map $\mathbf{x}_i$ is yielded by passing the image I_i through the segmentation network.

Here, we point out a problem of the training mask generated by the procedure described in the previous subsection. Because of the depth noise and registration error of the RGB-D sensor, the training masks include regions set as 1 and not part of the plant regions. An example of erroneous mask is shown in Fig. 5.4. MAP described in Sec. 5.2.1 is prone to those outliers since features are equally averaged, In our method, we replace it with weighted averaging based on the distance from the center of the feature distribution within the mask, which we call *robust average pooling (RAP)*:

$$\mathbf{x}_{RAP} = \frac{\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} v_i^{(h,w)} M^{(h,w)} \mathbf{x}_i^{(h,w)}}{\sum_{i=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} M^{(h,w)}}, \tag{5.4}$$

where $v^{(h,w)}$ denotes the weight on the corresponding feature $\mathbf{x}^{(h,w)}$ calculated as follows:

$$v_i^{(h,w)} = \begin{cases} \mathbf{x}_i^{(h,w)} \cdot \mathbf{x}_{MAP} & \text{if } \mathbf{x}_i^{(h,w)} \cdot \mathbf{x}_{MAP} \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

The averaged feature is then L2-normalized to form a convolution weight vector for the novel class:

$$\hat{\mathbf{x}}_{RAP} = \frac{\mathbf{x}_{RAP}}{\|\mathbf{x}_{RAP}\|_2}. \tag{5.6}$$

We directly use $\hat{\mathbf{x}}_{RAP}$ as a weight vector of the classification layer for the new class representing false negative plant regions in the model before the training.

5.3 Experiments

5.3.1 Experimental setup

The DNN model is implemented with PyTorch [145]. The pre-training and inference are performed on one NVIDIA GeForce GTX 1080Ti with 11GB of memory. Before the online learning, we pre-train the scene recognition model in a method described in Chapter 4, with modification of the model and the loss function described in Sec. 5.2.4. The angular margin m is set to 0.1.

We use five pairs of an image and a mask taken in different scenes for few-shot learning. For testing, we exploit 15 manually labeled images of scenes where influence of misclassification can be significant, such as the end of the plant rows in the experiment in Chapter 4.

5.3.2 Baselines

As baselines, we adopt weight imprinting with the ordinary masked average pooling and also a fine-tuning approach via model distillation. In the former, the averaged feature over the mask (eq. (5.2)) is used to calculate the classification weight instead of eq. (5.4).

In model distillation, we adopt output-level distillation described in [150]. We set a weight for distillation loss to 0.5 and temperature parameter T for the softmax function to 1.0. The entire network is optimized. We utilize the full mask of interacted regions since the task is fine-tuning with the existing plant class, instead of learning a novel class. The batch size is set to 5 and a constant learning rate of 1×10^{-4} is used. The model is trained for 15 epochs and we report the results on the epoch with the best mean IoU.

We hereafter denote the proposed robust weight imprinting as WI-RAP, the weight imprinting with the ordinary masked average pooling as WI-MAP, and the model distillation method as MD.

5.3.3 Online model refinement

Table 5.1 shows the comparison of IoU before and after the online learning as well as the results of model distillation. While WI-MAP resulted in degrading the performance, the proposed robust pooling led to better IoU. This qualitatively shows the advantage of our method. MD resulted in better mean IoU and per-class IoU on the *artificial object* and *ground* classes than the proposed method. The proposed method, however, resulted in better per-class IoU on the *plant* class than MD. Since our main purpose is to improve the accuracy on plant regions, this result shows better suitability of the proposed method to the online refinement.

We further look into the precision and the recall metrics. Table 5.2 shows the precision and the recall of each method. Our primary purpose is to refine the false negative plant regions which is relevant to the recall metric. The both few-shot learning approaches resulted in better recall of plant regions. The improvement of the recall, however, came at a cost of degradation of the precision. Compared to WI-MAP, the proposed robust imprinting, WI-RAP, yielded a comparative gain of recall (-0.92 from WI-MAP) with less degradation of precision (+2.44 from WI-MAP).

Table 5.1: Per-class and mean IoU before and after the training

	Plant	Artificial obj.	Ground	mIoU
Before	80.89	84.06	60.04	75.00
MD	80.88	**85.90**	**63.23**	**76.67**
WI-MAP	79.71	85.04	60.04	74.93
WI-RAP (proposed)	**81.19**	<u>85.49</u>	<u>60.05</u>	<u>75.57</u>

Bold denotes the best and <u>underline</u> denotes the second best results.

Table 5.2: Recall and precision before and after the training

	Plant	Artificial obj.	Ground
Before	89.02 / **89.86**	**92.28** / 90.41	<u>93.70</u> / <u>62.57</u>
MD	92.21 / <u>86.80</u>	<u>92.03</u> / 92.81	**94.63** / **65.58**
WI-MAP	**93.77** / 84.16	89.72 / **94.23**	<u>93.70</u> / <u>62.57</u>
WI-RAP (proposed)	<u>92.85</u> / 86.60	91.04 / <u>93.34</u>	<u>93.70</u> / <u>62.57</u>

The results are shown in the order of recall / precision.
Bold denotes the best and <u>underline</u> denotes the second best results.

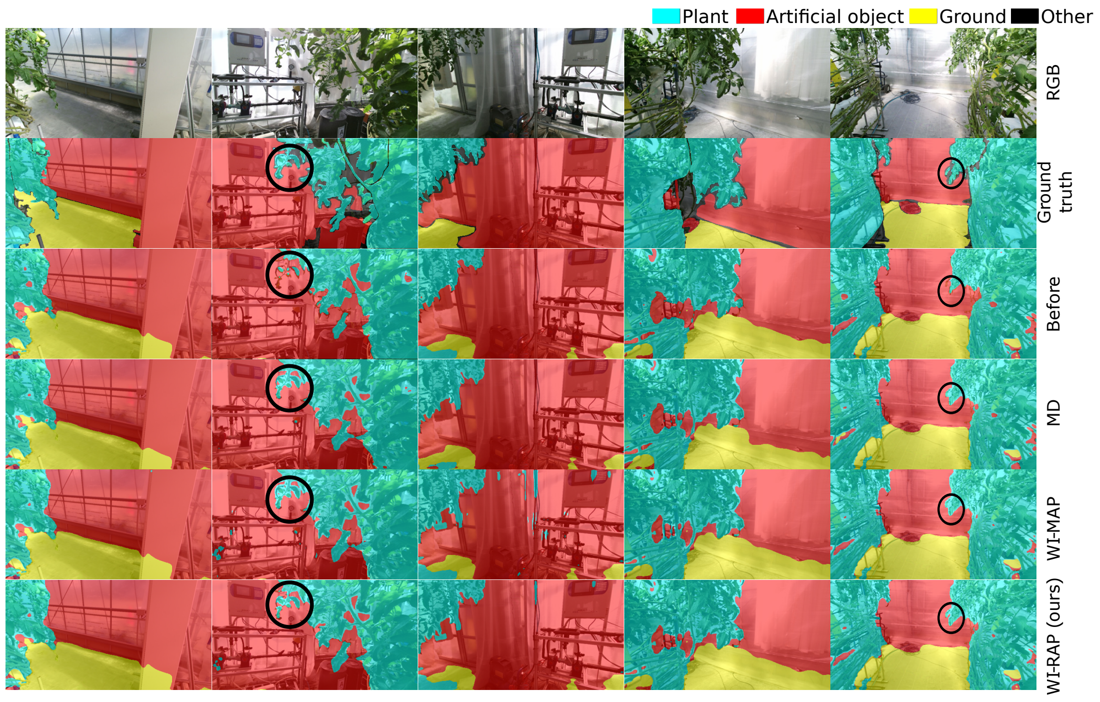


Figure 5.5: Qualitative evaluation of the online learning. After the training, the shapes of plant parts are better captured especially in the regions in the black circles. Compared to WI-MAP, there are less false positive predictions as *plant* class in the predictions of the proposed WI-RAP. MD provided similar or better predictions. The proposed method, however, resulted in similar results with only forward pass unlike MD which requires backpropagation.

Fig. 5.5 shows the qualitative results. After the online training by the proposed

Table 5.3: Mean IoU depending on different angular margin m

m	0.0	0.1	0.2	0.3	0.4	0.5
Before	74.89	75.00	74.46	72.77	74.25	66.61
WI-MAP	41.47	74.93	68.29	72.97	74.55	66.99
WI-RAP	46.80	75.57	74.96	72.76	74.09	67.00

method, false negative plant regions are corrected to the *plant* class. Especially, the regions in black circles show obvious improvement of segmentation. In those regions, detailed shapes of the leaves and the branches are better captured compared to the segmentation before training. Compared to WI-MAP, there are less false positives of prediction as *plant* in the predictions of the proposed WI-RAP. This is because while WI-MAP aggressively involves outliers in feature aggregation, WI-RAP robustly calculate the imprinted weights by the weighted averaging.

One may notice that MD resulted in better performance. It, however, required around 20 [sec] of training (excluding in-training validation) and 5 [GB] of GPU memory consumption. Although the batch size of 1 decreased the memory consumption to about 1.5 [GB], the results were worse. In contrast, the proposed method took 0.38 [sec] and consumed approx. 700 [MB] because it requires only one forward calculation for each training image. It is, therefore, more suitable to on-the-fly model refinement on modern onboard computers with GPU-acceleration such as NVIDIA Jetson.

5.3.4 Parameter evaluation

Next, we evaluate the effect of the angular margin m in the ArcFace loss. Table 5.3 shows the relationship between m used in model pre-training and the performance of few-shot learning. When $m = 0.0$, i.e., no angular margin constraint is imposed, few-shot learning resulted in significant degradation of mean IoU. This may be because when the features are learned without the angle margin, inter-class distances of the features are not encouraged to be larger, and thus the learned features are not discriminative enough with large overlaps. It will lead to confusion of features to be imprinted and those not. In contrast, when $m = 0.1$, the proposed method resulted in improving the performance, presumably thanks to better feature representations. The effectiveness of the weight imprinting, however, did not hold with all the parameters. When $m = 0.3, 0.4$, the weight imprinting resulted in slight degradation. The condition of m for effective imprinting is not clear yet. We leave further analysis of the parameter setting and the performance for future work.

5.4 Summary

We proposed a novel framework of online model refinement to deal with misclassification which may lead the robot to be stuck during navigation. We introduced a few-shot segmentation method based on weight imprinting, which allows for model refinement on the fly. Masks for the few-shot training are generated through observation of a human operator interacting with plant parts in the environment. To mitigate the effect of inaccurate masks due to depth noise, we proposed robust weight imprinting and showed that the proposed method outperformed the ordinary weight imprinting with masked average pooling in terms of IoU.

Chapter 6

Conclusions and Discussion

6.1 Conclusions

In this thesis, we described a scene recognition framework for navigation of mobile robots in plant-rich environments. The proposed scene recognition system explicitly considers traversable plant parts covering the paths, such as plant foliage.

For the scene recognition, we employed an image-based DNN with two decoder branches, i.e. Semantic Segmentation Module (SSM) for estimating general object classes, and Traversability Estimation Module (TEM) for estimating class-agnostic traversability. The SSM is trained utilizing multiple publicly available rich image datasets with pixel-wise labels. The proposed pseudo-label generation method effectively transfers knowledge from the source datasets, which are not necessarily relevant to the target dataset. The TEM is trained based on the robot's experience of traversals during data acquisition. The TEM is trained on the intermediate features from the trained SSM via PU learning framework exploiting label images where only part of traversable regions are labeled. The entire network can thus be trained without any manual annotation on the target images. In addition, additional sensors for data collection are also not required, unlike some recent work [22, 21]. We conducted navigation experiment in a real-world greenhouse and showed that the proposed system was able to navigate through the path between plant rows recognizing traversable plants growing out to the path.

We also proposed an online model refinement method to correct the model's misclassifications. Labels of image regions that should be recognized as *plant* are acquired by observing a human operator's interaction with the regions in the environment. For network training, we employ a few-shot learning method for semantic segmentation. Specifically, we introduce a method inspired by Siam et al. [136], which is an extension of *weight imprinting* by Brown et al. [135]. To mitigate the influence of noise in the labels acquired through the human's interaction, we proposed Robust Average Pooling (RAP) for aggregating features within a mask while suppressing contributions of outliers. As a result, the proposed method realized refinement of misclassified plant regions, while a conventional counterpart [136] struggled with the noisy labels.

6.2 Proposals for a system design and applications

Here, we elaborate on the possible navigation systems and applications based on our work.

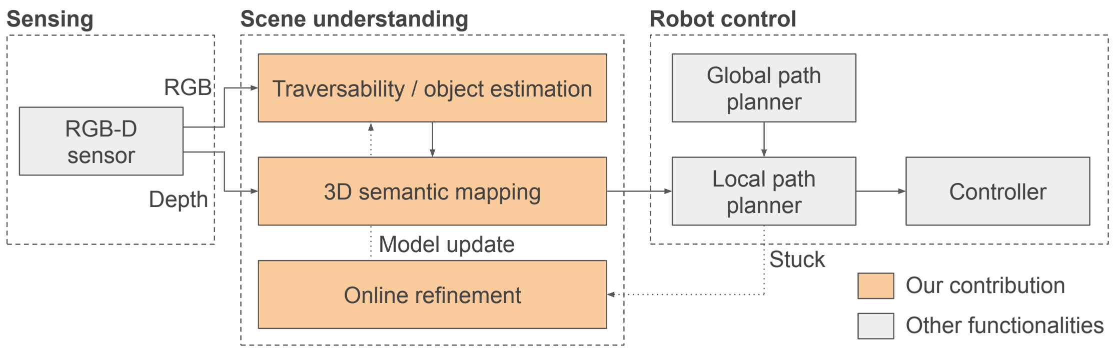


Figure 6.1: System diagram of the proposed navigation system. Our work was focused on the scene understanding functions.

6.2.1 The proposed navigation system

Fig. 6.1 shows the system diagram of the proposed system. The system consists of three main functionalities, namely *sensing*, *scene understanding*, and *robot control*. Our work mainly focused on the scene understanding functionality.

We use an RGB-D sensor for sensing. RGB images taken from the sensor is processed by the scene recognition model to estimate the pixel-wise traversability and object classes. The prediction results are then passed to the 3D semantic voxel mapping module. Once the 3D map is built, it is used for local path planning considering the traversability of the voxels, followed by the controller. When a feasible path cannot be planned by the local planner, the system triggers the online refinement function. If the cause of stuck is due to misclassification of traversable plant, a human operator indicate the part of traversable plant to the robot and the scene recognition model is refined. After that, the robot resumes the operation with the updated model.

By this framework, the robot is able to navigate in plant-rich environments such as greenhouses and forests by explicitly estimating the class and traversability of the objects and moving through traversable plants. Since the proposed scene recognition model is trained without manual annotation and just by giving unlabeled images of the target environments collected by a manually controlled robot, it is easy to deploy the system to a new environment. Thanks to the unique online refinement function, the system can deal with misclassification on the fly, instead of offline. We believe this feature is practically important for mobile robots which must continue their operation as long as possible.

In our work, we developed the traversability and object class estimation, and 3D semantic mapping as described in Chapter 3 and Chapter 4, as well as the online refinement described in Chapter 5. The robot control functionality remains as future work and discussed further in Sec 6.3.

6.2.2 Applications

Agricultural mobile robots

Agricultural mobile robots, especially those that work in confined greenhouses where GNSS cannot be used and plants grows out to the paths, are a possible application of the proposed system. In such environments, we can assume a prior map of the entire environment since their layout is known. Nevertheless, traversability estimation is necessary because regions which are mapped as free spaces are possibly covered by plant part. In addition, there can be both real obstacles and traversable objects on the paths, and it is necessary to distinguish them.

The system is suitable to tasks where human and a robot collaboratively work since it involves human in the loop. One example is harvesting, where farmers harvest crops and the robot autonomously helps them carry the crops.

Mobile Laser Scanning in forests

Mobile Laser Scanning (MLS) is gaining attention as a method to measure environments in the field of remote sensing. It enables easier measurement than Terrestrial Laser Scanning (TLS), which requires manually setting a laser scanner on multiple scanning points, and provides denser and more precise point cloud than Aerial Laser Scanning (ALS), which measures from high above the tree canopy. In current MLS systems, a robot equipped with laser scanners is manually controlled by a human operator. Automating this task will be beneficial to reduce the burden on the human, and to accelerate the measurement tasks.

Compared to navigation in greenhouses, MLS in wild forests is more challenging because there is no prior map and the robot needs to explore the environment. Simultaneous Localization and Mapping (SLAM) and view point planning are relevant techniques in such tasks. Together with them, the proposed framework allows for mapping the environment and local path planning to get to a next view point planned by the view point planner considering traversability of the plants.

6.3 Limitations and future research directions

Despite the possibility of the proposed system described in Sec. 6.2, there are still challenges in real-world applications. Here, we summarize some limitations of the proposed framework, and also discuss future research directions.

Analysis of robustness When applying it to the real-world tasks, robustness of the scene recognition to changes of environmental conditions can be a critical problem. For example, how often misclassification that may obstruct the robot operation occurs is an important indicator of the robustness of our system. If it is very often, the system will not be useful in the real world even with the online refinement. At this moment, we have not been able to study this aspect. To pave a path to reliable applications, it is crucial to analyze it in detail. Specifically, the robustness should be analyzed in an actual navigation task in a long period of time.

Integration with waypoints / control estimation In this work, we have focused on the scene recognition system and its training methods. The real-world navigation experiments were limited to a relatively structured greenhouse, and fully autonomous navigation considering the 3D semantic map is not implemented. In future work, the proposed scene recognition framework need to be integrated with estimation of optimal waypoints or controls to realize truly automatic navigation. Compared to data-driven methods like [20], traditional controllers like Model Predictive Controller used in [21] considering the traversability map as a cost seem to be a more promising choice.

Navigation in more unstructured environments We conducted real-world navigation in a greenhouse, which has paths with plant leaves growing out, but is relatively structured. To further examine the applicability of our method, we should conduct navigation on more unstructured environments with arbitrary shape and width of the paths.

Navigation in unstructured environments does not only require traversable plant detection, but also terrain analysis, which has been widely studied [31, 32, 33]. One way to implement a navigation system considering both traversable plants and terrain traversability is to simply integrate our method with existing terrain analysis methods. We could also jointly estimate terrain traversability and plant-induced traversability by the proposed scene recognition model, similarly to [22, 21].

Extension of PU learning-based traversability learning A recently popular approach to scene recognition and navigation in unstructured environments with vegetation is the DNN-based methods utilizing a robot's experience of traversals [23, 20, 22, 21]. As we mentioned in Sec. 2.1.2, however, those methods require both positive and negative experience of traversals. Collecting negative experience involves intentionally making a robot bump into obstacles, which may not be allowed in some environments. We thus employed PU learning to enable the traversability estimation model to learn from only positive experience and showed that the model can be trained in a PU framework with the feature extraction by the semantic segmentation model. We believe that PU learning will enable more practical training of traversability estimation models. To realize it, it is crucial to achieve good feature representation. One possible approach would be combining feature embedding via contrastive learning [151, 152] with the PU learning framework.

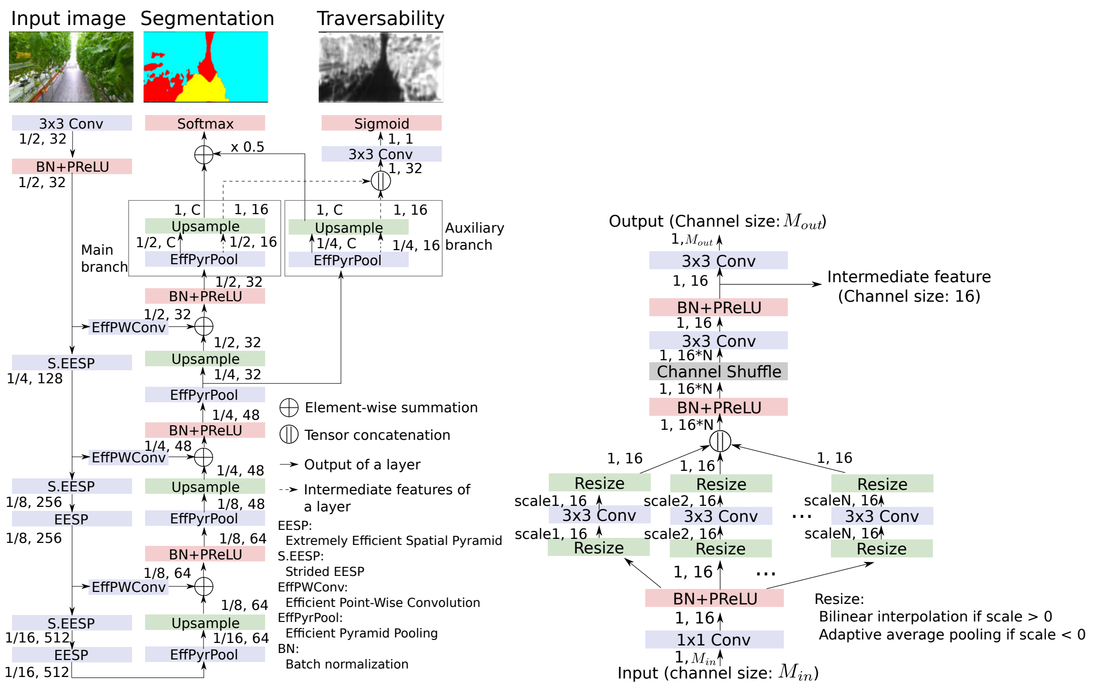
Multi-modal perception Throughout the work in this thesis, we relied on RGB images for object class and traversability estimation because we presumed that the appearance is highly informative for the estimation as human estimates them with the eyes. Nevertheless, using range sensors such as a LiDAR, or a stereo sensor may provide additional information about the object class and traversability. In fact, some researchers have used LiDARs for detecting vegetation [61, 62], though they did not apply it to actual robot navigation. Recently, deep learning techniques have been applied to 3D point clouds [153, 154, 155] and shown high classification ability in tasks such as semantic segmentation. Introducing such techniques may be beneficial for improving the estimation performance in complex environments.

Haptic information can also be used to verify the object traversability. Some legged robots use torque force information from their legs to estimate whether the terrain that they are stepping on is rigid or deformable [51]. Such force information from the objects can be used while traversing through the objects that are recognized as traversable plants by the proposed method as a verification for safety.

Appendix A

Details of the network structure

Fig. A.1(a) shows a complete structure of our network architecture. Our network is based on ESPNetv2 [101], a light-weight semantic segmentation network. It consists of modules such as Extremely Efficient Spatial Pyramid (EESP), Efficient Point-Wise Convolution (EffPWConv), Efficient Pyramid Pooling (EffPyrPool), which consist of several convolutions with different types of kernels, batch normalization, and activation layers. For detailed descriptions of those modules, we refer the readers to [101]. An auxiliary segmentation branch and a branch for estimating traversability, i.e., TEM are attached to the middle of the main network of ESPNetv2. The features for TEM are generated by concatenating the intermediate features from the EffPyrPool layer in each segmentation branch (See Fig. A.1(b)). The intermediate feature from the SSM is fed into a 3×3 convolution and the sigmoid function to produce pixel-wise probability predictions.



(a) Network architecture (b) Efficient Pyramid Pooling (EffPyrPool)

Figure A.1: A detailed diagram of our architecture. The numbers beside each layer shows the spatial resolution relative to the input image and the channel size of the output of the layer, respectively. Based on ESPNetv2 [101], an auxiliary segmentation branch is added to estimate pixel-wise uncertainty for the training of SSM (see 3.2.4). For estimating traversability, the intermediate features of the two segmentation branches are concatenated and fed into a 3×3 convolution followed by the sigmoid function. For the detail of each network layer, refer to [101].

Appendix B

Analysis of the prediction distributions

In Table B.1 , we show the distributions of the predictions by each of the source models over the target classes conditioned on the source classes, as well as the target label ID of the maximum probability for each source class and the actual target label ID used in the label conversion to see the validity of the heuristic label mappings. The distributions are calculated using the labeled test images of Greenhouse A. In the majority of the source classes, the target label of the maximum probability is the same as the one assigned heuristically. “Fence” in CamVid, “Road” in Cityscapes, and “Tree” in Freiburg Forest were assigned a target label different from the one with the maximum probability. In the model trained with Cityscapes, some source classes did not appear in the prediction on the target images.

Note that these distributions are calculated with the ground truth labels of the target images, which we do not expect to have in real environments, and thus this information is unable. In practice, as we show in the experiments, the definition of the mappings based on the heuristics resulted in good performances.

Table B.1: Prediction distribution. 1: Plant, 2: Artificial object, 3: Ground, 4: Other (not considered in the distribution analysis), in the target label space.

(a) CamVid

	Sky	Building	Pole	Road	Pavement	Tree	Sign symbol	Fence	Car	Pedestrian	Bicyclist	Road marking
1	1.72	19.6	11.9	0.64	0.98	95.3	26.4	93.07	0.00	0.00	36.00	28.6
2	92.0	80.0	71.3	38.3	37.8	4.70	73.6	6.93	81.0	98.3	61.3	59.3
3	6.27	0.42	16.9	61.0	61.2	0.00	0.00	0.00	19.0	1.70	2.70	12.1
max	2	2	2	3	3	1	2	1	2	2	2	2
ξ_i	4	2	2	3	3	1	2	2	2	4	4	2

(b) Cityscapes

	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle
1	0.00	6.20	0.50	0.00	4.55	1.20	0.00	0.00	83.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	28.8
2	54.3	21.1	99.5	0.00	90.0	81.2	0.00	0.00	16.8	0.00	0.00	100.0	0.00	85.6	0.00	100.0	0.00	0.00	69.5
3	45.7	72.7	0.00	0.00	5.45	17.6	0.00	0.00	0.20	0.00	0.00	0.00	0.00	14.4	0.00	0.00	0.00	0.00	1.70
max	2	3	2	-	2	2	-	-	1	-	-	2	-	2	-	2	-	-	2
ξ_i	3	3	2	2	2	2	2	2	1	3	4	4	4	2	2	2	2	2	2

(c) Prediction distribution (Freiburg Forest)

	Road	Grass	Tree	Sky	Obstacle
1	1.15	48.73	43.63	0.09	0.00
2	18.83	28.54	50.06	96.63	0.00
3	80.02	22.72	6.30	3.28	0.00
max	3	1	2	2	-
ξ_i	3	1	1	4	2

Appendix C

Policy of manual annotation on the test images

For the evaluation of our method, we created test datasets for Greenhouse A, B, and C with manually annotated labels. All annotation was done by the first author, who did not have specific experience of pixel-level annotation on image datasets.

We adopted coarse annotation to reduce the time of manual annotation. For example, small regions of objects other than plants that can be seen through the plant rows are annotated as plants, rather than the object classes that the regions actually belong to. The aim our method is to train the model for scene recognition in robot navigation to identify the object class of the regions in an image, especially regions of plants covering the paths, and to make a decision whether to traverse the object based on the object class. For this purpose, we suppose that pixel-level precision is not necessary. The model should rather be able to recognize the presence of plant etc. We, therefore, gave region-wise annotation rather than labels with pixel-level precision, so that the ability of recognizing the object class of image regions can be evaluated.

Appendix D

Qualitative evaluation on Greenhouse B and C

Fig. D.1 and D.2 show the qualitative results of the adaptation to Greenhouse B and C, respectively. Similar to the results on Greenhouse A shown in 3.3.2, more smooth and noise-less segmentation is achieved by the proposed method.

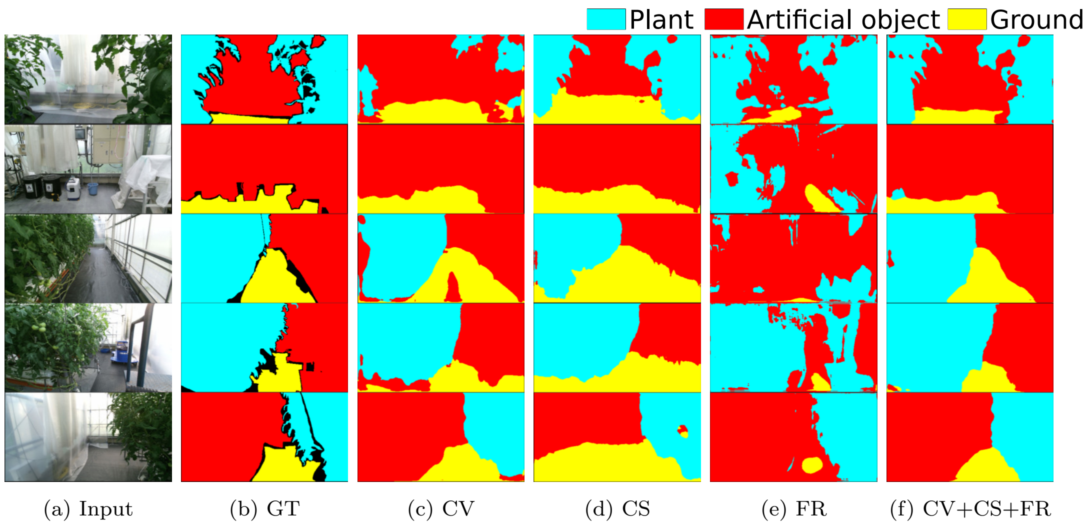
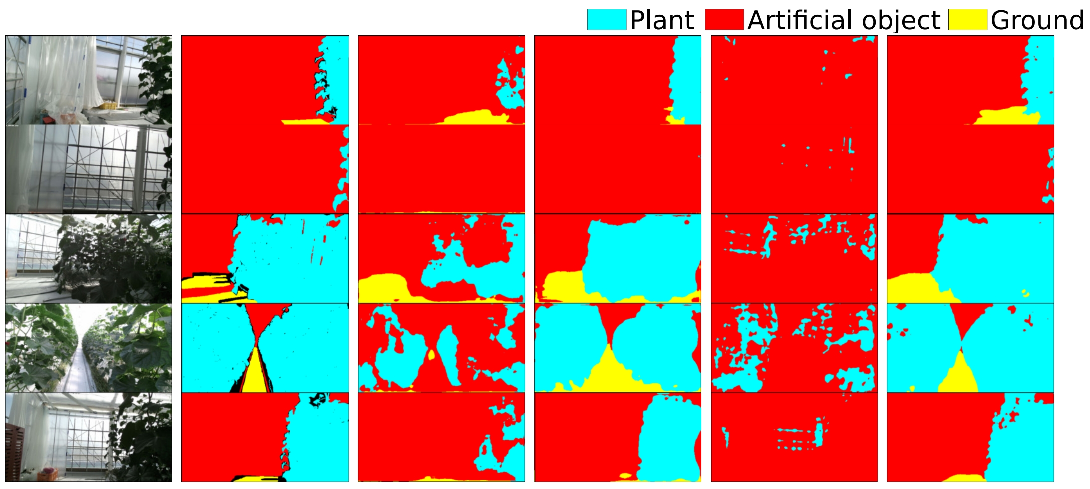


Figure D.1: Result of the adaptation on Greenhouse B



(a) Input (b) GT (c) CV (d) CS (e) FR (f) CV+CS+FR

Figure D.2: Result of the adaptation on Greenhouse C

Appendix E

Details of experimental settings

E.1 Implementation details of the baseline methods

E.1.1 Description of the feature for SP-SVM

We used the SVM model implemented in scikit-learn library [156]. The Radial Basis Function (RBF) kernel was used with the default parameters and one-versus-one decision function was chosen. For superpixel generation, we used a method by Felzenszwalb et al. [157] implemented in scikit-image library [158].

The feature design we adopt is based on the one used in a paper [34]. It consists of color features and texture features. The color features are mean RGB and HSV values, histograms of hue and saturation values. The texture features are calculated via applying filters in LM filter bank [159]. The filter responses are summarized by averaging and taking maximum within the superpixel and concatenated. In addition to the original feature design in [34], we added location features which are mean of x and y coordinates to exploit the spatial information.

E.1.2 Training details of the UDA baselines

Network architecture

DeepLab v2 [84] with ResNet101 backbone [81] is used as the semantic segmentation model in the original work of all the baseline methods of UDA. Although it is different from our model, we also adopt DeepLab v2 in the baselines to keep their implementation

Table E.1: Description of the feature for SP-SVM

Type	Description	Dim.
RGB value	RGB mean	3
HSV value	RGB mean in HSV color space	3
Hue	Histogram of hue	8
Saturation	Histogram of saturation	5
LM average	Average filter responses from LM filter set	18
LM Maximum	Histogram of maximum filter responses	18
Location	Mean x / y coordinate values normalized by the image width / height	2
	Total number of dimension	57

as much as possible. CRST and ProDA are trained on an NVIDIA Quadro RTX 8000, and Seg-Uncertainty is trained on an NVIDIA GeForce 1080Ti with 11GB memory.

CRST

In CRST [121], the source models are pre-trained by supervised training and we follow the implementation. We first train a model with a source dataset with its original label set for 200 epochs. We then replace the final classification layer with C-class classifier where C denotes the number of object classes in the target dataset. The model with the replaced classifier is fine-tuned with the source labels converted to the target label set defined in Table 3.1 for 50 epochs. Initial learning rate is 1×10^{-4} for the final classification layer and 1×10^{-3} for the rest of the layers. The polynomial learning rate scheduling [160] is used with the power of 0.9. Other hyper-parameters are unchanged from their source code[1].

Seg-Uncertainty

We train Seg-Uncertainty [122] with MRNet [161]. MRNet first trains, as a warm-up, a segmentation network with both the source and the target datasets via adversarial training at the segmentation output which was originally proposed in [117]. Other hyper-parameters are unchanged from their source code[2].

ProDA

ProDA [123] follows the same warm-up strategy as Seg-Uncertainty. We, therefore, use the same warm-up models trained in Seg-Uncertainty in ProDA. Although ProDA and Seg-Uncertainty both use DeepLabv2 as a segmentation network, the architecture of the segmentation networks is slightly different between Seg-Uncertainty and ProDA, such as the number of channels of the intermediate feature map. We modified the source code of ProDA to adjust to the network. Other hyper-parameters are unchanged from their source code[3].

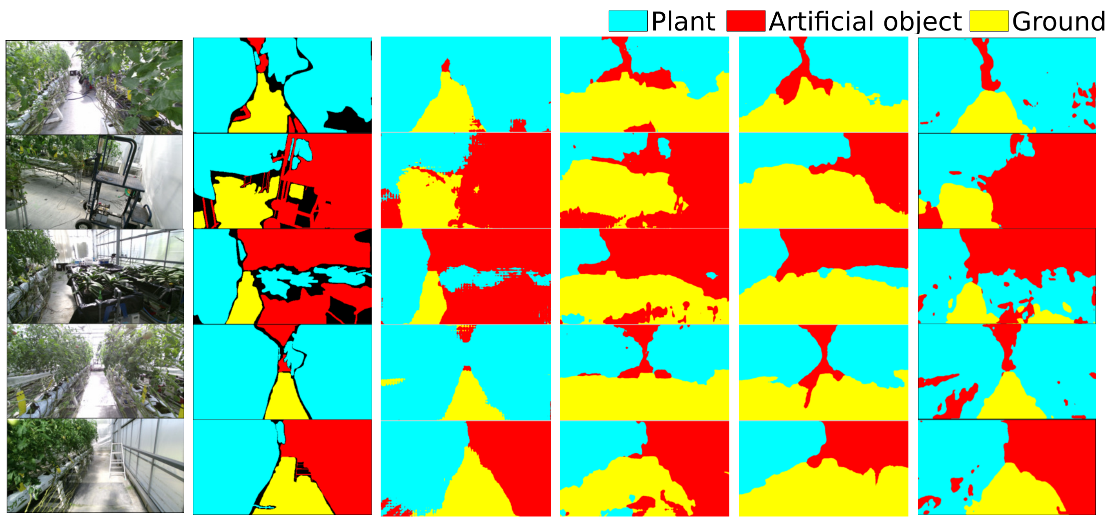
E.2 Results of the baseline methods

E.2.1 Qualitative evaluation of the UDA methods

Fig. E.1 shows qualitative results of the UDA baseline methods on Greenhouse A dataset. As shown quantitatively in 3.3.3, the proposed method (CV+CS+FR) outperformed the baseline methods, especially Seg-Uncertainty [122] and ProDA [123]. In the segmentation results of Seg-Uncertainty and ProDA, a large part of the bottom of the images are wrongly classified as ground regions. This is similar to the structural feature of the images in CS. This tendency was possibly enhanced by the adversarial learning on the output layer employed in the first stage of those methods.

Compared to Seg-Uncertainty and ProDA, CRST [121] produced better segmentation results. CRST chooses pseudo-labels simply based on the confidence of the predictions,

[1]`https://github.com/yzou2/CRST`
[2]`https://github.com/layumi/Seg-Uncertainty`
[3]`https://github.com/microsoft/ProDA`



(a) Input (b) GT (c) CRST (d) Seg-Unc. (e) ProDA (f) Proposed

Figure E.1: Result of the baseline methods

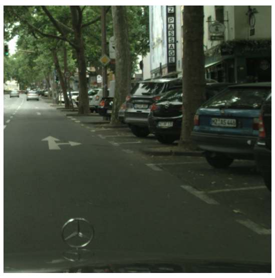
and resulting pseudo-labels are sparse. In other words, CRST conservatively generates pseudo-labels with only confidently predicted labels. As a result, the performance was better than the other two baselines. Our proposed method also generates pseudo-labels in a conservative way via multiple model's agreement. It may imply that it is a better approach to generate pseudo-labels in a conservative way when the source and the target have a large structural difference. The proposed method provides a natural way of doing so without relying on a specific dataset, which allows for avoiding biased training.

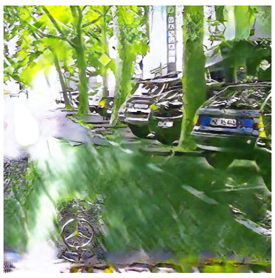
E.2.2 Effect of GAN-based image style transfer

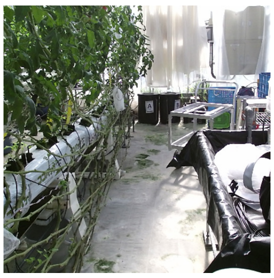
We trained the CycleGANs using CS and FR as source datasets and Greenhouse A as a target dataset. It is the first step of MADAN [126], a multi-source UDA method for semantic segmentation. For each source dataset, we trained a CycleGAN for 200 epochs.

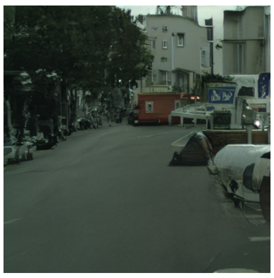
Fig. E.2 shows the results of image style transfer between CS/FR and Greenhouse A datasets. The style transfer resulted in inconsistency of image contents, e.g., the ground region in Fig. E.2(a) is transferred to green plant-like objects in Fig. E.2(b), and the bottom part of the plant row and a large part of an artificial object on the right in Fig. E.2(c) are transferred to ground in Fig. E.2(d). Similarly, plants on the right in Fig. E.2(e) are transferred to gray objects like a wall in Fig. E.2(f), and some plant parts in Fig. E.2(g) are mapped to sky in E.2(h). This may be due to the large domain shift that stems from the structural differences of the environments.

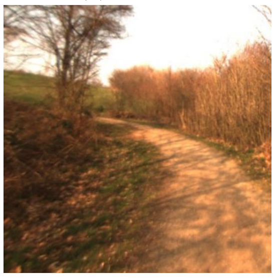
Although MADAN further employs training procedures to aggregate the data from multiple domains closer to each other with constraints on the semantic consistency [126], the inconsistency of the style transfer in the first step shown above will affect the adaptation performance.

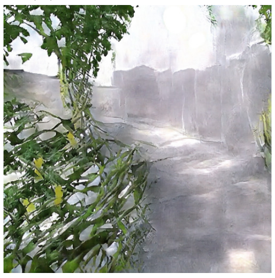

(a) Real CS

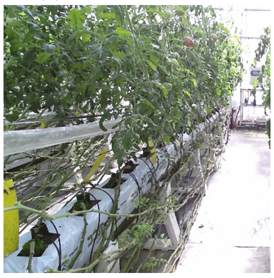

(b) CS → Greenhouse


(c) Real Greenhouse


(d) Greenhouse → CS


(e) Real FR


(f) FR → Greenhouse


(g) Real Greenhouse

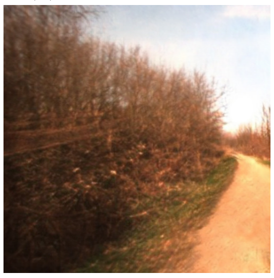

(h) Greenhouse → FR

Figure E.2: Results of image style transfer by CycleGAN in MADAN. The style transfer resulted in inconsistency of image contents. This may imply that such style transfer is not effective when the structures of the source and the target have a large discrepancy as in our problem setting.

Bibliography

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.* Cambridge, Mass.: MIT Press, 2005. DOI: 10.5555/1121596.

[2] A. K. Pandey and R. Gelin, "A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of Its Kind," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 40–48, 2018. DOI: 10.1109/MRA.2018.2833157.

[3] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, H. Hung, O. A. I. Ramírez, M. Joosse, H. Khambhaita, T. Kucner, B. Leibe, A. J. Lilienthal, T. Linder, M. Lohse, M. Magnusson, B. Okal, L. Palmieri, U. Rafi, M. van Rooij, and L. Zhang, "SPENCER: A Socially Aware Service Robot for Passenger Guidance and Help in Busy Airports," in *Field and Service Robotics*, New York: Springer International Publishing, 2016, pp. 607–622. DOI: 10.1007/978-3-319-27702-8_40.

[4] Waymo, "Waymo Safety Report," Tech. Rep., 2020. [Online]. Available: https://storage.googleapis.com/sdc-prod/v1/safety-report/2020-09-waymo-safety-report.pdf.

[5] Tier IV, "Tier IV Safety Report," Tech. Rep., 2020. [Online]. Available: https://assets.ctfassets.net/rfp71c5fx4wl/3YwptpgxDOmVddje77hxqX/f9468a13518bee9876ed7270c8ba76de/Safety_Report_Eng_9_29_master_compressed.pdf.

[6] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, and M. Smith, "Agricultural Robotics: The Future of Robotic Agriculture," Tech. Rep., 2018. [Online]. Available: http://arxiv.org/abs/1806.06762.

[7] MAFF, *Promotion of Smart Agriculture.* [Online]. Available: https://www.maff.go.jp/e/policies/tech_res/smaagri/attach/pdf/robot-1.pdf (visited on 11/08/2022).

[8] L. F. Oliveira, A. P. Moreira, and M. F. Silva, "Advances in forest robotics: A state-of-the-art survey," *Robotics*, vol. 10, no. 2, pp. 1–20, 2021. DOI: 10.3390/robotics10020053.

[9] Y. Qi, N. C. Coops, L. D. Daniels, and C. R. Butson, "Comparing tree attributes derived from quantitative structure models based on drone and mobile laser scanning point clouds across varying canopy cover conditions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 192, pp. 49–65, 2022. DOI: 10.1016/j.isprsjprs.2022.07.021.

[10] E. Frías, M. Previtali, L. Díaz-Vilariño, M. Scaioni, and H. Lorenzo, "Optimal scan planning for surveying large sites with static and mobile mapping systems," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 192, pp. 13–32, 2022. DOI: 10.1016/j.isprsjprs.2022.07.025.

[11] T. Tsubouchi, A. Asano, T. Mochizuki, S. Kondou, K. Shiozawa, M. Matsumoto, S. Tomimura, S. Nakanishi, Y. Chiba, and T. Hayami, "Forest 3D mapping and tree sizes measurement for forest management based on sensing technology for mobile robots," *Springer Tracts in Advanced Robotics*, vol. 92, pp. 357–368, 2014. DOI: 10.1007/978-3-642-40686-7_24.

[12] T. Bakker, H. Wouters, K. van Asselt, J. Bontsema, L. Tang, J. Müller, and G. van Straten, "A vision based row detection system for sugar beet," *Computers and Electronics in Agriculture*, vol. 60, no. 1, pp. 87–95, 2008. DOI: 10.1016/j.compag.2007.07.006.

[13] J. Xue, L. Zhang, and T. E. Grift, "Variable field-of-view machine vision based row guidance of an agricultural robot," *Computers and Electronics in Agriculture*, vol. 84, pp. 85–91, 2012. DOI: 10.1016/j.compag.2012.02.009.

[14] F. B. Malavazi, R. Guyonneau, J. B. Fasquel, S. Lagrange, and F. Mercier, "LiDAR-only based navigation algorithm for an autonomous agricultural robot," *Computers and Electronics in Agriculture*, vol. 154, pp. 71–79, 2018. DOI: 10.1016/j.compag.2018.08.034.

[15] W. Winterhalter, F. Fleckenstein, C. Dornhege, and W. Burgard, "Localization for precision navigation in agricultural fields—Beyond crop row following," *Journal of Field Robotics*, vol. 38, no. 3, pp. 429–451, 2020. DOI: 10.1002/rob.21995.

[16] T. D. Le, V. R. Ponnambalam, J. G. O. Gjevestad, and P. J. From, "A low-cost and efficient autonomous row-following robot for food production in polytunnels," *Journal of Field Robotics*, vol. 37, no. 2, pp. 309–321, 2020. DOI: 10.1002/rob.21878.

[17] A. Mandow, J. Gomez-de-Gabriel, J. Martinez, V. Munoz, A. Ollero, and A. Garcia-Cerezo, "The autonomous mobile robot AURORA for greenhouse operation," *IEEE Robotics & Automation Magazine*, vol. 3, no. 4, pp. 18–28, 1996. DOI: 10.1109/100.556479.

[18] A. Mowshowitz, A. Tominaga, and E. Hayashi, "Robot Navigation in Forest Management," *Journal of Robotics and Mechatronics*, vol. 30, no. 2, pp. 223–230, 2018. DOI: 10.20965/jrm.2018.p0223.

[19] E. Jelavić, D. Jud, P. Egli, and M. Hutter, "Robotic Precision Harvesting: Mapping, Localization, Planning and Control for a Legged Tree Harvester," *Field Robotics*, vol. 2, no. 1, pp. 1386–1431, 2022. DOI: 10.55417/fr.2022046.

[20] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An Autonomous Self-Supervised Learning-Based Navigation System," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021. DOI: 10.1109/LRA.2021.3057023.

[21] M. V. Gasparino, A. N. Sivakumar, Y. Liu, A. E. B. Velasquez, V. A. H. Higuti, J. Rogers, H. Tran, and G. Chowdhary, "WayFAST: Navigation With Predictive Traversability in the Field," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 651–10 658, 2022. DOI: 10.1109/LRA.2022.3193464.

[22] A. Polevoy, C. Knuth, K. M. Popek, and K. D. Katyal, “Complex Terrain Navigation via Model Error Prediction.” In *Proc. of the IEEE International Conference on Robotics and Automation*, IEEE, 2022, pp. 9411–9417. DOI: 10.1109/ICRA46639.2022.9811644.

[23] Dongshin Kim, Jie Sun, Sang Min Oh, J. Rehg, and A. Bobick, “Traversability classification using unsupervised on-line visual learning for outdoor robot navigation.” In *Proc. of the IEEE International Conference on Robotics and Automation*, IEEE, 2006, pp. 518–525. DOI: 10.1109/ROBOT.2006.1641763.

[24] C. Elkan and K. Noto, “Learning Classifiers from Only Positive and Unlabeled Data.” In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 213–220.

[25] I. Katramados, S. Crumpler, and T. P. Breckon, “Real-Time Traversable Surface Detection by Colour Space Fusion and Temporal Analysis.” In *Proc. of the International Conference on Computer Vision Systems*, 2009, pp. 265–274. DOI: 10.1007/978-3-642-04667-4_27.

[26] I. Bogoslavskyi, O. Vysotska, J. Serafin, G. Grisetti, and C. Stachniss, “Efficient traversability analysis for mobile robots using the Kinect sensor.” In *Proc. of the European Conference on Mobile Robots*, IEEE, 2013, pp. 158–163. DOI: 10.1109/ECMR.2013.6698836.

[27] C. J. Taylor and A. Cowley, “Parsing Indoor Scenes Using RGB-D Imagery,” in *Robotics*, The MIT Press, 2013. DOI: 10.7551/mitpress/9816.003.0056.

[28] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd, B. Siciliano and O. Khatib, Eds., ser. Springer Handbooks. Cham: Springer International Publishing, 2016. DOI: 10.1007/978-3-319-32552-1.

[29] T. Chang, S. Legowik, and M. N. Abrams, “Concealment and Obstacle Detection for Autonomous Driving.” In *Proc. of the Science & Technology for Development - Robotics & Applications*, 1999, pp. 1–15.

[30] P. Nordin and J. Nygårds, “Local Navigation using Traversability Maps,” *IFAC Proceedings Volumes*, vol. 43, no. 16, pp. 324–329, 2010. DOI: 10.3182/20100906-3-IT-2019.00057.

[31] À. Santamaria-Navarro, E. H. Teniente, M. Morta, and J. Andrade-Cetto, “Terrain Classification in Complex Three-dimensional Outdoor Environments,” *Journal of Field Robotics*, vol. 32, no. 1, pp. 42–60, 2015. DOI: 10.1002/rob.21521.

[32] B. Suger, B. Steder, and W. Burgard, “Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3D-lidar data.” In *Proc. of the IEEE International Conference on Robotics and Automation*, IEEE, 2015, pp. 3941–3946. DOI: 10.1109/ICRA.2015.7139749.

[33] J. Frey, D. Hoeller, S. Khattak, and M. Hutter, “Locomotion Policy Guided Traversability Learning using Volumetric Representations of Complex Environments.” In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2022, pp. 5722–5729. DOI: 10.1109/IROS47612.2022.9982190.

[34] D. Kim, S. M. Oh, and J. M. Rehg, "Traversability classification for UGV navigation: a comparison of patch and superpixel representations." In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 3166–3173. DOI: 10.1109/IROS.2007.4399610.

[35] R. Hadsell, A. Erkan, P. Sermanet, M. Scoffier, U. Muller, and Yann LeCun, "Deep belief net learning in a long-range vision system for autonomous off-road driving." In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 628–633. DOI: 10.1109/IROS.2008.4651217.

[36] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006. DOI: 10.1162/neco.2006.18.7.1527.

[37] Y. N. Khan, P. Komma, and A. Zell, "High resolution visual terrain classification for outdoor robots." In *Proc. of the IEEE International Conference on Computer Vision Workshops*, IEEE, 2011, pp. 1014–1021. DOI: 10.1109/ICCVW.2011.6130362.

[38] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324.

[39] H. Lee, K. Kwak, and S. Jo, "An incremental nonparametric Bayesian clustering-based traversable region detection method," *Autonomous Robots*, vol. 41, no. 4, pp. 795–810, 2017. DOI: 10.1007/s10514-016-9588-7.

[40] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Image classification for ground traversability estimation in robotics." In *Proc. of the International Conference on Advanced Concepts for Intelligent Vision Systems*, 2017, pp. 325–336. DOI: 10.1007/978-3-319-70353-4_28.

[41] D. Maturana, P.-W. Chou, M. Uenoyama, and S. Scherer, "Real-Time Semantic Mapping for Autonomous Off-Road Navigation," in *Field and Service Robotics*, vol. 5, 2018, pp. 335–350. DOI: 10.1007/978-3-319-67361-5_22.

[42] V. Suryamurthy, V. S. Raghavan, A. Laurenzi, N. G. Tsagarakis, and D. Kanoulas, "Terrain Segmentation and Roughness Estimation using RGB Data: Path Planning Application on the CENTAURO Robot." In *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2019, pp. 1–8. DOI: 10.1109/Humanoids43949.2019.9035009.

[43] T. Guan, Z. He, D. Manocha, and L. Zhang, "TTM: Terrain Traversability Mapping for Autonomous Excavator Navigation in Unstructured Environments," Tech. Rep., 2021. arXiv: 2109.06250. [Online]. Available: http://arxiv.org/abs/2109.06250.

[44] C. Brooks, K. Iagnemma, and S. Dubowsky, "Vibration-based Terrain Analysis for Mobile Robots." In *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 2005, IEEE, 2005, pp. 3415–3420. DOI: 10.1109/ROBOT.2005.1570638.

[45] B. Sebastian and P. Ben-Tzvi, "Support vector machine based real-time terrain estimation for tracked robots," *Mechatronics*, vol. 62, no. August, p. 102 260, 2019. DOI: 10.1016/j.mechatronics.2019.102260.

[46] G. Haddeler, M. Y. Chuah, Y. You, J. Chan, A. H. Adiwahono, W. Y. Yau, and C. M. Chew, “Traversability analysis with vision and terrain probing for safe legged robot navigation,” *Frontiers in Robotics and AI*, vol. 9, no. August, pp. 1–14, 2022. DOI: 10.3389/frobt.2022.887910.

[47] C. Rasmussen, “Combining laser range, color, and texture cues for autonomous road following.” In *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 4, IEEE, 2002, pp. 4320–4325. DOI: 10.1109/ROBOT.2002.1014439.

[48] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and R. Warner, “Toward Reliable Off Road Autonomous Vehicles Operating in Challenging Environments,” *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 449–483, 2006. DOI: 10.1177/0278364906065543.

[49] M. Happold, M. Ollis, and N. Johnson, “Enhancing Supervised Terrain Classification with Predictive Unsupervised Learning.” In *Proc. of the Robotics: Science and Systems*, 2006.

[50] J. Sock, J. Kim, J. Min, and K. Kwak, “Probabilistic traversability map generation using 3D-LIDAR and camera.” In *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 2016-June, IEEE, 2016, pp. 5631–5637. DOI: 10.1109/ICRA.2016.7487782.

[51] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning robust perceptive locomotion for quadrupedal robots in the wild,” *Science Robotics*, vol. 7, no. 62, 2022. DOI: 10.1126/scirobotics.abk2822.

[52] M. Wang, J. Zhou, J. Tu, and C. Liu, “Learning Long-range Terrain Perception for Autonomous Mobile Robots,” *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 5, 2010. DOI: 10.5772/7245.

[53] S. Zhou, J. Xi, M. W. McDaniel, T. Nishihata, P. Salesses, and K. Iagnemma, “Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain,” *Journal of Field Robotics*, vol. 29, no. 2, pp. 277–297, 2012. DOI: 10.1002/rob.21417.

[54] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness, “Towards learned traversability for robot navigation: From underfoot to the far field,” *Journal of Field Robotics*, vol. 23, no. 11-12, pp. 1005–1017, 2006. DOI: 10.1002/rob.20168.

[55] D. Lieb, A. Lookingbill, and S. Thrun, “Adaptive Road Following using Self-Supervised Learning and Reverse Optical Flow.” In *Proc. of the Robotics: Science and Systems*, vol. 1, Robotics: Science and Systems Foundation, 2005, pp. 273–280. DOI: 10.15607/RSS.2005.I.036.

[56] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, “Where Should I Walk? Predicting Terrain Properties From Images Via Self-Supervised Learning,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1509–1516, 2019. DOI: 10.1109/LRA.2019.2895390.

[57] Y. Onozuka, R. Matsumi, and M. Shino, "Weakly-Supervised Recommended Traversable Area Segmentation Using Automatically Labeled Images for Autonomous Driving in Pedestrian Environment with No Edges," *Sensors*, vol. 21, no. 2, p. 437, 2021. DOI: `10.3390/s21020437`.

[58] S. Matsuzaki, K. Yamazaki, Y. Hara, and T. Tsubouchi, "Traversable Region Estimation for Mobile Robots in an Outdoor Image," *Journal of Intelligent & Robotic Systems*, vol. 92, no. 3-4, pp. 453–463, 2018. DOI: `10.1007/s10846-017-0760-x`.

[59] R. Schmid, D. Atha, F. Scholler, S. Dey, S. Fakoorian, K. Otsu, B. Ridge, M. Bjelonic, L. Wellhausen, M. Hutter, and A.-a. Agha-mohammadi, "Self-Supervised Traversability Prediction by Learning to Reconstruct Safe Terrain." In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2022, pp. 12 419–12 425. DOI: `10.1109/IROS47612.2022.9981368`.

[60] M. Foroutan, W. Tian, and C. T. Goodin, "Assessing Impact of Understory Vegetation Density on Solid Obstacle Detection for Off-Road Autonomous Ground Vehicles," *ASME Letters in Dynamic Systems and Control*, vol. 1, no. 2, pp. 1–8, 2021. DOI: `10.1115/1.4047816`.

[61] J. Macedo, R. Manduchi, and L. Matthies, "Ladar-based Discrimination of Grass from Obstacles for Autonomous Navigation," in *Experimental Robotics VII*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 111–120. DOI: `10.1007/3-540-45118-8_12`.

[62] M. Hebert and N. Vandapel, "Terrain classification techniques from ladar data for autonomous navigation." In *Proc. of the Collaborative Technology Alliances Conference*, 2003.

[63] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *Journal of Field Robotics*, vol. 23, no. 10, pp. 839–861, 2006. DOI: `10.1002/rob.20134`.

[64] K. M. Wurm, R. Kummerle, C. Stachniss, and W. Burgard, "Improving robot navigation in structured outdoor environments by identifying vegetation from laser data." In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 1217–1222. DOI: `10.1109/IROS.2009.5354530`.

[65] D. M. Bradley, S. M. Thayer, A. Stentz, and P. Rander, "Vegetation detection for mobile robot navigation," Robotics Institute, Carnegie Mellon University, Tech. Rep., 2004. [Online]. Available: `http://www.ri.cmu.edu/pub_files/pub4/bradley_david_2004_2/bradley_david_2004_2.pdf`.

[66] D. M. Bradley, R. Unnikrishnan, and J. Bagnell, "Vegetation Detection for Driving in Complex Environments." In *Proc. of the IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 503–508. DOI: `10.1109/ROBOT.2007.363836`.

[67] C. Wellington and A. Stentz, "Online adaptive rough-terrain navigation in vegetation." In *Proc. of the IEEE International Conference on Robotics and Automation*, IEEE, 2004, pp. 96–101. DOI: `10.1109/ROBOT.2004.1307135`.

[68] A. Narenthiran Sivakumar, S. Modi, M. Valverde Gasparino, C. Ellis, A. E. Baquero Velasquez, G. Chowdhary, and S. Gupta, "Learned Visual Navigation for Under-Canopy Agricultural Robots." In *Proc. of the Robotics: Science and Systems*, 2021.

[69] A. E. Baquero Velasquez, V. A. Hisano Higuti, M. Valverde Gasparino, A. Sivakumar, M. Becker, and G. Chowdhary, "Multi-Sensor Fusion based Robust Row Following for Compact Agricultural Robots," *Field Robotics*, vol. 2, no. 1, pp. 1291–1319, 2022. DOI: 10.55417/fr.2022043.

[70] C. Sevastopoulos and S. Konstantopoulos, "A Survey of Traversability Estimation for Mobile Robots," *IEEE Access*, vol. 10, no. April, pp. 96 331–96 347, 2022. DOI: 10.1109/ACCESS.2022.3202545.

[71] B. Mehlig, *Machine Learning with Neural Networks.* Cambridge University Press, 2021. DOI: 10.1017/9781108860604.

[72] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, ser. Cornell Aeronautical Laboratory. Report no. VG-1196-G-8. Spartan Books, 1962.

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

[74] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2018. DOI: 10.5555/3086952.

[75] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980. DOI: 10.1007/BF00344251.

[76] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989. DOI: 10.1162/neco.1989.1.4.541.

[77] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[78] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. DOI: 10.1145/3065386.

[79] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions." In *Proc. of the 2015 Computer Vision and Pattern Recognition*, IEEE, 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[80] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." In *Proc. of the International Conference on Learning Representations,*, 2015, pp. 1–14.

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[82] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. DOI: `10.1007/s11263-015-0816-y`.

[83] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions." In *Proc. of the International Conference on Learning Representations*, 2016.

[84] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018. DOI: `10.1109/TPAMI.2017.2699184`.

[85] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, and M. Andreetto, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Tech. Rep., 2017. arXiv: `1704.04861v1`.

[86] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning Separable Filters." In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2013, pp. 2754–2761. DOI: `10.1109/CVPR.2013.355`.

[87] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Efficient ConvNet for real-time semantic segmentation." In *Proc. of the IEEE Intelligent Vehicles Symposium*, IEEE, 2017, pp. 1789–1794. DOI: `10.1109/IVS.2017.7995966`.

[88] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021. DOI: `10.1109/TNNLS.2021.3084827`.

[89] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2017, pp. 5987–5995. DOI: `10.1109/CVPR.2017.634`.

[90] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation." In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2015, pp. 3431–3440. DOI: `10.1109/CVPR.2015.7298965`.

[91] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. DOI: `10.1109/TPAMI.2016.2644615`.

[92] H. Su, F. Xing, X. Kong, Y. Xie, S. Zhang, and L. Yang, "Robust Cell Detection and Segmentation in Histopathological Images Using Sparse Reconstruction and Stacked Denoising Autoencoders," in *Medical Image Computing and Computer-Assisted Intervention*, vol. 9351, 2017, pp. 257–278. DOI: `10.1007/978-3-319-42999-1_15`.

[93] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," Tech. Rep., 2017. arXiv: `1706.05587`. [Online]. Available: `http://arxiv.org/abs/1706.05587`.

[94] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation." In *Proc. of the European Conference on Computer Vision*, 2018, pp. 833–851. DOI: 10.1007/978-3-030-01234-2_49.

[95] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2017, pp. 6230–6239. DOI: 10.1109/CVPR.2017.660.

[96] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep High-Resolution Representation Learning for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2021. DOI: 10.1109/TPAMI.2020.2983686.

[97] Y. Yuan, X. Chen, and J. Wang, "Object-Contextual Representations for Semantic Segmentation." In *Proc. of the European Conference on Computer Vision*, vol. 12351 LNCS, 2020, pp. 173–190. DOI: 10.1007/978-3-030-58539-6_11.

[98] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation," Tech. Rep., 2016. arXiv: 1606.02147v1. [Online]. Available: https://arxiv.org/pdf/1606.02147.pdf.

[99] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018. DOI: 10.1109/TITS.2017.2750080.

[100] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation," in *European Conference on Computer Vision*, 2018, pp. 561–580. DOI: 10.1007/978-3-030-01249-6_34.

[101] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "ESPNetv2: A Light-Weight, Power Efficient, and General Purpose Convolutional Neural Network." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2019, pp. 9182–9192. DOI: 10.1109/CVPR.2019.00941.

[102] L. C. Lulio, M. L. Tronco, and A. J. V. Porto, "JSEG-based image segmentation in computer vision for agricultural mobile robot navigation." In *Proc. of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, IEEE, 2009, pp. 240–245. DOI: 10.1109/CIRA.2009.5423201.

[103] M. Sharifi and XiaoQi Chen, "A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards." In *Proc. of the International Conference on Automation, Robotics and Applications*, IEEE, 2015, pp. 251–255. DOI: 10.1109/ICARA.2015.7081155.

[104] D. Aghi, S. Cerrato, V. Mazzia, and M. Chiaberge, "Deep Semantic Segmentation at the Edge for Autonomous Navigation in Vineyard Rows," Tech. Rep., 2021. arXiv: 2107.00700. [Online]. Available: http://arxiv.org/abs/2107.00700.

[105] Y.-K. Lin and S.-F. Chen, “Development of Navigation System for Tea Field Machine Using Semantic Segmentation,” *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 108–113, 2019. DOI: 10.1016/j.ifacol.2019.12.506.

[106] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression.” In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, IEEE, 2019, pp. 658–666. DOI: 10.1109/CVPR.2019.00075.

[107] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding.” In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2016, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350.

[108] G. Csurka, *Domain Adaptation in Computer Vision Applications*, G. Csurka, Ed., ser. Advances in Computer Vision and Pattern Recognition. Cham: Springer International Publishing, 2017. DOI: 10.1007/978-3-319-58347-1.

[109] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.

[110] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016. DOI: 10.1186/s40537-016-0043-6.

[111] S. Zhao, X. Yue, S. Zhang, B. Li, H. Zhao, B. Wu, R. Krishna, J. E. Gonzalez, A. L. Sangiovanni-Vincentelli, S. A. Seshia, and K. Keutzer, “A Review of Single-Source Deep Unsupervised Visual Domain Adaptation,” Tech. Rep., 2020, pp. 1–21. arXiv: 2009.00155. [Online]. Available: http://arxiv.org/abs/2009.00155.

[112] G. Wilson and D. J. Cook, “A Survey of Unsupervised Deep Domain Adaptation,” *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 5, pp. 1–46, 2020. DOI: 10.1145/3400066.

[113] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, “CyCADA: Cycle-Consistent Adversarial Domain Adaptation.” In *Proc. of the International Conference on Machine Learning*, 2018, pp. 1989–1998.

[114] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Perez, “ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation.” In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2019, pp. 2512–2521. DOI: 10.1109/CVPR.2019.00262.

[115] K. Saito, Y. Ushiku, and T. Harada, “Asymmetric tri-training for unsupervised domain adaptation.” In *Proc. of the International Conference on Machine Learning*, 2017, pp. 2988–2997.

[116] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-Adversarial Training of Neural Networks,” in *The Journal of Machine Learning Research*, 1, vol. 17, 2017, pp. 189–209. DOI: 10.1007/978-3-319-58347-1_10.

[117] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to Adapt Structured Output Space for Semantic Segmentation." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2018, pp. 7472–7481. DOI: 10.1109/CVPR.2018.00780.

[118] M. Biasetton, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised Domain Adaptation for Semantic Segmentation of Urban Scenes." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2019, pp. 1211–1220. DOI: 10.1109/CVPRW.2019.00160.

[119] I. J. Goodfellow, P.-A. Jean, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets." In *Proc. of the Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680. DOI: 10.5555/2969033.2969125.

[120] Y. Zou, Z. Yu, B. V. K. Vijaya Kumar, and J. Wang, "Unsupervised Domain Adaptation for Semantic Segmentation via Class-Balanced Self-training," in *European Conference on Computer Vision*, 2018, pp. 297–313. DOI: 10.1007/978-3-030-01219-9_18.

[121] Y. Zou, Z. Yu, X. Liu, B. V. K. V. Kumar, and J. Wang, "Confidence Regularized Self-Training." In *Proc. of the IEEE/CVF International Conference on Computer Vision*, IEEE, 2019, pp. 5981–5990. DOI: 10.1109/ICCV.2019.00608.

[122] Z. Zheng and Y. Yang, "Rectifying Pseudo Label Learning via Uncertainty Estimation for Domain Adaptive Semantic Segmentation," *International Journal of Computer Vision*, vol. 129, no. 4, pp. 1106–1120, 2021. DOI: 10.1007/s11263-020-01395-y.

[123] P. Zhang, B. Zhang, T. Zhang, D. Chen, Y. Wang, and F. Wen, "Prototypical Pseudo Label Denoising and Target Structure Learning for Domain Adaptive Semantic Segmentation." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2021, pp. 12 409–12 419. DOI: 10.1109/CVPR46437.2021.01223.

[124] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin, "Deep Cocktail Network: Multi-source Unsupervised Domain Adaptation with Category Shift." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2018, pp. 3964–3973. DOI: 10.1109/CVPR.2018.00417.

[125] S. Zhao, B. Li, C. Reed, P. Xu, and K. Keutzer, "Multi-source Domain Adaptation in the Deep Learning Era: A Systematic Survey," Tech. Rep., 2020. arXiv: 2002.12169. [Online]. Available: http://arxiv.org/abs/2002.12169.

[126] S. Zhao, B. Li, X. Yue, Y. Gu, P. Xu, R. Hu, H. Chai, and K. Keutzer, "Multi-source Domain Adaptation for Semantic Segmentation." In *Proc. of the Conference on Neural Information Processing Systems*, vol. 32, 2019, pp. 7287–7300.

[127] J. He, X. Jia, S. Chen, and J. Liu, "Multi-Source Domain Adaptation with Collaborative Learning for Semantic Segmentation." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2021, pp. 11 003–11 012. DOI: 10.1109/CVPR46437.2021.01086.

[128] S. J. Park, H. J. Park, E. S. Kang, B. H. Ngo, H. S. Lee, and S. I. Cho, "Pseudo Label Rectification via Co-Teaching and Decoupling for Multisource Domain Adaptation in Semantic Segmentation," *IEEE Access*, vol. 10, no. July, pp. 91 137–91 149, 2022. DOI: 10.1109/ACCESS.2022.3202190.

[129] I. Nigam, C. Huang, and D. Ramanan, "Ensemble Knowledge Transfer for Semantic Segmentation." In *Proc. of the IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2018, pp. 1499–1508. DOI: 10.1109/WACV.2018.00168.

[130] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for Data: Ground Truth from Computer Games," in *European Conference on Computer Vision*, vol. 9906, 2016, pp. 102–118. DOI: 10.1007/978-3-319-46475-6_7.

[131] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243. DOI: 10.1109/CVPR.2016.352.

[132] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-shot learning for semantic segmentation." In *Proc. of the British Machine Vision Conference*, 2017, pp. 167.1–167.13. DOI: 10.5244/c.31.167.

[133] K. Nguyen and S. Todorovic, "Feature weighting and boosting for few-shot segmentation." In *Proc. of the IEEE International Conference on Computer Vision*, 2019, pp. 622–631. DOI: 10.1109/ICCV.2019.00071.

[134] G. Li, V. Jampani, L. Sevilla-Lara, D. Sun, J. Kim, and J. Kim, "Adaptive Prototype Learning and Allocation for Few-Shot Segmentation." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2021, pp. 8330–8339. DOI: 10.1109/CVPR46437.2021.00823.

[135] H. Qi, M. Brown, and D. G. Lowe, "Low-Shot Learning with Imprinted Weights." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2018, pp. 5822–5830. DOI: 10.1109/CVPR.2018.00610.

[136] M. Siam, B. Oreshkin, and M. Jagersand, "AMP: Adaptive masked proxies for few-shot segmentation." In *Proc. of the IEEE International Conference on Computer Vision*, 2019, pp. 5248–5257. DOI: 10.1109/ICCV.2019.00535.

[137] J. Bekker and J. Davis, *Learning from positive and unlabeled data: a survey*, 4. Springer US, 2020, vol. 109, pp. 719–760. DOI: 10.1007/s10994-020-05877-5.

[138] Y. Yang, K. J. Liang, and L. Carin, "Object detection as a positive-unlabeled problem," Tech. Rep., 2020. arXiv: 2002.04672. [Online]. Available: https://arxiv.org/pdf/2002.04672.pdf.

[139] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009. DOI: 10.1016/j.patrec.2008.04.005.

[140] A. Valada, G. L. Oliveira, T. Brox, and W. Burgard, "Deep Multispectral Semantic Scene Understanding of Forested Environments Using Multimodal Fusion," in *International Symposium for Experimental Robotics*, 2017, pp. 465–477. DOI: 10.1007/978-3-319-50115-4_41.

[141] D. P. Kingma and J. Lei Ba, "Adam: A Method for Stochastic Optimization." In *Proc. of the International Conference on Learning Representations*, 2015.

[142] A. Zlateski, R. Jaroensri, P. Sharma, and F. Durand, "On the Importance of Label Quality for Semantic Segmentation." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2018, pp. 1479–1487. DOI: 10.1109/CVPR.2018.00160.

[143] X. Liu and S. Zhang, "Who is closer: A computational method for domain gap evaluation," *Pattern Recognition*, vol. 122, p. 108 293, 2022. DOI: 10.1016/j.patcog.2021.108293.

[144] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019. DOI: 10.1002/rob.21831.

[145] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury Google, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. K. Xamla, E. Yang, Z. Devito, M. Raison Nabla, A. Tejani, S. Chilamkurthy, Q. Ai, B. Steiner, L. F. Facebook, J. B. Facebook, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In *Proc. of the Conference on Neural Information Processing Systems*, 2019, pp. 8024–8035.

[146] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks." In *Proc. of the IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2017, pp. 464–472. DOI: 10.1109/WACV.2017.58.

[147] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System." In *Proc. of the IEEE International Conference on Robotics and Automation Workshop on Open Source Robotics*, 2009.

[148] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021. DOI: 10.1109/TPAMI.2019.2929257.

[149] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2019, pp. 4685–4694. DOI: 10.1109/CVPR.2019.00482.

[150] U. Michieli and P. Zanuttigh, "Knowledge distillation for incremental learning in semantic segmentation," *Computer Vision and Image Understanding*, vol. 205, p. 103 167, 2021. DOI: 10.1016/j.cviu.2021.103167.

[151] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations." In *Proc. of the International Conference on Machine Learning*, vol. PartF16814, 2020, pp. 1575–1585.

[152] X. Lai, Z. Tian, L. Jiang, S. Liu, H. Zhao, L. Wang, and J. Jia, "Semi-supervised Semantic Segmentation with Directional Context-aware Consistency." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2021, pp. 1205–1214. DOI: 10.1109/CVPR46437.2021.00126.

[153] R. C. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.” In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16.

[154] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds.” In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2020, pp. 11 105–11 114. DOI: 10.1109/CVPR42600.2020.01112.

[155] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, “PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation.” In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2020, pp. 9598–9607. DOI: 10.1109/CVPR42600.2020.00962.

[156] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[157] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient Graph-Based Image Segmentation,” *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004. DOI: 10.1023/B:VISI.0000022288.19776.77.

[158] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, “scikit-image: image processing in Python,” *PeerJ*, vol. 2, no. 1, e453, 2014. DOI: 10.7717/peerj.453.

[159] T. Leung and J. Malik, “Representing and recognizing the visual appearance of materials using three-dimensional textons,” *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001. DOI: 10.1023/A:1011126920638.

[160] P. Mishra and K. Sarawadekar, “Polynomial Learning Rate Policy with Warm Restart for Deep Neural Network.” In *Proc. of the IEEE Region 10 Conference*, IEEE, 2019, pp. 2087–2092. DOI: 10.1109/TENCON.2019.8929465.

[161] Z. Zheng and Y. Yang, “Unsupervised Scene Adaptation with Memory Regularization in vivo.” In *Proc. of the International Joint Conference on Artificial Intelligence*, California: International Joint Conferences on Artificial Intelligence Organization, 2020, pp. 1076–1082. DOI: 10.24963/ijcai.2020/150.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Jun Miura for his support throughout my BSc, MSc, and Ph.D study. I would like to thank him for giving me a lot of advices and suggestions about the research, and for not telling me what to do, which encouraged me to proactively carry out the research. Without his support and patience, I would not be able to continue my research and submit this thesis today.

I would like to express my gratitude to Prof. Shigeru Kuriyama, an examiner of the dissertation, for the fruitful discussion and suggestions.

I also express my gratitude to Prof. Naoki Uchiyama, Prof. Takashi Tsubouchi (University of Tsukuba), and Prof. Mikko Vastaranta (University of Eastern Finland: UEF) for the fruitful discussions, suggestions, and advices. In addition, I would like to thank Prof. Vastaranta also for giving me the chance to study in UEF. The experience of studying in a different field gave me a new perspective and knowledge, which will help me in the future research life.

I would like to express my gratitude to Specially Appointed Assoc. Prof. Takahiro Yamauchi for his cooperation in conducting data collection and the robot navigation experiments in the greenhouse.

Special thanks go to Mr. Hiroaki Masuzawa, who co-authored all my publications. He did not only support my research and gave me advice as a researcher, but also gave me a lot of advices about the life and career as a senior in the lab. I would also like to thank Assist. Prof. Kotaro Hayashi for giving me advice about the research career and also sometimes talking together. My gratitude also goes to the secretaries of the Active Intelligent Systems Laboratory (AISL): Ms. Mikiko Kobayashi, Ms. Asaha Murai, and a former secretary Ms. Akiko Yamamoto for their support in my research activities, and for talking with me about random things.

I thank all members and alumni of AISL for their cooperation, discussion, and fun we have had. Especially, I am grateful to the members of the Agri group: Chuo Nakano, Yoshinobu Uzawa, Tomohiko Sano, David Pich, and Hiroto Kakinoki for the discussion in the weekly meetings, and being good *kouhai* (junior students). I would also like to thank Kazuki Mano who has been a good friend and a research mate since B4. Besides, thanks also go to the staffs and students in the Leading Program: Prof. Hiromu Ishii, Mr. Hajime Tsukada, Ms. Chiharu Iwase, Yuma Taniyama, Ryuto Shinozaki, junior students, and alumni.

Dr. Shuji Oishi and Dr. Kenji Koide (National Institute of Advanced Industrial Science and Technology: AIST) have been and will be my role models since they were members of AISL, and were a motivation of my crazy decision to go to the Ph.D course. Together with them, I would like to thank Dr. Masashi Yokozuka and Dr. Atsuhiko Banno for the opportunity of the research internship at AIST, and for the discussion and the advices during the internship. I would also like to thank the Ph.D students who have influenced, motivated, and sometimes demotivated me by showing their strong progresses and abilities: Dr. Liliana Villamar Gomez, Dr. Chandra Kusuma Dewa, Dr. Yubao Liu, Oskar Natan, Mahmood Hassan, Kohei Honda, and friends on Twitter.

Last but not least, I would like to thank my family for their kind support. I would like to express sincere gratitude to my beloved wife, Ms. Min Matsuzaki, for her love and patience.

The work in this thesis was supported in part by the Leading Graduate School Program: "the Innovative Program for Training Brain-Science-Information-Architects by Analysis of Massive Quantities of Highly Technical Information About the Brain" by the Ministry of Education, Culture, Sports, Science and Technology, Japan.

List of Publications

Publications relevant to this thesis

- **Shigemichi Matsuzaki**, Jun Miura, and Hiroaki Masuzawa, "Multi-source Pseudo-label Learning of Semantic Segmentation for the Scene Recognition of Agricultural Mobile Robots", Advanced Robotics, pp. 1011-1029, vol. 36, issue 19, 2022, DOI: 10.1080/01691864.2022.2109427
- **Shigemichi Matsuzaki**, Hiroaki Masuzawa, and Jun Miura, "Image-based scene recognition for robot navigation considering traversable plants and its manual annotation-free training", IEEE Access, pp. 5115-5128, vol. 10, 2022, DOI: 10.1109/ACCESS.2022.3141594
- **Shigemichi Matsuzaki**, Hiroaki Masuzawa, and Jun Miura, "Online refinement of a scene recognition model for mobile robots by observing human's interaction with environments", IEEE International Conference on Systems, Man, and Cybernetics, pp. 216-222, Prague, the Czech Republic, Oct. 2022, DOI: 10.1109/SMC53654.2022.9945283
- **Shigemichi Matsuzaki**, Hiroaki Masuzawa, Jun Miura, and Shuji Oishi, "3D Semantic Mapping in Greehouses for Agricultural Robots with Robust Object Recognition using Robots' Trajectory", IEEE International Conference on Systems, Man, and Cybernetics, pp. 357-362, Miyazaki, Japan, Oct. 2018, DOI: 10.1109/SMC.2018.00070

Publications not forming this thesis

- Yoshinobu Uzawa, **Shigemichi Matsuzaki**, Hiroaki Masuzawa, and Jun Miura, "End-to-end path estimation and automatic dataset generation for robot navigation in plant-rich environments", 17th International Conference on Autonomous Systems, Zagreb, Croatia, June 2022.