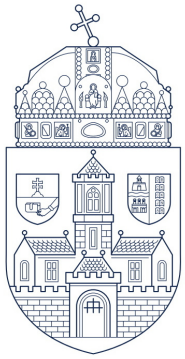
Óbuda University

PhD Thesis



New Deep Neural Network Applications in Robot Control and System Supervision

Mély Neurális Hálózatok Új Alkalmazásai
a Robotirányítás és a Rendszer-felügyelet Terén

Artúr István Károly

Supervisor: Dr. Péter Galambos

Doctoral School of Applied Informatics and Applied Mathematics

Budapest, 2023

ABSTRACT

Various fields, such as image and natural language processing and speech and image synthesis, among others, have seen great success with deep learning approaches. As a result, more and more scientific fields are adopting and leveraging these successful methods. Robotics is no exception, and it is currently experiencing a surge of new technologies and innovations inspired by deep learning results. However, the requirements of robotics solutions are unique, and addressing the drawbacks of deep learning approaches, such as the need for a large training dataset, time-consuming training and data preparation, high computational complexity, and limited generalization power, are necessary.

The focus of this dissertation is to provide solutions for some of these challenges. This is first done by investigating the potential of unsupervised learning technologies in detecting anomalies and identifying states in robotic systems in an online fashion. The dissertation also delves into transfer learning approaches and proposes a technique that integrates non-RGB data modalities (such as optical flow) into a transfer learning pipeline, utilizing RGB pretrained feature extractors. Furthermore, the dissertation explores automated dataset generation and annotation techniques and presents two methods: one for automatically annotating real-life visual data for robotic manipulation and the other for generating synthetic image datasets and automatically annotating them for object segmentation tasks in robotic manipulation.

KIVONAT

Számos tudományterület, köztük a képfeldolgozás, a természetes (emberi) nyelvek feldolgozása, kép és beszéd szintézis stb., jelentős fellendülést ért el a mély tanulás új eredményeinek köszönhetően. Következésképpen, egyre több tudományterület adaptálja és aknázza ki ezeknek az új eljárásoknak az előnyeit. Ez alól a robotika sem kivétel, ami jelenleg újszerű és innovatív megoldások áradatát élvezi, melyeket a mély tanulási eljárások inspiráltak/tettek lehetővé. Azonban, a robotikában a megoldások speciális kihívások elé vannak állítva, amik megkövetelik a mély tanulási módszerek hátrányainak (nagy mennyiségű tanító adathalmaz szükségessége, időigényes tanítási és adat előkészítési folyamat, nagy számítási kapacitás, korlátozott általánosítóképesség stb.) kiküszöbölését.

Ez a disszertáció ezen kihívások megoldására fókuszál. Az első téziscsoport a felügyelet nélküli tanítási módszerek lehetőségeit vizsgálja, az online állapotfelismerés és anomália detekció terén, a robotikában. A második téziscsoport a transfer learning témakörrel foglalkozik, és egy olyan eljárást mutat be, ami lehetővé teszi az RGB képeken előtanított neurális hálózatok használatát olyan transfer learning feladatokban, ahol a bemenő adatok nem feltétlenül RGB modalitásúak. A harmadik téziscsoport az automatikus adathalmaz generálás és annotáció módszereit vizsgálja. Itt két eljárás kerül bevezetésre, az egyik valós képi adathalmazok automatikus annotációjával foglalkozik, a robotkarral történő manipuláció kontextusában, a másik ugyanebben a kontextusban, szintetikus képi adathalmazok generálását és azok automatikus annotációját teszi lehetővé.

DECLARATION

Undersigned, Artúr István Károly, hereby I state that this Ph.D. thesis is my own work, wherein I only used the sources listed in the references. All parts taken from other works, either as word-for-word citations or rewritten, keeping the original meaning, have been unambiguously marked, and a reference to the source was included.

NYILATKOZAT

Alulírott Károly István Artúr kijelentem, hogy ezt a doktori értekezést önállóan készítettem, és abban csak az irodalmi hivatkozások listájában szereplő forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, November 22, 2023

.....

Artúr István Károly

Contents

Acknowledgment

I would like to express my sincere gratitude to my supervisor, Dr. Péter Galambos, for his guidance, encouragement, and invaluable feedback throughout the course of this research. His extensive knowledge and expertise have been a constant source of inspiration for me.

I would also like to extend my heartfelt thanks to my colleagues in ABC-iRob, for their support, encouragement, and assistance throughout my research project. Their valuable contributions have helped me in many ways to complete this work.

In addition, I would like to acknowledge the support of my family, who have always stood by me and provided me with the motivation and strength to pursue my goals.

Thank you all for your invaluable contributions and support throughout this research project.

Structure of the Thesis

The Thesis is divided into five parts. Part 1. provides an overview of the historical background, motivations, and necessary preliminary knowledge required for the thesis work. Parts 2. through 4. present the new scientific results, each comprising two major sections that focus on different, related scientific achievements. The sections follow a similar structure, beginning with an explanation of the theoretical background, problem formulation, and proposed approach, followed by a description of the experimental setup and methodology and the results and evaluation. The thesis statements summarizing the new scientific results are presented at the end of each part.

Part 2. introduces an online clustering algorithm for automatic state discovery and anomaly detection in robotic applications using One-Class Support Vector Machines (OCSVMs). The first major section describes the algorithm in detail and evaluates its performance on a real-life robot setup. The second major section shows how the approach can also be used for the evaluation of generative deep learning models. Part 3. presents a method called cross-modal mapping, which enables the use of non-RGB data modalities (such as optical flow) with feature extractors pre-trained on RGB data in a transfer learning scenario. The first major section demonstrates how this method can train deep learning models with transfer learning, using RGB pre-trained feature extractors and optical flow data as input. The second section describes a loss function and training strategy proposed to address class imbalance during experiments. Part 4. outlines two automated dataset generation and annotation pipelines. The first method describes an approach for automatically collecting and annotating real-life visual object segmentation datasets for robotic manipulation, and the second section presents a method for generating and annotating synthetic visual object segmentation datasets for robotic manipulation. Finally, Part 5. summarizes the contributions of the Thesis and outlines potential future research directions.

Notations and Symbols

TABLE 1
COMMON ABBREVIATIONS AND NOTATIONS

ABC-iRob	Antal Bejczy Center for Intelligent Robotics
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
API	Application Programming Interface
BAT	Blender Annotation Tool
BWT	Backward Transfer
CNN	Convolutional Neural Network
COCO	Common Objects in Context dataset
DAVIS	Densely Annotated Video Segmentation dataset
Dex-Net	Dexterity Network
DL	Deep Learning
DOF	Degrees of Freedom
FOV	Field of View
FPS	Frames Per Second
FTRG	Filling the Reality Gap
FWT	Forward Transfer
GAN	Generative Adversarial Networks
GQCNN	Grasp Quality Convolutional Neural Network
GUI	Graphical User Interface
HDRI	High Dynamic Range Image
HRI	Human-Robot Interaction
IoU	Intersection over Union
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute dataset
mAP	mean Average Precision
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology dataset
MTC	Moveit! Task Constructor framework
OCSVM	One-Class Support Vector Machine
OFSNet	Optical Flow Segmentation Network
PASCAL VOC	PASCAL Visual Object Classes dataset
PNG	Portable Network Graphic
RBF	Radial Basis Function
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RoI	Region of Interest
ROS	Robot Operating System
RPN	Region Proposal Network
STL	Standard Triangle Language
SVC	Support Vector Clustering
SVM	Support Vector Machine
TCP	Tool Center Point
U-Net	A fully convolutional neural network called U-Net
uNLC	unsupervised Non-Local Consensus voting method
VGG	A deep convolutional neural network model by the Visual Geometry Group at Oxford University
YOLO	A deep convolutional neural network model named You Only Look Once

List of Figures

List of Tables

Part 1

PRELIMINARIES

1.1 Historical background

Computers can easily solve formal problems that are challenging for humans. However, with the increasing demand for adaptive systems, there is a need to tackle tasks that are hard to formulate but can be easily handled by humans, such as the recognition and manipulation of objects. These tasks require a complex understanding of the environment. The automatic extraction of this required knowledge is called Machine Learning (ML). The way the data is presented to the ML system (feature representation) heavily influences how well the extracted knowledge represents the given problem. Deep Learning (DL) refers to ML approaches that extract features from raw data using multiple hierarchical trainable artificial neural network layers [5, 6]. Consequently, DL techniques have the ability to develop feature representations that are most suitable for specific tasks.

The theoretical foundation of DL had been established long before it gained widespread recognition, thanks in part to the winning entry in the ImageNet Challenge 2012 (LSVRC-2012) [7]. Krizhevsky, Sutskever, and Hinton introduced a deep convolutional neural network, now known as AlexNet, which surpassed previous state-of-the-art methods and outperformed other competitors, achieving a top-5 test error rate of 15.3%, more than 10 percentage points better than the second-best entry. Following this success, DL has been widely adopted for a variety of use cases, including speech recognition [8, 9, 10], image processing [11, 12, 13, 14], natural language processing [15, 16, 17], sentiment analysis and recommendation systems [18, 19], among others. Numerous major companies such as Google, Facebook, Amazon, and IBM have also established their own DL research teams.

The fields of machine learning and AI are gradually making their presence felt across a wide range of industries, subtly impacting our daily lives as well as professional services and military applications. The manufacturing industry, in particular, stands to benefit significantly from these advancements, with numerous research projects exploring the potential of Industry 4.0 concepts with great fervor.

DL has introduced innovative advancements in various aspects of robotics throughout its development. A general overview of the DL methods frequently used in robotics can be found in [20], while other surveys provide a more focused review of the most notable results, such as robot control through reinforcement learning [21, 22], robotic manipulation and grasping [22, 23], mobile robot navigation [23], and transfer learning for robotics [23, 24]. Despite the fact that DL-based solutions can be applied to a diverse range of

problems, they have the disadvantages of unpredictability and high computational complexity. In safety-critical systems, like self-driving cars and industrial robots, DL methods are never used on their own, and their output is always treated with uncertainty. As a result, these DL methods are tested against adversarial attacks and on benchmarks that assess their robustness. Due to the high computational complexity and time-consuming training process of DL systems, alternative model architectures, and corresponding training strategies have been introduced. For example, Pao et al. proposed the Random Vector Functional Link Neural Networks (RVFLNN) [25, 26], which utilize a novel training strategy for a shallow network architecture to avoid the time-consuming training process typical of DL systems. Building on the RVFLNN method, Chen et al. introduced Broad Learning Systems, which offer an incremental training strategy for the rapid adaptation and retraining of such models [27, 28]. New approaches in the field of DL have been proposed to address these issues, often taking advantage of the modular nature of DL models through an appropriate model structure and/or training strategy [29]. In addition to the training process, data collection and preparation for DL require a significant amount of resources, especially if done manually. This issue is particularly relevant in robotics, where data collection often involves performing actions on an actual robot. Such data collection can take months of robot hours and requires multiple robots, resulting in significant costs [30]. To minimize the amount of labeled data required for training, new DL approaches are utilizing unsupervised/semi-supervised methods and transfer learning [31, 32, 33]. Moreover, some approaches aim to reduce resource requirements by conducting data collection primarily in simulation instead of reality [34, 35].

The role of DL-based approaches in robotics research can be better understood by considering the major challenges in robotics and the corresponding DL solutions. Figure 1.1. provides a structured overview of these challenges, organizing the landscape into three main categories: perception, motion, and knowledge adaptation.

This work focuses primarily on perception-level difficulties, including object detection, segmentation, and other related tasks essential for robotic manipulation and mobile robot navigation. While previous studies have indicated that DL methods can provide promising results, they also demonstrate significant limitations, such as high computational complexity and the requirement of extensive training data. Given the importance of resource efficiency in robotics, it becomes crucial to search for improved DL-based solutions that can overcome these challenges. To address these issues, this dissertation explores unsupervised learning techniques, transfer learning, and automated dataset generation methodologies.

1.2 Unsupervised learning

Unsupervised learning techniques play a critical role in many deep learning approaches due to the high data demands of training complex models with numerous parameters. Fully supervised training requires a large number of labeled examples, which is a time-consuming and resource-intensive process to prepare. Therefore, researchers seek learning approaches that require fewer or no labeled examples [36]. Unsupervised learning can extract valuable information about data structure and hierarchies using only the data samples without requiring ground truth labels. The extracted knowledge representation can serve

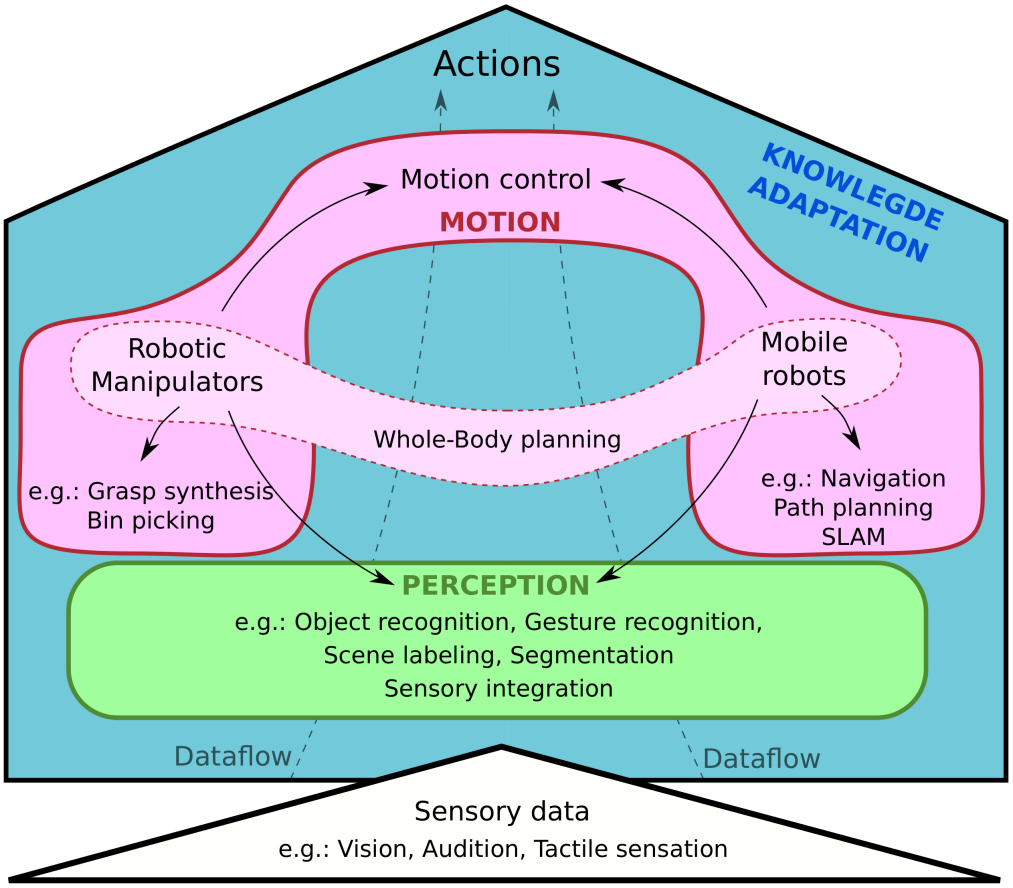


Fig. 1.1. Robotics challenges for DL

as a foundation for DL models that will require fewer labeled examples, as they already have a good understanding of the data's underlying nature and only need fine-tuning for specific tasks, or it can be used for clustering and anomaly detection.

The vast amount of data generated by modern manufacturing processes provides us with an abundance of information that can be distilled and utilized to develop effective interventions. While the acquisition and storage of raw data is no longer a major challenge, there is a pressing need for solutions that can effectively explore correlations and make real-time decisions. Consequently, numerous recent studies are focusing on developing online methods, including the reformulation of techniques originally designed for analyzing historical data [37, 38, 39]. Given the impracticality of conducting real-time manual data labeling, these techniques inherently necessitate an unsupervised approach.

Clustering, a kind of unsupervised learning, involves grouping data samples based on similarity [40]. It can be used for uncovering hidden patterns, structures, or relationships in the data. Unlike classification, clustering is an unsupervised process, meaning that it does not involve the use of class labels, and sometimes the number of clusters is not known in advance. According to [40], the similarity measure used in clustering can be expressed as a distance measure. This is because the similarity between two samples can be viewed as the distance between them in the feature space.

One approach for clustering is to use these similarity measures and construct the re-

gions of the feature space corresponding to the different clusters based on the computed distances between the training samples [40].

An alternative method involves identifying features that have a high ability to differentiate between samples or discovering principal directions within the feature space that allow for improved separation of data. These discriminating features can be derived from any subset of the input data, or they can be generated using a network architecture, as noted in [41].

In some cases, it may be more practical to utilize the computational mechanism of a supervised learning algorithm, even if no actual labels are available. Anomaly detection is one such example that can be approached as a clustering problem, whereby two clusters can be expected - one for normal samples and another for anomalies. In an unsupervised training process for anomaly detection, only training data from the normal sample cluster is extracted and, as a result, can be automatically labeled as belonging to the same class. The supervised method's computation can then be used to train a system to recognize similar samples as those belonging to this normal cluster. However, when an anomaly appears that is dissimilar to the training examples, it will be classified as a sample not belonging to the normal sample cluster and thus deemed an anomaly, as outlined in [KA2].

This idea is employed in the work presented in the first thesis group (Part 2.), utilizing the Support Vector Machine (SVM) technique for unsupervised online state discovery and anomaly detection in robot applications.

1.2.1 Support vector machines

Support Vector Clustering (SVC), which is typically implemented using support vector machines, is thoroughly explained in [42]. This method is built on the foundation of the work conducted by Schölkopf et al. [43], and Tax and Dunin [44], who pioneered the use of kernel functions [45] for support vector description [46] of data structures in a high-dimensional space.

The kernel function allows for the formulation of a distance measure in a higher-dimensional space, as opposed to directly separating data samples in the feature space. By utilizing this kernel function, the separation of data samples can be designed, as described in [45], resulting in highly complex and nonlinear decision boundaries within the feature space.

When utilizing SVC, the training samples are transformed into a high-dimensional space using a Gaussian kernel function. In this high-dimensional space, the data is encompassed within a hypersphere with a center of $\mathbf{a}$ and a radius of R. To regulate the number of outliers allowed, a penalty parameter is introduced. An outlier is defined as a sample for which $|\Phi(\mathbf{x}) - \mathbf{a}|_2^2 > R^2 + \xi$, where $\Phi(.)$ is the kernel function that maps the sample $\mathbf{x}$ from the feature space to the high-dimensional space, and ξ is a slack variable that allows for a soft margin, as detailed in [42].

The contour of the hypersphere creates boundaries within the feature space, which separates points that fall on the inside of the hypersphere when mapped to the high-dimensional space from those that fall on the outside. The shape of this decision boundary is influenced by the kernel function's parameters and the penalty coefficient for outliers, as noted in [42]. The appropriate adjustment of these parameters relies on the noise and

structural overlap within the data and is discussed in detail in [42]. If the parameters are correctly set, smooth, disjoint boundaries should manifest within the feature space.

The clusters are defined by the non-overlapping sets in the data space [42]. As a result, if any path connecting two points $\mathbf{x}_1$ and $\mathbf{x}_2$ in the feature space exits the hypersphere in the high-dimensional space, those two points are considered to belong to different clusters. In [42], this criterion is evaluated numerically by checking twenty points along the connecting line between $\mathbf{x}_1$ and $\mathbf{x}_2$.

A simpler way to perform clustering with support vector machines is to use the One-Class Support Vector Machine (OCSVM) [43, 44], which is the basis of the SVC algorithm. In cases where only two clusters are anticipated, such as anomaly detection, the cluster assignment method suggested in [42] may not be necessary, thereby simplifying the system.

The proposed unsupervised online algorithm for state discovery and anomaly detection in robotic applications, described in the first thesis group (Part 2.), also utilizes OCSVM models.

1.3 Transfer learning for Deep learning

Training a DL model from scratch is a resource-intensive process in terms of time and data requirements. If a DL-based approach provides a suitable solution for a robot application, it should be designed with adaptability in mind, allowing it to be applied to similar problems. This adaptability or robustness of the model always has to be considered, and thus, it is represented in the background layer in Figure 1.1. displaying it as an underlying requirement that is present at all levels of challenges. Transfer learning is a field that focuses on utilizing knowledge gained from solving previous problems to accelerate the training process or improve the performance of new solutions [47]. A problem can be defined by a domain $\mathcal{D}$ and a task $\mathcal{T}$ [47]. The domain is a combination of a feature space $\mathcal{X}$ and a corresponding marginal probability distribution $P(X)$, where $X = \{x_1...x_n\} \in \mathcal{X}$ [47]. Given a specific domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task can be described by $\mathcal{T} = \{\mathcal{Y}, f(.)\}$, where $\mathcal{Y}$ is a label space and $f(.)$ is a function that should be learned based on the training data, which consists of $\{x_i, y_i\}$ pairs, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ [47]. Two problems are considered different if either their domains or their tasks are different.

Transfer learning entails discovering and applying abstract knowledge to new problems. This abstract knowledge can take various forms, such as an abstract kinematic or dynamic model, a set of rules or laws, a formula or function, an algorithm, a policy, and so on. By utilizing this knowledge, we can develop new solutions for similar problems. For example, when dealing with the forward kinematics task for a given robot arm, this knowledge could be represented by an abstract robot model or a set of rules for creating one, as well as a method for solving the forward kinematics problem given a specific robot model in a predefined format. With this knowledge, we can construct new solutions for the forward kinematics problem with different manipulators. However, for some problems, it may not be possible or practical to derive the analytic formulation of such knowledge. In such cases, DL methods can be used to approximate the abstract knowledge [47].

In robotics, the primary transfer learning techniques include utilizing pre-trained DL models, sim-to-real approaches, extracting domain-invariant features, and learning through

imitation or demonstrations. These methods have proven to be effective in facilitating knowledge transfer from prior tasks to new problems [47].

The second thesis group (Part 3.) explores pre-trained model-based transfer learning approaches for domains with optical flow, a non-RGB data modality for mobile robot navigation.

1.3.1 Pre-trained models

Pre-trained models are a popular technique for transfer learning, where they are used as general feature extractors [48, 49]. They represent a general knowledge base required for a particular field of problems and are utilized to initialize the DL-based solution for a new specific problem. During training, the pre-trained models may be fine-tuned, or a smaller learning-based model may be trained from scratch to process the features extracted by the pre-trained model. The use of pre-trained models is most widespread in approaches for image processing, utilizing deep Convolutional Neural Networks (CNNs). For instance, various publicly available DL models have been pre-trained on the ImageNet dataset [50], including AlexNet [7], GoogleNet [51], the VGG model [52], and others. These models are highly effective in extracting high-level features from image data for a wide range of image-processing tasks.

1.3.2 Sim-to-real

A large portion of learning-based solutions for robotic manipulation and navigation rely on a trial-and-error approach. However, conducting such approaches directly on the physical robot is often not feasible due to concerns about safety, cost, and time constraints [53, 54]. Due to these challenges, there is a growing need to separate the solution preparation process from the physical world. One common approach to address this issue is to develop the solution in a simulated environment and subsequently apply it to real-world problems.

Preparing solutions in a simulated environment offers several advantages, including the availability of complete and precise information about the environment, as well as the rapid, decoupled computation and preparation of different environmental scenarios. However, a significant challenge is to adapt the obtained solution to the real-world problem to leverage these benefits. This issue is commonly referred to as bridging the "reality gap" [34, 35]. There are two primary approaches for addressing this challenge: randomization and the creation of simulations and data generation that is as close to reality as possible [34, 55, 35, 56, 57, 58].

Sim-to-real approaches that employ randomization are based on the concept that if a model can perform well in a highly randomized source domain, it is likely to have excellent generalization capabilities, which means it should be able to adapt to the real-world domain as well [34, 55]. The other approach for bridging the reality gap, which involves creating a simulation environment that closely resembles the real world, allows the trained DL model to be directly applied to the real-world problem [35, 57, 58].

1.3.3 Domain-invariant features

It is a common transfer learning scenario when a solution for a given problem with a specific domain exists, and a new solution for a new problem with the same task but a different domain is needed. Some of these approaches seek to identify features independent of a particular domain [18, 59]. These domain-independent features can be extracted by utilizing data from multiple domains $(\mathcal{D}1, \mathcal{D}2, ...)$. For instance, let $\mathcal{X}_1$ and $\mathcal{X}_2$ be the feature spaces of two domains, with features $\mathcal{X}_1 = \mathbf{x}_{11}, \mathbf{x}_{12}, ...$ and $\mathcal{X}_2 = \mathbf{x}_{21}, \mathbf{x}_{22}, ...$, respectively. When $\mathbf{x}_{1i}$ and $\mathbf{x}_{2j}$ are identical, and their marginal probability distributions in the two domains $(\mathcal{D}_1, \mathcal{D}_2)$ are similar, they are considered common features of the two domains. As an example, for datasets of images of objects to be grasped, common features could be the shapes that facilitate grasping the objects with a specific gripper configuration. Common features can often be more complex than simple, general low-level features (e.g., complex shapes and textures) [48, 49], and are often challenging to find using analytic methods. As such, DL is typically used to extract these common features.

1.3.4 Imitation and demonstration-based learning

Imitation learning, a learning-based method that generates solutions based on demonstrations, has been successfully employed in robotic manipulation skills [24, 60, 61, 62, 63]. In this approach, a demonstration is used as the solution for the source problem, with the aim of learning a solution for the target problem. The differences between the source and target problems may be due to the differing embodiment of agents. For example, a task of robotic manipulation can be taught to a robot by providing a demonstration of a human performing the task and requiring the robot to perform the same task [60].

Many approaches assume that the demonstration provides a close-to-optimal solution to the source problem, and therefore the performance of the solution for the target problem is measured by comparing it to the demonstration [62, 63, 64]. To achieve this, a state description is necessary. A solution for the target problem is described by a sequence of states that need to be similar to the demonstration states [62, 63]. To ensure comparability between the source and target problems, the state description must be independent of the problem type, which can be assured with the help of common features. Adapting the source solution to the target solution is also known as skill transfer or skill extraction.

To simplify the demonstration learning scenario, the source problem is typically defined in a way that is very similar to the target problem. This involves providing demonstrations through teleoperation rather than tracking the hand of a human performing the task. Human hands have different kinematic structures than conventional robot manipulators, and tracking them can introduce errors to the system. On the other hand, demonstrations provided through teleoperation are very similar to the target problem and can be used as starting points for further exploration [63]. This approach also allows for the collection of relevant internal data from the demonstration, such as forces, torques, and velocities, which can be measured with high accuracy, contrary to visual tracking. In such scenarios, the measurement errors are mainly due to the physical construction of the manipulator, which is the same during the demonstrations and during operation. Consequently, in demonstration learning tasks, usually, other modalities are utilized besides vision, such as tactile information [65].

1.3.5 Pre-trained models and modularity for perception

Visual modality is rich in information, and thus, it is often used in robot applications. However, visual data is often characterized by high-dimensional feature vectors that pose challenges during analysis and processing. Therefore, rather than training new DL solutions from scratch on large datasets, pre-trained image processing models like AlexNet, GoogleNet, and VGG networks are also often utilized in robot applications. Girshick et al. demonstrated that these pre-trained models' features could be used for visual object detection in their R-CNN (Region-based Convolutional Neural Network) proposal [66]. They later introduced the Fast R-CNN method, which significantly improves the R-CNN's training and inference speed [67]. In R-CNN, the pre-trained model extracts features from regions of interest (RoI) in input images, and object-specific support vector machines (SVMs) classify the extracted feature vectors. The RoIs are determined by a region proposal method. In contrast, Fast R-CNN infers the output from the feature maps of RoIs using fully connected neural network layers. Girshick showed that Fast R-CNN performs more accurate and faster object detection than R-CNN (9 times faster training and 213 times faster inference). This is mainly due to the model architecture, which allows the prediction layers to be optimized together with the feature extractor during a single-stage training procedure.

In their study, Ren et al. developed a region proposal network (RPN) using deep learning to improve the Fast R-CNN object detection method. To enhance the Fast R-CNN structure, they integrated the RPN using an attentional interface and coined it Faster R-CNN [68]. The authors reported that their approach achieved real-time object detection and outperformed other region proposal methods with Fast R-CNN. They were able to conduct inference at 17 frames per second (FPS), which was superior to previous methods.

In their research, Johnson, Karpathy, and Li introduced a fully convolutional localization network named DenseCap by using the Faster R-CNN method to propose regions for a recurrent neural network that produces complex image captions [17]. Valipour et al . utilized this model in their incremental learning scenario and combined it with the VGG-16 pre-trained model for object detection and localization [69]. They developed a DL-based perception pipeline that enabled a robot to recognize and manipulate various objects requested by a human operator through speech recognition using the CMU Sphinx module. Additionally, the team incorporated the Festival speech synthesis system [70] to enable the robot to provide feedback on its current state verbally. The researchers also created a method for correcting the localization network's classifications incrementally through HRI, utilizing speech recognition, and human gesture recognition systems. Their study highlights how different DL modules can be integrated to solve a complex robotics problem involving object detection and HRI.

Redmon et al. proposed a novel approach to object detection with their YOLO (You Only Look Once) network [71]. Unlike R-CNN and its variants, YOLO does not require separate region proposal networks; instead, it directly infers bounding boxes and class probabilities from globally extracted features. This two-part network comprises a CNN for feature extraction and a smaller neural network for classification and regression. This approach allows for end-to-end training while still leveraging the modular nature of DL models. The feature extractor and top network can be separated and are responsible

for different subtasks. Moreover, YOLO infers predictions globally, based on the whole image, instead of extracting local features. Redmon et al. also introduced a feature extractor architecture for YOLO, inspired by GoogleNet, which was pre-trained on ImageNet, achieving similar performance to the 2012 GoogleNet model. During the evaluation, their Fast YOLO model achieved an inference speed of 155 FPS with a mean average precision (mAP) of 52.7, outperforming the deformable parts model (DPM) and the VGG-16-based YOLO. Although VGG-16-based Faster R-CNN achieved the highest mAP (73.2), it was slower than YOLO, and the YOLO model demonstrated better generalization to artwork and natural images from the internet. It is crucial to note that the YOLO model retains modularity, allowing for resource-efficient transfer learning using pre-trained feature extractors rather than training the entire network from scratch.

The YOLO model has a particular relevance to the field of robotics because the prediction process of the bounding boxes for its object detection is based on the MultiGrasp system proposed by Redmon et al. which was originally designed for 2D grasp detection for robotic manipulation based on RGB image data [72].

The work in the second thesis group (Part 3.) is based on another DL model that utilizes pre-trained models for feature extraction, namely the U-Net and its variants [73]. The experiments for the third thesis group (Part 4.) were carried out with the help of R-CNN variant DL models, the Mask R-CNN [74].

1.3.6 Multimodal data and unsupervised pre-training for perception

In robotics, depth data plays a crucial role alongside visual information. However, unlike RGB data, pre-trained models for depth and point cloud information are not as abundant, as seen in the limited availability of models such as PointNet [75]. As a result, creative approaches are often required to handle this modality effectively.

Estimating the pose of an object is crucial in predicting a grasp. Zeng et al. proposed a fully convolutional neural network segmentation method for object 6D pose estimation using multi-view RGB-D sensors [76]. In this method, 2D images are segmented using a segmentation network based on the pre-trained VGG architecture, and a segmented 3D point cloud is constructed based on the segmentations. The former scanned objects are then fit onto the segmented scene to estimate the object poses. Using this method, the authors developed a pose estimator model for multiple objects with occlusion in a cluttered environment. The output of the network is a dense probability map for each of the given object labels, with values for every pixel of the image. The probability maps are then thresholded to construct the 3D segmented point cloud. This approach achieved third and fourth place in the 2016 Amazon Picking Challenge.

To process multiple modalities, a possible approach is to use separate convolutional structures and fuse them into a common layer at a higher level of the deep learning model, as demonstrated in [77]. Another approach is to encode depth data into color images and process them using convolutional neural networks pre-trained on RGB images, as proposed by Schwarz et al. [78]. This method performs classification and regression based on features extracted from both the images and the converted depth data. When compared with the method of Bo et al. [79], which used unsupervised learning of hierarchical feature representations from RGB-D objects, the approach of Schwarz et al. was able to learn from less data and achieved better accuracy for RGBD data. Moreover, the RGB-based

classification accuracy was also comparable, with a 92% accuracy compared to 92.1% for the method of Bo et al.

The method that Bo et al. utilized, the unsupervised pre-training of the feature extractor, is another popular approach for learning meaningful and representative features from inputs. This approach has been widely used in various applications such as RGB-D recognition [79], tactile object recognition [80], and robotic grasping [81]. Typically, an autoencoder structure is employed to extract features from inputs by minimizing the reconstruction error. This approach is particularly useful in problems where labeled datasets are not available for training. Schmitz et al. utilized a denoising autoencoder for pre-training their model in tactile object recognition [80]. They reported that the unsupervised pre-training together with dropout significantly improved the recognition rate, achieving an 88% recognition rate compared to 64.7% without pre-training and dropout.

The second thesis group (Part 3.) focuses on the utilization of data modalities other than RGB (such as optical flow) in conjunction with transfer learning using pre-trained models that were trained on RGB data.

1.4 Synthetic data

Continual learning and transfer learning are two of the most commonly used approaches to achieve good generalization in dynamic environments. Continual learning is a technique that enables models to accumulate knowledge over time while avoiding forgetting previously acquired knowledge [82, 4]. Transfer learning, on the other hand, relies on training models on datasets that force them to generalize well [47, 49]. Both methods have shown promise in improving the generalization capability of deep learning models in dynamic environments.

In the third thesis group (Part 4.), an automatic synthetic dataset generation and annotation pipeline is proposed, with which we demonstrate how using synthetic data in the fine-tuning phase of transfer learning approaches and in continual learning can be beneficial for object and grasp detection for robotic manipulation.

1.4.1 Photorealistic synthetic data

Alberto Garcia-Garcia and Pablo Martinez-Gonzalez proposed a system that generates a vast amount of photorealistic data for indoor semantic scene segmentation and robot manipulation using virtual reality (VR) technologies and Unreal Engine [83, 84]. Their DL models, trained on this data, demonstrated favorable qualitative results in monocular depth estimation, 6D pose estimation for synthetic samples, and 6D pose estimation [83]. Their study suggests that VR technologies can assist in incorporating realism into synthetic scenes, such as realistic robot interactions.

Roberts et al. created Hypersim, a photorealistic synthetic dataset for indoor scene understanding [85]. Their dataset stands out by providing publicly available 3D assets, which is not the case for most synthetic datasets that only offer rendered images. The use of publicly available 3D assets increases the flexibility of use cases and potential for future development. Another significant aspect of their work is the decoupling of the annotation pipeline from the rendering process, which allows for the generation or modification of

annotations without re-rendering a scene. The proposed method in this dissertation for generating annotations for synthetic data is also decoupled from the scene preparation and the rendering. It allows the approach to be used for any scene, not limited to robotic manipulation, and to change the annotations without re-rendering. Furthermore, we utilized only publicly available free 3D assets, unlike Hypersim, which required a cost of $57K, with $6K dedicated to purchasing the necessary 3D assets. Although Hypersim's scenes are of higher quality and they used a greater number of 3D assets compared to ours.

In addition, Roberts et al. evaluated the sim-to-real performance of models trained on the Hypersim dataset through experimentation. They discovered that pre-training on Hypersim resulted in improved semantic segmentation performance on NYUv2 [86] when compared to pre-training on PBRS [87]. However, performance was not improved compared to pre-training on SceneNet RGB-D [88]. The authors attributed these findings to the fact that PBRS contains significantly more samples than Hypersim, while SceneNet RGB-D contains even more. They suggested that the improved photorealism of Hypersim allowed it to achieve comparable results to these larger datasets. They also proposed that there may be a trade-off between photorealism and the amount of data required to achieve good sim-to-real performance. This idea is supported by the work of Huh et al. in [48], who found that increasing the number of classes or pre-training data beyond a certain point did not significantly improve transfer learning performance with the ImageNet dataset [50]. These results suggest that a smaller amount of well-chosen, high-quality data may be more beneficial for transfer learning than simply increasing the size of the dataset. Although these findings are for the pre-training phase of transfer learning, they are promising for the fine-tuning phase, as the generalization capability should be maintained as much as possible during fine-tuning, while the size of the fine-tuning set should be kept to a minimum.

1.4.2 Domain randomization

The technique of domain randomization involves introducing artificially high levels of variation into the synthetic dataset. This compels models trained on such datasets to disregard the effects of the randomized factors, allowing them to generalize to real-world data [89, 90, 91, 92].

In their research, Tobin et al. established that a deep learning model, trained solely on a vast amount of low-fidelity rendered images with domain randomization, could be effectively utilized in real-world settings [92]. By introducing randomization in camera and object positions, as well as lighting conditions, along with utilizing unrealistic environmental textures, they were able to demonstrate that their DL model (trained exclusively on domain randomized data) was capable of accurately detecting basic geometric objects in a real-world environment. These detections were precise and dependable enough to be employed in a robotic grasping pipeline.

Tremblay et al. demonstrated that the domain randomization method could be applied to more intricate scenarios to bridge the reality gap [91]. Their research revealed that DL models trained on their domain randomized data for vehicle detection on the KITTI dataset [93] were able to perform competitively against models trained on the Virtual KITTI dataset [94].

Given the achievements of photorealism and domain randomization techniques in bridg-

ing the reality gap, a question arises: Is it feasible to leverage both techniques to obtain the best of both worlds? To address this question, Tremblay et al. proposed a combination of the two methods for bridging the reality gap [89]. Their work involved the utilization of photorealistic synthetic images in conjunction with domain-randomized ones. They demonstrated that DL models trained exclusively on such a synthetic dataset could achieve state-of-the-art performance in robotic manipulation. Our experiments also investigate the impact of fusing domain-randomized and photorealistic data for fine-tuning purposes in robotic manipulation.

Eversberg and Lambrecht investigated the effectiveness of implementing photorealism and domain randomization techniques in a synthetic dataset for industrial object detection [90]. They utilized Blender, an open-source 3D creation suite, to generate their synthetic dataset. Their findings indicate that domain randomization techniques were more effective for the background and clutter objects (unrelated to the object of interest) compared to higher realism. On the other hand, for the object of interest, realistic textures and lighting appeared to lead to better performance than domain randomization techniques.

Prakash et al. proposed a method called structured domain randomization, which aims to generate domain-randomized synthetic data while preserving the structural context of the environment [95]. For instance, conventional domain randomized image datasets for vehicle detection randomly place the vehicles, camera, and other environment objects in the scene. In contrast, in a structured domain randomization dataset, the vehicles are placed on roads to maintain the structural context of the environment. The authors compared their approach with photorealistic approaches, such as the Virtual KITTI and the GTA V dataset [96], as well as a domain randomized dataset [91]. Their experiments showed that models trained on a dataset with structured domain randomization can outperform models trained on photorealistic or domain-randomized synthetic data. The advantage of structured domain randomization comes from the trained model's better understanding of the scene context than models trained on conventional domain randomized data while being exposed to a similar level of variation in the data distribution.

Part 2

STATE AND ANOMALY DETECTION BASED ON REAL-TIME CLUSTERING

2.1 Motivation

Modern robot applications are often designed to operate with a high degree of autonomy, which allows them to function in environments that are not strictly structured. Moreover, cutting-edge robot applications enable humans and robots to work together in shared workspaces, resulting in highly non-deterministic behavior. These conditions create significant uncertainties in robot operation, leading to fluctuations in cycle time and making the overall process difficult to execute and oversee. Therefore, many robot applications now incorporate machine learning (ML) technologies, such as deep learning (DL), into their motion planning/control and quality assurance processes. ML methods provide effective and flexible solutions for dealing with non-deterministic processes.

As a consequence, ML for robotics is currently a widely researched topic. Recent approaches based on DL have shown remarkable results in processing large amounts of high-dimensional data [36, 30, 97, 98, 99]. With the latest robot controllers that enable access to internal states such as joint angles, accelerations, and drive torques, as well as the availability of affordable and easy-to-use cameras that provide high-quality visual data, acquiring the necessary data for training robust DL models is no longer a problem. However, a large part of DL approaches rely on supervised learning [36]. Given the vast number of trainable parameters of such models (often up to billions [98]), the training process requires numerous labeled examples to achieve good generalization [36, 30]. Labeling the samples is a resource-intensive and time-consuming task that often requires manual intervention [100].

Efficient utilization of available resources and minimizing setup times are crucial in robotics. Therefore, there is a need for methodologies that facilitate training ML models with fewer or no labeled examples. Internationally recognized ML scientists such as Yann LeCun, Yoshua Bengio, and Geoffrey Hinton predict that the future of ML will rely on the development of more advanced and precise unsupervised learning methods [36]. This sug-

gests that research aimed at harnessing large datasets without requiring laborious manual labeling holds significant scientific importance.

As suggested by Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, one way to achieve this goal is by utilizing unsupervised learning, which can reveal hidden relationships, structures, associations, or hierarchies within the provided data without requiring additional data labeling [101]. Such methods typically rely on a similarity measure to group the samples. This process is commonly referred to as clustering [101]. The similarity measure can also be expressed as a distance measure since the similarity between two data points can be interpreted as the distance between the two points in the feature space [101].

In this part, I propose an online clustering algorithm, utilizing a dynamically constructed ensemble of OCSVMs (One-Class Support Vector Machine) for the real-time identification of recurring events and anomalies in robot applications (Section 2.2). I demonstrate how a similar approach can also be used to evaluate generative models (Section 2.3).

2.2 OCSVMs for clustering in data streams

Several unsupervised techniques have been successfully applied in various DL-related problems, including k-means clustering for learning low-level filters for Convolutional Neural Networks (CNNs) [77], decision tree and neural network hybrids [102], and the use of generative models for learning feature extractors [103]. However, classical clustering methods are often limited by large computational complexity when dealing with large data samples and prior assumptions about the number of clusters or data distribution parameters, among other factors. As a result, these methods are not always suitable for robotics, where quick decision-making and adaptation during online operations is often necessary.

The characteristics of an ideal online clustering system for robotics are the following:

- Required human activity must be minimal in the training and setup process.
- The system should work for an unknown number of clusters as well.
- Clustering must be performed online to avoid the long-term storage of dense data.
- For long operation, the so-called concept drift [104] and concept evolution [105] must be handled.
- All of the most commonly used data types in modern robot applications should be accepted as input. E.g., joint-space coordinates/velocities/accelerations, Cartesian pose/speed/acceleration, forces, and torques, visual (image) and depth data, etc.

These requirements necessitate the use of lightweight clustering methods. Prior to implementation and experimentation, various techniques have been explored to identify those that meet the requirements.

Several recent approaches have employed an online clustering method that is based on the k-means clustering algorithm [37, 106]. However, when the system operates continuously, unexpected events or states may arise at any moment during the operation, including the ML model training or inference. As a result, the number of clusters may change over

time, and it cannot be determined beforehand. For k-means clustering-based solutions, adjusting the number of clusters adaptively would necessitate recomputing the cluster centroids [37]. Although this solution is theoretically feasible, it is time-consuming, and real-time operation is not possible as response times need to be extremely short for monitoring robot applications. Therefore, k-means clustering-based algorithms and extensions of the online k-means clustering are not well-suited for real-time clustering in robot applications.

Online clustering has also been accomplished with the use of self-organizing maps, as demonstrated by past research [107, 108, 39, 38]. For such approaches, the online behavior is usually achieved by an underlying tree-like architecture, which can automatically grow over time until a pre-defined convergence criterion is met [39, 38].

Previous studies, such as those conducted by Ghafoori et al. [109], Gretton et al. [110], and Ma et al. [111], have successfully utilized the One-Class Support Vector Machine (OCSVM) method to detect anomalies in complex data flows. SVMs train a linear classifier for binary classification in the input data. A kernel function can be used to transform the linear classifier optimization problem into a higher dimensional space, where the data points may be linearly separable even if they weren't in the original feature space [45]. This approach results in nonlinear decision boundaries in the feature space at the cost of additional computations. For linear SVMs, there are methods that enable training times to scale linearly or even sublinearly with the number of training examples and the number of features while only scaling linearly with the number of features during inference [112, 113]. For SVMs with non-linear kernels, approximation methods have been introduced that are able to drastically decrease the training and inference times for SVM models for non-linear classification [114, 115, 116]. Some of these approximation methods have been demonstrated to be applicable in tasks involving very high-dimensional data, such as visual object detection [114, 116].

Our algorithm leverages the OCSVM technique to perform online unsupervised state discovery and anomaly detection in robotics applications, building on the promising results obtained by prior research. We have conducted experiments with a representative collaborative robot application to demonstrate the effectiveness of this approach.

2.2.1 Formalisms

The Support Vector Machine (SVM) method trains a linear classifier for binary classification using a decision function of the form:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{2.1}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}$ is the weight vector and b is the bias [46]. Predictions are made based on the values of $f(\mathbf{x})$, where $f(\mathbf{x}) \geq 0$ results in a prediction of $y = 1$, and $f(\mathbf{x}) < 0$ results in a prediction of $y = -1$ [46]. While standard SVMs determine the parameters of the decision function using a training dataset with ground-truth labels, the One-Class Support Vector Machine (OCSVM) is specifically designed to distinguish samples belonging to one class from those of any other class [43, 44]. As a result, unsupervised training is possible using a set of unlabeled samples that are assumed to belong to the same class, which is useful for anomaly detection when rare data must be distinguished from the rest.

When implementing OCSVMs, there are typically two approaches that are used. One method, as outlined in [43] by Schölkopf et al., involves separating the training samples from the origin using a hyperplane in the feature space and maximizing the distance between the hyperplane and the origin. This design results in a quadratic programming problem that is formalized as follows:

$$\begin{aligned} &\min_{\mathbf{w},\xi_i,\rho} \left(\frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \right) \\ &\mathbf{w}\cdot\Phi(\mathbf{x}_i) \geq \rho - \xi_i \; \forall i \\ &\xi_i \geq 0 \; \forall i \end{aligned} \tag{2.2}$$

where $\Phi(x_i)$ is the kernel function, ρ is the size of the margin, ξ_i is the loss defined by the distances of outliers from the hyperplane, and ν is a parameter to set the trade-off between the number of outliers and the accuracy of the decision boundary.

The quadratic programming problem (2.2) can be solved by the Lagrange-multiplier method (using specialized solver algorithms such as SMO [117, 118]), which leads to a decision function similar to (2.1)

$$f(\mathbf{x}) = sgn(\mathbf{w}\cdot\Phi(\mathbf{x}_i) - \rho) = sgn\left(\sum_{i=1}^{n}\alpha_i K(\mathbf{x},\mathbf{x}_i) - \rho\right), \tag{2.3}$$

where α_i are the Lagrange multipliers.

The second method, introduced by Tax and Duin in [44], involves enclosing the training samples in the feature space with a spherical surface and minimizing the volume of this hypersphere. The hypersphere is characterized by its center $\mathbf{a}$ and radius $R > 0$. The center $\mathbf{a}$ is determined by a linear combination of the support vectors. The hypersphere does not necessarily enclose all training samples and instead employs a soft margin, whereby a loss function is used to determine the penalty associated with samples outside of the hypersphere. The formulation of this method is described by (2.4) [44], where the parameter C is used to adjust the penalty function.

$$\begin{aligned} &\min_{R,\mathbf{a}} R^2 + C\sum_{i=1}^{n}\xi_i \\ &\|x_i - \mathbf{a}\|^2 \leq R^2 + \xi_i \; \forall i \\ &\xi_i \geq 0 \; \forall i \end{aligned} \tag{2.4}$$

The decision function that classifies a given sample $\mathbf{x}$ as a member of the class is obtained by solving the minimization problem (2.4) using the Lagrange multiplier method. The decision is based on whether the distance between the sample $\mathbf{x}$ and the center $\mathbf{a}$ is smaller than the radius R. This distance can also be computed using the kernel method [44].

The Radial Basis Function (RBF), also known as the Gaussian kernel, is the most frequently used kernel function. Its formulation for two data points, $\mathbf{x}_i$ and $\mathbf{x}_j$, is given by

$$K(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \tag{2.5}$$

where the kernel parameter σ controls the sensitivity of the kernel function and should be set to a suitable empirically chosen value. It is often substituted by $\gamma = \frac{1}{2\sigma^2}$, which simplifies the expression to

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right). \tag{2.6}$$

The series expansion of the RBF kernel function leads to an infinite series, in which the terms

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle, \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2, \langle \mathbf{x}_i, \mathbf{x}_j \rangle^3 \ldots$$

are present, which are also kernel functions on their own. This provides the flexibility to design the classifier in a space with arbitrary dimensionality, allowing for nonlinear decision boundaries to be formed in the feature space [119].

2.2.2 Clustering algorithm

Algorithm 1. shows the proposed clustering approach. The algorithm takes a data stream $\mathcal{S}$ as its input. In $\mathcal{S}$, data points $\mathbf{X}_i | i \in \{t, t-1, t-2, ...\}$ are available at a sampling rate (f Hz) defined by the robot controller setup, with $\mathbf{X}_t$ being the most recent data point at time step t. For 6 DOF robot arms $\mathbf{X}_i$ usually contains the measured forces and torques acting on the robot's TCP, the Cartesian pose of the TCP or the robot pose in joint space so the dimensionality of the data points (d) is usually 6, meaning $\mathbf{X}_i \in \mathbb{R}^6$. A sampling window at time step t ($\mathbf{W}^t$) is defined by the window size w, where $\mathbf{W}^t | w = \{\mathbf{X}_t, \mathbf{X}_{t-1}, ..., \mathbf{X}_{t-w+1}\}$, meaning that the sampling window always contains the w most up-to-date data points. The state recognition and the anomaly detection (OCSVM predictions) are performed on the flattened $\mathbf{W}^t | w$ samples ($\mathbf{W}^t | w \in \mathbb{R}^{dw}$), whenever a new data point is available in $\mathcal{S}$. This means, effectively the rate of predictions matches the rate at which new data points are available in $\mathcal{S}$, making this method suitable for real-time use during inference, as long as the inference process is shorter than $1/f$.

For training the OCSVM models, a non-overlapping training window is maintained $\mathbf{W}^t_{train}$. Similarly to the sampling window, $\mathbf{W}^t_{train}$ also contains w number of data points, but contrary to $\mathbf{W}^t$, $\mathbf{W}^t_{train}$ is not updated upon the availability of each new data point, but rather after receiving w number of data points. This means if $\mathbf{W}^t_{train}$ is updated at time step t, it contains the same data points as $\mathbf{W}^t$ ($\mathbf{W}^t_{train} = \mathbf{W}^t$), but its elements would not change until time step $t + w$ ($\mathbf{W}^t_{train} = \mathbf{W}^{t+1}_{train} = ... = \mathbf{W}^{t+w-1}_{train}$), when its new elements would be $\mathbf{W}^{t+w}_{train} = \{\mathbf{X}_{t+1}, \mathbf{X}_{t+2}, ..., \mathbf{X}_{t+w}\}$. A training set $\mathbf{T}$ is formed of the n most recent training windows $\mathbf{T} = \{\mathbf{W}^t_{train}, \mathbf{W}^{t-w}_{train}, ..., \mathbf{W}^{t-nw}_{train}\}$, where n is the number of training samples.

The algorithm creates an ensemble of trained OCSVM models ($\mathbf{E}$) dynamically. At each time step, the latest sample $\mathbf{W}^t$ is evaluated by all the OCSVMs in the ensemble. An OCSVM's output can be either 1, indicating that the sample belongs to the cluster represented by the given OCSVM, or -1, indicating that the sample does not belong to that cluster. If the stopping criterion is not met yet, and none of the trained OCSVMs in the ensemble can classify the current sample ($\mathbf{W}^t$), it is assumed to belong to a new, unseen cluster. To recognize this cluster, a new OCSVM is trained using the current training set

($\mathbf{T}$). Thus, the number of trained OCSVMs and recognizable clusters increases online as the robot continues to operate.

It is often beneficial to add a cool-down period (cd) before training a new OCSVM, meaning a new OCSVM will only be trained and added to $\mathbf{E}$ if none of the already trained OCSVMs in $\mathbf{E}$ recognized any of the previous cd number of samples ($\mathbf{W}^{t-cd+1}...\mathbf{W}^{t}$). As a result, the algorithm can avoid training an excessive amount of OCSVMs in the presence of noisy data.

Algorithm 1: Clustering algorithm

```
input: S, w, n, stopping_criterion, cd
/* Initialize internal variables                                   */
W^t = [], W^t_train = [], T = [] E = [];
C = [[]];
i = 0, count = 0;
stop = False, no_train_step = 0;
on new X_t in S:
    /* Update data structures                                      */
    i += 1;
    W^t.append(X_t);
    if W^t.size() > w then
        W^t.remove(0);
    if i == w then
        W^t_train=W^t;
        i = 0;
        T.append(W^t_train);
        if T.size() > n then
            T.remove(0);
    else
        W^t_train=W^{t-1}_train;
    /* Perform predictions                                         */
    p = [];
    for OCSVM in E do
        p.append(OCSVM.predict(W^t));
    /* Train a new OCSVM if needed                                 */
    if !stop and (all(p==-1) or E.size() == 0) and T.size() == n then
        if count < cd then
            count += 1;
        else
            E.append(OCSVM.train(T));
            count = 0;
    else
        no_train_step += 1;
        if no_train_step == stopping_criterion then
            stop = True;
    /* Update contingency table                                    */
    C_update(p, C);
```

The algorithm assumes that the robot application contains repeated operations (robot cycles). Otherwise, the number of OCSVMs may increase indefinitely. A stopping criterion for the training of new OCSVMs can also be specified by the number of time steps without encountering a sample that could not be classified. In practice, instead of the number of samples, it may be practical to specify a certain amount of time, after which no new OCSVMs will be trained if all samples in the time frame can be recognized by at least one

OCSVM in $\mathbf{E}$. This time frame should be selected with consideration to the time needed for the robot to complete one repetition of the application.

Once the stopping criterion is satisfied, the algorithm transitions to the inference phase, where it is presumed that all the states and events during robot operation were accurately mapped under typical conditions. A sample that cannot be recognized by any OCSVM in the ensemble is classified as an anomaly. Therefore, this technique enables automated identification of events that fall outside the scope of typical robot operation and are hence classified as anomalous.

During the training process, OCSVMs that represent very similar clusters may emerge. Algorithm 1. defines a contingency table $\mathbf{C}$, which is used for the "pruning" of the number of recognized clusters. $\mathbf{C}$ is a quadratic matrix, and its number of rows/columns is equal to the number of trained OCSVMs in the ensemble. The elements in $\mathbf{C}$ represent how many times an OCSVM "fires" together with another one (how many times did they both recognize the same sample). This is done by performing the prediction with each OCSVM in the ensemble, storing all the predictions in an array $\mathbf{p}$, and updating the corresponding elements of the contingency table based on $\mathbf{p}$. The function responsible for the update of the contingency table can be seen in Algorithm 2.

Algorithm 2: C_update function

```
input : p, C
def C_update(p, C):
    for (i = 0; i < p.size(); i++) do
        if p[i] == 1;                                  // OCSVM i activated
        then
            for (j = 0; j < p.size(); j++) do
                if p[j] == 1;                          // OCSVM j also activated
                then
                    C[i][j]++;
```

2.2.3 Discovering hierarchies

Once the training procedure has finished (the stopping criterion is met), the contingency table can be used to discover multi-level hierarchies in the dynamically constructed ensemble of OCSVMs, using the bottom-up hierarchy building strategy described in Algorithm 3. The hierarchy building is performed in a bottom-up fashion by creating groups of OCSVMs in the ensemble, where each individual OCSVM corresponds to some elementary state. The inputs to the algorithm are the ensemble of trained OCSVMs $(\mathbf{E})$, the contingency table $(\mathbf{C})$ for a finite, recorded segment of the input stream $(\mathcal{R})$, the recorded segment of the input stream $(\mathcal{R})$, and a threshold value th to determine the level at which two OCSVMs are considered to be similar and should be grouped together. The data stream segment $(\mathcal{R})$ should cover precisely one or more application cycles. The parameter th ranges from 0 to 1, where a value of 0 indicates that all OCSVMs belong to one group, while a value of 1 indicates that only OCSVMs with identical predictions for all

samples are grouped together. Given these inputs, the algorithm generates groups $(\mathbf{G})$ that indicate which OCSVMs in the ensemble can predict more generalized states together.

Algorithm 3: Bottom-up hierarchy building strategy

```
input : E, C, R, th
/* Initialize internal variables                                  */
H = [], G = [];
N = R.size();
/* Calculate entropies                                            */
for OCSVM in E do
    H.append(Entropy(OCSVM.predict(R)));
H /= max(H);
for h in H do
    /* Find the index of the minimal entropy OCSVM                */
    i_h = argmin(H);
    if i_h in G then
        H[i_h] = 2 ;                  // OCSVM i_h is already in a group
    else
        /* Create a new group                                     */
        G.append([i_h]);
        I = [];
        for j = 0; j < C[i_h].size(), j++ do
            I.append(Info_Gain(C,(i_h, j), N)); // Compute Information Gain
              for i_h
        I[i_h] = -1;
        for j = 0; j < I.size(); j++ do
            /* Start with the most similar OCSVM                  */
            i_g = argmax(I/max(I));
            if C[i_h][i_g]/C[i_h][i_h] > th then
                G[-1].append(i_g) ;          // Add i_g to the current group
                I[i_g] = -1;
            else
                break;
        H[i_h] = 2;
return: G
```

The hierarchy building is based on the predictions of all the OCSVMs in $\mathbf{E}$ for all samples in $(\mathcal{R})$. First, the entropy over time is computed for the activations of all OCSVM models. This is done according to Algorithm 4. which accumulates the length of continuous positive predictions (pulse durations) made by an OCSVM in terms of the number of time steps and stores them in an array ($\boldsymbol{\delta}$). After that, $\boldsymbol{\delta}$ is normalized, and the entropy is computed over it using (2.7). The hierarchy-building strategy uses the OCSVM with the lowest computed entropy as a basis for the next group if it is not a part of a group already. The intuition behind this approach is that OCSVMs that activate continuously for a few distinct segments of the data stream are more informative than the ones that frequently alternate between positive and negative predictions. The former kind of OCSVMs will make predictions that have a lower entropy over time, and thus they can be distinguished from OCSVMs that failed to learn informative decision boundaries.

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{2.7}$$

The lowest entropy OCSVM is then compared to all other OCSVMs in the ensemble.

Algorithm 4: Entropy function

```
input : o
def Entropy(o):
    /* Create array (δ) of pulse durations for positive
       outputs                                                    */
    acc = 0, δ = [];
    o_prev = o[0];
    for o in o do
        if o == 1 then
            acc +=1;
        if o == -1 and o_prev == 1 then
            δ.append(acc);
            acc = 0;
        o_prev = o;
    if o_prev == 1 then
        δ.append(acc);
    /* Return entropy of δ                                        */
    if δ.size() == 0 then
        return: 0
    else
        δ /= sum(δ);
        H = 0;
        for p in δ do
            if p > 0 then
                H -= p · log2 p ;        // Entropy according to (2.7)
        return: H
```

For each OCSVM pair, the information gain is computed based on C and (2.8). The implementation of the information gain computation is described by Algorithm 5. The function calculates the joint and marginal probability distributions necessary for computing the information gain (I), which measures the amount of information obtained from one random variable by observing another [120]. In our case, the two random variables are the positive predictions of the two OCSVMs, meaning that a large information gain between two OCSVMs signals that just by observing the predictions of one of the OCSVMs, we gain a lot of information on what the predictions of the other OCSVM might be. Computing the information gain ensures that we do not group a very general OCSVM that activates for almost all states with an OCSVM that only activates at certain events. However, I on its own is not enough to determine whether the two OCSVMs usually activate at the same or at opposite times (since the value of I would be great for both cases). The hierarchy building algorithm uses the OCSVM with the maximum information gain, which is most likely a good choice for grouping with the minimum entropy OCSVM, and adds it to the group, depending on the value of th. If the ratio of the OCSVMs firing together and them firing separately is bigger than th, the maximum information gain OCSVM is added to the group. The process is then repeated until all OCSVMs have been added to at least one group.

$$I(X,Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \frac{\log_2 p(x,y)}{p(x)p(y)} \tag{2.8}$$

Algorithm 5: Info_Gain function

input : $\mathbf{C}$, i, j, N

def *Info_Gain(*$\mathbf{C}$*, (*i*,* j*),* N*)*:

- $I = 0$;
- `/* Calculate joint probability distribution */`
- $p(x=1, y=1)$ = $\mathbf{C}[i][j]$ / N;
- $p(x=1, y=-1)$ = ($\mathbf{C}[i][i]$ - $\mathbf{C}[i][j]$) / N;
- $p(x=-1, y=1)$ = ($\mathbf{C}[j][j]$ - $\mathbf{C}[i][j]$) / N;
- $p(x=-1, y=-1)$ = (N - ($\mathbf{C}[i][i]$ + $\mathbf{C}[j][j]$ - $\mathbf{C}[i][j]$)) / N;
- `/* Calculate marginal probabilities */`
- $p(x=-1)$ = $p(x=-1, y=1)$ + $p(x=-1, y=-1)$;
- $p(x=1)$ = $p(x=1, y=1)$ + $p(x=1, y=-1)$;
- $p(y=1)$ = $p(x=1, y=1)$ + $p(x=-1, y=1)$;
- $p(y=-1)$ = $p(x=1, y=-1)$ + $p(x=-1, y=-1)$;
- `/* Return Information Gain (according to 2.8) */`
- **if** $p(x=-1, y=-1)$ *!= 0* **then**
  - I += $p(x=-1, y=-1) \cdot \frac{p(x=-1,y=-1)}{p(x=-1)p(y=-1)}$
- **if** $p(x=1, y=-1)$ *!= 0* **then**
  - I += $p(x=1, y=-1) \cdot \frac{p(x=1,y=-1)}{p(x=1)p(y=-1)}$
- **if** $p(x=-1, y=1)$ *!= 0* **then**
  - I += $p(x=-1, y=1) \cdot \frac{p(x=-1,y=1)}{p(x=-1)p(y=1)}$
- **if** $p(x=1, y=1)$ *!= 0* **then**
  - I += $p(x=1, y=1) \cdot \frac{p(x=1,y=1)}{p(x=1)p(y=1)}$
- **return:** I

To create multi-level hierarchies, the hierarchy-building algorithm can be applied multiple times with various values of th. Instead of computing the entropies and information gain for individual OCSVMs during each iteration, the groups formed in the previous iteration can be utilized. Within each group, the prediction can be determined by a majority vote. The method can also be used to prune unwanted OCSVMs from the ensemble by defining a maximum number of accepted groups (N_G^{MAX}). To achieve this, the OCSVMs belonging to the last $\mathbf{G}$.size() $- N_G^{MAX}$ groups can be deleted, as the groups are added to $\mathbf{G}$ in ascending order of entropy (lower entropy meaning more informative predictions). Here, $\mathbf{G}$.size() denotes the number of groups in $\mathbf{G}$.

Using the hierarchy-building algorithm with a high value for th (≥ 0.9), OCSVMs that recognize similar events can be merged together. After the merging and pruning, the algorithm can be repeated with lower-and-lower values for th until only one group is retained. As a result, a hierarchy will be formed, described by the groups of OCSVMs/OCSVM groups. It is important to note that at the end of the hierarchy building for small values of th, groups will most likely include OCSVMs or groups that produce opposite predictions. For such cases, instead of the majority vote-based prediction, the group's prediction can be considered positive if any of the members of the group produce a positive prediction. As a rule of thumb, for groups created with $th < 0.5$ this method should be used instead of the majority vote.

A recommended procedure for determining the parameters and using the clustering algorithm, together with the hierarchy-building strategy:

1. Determine the sampling rate (f) and the approximate average length of robot states or events to be discovered ($\mathcal{T}$). These two parameters are pre-determined given the hardware components and the given robot application.
2. Determine a minimum sample size (w_{min}) that is still large enough to carry meaningful information about the robot state (e.g., $w_{min} = 3$ if position information is received and acceleration might be informative regarding the robot states), preferably enabling linear separation.
3. Use (2.9.) to set the values of w and n, starting from $w = w_{min}$ and increasing. Lower values of w tend to capture lower-level (shorter) states, meaning more OCSVMs will be trained and their firing frequency will be higher. Higher values of w incorporate more data, which can stabilize the predictions. However, too high values of w risk having a low value for n, which can result in OCSVMs that activate for larger sections of the cycle than intended (in extreme cases some can even stay active all the time).
4. Determine the stopping criterion (e.g., two repetitions of the robot application without new OCSVMs being trained).
5. Set the threshold at which OCSVMs will be considered as similar (e.g., $th = 0.9$).
6. Merge or delete OCSVMs ($th = 0.9$ hierarchy building strategy), discover possible cluster hierarchies (repeated use of hierarchy building strategy with decreasing th until only one group is left).
7. Run the algorithm in inference mode for state/event prediction and anomaly detection.
8. If the results are not satisfactory, go back to step 3. and change the value of w.

2.2.4 The effects of using sliding window sampling

Increasing dimensionality

In robot applications, it is often challenging to use a single data point to characterize the current process accurately. To address this, the clustering algorithm (Algorithm 1.) uses sliding-window-based data stream sampling to create inputs for the OCSVMs. The length of the window (w) and the number of samples used for training (n) determine the approximate duration of the application states that the algorithm can handle. Windowing can lead to high dimensionality in the input array (dw number of dimensions), where d is the number of dimensions in the data points. This results in a trade-off between window length, number of training samples, sampling rate, and real-time performance. Despite this trade-off, the increased dimensionality due to windowing can enable linear separation of the samples in the dw dimensional space, similar to the kernel method. This means that even if individual data points are not linearly separable in the feature space R^d, they can still be linearly separated in the high-dimensional space created by the windowing process.

An example of how linear OCSVMs can handle non-linear classification in the feature space with sliding window-based sampling is shown in Figure 2.1. The data points used

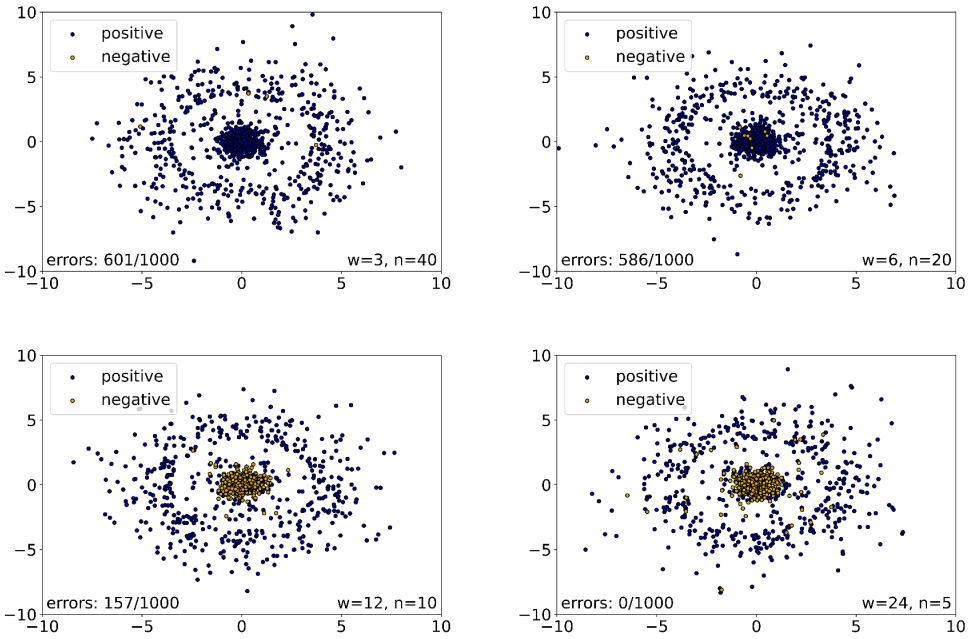


Fig. 2.1. Using varying sliding window sizes for input stream sampling to train a linear OCSVM for a non-linear problem

in the example are two-dimensional and were generated by alternating between two states, each lasting for 200 data points, with a 50 data point transition period between them. One state was characterized by a normal distribution with a mean of 0 and a standard deviation of 0.5 in both dimensions, and the other was also a normal distribution with a mean of 0 but with a standard deviation of 2.5, where only values with an absolute value of 3.5 and greater were retained. Multiple linear OCSVMs were trained using different sliding window widths (w) on a training sequence of 1000 data points with our algorithm. The figure displays the evaluation results of these OCSVM models, where the color of the data points corresponds to the OCSVMs' predictions, given the flattened window of the w most recent data points as input. Results indicate that using a higher value for w (12 and 24) enables the linear OCSVM models to learn non-linear decision boundaries in the feature space. It should be noted that the OCSVMs were trained in an unsupervised fashion without using the correct labels for the states. The error rate reported for each OCSVM in the figure represents the number of misclassified samples (excluding the transition period).

To further highlight this effect, we also evaluated the proposed clustering algorithm, using exclusively linear OCSVMs, on three datasets commonly used for testing classification models, the Iris Flower dataset [1], the UCI Wine dataset [2], and the UCI Digits dataset [3]. The datasets exhibit different numbers of features and classes. The Iris dataset consists of 150 instances which are three-dimensional vectors containing the sepal length, sepal width, and petal length in cm for the given instance. The task is to classify these instances into one of three classes. The Wine dataset contains 178 thirteen-dimensional instances. Each dimension constitutes one of thirteen (chemical) properties (such as alcohol level, color intensity, hue, etc.) for wines. The task is to classify instances into one of three types of wines. The Digits dataset contains 1797 instances, each having sixty-four

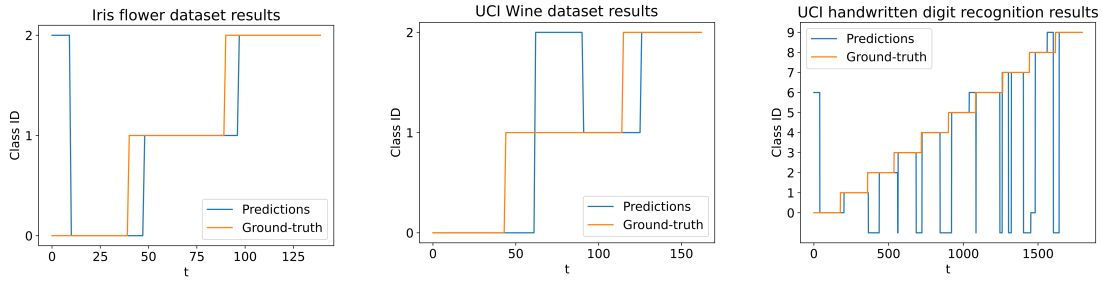


Fig. 2.2. Qualitative evaluation of the proposed clustering approach, using class labels from three different datasets. The clusters/group of clusters selected by the hierarchy-building strategy were associated with certain class IDs. A prediction of -1 means that none of the selected OCSVMs/groups recognized the sample

TABLE 2.1
RESULTS OF THE PROPOSED CLUSTERING ALGORITHM COMPARED TO BASELINE CLASSIFICATION MODELS. VALUES REPRESENT ACCURACY AS REPORTED BY [1, 2, 3]

	Xgboost	SVM	Random Forest	Neural Network	Logistic regression	Ours
Iris	92.105 (81.6-100)	94.737 (86.8-100)	89.474 (78.9-97.4)	97.368 (92.1-100)	94.737 (86.8-100)	**82.143**
Wine	97.778 (93.3-100)	80.000 (68.9-91.1)	100.000 (100-100)	97.778 (93.3-100)	93.333 (86.7-100)	**64.417**
Digits	96.160 (95.2-97)	97.607 (96.9-98.3)	96.939 (96.1-97.7)	96.327 (95.4-97.2)	94.880 (93.8-95.9)	**70.729**

dimensions. Each instance is a 8×8 sized normalized bitmap of a handwritten digit. The task is to classify instances into one of 10 classes, each representing a given digit.

It is a common approach to evaluate clustering performance against existing class labels, although class labels may not always be adequate for the data structure, and thus the evaluation of clustering methods [121]. Nevertheless, the proposed algorithm was able to develop clusters that closely relate to the class labels in all three test datasets. During the training procedure, the datasets were converted into data streams by retrieving their instances according to the corresponding class labels in ascending order. When reaching the end of a dataset, the stream would start over from the first instance (simulating the beginning of a new cycle). The best clusters (or groups of clusters) were selected using the hierarchy-building method. The training of new OCSVMs was terminated after 5000 time steps for all three datasets, although for the Iris and the Wine datasets, meaningful clusters appeared much sooner (around 1500 and 100 iterations). Figure 2.2. demonstrates the qualitative prediction results for a cycle after the conclusion of the training procedure for all three datasets.

For the Iris dataset, the parameters $w = 10$ and $n = 4$ were used with $\nu = 0.2$. On the Wine dataset, w was changed to 15, and for the Digits dataset, it was set to 40. Table 2.1. shows the quantitative results of the experiments in comparison to baseline classification methods.

The results show that even though our algorithm did not use class labels and only utilized linear OCSVM models, it was still able to achieve performance levels similar to the baseline classification methods, especially in the simpler Iris and Wine datasets. This is due to the windowing process, which (with the correct selection of w) ensures that the

samples are linearly separable and that the formed clusters will represent meaningful states even without explicitly labeling them.

Decreasing computational complexity

The proposed algorithm uses the $\mathbf{W}^t_{train}$ non-overlapping sliding windows for training dataset creation. This is because using overlapping windows (such as $\mathbf{W}^t$) with a fixed number of samples results in a shorter time period for collecting training data, leading to overfitting. In such a scenario, multiple OCSVMs are trained for specific parts of the event, which do not produce a decision boundary that generalizes well. On the other hand, if the time period for collecting training data is kept constant, n needs to be increased due to the overlap. In this case, if we consider the expected length of events in seconds $\mathcal{T}$ and the sampling rate f in Hz, to ensure that w and n are set appropriately, one can follow equations (2.9) and (2.10) for non-overlapping and overlapping windows, respectively.

$$n \approx \frac{\mathcal{T}f}{w} \tag{2.9}$$

$$n \approx \mathcal{T}f - w + 1 \tag{2.10}$$

It is clear that for the same w, there is a difference in n for overlapping and non-overlapping sliding windows. The computational complexity of the OCSVM training with a non-linear kernel is typically $\mathcal{O}(Dn^2)$ or $\mathcal{O}(Dn^3)$, and for a linear OCSVM, it is $O(Dn)$, where D is the number of features [114]. With the sliding window-based sampling $D = dw$, which means for a given w, this value will be the same for both overlapping and non-overlapping sliding window-based approaches. On the other hand, the difference in n between overlapping and non-overlapping sliding windows results in different computational times. The complete computational complexity of training OCSVMs with the proposed algorithm, using non-overlapping sliding windows for training, and (2.9) to determine w and n is

$$t_{train} \propto N^{OCSVM} w \mathcal{O}_{train}$$

linear:

$$\mathcal{O}_{train} = O(dn) \rightarrow t_{train} \propto N^{OCSVM} wdn \approx N^{OCSVM} \mathcal{T}fd \tag{2.11}$$

non-linear:

$$\mathcal{O}_{train} = O(dn^2) \rightarrow t_{train} \propto N^{OCSVM} wdn^2 \approx N^{OCSVM} \mathcal{T}fdn$$

where $\mathcal{O}_{train}$ is the original computational complexity of training an OCSVM directly on the data points.

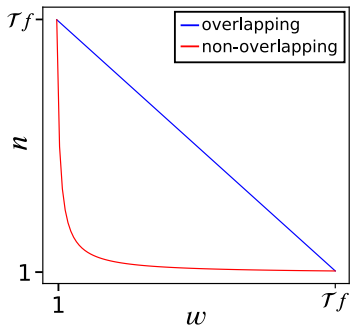


Fig. 2.3. Number of samples for the training set (n) for different values of w with a fixed sampling rate (f) and expected event length ($\mathcal{T}$)

Similarly, the computational complexity for the inference with our algorithm is

$$t_{inference} \propto N^{OCSVM} w \mathcal{O}_{inference}$$

linear:

$$\mathcal{O}_{inference} = O(d) \rightarrow t_{inference} \propto N^{OCSVM} wd \tag{2.12}$$

non-linear:

$$\mathcal{O}_{inference} = O(dn) \rightarrow t_{inference} \propto N^{OCSVM} wdn \approx N^{OCSVM} \mathcal{T} fd$$

where $\mathcal{O}_{inference}$ is the original computational complexity of predicting with an OCSVM directly on the data points.

Figures 2.3. and 2.4. show the difference in the number of training samples and the comparison of training times of OCSVMs for fixed $\mathcal{T}$ and f when using overlapping and non-overlapping sliding windows, considering a non-linear $(O(Dn^2))$ and a linear OCSVM $(O(Dn))$. The training set size was determined using equations (2.9) and (2.10) for non-overlapping and overlapping windows, respectively, based on the window size w. Both w and the number of training samples n fall in the range of 1 to $\mathcal{T}f$, where $w = 1$ represents a single data point per sampling window and $n = \mathcal{T}f$ means that the training set contains $\mathcal{T}f$ data points. When $w = \mathcal{T}f$, there is only one window in the training set ($n = 1$). In these extreme cases, the two windowing methods are equivalent, but $n = 1$ has no practical use, and $w = 1$ is equivalent to no windowing. The computational times are presented on a relative scale, which indicates the magnitude of the difference between the two methods relative to the duration of events $\mathcal{T}f$. The training and inference computational times were calculated based on (2.11) and (2.12), respectively, for a single OCSVM ($N^{OCSVM} = 1$). The findings indicate that adopting non-overlapping sliding windows in the training process can lead to considerably faster training times for both linear and non-linear OCSVMs. Moreover, using non-overlapping sliding windows can also result in faster inference times for non-linear OCSVMs.

Dehghani et al. provide quantitative analysis on the benefits of using overlapping sliding windows in time-series data for human action recognition [122]. They reveal that the perceived benefit of improved classification accuracy while using overlapping sliding windows is only due to the evaluation techniques which employ subject-dependent

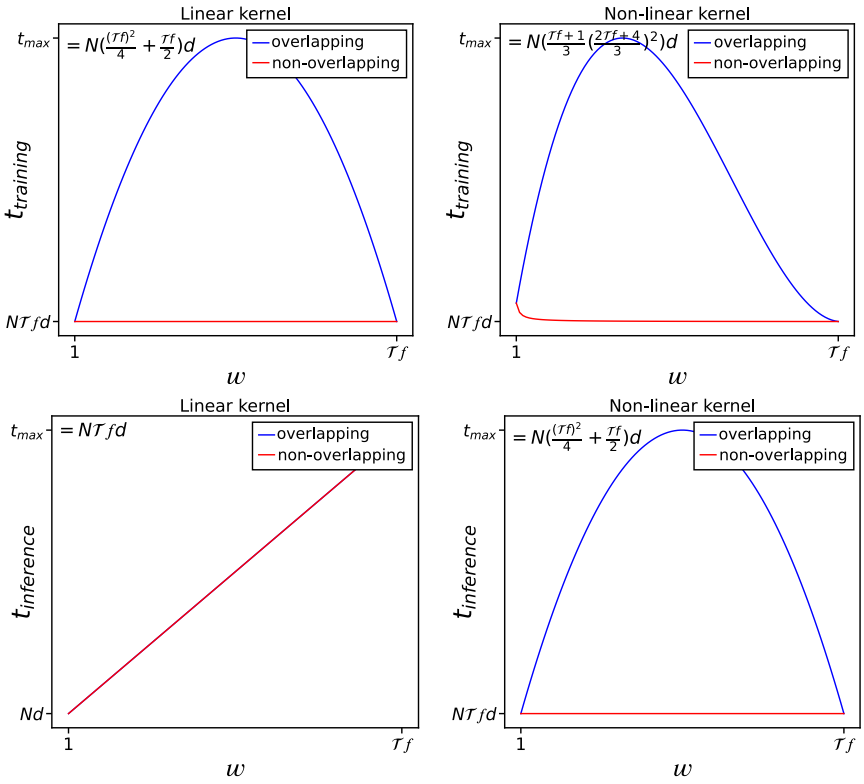


Fig. 2.4. Increase in training/inference times when using overlapping windows. ($N := N^{OCSVM}$)

cross-validation. They demonstrate that with subject-independent cross-validation, there is no noticeable performance gain when using overlapping sliding windows instead of non-overlapping ones. They also point out that using overlapping sliding windows increases the computational time and memory requirement of the process due to data duplication. This further supports our approach of using non-overlapping sliding windows.

The equivalent of subjects in the study by Dehghani et al. in our case, would be different robot setups/applications. For our method, the cross-environment generalization is not yet explored, and it is also not strictly required since the method learns in an unsupervised and online fashion, and it is intended to be re-trained for each new application.

Evaluating our method on the Iris and Wine datasets while using overlapping sliding windows for training revealed that the use of overlapping sliding windows does not provide a substantial increase in performance compared to using non-overlapping windows. In the meantime, the amount of stored data drastically increased, and the ν parameter of the OCSVMs had to be altered significantly to avoid overfitting. However, this resulted in an increased number of trained OCSVM models. On the Iris datasets, we achieved an accuracy of 85.0 (compared to 82.143 using non-overlapping windows) with $\nu = 0.4$, $w = 10$, and $n = 40$. After the same number of iterations, 65 OCSVMs were trained instead of the 10 OCSVMs when using non-overlapping windows. Similarly, in the Wine dataset, we achieved an accuracy of 69.325 (compared to 64.417 using non-overlapping windows), with $\nu = 0.6$, $w = 15$, and $n = 60$, while training 59 OCSVMs instead of 10.

2.2.5 Experimental results

The proposed online clustering algorithm was validated through a life-like collaborative robot application using the KUKA LBR iiwa 7-DOF industrial robot. The experiment imitates circumstances in an assembly application where the robot arm is dealing with non-ergonomic and highly repetitive jobs while the human worker is responsible for the tasks that require more developed manual skills. For the validation of an online clustering algorithm, concept drift also had to be taken into consideration.

The main requirements for the robot application, in order to serve as a suitable test-bed for the algorithm, are:

1. The application must consist of clearly separable states or events.
2. During the operation, there must be continually changing circumstances so concept drift can develop.
3. In the application, unplanned events (anomalies) should appear at unexpected times.
4. The application must exploit collaboration between the robot and the operator.

The experimental investigation was aimed to identify the states of the robot application automatically with the method shown in Algorithm 1.

In the experimental robot application, the robot draws polygons on a sheet of paper using a pencil as a tool. The application consists of two parts. During the first part, the human operator can teach the four vertices of a quadrilateral by moving the robot's flange by hand. This makes use of the collaborative nature of the robot application while enabling tasks that are similar in nature (corresponding stages of operation can be easily determined) but not identical since the human worker cannot move the flange to the exact same pose each time.

In the latter phase of the application, the robot operates autonomously without any collaboration. Its task involves producing polygons by linking previously learned vertices with straight lines. Upon completing a polygon, the robot creates a brief line segment that extends toward the center of the shape and proceeds to generate a smaller concentric polygon within the preceding one. For optimal tilt angle, the robot uses a distinct pencil orientation when drawing the first two edges compared to the last two edges. The process is then repeated for a total of 15 polygons following this procedure, after which the robot reverts to drawing the largest polygon to commence a new cycle.

The distinct states of drawing each side of the polygons are evident and easily distinguishable within the process. In our experiments, the clustering algorithm was used to identify and distinguish between four states that corresponded to drawing the four sides of the polygons. Concept drift appears in the experiment as the top of the pencil gradually wears away and as the robot draws smaller and smaller polygons in one cycle.

In addition, the output from the clustering process is utilized to identify anomalies such as a worn-out or broken pencil tip, thereby showcasing the algorithm's anomaly detection capabilities. If an anomaly is detected, the robot completes the ongoing polygon before using a fixed pencil sharpener located within its workspace to sharpen the pencil. It is important to note that the pencil may reach critical wear or break at any point during the robot's operation. Figure 2.5. shows the robot performing the drawing task. Such a robot application fulfills all the previously specified requirements.

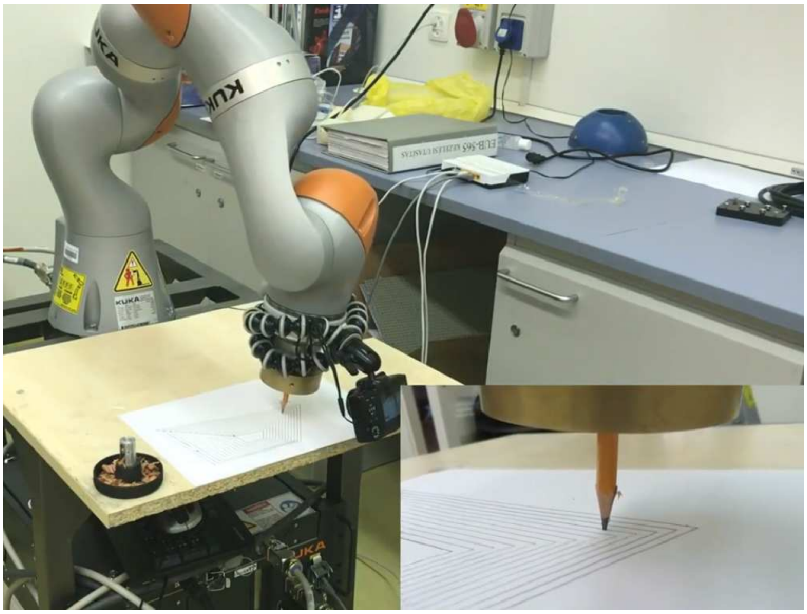


Fig. 2.5. The experimental robot application

During the evaluation, the main performance measure was whether the automatically detected states could be associated with the real-world working phases of the robot cell or not. Additionally, the assessment also examined the algorithm's capability to identify anomalies during the inference stage. The evaluation was carried out for the part of the application when the robot draws the four sides of the polygon. Accordingly, four states were associated with the four sides.

During the experiment, the values of w and n were set to 15 and 20, respectively. This translates to 0.075 seconds for each sample and approximately 1.5 seconds for a training set, assuming a sampling frequency of 200Hz. We used the scikit-learn software library [123] for the OCSVM implementation, which uses LIBSVM [118] and the OCSVM implementation from Schölkopf et al. ([43]), and we set $\nu = 0.05$. The OCSVMs used the RBF kernel (2.6) with $\gamma = 2.0$. It is important to note that in other applications, the values of w and n may require adjustment based on the expected duration of the working phases to be identified.

The input data stream for the algorithm comprised of force and torque values obtained from the robot's Tool Center Point (TCP), with the cartesian position of the TCP also being recorded simultaneously. The data stream was manually annotated using simple heuristics, which aided in the evaluation of the algorithm's prediction results. During the evaluation phase, the robot completed two full working cycles, which involved drawing 15 concentric polygons consecutively. The results indicate that after approximately 25000 data points (either at or just after the conclusion of the first full cycle), the number of OCSVMs ceased to increase, indicating no further appearance of unknown states, and the state transients became regular and clearly identified the states of the robot application. The progression of the number of OCSVMs during the first cycle can be seen in Figure 2.6.

Figure 2.7. shows the prediction results for one polygon drawing after the stopping criterion has been met. The upper graph represents the position of the robot's TCP along the

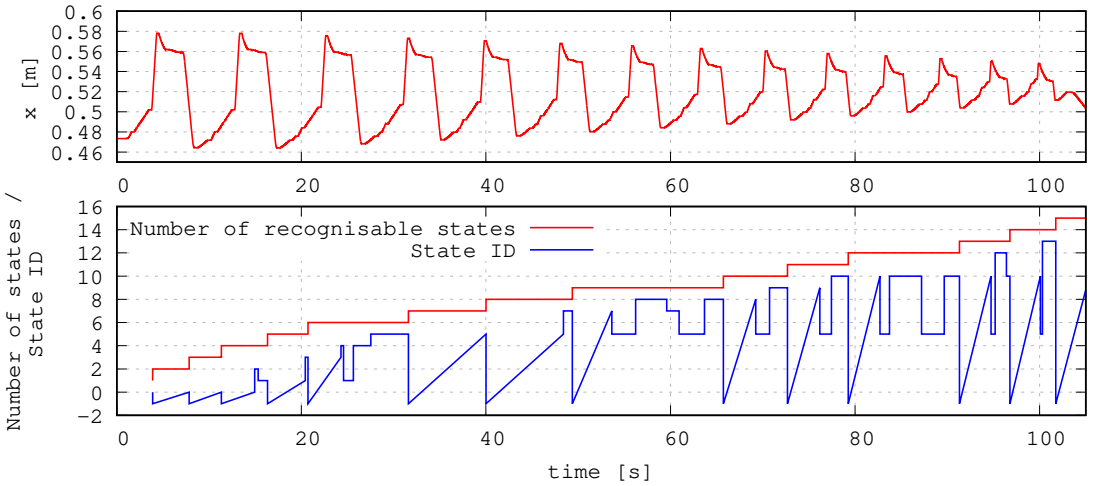


Fig. 2.6. Development of the number of OCSVMs in the ensemble during the first cycle of the robot operation. A state ID of -1 means an unrecognized state (none of the OCSVMs recognized the sample). In those cases, a new OCSVM was trained and added to the ensemble. If multiple OCSVMs activated, the one with the lowest ID was used

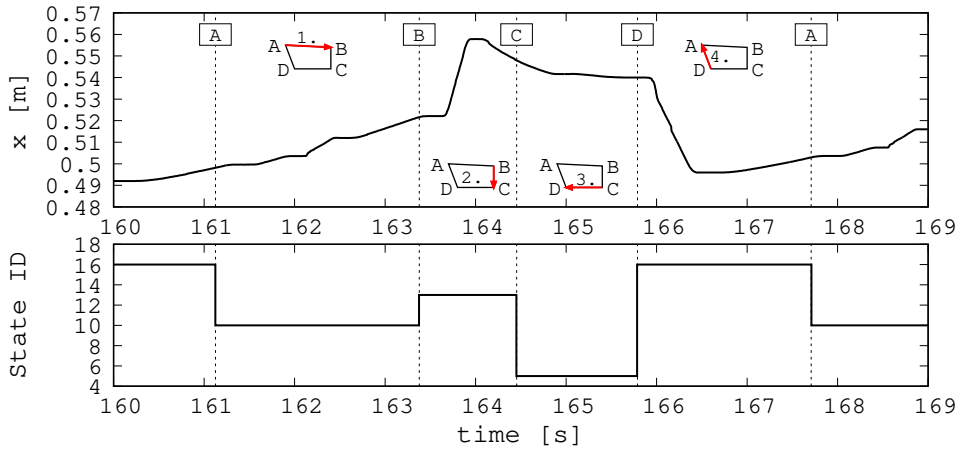


Fig. 2.7. Results of the clustering in inference mode

x-axis, while the lower graph portrays the ID of the active OCSVM model (corresponding to detected states) at all times. The figure clearly shows that the state transitions coincide with the manually determined location where the vertices (A, B, C, and D) are drawn.

During the evaluation, the algorithm could identify over **95%** of the states after the first cycle (during which new OCSVMs were still trained) and **100%** of anomalies (manually breaking the tip of the pencil). Using the hierarchy-building strategy, we also obtained groups of OCSVMs that predicted states that corresponded to orientation changes of the gripper. Naturally, since the orientation is only changed after drawing two sides of the polygon, the group, which could recognize a given orientation, incorporated both OCSVMs recognizing the polygon sections with the same orientation, thereby establishing a hierarchy between OCSVMs.

In addition to the experimental setup, the proposed algorithm was successfully deployed in an industrial context during the 2018 Pioneers Industry 4.0 Hackathon challenge held in Linz. The algorithm was part of the winning proposal in the Digital Twin category by the joint teams of REACH Solutions and MaxWhere, and it was applied to a packaging robot application for automatic state discovery and anomaly detection using a data stream of the robot's Cartesian pose.

The proposed algorithm has a limitation in that it may not generate accurate predictions for binary features. This could be due to the fact that the data points are closely clustered together, with little variation to distinguish meaningful differences. Decision-tree-based methods may be better suited for binary data in this regard.

2.3 OCSVMs for evaluating generative models

With the outstanding recent achievements of diffusion-based image synthesis models such as Stable Diffusion models or DALL-E [99, 97], interest in generative models is higher-than-ever. Such approaches are able to learn the underlying probability distribution in a provided dataset and sample new data points using their own representation of the data distribution. This means that a dataset can be enriched by novel synthesized data points, which is a great potential for training large DL models that require huge datasets. Robotics, in particular, could benefit tremendously from synthetic data generation, as the collection of real data in the context of robotics often involves performing the robot application with the real robot as well. Naturally, using real robots for extended amounts of time for the sole purpose of large-scale data collection is very expensive and time-consuming [30]. Synthetic data can be generated at much lower costs, and as such, it is very beneficial to use.

However, one big drawback of generative models is that the evaluation of the quality of the synthesized data often relies on subjective metrics. This makes it difficult to compare multiple models and decide which one would be better suited for a certain task.

In this section, an evaluation method is proposed for generative models, which is based on the clustering method described in Section 2.2.

2.3.1 Evaluation of Generative Adversarial Networks

In recent years, Generative Adversarial Networks (GANs) [124] have become widely popular among various types of generative models due to their ability to generate high-quality data automatically. GANs are composed of two neural networks that compete with each other, namely a generator and a discriminator. The discriminator network is responsible for distinguishing real and generated samples, while the generator network aims to generate synthetic samples of high quality from random input vectors in a way that deceives the discriminator network. This joint training process enables the generator to learn and replicate the underlying probability distribution of the original dataset. As a result, the discriminator network can later be used as a feature extractor, with the advantage that its training does not require supervision [103].

Once both networks have been trained, it is possible to examine the output of the generator network by interpolating in its input space [124, 103]. The generated results exhibit a smooth transition when the corresponding input vectors (inputs to the generator network) are sampled along the interpolated paths. This smoothness suggests that the discriminator network developed a meaningful representation of the feature space, which implies that it can extract reliable, general low-level features even when the training sets are discontinuous in the feature space [124, 103]. This makes GANs well-suited for training good-quality feature extractors for large datasets without supervision. As an example, Deep Convolutional GANs (DCGANs) developed by Radford et al. incorporate the advantages of GANs with the efficiency of convolutional layers, allowing for the generation of higher resolution images [103]. By training DCGANs on large image datasets, pre-trained feature extractors can be obtained, which can be used later for specific purposes through transfer learning techniques.

However, GANs struggle with the drawback that there is no simple, objective measure for their performance. Thus, a significant amount of research is being conducted to identify appropriate evaluation metrics for GANs. To this end, the Google Brain team conducted a comprehensive study comparing various GAN models and popular evaluation metrics [125]. They also introduced new benchmark datasets that employ precision and recall metrics to evaluate GANs. In conclusion, the team emphasized the need for future research on GANs to be based on objective evaluation metrics. The most popular techniques for GAN evaluation utilize an independent pre-trained feature extractor, such as the Inception model [126]. This approach involves the statistical analysis of the feature distribution of the feature extractor's output for actual and generated input samples, resulting in evaluation metrics such as the Inception Score (IS) and the Fréchet Inception Distance (FID) [127, 128]. Alternatively, some techniques aim to establish a geometrical interpretation for comparing actual and generated samples, including the Geometry score [129].

As per the aforementioned techniques, an effective evaluation metric can leverage an independent model like the Inception network. It can also be advantageous if the method has a geometric interpretation. This section demonstrates the potential benefits of the OCSVM-based clustering approach (Algorithm 1.) for evaluating generative models.

2.3.2 Methodology

The proposed approach aims to establish a performance metric for generative models that is independent of the model's training procedure and architecture. Similar to the Fréchet Inception Distance, the clustering-based method offers a probabilistic interpretation, but it does not rely on features from a pre-trained feature extractor. As such, it can evaluate generative models without being tied to the semantics of the generator's output data. While most generative models are used for image generation, where pre-trained feature extractors, such as the Inception model, can be used to assess generated samples, training a generative model on non-visual data without an existing pre-trained feature extractor can also be beneficial (since GAN models can be used to acquire a pre-trained feature extractor, without data labeling). However, when no pre-trained feature extractor exists for the given data type, evaluation methods based on pre-trained feature extractors cannot be used. The proposed method overcomes this issue by providing an evaluation method that is not dependent on a pre-trained feature extractor. Additionally, the proposed method can be applied in scenarios where data is not interpretable by humans, making it hard to evaluate the generative model empirically.

Furthermore, SVMs have a simple geometric interpretation (learning a separating hyperplane) which makes this measure similar to the Geometry Score as well. The proposed method only requires the original and generated samples for evaluation and does not rely on the generative method and its structure. Thus, it can be expanded for multiple generative model types and training procedures. As a proof of concept, the evaluation of some test GANs is elaborated.

To train the clustering method, we randomly select a subset of the real dataset with n samples and set w to 1. We apply the algorithm to all n samples, resulting in a single trained OCSVM, which can be employed for evaluation. Subsequently, we can evaluate different generative models trained on the same dataset by feeding their generated samples to the OCSVM and computing the probability of a randomly drawn synthetic sample being classified as an outlier. The generative models with smaller outlier probabilities generate samples that have a probability distribution more similar to the original ones. Moreover, apart from the relative comparison of different generative models, the outlier probability can also indicate the absolute performance of a single generative model if we examine it based on a pre-determined probability of the possible outlier probability for samples drawn from the real dataset.

Naturally, the performance measure relies on the specific OCSVM model used, which means that comparisons are only valid if they employ the same OCSVM. Additionally, it is crucial to acknowledge that a low outlier percentage does not guarantee diversity. Thus, the method provides an upper bound for the model's performance and is susceptible to mode collapse as it currently stands. Due to mode collapse, the GAN may generate samples that originate from a subset of the input space, denoted as $S_{generated} \subset S$. If the centroid of this subset lies within the decision boundary of the OCSVM, the generated samples will have very few outliers. To overcome this issue, the proposed method could be augmented with a diversity measure in the future, allowing for the detection of varying degrees of mode collapse.

2.3.3 GAN evaluation experiments

We conducted two experiments to illustrate the potential of the proposed performance measure in evaluating GAN models. The first experiment showcases how our approach can be a reliable evaluation method for the comparative and absolute assessment of GAN models using a simple custom GAN architecture and data from the MNIST dataset [130]. Additionally, the experiment displays the impact of mode collapse on the performance scores. In the second experiment, a comparative analysis is performed between our method and the Inception Score (IS) and Fréchet Inception Distance (FID) scores, utilizing pre-trained IC-GAN models [131] and data from the ImageNet dataset [50].

Custom GAN on MNIST data

In order to test the applicability of the proposed method, a test task for model comparisons was set up. This task is to generate samples similar to the MNIST dataset [130], which consists of 28x28 pixel grayscale images of handwritten digits.

For the experimentation, a simple GAN structure was created, as illustrated in Figure 2.8. The generator network is a fully connected neural network comprising three hidden layers and an output layer. The hidden layers consist of 256, 512, and 1024 neurons, respectively, while the output layer has 784 neurons, which is equal to the number of pixel values in a 28x28 image. The network's input is a latent vector $\mathbf{z} \in \mathbb{R}^{100}$, with the vector elements drawn from a noise source with a standard normal distribution. The Leaky ReLU activation function is applied to the hidden layers, while the hyperbolic tangent function is used for the output layer. The network's weights are initialized randomly via a uniform distribution.

The discriminator network is also a fully connected neural network with three hidden layers, and it takes a flattened image as its input ($\mathbf{x} \in \mathbb{R}^{784}$). The hidden layers are made up of 1024, 512, and 256 neurons, respectively, and use Leaky ReLU activation. The output layer has one neuron and uses a hyperbolic tangent activation function. Dropout was applied after all hidden layers with a dropout rate of 0.5 to avoid overfitting.

The generator network, $G()$, takes a latent vector ($\mathbf{z} \in \mathbb{R}^{100}$) and produces a flattened image, denoted as $x = G(\mathbf{z})$, ($\mathbf{x} \in \mathbb{R}^{784}$). The discriminator network, $D()$, accepts a flattened image as its input that can come either from the generator or from the original dataset. The output of $D()$ for a given ($\mathbf{x} \in \mathbb{R}^{784}$), denoted as $y = D(\mathbf{x})$, is a scalar value ($y \in \mathbb{R}$) signaling whether the given input is a generated sample or a real sample that is drawn from the original dataset.

During training, the discriminator is trained to be able to distinguish the generated images from the real ones. The objective of the generator network is to fool $D()$ by generating synthetic images that are very similar to the original ones. The training process of these competing networks can be interpreted as a mini-max two-player game between $D()$ and $G()$ [124], and it can be formulated as

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim P_{data}(\mathbf{x})}[\log_2 D(x)] + \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})}[\log_2(1 - D(G(\mathbf{z})))]. \quad (2.13)$$

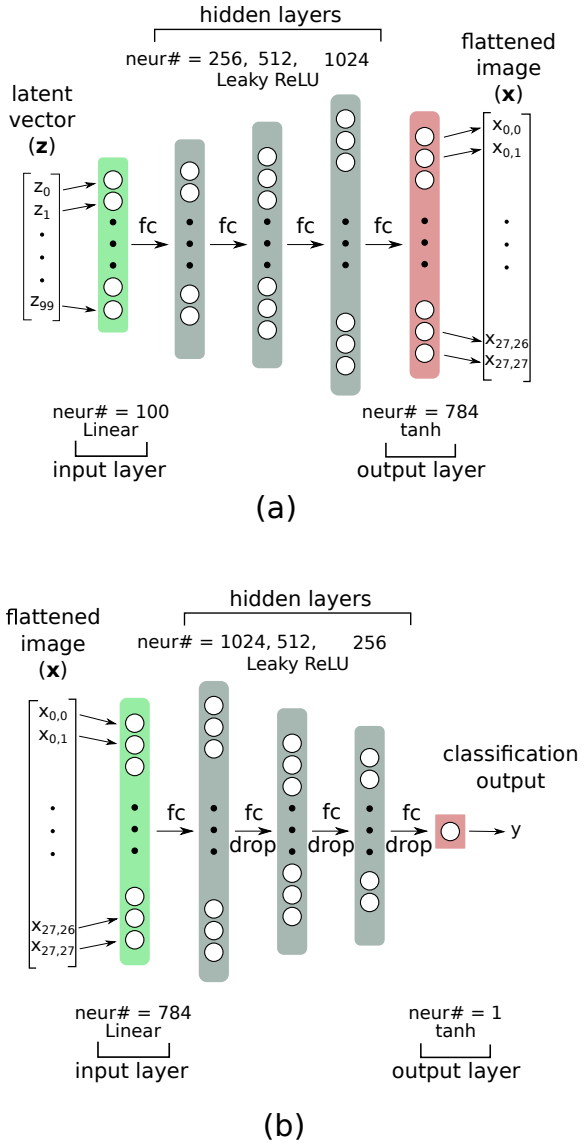


Fig. 2.8. Test model architecture (a) Generator architecture (b) Discriminator architecture

The discriminator network utilizes the cross-entropy loss function. For a batch of training samples, which can be computed as

$$\frac{1}{b}\sum_{i=1}^{b}[\log_2 D(\mathbf{x}^{(i)}) + \log_2(1 - D(G(\mathbf{z}^{(i)})))]. \tag{2.14}$$

The loss for the generator network for a training batch can be described as the second term of (2.14), so the generator loss is

$$\frac{1}{b}\sum_{i=1}^{b}\log_2(1 - D(G(\mathbf{z}^{(i)}))). \tag{2.15}$$

For the minimization of the loss function, both networks used the Adam optimizer with a learning rate of 0.0001.

IC-GAN on ImageNet data

The MNIST dataset contains 28×28 pixel grayscale images, resulting in a 784-dimensional sample space. Despite its considerable size, this dimensionality pales in comparison to samples encountered in traditional image-processing tasks. To illustrate the applicability of the proposed evaluation method to high-dimensional data, we employ it to assess GAN models for ImageNet data. For this experiment, we focus on a tiny subset of the ImageNet dataset, comprising 169 images exclusively from the "West Highland White Terrier" class, to evaluate various Instance-Conditioned GAN (IC-GAN) models. IC-GANs facilitate instance-based conditioning for image generation by training on a partitioned data manifold with overlapping regions, each represented by a data point and its nearest neighbors [131]. Consequently, they capture the local distribution around each data point. When conditioned with a specific instance, IC-GANs can generate samples that map to the proximity of the conditioning instance on the data manifold, ensuring the generated samples resemble the conditioning instance.

For our experiments, we employed pre-trained IC-GAN models[1] without further training, conditioning them with an instance from the ImageNet dataset. The chosen conditioning instance is from the "West Highland White Terrier" class; thus, in evaluating the models, we compare the generated samples with a set of 169 images from the ImageNet dataset belonging to the same class. The output of the IC-GAN models is a 256×256 RGB image. Accordingly, all real images were resized to a 256×256 resolution, resulting in a 196,608-dimensional sample space. Employing our proposed method, we utilized all 169 real samples to fit OCSVMs for GAN model evaluation.

We present results for three distinct IC-GAN models, employing the Inception Score (IS), Fréchet Inception Distance (FID), and outcomes from our proposed evaluation method. All three models utilize the BigGAN backbone [132]. One model, IC-GAN-COCO, underwent training on the COCO-Stuff dataset [133], while the other two were trained on ImageNet. One of the latter two models, which we refer to as IC-GAN-ImageNet-halfcap, used a reduced channel multiplier (as in [131]), affecting the network width. The other model we refer to as IC-GAN-ImageNet.

2.3.4 Results

Custom GAN on MNIST data

In order to demonstrate the use of the proposed evaluation method, GAN models were trained for 10000, 25000, and 50000 iterations with batch sizes of 50 and 100. The discriminator weights were updated two times for each generator weight update ($k = 2$), and the learning rates were 0.0001 for all models. The GANs were implemented through the Python API of the TensorFlow deep learning framework [134].

To demonstrate the consistency of the comparative results, multiple OCSVM models were created, all trained on 20000 randomly sampled images of the MNIST dataset ($n = 20000$, $w = 28 \times 28 = 784$). To match the output range of the GAN generator, the image

[1] Implementation and weights available at `https://github.com/facebookresearch/ic_gan`

Fig. 2.9. Outliers from the MNIST dataset

pixels were scaled to the $[-1, 1]$ interval. Implementation of the OCSVM models was carried out using the scikit-learn Python module [123].

For each GAN, the non-outlier probability was determined by calculating the average percentage of non-outliers from 500 generated images, repeated a hundred times.

Two GAN models were trained for all iteration/batch size combinations. The GANs are named like a_b_c where a is the number of iterations, b is the batch size, and c is the model number. The evaluation was performed with linear OCSVM models using $\nu = 0.01$ and $\nu = 0.05$. For both values of ν, three OCSVM models were trained with different randomly sampled training sets.

In the case of OCSVM models with $\nu = 0.01$, the probability of a real sample from the MNIST dataset being an outlier is 1%. For models with $\nu = 0.05$, this probability increases to 5%, based on the OCSVM formulation in (2.2). This property was confirmed through the collection of additional testing sets of samples from the MNIST dataset, which were disjoint from the training sets. The outlier probabilities were calculated in these sets using the different OCSVM models. The outliers within the MNIST dataset were mostly represented by samples that appeared faulty or anomalous. This suggests that OCSVMs are capable of identifying samples that a human observer would also consider faulty. Figure 2.9. displays some samples from the MNIST dataset that were classified as outliers by an OCSVM model with $\nu = 0.01$.

TABLE 2.2
AVERAGE PROBABILITY OF GENERATING NON-OUTLIER SAMPLES FOR DIFFERENT GAN MODELS

		OCSVM					
		$\nu = 0.01$			$\nu = 0.05$		
GAN model	10000_50_1	0.982	0.981	0.978	0.957	0.951	0.950
	10000_50_2	0.861	0.861	0.853	0.803	0.800	0.799
	10000_100_1	0.899	0.900	0.897	0.849	0.840	0.842
	10000_100_2	0.856	0.852	0.842	0.787	0.772	0.778
	25000_50_1	0.916	0.916	0.917	0.852	0.850	0.847
	25000_50_2	0.934	0.933	0.933	0.874	0.870	0.871
	25000_100_1	0.966	0.967	0.965	0.922	0.920	0.921
	25000_100_2	0.920	0.919	0.915	0.853	0.845	0.845
	50000_50_1	0.979	0.978	0.976	0.947	0.939	0.942
	50000_50_2	0.964	0.963	0.957	0.922	0.918	0.918
	50000_100_1	0.956	0.954	0.950	0.895	0.886	0.888
	50000_100_2	0.959	0.958	0.956	0.904	0.898	0.904

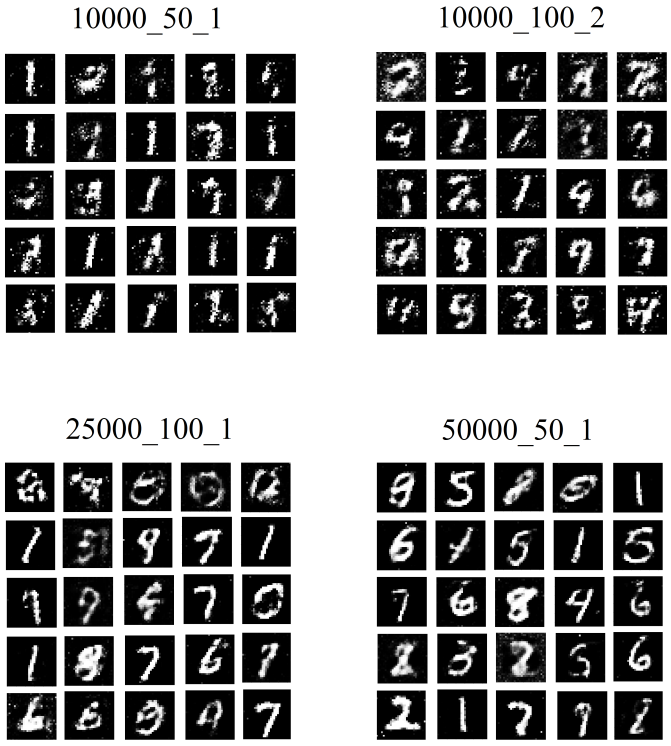


Fig. 2.10. Generated images of some models

Table 2.2. shows the results of the experiments, and Figure 2.10. shows non-cherry-picked and randomly sampled generated images from some of the models that were used for the experiments.

Table 2.2. demonstrates that higher scores (less probability of a generated sample being an outlier) are achieved with more training iterations of the GANs. The batch size does not significantly affect the scores. The use of different OCSVM models to evaluate the GANs produces similar results, maintaining the relative ranking according to the performance measure. Thus, the proposed method is robust enough to be employed as an independent and objective evaluation method.

When comparing the scores with the generated images in Figure 2.10., we can observe that models with higher scores tend to generate visually more appealing samples. It is noteworthy that model 10000_50_1 achieved an exceptionally high score. However, the generated images reveal that this is due to mode collapse, as the model predominantly generates images of the number one while ignoring other classes. Model 25000_100_1 also exhibits a mild form of mode collapse. It is important to note that this measure does not account for diversity as it is. A model that generates very similar samples within the decision boundary of the OCSVM will have a low probability of generating outliers. This limitation can be addressed by employing a diversity metric for the generated samples and performing evaluation using a combined measure.

The presented results clearly suggest that the proposed clustering-based approach using OCSVM models for computing the evaluation metric is a feasible approach for evaluating generative models. This method combines probabilistic and geometric approaches and

has several advantages over previous methods. One key benefit is that it does not require an external benchmarking dataset or a pre-trained feature extractor, making it applicable to any kind of data. Moreover, the OCSVMs used for evaluation are trained directly on the training dataset for the generative model in an unsupervised fashion. For large datasets with many features, linear OCSVMs can be used as demonstrated, which can be trained faster than large deep neural networks such as the Inception model. Furthermore, it has already been proven that the training time of linear SVMs decreases with the increase of the training dataset size for a fixed generalization error [135]. The proposed evaluation metric is also robust to changes in OCSVM parameters, such as ν, and consistently ranks models regardless of these values. It can be used to evaluate a single generative model or to compare two or more models trained on the same dataset.

However, a drawback of this approach is its inability to detect mode collapse, which should be evaluated using a separate measure of sample diversity. It is also not yet clear how well this method performs on special architectures, such as cGANs and other types of generative models. Overall, the proposed OCSVM-based evaluation metric presents a promising alternative for evaluating generative models.

IC-GAN on ImageNet data

Figure 2.11 depicts the generated samples from the IC-GAN-ImageNet, IC-GAN-ImageNet-halfcap, and IC-GAN-COCO models. Employing our proposed evaluation method, three distinct linear OCSVMs were trained on the 169 real samples, with ν values set to 0.5, 0.6, and 0.7. The ν value controls the metric's sensitivity. To accommodate high-quality generated data that closely resembles real samples, ν is incrementally adjusted until the metric is no longer "saturated" (saturation implies the absence of outliers in the generated samples, leading to the GAN achieving the maximum possible score of 1.0). Table 2.3 presents the performance scores computed using our method, denoted by the corresponding ν values, along with the IS and FID scores.

Fig. 2.11. Generated samples from the IC-GAN-ImageNet, IC-GAN-ImageNet-halfcap and IC-GAN-COCO models

Observing Figure 2.11, it is evident that the IC-GAN-COCO model struggled to generalize to the "West Highland White Terrier" class under the selected instance conditioning. This outcome is expected, considering the model was trained on the COCO-Stuff dataset, which lacks images from the "West Highland White Terrier" class. This observation is

TABLE 2.3

COMPARISON OF EVALUATION METRICS (IS, FID, AND OUR PROPOSED METHOD USING $\nu = 0.5$, $\nu = 0.6$, AND $\nu = 0.7$) ON DIFFERENT IC-GAN MODELS. AN UPWARD ARROW NEXT TO THE SCORE NAME MEANS HIGHER VALUES ARE BETTER, AND A DOWNWARD ARROW MEANS LOWER VALUES ARE BETTER.

Model	IS↑	FID↓	$\nu = 0.5$↑	$\nu = 0.6$↑	$\nu = 0.7$↑
IC-GAN-ImageNet	1.5728	**4.2307**	0.92	0.92	0.8
IC-GAN-ImageNet-halfcap	1.4196	4.5334	**1.0**	**0.96**	**0.88**
IC-GAN-COCO	**1.7246**	6.2961	0.28	0.16	0.08

further supported by the low scores associated with this model in our proposed evaluation method and the highest FID value. On the other hand, the Inception Score assigns a high score to this model, highlighting one of the drawbacks of IS. IS is derived from the KL divergence between label distributions and the sum of label distributions (marginal distribution) of the generated samples after processing through the Inception model [127]. The score is high for dissimilar label and marginal distributions, meaning sharp label distributions (representing distinct objects in an image) and a nearly uniform marginal distribution (indicating the generator's ability to produce samples from multiple classes). Consequently, IS falls short in capturing intra-class diversity, leading to conflicting results in our experiment compared to FID, our proposed method, and even human judgment. In contrast, our proposed performance measure assigns a low score to the IC-GAN-COCO model, consistent with the computed FID values and human evaluation of the generated samples.

Our proposed method assigns a higher score to the IC-GAN-ImageNet-halfcap model compared to the IC-GAN-ImageNet model, in contrast to the findings from FID, where the IC-GAN-ImageNet model emerged as the top performer. Although the score difference between these two models is not as significant as their difference from the IC-GAN-COCO model, a potential explanation could be a mild case of mode-collapse in the IC-GAN-ImageNet-halfcap model. While subjective judgment is required to determine whether the outputs of IC-GAN-ImageNet or IC-GAN-ImageNet-halfcap appear more realistic, it is apparent that IC-GAN-ImageNet generates a more diverse set of samples, including images with multiple dogs and out-of-frame faces. Conversely, most outputs from the IC-GAN-ImageNet-halfcap model depict a single front-facing dog. The higher score of the IC-GAN-ImageNet-halfcap model might be attributed to this narrower distribution of generated samples, leading to a larger ratio falling within the decision boundaries of the OCSVMs. A more in-depth analysis involving multiple classes and a more extensive set of generated samples is needed to make a definitive judgment on the superiority of these two models. Nevertheless, the experiment demonstrates that our proposed evaluation metric aligns with FID and human evaluation.

2.4 New scientific results

Thesis 1

I present a new clustering algorithm (Algorithm 1.) for the automatic online and real-time classification of operation states and detection of anomalies in robotic applications utilizing finite state descriptor dimensions and continuous numerical features. Besides robotics applications, the use of the algorithm can be generalized to the evaluation of generative machine learning models. The effectiveness of the proposed method was demonstrated on a representative collaborative robot application, as well as through successful implementation in a real-world industrial setting.

Sub-thesis 1.1

I showed that a dynamically constructed ensemble of OCSVMs together with a contingency table can be used to discover a multi-level hierarchy of elementary states using a bottom-up hierarchy-building strategy (Algorithm 3.).

Sub-thesis 1.2

Through representative examples, I showed that using non-overlapping sliding windows in the input data stream for acquiring training samples significantly reduces the computational time without significantly degrading the prediction performance. I provided formulas for determining the computational requirements depending on the parameters of the method (2.11) and (2.12).

Sub-thesis 1.3

I demonstrated that the OCSVM-based anomaly detection approach, initially designed for robotics applications, can be effectively used to evaluate generative machine learning models through statistical analysis of synthesized outputs. Unlike current evaluation methods, this approach is able to evaluate models independently of the output data semantics, as it does not require a pre-trained feature extractor.

Related publications: [KA1, KA2, KA3]

Part 3

CROSS-MODAL MAPPING-BASED TRANSFER LEARNING USING PRE-TRAINED RGB FEATURE EXTRACTORS

Transfer learning in deep learning (DL) typically involves reusing the feature extractor of a large deep learning model pre-trained for image recognition on a massive dataset of RGB images. The image recognition head is replaced with a different network, enabling "fine-tuning" on a smaller dataset for a specific task without altering the weights of the pre-trained feature extractor [47, 48]. Here, we consider the source domain to be the large-scale dataset for image classification and the target domain to be the smaller dataset for a specific task. Transfer learning assumes that if the source domain dataset is adequately representative, the pre-trained feature extractor can produce meaningful features for the target domain data without updating its weights. This approach is advantageous because it allows the utilization of high-level, descriptive features without training a large feature extractor model, which would require significantly more training data and time. However, certain problems may necessitate using modalities other than RGB for which pre-trained feature extractors do not exist. This part demonstrates how transfer learning can still be employed for such problems by specifying an appropriate cross-modal mapping for the input data. We support this statement with experiments focused on moving object segmentation in video sequences that utilize optical flow modality in the input of RGB pre-trained feature extractors.

First, the cross-modal mapping approach is introduced, which enables the encoding of optical flow and grayscale image data as an RGB image. Then, a new DL model, the OFSNet (Optical Flow Segmentation Network), is proposed, with which we demonstrate how the cross-modal mapping approach can be utilized with an RGB pre-trained feature extractor for the segmentation of moving obstacles in a real-world mobile robot navigation task leveraging optical flow information. Additionally, we provide evaluation results on the DAVIS dataset for moving object segmentation, using the OFSNet model and U-Net variants employing the cross-modal mapping method against other state-of-the-art approaches. Finally, we propose a new loss function and corresponding training strategy,

which were utilized for the training of the OFSNet model to mitigate the effects of the imbalance between the object and background mask sizes.

3.1 Optical-flow input for models pre-trained on RGB data

3.1.1 Motivation and related approaches

Detecting motion and segmenting moving objects in video sequences is a crucial computer vision task that supports object tracking and recognition in various fields, such as video surveillance, mobile robotics, and self-driving cars. However, solving this problem remains challenging due to several factors, like camera motion, dynamic backgrounds, occlusions, shadows, and changes in lighting conditions.

Background subtraction methods are often utilized for detecting moving objects in video sequences, assuming a static background, particularly in video surveillance systems [136]. These methods involve estimating a model of the background and then classifying the pixels of a given video frame as either background or foreground based on their similarity to the background model. Various techniques exist for background modeling, including those that rely on local patterns of textures and photometric features [137, 138], as well as those that use a mixture of Gaussian distributions [139, 140]. Recently, Patil and Murala proposed a background estimation and foreground segmentation method using a DL approach, specifically the Generative Adversarial Networks architecture [141]. Other studies have shown that beyond simple images, additional modalities can also be utilized for background modeling. For example, Sun et al. proposed a method that leverages RGB-D data for this purpose [142].

One limitation of background modeling methods is that they often require an initial background model, meaning the first few video frames cannot contain any moving objects, which cannot always be ensured in real-life scenarios. Additionally, these methods are typically restricted to situations where the camera is stationary, although some special exceptions exist [143]. Consequently, while background subtraction methods are well-suited to problems like video surveillance, they are not suitable for detecting moving objects based on frames from onboard cameras of mobile robots and autonomous vehicles.

Other methods use edge and boundary detection techniques to track moving objects in a scene [144, 145, 146]. These methods typically calculate the difference between edge maps from consecutive frames to identify moving edges. Additionally, several proposals for predicting object boundaries use DL models [147, 148, 149, 150]. Such techniques can also be employed for object detection and semantic segmentation [151, 152]. Gupta et al. introduced a method for object detection that utilizes contour detection based on RGB-D data, and a deep learning model [153].

Kao et al. [154] proposed a geometric approach for segmenting moving objects, which employs optical flow and estimated depth. In scenarios where the camera is also in motion, such as self-driving cars or mobile robots, the optical flow must be adjusted according to the camera's motion. This issue was thoroughly discussed by Thompson and Pong [155]. To tackle this challenge, Bideau et al. proposed a hierarchical method that first employs an optical flow-based geometric approach, followed by a convolutional neural network for motion segmentation [156].

A method proposed by Bors and Pitas [157] utilizes a median radial basis function network for optical flow estimation, and moving object segmentation is carried out based on the smooth optical flow predicted by their model. Joint estimation for optical flow and moving object segmentation has been investigated by Junhwa, and Stefan [158]. Fragkiadaki et al. proposed a method for moving object detection based on optical flow boundaries and DL [159]. Their approach employs a two-stream convolutional network that utilizes RGB and optical flow data to generate a moving objectness score for the given region of a bounding box, with moving objectness proposals originating from the optical flow boundaries.

Zhou et al. proposed a method for zero-shot video object segmentation called MATNet [160]. The approach utilizes a two-stream encoder to process both RGB video frames and optical flow data and a decoder that considers predicted object boundaries. To obtain optical flow, the PWC-Net, a convolutional neural network, is employed [161]. Additionally, the method incorporates a Motion Attentive Transition (MAT) block, which enables the encoder to learn interleaved motion and appearance representation, preventing overfitting on object appearance [160].

To prevent overfitting when training complex deep learning models on small, specialized datasets, it is recommended to utilize pre-trained networks with transfer learning. While numerous pre-trained feature extractors exist for RGB image data, using other modalities with a close connection to visual information, such as depth, optical flow, or surface normals, can sometimes be a better choice, as demonstrated by many of these previous approaches. However, incorporating additional modalities, such as optical flow data, into transfer learning can be challenging. To address this issue, we propose a method to utilize feature extractors, which were pre-trained on RGB image data, in domains where modalities other than RGB might be better suited, by taking advantage of a cross-modal mapping for the inputs. To demonstrate this method, we introduce Optical Flow Segmentation Network (OFSNet), a deep neural network designed for moving object segmentation in video sequences for mobile robot navigation, which exploits optical flow information and transfer learning with RGB pre-trained feature extractors.

3.1.2 Methodology

For DL-based semantic image segmentation, pre-trained networks are commonly utilized. In their research ([162]), Long et al. proposed the use of "fully-convolutional" neural network architectures for image segmentation, which employed pre-trained models such as AlexNet from [7], VGG net from [52], and GoogleNet [126]. Their approach showed that fully convolutional architectures result in a flexible model that can process inputs of arbitrary sizes while achieving state-of-the-art accuracy and speed. The authors also introduced the skip architecture, which combines deep semantic information with shallow appearance information to produce refined segmentation results. The U-Net model, initially developed for semantic segmentation in biomedical fields [73], is a popular approach that employs a fully convolutional network architecture and skip connections. The architecture comprises a compressing and expanding part, where the compressing part is a traditional convolutional feature extractor, and the expanding part applies inverse convolution operations symmetrically to the compressing part. Additionally, the U-Net model employs skip connections, which helps to smooth the loss function's surface and decrease

the likelihood of becoming stuck in a suboptimal solution [163]. Networks with this structure are commonly referred to as U-Net variants. In our experiments, we also utilize the fully convolutional U-Net with the pre-trained ResNet50 and ResNet152 feature extractors [164]. Furthermore, our proposed model for moving object segmentation in mobile robotics, OFSNet, is based on the U-Net architecture and can also be considered a U-Net variant.

However, for detecting motion in video sequences, processing individual still frames is usually insufficient. Therefore, temporal information should also be considered. DL-based methods for detecting moving objects typically utilize temporal information during the prediction process [165, 166, 167, 168]. In order to train these models, datasets containing labels for motion detection and moving object segmentation are necessary. The KITTI [93] and DAVIS [169] datasets are two of the most commonly used for this purpose. These datasets include video sequences annotated with segmentation mask labels. The KITTI dataset primarily focuses on self-driving cars. It, therefore, consists of videos displaying traffic situations, whereas the DAVIS dataset is more diverse, including videos of vehicles, humans, animals, and other objects. In our experiments, we utilize the DAVIS 2016 benchmark dataset.

Acquiring a publicly available dataset for segmenting moving objects in mobile robotics, tailored to a specific task, can pose a significant challenge, often necessitating the development of a customized training dataset for each specific use case. However, creating such a dataset requires manual labeling of video frames, which is best avoided due to the time-intensive nature of the process.

Pathak et al. demonstrated that optical flow can be employed to generate labels to train a model for still image segmentation [170]. Their methodology involves an unsupervised feature learning approach that utilizes optical flow-based segmentation of video sequences known as uNLC (unsupervised Non-Local Consensus Voting) [171]. The unsupervised optical flow-based segmentation serves as a pseudo ground truth to train their feature extractor. To tackle the issue of insufficient labeled training data for moving object segmentation, Bideau et al. proposed MoA-Net. This model employs optical flow-based labeling in a self-supervised manner [172]. Like these methods, we also utilize an optical flow-based automatic labeling approach (uNLC from [171]) for our experiments. This approach allows us to leverage supervised learning techniques in an unsupervised setting by generating the ground-truth segmentation masks automatically (self-supervision).

3.1.3 Encoding optical flow and grayscale image as RGB data

Eitel et al. presented a two-stream CNN architecture for image recognition that incorporates depth information by encoding it into color data [173]. Similarly, we encode optical flow information in our proposed method using the cross-modal mapping method defined in (3.2).

Our proposed model, OFSNet, is built on top of a pre-trained (Inception v3) feature extractor [51], which was trained for image recognition on real-world RGB images from the ImageNet dataset [50]. To utilize the pre-trained model, we encode the optical flow data and video frames in a way that mimics RGB data, allowing them to be input to the Inception model. We call this encoding a cross-modal mapping as it can be used to create an RGB representation of the non-RGB optical flow modality.

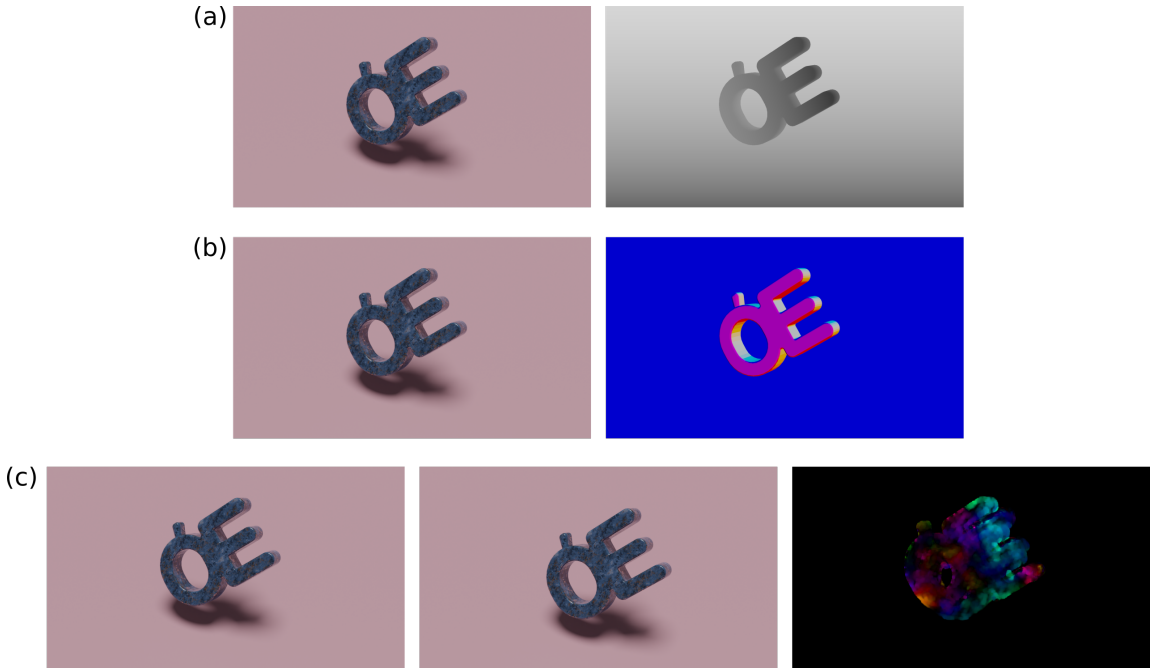


Fig. 3.1. Image representation of non-RGB modalities. **a**: RGB image and corresponding depth data as a grayscale image, **b**: RGB image and corresponding surface normals as RGB image, **c**: Consecutive RGB frames and corresponding optical flow as RGB image (using the 2D polar color map method)

Vision-related modalities that are not RGB, often have a conventional representation and are frequently visualized as images or in conjunction with them. For instance, depth data is typically presented as a grayscale image where the pixel intensity indicates the distance between the surface and the camera. Surface normals are represented by 3D vectors, which, for visualization purposes, are interpreted as red, green, and blue channels to form an RGB image. Optical flow data comprises 2D vectors depicting pixel movement between two frames. It is often represented by 2D arrows drawn over images, symbolizing the optical flow vectors. Other methods represent optical flow by assigning colors to each pixel based on the direction and magnitude of the optical flow vector at that pixel, using a 2D polar color map. Figure 3.1. shows some examples of these representations.

Such encoding methods are frequently used for visualization purposes. Understanding the semantic meaning of these representations makes it easy to evaluate the information conveyed by different modalities. For instance, Figure 3.1. demonstrates that the object in the depth image is closer to the camera than any other portion of the image. The surface normals reveal that the background is horizontal (indicated by only a 'Z' component, rendering it blue). The optical flow visualization indicates that only the object was in motion between the two frames, and it was approaching the camera (opposite sides of the object move in opposite directions, causing it to appear larger). By converting non-RGB modalities into RGB representations using such encoding techniques, we can obtain a visual presentation of additional information that is not immediately apparent from RGB data alone. As we interpret these representations as (RGB) images, identifying edges, shapes, patches, colors, etc., and use our additional semantic knowledge to decode their meaning, we suggest that pre-trained RGB feature extractors learn features that are general enough to be utilized in a transfer learning scenario with such cross-modal mapped representa-

tions. During the fine-tuning phase of the transfer learning process, the network head's objective is to learn how to leverage the additional semantic information provided by the non-RGB modality.

In order to confirm our hypothesis, we assess the accuracy of the predictions made by OFSNet and different U-Net variants when segmenting moving objects in video sequences. To accomplish this, we utilize pre-trained RGB feature extractors and apply transfer learning using cross-modal mapped optical flow and grayscale image data.

For feature extraction, OFSNet employs the pre-trained Inception v3 model, which is specifically designed to process RGB images with a resolution of 299x299 pixels. Prior to being fed into the model, the video frames undergo a series of preprocessing steps. Firstly, they are cropped to a rectangular shape and resized to 299x299 pixels using bilinear interpolation. Subsequently, the optical flow is calculated between every two consecutive frames, and its horizontal and vertical component is mapped onto the R and G image channels, while a monochrome representation of the latter frame corresponds to the B channel of the resulting RGB image. In order to ensure that all values fall within the range of zero to one, a channel-wise normalization is applied. The resulting structures contain three channels, which can be interpreted as the R, G, and B channels of an image. Figure 3.2. demonstrates some of the inputs generated using this approach for a short sequence from both our own and the DAVIS 2016 dataset.

We use Gunnar Farneback's optical flow computation technique to create the cross-modal mapped inputs due to its high level of accuracy and favorable computational performance [174]. The Farneback algorithm is a two-frame optical flow calculation technique that employs polynomial expansion to approximate the neighborhoods of image pixels. This approximation is achieved using quadratic polynomials, which represent the local signal model in a local coordinate system [174]. The Farneback optical flow method generates an image pyramid consisting of multiple resolution levels, with point tracking initiated at the lowest resolution level and continued until convergence. Consequently, the method can accommodate larger pixel motions. In our experiments, an image scale of 0.5 was used, resulting in pyramid layers that were half the size of the preceding one, following the classical pyramid approach. Three layers were constructed during the pyramid-building process.

For the formalized mathematical description of the cross-modal mapping for the input formulation, we describe images as third-order tensors of pixel intensities ($\mathcal{I}$), where the dimensions of the tensor are represented by w, h and c which stand for image width, height and the number of channels respectively. x_{ijk} denotes the intensity of the pixel in the i^{th} column, and j^{th} row for the k^{th} channel. The tensor slice, which contains all pixel intensities for one channel (k), is denoted as $\mathcal{I}_{::k}$. Thus, an RGB image would be represented as a third-order tensor ($\mathcal{I}^{RGB}$) with w and h determined by the image resolution and $c = 3$ for the three channels. $\mathcal{I}^{RGB}_{::1}$, $\mathcal{I}^{RGB}_{::2}$, and $\mathcal{I}^{RGB}_{::3}$ stand for the R (red), G (green) and B (blue) intensities respectively. In the case of monochrome images ($\mathcal{I}^{Y}$), such as a grayscale image, due to $c = 1$, $\mathcal{I}^{Y}$ becomes a second-order tensor. Similarly, optical flow can be represented as a third-order tensor ($\mathcal{F}$), with dimensions of w, h, and $c = 2$, where the slices $\mathcal{F}_{::1}$ and $\mathcal{F}_{::2}$ stand for the horizontal and vertical components of the optical flow vectors for each pixel.

We use the OpenCV library [175] for image processing functionalities such as cropping, rescaling, RGB to grayscale conversion, and optical flow computation with the

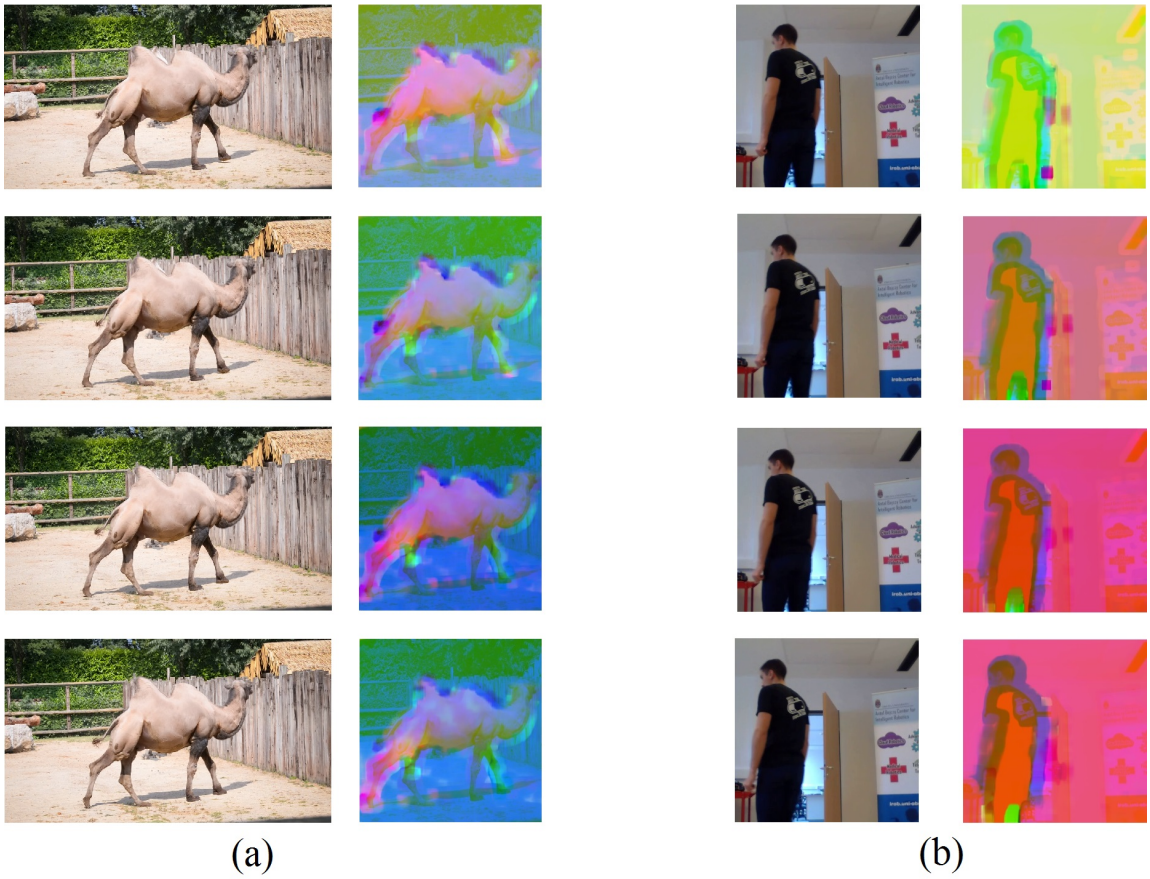


Fig. 3.2. Inputs formed from optical flow and a grayscale frame **a:** Left column: DAVIS 2016 validation set camel sequence frames, Right column: The corresponding cross-modal mapped inputs. **b**: Left column: Frames from a video in our own dataset, Right column: The cross-modal mapped inputs corresponding to the provided frames.

Farneback method. The optical flow computation can be carried out on two consecutive grayscale-converted frames. The RGB to grayscale conversion method used by OpenCV is described in the CCIR 601 standard [176] according to

$$\mathcal{I}^{Y} = 0.299\mathcal{I}^{RGB}_{::1} + 0.587\mathcal{I}^{RGB}_{::2} + 0.114\mathcal{I}^{RGB}_{::3} \tag{3.1}$$

Given the computed optical flow ($\mathcal{F}$) and the corresponding later RGB frame converted

into a grayscale image ($\mathcal{I}^Y$) using (3.1), the cross-modal mapping for combining optical flow and a grayscale frame can be described as

$$\begin{aligned}
\hat{R} &= \mathcal{F}_{::1} + abs(min(\mathcal{F}_{::1}))\mathbf{J}^{w\times h}) \\
\hat{G} &= \mathcal{F}_{::2} + abs(min(\mathcal{F}_{::2}))\mathbf{J}^{w\times h}) \\
\mathcal{I}^{RGB}_{::1} &= \frac{\hat{R}}{max(\hat{R})} \\
\mathcal{I}^{RGB}_{::2} &= \frac{\hat{G}}{max(\hat{G})} \\
\mathcal{I}^{RGB}_{::3} &= \frac{\mathcal{I}^Y}{max(\mathcal{I}^Y)}
\end{aligned} \tag{3.2}$$

where $\mathcal{I}^{RGB}$ is the cross-modal mapped RGB representation of the combined optical flow and grayscale modalities, $\mathbf{J}^{w\times h}$ is a $w \times h$ sized matrix of ones, the functions $min()$ and $max()$ give the minimum, and the maximum valued elements in a matrix respectively, and $abs()$ computes the absolute value element-wise.

The cross-modal mapped inputs generated by (3.2) consist of the R and G channels, which correspond to the normalized horizontal and vertical channels of the optical flow, respectively, and the B channel which is dedicated to the monochrome (grayscale) image. As depicted in Figure 3.2., the color tone of the consecutive inputs may change dramatically as a result of channel-wise normalization. During normalization, all optical flow vectors are offset and rescaled independently along both dimensions. As a consequence, the pixel with the greatest displacement towards the left of the image has an R channel intensity of 0, while the pixel with the greatest displacement towards the right side has an intensity of 1. This implies that if an object is moving from left to right in a sequence, only the object will appear red (have a high intensity in the R channel), while if the object is moving from right to left, everything else will appear red (have a high intensity in the R channel) except for the object. The same principle applies to vertical displacement in the images and the G channel in the cross-modal mapped input, as demonstrated in Figure 3.2. Nonetheless, despite this characteristic, the moving objects do not blend entirely into the background. Due to the optical flow computation and the cross-modal mapping, a well-defined boundary is generated separating the moving objects and the background.

By normalizing the optical flow, we also account for the motion of the camera. When the camera is moving horizontally to the right or turning to the right, the whole visual scene seems to move to the left. In this case, anything moving along with the camera would have a high intensity along its R channel, while other parts wouldn't. The same effect would be observed if a stationary camera captured an object moving horizontally from left to right in the scene. This aspect makes it simpler for the OFSNet model to ignore camera motion. The inclusion of the grayscale frame is another essential aspect that enables the model to detect moving objects by utilizing their appearance. This characteristic improves the predictions' consistency over time, as moving objects do not vanish and reappear from the inputs when there is temporarily no motion in the scene.

Figure 3.3. demonstrates how the proposed cross-modal mapping impacts the inputs when there are different object and camera motions. To incorporate more detail, we utilized a Voronoi texture for the background, which aids in determining whether the camera

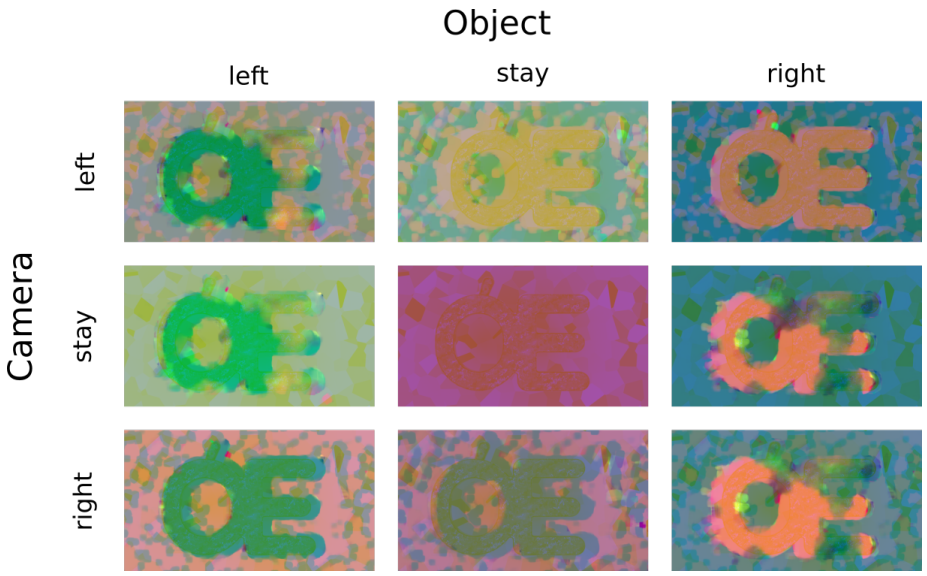


Fig. 3.3. Effect of moving object and camera on the cross-modal mapped inputs (only movements along the horizontal image direction are represented)

or the object is moving. The cross-modal mapped inputs are arranged in a matrix based on the object and camera motion (three options for both: moving left, right, or stationary). The figure depicts that the coloring of the moving object is not significantly influenced by the camera motion. When the object is moving left or right (left and right columns), its coloring remains consistent regardless of the camera's motion (rows). The figure solely illustrates movement in the horizontal direction, hence only exhibiting the effect of motion on the R channel. A limitation of this method to counteract the effects of camera motion using the cross-modal mapping approach is that the optical flow computation may not be precise for large, homogeneous surfaces. Therefore, if objects are in front of such a surface, it may be difficult to distinguish whether the movement in the images is due to camera or object movement. As a result, in our OFSNet experiments with a real mobile robot, we pre-filter the optical flow field to compensate for the motion in the scene caused by the 3D motion of the camera [KAO1].

3.1.4 OSFNet network architecture

The OFSNet model was implemented using TensorFlow's Python API [134] and the Inception v3 pre-trained model for RGB feature extraction [51]. The final classification layer of the Inception v3 model was removed, leaving only the final average pooling layer that yields a volume of 1x1x2048. The upsampled segmentation mask predictions were generated using transposed convolution layers and using skip connections. The OFSNet model performs a single-stage segmentation mask prediction, following the same approach as the U-Net model and its variations [73].

The model's output is a segmentation mask with a resolution of 30x30 pixels, which is intentionally kept low due to the time constraints required for the real-time detection of moving obstacles for mobile robots. This allows the model to achieve a sufficient prediction speed of over 1 frame per second. However, the downside of using a low-resolution

TABLE 3.1
THE STRUCTURE OF THE OFSNET MODEL, INCLUDING THE INCEPTION V3 FEATURE EXTRACTOR FROM #1 TO #12. THE NETWORK STRUCTURE FROM #13 TO #18 IS OUR CONTRIBUTION, AND ONLY THE PARAMETERS OF THIS PART WERE MODIFIED DURING THE TRAINING PROCESS.

#	type	**patch size/stride** or remarks	**input size**
Layers from Inception v3 model			
1	conv	3x3/2	299x299x3
2	conv	3x3/1	149x149x32
3	conv padded	3x3/1	147x147x32
4	pool	3x3/2	147x147x64
5	conv	3x3/1	73x73x64
6	conv	3x3/2	71x71x80
7	conv	3x3/1	35x35x192
8	3 x Inception	see [51] figure 5	35x35x288
9	5 x Inception	see [51] figure 6	17x17x768
10	2 x Inception	see [51] figure 7	8x8x1280
11	pool	8x8	8x8x2048
12	linear	Inception v3 features	1x1x2048
Layers for segmentation			
13	transposed conv	3x3/2	1x1x2048
14	transposed conv	4x4/2	3x3x1280
15	skip connection	#9+#14	8x8x1280
16	transposed conv	16x16/2	8x8x1280
17	linear	logits	30x30x1
18	sigmoid	classifier	30x30x1

output is the loss of detailed information about the shape of the detected obstacles. While this lack of detail may be a limitation for some applications, it is generally acceptable for obstacle avoidance, where the overall size of the obstacle is often more important than its precise shape.

The proposed model's structure is presented in Table 3.1., where each layer's input size is equal to the output size of the previous layer. The last layer (#18) is responsible for performing an element-wise sigmoid activation, resulting in a binary pixel-wise segmentation in a 30x30 image. Here, the intensity of each pixel corresponds to the predicted probability of the pixel belonging to the moving object.

3.1.5 Experimental results

Our experiments demonstrate how large deep learning feature extractors pre-trained on RGB image data can be applied in a transfer learning scenario, even when the target domain does not exclusively employ the same RGB modality. We evaluate our method by comparing U-Net variants trained on cross-modal mapped data to state-of-the-art solutions for moving object segmentation, including a popular unsupervised approach, uNLC, on the DAVIS 2016 dataset, and comparing the performance of the OFSNet model to uNLC on custom real-world data from an industrial Automated Guided Vehicle (AGV) system prototype.

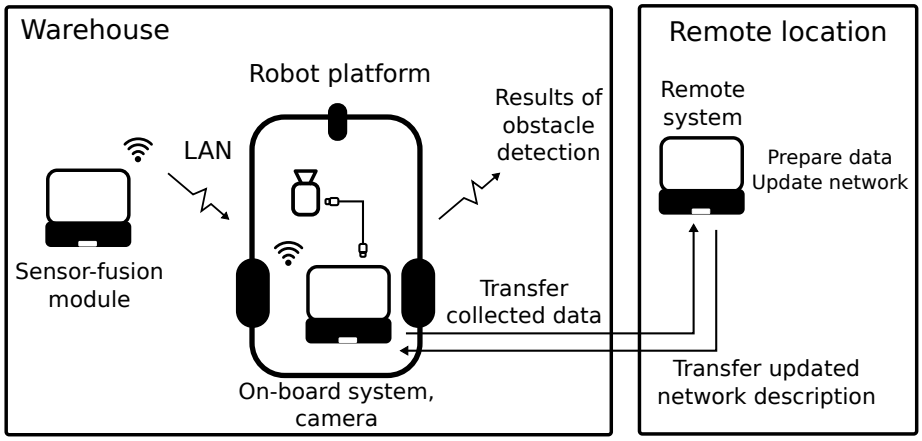


Fig. 3.4. System architecture for data collection using the industrial AGV system prototype

Dataset preparation and training procedure

The training process of OFSNet involved three stages, each using a self-supervised learning scheme that utilized automatically generated segmentation annotations predicted by the uNLC method. This approach was inspired by the work in [170] and [172]. Unlike [170], we treated the output of the uNLC method as the true ground-truth labels instead of interpreting them as pseudo-ground-truth. The first two training stages employed different optimization objectives (loss formulations) using the DAVIS 2016 dataset [169]. The third stage utilized a task-specific dataset, which we refer to as the fine-tuning dataset.

The OFSNet model was trained in separate stages to achieve an initial model that is trained for the general task of moving object segmentation. This initial model could then be specialized for a narrower domain using the fine-tuning dataset. This approach resulted in an OFSNet that is trained for general moving object segmentation, which could be adapted to various domains by repeating the third training stage with different training data. For our mobile robot navigation scenario, we used the fine-tuning dataset to specialize this general OFSNet for moving object segmentation.

The DAVIS 2016 dataset comprises 50 video sequences, which include a total of 3,455 annotated frames, all recorded at 24 frames per second and resolutions of 1080p and 480p. The dataset is divided into training and validation sets, with 30 and 20 video sequences, respectively. For the first two training stages, we used the training set of the DAVIS 2016 dataset at 480p resolution to train the models, and we evaluated their performance on the 480p resolution validation set.

The fine-tuning dataset was gathered from two sources. The first, which we call Lab-Data, was recorded in our laboratory using the built-in webcam of a notebook attached to a moving platform and positioned 1 meter above the ground, facing horizontally. It contains short video sequences, ranging from 1 to 5 seconds in length, of moving obstacles, such as humans and rolling chairs, captured with the moving camera. The obstacles moved with velocities between 0 and 5 meters per second relative to the ground. In total, we recorded 16 video sequences containing 2,154 frames, which were annotated with segmentation masks using the uNLC method. These 16 sequences also included some in which only the camera was moving, with no moving objects in the scene. For these sequences, a segmen-

Fig. 3.5. Ground-truth segmentation masks generated by the uNLC method for the FieldData set

tation map consisting of zeros was used as the ground truth for all frames. The proportion of such frames in the LabData set is roughly 30%.

The other source of data for the fine-tuning dataset is an industrial AGV system prototype. The data collection system architecture is demonstrated by Figure 3.4. With the depicted data collection setup, *bag* files were recorded containing RGBD video sequences from the onboard camera and motion information of the robot platform provided by the sensor-fusion module. A total of 33 video sequences, each consisting of 100 frames, were recorded. Prior to cross-modal mapping, the corresponding depth and robot motion data were utilized for optical flow compensation to account for camera motion. This dataset, which we refer to as FieldData, was also annotated with segmentation masks utilizing the uNLC method. Empty segmentation masks were included for sequences that lacked moving obstacles. (Similar to the LabData, roughly 30% of frames did not contain any moving obstacles.) Ground-truth masks generated by the uNLC method for some frames of the FieldData set are shown in Figure 3.5.

To create the fine-tuning dataset, we combined the LabData and FieldData sets. For model evaluation, we selected 17 video sequences from the FieldData set as the validation set. The remaining FieldData and all of the LabData were used for training, resulting in 3,754 labeled frames for training and 1,700 for evaluation. Figure 3.6. displays the physical AGV setup with the onboard camera, which served as the testing and implementation site for the final OFSNet.

Additionally, we manually labeled a small, randomly selected subset of the fine-tuning dataset to provide a quantitative evaluation of the trained OFSNet model and to compare

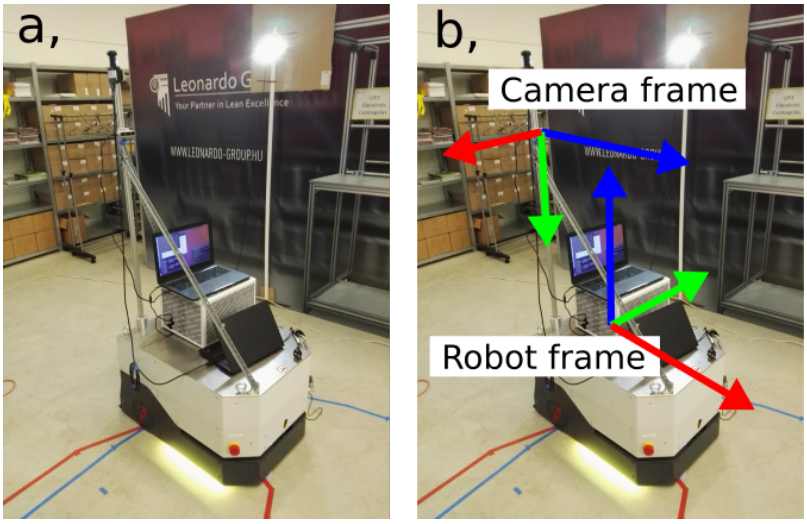


Fig. 3.6. Testing AGV setup. **a**: Robot platform, camera and test environment, **b**: Camera fixture

its performance to the uNLC method. We refer to this manually labeled subset as the testing set.

During all three training stages, we employed mini-batch training with a batch size of 20. The mini-batch elements were selected randomly from cross-modal mapped inputs of the corresponding dataset (DAVIS 2016 for stages 1 and 2 and fine-tuning dataset for stage 3). We also utilized data augmentation by randomly altering the saturation and contrast of the inputs and flipping them horizontally. In all training stages, we used the Adam optimizer [177] with a learning rate of 0.001.

During the first stage, we trained the network on the DAVIS 2016 training set exclusively using Cross-Entropy loss for 20,000 iterations. During the second stage, we continued to train the network on the DAVIS 2016 training data, but for 5,000 iterations using the compound loss function described in (3.7). In the third stage, we trained the model on the fine-tuning dataset for 5,000 iterations using the compound loss from (3.7).

DAVIS 2016 results

In order to assess the effectiveness of the cross modal mapping method for transfer learning when training models for general moving object detection, we evaluated OFSNet and U-Net variants' performance on the validation set of the DAVIS 2016 dataset using the recommended benchmark metrics by [169]. Our results suggest that the general OFSNet is not as effective as other state-of-the-art methods for general moving object segmentation according to the metrics. However, it is important to note that the segmentation maps generated by OFSNet have a resolution of 30x30 pixels, which were then upscaled and padded to match the resolution of the DAVIS 2016 ground truth segmentation maps, which have a resolution of 854x480 pixels, for evaluation purposes. Because the resolution of the OFSNet output is significantly lower than other methods, it would be unfair to make direct comparisons between them. Nonetheless, we have included these results for the sake of completeness.

Additionally, to showcase the effectiveness of the cross-modal mapping approach, we decided to train and evaluate three U-Net variants, namely U-Net-F, U-Net-PWC, and

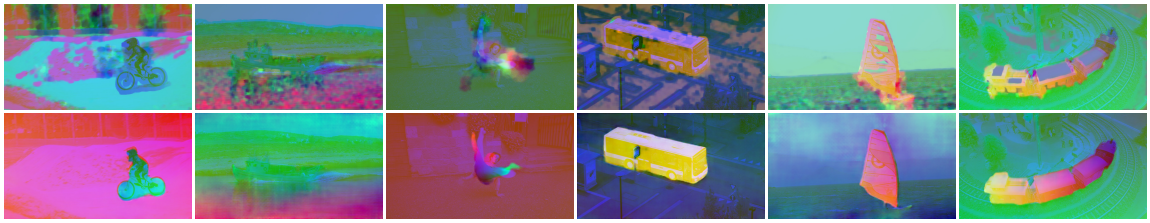


Fig. 3.7. Comparison of cross-modal mapped inputs using the Farenback and PWC-Net methods for optical flow computation. Top row: cross-modal mapped inputs using the Farenback optical flow computation method. Bottom row: cross-modal mapped inputs using PWC-Net for optical flow computation

U-Net++PWC with an output resolution of 864x480, rather than comparing the OFS-Net model to other state-of-the-art moving object detection approaches. U-Net-F and U-Net-PWC follow the standard U-Net architecture according to [73], while U-Net++PWC uses the U-Net++ architecture from [178]. Both U-Net-F and U-Net-PWC incorporate a pre-trained ResNet50 feature extractor backbone, and U-Net++PWC uses the pre-trained ResNet152. All three models use cross-modal mapped inputs (without cropping and re-scaling). The U-Net and U-Net++ network implementations from [179] were used for the models, which were trained on the DAVIS 2016 dataset's training set. U-Net-F and U-Net-PWC used the same self-supervised training approach as OFSNet (i.e., minibatch, number of training iterations, loss function, learning rate, optimizer, etc.), while U-Net++PWC was trained using the ground-truth segmentation masks rather than the uNLC predicted ones.

To compute the optical flow, we utilized the Farneback method for U-Net-F, while for U-Net-PWC and U-Net++PWC, we employed the PWC-Net [161]. The PWC-Net is computationally more demanding than the Farneback method, but it generates a smoother optical flow field. For general moving object detection tasks, there are no strict time requirements as with moving obstacle detection for mobile robotics, so the additional computations required for using PWC-Net are an acceptable tradeoff for better accuracy. It is worth noting that Zhou et al. also utilized PWC-Net for predicting optical flow in their moving object segmentation solution, MATNet [160]. Figure 3.7. demonstrates the differences between cross-modal mapped inputs using the Farneback and the PWC-Net optical flow computation methods. The superiority of the PWC-Net predicted optical flow over the Farneback method for cross-modal mapped inputs is apparent in how the moving objects are usually more distinctly recognizable. The improved quality of the optical flow employed in the cross-modal mapped inputs is the reason why U-Net-PWC performs better than U-Net-F and even the uNLC method in certain evaluation categories.

Table 3.2. presents the quantitative evaluation results of the uNLC, OFSNet, U-Net-F, U-Net-PWC, U-Net++PWC, SegFlow (from [180]) and MATNet methods on the DAVIS 2016 benchmark. The J, F, and T metrics were adapted from [169], where J (Jaccard index) is defined as the intersection over union for the predicted and ground truth segmentation maps and assesses the region similarity, while F (F-measure) measures the contour accuracy and T is the temporal stability. The metrics with an upward arrow indicate that higher values are preferred, whereas those with a downward-pointing arrow suggest that lower values are desirable.

The evaluation results indicate that the OFSNet model does not achieve state-of-the-

TABLE 3.2
EVALUATION OF OUR MODEL (OFSNET), U-NET VARIANTS TRAINED WITH CROSS-MODAL MAPPED INPUTS AND OTHER STATE-OF-THE-ART METHODS ON THE DAVIS 2016 BECHMARK

	$\uparrow J_{mean}$	$\uparrow J_{recall}$	$\downarrow J_{decay}$	$\uparrow F_{mean}$	$\uparrow F_{recall}$	$\downarrow F_{decay}$	$\downarrow T$
uNLC	0.551	0.558	0.126	0.523	0.519	0.114	0.523
OFSNet	0.244	0.088	-0.036	0.150	0.000	-0.003	0.150
U-Net-F	0.483	0.527	**0.039**	0.461	0.458	**0.059**	**0.461**
U-Net-PWC	0.505	**0.584**	**0.079**	**0.548**	**0.643**	**0.058**	0.548
U-Net++PWC	**0.618**	**0.736**	**0.039**	**0.653**	**0.797**	**0.056**	0.653
SegFlow	0.674	0.814	0.062	0.667	0.771	0.051	0.282
MATNet	0.824	0.945	0.055	0.807	0.902	0.045	0.216

art performance (comparable to uNLC), which is likely due to its network architecture with limited output resolution. In contrast, the U-Net variants, U-Net-F and U-Net-PWC, achieve comparable or even superior results to the uNLC method (bold font indicates superior results), and U-Net++PWC achieves results comparable to SegFlow. This finding suggests that incorporating non-RGB modalities into a transfer learning pipeline via the cross-modal mapping approach is a promising direction to achieve state-of-the-art results even with limited available data. However, the results also show that MATNet, an approach that considers object boundaries during the decoding process, outperforms the U-Net variants that use cross-modal mapping, in all categories. Additionally, it is worth noting that the cross-modal mapping approach used in this experiment only incorporates object appearance and motion information (optical flow). Future work could potentially develop an approach that also includes object boundary information, in a similar fashion to the HED-U-Net approach [181]. Unlike MATNet, U-Net-F and U-Net-PWC utilize automatically generated uNLC segmentations rather than the perfect ground-truth segmentations provided in the DAVIS 2016 dataset, enabling these models to be used in an unsupervised manner through self-supervision.

To qualitatively evaluate the performance of the uNLC, U-Net-F, U-Net-PWC and U-Net++PWC methods, a selection of frames from the validation set of the DAVIS 2016 benchmark, along with the corresponding ground truth segmentation masks and prediction results, are shown in Figure 3.8.

Real-world mobile robot navigation results

We evaluated the performance of the OFSNet model by comparing it with uNLC on our manually labeled testing set. The evaluation results are summarized in Table 3.3. The evaluation metrics used were PA (average pixel-wise accuracy), SA (average ratio of the size of predicted and ground-truth object masks in pixel units), DoC (average distance of the center of masses of predicted and ground truth masks in pixel units), Dice coefficient, precision, recall, and TS (temporal stability). TS was computed as the rational size change of the predicted object masks from frame to frame (ranging from 0 to 1, with values closer to 1 being better). The SA, DoC, and Dice metrics were computed only on frames that contain moving obstacles. In contrast, precision and recall were computed for the background, which allowed us to include frames without moving obstacles. The TS measure was computed for the whole validation set of the fine-tuning dataset without the

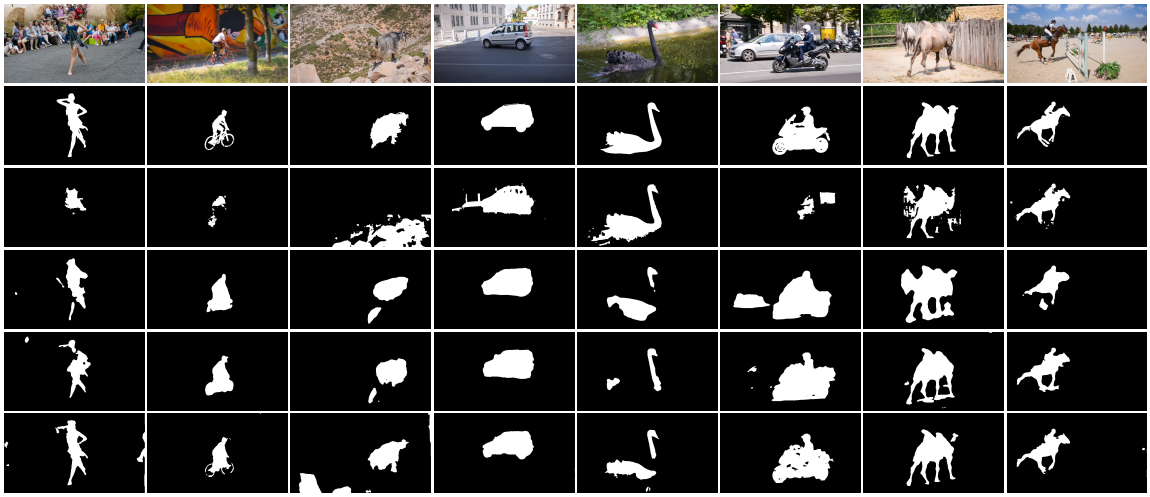


Fig. 3.8. Qualitative evaluation of uNLC, U-Net-F, U-Net-PWC and U-Net++PWC predictions. Rows from top to bottom: RGB frames, ground truth segmentation masks, uNLC predictions, U-Net-F predictions, U-Net-PWC predictions, U-Net++PWC predictions

TABLE 3.3
EVALUATION OF THE OFSNET AND UNLC MODELS ON OUR MANUALLY LABELED TEST SET

	↑ PA	↑ SA	↓ DoC	↑ Dice	↑ Prec., Recall	TS
uNLC	0.763	0.237	6.575	0.278	0.801, 0.941	0.895
OFSNet	0.827	0.406	4.757	0.374	0.822, 0.978	0.945

need for manually labeled frames since it does not require ground truth segmentations for its computation.

The evaluation was conducted using the uNLC predictions that were cropped to a rectangular shape and rescaled to 30x30 pixels. The results of the evaluation demonstrate that OFSNet outperformed the uNLC method in all categories. To provide a better insight into the performance of OFSNet, we included the predicted segmentation masks of uNLC and OFSNet for several frames from our fine-tuning dataset for qualitative assessment in Figure 3.9.

Based on the results, it is evident that OFSNet is incapable of segmenting small objects or capturing fine details. Still, it can effectively handle camera motion and larger moving objects, such as humans. In mobile robot navigation, detecting moving objects is critical for dynamic obstacle avoidance, and the model's inability to capture fine details is of little significance. The labels were generated using the uNLC method, which only expects a single moving object in the scene. This explains why OFSNet struggles to accurately segment multiple moving objects in a single frame. With more accurate labeling, the model could potentially be improved to better segment multiple moving objects. The success of the OFSNet model can be attributed to its ability to utilize a pre-trained RGB feature extractor through our proposed cross-modal mapping approach. Future research should focus on developing a network architecture capable of generating high-resolution segmentation masks at a faster rate and incorporating additional information into the cross-modal mapping, such as object boundaries and depth data.

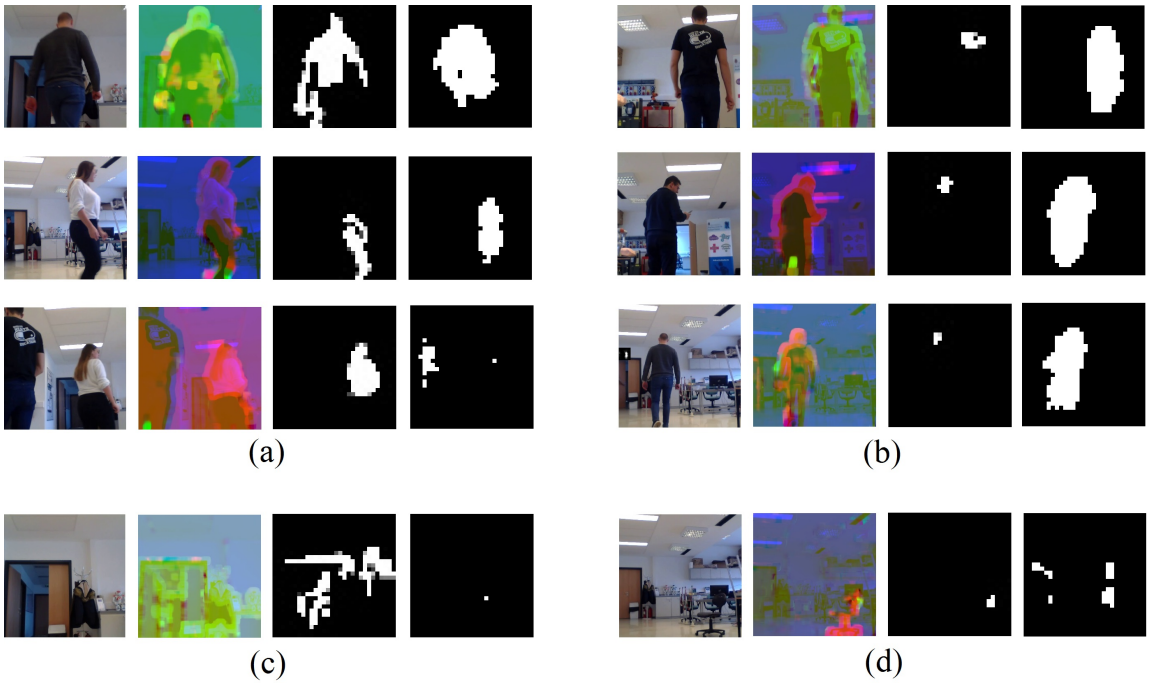


Fig. 3.9. Results of OFSNet on our fine-tuning dataset compared to the uNLC method that was used for the automatic labeling. The left columns depict the frames of the video sequence, the images in the second columns are the corresponding cross-modal mapped inputs, the third columns are the uNLC results, and the fourth columns are the predicted segmentations from OFSNet. **a**: The uNLC method produces finer segmentation maps, and both methods fail to capture multiple moving objects (third row). **b**: Sometimes, uNLC fails to segment the whole object, but OFSNet does it successfully. **c**: OFSNet can compensate for the motion of the camera; if there is no moving object on the scene, the prediction is a map of zeros. **d**: The uNLC method can better segment smaller objects, but OFSNet fails to do so.

3.2 Compound loss for mitigating class imbalance

3.2.1 Motivation

In the process of training OFSNet, we discovered that the class imbalance created by the relatively small size of the moving objects in the training dataset resulted in a tendency for the predicted masks to be biased toward the background when we employed the Cross-Entropy loss function. The Cross-Entropy loss for a binary segmentation problem for a single frame can be expressed as follows:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} y_i log_2(p_i) + (1-y_i) log_2(1-p_i), \tag{3.3}$$

where L_{CE} is the Cross-Entropy loss for a single frame, and N represents the total number of pixels in the output, which is 900 in our specific case (for the OFSNet model). The correct label for each pixel is represented by y_i, where a value of 0 denotes the background, and a value of 1 denotes the moving object. The predicted probability of the i^{th} pixel belonging to the moving object is represented by p_i.

In situations where the moving object only occupies a small portion of the frame, if the model predicts that all pixels belong to the background, the calculated loss will be lower compared to when it predicts that they all belong to the moving object. This indicates that if we solely rely on Cross-Entropy loss, the model will have a tendency to favor background predictions rather than foreground predictions for a specific output pixel.

To address this issue, there are two straightforward approaches. The first one involves implementing a weighted Cross-Entropy loss [73], while the second approach involves directly optimizing a measure that prevents this bias. One such measure is the Sørensen-Dice coefficient or the Dice coefficient [182, 183]. This coefficient is utilized for measuring the overlap between two sets and ranges from 0 to 1. A value of 1 indicates that there is perfect overlap between the two sets, while a value of 0 means that there is no overlap. The Dice coefficient can be computed for two sets, A and B, in the following manner:

$$Dice = \frac{2|A \cap B|}{|A| + |B|}, \tag{3.4}$$

where $|A \cap B|$ means the common elements of A and B and $|A|$ and $|B|$ are the number of elements in A and B, respectively.

In our segmentation task, our objective is to maximize the Dice coefficient between the ground-truth mask and the predicted mask. To achieve this, we can employ the Soft Dice coefficient.

$$SoftDice = \frac{2\sum_{i=1}^{N} y_i p_i}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} y_i}, \tag{3.5}$$

where N denotes the total number of pixels in the output, while y_i represents the accurate label for the i^{th} output pixel. In our case, a value of 0 represents the background, and a value of 1 represents the foreground, which in our context, corresponds to the moving

object. The predicted probability of the i^{th} output pixel belonging to the moving object is represented by p_i.

From (3.5), a loss function can be formulated, which is referred to as Soft Dice loss (L_{SD}):

$$L_{SD} = 1 - SoftDice. \tag{3.6}$$

Optimizing for the Soft Dice loss ensures maximum overlap between the ground truth and the predicted segmentation, and its value does not depend on the image resolution.

3.2.2 Loss formulation

The proposed compound loss is the linear combination of the Cross-Entropy loss and the Soft Dice loss:

$$L = (1 - \alpha)L_{CE} + \alpha L_{SD}, \tag{3.7}$$

where the weight factor α is utilized to modulate the relative importance of the two methods.

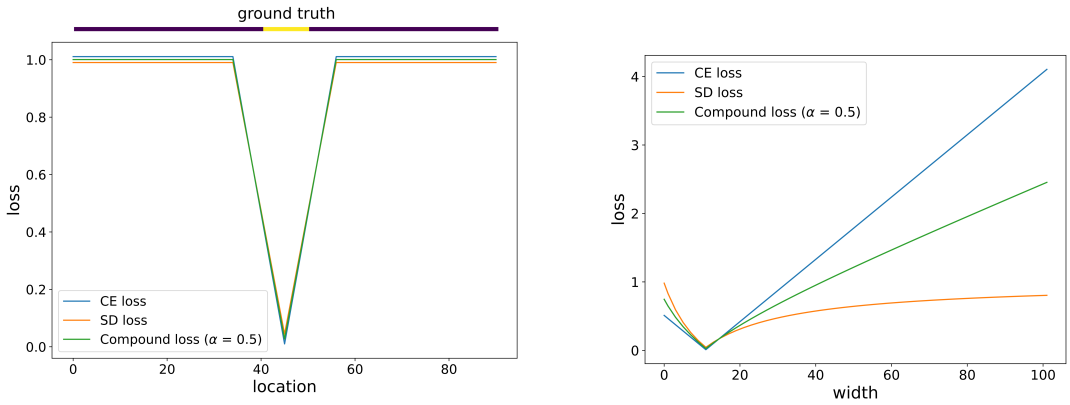


Fig. 3.10. Comparison of Cross-Entropy (CE), Soft Dice (SD) and Compound losses

The compound loss involves leveraging the Soft Dice loss to counteract the bias toward predicting the background that arises due to the class imbalance and the Cross-Entropy loss. This is achievable because the Dice loss attains its highest value when all pixels are predicted as background. However, if all pixels are predicted to belong to the moving object, the Soft Dice loss will be less than its maximum value. Thus, it penalizes false negative predictions more severely than false positives.

Figures 3.10. and 3.11. exhibit the aforementioned phenomenon with a simple 1D mask example. The ground truth 1D mask is displayed on top of the left-hand side graph in Figure 3.10., with yellow pixels representing the foreground and purple pixels representing the background. The ground truth mask comprises a 1D array of 100 pixels, with an 11-pixel-long foreground segment located precisely in the middle of the array. The left-hand graph illustrates the values of the Cross-Entropy, Soft Dice, and compound loss functions ($\alpha = 0.5$ for the compound loss) for predictions in which a foreground segment of the

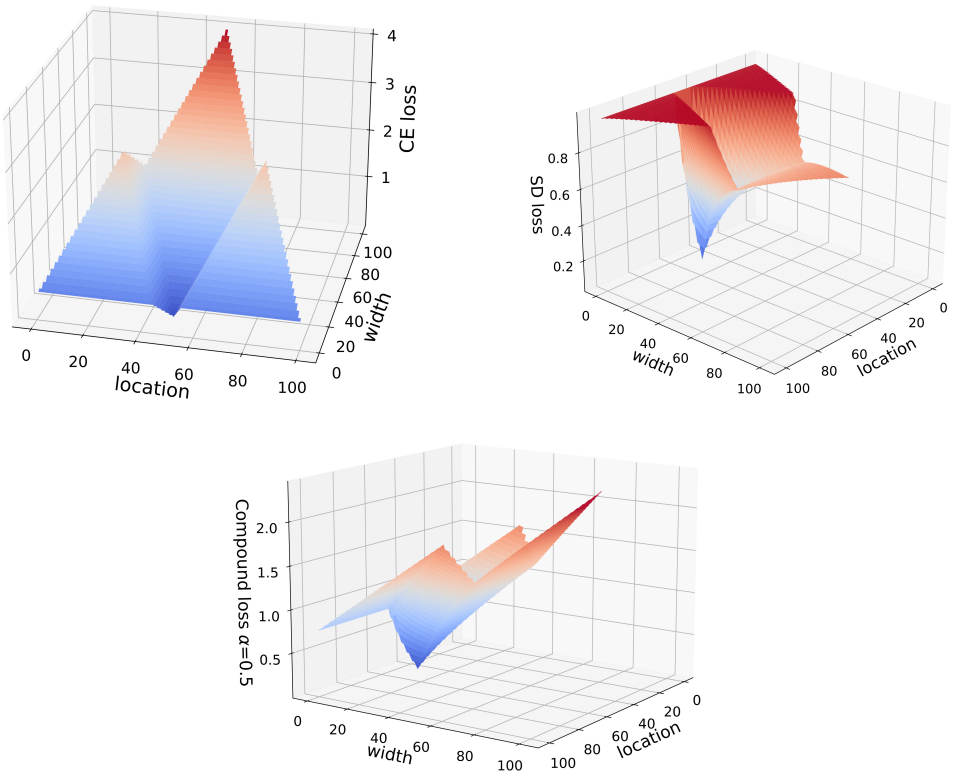


Fig. 3.11. Comparison of Cross-Entropy, Soft-Dice and Compound losses

same size (11 pixels) is shifted across the 1D array. The values on the horizontal axis denote the position of the center of the 11-pixel-long positive segment in the predictions.

From the graph, it is evident that all three loss functions treat object localization in a similar manner. However, the right-hand-side graph exposes the primary difference between the Cross-Entropy and Soft Dice losses and highlights the advantages of utilizing the compound loss. In this case, instead of shifting the foreground segment in the predictions, it is centered in the middle of the 1D array, and only its width is adjusted. The graph shows the steep increase in the Cross-Entropy loss for false positive pixel predictions, whereas, for false negative predictions, the Soft Dice loss produces a much steeper curve. The compound loss, based on the value of the α parameter, can capitalize on both of these desired characteristics. It no longer has a plateau for false positive predictions, as is the case with the Soft Dice loss, which would negatively affect the gradient magnitude and, consequently, the training process. It also penalizes false negative predictions more than the Cross-Entropy loss.

In Figure 3.11., the values of the Cross-Entropy, Soft Dice, and compound losses are shown for all possible combinations of the location and width of the predicted foreground segment. One can observe that the Cross-Entropy loss tends to slope globally towards smaller widths, whereas the Soft Dice loss attains its maximum value in the same region. The compound loss, on the other hand, exhibits a more distinct minimum compared to the Cross-Entropy loss while still having a steep surface for large predicted widths. It is

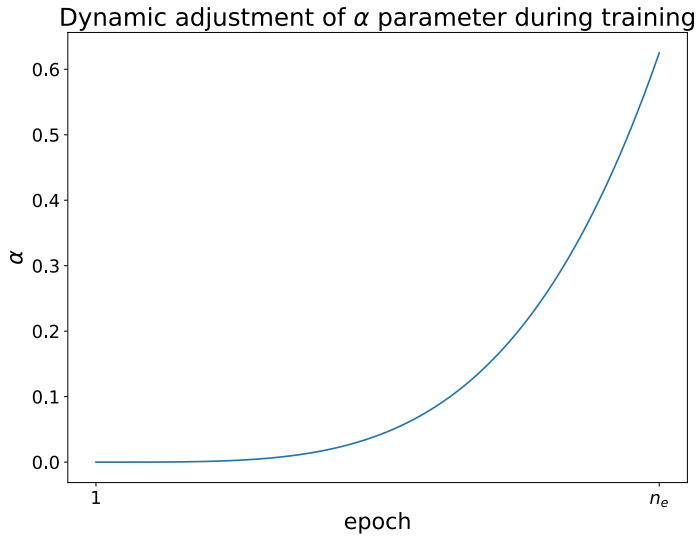


Fig. 3.12. Training strategy: weighting parameter (α) development over the training process

important to note that, in reality, the predictions can have disjoint regions as well, and the loss surfaces tend to be much more complex in higher dimensions, depending on the number of trainable parameters. Nonetheless, this simple example provides an intuitive illustration of the Cross-Entropy loss's bias toward predicting background and the rationale behind the proposed compound loss function.

In our experiments, we observed that assigning a weight value larger than 0.7 to α can lead to unstable optimization for the Soft Dice loss. As a result, we devised an empirical technique that employs the Cross-Entropy loss at the initial stages of the training process to provide an approximate localization of the moving objects. After this step, we apply the Soft Dice loss to maximize the overlap between the predicted and ground-truth masks. Therefore, we continuously modify the value of α during the training process. The value of α is determined as follows:

$$\alpha = \frac{\left(\frac{j}{n_e}\right)^4}{1.6}, \tag{3.8}$$

where the value of j represents the current iteration count during training, where n_e is the total number of iterations. To delay the significant use of the Soft Dice loss until later in the training process, the power of 4 is applied. Additionally, to ensure that the value of α remains at or below 0.625, a division by 1.6 is implemented. Figure 3.12. depicts the value of the α parameter throughout the training process, using (3.8).

3.3 New scientific results

Thesis 2

I developed a Deep Learning model (OFSNet), and a corresponding loss function for moving object segmentation in video sequences, enabling moving obstacle avoidance in indoor environments for mobile robot navigation. The proposed method has been validated in a real-world industrial Automated Guided Vehicle (AGV) system prototype.

Sub-thesis 2.1

I demonstrated that feature extractors pre-trained on real-world RGB images can generalize to combined optical flow and grayscale input data with appropriate formatting/cross-modal mapping (3.2). This conclusion was supported by experiments on the DAVIS 2016 dataset and real-world data acquired from an industrial Automated Guided Vehicle (AGV) system prototype.

Sub-thesis 2.2

I introduced a compound loss function (3.7) and a corresponding empirical training approach that utilizes a dynamic linear combination of Cross-Entropy Loss and Soft Dice Loss functions to overcome their counter-effecting biases. The effectiveness of this loss function and training strategy was demonstrated through the training of the OFSNet model. I demonstrated that the weighting parameter (α) can be dynamically adjusted (3.8) to ensure the best overlap between the ground truth and the predicted segmentation masks, even in unbalanced (in terms of object-to-background ratio) datasets.

Related publications: [KA4, KA5, KA6]

Part 4

AUTOMATIC LARGE-SCALE VISUAL DATASET GENERATION

Robots are excellent at carrying out repetitive tasks with high precision. Modern robotic solutions, however, may require high flexibility from the robots as well, such as being able to manipulate randomly organized objects in a cluttered scene. Current state-of-the-art achieves this level of flexibility with the use of machine learning technologies [184, 185, 89, 186]. The most commonly used sensor modality for such tasks is vision. Modern cameras are relatively cheap and small in size, and there are several pre-trained models for visual data readily available.

Typically, pre-trained models are trained on widely used datasets such as ImageNet [50] for image classification, the COCO dataset [187] for object detection and segmentation, or domain-specific datasets like PASCAL-VOC [188], and KITTI [93] for autonomous vehicles. These deep learning models, which have been pre-trained on large-scale datasets, possess a comprehensive understanding of visual scenes and are adaptable to various domains, such as sensory integration [189] and medical applications [190].

Despite their versatility, pre-trained models may not always be optimal for a specific robotic task. In such cases, additional training is necessary on a smaller dataset that is tailored to the particular problem at hand. Even with deep neural networks, this dataset can still be relatively modest, comprising only a few hundred or a few thousand labeled images for supervised learning. The time required for labeling these images heavily depends on the nature of the task for which the dataset is being prepared. For robotic manipulation, it is essential to have precise information on the location and pose of the objects in the scene. If a DL model is to derive this information from images, semantic segmentation of the scene is typically required. However, manual annotation of an object segmentation dataset comprising hundreds or thousands of images can be exceedingly time-consuming, particularly when multiple objects are present in the scene, which is not uncommon in robotic applications. A generally accepted solution for this problem is the use of synthetic data.

We propose an automated annotation method for object segmentation in real images, specifically tailored for robotics applications. Our method leverages the unique features of robotics, such as access to the robot's pose information and the ability to mount a camera on the robot. Additionally, we take advantage of the fact that objects with known geometries can be placed in predetermined poses within the robot's workspace. Under

these assumptions, we can describe the digital twin [191] of the whole scene with a high degree of accuracy, enabling us to compute virtual camera projections. The virtually projected information augments the actual photos with annotations that make the ML dataset. This approach extends the digital twin paradigm to the field of ML dataset creation and validation.

Furthermore, we propose a synthetic data generation approach using computer graphics to generate photorealistic randomized synthetic image datasets. The synthetic dataset generation procedure employs automated annotation for generating instance segmentation masks. A method called Filling the Reality Gap (FTRG) is also introduced, which uses the proposed automated real data annotation and synthetic data generation processes together, to overcome the reality gap.

Both dataset generation approaches are evaluated qualitatively by creating datasets with them and quantitatively by evaluating neural networks trained on the datasets created by them. The effectiveness of the FTRG method is showcased on a robotic pick-and-place task, and the benefits of using the proposed synthetic data generation pipeline are also highlighted in two additional approaches: using synthetic data during the experience replay of continual learning approaches for scene classification and utilizing the flexibility of synthetic data generation to create task-specific grasp-detection datasets.

4.1 Creating real-life segmentation datasets

4.1.1 Motivation

Training a DL model for object segmentation using a synthetic dataset has numerous advantages. For instance, it is a relatively rapid and inexpensive process, and the annotation can be automated. Additionally, the labels are typically more reliable than those obtained from a public online annotation service. Moreover, with suitable tools, it is possible to simulate a wide range of scenes. Nevertheless, the major disadvantage of a synthetic dataset is that it is not directly derived from the target domain in which the DL model is intended to operate.

According to Tremblay et al. [89], the problem of constructing a synthetic dataset that is effective in the target domain can be mitigated by integrating domain randomization with photorealistic data. Their research demonstrated that utilizing both highly randomized (i.e., non-photorealistic) scenes and photorealistic synthetic images yields superior outcomes when compared to using either of them exclusively. This finding suggests that incorporating real data into the dataset (i.e., mixing synthetic and real data) may further improve the performance of DL models for object segmentation. However, the automated annotation of real samples presents a challenge.

As a result, we propose a method for automatically generating instance segmentation-type annotations for real-world images in a robotic manipulation setting. This allows us to create datasets containing real-world images of objects annotated with instance segmentation masks and use these datasets to directly train DL models or to enhance the performance of models trained on synthetic images by mixing synthetic and real data.

4.1.2 Annotation procedure

The proposed annotation procedure (Algorithm 6.) is capable of automatically generating instance segmentation masks for real images. To achieve this, it utilizes a virtual representation of the significant elements of the real scene, including the camera and objects. This virtual scene is a digital twin of the physical environment, with the virtual camera and objects mirroring the pose of their real counterparts. By doing this, we are able to calculate the segmentation masks for the objects in the virtual scene and then map these annotations to the corresponding images captured by the camera in the real-world scene.

The generation of instance segmentation masks is accomplished by computing the perspective projection of 3D points located on the surfaces of the objects onto the image plane. We follow the formalism described in [192], which defines the perspective projection $\bar{\mathbf{x}} = (u, v, 1)^\top$ of a 3D point ${}^w\mathbf{X} = ({}^wX, {}^wY, {}^wZ, 1)^\top$ (given in the world frame) as:

$$\bar{\mathbf{x}} = \mathbf{K}\mathbf{\Pi}\ {}^c\mathbf{T}_w\ {}^w\mathbf{X}, \tag{4.1}$$

where $\mathbf{K}$ represents the camera matrix, which contains the intrinsic parameters of the camera and can be determined by camera calibration. To accomplish this, we utilize the OpenCV library [175]. The projection matrix $\mathbf{\Pi}$ is in the form of $[\mathbf{I}|\mathbf{0}]$, where $\mathbf{I}$ is a 3×3 identity matrix and $\mathbf{0}$ is a column vector of three zeros. Finally, ${}^c\mathbf{T}_w$ is the 4×4 homogeneous transformation matrix that describes the transformation between the world and the camera frame.

Let $P({}^w\mathbf{X})$ denote the perspective projection of the point ${}^w\mathbf{X}$, F a face defined by a set of points ($F = \{{}^w\mathbf{X}_1, {}^w\mathbf{X}_2, ..., {}^w\mathbf{X}_n\}$), $R({}^w\mathbf{X})$ a ray coming from the origin of the camera frame and going through the point ${}^w\mathbf{X}$, and ${}^w\mathbf{X}^{\mathcal{O}}_{all}$ all the possible points on the surface of object $\mathcal{O}$. To create segmentation masks, a finite set of points on the surface of each object needs to be selected: ${}^w\mathbf{X}^{\mathcal{O}} = \{{}^w\mathbf{X} | {}^w\mathbf{X} \text{ on the surface of } \mathcal{O}\}$, ${}^w\mathbf{X}^{\mathcal{O}} \subseteq {}^w\mathbf{X}^{\mathcal{O}}_{all}$. The power set $\mathbb{P}({}^w\mathbf{X}^{\mathcal{O}})$ contains all the possible (not necessarily meaningful) faces for object $\mathcal{O}$, for a given set of surface points ${}^w\mathbf{X}^{\mathcal{O}}$. Polygons can be formed in the image plane by projecting each point of a face: $Poly^F = \{P({}^w\mathbf{X}_i) \text{ for } {}^w\mathbf{X}_i \in F\}$, and using the projections as the vertices of the polygon. A set of faces $F^{\mathcal{O}} \subseteq \mathbb{P}({}^w\mathbf{X}^{\mathcal{O}})$ have to be chosen for object $\mathcal{O}$, such that all the projections given by $P({}^w\mathbf{X}_j)$ for ${}^w\mathbf{X}_j \in {}^w\mathbf{X}^{\mathcal{O}}_{all}$ fall inside at least one of the polygons of $Poly^{F_k}$, for $F_k \in F^{\mathcal{O}}$, but projections $P({}^w\mathbf{X})$, where $R({}^w\mathbf{X})$ does not intersect the object in 3D space, do not fall into any of the polygons from $Poly^{F_k}$, for $F_k \in F^{\mathcal{O}}$.

For a simple cube, we can select its vertices as the set of surface points (${}^w\mathbf{X}^{\mathcal{O}} =$ vertices), while the set of faces $F^{\mathcal{O}}$ would naturally be the six faces of the cube. Projecting the points of these faces would yield six tetragons in the image plane. It's evident that any point on the surface of the cube would fall inside at least one of these tetragons. Any other point which is not on the surface or inside the cube, and also not between the camera and the cube or behind the cube (so the ray from the origin of the camera frame going through the point does not intersect the object) would get projected outside of all the tetragons. Thus, merging all the tetragons into a single polygon gives us the segmentation mask for the cube.

However, selecting surface points and defining faces manually for complex objects is not practical. Fortunately, the Standard Triangle Language (STL) format is a widely used

virtual representation for 3D object models. This format provides a 3D surface model of the object, represented by an object mesh consisting of triangles formed by vertices. This means we can use the STL representation directly by defining ${}^{w}\mathbf{X}^{\mathcal{O}} = {}^{w}\mathbf{T}_{o}\,{}^{o}\mathbf{X}^{\mathcal{O}}$, where ${}^{o}\mathbf{X}^{\mathcal{O}}$ are the vertices in the STL format defined in the object frame, and ${}^{w}\mathbf{T}_{o}$ is the 4×4 homogeneous transformation matrix that describes the transformation between the object and world frames. $F^{\mathcal{O}}$ can be chosen according to the triangles in the STL representation. In typical robotics scenarios, it can be assumed that a 3D model of the objects is available or can be created with ease. A photogrammetry application can also be used to obtain a 3D mesh, as these methods are increasingly accessible with mobile phones [193]. In our experiments, we utilized Qlone [194] to scan clutter objects. Using the STL representation of the object mesh has the added benefit that it is only necessary to measure the pose of the object frame relative to the world frame to determine ${}^{w}\mathbf{T}_{o}$, instead of measuring every individual point relative to the world frame (${}^{w}\mathbf{X}^{\mathcal{O}}$).

Before annotating the images, it is necessary to determine the transformation matrix ${}^{c}\mathbf{T}_{w}$ from (4.1). In our experiments, we utilize an industrial robot and attach the camera in a known, fixed pose to the robot's end effector. As a result, the transformation between the camera frame and the robot's tool center point (${}^{c}\mathbf{T}_{TCP}$) remains constant regardless of the robot's pose. The transformation between the robot TCP and the world frame (${}^{TCP}\mathbf{T}_{w}$) can be obtained from the robot controller at any given time. Consequently, we can express the transformation between the camera and the world frame as ${}^{c}\mathbf{T}_{w} = {}^{c}\mathbf{T}_{TCP}\,{}^{TCP}\mathbf{T}_{w}$.

Algorithm 6. describes the process of automated annotation. It assigns a unique color ID (RGB values) to each object ($\mathcal{O}_1, \mathcal{O}_2, ...$). The segmentation mask $\mathbf{M}$ used by the algorithm has 5 channels, with the first three representing the object's color ID (RGB) and the remaining two storing information about the triangle and object to which each pixel belongs. The algorithm proceeds by projecting the vertices of each triangle in the objects' mesh in order. Let $\mathcal{O}$ and $\mathcal{T}$ denote the current object and triangle being projected, respectively. For each pixel falling inside the projection of $\mathcal{T}$, the algorithm performs a test with three possible outcomes.

If the pixel is black, it is colored with the color ID of $\mathcal{O}$. If the pixel is already colored with the same color ID as $\mathcal{O}$, it is already marked as belonging to $\mathcal{O}$, so no action is necessary. If the pixel is already colored with the color ID of a different object $\tilde{\mathcal{O}}$, a decision must be made regarding its color. The algorithm accomplishes this by looking up the triangle $\tilde{\mathcal{T}}$ of the object $\tilde{\mathcal{O}}$ based on the last two channels of the mask and calling the *IsOccluded* function (Algorithm 7.) on the two triangles $\mathcal{T}$ and $\tilde{\mathcal{T}}$.

The *IsOccluded* function determines whether $\mathcal{T}$ is occluded by $\tilde{\mathcal{T}}$ by first checking trivialities, such as whether all vertices of one triangle are closer to the camera than the other. If the trivialities do not apply, the algorithm uses the *SignedVolume* function, whose definition is provided in Algorithm 8., to make the decision. If $\mathcal{T}$ is occluded by $\tilde{\mathcal{T}}$, no action is taken. Otherwise, the pixel is colored with the color ID of the current object $\mathcal{O}$.

To account for the lens distortions of the camera, it is necessary to adjust the actual 3D coordinates using the formula provided in [195]:

Algorithm 6: Projection algorithm

input : Image shape: $[w, h, 3]$, List of objects: $\mathbf{O} = [\mathcal{O}_1, \mathcal{O}_2, \dots]$

```
/* Init annotation as black image                                    */
```
Init: $\mathbf{M}$ = $zeros((w, h, 5))$;
for $\mathcal{O} \in \mathbf{O}$ **do**
  for $\mathcal{T} \in \mathcal{O}$.*triangles* **do**
    `/* Projection as in (4.1)                                       */`
    v_1^i, v_2^i, v_3^i = Project($\mathcal{T}$.vertices);
    temp_img = $zeros((w, h))$;
    `/* Get internal pixels of the triangle                         */`
    $\mathbf{P}$ = Where(DrawTriangle(temp_img, (v_1^i, v_2^i, v_3^i), color=1) == 1);
    for $\mathbf{p} \in \mathbf{P}$ **do**
      if $\mathbf{M}[\mathbf{p}][0:3] == [0, 0, 0]$ **then**
        `/* It was background before                                 */`
        $\mathbf{M}[\mathbf{p}][0:3] = \mathcal{O}$.color_id;
        $\mathbf{M}[\mathbf{p}][3] = \mathcal{O}$.id;
        $\mathbf{M}[\mathbf{p}][4] = \mathcal{T}$.id;
      else if $\mathbf{M}[\mathbf{p}][0:3] == \mathcal{O}$.*color_id* **then**
        `/* It is the same object                                   */`
        Pass;
      else
        $\tilde{\mathcal{T}}$ = $\mathbf{O}$.GetTriangle($\mathbf{M}[\mathbf{p}][3], \mathbf{M}[\mathbf{p}][4]$);
        if *IsOccluded*($\mathcal{T}$, *by* $= \tilde{\mathcal{T}}$) **then**
          `/* `$\tilde{\mathcal{T}}$` occludes `$\mathcal{T}$`                                   */`
          Pass;
        else
          $\mathbf{M}[\mathbf{p}][0:3] = \mathcal{O}$.color_id;
          $\mathbf{M}[\mathbf{p}][3] = \mathcal{O}$.id;
          $\mathbf{M}[\mathbf{p}][4] = \mathcal{T}$.id;

return: $\mathbf{M}$

$$
\begin{aligned}
\tilde{x} &= {}^cX/{}^cZ \\
\tilde{y} &= {}^cY/{}^cZ \\
r &= \tilde{x}^2 + \tilde{y}^2 \\
\begin{bmatrix} {}^cX' \\ {}^cY' \\ {}^cZ' \end{bmatrix} &= {}^cZ \begin{bmatrix} \tilde{x}(1 + k_1 r + k_2 r^2 + k_3 r^3) + 2p_1\tilde{x}\tilde{y} + p_2(r + 2\tilde{x}^2) \\ \tilde{y}(1 + k_1 r + k_2 r^2 + k_3 r^3) + 2p_2\tilde{x}\tilde{y} + p_1(r + 2\tilde{x}^2) \\ 1 \end{bmatrix}
\end{aligned}
\tag{4.2}
$$

where the original 3D coordinates of a point with respect to the camera frame are denoted by ${}^cX$, ${}^cY$, and ${}^cZ$. The distortion vector $[k_1, k_2, p_1, p_2, k_3]$, which is determined during the camera calibration process [195], is used to modify the coordinates, and the resulting distortion-corrected 3D coordinates of the point relative to the camera frame are given by ${}^cX'$, ${}^cY'$, and ${}^cZ'$.

By substituting ${}^c\mathbf{X} = {}^c\mathbf{T}_w \; {}^w\mathbf{X}$ into (4.1), where ${}^c\mathbf{X} = ({}^cX, {}^cY, {}^cZ, 1)^\top$, we can project 3D points onto the image plane, accounting for camera distortions, by using ${}^cX'$, ${}^cY'$, and ${}^cZ'$ instead of ${}^cX$, ${}^cY$, and ${}^cZ$.

Algorithm 7: IsOccluded function

```
input : Triangle: 𝒯, Other triangle: 𝒯̃
def IsOccluded(𝒯, by = 𝒯̃):
    if all(𝒯̃.vertices.z < 𝒯.vertices.z) then
        /* 𝒯̃ is closer to the camera                          */
        return: True
    else if all(𝒯.vertices.z < 𝒯̃.vertices.z) then
        /* 𝒯 is closer to the camera                          */
        return: False
    else
        /* There is an overlap in z                           */
        occluded = False;
        orig = [0,0,0];
        for ṽ1, ṽ2 ∈ Pairs(𝒯̃.vertices) do
            for v1, v2 ∈ Pairs(𝒯.vertices) do
                /* Check if the sides of 𝒯̃ intersect the 𝒯-orig
                   tetrahedron                                */
                if sign(SignedVolume(ṽ1, v1, v2, orig)) ==
                  sign(SignedVolume(ṽ2, v1, v2, orig)) then
                    Pass;
                else if sign(SignedVolume(ṽ1, ṽ2, v1, v2)) ==
                  sign(SignedVolume(ṽ1, ṽ2, v2, orig)) ==
                  sign(SignedVolume(ṽ1, ṽ2, orig, v1)) then
                    occluded = True;
                    break;
                else
                    Pass;
            if occluded == True then
                break;
        return: occluded
```

Projecting every vertex in the scene can be computationally expensive, especially with a large number of complex objects. To improve this process, we utilize information from the STL files to filter vertices for projection selectively. The filtering approach is limited to situations where all object models consist of closed surfaces, meaning that each object has a well-defined inside and outside. In STL files, mesh triangles are defined such that their surface normals point outwards of the object. By analyzing these surface normals, we can determine which mesh triangles are obstructed by the camera-facing parts of the object and can be omitted from the projection process. Figure 4.1. demonstrates how our method detects triangles that face away from the camera. Specifically, we classify triangles whose surface normal and the camera's z axis form an angle smaller than 45 degrees as belonging to the backside of the object. The specific angle value (45 degrees in our case) is dependent on the Field of View (FOV) of the camera. This approach effectively filters out up to half of the triangles and their vertices, significantly reducing computational time for further processing.

When multiple objects are present in a scene, or when dealing with complex geometries, some surfaces facing the camera may be obscured due to occlusions. To address this issue, our projection algorithm (Algorithm 6.) incorporates additional functions outlined in Algorithms 7. and 8. These functions enable us to account for potential occlusions in

Algorithm 8: SignedVolume function

input : 3D points: a, b, c, d

def *SignedVolume*(a, b, c, d)**:**

  return: dot(cross$(b - a, c - a), d - a)$

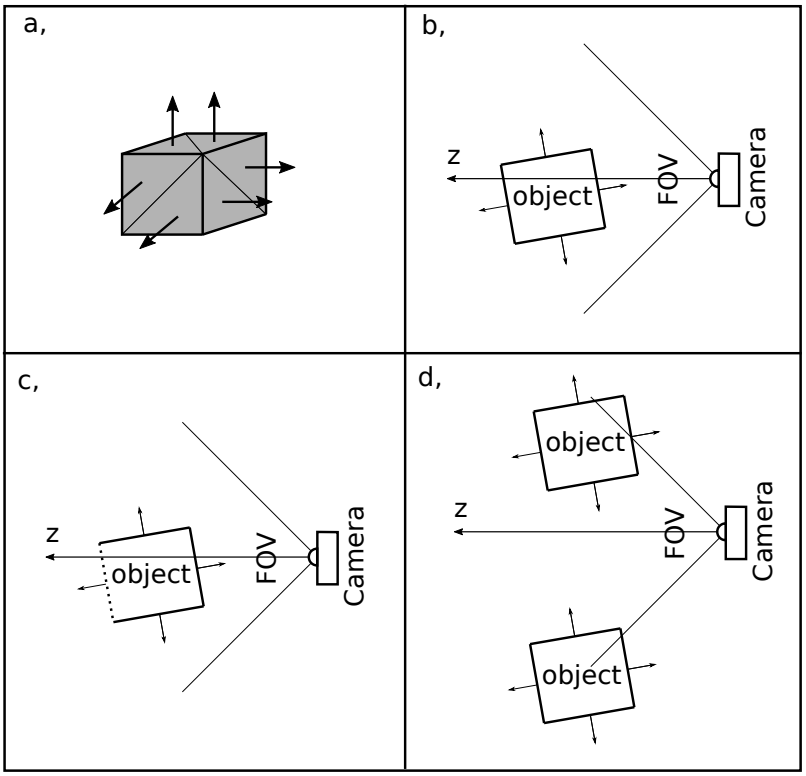


Fig. 4.1. Filtering of triangles; **a**: 3D surface model of a simple object with surface normals of the mesh triangles, **b**: cross-section view of the scene with the camera and the object, **c**: the triangles on the dotted face are not projected, **d**: all other faces might be visible depending on the location of the object in the camera's Field of View (FOV)

the scene during the projection process. These functions operate under the assumption that the objects in the scene do not intersect each other's volumes (i.e., there are no intersecting triangles) and that the size of the objects and their distances from the camera are similar in scale (i.e., there are no projected triangles that completely enclose other projected triangles). While it is possible for these conditions to be violated in certain scenarios, such as when a single object is much closer to the camera than the others or when there is a large object in the background, such situations are both rare and unlikely to be relevant for the purposes of automated segmentation for object detection. Therefore, our algorithm does not take such edge cases into consideration.

To generate annotations for each image in our dataset, we perform the process outlined in Algorithm 6. individually for each image. Our experiments indicate that, for scenes containing multiple complex objects (consisting of approximately 5,000 triangles after filtering), the automated annotation process typically takes less than 10 seconds to complete. For simpler objects with low-polygonal meshes, computation times can be reduced even further.

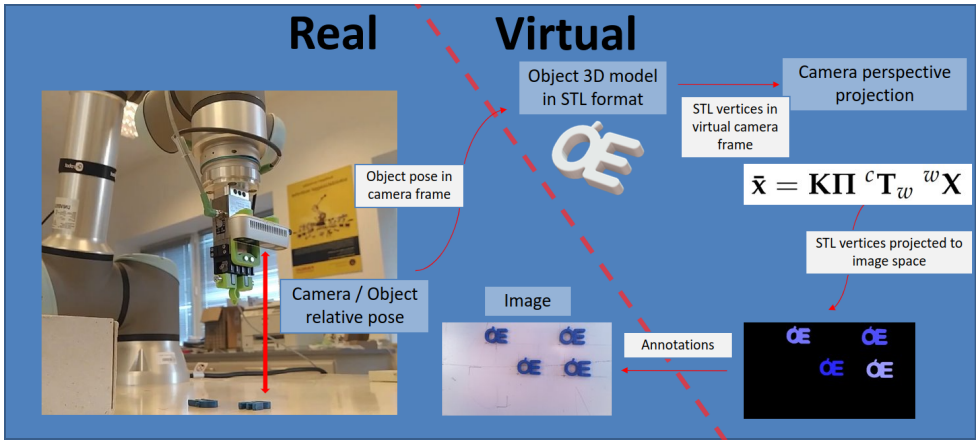


Fig. 4.2. Real data annotation setup

4.1.3 Experimental results

Our experimental setup for automatically annotating real-life data is depicted in Figure 4.2. Before the annotation process can begin, the preliminary steps that must be completed are:

1. Acquiring object meshes in STL format (${}^{o}\mathbf{X}^{\mathcal{O}}$ for all objects)
2. Camera calibration (determining $\mathbf{K}$)
3. Attaching the camera to the robot (measuring ${}^{c}\mathbf{T}_{TCP}$)
4. Placing the objects in known poses (measuring ${}^{w}\mathbf{T}_{o}$)

Once the preliminary steps are completed, data collection and automated annotation can commence. The camera is mounted on a robot, which is then moved to a pre-generated set of target poses scattered in a grid pattern on the surface of concentric spheres centered around the scene. At each pose, the robot stops, and the camera captures an image of the scene. Metadata regarding the current pose of the robot's TCP relative to the world frame is attached to the image. This metadata enables computation of ${}^{c}\mathbf{T}_{w}$ and the object masks.

To showcase the potential of our method for creating automatically annotated datasets for object segmentation, we designed a sample setup. In this particular scenario, the objective was to generate an object segmentation dataset for a wooden storage box containing microscope slides. This task is particularly challenging to achieve through manual annotations due to the following reasons:

- There are several objects in the scene at once (box and individual glass slides)
- The glass slides are tightly packed inside the box, so there are a lot of occlusions
- Due to the occlusions, the shape of the mask for the box is complicated, so there is no primitive shape that could directly describe it. As a result, the manual annotation would require drawing a polygon around the box instead of a primitive shape. Drawing polygons takes more time and requires higher precision during the annotation.
- The setup contains transparent (glass) objects, which are hard to segment accurately by hand

To facilitate automated annotations, we first created 3D meshes of the wooden box and glass slides. Our setup consisted of a Universal Robot UR16e robot arm and an Intel Realsense D435 camera, with the camera mounted on the robot's end effector. The wooden storage box containing the glass slides was positioned within the robot's workspace at a precisely known pose.

The robot executed a pre-determined motion along a set of parameterized concentric spheres located above the box, with the camera attached to its end effector. To capture multiple images with varying camera orientations, a grid of points was defined on these spherical surfaces at which the robot would halt and capture an image. A visualization of the robot's path and the intermediate photo poses can be found in Figure 4.3. The images were then saved to disk along with the corresponding TCP poses. Using (4.1), in conjunction with the projection algorithm (Algorithm 6.), we were able to automatically generate segmentation masks for each object in every image of the dataset.

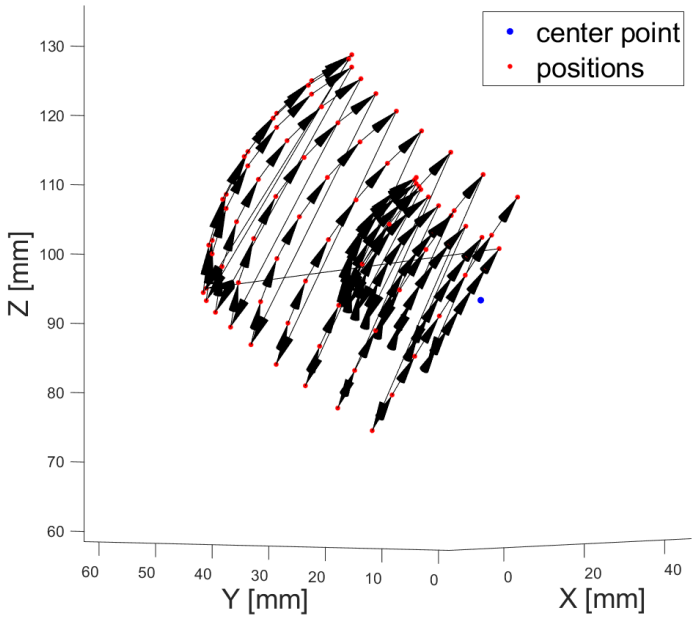
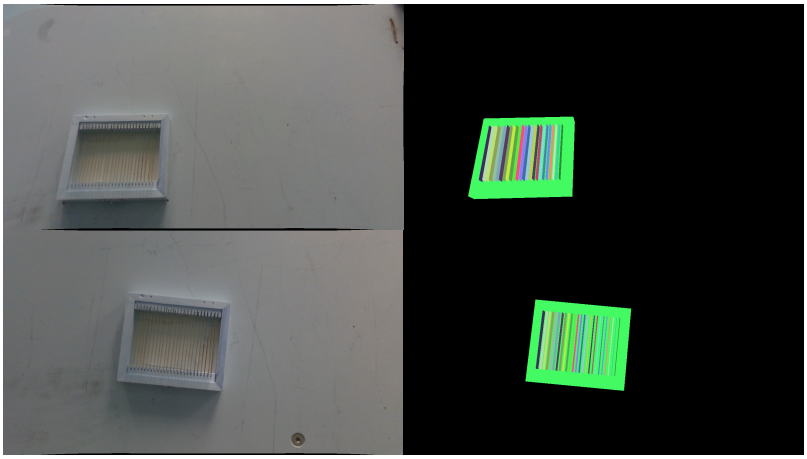


Fig. 4.3. Robot path for data collection

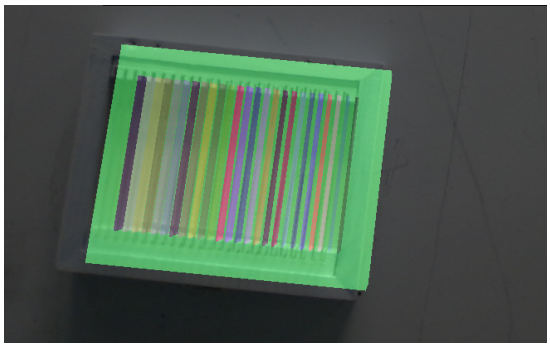
The initial results of the generated dataset and the corresponding annotations are shown in Figure 4.4. Our observations suggest that the annotations are reasonably precise when the objects are positioned near the center of the image. However, errors tend to increase as the objects move toward the edges. The accuracy achieved in the first trial is inadequate to train a model that can perform accurate segmentation of small objects such as the glass slides.

It is likely that the initial annotation errors occurred due to imprecisely measured transformation matrix parameters (${}^{c}\mathbf{T}_{TCP}$ and ${}^{w}\mathbf{T}_{o}$) and poorly calibrated camera parameters. To address these inaccuracies, we employed a genetic algorithm-based optimization process to enhance the precision of the transformation matrix ${}^{c}\mathbf{T}_{TCP}$ and the camera's intrinsic parameters ($\mathbf{K}$).

We chose some specific points on the box object (its corners) and measured their position with respect to the object frame. For simplicity, we only used the box in this process. These points were projected onto the image plane using the same method as for the mesh



Images and annotations



Masks overlayed on image

Fig. 4.4. Initial annotations, the automatic segmentation accuracy degrades around the edges of the images

vertices. We manually marked these special points on a small number (less than 10) of the real-world images. We used these points as a reference to optimize the accuracy of the parameters by minimizing the 2D Euclidean distance between the projected points and the manually marked points in the images.

Correction coefficients were introduced for each element of the ${}^{c}\mathbf{T}_{TCP}$ and $\mathbf{K}$ matrices, which were used as the variables in the optimization procedure. The original elements in ${}^{c}\mathbf{T}_{TCP}$ and $\mathbf{K}$ were multiplied by their respective correction coefficient to calculate the projected points. By changing the correction coefficients, the projection of the points was adjusted accordingly, allowing optimization to minimize the 2D Euclidean distances between the projected points and the manually marked points until a predefined threshold.

After applying the genetic algorithm-based optimization, the corrected ${}^{c}\mathbf{T}_{TCP}$ and $\mathbf{K}$ matrices were utilized to reconstruct the segmentation masks. An example of the precise annotation achieved using the corrected transformation matrix and camera intrinsic parameters is shown in Figure 4.5. This particular image was deliberately captured from a non-predefined angle and was not included in the manually marked images used to obtain special points. The accuracy of the annotations validates the effectiveness of the automated annotation procedure, utilizing the corrected ${}^{c}\mathbf{T}^{TCP}$ and $\mathbf{K}$ matrices in generating object segmentation datasets for robotics applications.

We conducted an additional experiment for generating datasets, wherein we employed

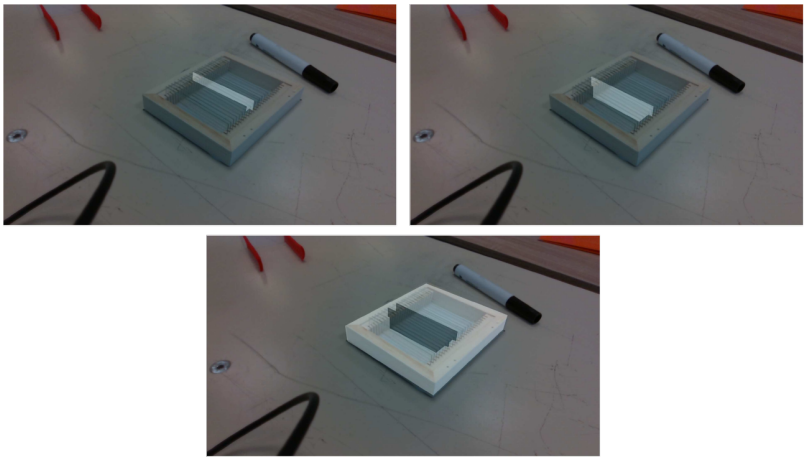


Fig. 4.5. Annotations (created by using the corrected ${}^{c}\mathbf{T}_{TCP}$ and $\mathbf{K}$ matrices) overlayed on an image

a more accurate calibration process for both the camera and robot. Figure 4.6 exhibits sample images depicting the segmentation masks produced by our automatic annotation pipeline for real-world data after employing the precise camera calibration. Unlike before, we did not utilize the genetic algorithm-based parameter optimization to generate these masks. As demonstrated, the system can directly generate precise segmentation masks with a high degree of accuracy after undergoing precise calibration. The dataset comprised 400 samples featuring various types and numbers of objects, varying degrees of clutter, and illumination, annotated with instance segmentation masks.

4.2 Synthetic dataset preparation

4.2.1 Motivation

Robots in modern robotic applications often encounter dynamic environments that undergo constant changes in illumination, clutter, and object occlusions [4]. In such scenarios, data-driven approaches, such as Deep Learning (DL), are commonly employed to enable robots to perceive their surroundings through visual data processing. The generalization capability of DL models is critical in dynamic environments. In this section, two methods for achieving excellent generalization in DL models are explored. The first approach is continual learning [82, 4], which involves accumulating knowledge over time in a dynamic environment while avoiding the forgetting of previously acquired knowledge [82]. The second approach is transfer learning [47, 49], which relies on datasets that facilitate the training of models capable of good generalization.

In both of these approaches, the training data plays a crucial role. Studies have demonstrated that knowledge acquired by training on a dataset from a source domain transfers better to a target domain when the source and target domains share similarities [49]. Additionally, introducing variations in the data distribution of the training set can assist the trained model in adapting to differences in the data distributions between the source and target domains [91, 92]. Since many DL-based solutions employ supervised learning techniques, the training datasets often require manual annotation [36]. However, the manual

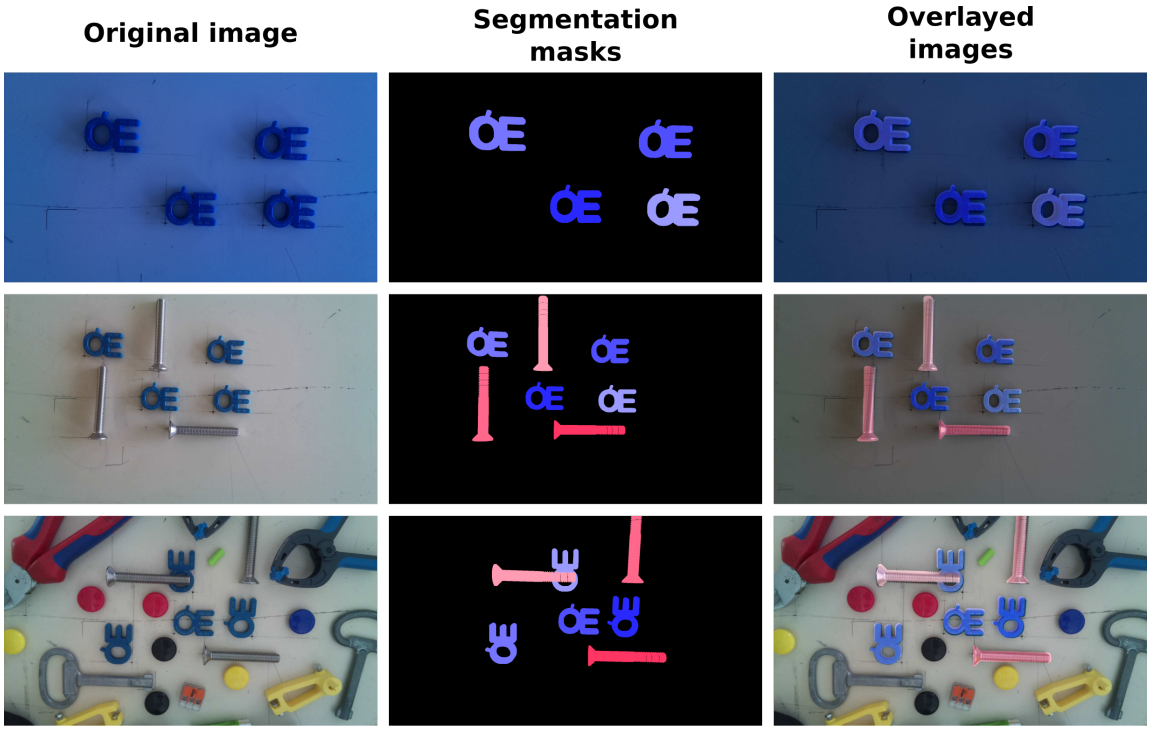


Fig. 4.6. Examples of automatically annotated real images

annotation of data can be highly time-consuming or impractical, particularly in complex tasks such as semantic scene segmentation or instance segmentation, which are commonly used in robotic applications. Moreover, DL models necessitate extensive training datasets. As a result, annotating the training data can pose a significant challenge in developing new DL-based solutions for robotics.

The benefits of utilizing automated methods for data annotation and synthetic data for DL model training are apparent. Synthetic data can be easily annotated automatically, and diverse variations in the data distribution can be incorporated. Nonetheless, using synthetic data alone does not eliminate the generalization obstacle. Ultimately, the trained model must be capable of processing real data. This requirement can only be fulfilled if the model trained on the synthetic dataset can also generalize to the real data, which is commonly known as "bridging the reality gap" [92, 91].

There are two primary strategies for bridging the reality gap. The first approach is to generate a synthetic dataset that closely resembles the real dataset [83, 84, 85, 89, 90]. In the case of synthetic images, this is accomplished by photorealism. The second approach is domain randomization. This method incorporates exaggerated levels of variability into the synthetic dataset, causing models trained on such datasets to disregard the effects of the randomized factors and generalize to real data as well [89, 90, 91, 92].

The advantages of using a synthetic dataset are often discussed in terms of transfer learning. It is widely accepted that pre-training on a large synthetic dataset and then fine-tuning on real data can be more effective than training solely on real data [85, 196]. For instance, Nowruzi et al. investigated the benefits of including a synthetic dataset along with real annotated data for object detection [196]. In their experiments, they trained a DL model (SSD single shot detector [197] with MobileNet as the backbone [198]) from

scratch on a dataset containing both synthetic and real data. They discovered that the additional synthetic data considerably reduced the requirement for training on real samples (10%, 5%, and 2.5% of the original real dataset were used). Furthermore, when training the model from scratch, pre-training on synthetic data and fine-tuning on real data outperformed training the model on a mixed dataset (comprising both real and synthetic samples). The pre-trained model also has the advantage that it can be adapted to other domains by fine-tuning on a small amount of real data, thereby eliminating the need for another lengthy training process involving the entire synthetic dataset. Thus, creating a synthetic dataset can often be viewed as an alternative when a large annotated real dataset for pre-training is not available.

We propose a technique for automatically generating and annotating synthetic image data and show that the advantages of synthetic data can also be leveraged during fine-tuning. Our approach assumes the availability of a pre-trained model and focuses on the fine-tuning step. We demonstrate how synthetic data can improve the fine-tuning process of transfer learning methods for robotic perception for manipulation and the training of continuous learning techniques that use experience replay for image classification.

4.2.2 OE and SynLORIS synthetic scenes and FTRG method

When training a DL model using a synthetic dataset, challenges may arise when the model later performs inference on real-world data due to differences between the simulated and target domains. These differences can include neglected physics interactions and a simplified world model. These challenges are even more significant in reinforcement learning, which requires interaction with the environment in addition to perception [58, 199]. Nonetheless, the quality of the synthetic dataset remains important in perception tasks, as demonstrated in [200]. Therefore, sim-to-real approaches in reinforcement learning can provide valuable insights into the requirements that the simulation software used for synthetic data generation must meet.

The key lessons to learn from sim-to-real approaches in reinforcement learning are the importance of close-to-reality simulation and domain randomization [58, 199, 91]. In the context of perception for object segmentation tasks, achieving close-to-reality simulation is accomplished by generating photorealistic renders. Therefore, the primary requirement for simulation software is the ability to produce such high-quality renders.

It has been shown that introducing randomization to certain parameters of the scene, such as lighting conditions, textures, and backgrounds, can enhance generalization and improve adaptation to real-world scenarios [91]. This technique is known as domain randomization. Therefore, the simulation software used as a foundation for generating synthetic data must have the capability to randomize these aspects of the scene as conveniently as possible.

In addition to domain randomization and photorealism, the presence of physics simulation is not an essential requirement but rather a preference, as it enables the creation of natural-looking piles of objects (e.g., for robotic bin-picking tasks), which may be difficult to construct manually.

Finally, it is essential to consider the accessibility and setup costs of the simulation software. It is advisable to use a lightweight and easy-to-learn simulation software rather than a cumbersome one. Open-source software is advantageous because of its versatility

TABLE 4.1
COMPOSITION OF THE OE SYNTHETIC DATASET

Image ID	training (t) validation (v)	textures	objects	lighting
0-399	t	photoreal.	OE	static
400-499	v	photoreal.	OE	static
500-899	t	photoreal.	OE	rand.
900-999	v	photoreal.	OE	rand.
1000-1399	t	rand.	OE	rand.
1400-1499	v	rand.	OE	rand.
1500-1899	t	photoreal.	OE, bolt	rand.
1900-1999	v	photoreal.	OE, bolt	rand.
2000-2399	t	rand.	OE, bolt	rand.
2400-2499	v	rand.	OE, bolt	rand.

and ease of access. Therefore, a well-documented open-source simulation software can be an excellent foundation for any synthetic data generation pipeline.

Based on these considerations, we have selected the Blender 3D suite as the foundation for our synthetic data generation pipeline. Blender is an open-source software with a thriving and supportive community [201]. As a result, there are numerous online resources available to learn its features and functionalities. Although the use of Blender falls beyond the scope of this work, we encourage readers to refer to these resources for any unfamiliar terminology or additional inquiries about Blender's capabilities. Blender is a general-purpose 3D creation tool that is not limited to a specific domain type, unlike certain driving or robotics simulators. Its primary function is the creation of computer graphics, and it offers an abundance of tools for manipulating visual scenes, including 3D object models, lighting and camera configurations, geometry modifications, textures and shading, image post-processing, and more. Blender can generate photorealistic renders and incorporates physics simulation through the Bullet physics engine. Additionally, it includes a Python API, which facilitates integration into a DL training workflow, as most DL frameworks support the Python programming language. Lastly, accessibility and ease of setup were key factors in our decision. We believe that an open-source computer graphics software with extensive documentation, like Blender, is a strong foundation for synthetic data generation.

For our experiments, we created two virtual scenes. The first one is a tabletop environment that includes objects available at our laboratory to test our proposal of using synthetic data for the fine-tuning of DL models. We refer to this scene as the OE scene. The second scene, SynLORIS, was created to replicate a scene from the OpenLORIS Object dataset [4] and was used to compare the performance of continual learning models with and without synthetic data for experience replay. SynLORIS is a synthetic version of one of the real scenes in the OpenLORIS Object dataset. Both scenes were created with the aim of generating a fine-tuning dataset.

Fig. 4.7. Example rendered frames from the OE synthetic dataset

OE scene

The OE scene includes two categories of objects: 3D printed 'OE' logos and DIN EN ISO 10642 M8x55 bolts. The clutter objects present in the scene are clamps and pliers, which were photo-scanned using the Qlone [194] photogrammetry application in our laboratory. The background of the scene is a planar tabletop. The positions of the OE logos, bolts, clutter objects, and the distance between the camera and the background plane were randomized.

To achieve realistic simulation in our experiments, we utilized photorealistic textures and realistic lighting for the objects and background. To create the photorealistic shading of the tabletop, we applied an image texture, while the clutter objects utilized textures obtained from photo-scanning. The OE logos and bolts had shaders specifically designed using Blender. Additionally, we created randomized textures for domain randomization, which were intentionally unrealistic.

A dataset of 2,500 images (referred to as the OE synthetic dataset) was generated from the scene. Table 4.1. provides a detailed description of the dataset's composition. Some rendered frames of the OE synthetic dataset can be seen in Figure 4.7.

SynLORIS

We modeled our SynLORIS scene based on a real scene from the OpenLORIS Object dataset. To achieve a similar environment, we replicated the placement of the desk and background elements, as well as the lighting direction. We selected 3D assets that resemble the real objects and textures but limited our selection to freely available ones from BlenderKit's library. The scene was not created with the aim of producing highly realistic renders or perfectly recreating the objects because, for the OpenLORIS Object benchmark,

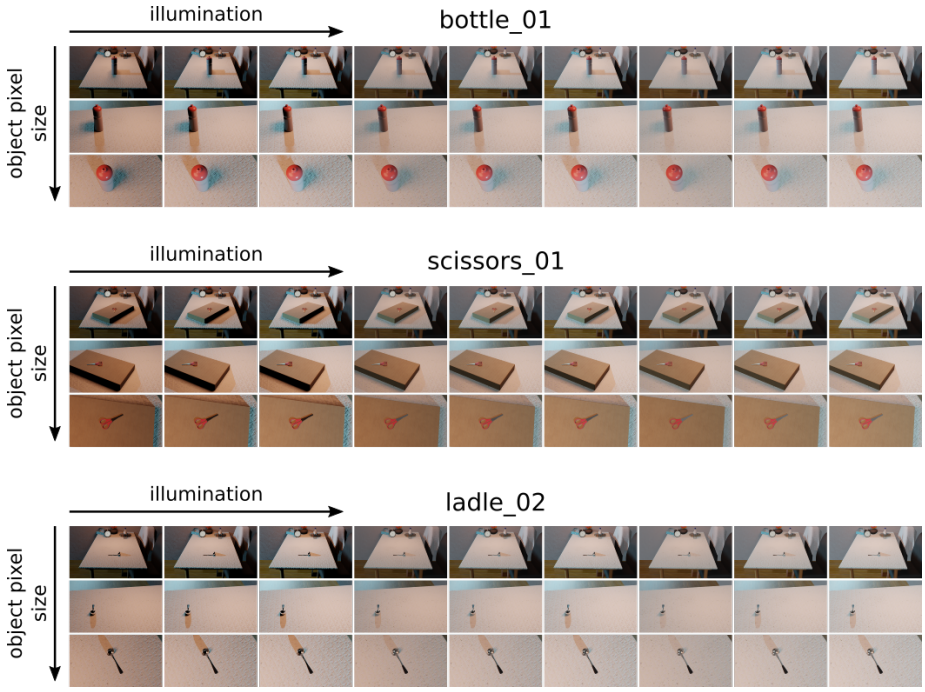


Fig. 4.8. Synthetic images rendered from the SynLORIS scene

the images are resized to 50×50 pixels, which means fine details would have been lost anyway.

In the SynLORIS scene, we incorporated variations in two factors highlighted in [4]: illumination and object pixel size. The scene features three sources of illumination: an HDRI, an area light simulating light coming through the window, and a point light source situated above the table. We altered the power of the lamps and the strength of the HDRI lighting to achieve different illumination effects. To simulate variations in object pixel size, we moved the camera closer and farther away from the objects along a 3D spline that we manually defined.

In a similar fashion to the single-factor experiments conducted in OpenLORIS Object, we designed nine illumination-related tasks in the SynLORIS scene, each with a different illumination level. We considered a subset of seven objects from the OpenLORIS Object dataset, all of which share the same scene. To create the dataset, we generated 30 synthetic images for each object in each task. Regardless of the object or task, we used the same camera path. Overall, the dataset contains 1,890 rendered images (9 tasks, 7 objects per task, 30 images per object). Figure 4.8. displays some of the rendered frames. We utilize the SynLORIS dataset in our experiments to demonstrate how an image classification model using experience replay can benefit from synthetic data.

Blender Annotation Tool (BAT)

To generate segmentation-type annotations for the OE synthetic dataset, we created the Blender Annotation Tool (BAT), a Blender addon[1]. BAT can be used via a simple user interface (referred to as BAT panel) located in a dedicated tab of the "n-panel" of the 3D

[1] `https://github.com/ABC-iRobotics/blender_annotation_tool`

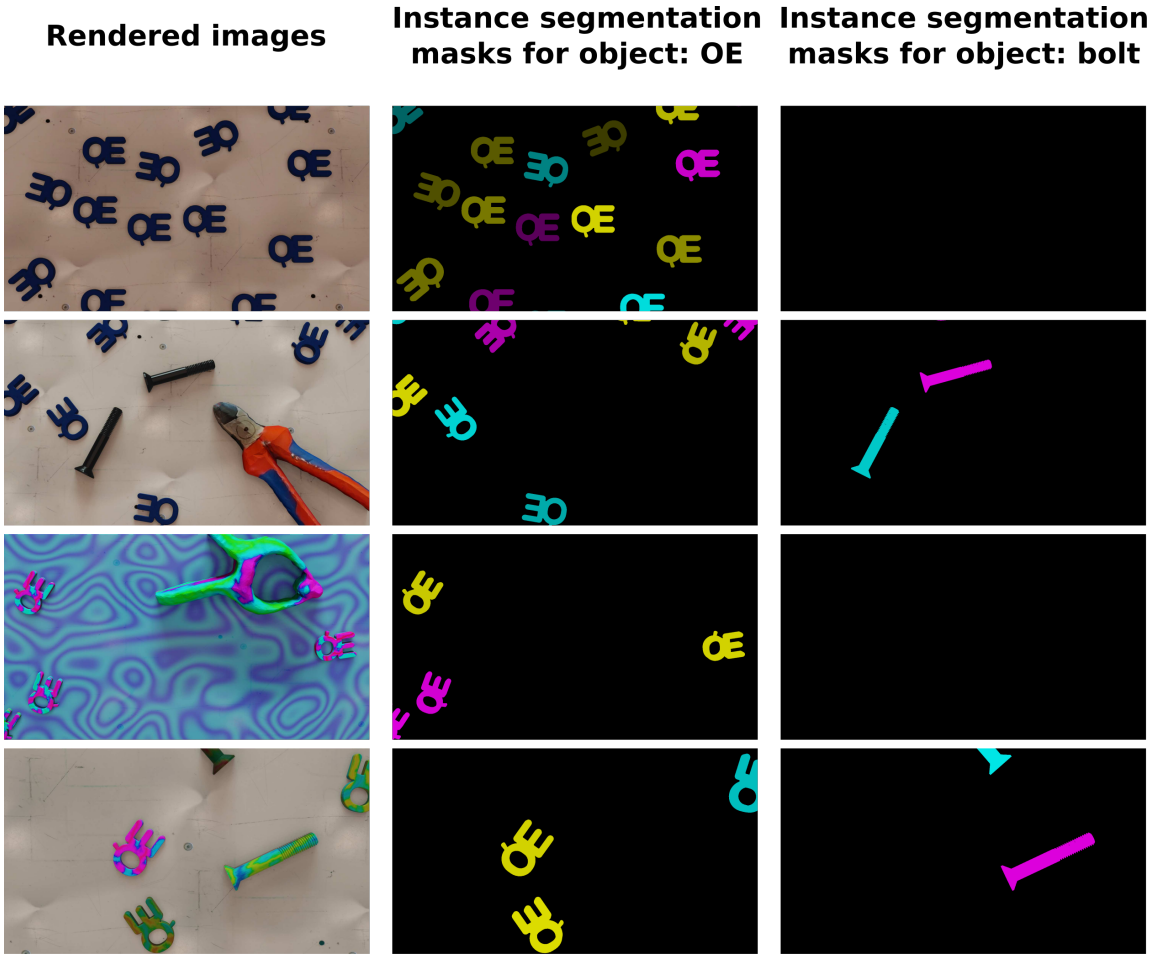


Fig. 4.9. Example BAT annotations for the OE synthetic dataset

Viewport. The class for the background is added by default with black color. The BAT panel can be used to create, delete or rename classes, change the color ID of a class or the collection of objects associated with it, and for toggling whether the object collection should be treated as a collection of instances or not.

In order to generate segmentation masks, BAT utilizes the viewport renderer OpenGL. Consequently, BAT is dependent on an open GUI of Blender and cannot operate in the background. Despite this limitation, the time and resources required for generating annotations are significantly less than rendering (two orders of magnitude based on our experience). Moreover, the rendering of synthetic images can be executed separately from the annotation generation, enabling us to leverage a powerful headless server for rendering while using a separate system with limited resources for annotation generation. Examples of the produced annotations are shown in Figure 4.9.

FTRG method

We propose a method for generating a mixed-reality dataset by utilizing the described automatic annotation techniques. This method is called "Filling the Reality Gap" (FTRG), as it seamlessly combines synthetic and real data to create rendered counterparts of real

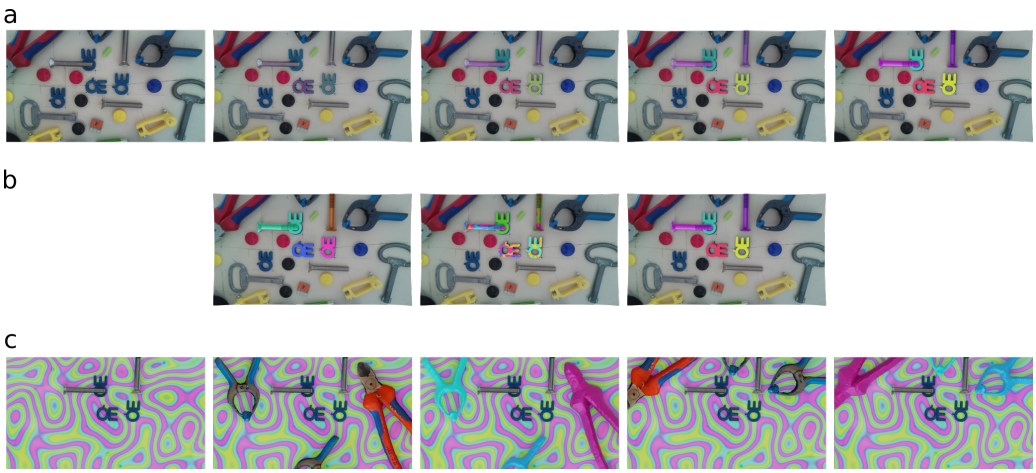


Fig. 4.10. Samples from our FTRG dataset; **a**: Seamless transition from real to synthetic textures on a selected subset of objects, **b**: Random texture for a selected subset of objects of interest with real background and clutter, **c**: Real objects in a synthetic scene with synthetic clutter

images. By combining the automatic real-data annotation pipeline with the synthetic data generation and annotation method (using BAT), we are able to mix elements of both real and virtual scenes, including objects of interest, backgrounds, and clutter objects, in a single image. By controlling the degree to which the virtual scene is mixed with the real one, we can create a seamless transition between the two environments. The name of our method, FTRG, reflects its purpose in bridging the gap between reality and the virtual environment.

The FTRG method involves several steps. First, we use our automatic real-data annotation pipeline to create a real dataset. Next, a synthetic version of the scene is created using Blender. To ensure consistency between the real and synthetic images, Blender's motion tracking module determines the camera pose, motion, and internal parameters based on the real dataset images. After rendering, the real and synthetic images can be seamlessly blended using Blender's compositor workspace. To create our FTRG dataset, we combined various elements of the real and synthetic images (such as real backgrounds with synthetic objects or synthetic backgrounds with real objects). The labels for the FTRG dataset are inherited from either the real or the synthetic scene using BAT. Figure 4.10. illustrates some possible ways of mixing synthetic and real data in the FTRG dataset.

To demonstrate the effectiveness of the FTRG method, in our experiments, we conducted a comparison of Mask-RCNN [74] networks that were fine-tuned using various datasets. Specifically, we compared the performance of networks fine-tuned using the FTRG method with those fine-tuned using photorealistic synthetic data, domain randomized synthetic data, or real images.

Continual learning experiments

For our experimentation with continual learning, we utilized the OpenLORIS Object benchmark proposed by She et al. in [4]. They proposed an evaluation approach for continual learning techniques based on the train-test accuracy matrix, which is illustrated in Table 4.2. They introduced two metrics: Forward Transfer (FWT) and Backward Transfer

TABLE 4.2

TRAIN-TEST ACCURACY MATRIX R FROM [4]; Tr REPRESENTS TRAINING DATA, Te REPRESENTS TESTING DATA, $R_{i,j}$ IS THE ACCURACY OF THE MODEL TRAINED ON Tr_i AND EVALUATED ON Te_j, N IS THE NUMBER OF TASKS

R	Te_1	Te_2	...	Te_N
Tr_1	R_{11}	R_{12}	...	R_{1N}
Tr_2	R_{21}	R_{22}	...	R_{2N}
...	...	...	...	...
Tr_N	R_{N1}	R_{N2}	...	R_{NN}

(BWT). FWT is the average accuracy calculated for the upper triangle of the train-test accuracy matrix, marked in blue in the table. BWT, on the other hand, is the average accuracy for the lower triangle of the train-test accuracy matrix, marked in red in the table. BWT measures how well a model retains information from previous tasks, while FWT measures how well a model adapts to new tasks after training on the previous ones.

She et al. demonstrated that maximizing FWT is particularly difficult for many continual learning techniques. In our interpretation, FWT measures the ability of a trained model to generalize to new tasks. Our experiments aim to test the hypothesis that the use of synthetic data in the form of rendered images can enhance the FWT of specific continual learning models. To this end, we employed continual learning with experience replay [202] and evaluated its performance on the single-factor benchmarks introduced by She et al.

We limited our experimentation to seven objects from the OpenLORIS Object dataset, namely bottle_01, bowl_01, cup_02, cup_04, ladle_02, paper_cutter_04, and scissors_01, as these objects were also used to generate our synthetic dataset, SynLORIS. The evaluation was based on four factors: illumination, occlusion, clutter, and object pixel size, and each factor had nine tasks with varying levels of the corresponding factor. We assessed two models for each factor in our experiments. One of the models was trained exclusively with data from the original OpenLORIS Object training set, while the other was trained with data from both the OpenLORIS Object and the corresponding SynLORIS dataset. Both models had a memory budget of 2000 and were trained for 100 iterations on each task. The OpenLORIS Object dataset's validation set for the seven objects was employed for testing across all factors and tasks. We evaluated each model using the same metrics employed by She et al. for all four factors.

In Figure 4.11., the train-test accuracy matrices for all four factors are displayed to allow for qualitative assessment. These matrices are presented as images, where the pixel intensity corresponds to the value of an element in the matrix. A pixel with a value of 0 is represented as black, while a pixel with a value of 1 is represented as white.

It should be noted that the SynLORIS dataset incorporates changes in only two factors, namely illumination and object pixel size. This is apparent in Fig 4.11., which demonstrates that significant differences in the train-test accuracy matrices are only noticeable in these two factors. In such instances, the model that had access to synthetic samples performed better than the model that solely underwent training with real samples.

Table 4.3. presents the outcomes of our continual learning experiments utilizing the metrics introduced by She et al. in [4]. The first row (m1) displays the results obtained from exclusively training models on the OpenLORIS Object dataset. In contrast, the sec-

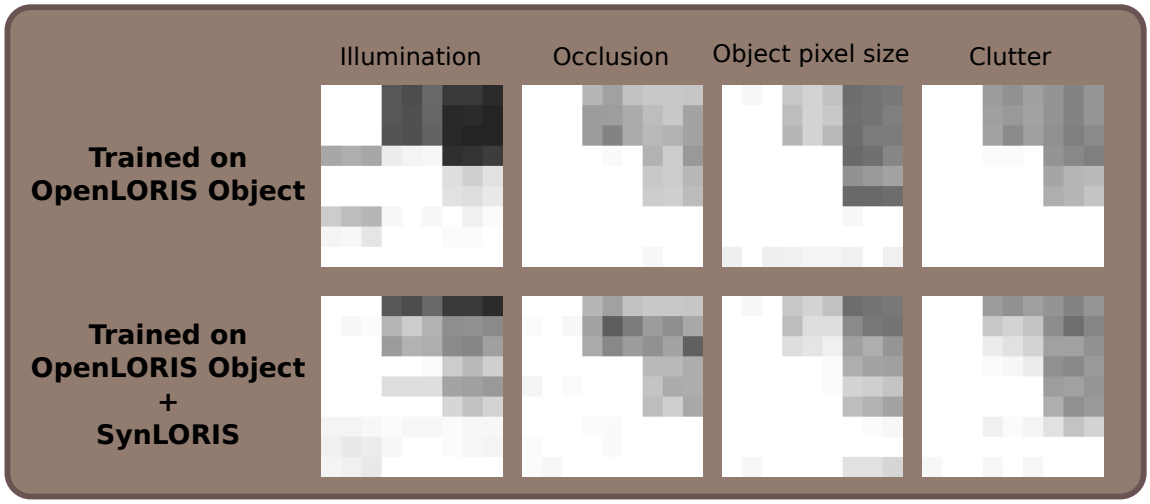


Fig. 4.11. Train-test accuracy matrices of image classification models, using experience replay and data from the OpenLORIS Object and the SynLORIS datasets, as evaluated by the OpenLORIS Object benchmark on all four factors. (brighter color means greater accuracy)

TABLE 4.3
QUANTITATIVE RESULTS FROM OUR CONTINUAL LEARNING EXPERIMENTS. VALUES PER CELL FROM TOP TO BOTTOM: ACCURACY, BWT, FWT, OVERALL ACCURACY (AS DESCRIBED IN [4]))

	Illumination	Occlusion	Object pixel size	Clutter
m1	0.954	0.999	0.991	1.0
	0.945	0.999	0.991	1.0
	0.539	0.782	0.683	0.704
	0.77	0.903	0.854	0.868
m2	0.979	0.996	0.991	0.993
	0.979	0.995	0.993	0.994
	0.685	0.74	**0.768**	0.716
	0.848	0.882	0.892	0.869

ond row (m2) showcases the performance of models that underwent training on data originating from both the OpenLORIS Object and the SynLORIS datasets. It is worth noting that the models trained on both real and synthetic data exhibited considerably better FWT for the illumination and object pixel size factors when compared to models that exclusively used real data.

Experiments on synthetic data for fine-tuning

We conducted experiments to showcase the influence of incorporating synthetic data into the fine-tuning stage of deep neural network training. Additionally, we emphasize the advantages of using the FTRG approach over solely relying on photorealistic synthetic images and/or domain randomization.

We trained multiple Mask-RCNN models, utilizing different instance segmentation datasets, and assessed their performance using the same test dataset[2]. The models all shared the same network architecture and were initialized with the same set of pre-trained

[2]Mask-RCNN implementation from: `https://github.com/matterport/Mask_RCNN`

TABLE 4.4
PERFORMANCE OF MODELS (MAP @ IOU$\geq$ 0.5) EVALUATED ON FIVE RANDOMLY SELECTED SUBSETS OF OUR TESTING SET

	subset1	subset2	subset3	subset4	subset5
MRCNN-R	0.8889	0.8983	0.9029	0.9077	0.8674
MRCNN-P	0.8763	0.8599	0.8505	0.8296	0.8395
MRCNN-DR	0.8654	0.8307	0.8437	0.8132	0.7949
MRCNN-DR-R	0.9955	0.9817	0.9831	0.9774	0.9651
MRCNN-DR-P-R	0.9984	0.9887	0.9891	**0.9812**	0.9709
MRCNN-FTRG	**1.0**	**0.99**	**0.9895**	0.98	**0.9715**

weights (which were originally trained on the COCO dataset [187]). We adopted the same hyperparameters and a fixed number of training iterations for fine-tuning across all the models. In total, we trained five models and named each one based on the type of data utilized for its training.

- *MRCNN-R*: This model was fine-tuned using only real data. The training dataset was annotated by our automated real dataset annotation method.
- *MRCNN-P*: This model was fine-tuned using only photorealistic samples from the OE synthetic dataset
- *MRCNN-DR*: This model was fine-tuned using only synthetic images with unrealistic textures from the OE synthetic dataset.
- *MRCNN-DR-R*: This model was fine-tuned using both domain-randomized synthetic samples from the OE synthetic dataset and samples from the real dataset.
- *MRCNN-DR-P-R*: This model was fine-tuned using synthetic samples from the OE synthetic dataset (both photorealistic and domain randomized) and real samples as well.
- *MRCNN-FTRG*: This model was fine-tuned using the FTRG dataset, which combines the real dataset with the domain-randomized synthetic dataset using the FTRG method.

Our test data was gathered by capturing real images in diverse settings (comprising varying backgrounds and object arrangements), levels of clutter (as well as the types of objects present in the clutter), and illumination conditions. We annotated this test set utilizing the proposed automated real data annotation method. Table 4.4. lists the mean average precision (mAP) for all six models at intersection over union (IoU) values greater than or equal to 0.5. The results presented in the table were obtained from five different randomly selected subsets of the test dataset.

Our findings indicate that models which were exclusively fine-tuned on synthetic data produced inferior results when compared to models trained solely on real data. However, models that were trained on datasets containing both synthetic and real data surpassed the performance of models trained solely on real data. This suggests that utilizing synthetic data during the fine-tuning phase of deep neural network training can lead to better results than exclusively training on real data.

Notably, the model trained on our FTRG dataset achieved superior results in four out of five cases, outperforming all other models. We believe the slight improvement compared to

the *MRCNN-DR-P-R* model can be attributed to the fact that our FTRG dataset comprises samples that contain a mixture of real and synthetic elements. This property enables the *MRCNN-FTRG* model to adjust better to diverse real-life domains.

4.2.3 Fine-tuning GQCNNs with task-specific synthetic data

Motivation

Robots are extensively used in industrial packaging and assembly processes due to their ability to perform tasks that would be too tedious and repetitive for humans; however, as the industry shifts towards greater flexibility, new requirements for robot applications emerge, resulting in novel challenges in robotic manipulation. One such challenge is grasp planning, which involves finding appropriate grasps for an object based on a quality measure, such as wrench-space metrics computed analytically [203], empirical evaluation from physical trials [30], or similarity to human-provided grasps [204].

Grasp planning is necessary for some robotic manipulation tasks because the robot's program cannot be "rigid" and repeat the exact same movements over and over if the pose and/or the object's geometry are unknown in advance. Therefore, grasp planning is used to find appropriate grasps for a given object according to a quality measure. This allows for flexibility in the robot manipulation pipeline, allowing for quick and easy adjustments to be made for novel objects or objects with unknown poses.

Determining the quality of grasps is a complex problem that depends on several factors such as object geometry, gripper geometry, dynamic properties like friction forces, and the task to be performed after the object is grasped [205]. Analytical approaches can provide reliable solutions based on simulated contacts and dynamics, but they often require prior knowledge of object properties such as geometry, material, and inertia matrix. Besides, their computation can be time-consuming [203]. Furthermore, object detection and pose estimation must be performed separately, and it is challenging to incorporate the relevant information about the task into analytical grasp planning.

On the other hand, data-driven approaches for grasp planning can integrate object detection, pose estimation, and grasp planning into a single model [206]. These models can learn general grasping policies from large datasets and apply them to novel objects [30, 206]. Moreover, grasp prediction with such models is significantly faster compared to analytical methods [206]. However, a large dataset is required to train these models. Collecting a large-scale grasping dataset using physical robots can be time-consuming and expensive, taking up to months [30]. As a result, offline-generated simulation-based synthetic data is usually preferred, where analytical grasp quality approaches can be used for ground-truth generation [206].

Miller and Allen developed Graspit!, which is a simulator used for evaluating robotic grasps [203]. With Graspit!, users can import arbitrary gripper and object geometries and evaluate a large number of grasps for a given gripper-object combination to identify the best quality grasps using a sampling-based approach. Although Graspit! can also simulate and compute the quality of grasps in a dynamic environment, this is beyond the scope of our research for now. Goldfeder et al. created the Columbia Grasp Database, along with a data-driven approach for grasp planning, using the Graspit! simulator [207, 208]. Their approach involved matching partial object geometry information against a large dataset

of 3D models and selecting appropriate grasps from a set of pre-computed grasps for the matched objects. A similar approach called Dex-Net 1.0, which used pre-computed grasps, was introduced by Mahler et al. who utilized cloud computing to significantly reduce the application runtime [209]. They used Multi-View Convolutional Neural Networks to measure the similarity between objects. Mahler et al. later compiled a vast synthetic dataset of 6.7 million point clouds with grasps and associated analytic grasp metrics, called Dex-Net 2.0, based on their results in Dex-Net 1.0 [206]. They proposed a convolutional neural network architecture called Grasp Quality Convolutional Neural Network (GQCNN), which could predict the probability of successful grasps directly from the depth image data. They demonstrated that a GQCNN trained on Dex-Net 2.0 synthetic dataset outperformed other state-of-the-art approaches that used point cloud registration while being $3\times$ faster. Moreover, they achieved great precision (99%) on a set of novel objects.

One limitation of the aforementioned methods is that they only consider grasping in isolation, without taking into account the specific task that will be performed after the grasp. According to Costanzo et al., using fixed grasps alone can render a robot manipulation task infeasible [210]. They propose that in-hand manipulation techniques, such as object pivoting based on tactile feedback (detailed in [211]), may be necessary for a successful task solution, and thus grasps should be selected accordingly. Task-specific grasp planning methods aim to address this issue by incorporating a task encoding into the grasp prediction process [212, 213, 214]. However, the task encoding in such approaches is typically categorical and represents a generalized use case, such as poking or pouring water. While these methods work well for general scenarios, they may not be sufficient for specific pick-and-place tasks in robotic assembly where object geometries and assembly order limit the number of suitable grasps. In such cases, a categorical task encoding would require the representation of each unique assembly as its own category, which would necessitate the model's retraining from scratch each time a new assembly task is presented. A more appropriate solution for these scenarios would be a transfer learning-based approach.

Transfer learning is a technique that involves reusing large deep-learning models that have been pre-trained on vast datasets. During transfer learning, only the top layers of the network are modified, while the lower layers, which extract more general features, are left untouched. This approach allows us to use a relatively small dataset without overfitting since only a fraction of the network's parameters need to be adjusted. The process of modifying the top layers is called fine-tuning the network. We propose an automated synthetic dataset generation pipeline using sampling-based analytic grasp quality assessment (Graspit!) and computer graphics (Blender) to fine-tune GQCNN models for task-specific grasp planning in robotic assembly. Our focus is on predicting geometrically well-placed grasps and not examining dynamic grasp qualities. We present and discuss the results through a simple yet representative simulated task.

Methodologies

Similarly to the Columbia Grasp Database by Goldfeder et al. our synthetic dataset generation pipeline also utilizes Graspit! for determining grasps qualities. Figure 4.12. shows the proposed synthetic dataset generation pipeline. As the figure demonstrates, Graspit! is used to automatically generate grasps for two scenarios: picking and placing the object. In the picking scenario, Graspit! provides multiple possible grasps for picking the object.

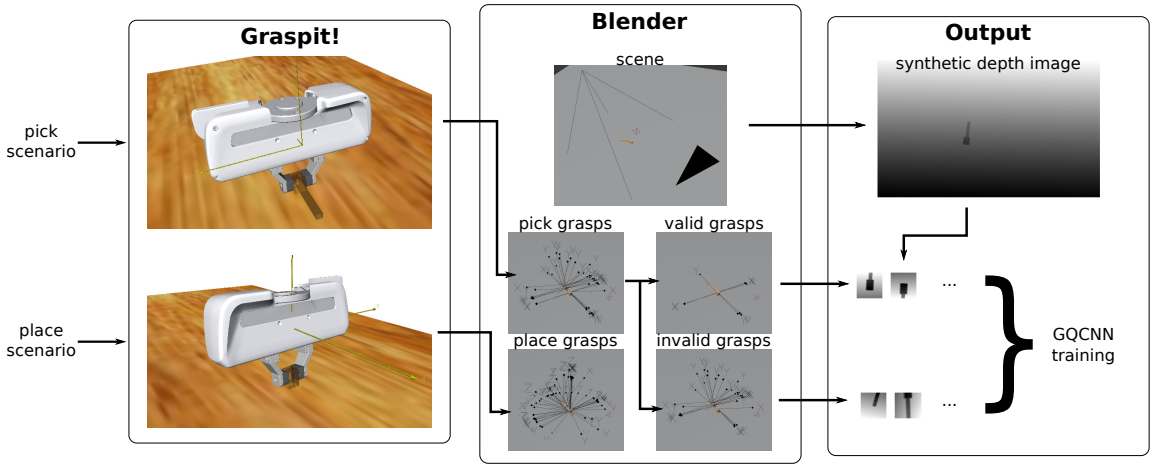


Fig. 4.12. Synthetic data generation and GQCNN training pipeline

In the placing scenario, Graspit! provides grasps suitable for placing the object. For our experiments, we use a Franka Emika Panda robot, its signature parallel jaw gripper, and a simple object composed of box primitives. Our experiment simulates an insertion-type assembly subtask where the object is grasped at the thicker part. By sequentially chaining pick-and-place setups according to the assembly order, a more complex assembly can be composed. For the synthetic dataset generation, we require the 3D models of objects, the gripper model, and knowledge of the assembly process, including the location of parts relative to each other and the assembly order.

Our proposed method incorporates Graspit! with the Robot Operating System (ROS), an open-source robotics framework that provides useful tools for creating and managing robotic applications [215]. With the help of the Graspit! ROS interface[3] in conjunction with Graspit! Commander[4] the Graspit! grasp planning process can be automated using the Python programming language and ROS. We store the Graspit! grasp planning outputs in multiple JSON files, each named according to whether the grasps are for the pick or place scenario. The files contain the object pose, a list of search energies from Graspit! (which can be used to create a hierarchy of predicted grasps), and a list of predicted grasp poses. Both the object pose and the grasp poses are expressed relative to the world frame and consist of a 3D vector for object/gripper location and a quaternion (in $[x, y, z, w]$ format) for object/gripper orientation.

To generate the synthetic depth images, we have utilized Blender, a free and open-source 3D computer graphics software. We leveraged Blender's Python API to automate the rendering process and manipulate the intrinsic and extrinsic camera parameters to match a real-world calibrated camera. In our experiments, we used the camera parameters of a PrimeSense camera provided with the GQCNN implementation[5].

After setting up the camera, the grasp poses obtained from Graspit! are loaded and transformed to the object frame in Blender, regardless of the object pose in Graspit! and Blender (the grasp poses from both the pick and the place scenario are represented in the same frame). Then the pick grasps are sorted into two categories based on their geometric

[3] https://github.com/graspit-simulator/graspit_interface
[4] https://github.com/graspit-simulator/graspit_commander
[5] https://github.com/BerkeleyAutomation/gqcnn

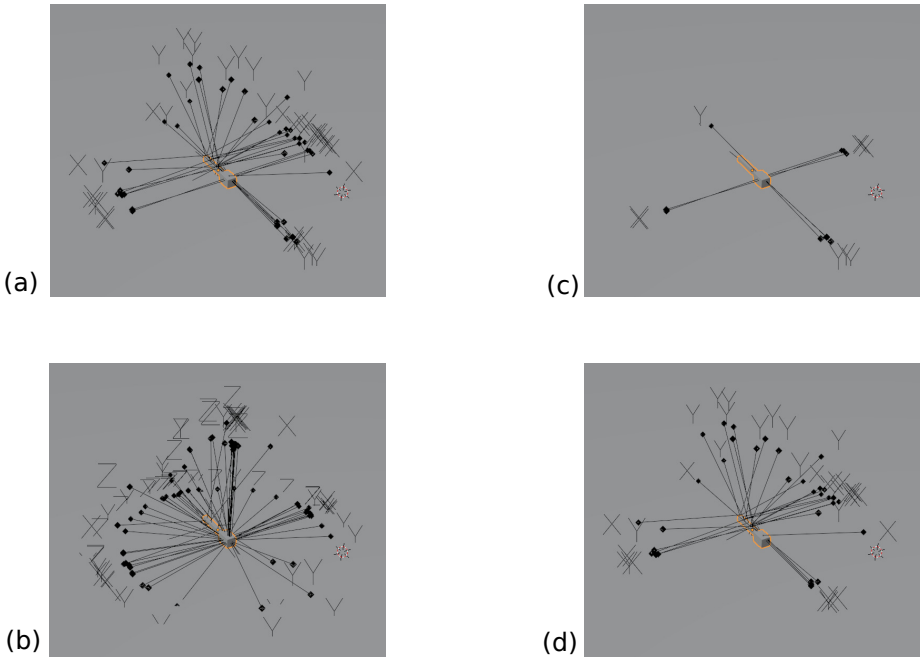


Fig. 4.13. Proximity-based classification of pick grasps visualized inside Blender: (**a**) pick grasps, (**b**) place grasps, (**c**) valid grasps out of all the pick grasps, (**c**) invalid grasps out of all the pick grasps

proximity to place grasps, valid and invalid grasps. The group of pick grasps with a corresponding place grasp is classified as valid, while the pick grasps with no corresponding place grasps are classified as invalid. We considered two grasps "corresponding" if the distance between their origins was less than 5mm and their relative orientation difference is less than 10 degrees. We also pre-filtered the grasps to remove those that were too similar.

Once all the pick grasps were classified as valid or invalid, a JSON object is generated that contains a list of 2D grasps, represented by a 2D point (in pixel units), an angle, and a label (1 for valid grasps and 0 for invalid grasps). After this, an RGB-D image is rendered where the RGB image was saved as a PNG file, and the depth data is saved separately as a NumPy array. We also generated segmentation masks for each rendered frame using the Blender Annotation Tool, although we did not utilize them in our experiments.

The Blender scene was prepared in a way that the object's location along the X and Y axes and its orientation along the Z axis were randomized throughout each frame. This process resulted in a collection of rendered synthetic depth images and corresponding 2D grasps for multiple object poses. The proposed pipeline uses Blender with Python programming language to automate the rendering process and generate synthetic data. Figure 4.13. shows the pick grasps, the place grasps, and the separation of the pick grasps into valid and invalid grasps.

A Python script is utilized to generate a synthetic grasp dataset in the format of Dex-Net 2.0 using the output from Blender. The script loads the depth images and corresponding 2D grasps. It rotates, crops, and resizes the depth images to create 32×32 sized depth images for each grasp, where the grasp point is positioned in the center of the image, and the gripper's opening direction is horizontal. Since we only generate grasps which are appropriate for picking (but not necessarily for placing), additional 12 invalid grasps are added along with each grasp (both valid and invalid), which have the same position as the

original grasp, but they are rotated in either a positive or negative direction along the Z axis in increments of 15° (up to $+90°$and $-90°$). In our experiments, adding these invalid grasps significantly improved the accuracy of the predicted grasps' orientation. During our experiments, we also found that cropping a 128×128 square area of the depth image first and rescaling that to 32×32 proved better than cropping 64×64 or 32×32 directly, although the best value most likely depends on the specific setup (camera-object distance). The 32×32 depth images are bundled in a single NumPy array and saved as an npz file. Similarly, the labels and the robot gripper poses (containing the depth of the grasp points) are saved as npz archives.

Finally, the GQCNN, which was pre-trained on the Dex-Net 2.0 dataset, is fine-tuned on our automatically generated synthetic grasp planning dataset, using the scripts provided for fine-tuning with the GQCNN implementation[6]. The fine-tuning process involved training for 60 epochs, with a train/validation split of 0.9/0.1. We utilized a sparse loss with a momentum-based optimizer, a decay rate of 0.999, and a base learning rate of 0.01.

Once the training phase is complete, we assess the GQCNN models using synthetic depth images from Blender and the Moveit! Task Constructor (MTC) framework [216]. This framework enables us to define complex tasks, such as our pick-and-place experiment, using a modular approach that breaks down the task into elementary subtasks (known as stages). MTC performs the robot motion planning for the entire task, taking collisions in the scene into account. In our experiments, we introduce a grasp generator stage that uses the ROS-based GQCNN grasp planning service from the GQCNN implementation to obtain the best grasp from a GQCNN for a given depth image. With this new generator stage, we adapt the MTC pick-and-place demo scene for our experiments, which utilizes the Franka Emika Panda robot. We recreate the table, object, and camera in the MTC scene to match their relative positions in Blender. To assess the GQCNN models, we perform robot motion planning using MTC, using the best predicted grasp from the GQCNN model. During the evaluation, we define a fixed place pose for the object and utilize MTC to identify the stage in which task execution failure occurs. We consider grasps that result in a collision between the robot arm/hand and the table during the inverse kinematics computation stage for placing as failed grasps, while those that do not result in such collisions are deemed successful. It's worth noting that this evaluation solely considers geometric criteria in the form of collisions, with no regard for the quality of the grasps concerning force closure or other dynamic properties. Nevertheless, we hypothesize that the predicted grasps using our fine-tuned GQCNN models will be robust, given that the pre-trained GQCNNs already consider dynamics and our positive grasps dataset is generated from Graspit!, which also provides robust grasps. However, real-world trials are necessary to validate this hypothesis in the future.

In summary, the main components of the synthetic data generation pipeline, GQCNN training, and evaluation process, with their purpose and requirements, are:

- **Graspit!**: This tool is utilized for automated and robust grasp generation in two scenarios: picking and placing. For both scenarios, we require 3D models of the robot gripper, the object, and the table. In the picking scenario, the object must be placed in a stable, natural position on the table, while in the placing scenario, the object must be positioned in its final pose relative to all the relevant assembly parts

[6]Available online at `https://github.com/BerkeleyAutomation/gqcnn`

that preceded it. Grasp poses for both scenarios are expressed as 3D poses of the robot gripper's frame relative to the Graspit! world frame.

- **Blender**: Used for generating synthetic RGB-D frames and ground-truth grasp information for training and evaluating the GQCNN models. To set up the Blender scene, a camera and 3D models for the object and the table are necessary. For each rendered frame, the object should be randomly placed in a natural lying pose on the table within the camera's field of view. The object position along the X and Y axes and its rotation around the Z axis of the Blender world frame should be randomized. The intrinsic parameters of the Blender camera must match those of the camera used in the real-world setup, which can be determined by camera calibration. Section 4.2.3 discusses the impact of different extrinsic camera parameters during training and inference. The pick and place grasps from Graspit! are transformed into a common coordinate frame within Blender, namely the object's frame, and the pick grasps are classified as valid or invalid based on their proximity to place grasps. The valid and invalid grasp points are projected onto the image plane using Blender's camera projection, and their orientation is used to create ground-truth grasp information for training the GQCNN models.
- **Moveit! Task Constructor**: Used for automating the assessment of the GQCNN models once trained. This tool requires the robot model, 3D models of the object and table, and a cylinder-shaped dummy object to represent the camera. Although the camera, object, and table can be placed anywhere within the robot's workspace, their relative poses must match those in Blender when the depth image was generated for the model under evaluation. MTC performs motion planning for the entire task, taking into account collision checking, using the grasp predicted by the GQCNN and a fixed place pose. The prediction is regarded as successful if the motion planning is accomplished without collisions, and as unsuccessful, if the inverse kinematics solution for the place pose fails due to a collision.

Figure 4.14. shows the flowchart representation of the automated synthetic grasp dataset generation.

Experimental setup

For our experiments, we evaluate the performance of different GQCNN models by assessing the success rate of collision-free motion plans in a peg-in-hole-like insertion task. The experimental setup, visualized by RViz, can be seen in Figure 4.15. The peg object in the experiments is made up of two box primitives, one of which is sized 10cm in the X direction and 1cm in both Y and Z directions, with the origin being the geometric center of the box, the other one is sized 3cm in X and 2cm in Y and Z directions and is located 3.5cm in the X direction. In the setup, the object is lying on its side (the Z axis of the object frame pointing upwards) on the surface of a table. The table is simply represented as a flat surface.

Since the 3D grasp pose predicted by the GQCNNs is oriented according to the line between the depth camera and the grasp point, we included an additional platform for placing in MTC to avoid unnecessary collisions between the robot arm and the table. In-

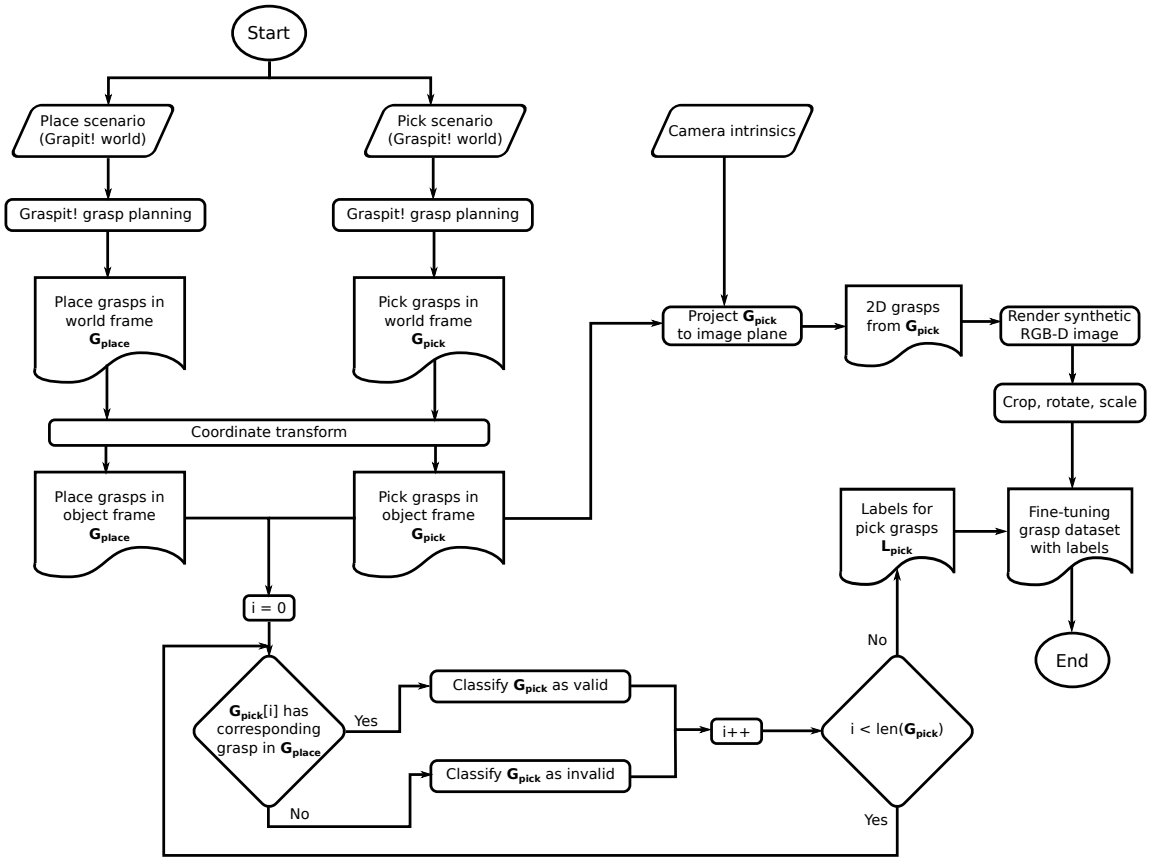


Fig. 4.14. Flowchart of the automated synthetic grasp dataset generation procedure

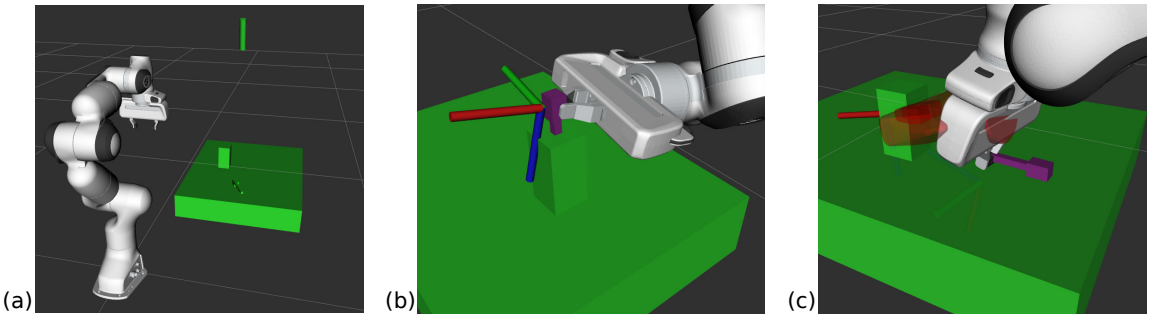


Fig. 4.15. Experimental setup used for evaluation: (**a**) Scene setup with the robot, object camera, and table, (**b**) Planned grasps which do not result in collisions are evaluated as successful, (**c**) Grasps that result in collisions between the robot and the table are evaluated as unsuccessful

TABLE 4.5
EVALUATION OF OUR FINE-TUNED GQCNN AGAINST THE DEX-NET 2.0 PRE-TRAINED GQCNN FOR DEPTH IMAGES GENERATED FROM A CAMERA WITH THE SAME EXTRINSIC PARAMETERS AS IN THE TRAINING SETUP.

Model	Success rate (task)	Success rate (pick)	Success rate (place)
Dex-Net 2.0 GQCNN	0.65	1.0	0.65
Fine-tuned GQCNN	**0.9**	1.0	0.9

side Graspit! we simply use the "table" object instead. For the gripper (Panda Hand[7]), we defined the virtual contacts using the user interface of Graspit! and modified the inventor files to include appropriate scaling so the size of the gripper in Graspit! matches the real-life gripper. We ran the Graspit! grasp planning a total of 3 times for pick grasps and 5 times for place grasps, from which we acquired 23-26 pick grasps, and around 30 place grasps after the initial filtering (removing multiples of pick grasps and place grasps or grasps with high contact energy). Based on geometric proximity, 5 pick grasps were classified as valid, and the rest as invalid (see Figure 4.13).

To create the fine-tuning dataset, we used the first 20 RGB-D frames generated in Blender. The object pose was randomized for each of these frames, and a total of 489 pick grasps were identified as either valid or invalid. To ensure accurate prediction of gripper orientation, we added 12 invalid grasps for each of the 489 original ones while converting the data into Dex-Net 2.0 format. This process resulted in 6357 grasps, of which only 1.57% were positively labeled.

We performed fine-tuning on the Dex-Net 2.0 pre-trained GQCNN model using the aforementioned dataset. Fine-tuning was conducted for 60 epochs with a batch size of 64. Subsequently, we compared the fine-tuned model with the original Dex-Net 2.0 pre-trained model by evaluating the success rate of pick-place operations using MTC. We used 20 additional rendered frames, which were not part of the training or validation data, and shared the same camera placement as the training setup. Additionally, we generated 40 other RGB-D frames with a different camera placement than the training setup. We report the results of these evaluations separately to showcase the sensitivity of our approach to variations in the setup.

Results using camera extrinsic parameters from training setup

Table 4.5. shows the results of our evaluation using synthetic depth images from Blender and the same camera extrinsics as in the setup for generating the training data. In these evaluation frames, only the object pose was randomized. It is evident from the table that the fine-tuned GQCNN performed significantly better than the original Dex-Net 2.0 pre-trained model.

It is worth noting that during our experiments, none of the predicted grasps from either the original Dex-Net 2.0 pre-trained model or the fine-tuned model resulted in a collision during the object picking phase. The failed grasps that were observed were due to the models' inability to predict a suitable grasp in the given task context, leading to collisions

[7]We based our solution on the model provided here: `https://github.com/JenniferBuehler/graspit-pkgs/issues/55#issue-515423230`

TABLE 4.6
EVALUATION OF OUR FINE-TUNED GQCNN AGAINST THE DEX-NET 2.0 PRE-TRAINED GQCNN FOR DEPTH IMAGES GENERATED FROM A CAMERA WITH THE DIFFERENT EXTRINSIC PARAMETERS FROM THE TRAINING SETUP.

Model	Success rate (task)	Success rate (pick)	Success rate (place)
Dex-Net 2.0 GQCNN	0.5	1.0	0.5
Fine-tuned GQCNN	**0.55**	1.0	0.55

during the object placing phase. It is expected that the success rate for picking the object would be higher than placing it because only a few grasps that are feasible for picking the object would also be suitable for placing it. Therefore, the likelihood of predicting a successful grasp for both the picking and placing phases is lower. Our results indicate that fine-tuning a GQCNN on a small number of synthetic grasps labeled with task-specific information can significantly improve the model's ability to predict a grasp that will be successful for the entire task.

Our approach can be easily adapted to new objects or assembly tasks with the help of the assembly information, scene setup, and 3D models of objects and the robot. This allows for flexible adjustment of the grasp prediction pipeline for new scenarios, even within a single day. The entire process, including scene setup in Blender, fine-tuning dataset generation, and GQCNN model training, can be quickly accomplished, especially since the scene, robot, and camera setups typically do not require frequent modifications compared to the assembly tasks which may change frequently.

Results using different camera extrinsic parameters

In Table 4.6., we present the results obtained by comparing the performance of our fine-tuned GQCNN model with the Dex-Net 2.0 pre-trained GQCNN on 40 synthetic depth images. These images were generated using a camera location different from the one used in the setup for generating the fine-tuning dataset. It is observed that the performance difference between the two models is not as significant as it was in the case of using camera extrinsics from the training setup. This suggests that the fine-tuned model's predictions for setups different from the training setup are more aligned with the predictions of the original GQCNN. However, this limitation could potentially be overcome by automatically varying the camera pose during the generation of the training dataset in Blender. This approach holds promise for future improvements to make the fine-tuned GQCNN models more robust to changes in camera setup.

4.3 New scientific results

Thesis 3

I defined and realized two procedures to create and label object segmentation datasets automatically. I showed that these datasets can be utilized to train deep learning models for visual perception tasks in robotic manipulation, such as scene recognition or object and grasp detection. The first method employs the projection algorithm (6.) to generate instance segmentation masks for real-world images with known geometry. The second method utilizes computer graphics to generate and label synthetic rendered images automatically.

Sub-thesis 3.1

I showed that incorporating synthetic samples during the training process of a continual learning model that utilizes experience replay can significantly enhance its forward transfer. I demonstrated the validity of this statement by creating a synthetic version of a subset of the OpenLORIS Object dataset and comparing two continual learning models: one that was trained using synthetic data and one that was not.

Sub-thesis 3.2

I introduced a solution to address the "reality gap" challenge when transferring deep learning models trained on synthetic data to the real world. This solution, named FTRG (Filling The Reality Gap), involves the integration of automated real and synthetic data annotation techniques to enable a smooth transition between synthetic and real components within a single image. Through comparative analysis of Mask R-CNN models trained on datasets utilizing different methods to overcome the "reality gap", including domain randomization and photorealistic synthetic data, I demonstrated that the FTRG method can achieve beyond the state-of-the-art performance.

Sub-thesis 3.3

I proposed a method for creating task-specific grasp detection datasets for robotic assembly tasks that consider grasp poses that do not necessarily result in collision-free placing. This method involves automated synthetic data generation, labeling, and sampling-based grasp planning techniques, leveraging known object and assembly geometries and assembly order. I demonstrated the effectiveness of the method by fine-tuning a GQCNN network on a generated dataset and showing that the fine-tuned GQCNN outperforms the original in an asymmetric insertion-type robotic assembly task.

Related publications: [KA7, KA8, KA9, KA10, KA11]

Part 5

SUMMARY

This Thesis aims to enhance the current state-of-the-art solutions for robotics based on deep learning approaches. The research conducted within this thesis addresses relevant issues faced by the industry, including the requirement for an extensive dataset to train deep learning models, the time-intensive process of labeling training data, challenges associated with online predictions, leveraging non-RGB modalities with transfer learning, and ensuring the trained models possess good generalization capabilities. The presented solutions are not only scientifically relevant but also practically applicable.

In Part 2. of this Thesis, an unsupervised online clustering approach for state discovery and anomaly detection in robot applications is presented. The proposed solution addresses the challenges of requiring a large training dataset and time-consuming labeling procedure by utilizing a dynamically trained ensemble of OCSVM models with unsupervised learning (Algorithm 1.). This approach eliminates the need for a pre-assembled offline training dataset and instead learns from online data in the form of a data stream from the robot system.

In addition to the experimental setup presented in this Thesis, which evaluates the proposed clustering approach on a representative real-world collaborative robot application, the method has also been successfully applied in various other industrial scenarios. For instance, during the 2018 Pioneers Industry 4.0 Hackathon challenge held in Linz, it was deployed in a packaging robot application for automatic state discovery and anomaly detection using a data stream of the robot's Cartesian pose. It was part of the winning proposal in the Digital Twin category by the joint teams of REACH Solutions and MaxWhere. Another industrial application was carried out under the S3FOOD EU-funded project for digital innovation in the agrifood industry in collaboration with the Innoskart ICT Cluster, Mortoff Ltd., and GoodMills Magyarország, where the proposed approach was applied for system supervision by automatically clustering and recognizing failures during the flour packaging process based on a data stream from several sensor sources in an industrial milling company.

In addition, experimental results have demonstrated that the proposed method can be utilized to evaluate generative machine learning models. This finding has broader implications beyond robotics and system supervision, particularly given the current surge of interest in such technologies, fueled by the success of diffusion models [99].

In Part 3. of this Thesis, the proposed cross-modal mapping approach is described in (3.2), which incorporates non-RGB modalities (grayscale image and optical flow) into

the prediction pipeline of a deep learning model that uses an RGB pre-trained feature extractor and transfer learning. The approach is evaluated using the OFSNet model for moving obstacle detection in a mobile robot navigation scenario. This technique addresses the challenge of requiring a large labeled training dataset by using transfer learning for data modalities where pre-trained models are not available due to the lack of large-scale labeled data. The Thesis also proposes a corresponding training strategy in (3.8) and a loss function in (3.7) for object segmentation tasks with class imbalance.

The experimental results illustrate the real-world application of the proposed cross-modal mapping approach, compound loss function, and trained OFSNet model in an industrial AGV system prototype. This application was developed in collaboration with GAMMA DIGITAL Kft. under the GINOP 2.2.1-15-2017-00097 program (Gazdaságfejlesztési és Innovációs Operatív Program).

Part 4. of this Thesis proposes two automated visual dataset generation methods for robotic manipulation: one for collecting and annotating real image data using a robotic arm (Algorithm 6.) and another for generating and annotating synthetic datasets for object segmentation using computer graphics. These approaches tackle the time-consuming task of labeling large datasets by automating the data collection and annotation procedures. Moreover, the thesis introduces the FTRG method, which combines both of these automated dataset generation pipelines to address the challenge of training models with good generalization capabilities.

In addition to demonstrating the benefits of the proposed dataset generation methods on experimental results in the context of object segmentation for robotic manipulation, image recognition for continuous learning, and grasp planning in robotic assembly tasks, these methods have been successfully applied in other robotic manipulation tasks. Specifically, the proposed methods have been utilized in the development of a real-world robot setup for the automated, flexible preparation of digital pathology archives in the 2019-1.3.1-KK-2019-00007 (KIKOK) project, as well as in the agrifood industry for the development of a robotized mushroom harvesting setup.

Other publications related to the Ph.D. thesis and the accompanying research work: [KAO1, KAO3]

5.1 Future Work

As with any scientific research, a PhD Thesis represents only a snapshot of our current understanding of a particular topic. While the research presented in this thesis provides valuable insights and contributes to the existing literature, there are still many potential directions for future research that remain unexplored. In this section, possible future research directions in deep learning-based approaches for robotics are discussed, building upon the findings and limitations of the methods proposed in this Thesis.

When using the clustering algorithm (Algorithm 1.) described in the first thesis (Part 2), it is important to consider the sampling rate and expected length of events when selecting the parameters of the method, such as w and n. In the future, it may be possible to develop a metric or evaluation method to compare OCSVM ensembles without the need for a labeled dataset. This would facilitate the "vertical development" of the proposed solution by enabling the use of multiple ensembles at the same time, each trained with different parameters, to increase robustness and reduce the need for parameter tuning.

One possible direction for further research is to explore alternative unsupervised methods that can handle binary data more effectively than OCSVM predictors in the current algorithm. An example of such an alternative approach could be a modified random forest method, as proposed in [217]. Incorporating multiple unsupervised predictors into the algorithm could broaden the range of data sources that the method can be applied to.

In terms of evaluating generative models, one important drawback of the proposed method is its inability to detect mode collapse. However, there is potential for improvement in the future, as a metric could be integrated into the method to measure data diversity and thus also account for mode collapse.

As showcased in Part 3. of the Thesis, the proposed cross-modal mapping approach (3.2) only considers object appearance (grayscale image) and motion (optical flow) information. In the evaluation, it was observed that the performance of U-Net variants utilizing the cross-modal mapping approach was inferior to that of MATNet [160], which leverages object boundaries as well for precise segmentation predictions. A promising direction for future research would be to expand the cross-modal mapping formulation to encompass object boundaries. Such an extension could potentially enhance the accuracy of the predicted segmentation masks for moving objects.

Likewise, the integration of cross-modal mapping for alternative modalities could be explored in various fields, such as employing depth data for object detection or depth and surface normal data in computer graphics. Additionally, the implications of different cross-modal mapping methods could be further investigated, such as comparing the utilization of polar colormap-based RGB optical flow encoding with our strategy of combining optical flow and grayscale image to form RGB images.

The findings presented in Part 4. are particularly promising for future advancements. The experimental results show that the real data annotation process necessitates precise camera and robot calibration. This technique could be enhanced substantially if a more precise and adaptable calibration procedure could be established for the robot-camera setup. To achieve this, it may be necessary to develop and integrate hand-eye calibration methodologies into the dataset generation pipeline.

The automatic annotation of real images has certain limitations with respect to the object models. The method is currently only capable of handling closed object models that

have distinct inside and outside regions. Also, significant variations in the size of object models might cause anomalies when computing the overlap between projected triangles from the STL representation. Furthermore, the algorithm cannot handle objects that intersect with each other. To address these limitations, further enhancements can be made to the proposed algorithm (Algorithm 6).

The synthetic dataset generation pipeline can be enhanced by improving BAT's capability to automatically generate various types of annotations, such as depth data, optical flow, surface normals, and object boundaries, in addition to segmentation-type annotations for the rendered images. Also, the FTRG method can be improved by allowing the direct import of real-life camera calibration results into the computer graphics software instead of relying on camera tracking, which involves extra manual configuration. Moreover, future research can explore the potential of the FTRG method in pre-training DL models and analyzing the generalization abilities of the pre-trained models.

A larger-scale experiment could be conducted for continuous learning with experience replay, where all four environmental factors are randomized, utilizing a larger subset or the whole OpenLORIS Object dataset.

Regarding the grasp planning approach, the classification of grasps as valid or invalid currently only takes into account the geometry of the robot gripper. However, utilizing MTC not only as an evaluation tool but also as a means to determine the validity of pick and place grasp proposals generated through the sampling-based approach could result in a more precise grasp planning dataset that is specific to a particular setup involving a robot and an environment. Moreover, the applicability of this approach to other setups, such as those with different camera extrinsics, environments, or manipulators, should be further investigated.

REFERENCES

[1] R. A. Fisher, "Iris," UCI Machine Learning Repository, 1988, DOI: https://doi.org/10.24432/C56C76.

[2] S. Aeberhard and M. Forina, "Wine," UCI Machine Learning Repository, 1991, DOI: https://doi.org/10.24432/C5PC7J.

[3] E. Alpaydin and C. Kaynak, "Optical Recognition of Handwritten Digits," UCI Machine Learning Repository, 1998, DOI: https://doi.org/10.24432/C50P49.

[4] Q. She, F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang *et al.*, "OpenLORIS-Object: A robotic vision dataset and benchmark for lifelong deep learning," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 4767–4773.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[6] L. Deng, D. Yu *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8599–8603.

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[10] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[11] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object Recognition with Gradient-Based Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. [Online]. Available: https://doi.org/10.1007/3-540-46805-6_19

[12] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[13] Q. V. Le, R. Monga, M. Devin, G. Corrado, K. Chen, M. Ranzato, J. Dean, and A. Y. Ng, "Building high-level features using large scale unsupervised learning," *CoRR*, vol. abs/1112.6209, 2011. [Online]. Available: http://arxiv.org/abs/1112.6209

[14] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[15] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[16] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational intelligence magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[17] J. Johnson, A. Karpathy, and F. Li, "Densecap: Fully convolutional localization networks for dense captioning," *CoRR*, vol. abs/1511.07571, 2015. [Online]. Available: http://arxiv.org/abs/1511.07571

[18] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520.

[19] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[20] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: A review of recent research," *CoRR*, vol. abs/1707.07217, 2017. [Online]. Available: http://arxiv.org/abs/1707.07217

[21] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013. [Online]. Available: https://doi.org/10.1177/0278364913495721

[22] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation - the state of the art," *CoRR*, vol. abs/1701.08878, 2017. [Online]. Available: http://arxiv.org/abs/1701.08878

[23] L. Tai and M. Liu, "Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not," *CoRR*, vol. abs/1612.07139, 2016. [Online]. Available: http://arxiv.org/abs/1612.07139

[24] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[25] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.

[26] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[27] C. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 1, pp. 10–24, 2017.

[28] C. L. P. Chen and Z. Liu, "Broad learning system: A new learning paradigm and system without going deep," in *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, May 2017, pp. 1271–1276.

[29] A. Yamaguchi and C. G. Atkeson, "Differential dynamic programming for graph-structured dynamical systems: Generalization of pouring behavior with different skills," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 1029–1036.

[30] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[31] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 17–36.

[32] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 37–49.

[33] G. E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_32

[34] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *CoRR*, vol. abs/1703.06907, 2017. [Online]. Available: http://arxiv.org/abs/1703.06907

[35] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," *CoRR*, vol. abs/1709.07857, 2017. [Online]. Available: http://arxiv.org/abs/1709.07857

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[37] E. Liberty, R. Sriharsha, and M. Sviridenko, "An algorithm for online k-means clustering," in *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*. SIAM, 2016, pp. 81–89.

[38] P. Xu, C.-H. Chang, and A. Paplinski, "Self-organizing topological tree for online vector quantization and data clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 3, pp. 515–526, 2005.

[39] M. M. Campos and G. A. Carpenter, "S-tree: self-organizing trees for data clustering and online vector quantization," *Neural Networks*, vol. 14, no. 4-5, pp. 505–525, 2001.

[40] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.

[41] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR, abs/1206.5538*, vol. 1, no. 2665, p. 2012, 2012.

[42] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *Journal of machine learning research*, vol. 2, no. Dec, pp. 125–137, 2001.

[43] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," *Advances in neural information processing systems*, vol. 12, 1999.

[44] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[45] B. Schölkopf, "The kernel trick for distances," *Advances in neural information processing systems*, vol. 13, 2000.

[46] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[47] 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.

[48] M. Huh, P. Agrawal, and A. A. Efros, "What makes imagenet good for transfer learning?" *CoRR*, vol. abs/1608.08614, 2016. [Online]. Available: http://arxiv.org/abs/1608.08614

[49] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Advances in neural information processing systems*, vol. 27, 2014.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[52] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[53] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *CoRR*, vol. abs/1703.02949, 2017. [Online]. Available: http://arxiv.org/abs/1703.02949

[54] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.

[55] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *CoRR*, vol. abs/1710.06537, 2017. [Online]. Available: http://arxiv.org/abs/1710.06537

[56] S. James and E. Johns, "3D Simulation for Robot Arm Control with Deep Q-Learning," *CoRR*, vol. abs/1609.03759, 2016. [Online]. Available: http://arxiv.org/abs/1609.03759

[57] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, T. Chen, A. Hutter, S. Zakharov, H. Kosch, and J. Ernst, "Depthsynth: Real-time realistic synthetic data generation from CAD models for 2.5D recognition," *CoRR*, vol. abs/1702.08558, 2017. [Online]. Available: http://arxiv.org/abs/1702.08558

[58] Y. You, X. Pan, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *CoRR*, vol. abs/1704.03952, 2017. [Online]. Available: http://arxiv.org/abs/1704.03952

[59] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders." in *IJCAI*, 2015, pp. 4119–4125.

[60] J. Lee, "A survey of robot learning from demonstrations for human-robot collaboration," *CoRR*, vol. abs/1710.08789, 2017. [Online]. Available: http://arxiv.org/abs/1710.08789

[61] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[62] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," *CoRR*, vol. abs/1703.07326, 2017. [Online]. Available: http://arxiv.org/abs/1703.07326

[63] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," *CoRR*, vol. abs/1709.10089, 2017. [Online]. Available: http://arxiv.org/abs/1709.10089

[64] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[65] G. Sutanto, N. D. Ratliff, B. Sundaralingam, Y. Chebotar, Z. Su, A. Handa, and D. Fox, "Learning latent space dynamics for tactile servoing," *CoRR*, vol. abs/1811.03704, 2018. [Online]. Available: http://arxiv.org/abs/1811.03704

[66] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[67] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[68] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[69] S. Valipour, C. P. Quintero, and M. Jägersand, "Incremental learning for robot perception through HRI," *CoRR*, vol. abs/1701.04693, 2017. [Online]. Available: http://arxiv.org/abs/1701.04693

[70] P. Taylor, A. W. Black, and R. Caley, "The architecture of the festival speech synthesis system," in *THE THIRD ESCA WORKSHOP IN SPEECH SYNTHESIS*, 1998, pp. 147–151.

[71] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[72] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," *CoRR*, vol. abs/1412.3128, 2014. [Online]. Available: http://arxiv.org/abs/1412.3128

[73] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *Computing Research Repository (CoRR)*, vol. abs/1505.04597, 2015, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1505.04597

[74] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[75] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *CoRR*, vol. abs/1612.00593, 2016. [Online]. Available: http://arxiv.org/abs/1612.00593

[76] A. Zeng, K. T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1386–1383.

[77] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3D object classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 656–664.

[78] M. Schwarz, H. Schulz, and S. Behnke, "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1329–1335.

[79] L. Bo, X. Ren, and D. Fox, *Unsupervised Feature Learning for RGB-D Based Object Recognition*. Heidelberg: Springer International Publishing, 2013, pp. 387–402. [Online]. Available: https://doi.org/10.1007/978-3-319-00065-7_27

[80] A. Schmitz, Y. Bansho, K. Noda, H. Iwata, T. Ogata, and S. Sugano, "Tactile object recognition using deep learning and dropout," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 1044–1050.

[81] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[82] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[83] P. Martinez-Gonzalez, S. Oprea, A. Garcia-Garcia, A. Jover-Alvarez, S. Orts-Escolano, and J. Garcia-Rodriguez, "Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation," *Virtual Reality*, vol. 24, no. 2, pp. 271–288, 2020.

[84] A. Garcia-Garcia, P. Martinez-Gonzalez, S. Oprea, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Jover-Alvarez, "The robotrix: An extremely photorealistic and very-large-scale indoor dataset of sequences with robot trajectories and interactions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6790–6797.

[85] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind, "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 912–10 922.

[86] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European conference on computer vision*. Springer, 2012, pp. 746–760.

[87] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser, "Physically-based rendering for indoor scene understanding using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5287–5295.

[88] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation?" in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2678–2687.

[89] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *arXiv preprint arXiv:1809.10790*, 2018.

[90] L. Eversberg and J. Lambrecht, "Generating images with physics-based rendering for an industrial object detection task: Realism versus domain randomization," *Sensors*, vol. 21, no. 23, p. 7901, 2021.

[91] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 969–977.

[92] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[93] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[94] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4340–4349.

[95] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, "Structured domain randomization: Bridging the reality gap by context-aware synthetic data," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7249–7255.

[96] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" *arXiv preprint arXiv:1610.01983*, 2016.

[97] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8821–8831.

[98] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[99] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[100] A. Hanbury, "A survey of methods for image annotation," *Journal of Visual Languages & Computing*, vol. 19, no. 5, pp. 617–627, 2008.

[101] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[102] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," *arXiv preprint arXiv:1806.06988*, 2018.

[103] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015. [Online]. Available: http://arxiv.org/abs/1511.06434

[104] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[105] M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham, "Addressing Concept-Evolution in Concept-Drifting Data Streams," in *2010 IEEE International Conference on Data Mining*, 2010, pp. 929–934.

[106] L. Morissette and S. Chartier, "The k-means clustering technique: General considerations and implementation in mathematica," *Tutorials in Quantitative Methods for Psychology*, vol. 9, no. 1, pp. 15–24, 2013.

[107] T. Kohonen and T. Honkela, "Kohonen network," *Scholarpedia*, vol. 2, no. 1, p. 1568, 2007.

[108] T. Kohonen, *Self-organization and associative memory*. Springer Science & Business Media, 2012, vol. 8.

[109] Z. Ghafoori, S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. A. Leckie, "Anomaly detection in non-stationary data: Ensemble based self-adaptive OCSVM," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 2476–2483.

[110] A. Gretton and F. Desobry, "On-line one-class support vector machines. an application to signal segmentation," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 2. IEEE, 2003, pp. II–709.

[111] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3. IEEE, 2003, pp. 1741–1745.

[112] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *the Journal of machine Learning research*, vol. 9, pp. 1871–1874, 2008.

[113] E. Hazan, T. Koren, and N. Srebro, "Beating SGD: Learning SVMs in sublinear time," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[114] S. Vempati, A. Vedaldi, A. Zisserman, and C. Jawahar, "Generalized RBF feature maps for efficient detection." in *BMVC*, 2010, pp. 1–11.

[115] M. Claesen, F. De Smet, J. A. Suykens, and B. De Moor, "Fast prediction with SVM models containing RBF kernels," *arXiv preprint arXiv:1403.0736*, 2014.

[116] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel SVMs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 66–77, 2012.

[117] Z.-Q. Zeng, H.-B. Yu, H.-R. Xu, Y.-Q. Xie, and J. Gao, "Fast training support vector machines using parallel sequential minimal optimization," in *2008 3rd international conference on intelligent system and knowledge engineering*, vol. 1. IEEE, 2008, pp. 997–1001.

[118] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[119] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[120] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[121] I. Färber, S. Günnemann, H.-P. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek, "On using class-labels in evaluation of clusterings," in *MultiClust: 1st international workshop on discovering, summarizing and using multiple clusterings held in conjunction with KDD*, 2010, p. 1.

[122] A. Dehghani, O. Sarbishei, T. Glatard, and E. Shihab, "A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors," *Sensors*, vol. 19, no. 22, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/5026

[123] 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 E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[124] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[125] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs created equal? a large-scale study," *Advances in neural information processing systems*, vol. 31, 2018.

[126] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[127] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in neural information processing systems*, vol. 29, 2016.

[128] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[129] V. Khrulkov and I. Oseledets, "Geometry score: A method for comparing generative adversarial networks," in *International conference on machine learning*. PMLR, 2018, pp. 2621–2629.

[130] 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.

[131] A. Casanova, M. Careil, J. Verbeek, M. Drozdzal, and A. Romero-Soriano, "Instance-conditioned GAN," *CoRR*, vol. abs/2109.05070, 2021. [Online]. Available: https://arxiv.org/abs/2109.05070

[132] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *CoRR*, vol. abs/1809.11096, 2018. [Online]. Available: http://arxiv.org/abs/1809.11096

[133] H. Caesar, J. R. R. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and stuff classes in context," *CoRR*, vol. abs/1612.03716, 2016. [Online]. Available: http://arxiv.org/abs/1612.03716

[134] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org, visited on 2019-04-15. [Online]. Available: https://www.tensorflow.org/

[135] S. Shalev-Shwartz and N. Srebro, "SVM optimization: inverse dependence on training set size," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 928–935.

[136] M. Piccardi, "Background subtraction techniques: a review," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, vol. 4. IEEE, 2004, pp. 3099–3104.

[137] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 657–662, April 2006.

[138] K. L. Chan, "Segmentation of moving objects in image sequence based on perceptual similarity of local texture and photometric features," *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 1, p. 62, Jul 2018. [Online]. Available: https://doi.org/10.1186/s13640-018-0308-4

[139] S. Varadarajan, P. Miller, and H. Zhou, "Spatial mixture of gaussians for dynamic background modelling," in *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, Aug 2013, pp. 63–68.

[140] S. S. Mohamed, N. M. Tahir, and R. Adnan, "Background modelling and background subtraction performance for object detection," in *2010 6th International Colloquium on Signal Processing its Applications*, May 2010, pp. 1–6.

[141] P. Patil and S. Murala, "FgGan: A cascaded unpaired learning for background estimation and foreground segmentation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Jan 2019, pp. 1770–1778.

[142] Y. Sun, M. Liu, and M. Q. . Meng, "Active perception for foreground segmentation: An RGB-D data-based background modeling method," *IEEE Transactions on Automation Science and Engineering*, pp. 1–14, 2019.

[143] D. Zamalieva and A. Yilmaz, "Background subtraction for the moving camera: A geometric approach," *Computer Vision and Image Understanding*, vol. 127, pp. 73–85, 2014.

[144] C. Kim and J. Hwang, "A fast and robust moving object segmentation in video sequences," in *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, Oct 1999, pp. 131–134 vol.2.

[145] P. Smith, T. Drummond, and R. Cipolla, "Layered motion segmentation and depth ordering by tracking edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 4, pp. 479–494, April 2004.

[146] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *Fourth International Conference on Image and Graphics (ICIG 2007)*, Aug 2007, pp. 519–523.

[147] I. Kokkinos, "Surpassing humans in boundary detection using deep learning," *Computing Research Repository (CoRR)*, vol. abs/1511.07386, 2015, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1511.07386

[148] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3982–3991.

[149] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 193–202.

[150] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," *Computing Research Repository (CoRR)*, vol. abs/1807.10097, 2018, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1807.10097

[151] Y. Liu, M. Cheng, J. Bian, L. Zhang, P. Jiang, and Y. Cao, "Semantic edge detection with diverse deep supervision," *Computing Research Repository (CoRR)*, vol. abs/1804.02864, 2018, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1804.02864

[152] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4545–4554.

[153] S. Gupta, R. B. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," *CoRR*, vol. abs/1407.5736, 2014, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1407.5736

[154] J. Kao, D. Tian, H. Mansour, A. Vetro, and A. Ortega, "Moving object segmentation using depth and optical flow in car driving sequences," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 11–15.

[155] W. B. Thompson and T.-C. Pong, "Detecting moving objects," *International Journal of Computer Vision*, vol. 4, no. 1, pp. 39–57, Jan 1990. [Online]. Available: https://doi.org/10.1007/BF00137442

[156] P. Bideau, A. RoyChowdhury, R. R. Menon, and E. Learned-Miller, "The best of both worlds: Combining CNNs and geometric constraints for hierarchical motion segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 508–517.

[157] A. G. Bors and I. Pitas, "Optical flow estimation and moving object segmentation based on median radial basis function network," *IEEE Transactions on Image Processing*, vol. 7, no. 5, pp. 693–702, May 1998.

[158] J. Hur and S. Roth, "Joint optical flow and temporally consistent semantic segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 163–177.

[159] K. Fragkiadaki, P. Arbelaez, P. Felsen, and J. Malik, "Learning to segment moving objects in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4083–4090.

[160] T. Zhou, S. Wang, Y. Zhou, Y. Yao, J. Li, and L. Shao, "Motion-attentive transition for zero-shot video object segmentation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 13 066–13 073.

[161] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.

[162] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Computing Research Repository (CoRR)*, vol. abs/1411.4038, 2014, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1411.4038

[163] H. Li, Z. Xu, G. Taylor, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Computing Research Repository (CoRR)*, vol. abs/1712.09913, 2017, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1712.09913

[164] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[165] Y. Chen, J. Wang, B. Zhu, M. Tang, and H. Lu, "Pixel-wise deep sequence learning for moving object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[166] W. Wang, J. Shen, and L. Shao, "Video salient object detection via fully convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 38–49, Jan 2018.

[167] Y. Wang, Z. Luo, and P.-M. Jodoin, "Interactive deep learning method for segmenting moving objects," *Pattern Recognition Letters*, vol. 96, pp. 66–75, 2017.

[168] L. Yang, J. Li, Y. Luo, Y. Zhao, H. Cheng, and J. Li, "Deep background modeling using fully convolutional network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 254–262, Jan 2018.

[169] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 724–732.

[170] D. Pathak, R. B. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move," *Computing Research Repository (CoRR)*, vol. abs/1612.06370, 2016, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1612.06370

[171] A. Faktor and M. Irani, "Video segmentation by non-local consensus voting." in *BMVC*, vol. 2, no. 7, 2014, p. 8.

[172] P. Bideau, R. R. Menon, and E. Learned-Miller, "MoA-Net: Self-supervised motion segmentation," in *European Conference on Computer Vision*. Springer, 2018, pp. 715–730.

[173] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 681–687.

[174] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer, 2003, pp. 363–370.

[175] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[176] ITU-R, "ITU-R Recommendation BT.601," International Telecommunication Union - Radiocommunication Sector (ITU-R), Geneva, CH, Standard, Mar. 2011.

[177] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computing Research Repository (CoRR)*, vol. abs/1412.6980, 2014, visited on 2019-04-15. [Online]. Available: http://arxiv.org/abs/1412.6980

[178] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested U-Net architecture for medical image segmentation," *CoRR*, vol. abs/1807.10165, 2018. [Online]. Available: http://arxiv.org/abs/1807.10165

[179] P. Iakubovskii, "Segmentation Models Pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2019.

[180] J. Cheng, Y. Tsai, S. Wang, and M. Yang, "SegFlow: Joint learning for video object segmentation and optical flow," *CoRR*, vol. abs/1709.06750, 2017. [Online]. Available: http://arxiv.org/abs/1709.06750

[181] K. Heidler, L. Mou, C. A. Baumhoer, A. J. Dietz, and X. X. Zhu, "HED-UNet: Combined segmentation and edge detection for monitoring the antarctic coastline," *CoRR*, vol. abs/2103.01849, 2021. [Online]. Available: https://arxiv.org/abs/2103.01849

[182] T. Sørensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons," *Biol. Skr.*, vol. 5, pp. 1–34, 1948.

[183] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[184] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *2010 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2010, pp. 2578–2585.

[185] M. Tzelepi and A. Tefas, "Semantic scene segmentation for robotics applications," in *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE, 2021, pp. 1–4.

[186] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[187] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[188] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[189] M. Dobeš, R. Andoga, and L. Főző, "Sensory integration in deep neural networks," *Acta Polytechnica Hungarica*, vol. 18, no. 3, pp. 245–254, 2021.

[190] F. R. Fathabadi, J. L. Grantner, I. Abdel-Qader, and S. A. Shebrain, "Box-trainer assessment system with real-time multi-class detection and tracking of laparoscopic instruments, using CNN," *Acta Polytechnica Hungarica*, vol. 19, no. 2, 2022.

[191] B. Tipary and G. Erdős, "Tolerance analysis for robotic pick-and-place operations," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5-6, pp. 1405–1426, Nov. 2021. [Online]. Available: https://link.springer.com/10.1007/s00170-021-07672-5

[192] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: a hands-on survey," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 12, pp. 2633–2651, 2015.

[193] J. Fraley, A. Imeri, I. Fidan, and M. Chandramouli, "A comparative study on affordable photogrammetry tools." in *ASEE Annual Conference proceedings*, 2018.

[194] M. E. Gurses, A. Gungor, S. Hanalioglu, C. K. Yaltirik, H. C. Postuk, M. Berker, and U. Türe, "Qlone®: a simple method to create 360-degree photogrammetry-based 3-dimensional model of cadaveric specimens," *Operative Neurosurgery*, vol. 21, no. 6, pp. E488–E493, 2021.

[195] Y. Wang, Y. Li, and J. Zheng, "A camera calibration technique based on openCV," in *The 3rd International Conference on Information Sciences and Interaction Sciences*. IEEE, 2010, pp. 403–406.

[196] F. E. Nowruzi, P. Kapoor, D. Kolhatkar, F. A. Hassanat, R. Laganiere, and J. Rebut, "How much real data do we actually need: Analyzing object detection performance using synthetic and real data," *arXiv preprint arXiv:1907.07061*, 2019.

[197] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[198] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[199] Q. Vuong, S. Vikram, H. Su, S. Gao, and H. I. Christensen, "How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies?" *CoRR*, vol. abs/1903.11774, 2019. [Online]. Available: http://arxiv.org/abs/1903.11774

[200] M. Benitez, "Quantifying generalization - CVEDIA detection technology vs. seven open source datasets," CVEDIA PTE Ltd, White Paper, 2020.

[201] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[202] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf

[203] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[204] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910, 2012.

[205] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous robots*, vol. 38, no. 1, pp. 65–88, 2015.

[206] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[207] C. Goldfeder and P. K. Allen, "Data-driven grasping," *Autonomous Robots*, vol. 31, no. 1, pp. 1–20, 2011.

[208] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 1710–1716.

[209] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1957–1964.

[210] M. Costanzo, G. De Maria, G. Lettera, and C. Natale, "Can robots refill a supermarket shelf?: Motion planning and grasp control," *IEEE Robotics & Automation Magazine*, vol. 28, no. 2, pp. 61–73, 2021.

[211] M. Costanzo, "Control of robotic object pivoting based on tactile sensing," *Mechatronics*, vol. 76, p. 102545, 2021.

[212] M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic, "Affordance detection for task-specific grasping using deep learning," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 91–98.

[213] E. Nikandrova and V. Kyrki, "Category-based task specific grasping," *Robotics and Autonomous Systems*, vol. 70, pp. 25–35, 2015.

[214] H. Dang and P. K. Allen, "Semantic grasping: planning task-specific stable robotic grasps," *Autonomous Robots*, vol. 37, no. 3, pp. 301–316, 2014.

[215] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[216] M. Görner, R. Haschke, H. Ritter, and J. Zhang, "MoveIt! Task Constructor for Task-Level Motion Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[217] F. Kruber, J. Wurst, and M. Botsch, "An unsupervised random forest clustering technique for automatic traffic scenario categorization," in *2018 21st International conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2811–2818.

PUBLICATIONS RELATED TO THE THESIS

[KA1] A. I. Károly, R. Fullér, and P. Galambos, "Unsupervised clustering for deep learning: A tutorial survey," *Acta Polytechnica Hungarica*, vol. 15, no. 8, pp. 29–53, 2018.

[KA2] A. I. Károly, J. Kuti, and P. Galambos, "Unsupervised real-time classification of cycle stages in collaborative robot applications," in *2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2018, pp. 000 097–000 102.

[KA3] A. I. Károly, M. Takács, and P. Galambos, "OCSVM-based evaluation method for generative neural networks," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.

[KA4] A. I. Károly, P. Galambos, J. Kuti, and I. J. Rudas, "Deep learning in robotics: Survey on model structures and training strategies," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 266–279, 2020.

[KA5] A. I. Károly, R. N. Elek, T. Haidegger, K. Széll, and P. Galambos, "Optical flow-based segmentation of moving objects for mobile robot navigation using pre-trained deep learning models," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 3080–3086.

[KA6] A. I. Károly, R. N. Elek, T. Haidegger, and P. Galambos, "Moving obstacle segmentation with an optical flow-based DNN: an implementation case study," in *2021 IEEE 25th International Conference on Intelligent Engineering Systems (INES)*. IEEE, 2021, pp. 000 189–000 194.

[KA7] A. I. Károly and P. Galambos, "Automated dataset generation with Blender for deep learning-based object segmentation," in *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2022, pp. 000 329–000 334.

[KA8] A. I. Károly, Á. Károly, and P. Galambos, "Automatic generation and annotation of object segmentation datasets using robotic arm," in *2022 IEEE 10th Jubilee International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)*. IEEE, 2022, pp. 000 063–000 068.

[KA9] A. I. Károly, S. Tirczka, T. Piricz, and P. Galambos, "Robotic manipulation of pathological slides powered by deep learning and classical image processing,"

in *2022 IEEE 22nd International Symposium on Computational Intelligence and Informatics and 8th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (CINTI-MACRo)*. IEEE, 2022, pp. 000 387–000 392.

[KA10] A. I. Károly and P. Galambos, "Task-specific grasp planning for robotic assembly by fine-tuning GQCNNs on automatically generated synthetic data," *Applied Sciences*, vol. 13, no. 1, p. 525, 2023.

[KA11] A. I. Károly, S. Tirczka, H. Gao, I. J. Rudas, and P. Galambos, "Increasing the Robustness of Deep Learning Models for Object Segmentation: A Framework for Blending Automatically Annotated Real and Synthetic Data," *IEEE Transactions on Cybernetics*, 2023.

OTHER PUBLICATIONS

[KAO1] R. N. Elek, A. I. Károly, T. Haidegger, and P. Galambos, "Towards optical flow ego-motion compensation for moving object segmentation." in *ROBOVIS*, 2020, pp. 114–120.

[KAO2] T. Murooka, A. I. Karoly, F. von Drigalski, and Y. Ijiri, "Simultaneous planning of grasp and motion using sample regions and gradient-based optimization," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1102–1109.

[KAO3] S. Tarsoly, A. I. Karoly, and P. Galambos, "Lessons learnt with traditional image processing techniques for mushroom detection," in *2022 IEEE 10th Jubilee International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)*. IEEE, 2022, pp. 000 225–000 232.