OpenDR

—

Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020
Duration: 48 months
Lead contractor: Aristotle University of Thessaloniki

Deliverable D7.4: Final public version of the OpenDR toolkit

Date of delivery: 29 Dec 2023

Contributing Partners: AUTH, TAU, AU, TUD, ALU-FR, CYB, PAL

Version: v4.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449.

Title	**D7.4: Final public version of the OpenDR toolkit**
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Open Research Data Pilot
Dissemination Level:	**PU**blic
Authors	Niclas Vödisch (ALU-FR), Ahmet Selim Çanakçı (ALU-FR), Abhinav Valada (ALU-FR), Bas van der Heijden (TUD), Jelle Luijkx (TUD), Jens Kober (TUD), Robert Babuska (TUD), Kakaletsis Efstratios (AUTH), Symeonidis Charalampos (AUTH), Tzelepi Maria (AUTH), Nousi Paraskevi (AUTH), Tosidis Pavlos-Apostolos (AUTH), Manousis Theodoros (AUTH), Nikolaidis Nikolaos (AUTH), Tefas Anastasios (AUTH), Babis Emmanouil (AUTH), Tsampazis Konstantinos (AUTH), Kirtas Emmanouil (AUTH), Avramelou Loukia (AUTH), Spanos Dimitrios (AUTH), Katsikas Dimitrios (AUTH), Moustakidis Vasileios (AUTH), Nikolaos Passalis (AUTH), Halil Ibrahim Ugurlu (AU), Amir Ramezani Dooraki (AU), Erdal Kayacan (AU), Alexandros Iosifidis (AU), Thomas Peyrucain (PAL), Olivier Michel (CYB), Roel Pieters (TAU), Moncef Gabbouj (TAU), Anton Muravev (TAU)
Lead Beneficiary	PAL (PAL Robotics)
WP	7
Doc ID:	OPENDR_D7.4.pdf

Document History

Version	**Date**	**Reason of change**
v1.0	18/09/2023	Deliverable structure template ready
v2.0	15/11/2023	Initial content from all partners
v3.0	20/11/2023	Final version for review
v4.0	29/12/2023	Final version to be submitted

Contents

Executive Summary

This document aims at supplementing the final version of OpenDR toolkit released in M48. It provides details about accessing, downloading and using the toolkit for all the provided installation methods, namely: by cloning the GitHub repository, by installing it using *pip* or through the provided *docker* images. Finally, it also details additional research and development tasks conducted that are related to the development of the toolkit.

1 Introduction

OpenDR aims at developing an open, non-proprietary modular toolkit that can be easily used by robotics companies and research institutions to efficiently develop, evaluate and deploy AI and cognition technologies to robotics applications. At a high level, OpenDR contains a selection of cognition and perception algorithms, along with general-purpose functionalities that are necessary for common robotics tasks. This technical report (Deliverable D7.4) aims at supplementing the final public version of OpenDR toolkit released on M48. It provides details about accessing, downloading, installing and using the toolkit. Furthermore, it also details additional research and development tasks conducted that are related to the development of the toolkit.

2 Changelog

Notable changes with the final version of the toolkit are summarized here.

Features:

- **#467 - Object Detection 2D Class Filtering:** The class filtering feature for 2D object detection has been incorporated to enhance selectivity in the output of deployed object detectors. This addition allows users to filter out undesired object classes and retain only those pertinent to their specific use case. The implementation comes in the form of a wrapper, requiring two key parameters: the object detector and a list of the user's classes of interest. For optimal functionality, it is imperative that the provided classes are a subset of the object categories recognizable by the specified detector. An exemplary demonstration utilizing the YOLOv5 object detector is provided, illustrating the filtering capability. By default, this demo highlights the 'person' class detections, effectively excluding all other categories.
- **#476 - YOLOv5s Inference Demo with Optimized Weights for Agricultural Use**: This feature in vehicle detection technology is an inference demo that leverages the optimized weights of a fine-tuned YOLOv5s model. This enhancement is specifically designed to elevate the model's performance in identifying tractors, which are crucial to agricultural operations. The fine-tuning process meticulously calibrates the model's weights, resulting in heightened sensitivity and precision in detecting agricultural trucks, with an emphasis on tractors. By applying these specialized weights during inference, the model exhibits superior detection capabilities, effectively discerning tractors from a variety of vehicles in diverse farm environments. Inference demo utilizes two real-world images of tractors as examples, in fields to illustrate the refined accuracy and efficiency of the model, affirming the potential of advanced object detection in supporting the agricultural sector's evolving needs.
- **#424 - Continual SLAM**: Adds a new visual SLAM tool that can continuously adapt to new domains.
- **#473 - RL-based Learner for Active Face Recognition**: Adds an Active Face Recognition agent. The implementation relies on stable_baselines3 and trains on Webots. A Webots world is provided and the corresponding environment file to train an agent in the task of Active Face Recognition, in which the goal is to maximize FR confidence while controlling a Mavic2 drone.

- **#479 - Adaptive HR Pose Estimation**: A new method is added for high-resolution active perception. The new method uses an adaptive ROI selection in High-Resolution images. This addition allows users to decide between the ”Primary” or ”Adaptive” ROI selection method for high-resolution pose estimation. The new ”Adaptive ROI selection” tool manages to further focus on the target subjects by eliminating the background information that passes through the model. Similar to the ”Primary” method, the ”Adaptive” methodology can be fine-tuned appropriately depending on the users' preferences for increasing the FPS or the precision by choosing the appropriate resizing variables. These resizing variables control the image size that passes through the pose estimation model on each step.
- **#423 - Fall and wave detection ROS nodes**: Fall detection ROS1/2 nodes to work both by subscribing to an image topic and running pose estimation internally (original mode) and by subscribing to a pose topic and only running fall detection (new mode).
- **#368 - Full integration of Voxel Pseudo Image Tracking (VPIT)**: VPIT is a 3D single object tracking method that works with Lidar point clouds, tracks objects at high inference speed and is optimized for embedded devices, such as Jetson TX2 and Xavier.
- **#451 - Robotti human detection simulation demo**: Add Webots simulation performing person detection on a field with the Robotti.
- **#402 - Binary High Resolution Learner**: This PR adds a binary high resolution learner to the toolkit.
- **#443 - Intent recognition tool:** Implements an intent recognition tool that can classify text data into the following intents: Complain, Praise, Apologise, Thank, Criticize, Agree, Taunt, Flaunt, Joke, Oppose, Comfort, Care, Inform, Advise, Arrange, Introduce, Leave, Prevent, Greet, Ask for help. This tool is meant to be used together with the speech transcription tool for speech-to-text conversion and hence the ROS nodes are expected to subscribe to output of speech transcription ROS node. Similarly, python demo is provided with integration of the speech transcription tool. The model implements an architecture that is trained in a multimodal manner (audio, visual, text) with the goal of improving inference on one of the modalities, in our case, text.
- **#436 - Add RGB gesture recognition**: Implements RGB-based hands gesture recognition (detection) based on hagrid dataset for 18 hand gestures. We currently provide one pretrained model. ROS1/ROS2 nodes are included, as well as a webcam demo. Algorithmically it relies on existing implementation of nanodet.
- **#442 - FSeq2-NMS**: Implementation of the FSeq2-NMS method (for person detection only) using SSD as detector.
- **#433 Speech Transcription with Whisper and Vosk**: Integrates two speech transcription libraries: Whisper by openAI and Vosk. Adds target classes for the output of each transcription library, integrates the download, load, infer and eval, etc., of the two libraries into OpenDR learner classes, add a demo for the speech transcription task, implements ROS node for the speech transcription task with two backbones, Whisper and Vosk, implements ROS2 node for the speech transcription task, and

add documents for ROS bridge, ROS messages, transcription target, and learner class.

- **#404 - ROS2 node for EfficientLPS**: Adds EfficientLPS ROS2 nodes.

Enhancements:

- **#456 - Test-tools improvement**: CI update to install the toolkit once and then use this single installation to run all the tool tests in parallel, instead of reinstalling the whole toolkit in each separate VM for every tool.
- **#419 - ROS nodes FPS performance measurements**: Adds time performance measurement of tools' inference. For each node, the time it takes to run (only) inference is measured and published in a performance topic. Publishing this message is optional, i.e. the relevant topic needs to be set via argparse. This message can be subscribed to, or echoed to show the current FPS.
- **#462 - Adding prompt when transcribe with Whisper**: Adds initial_prompt to WhisperLearner.infer. initial_prompt is a string that suggests the context of the transcription. For example names of people that will appear in the transcription. Furthermore, the ROS and ROS2 node, documents, and demo are updated accordingly.
- **#455 - Refactoring: pythonic joins in test_clang_format.py/test_cppcheck.py**: These slight code adjustments aim to enhance Pythonic quality by incorporating the str.join function.
- **#409 - High Resolution Pose Estimation webcam demo**: Provides a high resolution pose estimation webcam demo.
- **#408 - Object detection 2d camera demos**: Provides a object detection 2d camera demos.
- **#405 - Facial expression recognition demo update**: Updates the facial expression recognition demos.
- **#394 - Wave detection demo based on pose estimation**: Adds a wave detection demo to the existing list of lightweight open pose demos.
- **#488 - Real Time Object Detection in Agricultural Applications**: Provides additional pretrained weights for object detection in the RoboWeedMap dataset for nanodet-plus-fast. It also, enhances the inference capabilities with dynamic and non-dynamic inputs for ONNX inference, adds better inference optimizations and faster post processing. Provides a new Nanodet-based model named nanodet-plus-fast, designed for fast object detection in embedded devices, such as NVIDIA JETSON TX2. Finally, some new features were added in the training pipeline for finetuning and better experiment logging readability.

Bug Fixes:

- **#471 - Updated test_suite_develop.yml based on latest test_suite.yml**: test_suite_develop.yml should be identical with test_suite.yml to run scheduled tests on develop as stated here.
- **#466 - Minor fix on yolov5 webcam demo**: Getting the error `RuntimeError: cuDNN error: CUDNN_STATUS_NOT_INITIALIZED` was fixed by importing torch which seems to initialize it properly, similar to inference demo which works fine.

- **#463 - GPU installation fix**: This PR fixes GPU detectron2 and torch installation in install.sh and introduces a temporary fix for various conflicting dependency versions.
- **#465 - Fix ROS1 nodes argparse issue with .launch files**: This PR applies the fix suggested in issue #460 to all ROS1 nodes. The fix was tested and doesn't interfere with running the nodes normally.
- **#472 -Fix fmpgmapping**: Fixed a bug when publishing the full posterior map where the comparison for the map model was done with the wrong enumerations, leading to publishing the wrong initial alpha and beta parameters for uninformed priors.
- **#390 - Fix package creator and sources:** Improved automatic package creation.
- **#469 Apply cuDNN init fix to all Object Detectors 2D:** The fix applied to yolov5 webcam demo for the cudnn not initialized error (#466), needs to be applied to all object detectors.
- **#454 - Added unzip installation as base ubuntu dependency and tool tests fixes**: Replaced bcolz with bcolz-zipline to fix broken installation, in tests_suite.yml fixed a call for pip install requirements, more strict pillow version, changed ultralytics requirements, and fixed missing unzip dependencies.
- **#438 - Fix tests on master branch**: Errors with mobile_manipulation test were probably a consequence of issues with packages installation (solved in #431), re-enabling tests.
- **#430 Bump flask from 1.1.2 to 2.3.2**: Bump flask from 1.1.2 to 2.3.2, triggered by dependabot.
- **#426 EfficientLPS panoptic segmentation coloring bug**: EfficientLPS panoptic segmentation tool had a small bug in the panoptic/instance coloring functions.
- **#410 - Fix the dependency conflict of geffnet installation**: Fixes the dependency error between EfficientPS and EfficientLPS. EfficientLPS is unable to use efficient-Net (geffnet) library that can be installed using EfficientPS even though the versions are the same. The reason is that the original efficientNet library is modified for EfficientLPS. However, EfficientPS is still compatible with the modified version.
- **#421 - Fix link to nanodet documentation**: Nanodet MD file name was not correct in the reference manual index.
- **#420 - Fix bug in GEM ROS2 node**: Not possible to set qos_profile for message filter subscribers, therefore removing this argument.
- **#401 - Yolov5 training bugfix**: Introduces a quick fix to using custom trained models for YOLOv5 as well as an example of converting an OpenDR dataset into YOLOv5 .yml format for training.
- **#397 - Fall Detection - alternative infer input**: Fixes #282, by changing the infer method of the fall detection learner. The infer method now accepts pose lists as input in addition to OpenDR or OpenCV images.
- **#392 - Lightweight OpenPose tool fixes and improvements**: Various minor fixes and improvements for the lightweight open pose tool, learner, demos and docs. Most are based on the review of #356, where some issues of the new tool were inherited from this one.

- **#478 Synchronization and bugfixes:** Bumps av versions and fixes download function in High Resolution Pose Estimation.
- **#459 Active face recognition demo and bug fixes on Face Recognition**: Changed Face Recognition inference return. Now class is 0/1 if not found/found. Added feature extraction method to get access to features from a face image.

3 Accessing the OpenDR toolkit

The toolkit is developed using the well-established GitHub platform, following robust development methodologies, including continuous integration and strict code review guidelines, as described in D2.2, D7.2 and D7.3. The most recent version of the toolkit can be accessed at:

`https://github.com/opendr-eu/opendr`

The `master` branch contains the latest stable version of the toolkit and the `develop` branch a version that includes the latest additions, refactors and module upgrades. Although CI tests will maintain stability and high quality in both branches, the `develop` one is likely to change often so it is less adapted for daily usage or production.

OpenDR provides an intuitive and easy-to-use **Python interface**, a **C API** for selected tools, a **ready-to-use ROS/ROS2 nodes** and **a wealth of usage examples and supporting tools**. OpenDR is built to support Webots Open Source Robot Simulator, while extensively following industry standards, such as ONNX model format and OpenAI Gym Interface. Detailed installation instructions, documentation can be found in the OpenDR repository and wiki. For completeness, we also provide the installation and usage instructions in Section 5.

4 Toolkit Research and Development

4.1 Using Part-based Representations for Explainable Deep Reinforcement Learning

4.1.1 Introduction and objectives

Deep Reinforcement Learning (RL) has achieved state-of-the-art performance in various applications. However, the use of RL agents in critical environments, where safety is highly prioritized, is hindered due to the limited transparency of the models. Extracting the rationale of a deep learning (DL) model in a human-interpretable way remains a challenging task, but doing so would be highly useful for improving both the performance and trustworthiness of the model, as well as preventing failures. To this end, post-hoc explanation methods have been extensively studied over the years, providing rationales for the predictions of the model. However, such approaches cannot always provide a reliable explanation, with pre-hoc methods for explainable AI gaining increasing attention recently.

Utilizing deep learning models to learn part-based representations holds significant potential for interpretable-by-design approaches, as these models incorporate latent causes obtained from feature representations through simple addition. However, training a part-based learning model presents challenges, particularly in enforcing non-negative constraints on the model's parameters, which can result in training difficulties such as instability and convergence issues. Moreover, applying such approaches in Deep Reinforcement Learning (RL) is even more demanding due to the inherent instabilities that impact many optimization methods. To this end, OpenDR examined the possibility of employing a non-negative training approach for actor models in RL, enabling the extraction of part-based representations that enhance interpretability while adhering to non-negative constraints. We employed a non-negative initialization technique, as well as a modified sign-preserving training method, which can ensure better gradient flow compared to existing approaches. We demonstrate the effectiveness of the proposed approach using a simple benchmark, i.e., Cartpole benchmark, implemented in the Webots simulation.

A paper describing the developed methodology was accepted and presented in ECML PKDD Workshop Uncertainty meets Explainability:

- M. Kirtas, K. Tsampazis, L. Avramelou, N. Passalis, and A. Tefas, "Using Part-based Representations for Explainable Deep Reinforcement Learning", ECML PKDD - Workshop: Uncertainty meets Explainability, 2023

The corresponding publications can be found in Appendix 7.1.

4.2 Few-Shot Panoptic Segmentation With Foundation Models

4.2.1 Introduction and objectives

In this work, we present a method for *Segmenting Panoptic Information with Nearly 0 labels* (SPINO). We first leverage a frozen DINOv2 [12] backbone to extract visual features. Subsequently, we train two task-specific heads for semantic segmentation and boundary estimation with as few as ten annotated images to perform few-shot panoptic segmentation. To enable real-time inference and further enhance prediction quality, we generate panoptic pseudo-labels in an offline manner for a larger bag of raw images, suitable for training any existing panoptic segmentation model [3, 11]. We conduct extensive evaluations on several public [4, 10] and in-house datasets, demonstrating that our SPINO approach yields highly competitive results when compared to fully supervised learning models. The main contributions are summarized as follows: 1) proposing the first method for few-shot panoptic segmentation based on unsupervised foundation models, 2) introducing a novel pseudo-label generation scheme trainable with as few as ten annotated images, 3) demonstrating the competitiveness of SPINO against supervised training with ground truth labels, 4) exploring the impact of various architectural design choices through extensive evaluations, and 5) providing open access to the code and trained models at `http://spino.cs.uni-freiburg.de`. A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.2. At the time of submission of this deliverable, the work is still under review.

- [9] M. Käppeler, K. Petek, N. Vödisch, W. Burgard, and A. Valada, "Few-Shot Panoptic Segmentation With Foundation Models", *arXiv preprint arXiv:2309.10726*, 2023.

4.2.2 Description of the work performed

In this section, we introduce our proposed method, SPINO, for few-shot panoptic segmentation. Illustrated in Figure 2, our approach leverages the capabilities of the recent foundation

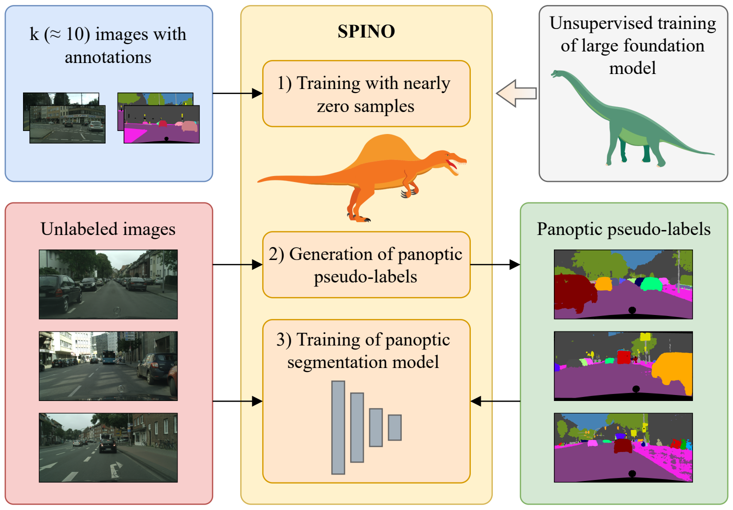
k (≈ 10) images with annotations
SPINO
Unsupervised training of large foundation model
1) Training with nearly zero samples
Unlabeled images
2) Generation of panoptic pseudo-labels
Panoptic pseudo-labels
3) Training of panoptic segmentation model

Figure 1: SPINO enables few-shot panoptic segmentation by exploiting descriptive image features from unsupervised task-agnostic pretraining. We generate panoptic pseudo-labels by learning from only $k \approx 10$ annotated images in an offline manner. We can then leverage these pseudo-labels to train any panoptic segmentation model enabling online deployment.

model DINOv2 [12] to extract rich image features essential for both semantic segmentation and boundary estimation. A distinctive aspect of SPINO is the novel pseudo-label generation scheme, designed to disentangle semantic regions of "thing" classes into individual instances by predicting object boundaries. This innovative strategy allows SPINO to bootstrap from a remarkably small number of ground truth annotations, e.g., 10, enabling the generation of high-quality panoptic pseudo-labels. To enhance real-time inference and further refine the quality of panoptic predictions, we train a panoptic segmentation model with the generated pseudo-labels.

The panoptic segmentation scheme comprises three main components, as depicted in Figure 2. These components include learnable modules for semantic segmentation and boundary estimation, along with a static component responsible for fusing their predictions. The semantic segmentation module is composed of a frozen DINOv2 backend, a bilinear 14x-upsampling layer, and a final n-class MLP with 4 layers, where n represents the number of semantic classes. The boundary estimation module follows a similar design, employing 4x-upsampling and binary classification with $n = 2$.

A key highlight of SPINO lies in its ability to train the pseudo-label generator with an exceptionally low number of ground truth annotations, a critical advantage for scenarios with limited labeled data. Even with the unsupervised training procedure of DINOv2, SPINO achieves robust results. The training process of our pseudo-label generator involves employing various data augmentation techniques on the input RGB image, such as random cropping, horizontal flipping, and color jitter. Subsequently, the augmented image is fed into the two task-specific heads, and the respective loss functions are computed.

In summary, SPINO introduces an innovative solution for few-shot panoptic segmentation, showcasing its ability to achieve high-quality results with minimal ground truth annotations. The combination of DINOv2 as a foundation model and the proposed pseudo-label generation scheme demonstrates the potential for robust panoptic segmentation in resource-constrained scenarios. We provide qualitative visualizations of our pseudo-labels in Figure 3 for both public datasets as well as our in-house data including outdoor urban and indoor office environments.

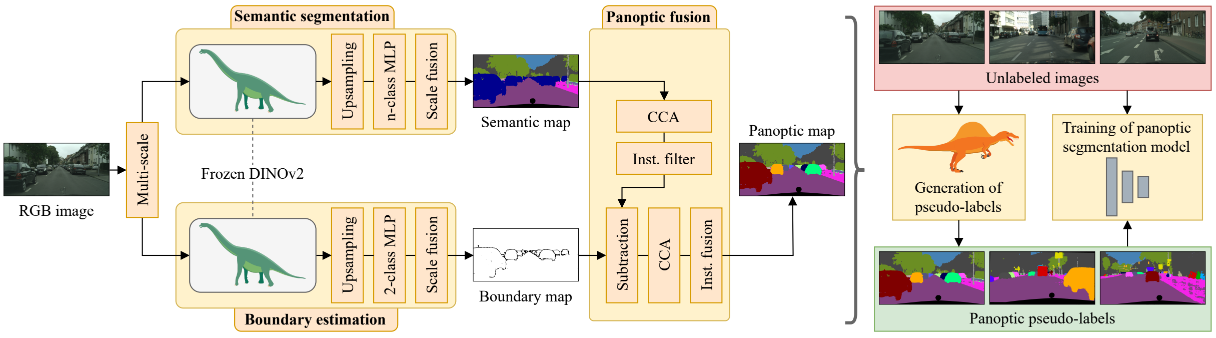

Figure 2: Overview of our proposed SPINO approach for few-shot panoptic segmentation. SPINO consists of two learning-based modules for semantic segmentation and boundary estimation that leverage features from the recent foundation model DINOv2 [12]. A panoptic fusion scheme combines their outputs using connected component analysis (CCA) and multiple small instance filtering steps. SPINO creates pseudo-labels for a large number of unlabeled images using only $k \approx 10$ images with ground truth annotations. These pseudo-labels can then be utilized to train any panoptic segmentation model.

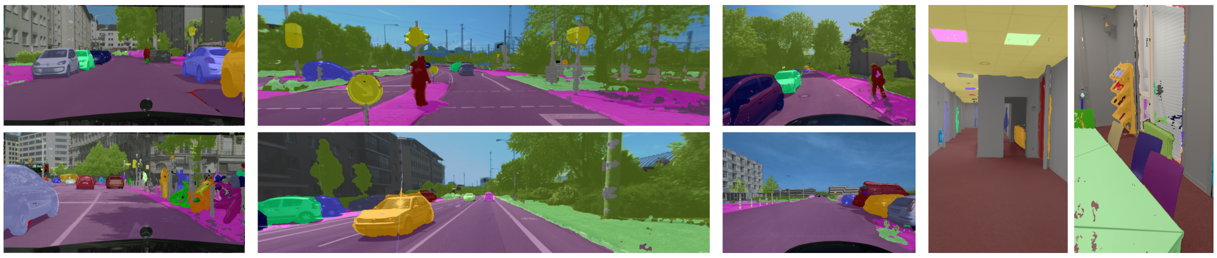

Figure 3: Qualitative performance of our pseudo-label generator in four diverse domains from both public and in-house data sources. From left to right, we show Cityscapes [4], KITTI-360 [10], in-house automated driving, and an in-house office environment.

4.2.3 Conclusion

In this work, we introduced SPINO for few-shot panoptic segmentation by exploiting descriptive image representations from the unsupervised foundation model DINOv2. We demonstrated that SPINO can generate high-qualitative pseudo-labels after being trained on as little as ten annotated images. These pseudo-labels can then be used to train any existing panoptic segmentation method yielding results that are highly competitive to fully supervised learning approaches relying on human annotations. Finally, we extensively evaluated several design choices for the proposed pseudo-label generator. To facilitate further research, we made our code publicly available. In future research, we will refine the boundary estimation and employ SPINO in additional domains.

4.3 Collaborative Dynamic 3D Scene Graphs for Automated Driving

4.3.1 Introduction and objectives

Spatial and semantic understanding is of paramount importance for ensuring the safe and autonomous navigation of mobile robots and self-driving cars. In the realm of automated driving (AD), recent advancements leverage high definition (HD) map information as potent priors for various downstream tasks, including perception [15], localization [2], planning [5], and control [13]. However, the conventional approach to constructing HD maps, typically in a top-down manner via traffic authorities or labor-intensive labeling efforts, contrasts with the accuracy achieved by automatic bottom-up AD mapping approaches. These bottom-up methods, while excelling in tasks like occupancy or semantic mapping, cannot fulfill the requirements outlined for AD map representations [14], such as completeness, accuracy, scalability, frequent updates, topological information grounded in rich sensor data, and efficient querying. In response to these challenges, we propose *Collaborative URBan Scene Graphs* (CURB-SG) as a solution, constructing a hierarchical graph structure of the environment, as depicted in Figure 4. CURB-SG addresses the limitations of conventional SLAM maps by facilitating vision and language queries and efficiently covering large areas by exploiting multiple agents.

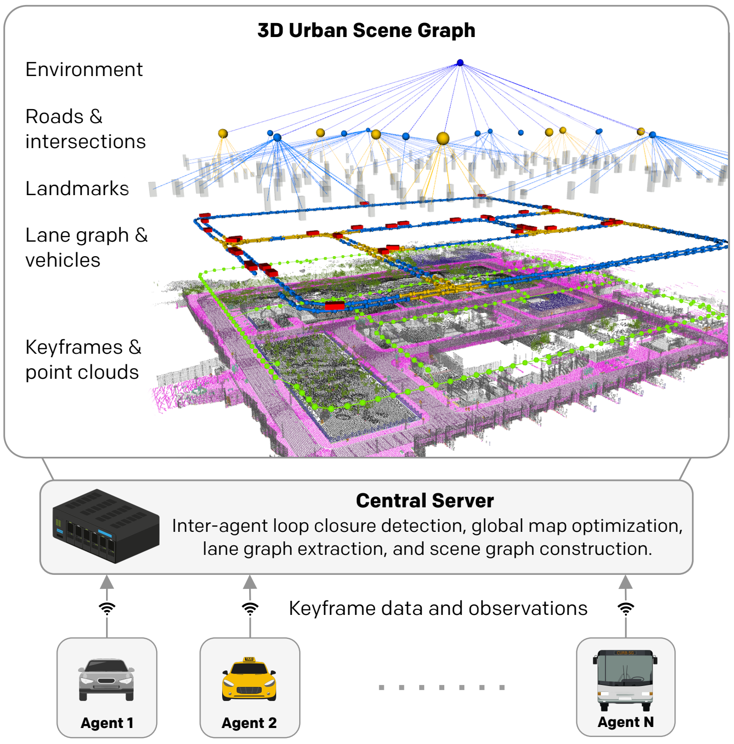
3D Urban Scene Graph
Environment
Roads & intersections
Landmarks
Lane graph & vehicles
Keyframes & point clouds
Central Server
Inter-agent loop closure detection, global map optimization, lane graph extraction, and scene graph construction.
Keyframe data and observations
Agent 1
Agent 2
Agent N

Figure 4: For our proposed collaborative urban scene graphs (CURB-SG), multiple agents send keyframe packages with their local odometry estimates and panoptic LiDAR scans to a central server that performs global graph optimization. We subsequently partition the environment based on a lane graph from agent paths and other detected cars. Together with the 3D map, the lane graph forms the base of the large-scale hierarchical scene graph.

The novelty of CURB-SG lies in its dynamic 3D scene graph representation of urban driving environments, effectively covering large areas through multi-agent observations. To achieve this, we introduce an online lane graph for partitioning urban environments, serving as a common link among multiple scene graph layers. This approach is inspired by an analogy to indoor environments, where cities (buildings) are divided into intersections and roads (rooms), containing static landmarks such as traffic signs (furniture), and dynamic objects such as vehicles (humans). Furthermore, our method leverages a centralized collaborative SLAM approach, combining panoptic LiDAR data and local odometry estimates into a single 3D map while optimizing a global pose graph through inter-agent loop closures.

Our contributions include the introduction of a novel algorithm for representing urban driving environments as dynamic 3D scene graphs, efficient partitioning of urban environments using lane graphs constructed on-the-fly from panoptic LiDAR observations, an efficient collaborative graph SLAM method, extensive evaluations using the CARLA simulator [6], and the public availability of code and sample data at `http://curb.cs.uni-freiburg.de`. Through CURB-SG, we aim to pave the way for advanced AD mapping approaches that fulfill the evolving demands of autonomous systems.

A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.3. At the time of submission of this deliverable, this work is still under review.

- [7] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada, "Collaborative Dynamic 3D Scene Graphs for Automated Driving", *arXiv preprint arXiv:2309.06635*, 2023.

4.3.2 Description of the work performed

In this section, we present Collaborative URBan Scene Graphs (CURB-SG), an innovative approach designed to address the challenges of constructing efficient hierarchical graph structures for the representation of large-scale outdoor scenes in the context of automated driving (AD). Figure 5 illustrates the various components of CURB-SG, highlighting its collaborative LiDAR SLAM backend and the dynamic scene graph generation mechanism. CURB-SG leverages the collaborative efforts of multiple agents, each equipped with LiDAR sensors, to provide a robust and globally consistent 3D map of the environment.

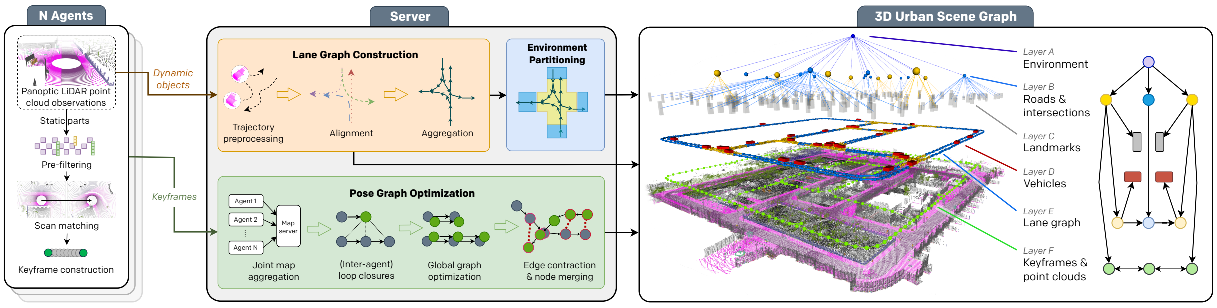

Figure 5: Overview of CURB-SG: Multiple agents obtain panoptically segmented LiDAR data and provide an odometry estimate based on the static parts of the point cloud. A centralized server instance then performs (PGO) including inter-agent loop closure detection and edge contraction based on the agents' inputs. Tightly coupled to the pose graph, we aggregate a lane graph from panoptic observations of other vehicles as well as the agent's trajectories. Next, the lane graph is partitioned to retrieve a topological separation that allows for the hierarchical abstraction of larger environments.

The collaborative LiDAR SLAM backend is a cornerstone of CURB-SG, building upon the well-established HDL Graph SLAM [8] and extending it to a multi-agent scenario. Agents capture sparse 3D point clouds containing panoptic segmentation labels, where keyframes are generated based on LiDAR odometry and transmitted to a central server. These keyframes carry estimated poses and static LiDAR point clouds, forming the basis for collaborative SLAM. The server performs centralized SLAM by detecting intra- and inter-agent loop closures, optimizing the global pose graph through pose graph optimization, and addressing long-term mapping challenges through an innovative edge contraction technique. This ensures continuous refinement of the 3D map, distinguishing between static and dynamic elements, and enhancing the robustness of the representation.

The scene graph generation component in CURB-SG introduces a novel hierarchical representation, going beyond traditional indoor scene graph constructions. The hierarchical structure decomposes a constructed lane graph into intersecting and non-intersecting road areas, facilitating both spatial and semantic abstraction. This hierarchical environment representation consists of layers, including global-level information, intersections and roads, static landmarks, dynamic vehicles, the lane graph, and the semantic map. The dynamic lane graph generation is a pivotal aspect, incorporating trajectories of ego agents and observed vehicles, providing a robust spatial partitioning that efficiently serves downstream tasks such as trajectory prediction. The spatial partitioning is based on the obtained lane graph, allowing for an intuitive division of the city into intersecting and non-intersecting road areas, addressing the unique challenges posed by outdoor driving scenarios.

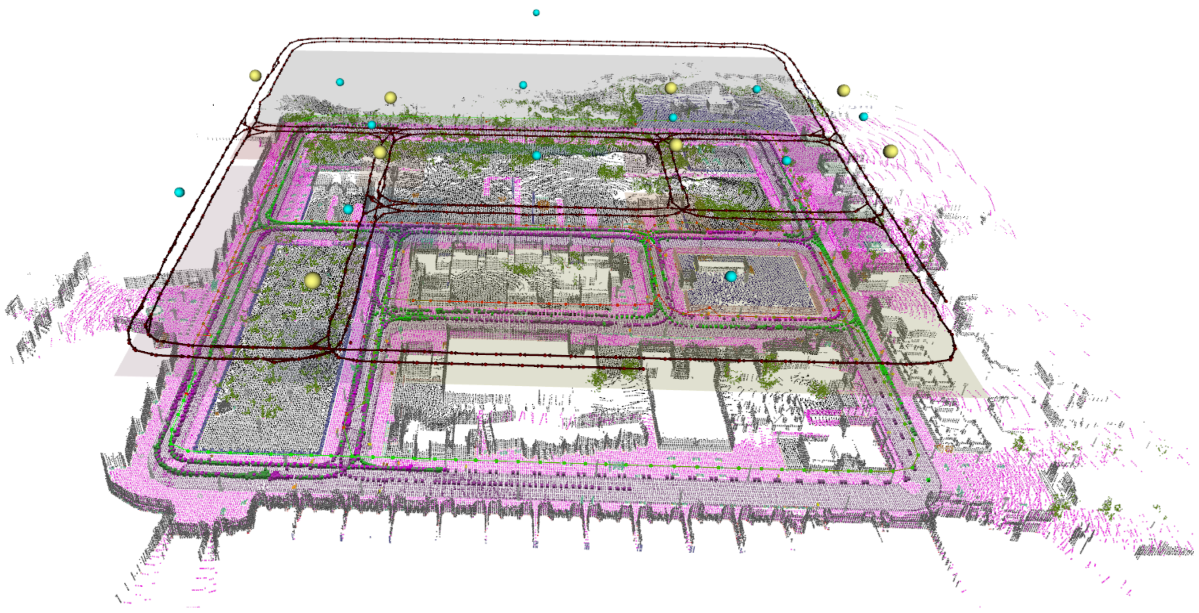

Figure 6: Qualitative visualization of the intersection detection quality. Yellow nodes represent intersections, whereas blue nodes represent non-intersecting road areas. As shown, these areas can be predicted with high precision even without a complete lane graph.

Dynamic lane graph generation is a key aspect of the proposed scene graph generation, contributing to the comprehensive understanding of the spatial layout and semantic content of urban scenes. The integration of static landmarks, dynamic vehicles, and the semantic map within this hierarchical framework provides a holistic view of the environment, facilitating efficient querying for various AD tasks. The spatial partitioning based on the lane graph allows for an intuitive division of the city into intersecting and non-intersecting road areas, addressing the unique challenges posed by outdoor driving scenarios. Our adaptive approach to spatial partitioning and abstraction aligns with the complex and dynamic nature of urban environments, making it a valuable contribution to the field of autonomous systems.

In conclusion, CURB-SG offers a comprehensive solution to the representation and understanding of large-scale outdoor scenes in the context of automated driving. By combining collaborative LiDAR SLAM with dynamic lane graph generation and a hierarchical scene graph structure, CURB-SG addresses the challenges of long-term and large-scale mapping, contributing to the advancement of autonomous systems. One can see the qualitative results in Figure 6.

4.3.3 Conclusion

In this work, we introduced CURB-SG as a novel approach to building large-scale hierarchical dynamic 3D urban scene graphs from multi-agent observations. We furthermore demonstrated how our collaborative SLAM approach facilitates frequent map updates and rapid exploration while scaling to large environments. To foster further research in this direction, we made our code publicly available. In future work, we will address the reliance on simulated panoptic labels and known initial poses of the agents. Orthogonal to that, follow-up work could address a decentralized variant that operates under real-time constraints. Furthermore, we plan to include pedestrian information as well as additional topological elements such as road boundaries.

4.4 Learning to estimate incipient slip with tactile sensing to gently grasp objects

4.4.1 Introduction and objectives

To gently grasp objects, robots need to generate enough friction without creating damage by applying the right amount of force. In practice, implementing this force regulation is challenging since it requires knowledge of the friction coefficient, which can vary from object to object and even from grasp to grasp. Fortunately, tactile sensing can provide information about friction notably by detecting the moment when the object slips away from the grasp. These tactile sensors capture distributed information about the deformation of the artificial skin in the normal and tangential direction, from which slippage can be detected. However, current approaches only react to slip, which leads to significant object movement. The movement can in turn induce a failure of the grasp and damage.

Our goal in this study is to create a tactile-enabled gripper that maintains a squeezing force on an arbitrary object so that the safety margin remains constant (Fig. 7A). To do so, we designed an impedance control gripper (Fig. 7B) which regulates its grasping force in real-time. The gripper has two soft tactile sensing fingertips able to capture the 3D deformation of a membrane using an embedded camera (Fig. 7C). The images of the interaction are fed to a CNN to estimate the frictional safety margin Γ (Fig. 7D,F). Γ is then used to adjust the grasping force in real-time (Fig. 7E), improving object manipulation and minimizing object slip (Fig. 7G).

A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.4. At the time of submission of this deliverable, this work is still under review.

- [1] D-J. Boonstra, L. Willemet, J. Luijkx, M. Wiertlewski, “Learning to estimate incipient slip with tactile sensing to gently grasp objects”, *under review*, 2023.

4.4.2 Description of the work performed

For this project, we developed and tested a tactile sensing gripper, named FUSE, which integrates Chromatouch tactile sensors and a custom robotic gripper. The Chromatouch mechanism employs a color-mixing principle, utilizing 3D-printed layers with colored markers and a transparent silicone cast. Tactile images are captured at 100 Hz using a per-finger embedded USB

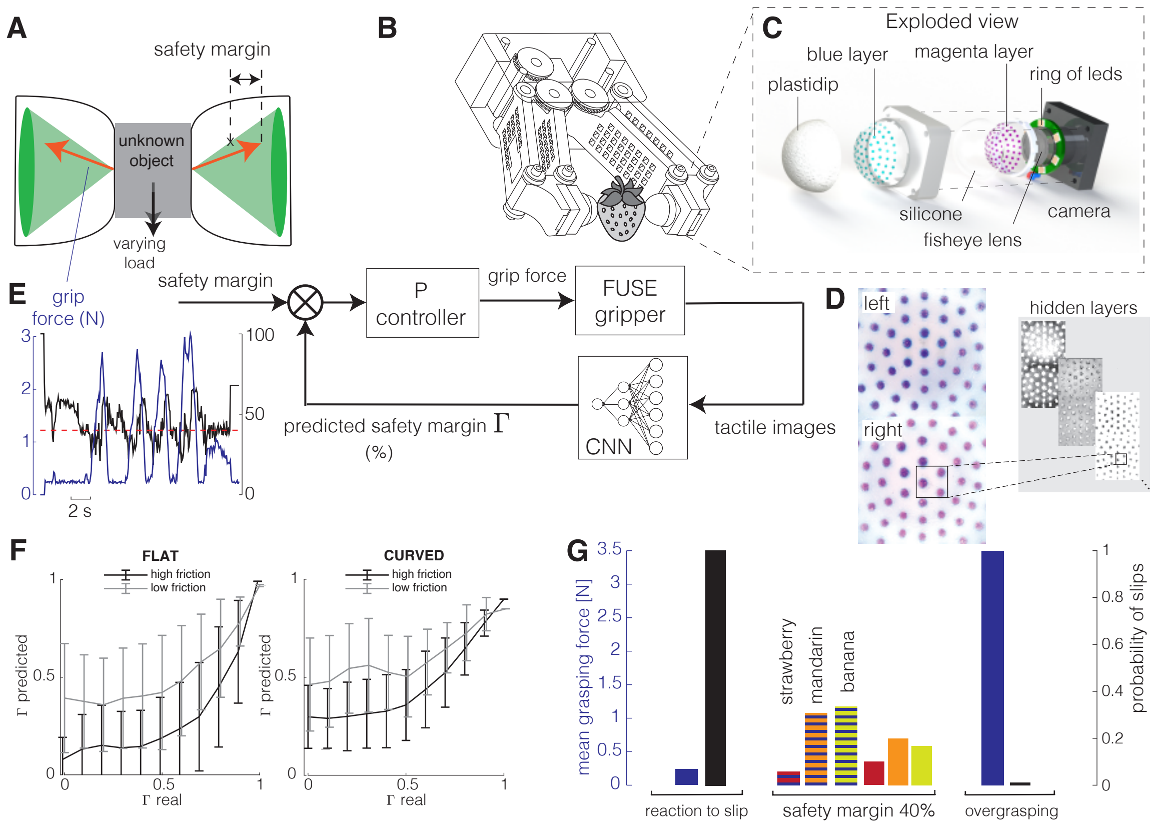

Figure 7: **A**. Typical evolution of the interaction force when manipulating an object. The grip force is maintained with a safety margin Γ over the minimum required grip force defined by the friction cone. **B**. Render of the custom-made parallel FUSE gripper. **C**. Exploded view of the tactile sensor ChromaTouch. **D**. Hidden layers of the convolutional neural network are used to predict the safety margin. **E**. Grip force control to maintain a constant safety margin. **F**. Deviation of the prediction compared to the real safety margin. **G**. Mean grasping force and probability of slips for three control strategies (reaction to slip, constant safety margin of 40%, and overgrasping strategy with a fixed 3.5 N grasping force).

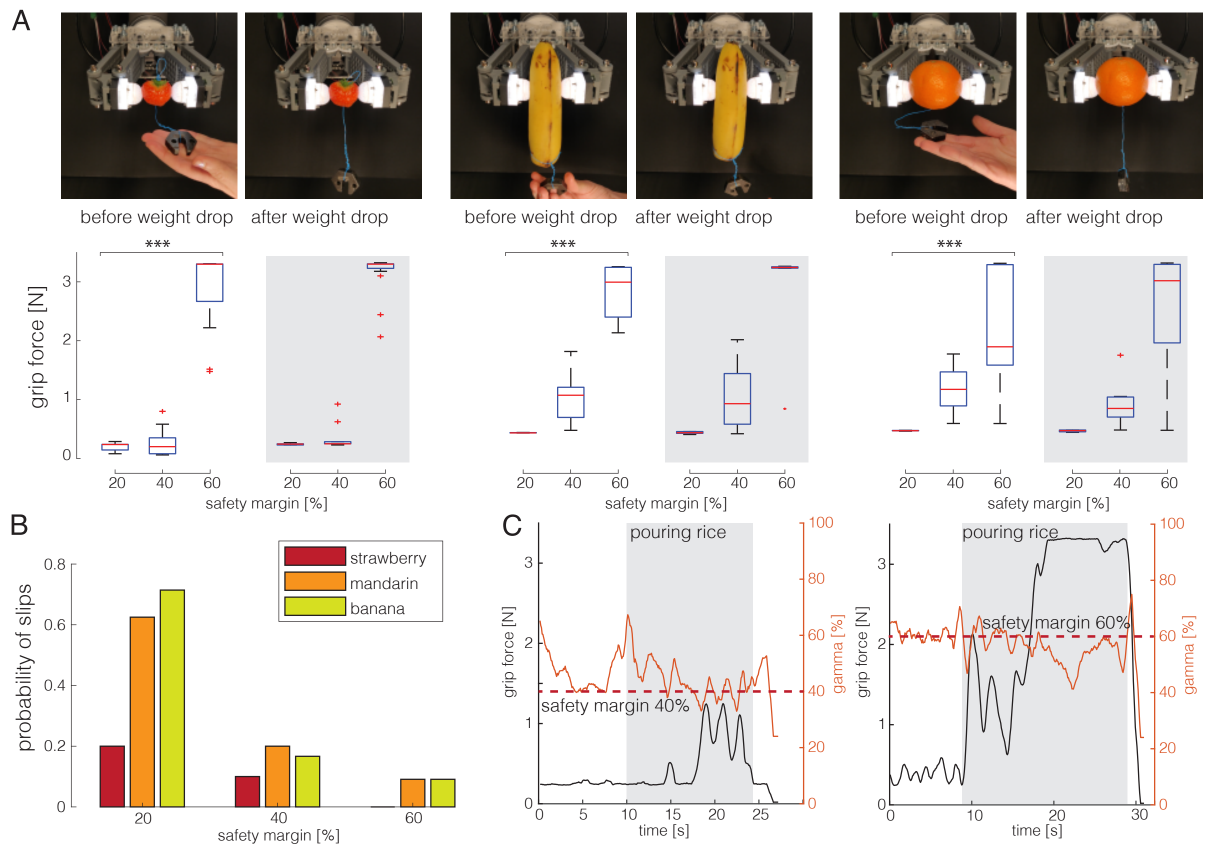

Figure 8: **A**. Mean grasping force for 3 different fruits and 3 safety margin commands. The gray zone represents the period after the weight is dropped. **B**. Probabilities of slip as a function of the safety margin for the 3 fruits. **C**. Grip force and predicted safety margin when the FUSE gripper is grasping a cup being filled with rice. The safety margin was controlled at 40% and 60% respectively on the left and on the right.

camera. The gripper design is open-source and available on GitHub. In our experimental setup, the gripper interacts with objects to collect data for training and evaluation. A force sensor embedded in the object, compensated by DC-motors, measures 3D force interactions at the fingertips. Safety margin Γ is estimated during trials involving a predefined grasping force and a subsequent ramp-pulling force. The training of our model involves linking tactile images and safety margin Γ using ShuffleNetV2. Tactile images are resized and concatenated before being fed into the network. ColorJitter and GaussianBlur augmentations enhance the model's generalization. The trained CNN outputs a single floating-point value representing predicted Γ. To control the gripper's grip force, a simple P controller utilizes the CNN output, adjusting the force to maintain a target Γ. The minimum and maximum applied grip forces are set to 0.25 N and 3.2 N, respectively, to avoid damage to the tactile sensor. The working principle of the tactile sensor involves calibrating acquired images towards actual 3D deformation using a filtered version of the Hertz contact model. This enables precise measurements of sub-millimeter displacements in both normal and tangential planes. In terms of results, the accuracy of the trained CNN is demonstrated by comparing predicted Γ with ground truth from force measurements. The Mean Squared Error (MSE) loss over validation datasets indicates a combined average of 0.01821, reflecting a high accuracy of 98.2The validation of the gripper's controller involves real-time grasping tasks. The CNN model, combined with a Proportional controller, successfully adjusts grip force based on safety margin targets. Experiments on delicate fruits show increased grip force with higher safety margin targets, leading to a decrease in slip probability and effective real-time control (Fig. 8).

4.4.3 Conclusion

We proposed a novel method for gripper force control based on the frictional safety margin (Γ), extracted from tactile sensor images using a convolutional neural network. The predicted Γ achieved an average accuracy of 98.2% compared to force measurements, showcasing robustness across different friction conditions. However, limitations arise in predicting Γ on surfaces with minimal friction or high slipperiness. To address this, experiments were confined to a controlled force range (0.25 to 3.2 N) to ensure accuracy and prevent sensor damage.

The study acknowledges challenges in accurately estimating safety margins on slippery surfaces due to sparse training data in lower safety margin ranges. This limitation could pose challenges in handling fragile and slippery objects, as underestimating safety margins may lead to excessive grasping force. Despite this, optimal safety margins around 40% are recommended, and the control approach's flexibility allows for margin adjustments.

Uncertainties in prediction are highlighted by a large standard deviation of estimated safety margins, potentially causing grasp force fluctuations, particularly under external perturbations. Real-life experiments with fragile objects demonstrate the model's generalization to complex scenarios, indicating enhanced grasping performance and reduced object damage. The trained network's prediction speed at 50 Hz on a desktop CPU enables real-time control, with the gripper's reaction time after a weight drop measured at approximately 100 ms, comparable to human reaction times.

5 Installing and using the OpenDR toolkit

To maximize the visibility and ease-of-use of the toolkit, we provide three different ways for installing the toolkit:

1. By cloning the GitHub repository
2. Using *pip*
3. Using *docker*

The first way provides a fully functional version of the toolkit that can be installed in various platforms. *pip* is a straightforward way to install and experiment with the Python API of the toolkit, while docker images are provided to experiment with toolkit functionalities in a preconfigured environment with very little effort, as well as for other containerized applications.

The following subsection provides an overview of the installation process. OpenDR is designed to be easy-to-use and install in order to maximize its impact. To this end, installation scripts have been prepared to ensure that this process will be very easy, even for novice users. Up-to-date instructions and additional details are available on OpenDR's GitHub repository.

5.1 Installation by cloning the GitHub repository

5.1.1 Installation procedure

To install the toolkit on a Linux system, please first make sure that *git* is available on the system:

```
sudo apt install git
```

Then, the toolkit should be downloaded locally:

```
git clone --depth 1 --recurse-submodules -j8  \
    https://github.com/opendr-eu/opendr
```

To install the toolkit an installation script is available:

```
cd opendr
./bin/install.sh
```

The installation script automatically installs all the required dependencies. Note that we can set the training/inference device using the `OPENDR_DEVICE` variable. The toolkit defaults to using CPU. If we want to use GPU, we can set this variable accordingly *before* running the installation script:

```
export OPENDR_DEVICE=gpu
```

The installation script creates a *virtualenv*, where the toolkit is installed. OpenDR environment can be activated similar to any other *virtualenv*:

```
source ./bin/activate.sh
```

All functionality (e.g., ROS, tools, demos, etc.) are then readily available.

5.2 Installation using *pip*

To increase the visibility of the toolkit, PyPI packages have been prepared for each tool.

5.2.1 Installation procedure

When installing the Python-API of the toolkit, it is necessary to first install the required dependencies:

```
sudo apt install python3.8-venv libfreetype6-dev git build-essential cmake \
   python3-dev wget libopenblas-dev libsndfile1 libboost-dev libeigen3-dev
python3 -m venv venv
source venv/bin/activate
pip install wheel
```

Then you can install the toolkit with:

```
pip install opendr-toolkit-engine
pip install opendr-toolkit
```

If your CPU does not support AVX2, you will need to set `export DISABLE_BCOLZ_AVX2=true` prior to installing the toolkit.

5.2.2 Installing only a particular tool using pip

The instructions above will install the entire toolkit. It is however possible to only install a specific OpenDR tool as the package has been split to that effect. If you wish to only perform pose estimation you can:

```
pip install opendr-toolkit-engine
pip install opendr-toolkit-pose-estimation
```

Note that `opendr-toolkit-engine` must always be installed in the system. The following packages are distributed:

```
opendr-toolkit-activity-recognition
opendr-toolkit-speech-recognition
opendr-toolkit-semantic-segmentation
opendr-toolkit-skeleton-based-action-recognition
opendr-toolkit-face-recognition
opendr-toolkit-facial-expression-recognition
opendr-toolkit-panoptic-segmentation
opendr-toolkit-pose-estimation
opendr-toolkit-fall-detection
opendr-toolkit-compressive-learning
opendr-toolkit-hyperparameter-tuner
opendr-toolkit-heart-anomaly-detection
opendr-toolkit-human-model-generation
opendr-toolkit-multimodal-human-centric
opendr-toolkit-object-detection-2d
opendr-toolkit-object-tracking-2d
opendr-toolkit-object-detection-3d
opendr-toolkit-object-tracking-3d
opendr-toolkit-ambiguity-measure
```

5.3 Installation using *docker*

5.3.1 Procedure

Appropriate dockerfiles that can run on any Linux system have been prepared and docker images are publicly available on dockerhub. First, docker needs to be installed in your system. For Ubuntu you can follow this procedure. When installed, running the OpenDR docker image is very easy. For example, for the CPU image all you need is to execute:

```
sudo docker run -p 8888:8888 opendr/opendr-toolkit:cpu_v2.0.0
```

or for the cuda-enabled one:

```
sudo docker run --gpus all -p 8888:8888 opendr/opendr-toolkit:cuda_v2.0.0
```

Both commands will pull the image and launch it, and a *Jupyter* notebook server is started that listens on port 8888. This can be accessed by clicking on the link similar to `http://127.0.0.1:8888/?token=TOKEN` that appears in the console. Alternatively you can run an interactive session with:

```
sudo docker run -it opendr/opendr-toolkit:cpu_v2.0.0 /bin/bash
```

or

```
sudo docker run --gpus all -it opendr/opendr-toolkit:cuda_v2.0.0 /bin/bash
```

respectively for a cpu or cuda session. However, if you start an interactive session do not forget to enable the `venv` with the command:

```
source bin/activate.sh
```

If you want to display GTK-based applications from the Docker container (e.g., visualize results using OpenCV `imshow()`), then you should mount the X server socket inside the container:

```
xhost +local:root
sudo docker run -it -v /tmp/.X11-unix:/tmp/.X11-unix \
   -e DISPLAY=unix$DISPLAY opendr/opendr-toolkit:cpu_v2.0.0 /bin/bash
```

5.4 ROS

With the third version of the toolkit, ROS and ROS2 nodes to interact with the toolkit have been provided for all included tools.

5.4.1 Environment setup for ROS

The instructions provided here will assume ROS `noetic` is already installed in your system, that a webcam is available and that you already installed the OpenDR toolkit (see section 5.1.1).

1. Move to the workspace: `cd projects/opendr_ws`
2. Install the package for the webcam: `sudo apt install ros-noetic-usb-cam`
3. Source the ROS installation: `source /opt/ros/noetic/setup.bash`
4. Build the workspace: `catkin_make`
5. Source it: `source devel/setup.bash`
6. Start roscore: `roscore &`

5.4.2 ROS2

With the third version of the toolkit, ROS2 nodes to interact with the toolkit have been provided for all included tools.

5.4.3 Environment setup for ROS2

The instructions provided here will assume ROS2 `foxy` is already installed in your system, that a webcam is available and that you already installed the OpenDR toolkit.

1. Activate the `virtualenv` at the root of the OpenDR folder: `source bin/activate.sh`
2. Move to the ROS2 workspace: `cd projects/opendr_ws_2`
3. Source the ROS2 installation: `source /opt/ros/foxy/setup.bash`
4. Build the workspace: `colcon build`
5. Source it: `source install/setup.bash`

5.5 OpenDR on embedded devices

Two different docker images, that can be deployed on embedded devices, namely Nvidia's TX2, NX and AGX. The docker images are publicly available on dockerhub. The embedded devices should be flashed with Jetpack 4.6. To enable GPU usage on the embedded device within docker, first edit

```
/etc/docker/daemon.json
```

in order to set the default docker runtime:

```
{
    "runtimes": {
        "nvidia": {
            "path": "nvidia-container-runtime",
            "runtimeArgs": []
        }
    },
    "default-runtime": "nvidia"
}
```

Restart docker afterwards:

```
sudo systemctl restart docker.service
```

To run an interactive session :

```
sudo docker run -it opendr/opendr-toolkit:tx2_v3 /bin/bash
sudo docker run -it opendr/opendr-toolkit:nx_v3 /bin/bash
sudo docker run -it opendr/opendr-toolkit:agx_v3 /bin/bash
```

This will give you access to a bash terminal within the docker. After that you should enable the environment variables inside the docker with:

```
cd opendr
source bin/activate_nvidia.sh
source /opt/ros/noetic/setup.bash
source projects/opendr_ws/devel/setup.bash
```

The embedded devices docker comes preinstalled with the OpenDR toolkit. It supports all tools under perception package, as well as all corresponding ROS nodes. You can enable a USB camera, given it is mounted as /dev/video0, by running the container with the following arguments:

```
xhost +local:root
sudo docker run -it --privileged -v /dev/video0:/dev/video0 \
  opendr/opendr-toolkit:nx_v2 /bin/bash
```

To use the docker on an embedded device with a monitor and a usb camera attached, as well as network access through the hosts network settings you can run:

```
xhost +local:root
sudo docker run -it --privileged --network host \
   -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=unix$DSIPLAY  \
   -v /dev/video0:/dev/video0 opendr/opendr-toolkit:nx_v2 /bin/bash
```

5.6 Using the OpenDR toolkit

OpenDR provides extensive documentation for each tool that is part of the toolkit and is available here, as well as pointing to the available demo. A list of the available ROS nodes is available here, specific instructions are provided for each of the available nodes. Excerpts from the documentation are available in the annex.

It is worth noting that the toolkit recorded an impressive 2,000+ unique GitHub clones (originated from users who wanted to have full access to all of the capabilities provided by the toolkit) in just one year. Indeed, the total number of clones exceeded 100,000, which includes CI clones. Furthermore, an estimated 3,000 pulls have been performed for the ready to use docker images (which mainly targets less experienced developers who want to directly try the toolkit) and an estimated 10,000 pip downloads (originated from users who use selected parts of the Python API of the toolkit, e.g., individual tools without installing the whole toolkit). The larger number of pip downloads indicates that the target group is familiar with Python and prefers to install only the parts of the toolkit that are relevant to their needs. Note that the statistics for docker and pip have been adjusted to remove traffic that might have been automatically generated, e.g. by the CI system. Based on these statistics, OpenDR consortium estimates that at least 15,000 downloads from developers have been performed in this period, exceeding by far the corresponding M48 KPI target (a total of 500 downloads for the toolkit).

5.7 Customization

OpenDR can be readily customized to meet the needs of several application areas since the source code for all the developed tools is provided and the Apache 2.0 license is very permissive. Several ready-to-use examples, which are expected to cover a wide range of different needs, are provided. For example, users can readily use the existing ROS nodes by, for instance, including the required triggers or by combining several nodes into one to build more complex custom systems. An example of this is for instance the face recognition ROS node. In short, the user can use these nodes as a template to customize the toolkit to their needs. More in depth instructions on how to customize the toolkit are available in the dedicated page.

6 Conclusions

This document presents the work performed in WP7 about toolkit integration resulting in a final public version of the OpenDR toolkit. Notable changes includes continual SLAM, RL-based active perception, class filtering for object detection, model weights for new tasks, adaptive high resolution pose estimation, fall and wave detection ROS nodes, Voxel Pseudo Image tracking, intent recognition, Robotti human interaction simulation, RGB-based gesture recognition, new methods for non-maximum suppression, as well as speech transcription with Wishper and Vosk. Furthermore, this deliverable provided details about accessing, downloading and using the toolkit for all the provided installation methods. Finally, it also detailed additional research and development tasks conducted that were related to the development of the toolkit.

References

[1] D.-J. Boonstra, L. Willemet, J. Luijkx, and M. Wiertlewski. Learning to estimate incipient slip with tactile sensing to gently grasp objects. *under review*, 2023.

[2] D. Cattaneo, D. G. Sorrenti, and A. Valada. CMRNet++: Map and camera agnostic monocular visual localization in LiDAR maps. *Int. Conf. on Robotics and Automation Workshop on Emerging Learning and Alg. Methods for Data Association in Robotics*, 2020.

[3] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen. Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 12472–12482, 2020.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] A. Diaz-Diaz, M. Ocaña, A. Llamazares, C. Gómez-Huélamo, P. Revenga, and L. M. Bergasa. HD maps: Exploiting opendrive potential for path planning and map monitoring. In *IEEE Intelligent Vehicles Symposium*, pages 1211–1217, 2022.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Conf. on Robot Learning*, 2017.

[7] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada. Collaborative dynamic 3d scene graphs for automated driving. *arXiv preprint arXiv:2309.06635*, 2023.

[8] K. Koide, J. Miura, and E. Menegatti. A portable three-dimensional LiDAR-based system for long-term and wide-area people behavior measurement. *Int. Journal of Adv. Robotic Systems*, 16(2), 2019.

[9] M. Käppeler, K. Petek, N. Vödisch, W. Burgard, and A. Valada. Few-shot panoptic segmentation with foundation models. *arXiv preprint arXiv:2309.10726*, 2023.

[10] Y. Liao, J. Xie, and A. Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2023.

[11] R. Mohan and A. Valada. Perceiving the invisible: Proposal-free amodal panoptic segmentation. *IEEE Robotics and Automation Letters*, 7(4):9302–9309, 2022.

[12] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[13] R. Trumpp, M. Büchner, A. Valada, and M. Caccamo. Efficient learning of urban driving policies using bird's-eye-view state representations. *Int. Conf. on Intelligent Transportation Systems*, 2023.

[14] K. Wong, Y. Gu, and S. Kamijo. Mapping for autonomous driving: Opportunities and challenges. *IEEE Intelligent Transportation Systems Magazine*, 13(1):91–106, 2020.

[15] B. Yang, M. Liang, and R. Urtasun. HDNET: Exploiting HD maps for 3D object detection. In *Conf. on Robot Learning*, 2018.

7 Appendix

7.1 Using Part-based Representations for Explainable Deep Reinforcement Learning

The appended papers follow.

Using Part-based Representations for Explainable Deep Reinforcement Learning

Manos Kirtas[0000−0002−8670−0248], Konstantinos Tsampazis, Loukia Avramelou, Nikolaos Passalis[0000−0003−1177−9139], and Anastasios Tefas[0000−0003−1288−3667]

Computational Intelligence and Deep Learning Research Group
School of Informatics, Aristotle University of Thessaloniki, Greece.
{eakirtas, tsampaka, avramell, passalis, tefas}@csd.auth.gr

Abstract. Utilizing deep learning models to learn part-based representations holds significant potential for interpretable-by-design approaches, as these models incorporate latent causes obtained from feature representations through simple addition. However, training a part-based learning model presents challenges, particularly in enforcing non-negative constraints on the model's parameters, which can result in training difficulties such as instability and convergence issues. Moreover, applying such approaches in Deep Reinforcement Learning (RL) is even more demanding due to the inherent instabilities that impact many optimization methods. In this paper, we propose a non-negative training approach for actor models in RL, enabling the extraction of part-based representations that enhance interpretability while adhering to non-negative constraints. To this end, we employ a non-negative initialization technique, as well as a modified sign-preserving training method, which can ensure better gradient flow compared to existing approaches. We demonstrate the effectiveness of the proposed approach using the well-known Cartpole benchmark.

Keywords: Part-based Learning · Explainable Reinforcement Learning · Non-negative Constraints · Proximal Policy Optimization

1 Introduction

Deep Reinforcement Learning (RL) has achieved state-of-the-art performance in various applications, including robotics [1, 2]. However, the use of RL agents in critical environments, where safety is highly prioritized, is hindered due to the limited transparency of the models. Extracting the rationale of a deep learning (DL) model in a human-interpretable way remains a challenging task, but doing so would be highly useful for improving both the performance and trustworthiness of the model, as well as preventing failures [3]. To this end, *post-hoc* explanation methods have been extensively studied over the years, providing rationales for the predictions of the model [4, 5]. However, such approaches cannot always provide a reliable explanation [6, 7], with *pre-hoc* methods for explainable AI gaining increasing attention recently [8]. The *pre-hoc* approaches aim to design inherently explainable models, providing a transparent mechanism to the decision-making process in such a way that one can calibrate user trust and predict the system's capabilities.

To this end, extracting a part-based representation of deep learning models provides great potential for interpretable-by-design approaches, since they are based on the simple addition of latent causes acquired from feature representations, making models easily interpretable by human actors due to the elimination of canceling neurons [9, 10]. However, training a part-based learning model is challenging since it requires non-negative constraints to the model's parameters, leading to training difficulties, such as instabilities and convergence issues [11]. Furthermore, existing approaches for part-based learning are limited, e.g., applied solely on autoencoders [10, 12], and models that are not usually used in DL models, such as Pyramid Neural Networks [13, 14], resulting in a significant performance degradation [11], making them unsuitable for RL.

In this work, we propose a non-negative training approach for actor models in RL approaches, allowing for extracting part-based representations that can provide increased interpretability, while also building upon non-negative constraints that are known to be conceptually tied to human cognition [15, 16]. To this end, the proposed method employs a non-negative initialization method, along with an appropriately modified sign-preserving training method. More specifically, we propose using an exponential distribution-based non-negative initialization method for the actor model. Then, we introduce a sign-preserving alternative of Stochastic Gradient Ascent (SGA) that is used to train the actor model in a non-negative manner. The proposed optimization method allows better gradient flow, compared to existing clipping-based approaches, reducing the phenomenon of vanishing gradients and increasing the stability of the training process. As a result, the proposed pipeline enables more efficient training of inherently explainable models that are based on the non-negative part-based representation of the actor. Note that even though the proposed method is presented within the Proximal Policy Optimization (PPO) [17] algorithm, this is without loss of generality and could be readily adapted to any other Deep RL approach. We demonstrate the effectiveness of the proposed method in a traditionally used benchmark, named Cartpole, in a high-fidelity 3D robotics simulation.

The remainder of this paper is structured as follows. The proposed method is introduced and described in detail in Section 2, while the experimental evaluation is provided in Section 3. Finally, conclusions are drawn in Section 4.

2 Proposed Method

In this work, we focus on training non-negative agents using policy gradient-based approaches, such as the PPO algorithm [17], but without loss of generality, since the proposed method can also be directly applied to other RL methods as well, such as Q-learning based approaches. More specifically, PPO utilizes actor-critic networks, where the actor model decides which action should be taken, with its parameters denoted as $\boldsymbol{\theta}$. On the other hand, the critic network, equipped with parameters $\tilde{\boldsymbol{\theta}}$, informs the actor about the quality of its actions and guides the actor on how to adjust them during training. The PPO method trains the actor based on the policy gradient approach, while the critic evaluates the actions by computing the corresponding state/action values. For simplicity, we assume that both models have the same number of layers.

First, we propose an initialization scheme for the parameters of the actor model using an exponential distribution to ensure a positive-only initialization. Then, to train the actor model in a part-based representation manner, we propose a non-negative optimization approach based on Stochastic Gradient Ascent (SGA), ensuring that the canceling neurons of the network will be diminished by constraining parameters to the non-negative space, making them more easily interpretable by humans. Thus, only the actor network is trained in a non-negative manner, since it is responsible for the actions made by the agent during the deployment and, as a result, it is the one that needs to be explainable during deployment. Constraining to positive values only the parameters of the actor model, without similarly restricting the critic model, allows for reducing the risk of convergence issues that usually arise in non-negative neural networks [11]. At the same time, this does not reduce the interpretability of the actor model during deployment, since the critic is only used during the training process.

In the case of RL approaches with a policy gradient, the actor model is trained to learn a policy, $\pi(a|\mathbf{s})$, by observing the state s and returning the probability of selecting the action a. The groundbreaking results of the PPO are attributed to the constraints utilized in the policy parameter steps. More precisely, PPO employs the action probability ratio between the policy parameterization formulated as:

$$r_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(a|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}_{old}}(a|\boldsymbol{s}_t)} \in \mathbb{R}, \tag{1}$$

where $\pi_{\boldsymbol{\theta}}(a|\boldsymbol{s}_t)$ is the probability that policy π, with actor's parameters $\boldsymbol{\theta}$, selecting an action a when the agent observes environment state $\boldsymbol{s}_t$ at time step t and the previous step parameters are denotes as $\boldsymbol{\theta}_{old}$. Using the clipped version of the ratio $r_t(\boldsymbol{\theta})$ around the value of 1 within ϵ, the policy exploration can be constrained to the close vicinity of the parameter space. The clipped policy ratio is defined as:

$$r_t^{clip}(\boldsymbol{\theta}) = \text{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) \in [1-\epsilon, 1+\epsilon], \tag{2}$$

where ϵ is the constraint range of policy update and by default is set to $\epsilon = 0.2$, while the clip function is defined as:

$$\text{clip}(x, m, M) = \max(\min(x, M), m) \in [m, M]. \tag{3}$$

The final objective function of the PPO is defined as:

$$L^{actor}(\mathbf{s}_t; \boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \mathbb{E}_t \left[\min\left(r_t^{clip}(\boldsymbol{\theta}) A_t(\tilde{\boldsymbol{\theta}}), r_t^{clip}(\boldsymbol{\theta}) A_t(\tilde{\boldsymbol{\theta}})\right)\right] \in \mathbb{R}, \tag{4}$$

where $A_t(\tilde{\boldsymbol{\theta}})$ is the advantage. In this work, we use the General Advantage Estimation (GAE) [18] approach. The Temporal Difference (TD) residual for each time step t is calculated as:

$$\delta_t(\tilde{\boldsymbol{\theta}}) = R_t + \gamma V^{\pi}_{\tilde{\boldsymbol{\theta}}_t}(\mathbf{s}_{t+1}) - V^{\pi}_{\tilde{\boldsymbol{\theta}}_t}(\mathbf{s}_t) \in \mathbb{R}, \tag{5}$$

where R_t is the reward the agent receives at time step t, $V^{\pi}_{\tilde{\boldsymbol{\theta}}_t}(\mathbf{s}_t)$ is the value estimation predicted by the critic policy π for current state s_t based on critic parameter $\tilde{\boldsymbol{\theta}}_t$, γ is the discount factor and λ is the smoothing parameter. In this work, we use $\gamma = 0.99$ and $\lambda = 0.95$. Then, the advantage A_t is defined as:

$$A_t(\tilde{\boldsymbol{\theta}}) = \sum_{i=0}^{n-t} \gamma^i \lambda^i \delta_{t+i}(\tilde{\boldsymbol{\theta}}) \in \mathbb{R}, \tag{6}$$

where n is the total number of steps within an episode and t is the time step.

On the other hand, the critic network is typically trained to minimize the temporal difference between the returns and it is formulated as:

$$L^{critic} = \mathbb{E}_t[\delta_t(\tilde{\boldsymbol{\theta}})^2] \in \mathbb{R} \tag{7}$$

Traditionally used initialization schemes, such as Kaiming [19] and Xavier [20], oriented to ANNs that apply the ReLU activation function, initialize the parameters of the k-th layer around zero, drawing values from a Gaussian distribution, $\theta \sim \mathcal{N}(0, \sigma_k)$, where θ denotes a parameter from $\boldsymbol{\theta}$ and the standard deviation depends on the number of inputs and neurons. To this end, even by applying optimization methods that constrain parameters to positive values, it results in a very low variance of parameters during the first epochs of training that can lead to convergence difficulties or even halt the training process. To this end, inspired by [12], we propose to initialize actor parameters $\boldsymbol{\theta}$ in the positive domain using an exponential distribution given by:

$$\theta \sim \text{Exp}(\lambda) = \frac{\ln(U(0,1))}{\lambda} \in \mathbb{R}_+, \tag{8}$$

where $U(0,1)$ is a uniform distribution between $(0,1)$, $\mathbb{R}_+$ denotes the set of positive real values, and λ is the rate parameter of the distribution and it is a hyperparameter that by default is set to $\lambda = 100$. Even though initializing the actor parameters allows us to obtain a part-based representation before training, the traditionally used optimization algorithms, such as SGA, do not allow one to preserve the initial sign of the parameters. Therefore, we propose a sign-preserving optimization method that is based on the SGA. More specifically, we propose a sign-preserving alternative of SGA that modifies the update term, which is attributed to the sign change, ensuring that the trainable parameters will remain non-negative during the training phase. To this end, the actor's parameters are updated as:

$$\boldsymbol{\theta} = \left| \boldsymbol{\theta}_{old} + \eta_a \frac{\partial L^{actor}}{\partial \boldsymbol{\theta}_{old}} \right|, \tag{9}$$

where the η_a denotes the learning rate of the actor and $|\cdot|$ the absolute value operator.

On the other hand, critic model is trained as usual employing the standard Stochastic Gradient Descent (SGD) denoted as:

$$\tilde{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}}_{old} - \eta_c \frac{\partial L^{critic}}{\partial \tilde{\boldsymbol{\theta}}_{old}}, \tag{10}$$

where the η_c defines the learning rate of the critic model and L^{critic} denotes the loss function of the critic. We call the proposed sign preservation method *Absolute Stochastic Gradient Ascent* (ASGA), since it employs the absolute value operator to preserve the positive sign of parameters during the training process.

The proposed training method is presented algorithmically in Algorithm 1. More specifically, it receives as input the initialization hyperparameters for the actor model, which is initialized with the exponential distribution using $\lambda \in \mathbb{R}$, and for the critic model, which is initialized by a normal distribution using $\boldsymbol{\sigma} \in \mathbb{R}^n$, where n is the number of layers. The former network is optimized with the proposed sign-preserving alternative of SGA using learning rate η_a and the latter using the SGD optimizer with learning rate η_c. The algorithm outputs both actor's $\boldsymbol{\theta}$ and critic's $\tilde{\boldsymbol{\theta}}$ parameters, ensuring that the actor's parameters will be non-negative. Firstly, the proposed method initializes the parameters for both models (lines 2-3), using an exponential distribution for the actor (line 3), constraining to positive-only parameters, and a normal distribution for critic parameters (line 4). In turn, the method iterates over the environment for $T_{episodes}$ episodes (lines 6-12), applying the obtained policy ($\pi_{\boldsymbol{\theta}_{old}}$) for T_{steps} steps (lines 6-8), collecting the states, trajectories and rewards (line 7). Then, the PPO algorithm calculates the GAE (line 8) and the proposed method trains both actor and critic for T_{epochs} (line 9-12) using mini-batches (line 10-12). More precisely, the actor's parameters are updated using the proposed sign-preserving gradient ascent (line 11), while the actor is trained as usual by applying the SGD optimizer (line 12). This results in non-negative trained parameters for the actor model, making the part-based representation of the actor feasible.

3 Experimental Evaluation

We experimentally evaluated the proposed method on the typical RL benchmark Cartpole implemented using the Deepbots framework [21, 22]. More specifically, the simulated environment is composed of a four-wheeled cart that has a long pole attached to it by a free hinge. On the top of the pole, there is a sensor to measure their vertical angle. The pole acts as an inverted pole pendulum and the goal is to keep it vertical by moving the cart forward and backward.

We applied the PPO algorithm to all evaluated cases, with the actor network getting the observations of the agent as input, consisting of two hidden layers of 10 neurons each, and outputs the action of the agent. Similarly, the critic network gets the observations as input, and it outputs the advantage of each state. The critic network also consists of two hidden layers of 10 neurons. On both networks, we employed the ReLU activation function in the hidden layers, with the actor

Algorithm 1: Non-Negative Actor PPO Training

```
Input  : $\lambda \in \mathbb{R}$: the rate parameter of the exponential distribution,
         $\boldsymbol{\sigma} \in \mathbb{R}^n$: a vector containing the standard deviation for the Gaussian
         distribution for each layer,
         $\eta_a \in \mathbb{R}$ and $\eta_c \in \mathbb{R}$: learning rates for actor and critic.
Output: $\boldsymbol{\theta}$ : actor's parameters, and
         $\tilde{\boldsymbol{\theta}}$: critic's parameters.
1  begin
2      for $k = 1 \dots n$ do
3          $\theta^{(k)} \sim Exp(\lambda)$ ;                                  // Initialize Actor's Parameters
4          $\tilde{\boldsymbol{\theta}}^{(k)} \sim \mathcal{N}(0, \sigma_k)$ ;     // Initialize Critic's Parameters
5      for $i = 1 \dots T_{episodes}$ do
6          for $j = 1 \dots T_{iter}$ do
7              Apply policy $\pi_{\boldsymbol{\theta}_{old}}$ and collect state, actions and rewards;
8              $A_j(\tilde{\boldsymbol{\theta}}) = \sum_{i=0}^{n-t} \gamma_i \lambda_i \delta_{t+i}(\tilde{\boldsymbol{\theta}})$ ;  // Compute Advantage estimates
9          for $j = 1 \dots T_{epochs}$ do
10             for every batch do
11                 $\boldsymbol{\theta} = \left| \boldsymbol{\theta} + \eta_a \frac{\partial L^{actor}}{\partial \boldsymbol{\theta}} \right|$ ;   // Update Actor's Parameters
12                 $\tilde{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}} - \eta_c \frac{\partial L^{critic}}{\partial \tilde{\boldsymbol{\theta}}}$ ;   // Update Critic's Parameters
13     return $\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}$;
```

model setting an upper bound to ReLU at 2 (ReLU2) for better visualization of the model. The observations contain the cart position, velocity on the x-axis, vertical angle of the pole, and pole's velocity at its tip. The available discrete actions at each step are either to move forward or backward. For each step, the agent is rewarded with +1 and each episode ends after $T_{steps} = 195$ steps or earlier if the pole has fallen $\pm 15°$ off vertical or if the cart has moved more than ± 39 centimeters on the x-axis. The networks are optimized for $T_{episodes} = 10^4$ episodes with learning rates equal to $\eta_a = 0.1$ and $\eta_c = 0.003$ for actor and critic, respectively. The PPO iterates for $T_{iter} = 5$ over batches of 8 collected samples.

We evaluate the proposed method on the aforementioned setup against two baseline methods using: a) Clipping Stochastic Gradient Ascent (CSGA) with Kaiming initialization and b) CSGA with Xavier initialization. These are based on the clipping approach proposed in [12, 11], after appropriate adaptation for use in PPO. More specifically, the CSGA applies a clipping function to the typical stochastic gradient ascent and it is formulated as:

$$\boldsymbol{\theta}' = \max\left(0, \boldsymbol{\theta} - \eta \frac{\partial J}{\partial \boldsymbol{\theta}}\right), \tag{11}$$

where the max(·) operator is applied element-wise. In all cases, the evaluated optimization methods are used on the actor model, ensuring the non-negativity

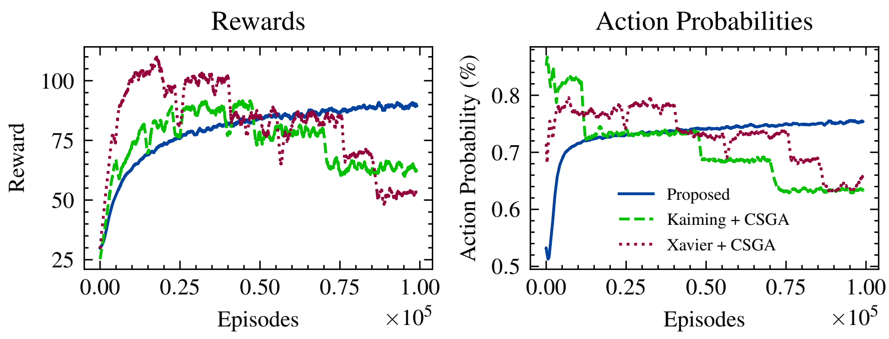

Fig. 1. On the left, the figure depicts the obtained reward during training that is smoothed using a moving average filter with a window of 100. On the right, the action probabilities for each method are depicted using the same moving average setting.

of its parameters, with the critic model being trained as usual with the SGD optimizer.

Table 1. Average and variance of rewards both for training and evaluation phase over 5 runs.

Method	Training	Evaluation
CSGA (Kaiming Init.)	62.83 ± 39.64	89 ± 98.59
CSGA (Xavier Init.)	53.67 ± 35.47	58.2 ± 78.4
Proposed	$\mathbf{89.45 \pm 1.04}$	$\mathbf{140.4 \pm 43.9}$

We conducted five evaluation runs for each method using different seeds during the training phase. In Table 1 the average and variance of the rewards over 5 runs are reported for all the evaluated methods. Both baselines lead to an unstable training process, resulting in high variance reward values at the end of the training. In contrast, the proposed method offers significantly more consistent training, resulting in low-variance rewards after training, as well as higher performance. This behavior is also highlighted in the evaluation performance, where the proposed method holds the pole for more than 50 steps in the average case contrary to the other evaluated baselines.

To highlight such different behavior, we report in Figure 1, the average reward between different runs and the average action probability during training. For each case, we smooth the reported result using a moving average filter, setting the window value equal to 100. Similarly to the results reported in Table 1, we observe that during training, the two baselines are highly unstable, resulting in a poor local minimum and, as a result, significantly lower rewards. On the other hand, the proposed method allows for more consistent training, achieving significantly higher performance than the baselines.

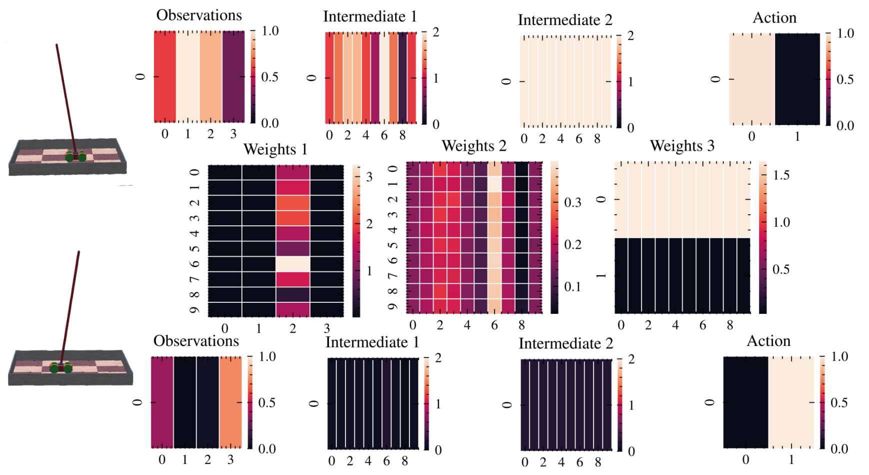

Fig. 2. Part-based representation of the actor model. At the top and bottom rows, the input and responses of each layer are depicted. In the central row, the weights of the actor model are depicted. The biases are omitted for simplicity.

This instability in training can also be observed in the action probability plot, where the baselines lead to increases at a high rate during the initial stage of training, before subsequently gradually decreasing the corresponding probabilities. This indicates that baselines converge quickly to a bad local policy, with the clipping update term introducing difficulties in the training process. Such difficulties can be attributed to the fact that the clipping method zeros out synapses when they try to change sign, reducing the learning capacity of the model. This can also lead to vanishing gradient phenomena, which in turn can lead to bad local minima or even halt the training process [23]. On the other hand, the proposed optimization method ensures that the parameters will remain non-negative without suppressing weights to zero, allowing gradients to flow through the network since the absolute value operator has a non-zero derivative both for positive and negative values. In this way, it provides a smooth training process and consistent results, as well as a more explainable representation for the actor model, allowing one to extract rationales of the agent due to the non-negative constraints that diminish canceling neurons and leading to part-based representations.

To highlight this, in Figure 2 we provide an example of the part-based representation acquired using the proposed method. More specifically, two cases are presented: a) when the pole is dropped from the front of the cart (top row) and b) when the pole is falling from the rear side of the cart (bottom row). On the middle row, the weights of the model are depicted, omitting biases for simplicity. The observations are presented on the first plot from the left and are normalized between [0, 1], with columns on the plot denoting from 0 to 3: 0) cart position, 1) cart velocity, 2) pole angle and 3) endpoint velocity. The action made by the actor is represented on the right plot with the zeroth neuron response denoting the forward move of the agent (left as depicted in the figure) and the first neuron the backward move (right as depicted in the figure).

As depicted in Figure 2, applying part-based learning by using the proposed method allows one to extract visually meaningful representations. More precisely, when the pole is dropped from the positive side of the x-axis, meaning on the front side of the cart, and has large values for the pole angle, the agent moves the cart toward the same direction as the falling pole trying to keep it vertical, as depicted in the first row of the figure with the lighter heatmaps. On the other hand, when the pole has lower values on the pole angle (darker colors in the observations' heatmap are used to denote this), the agent moves the cart backward. Lighter heatmaps when multiplied with the final layer's weights (weights 3 in the figure) of the actor lead to firing the zeroth neuron of the layer that is translated on moving the cart forward. We can safely conclude that the agent has high confidence in its decision by comparing the intermediate representation of two cases after the second layer, since the two representations are significantly different. As depicted, in the first case (first row of the figure), the last layer zeroes out the response of the first neuron, maintaining the large values on the input of the zeroth neuron. On the contrary, in the second case, the input of the last layer already has values close to zero, with the first layer increasing the values of the input to fire the zeroth neuron. We have to mention that the action heatmap of the figure is after the softmax activation function, translating the response of the last layer to action probabilities.

We can also extract the rationale for the agent based on other observations. For example, the second observation (meaning the column denoted by 1 in the figure) refers to the cart velocity (normalized between range [0, 1]). This means that if the value of cart velocity is lower than 0.5, then the cart has a direction to the negative side of the x-axis (backward movement). Accordingly, if the cart velocity value is greater than 0.5, then the cart is moving forward, towards the positive side of the x-axis. As depicted in the observations' heatmap, the agent moves toward the opposite direction of the direction that the pole is falling. This is expected since the observations are extracted from an irrational agent, where the actor parameters have been randomly initialized. However, the trained agent using the proposed method outputs actions that moves the cart in the same direction as the falling pole to keep the pole vertical. This can also be observed in the heatmap of the weights of the first layer, where the third column has an inverse color with respect to the second column. Indeed, there is an inverse correlation between the cart position and the pole's angle.

We extend our analysis to obtain further insight regarding the response of the actor model, leveraging the advantages provided by the part-based representation of the model. To this end, we optimized the observation vector keeping the trained weights frozen in order to minimize the distance between the output of the model and a given action. The inputs are optimized for 5 epochs using SGD optimizer with the proposed sign-preserving update function applying mean squared error loss. We applied the proposed optimization method to ensure

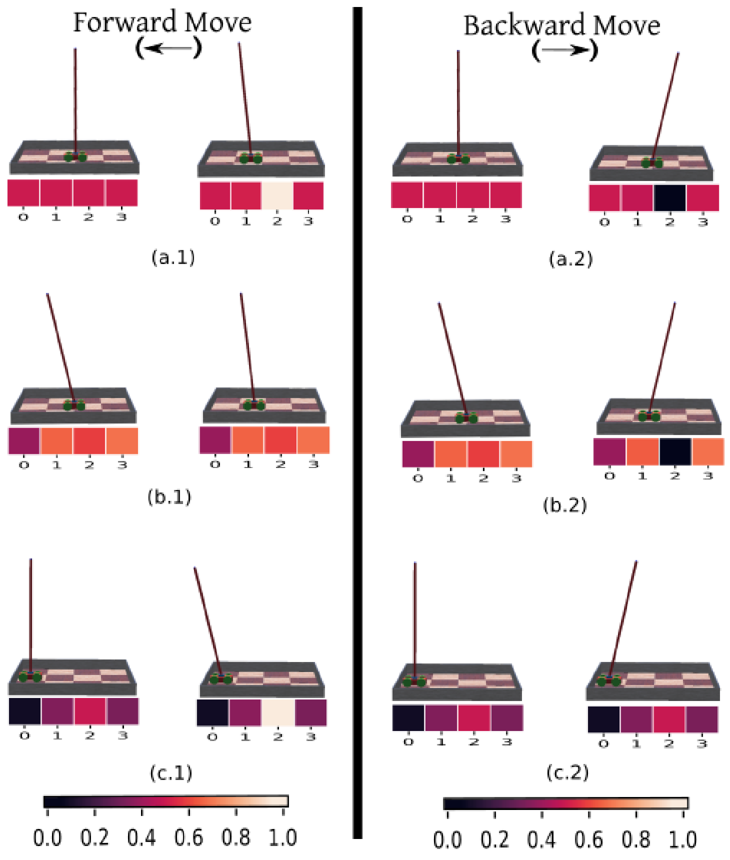

Fig. 3. Optimized input of actor model to maximize the action probability of a given action using three different initialization. In the first column, the observations are optimized to maximize the forward action probability. In the second column, the observations are optimized to maximize the backward action probability. A different initialization of the observation vector is used for each row.

that the values of the observations will remain non-negative according to the actual normalized observations acquired from the environment. The observations obtained are presented in Figure 3, where in the left column they are optimized to maximize the action probability of the forward action, and, respectively, on the right side they are optimized to maximize the action probability for the backward action. For each case, we report both the initial vector of observation and the one after the optimization process.

To evaluate which of the observations is more significant for each action, we initialize each observation to 0.5 and then optimize them to maximize the respective action probability, depicted in the (a) row. As expected, for both actions, the observation that changed significantly is the pole angle (in the third column).

Indeed, when the observations are optimized in order to obtain a forward action from the actor model, the pole angle value converges to its maximum, which is value one. On the other hand, when the observations are optimized in order to obtain a backward action, the pole angle value converges to a value close to zero. In both cases, the other observations remain close to the initialized values, and it seems that they are not significantly affected by the optimization process. This is an expected behavior since the pole angle is the most significant indicator, revealing the direction in which the pole is falling, and as a result the one that significantly contributes to the prediction of the actor model.

We repeat the aforementioned experimental setup by applying different initializations on the observations vector and reporting the results at the (b) and (c) rows. More specifically, we randomly draw values from a Gaussian distribution with 0.5 mean and 0.2 standard deviation. As shown, except for the pole angle observation, which is significantly changed in most cases, the rest of the observations are not affected by the optimization process. Regarding the observation of the pole angle, when it has opposite direction with the given action, the obtained direction of the pole angle after optimization is inverted, such in case (b.2). In cases where the initialized observation vector has the same direction of the pole angle with the given action, as in cases of (b.1), the optimization process slightly changes the initial vector and the value of the pole angle. Finally, it is observed that when the initialized pole is vertical, such as in (a) and (c) cases, the optimization process leads to maximizing or minimizing the pole angle value according to the given action, resulting in the same direction with the action.

4 Conclusions

In this study, we have introduced a novel training approach that focuses on non-negativity in deep RL using PPO. The proposed approach enables the extraction of part-based representations, which offers enhanced interpretability while following non-negative constraints associated with human cognition. To achieve this objective, the proposed method employs a non-negative initialization technique, followed by a modified sign-preserving training method. More specifically, we proposed employing an exponential distribution-based non-negative initialization method for the actor model and then using an appropriately modified sign-preserving alternative to Stochastic Gradient Ascent (SGA) for training the actor model in a non-negative manner. By adopting the proposed method, we can mitigate issues related to the reduction of the learning capacity of models and the vanishing gradients due to the use of clipping mechanisms involved in existing approaches. This helps mitigate issues such as vanishing gradients and enhances training stability. Consequently, the proposed pipeline enables more efficient training of inherently explainable models based on the non-negative part-based representation of the actor. To validate the effectiveness of the proposed method, we conducted experiments on the well-established Cartpole benchmark. The results demonstrate the effectiveness of the proposed method in achieving superior performance and showcasing the advantages of the proposed non-negative training methodology.

The promising results reported in this paper highlight several interesting future research directions. First, the proposed method can also be extended to handle value-based RL approaches, such as DQN [24]. Furthermore, extending the part-based representation learning to the actor model could also provide further insight into the training dynamics of the RL process, as well as allow for better adapting it to the task at hand, potentially leading to more robust algorithms. Finally, combining the proposed method with distillation approaches that can transfer knowledge from the intermediate layers of traditional DL models, for example [25], could potentially allow for better guidance of the optimization process and learning more accurate policies.

Acknowledgments

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

1. 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, 2013.
2. B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.
3. B. Hayes and J. A. Shah, "Improving robot controller transparency through autonomous policy explanation," in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, (New York, NY, USA), p. 303–312, Association for Computing Machinery, 2017.
4. M. T. Keane and E. M. Kenny, "How case-based reasoning explains neural networks: A theoretical analysis of xai using post-hoc explanation-by-example from a survey of ann-cbr twin-systems," in *Case-Based Reasoning Research and Development* (K. Bach and C. Marling, eds.), (Cham), pp. 155–171, Springer International Publishing, 2019.
5. S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.
6. D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, (New York, NY, USA), p. 180–186, Association for Computing Machinery, 2020.
7. Y. Zhou, S. Booth, M. T. Ribeiro, and J. Shah, "Do feature attribution methods correctly attribute features?," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 9623–9633, Jun. 2022.

8. C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, “Interpretable machine learning: Fundamental principles and 10 grand challenges,” *Statistics Surveys*, vol. 16, no. none, pp. 1 – 85, 2022.
9. A. Lemme, R. F. Reinhart, and J. J. Steil, “Online learning and generalization of parts-based image representations by non-negative sparse autoencoders,” *Neural Networks*, vol. 33, pp. 194–203, 2012.
10. E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, “Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2486–2498, 2016.
11. J. Chorowski and J. M. Zurada, “Learning understandable neural networks with nonnegative weight constraints,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 62–69, 2015.
12. B. O. Ayinde and J. M. Zurada, “Deep learning of constrained autoencoders for enhanced understanding of data,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 3969–3979, 2018.
13. M. Ferro, B. Fernandes, and C. Bastos-Filho, “Non-negative structured pyramidal neural network for pattern recognition,” in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–7, 2018.
14. M. S. A. Ferro, B. J. T. Fernandes, and C. J. A. Bastos-Filho, “Non-negative pyramidal neural network for parts-based learning,” in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1709–1716, 2017.
15. D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
16. K. Tanaka, “Columns for Complex Visual Object Features in the Inferotemporal Cortex: Clustering of Cells with Similar but Slightly Different Stimulus Selectivities,” *Cerebral Cortex*, vol. 13, pp. 90–99, 01 2003.
17. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint - 1707.06347*, 2017.
18. J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” *arXiv preprint - 1502.05477*, 2015.
19. K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE International Conference on Computer Vision*, December 2015.
20. X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.
21. M. Kirtas, K. Tsampazis, N. Passalis, and A. Tefas, “Deepbots: A webots-based deep reinforcement learning framework for robotics,” in *Proceedings of the International Conference on Artificial Intelligence Applications and Innovations*, pp. 64–75, 2020.
22. M. Kirtas, K. Tsampazis, P. Tosidis, N. Passalis, and A. Tefas, “Chapter 21 - deep learning for robotics examples using opendr,” in *Deep Learning for Robot Perception and Cognition* (A. Iosifidis and A. Tefas, eds.), pp. 579–596, Academic Press, 2022.
23. M. Kirtas, N. Passalis, G. Mourgias-Alexandris, G. Dabos, N. Pleros, and A. Tefas, “Robust architecture-agnostic and noise resilient training of photonic deep learning models,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

24. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
25. N. Passalis and A. Tefas, “Learning deep representations with probabilistic knowledge transfer,” in *Proceedings of the European Conference on Computer Vision*, pp. 268–284, 2018.

7.2 Few-Shot Panoptic Segmentation With Foundation Models

The appended paper [9] follows.

Few-Shot Panoptic Segmentation With Foundation Models

Markus Käppeler[1*], Kürsat Petek[1*], Niclas Vödisch[1*], Wolfram Burgard[2], and Abhinav Valada[1]

***Abstract*— Current state-of-the-art methods for panoptic segmentation require an immense amount of annotated training data that is both arduous and expensive to obtain posing a significant challenge for their widespread adoption. Concurrently, recent breakthroughs in visual representation learning have sparked a paradigm shift leading to the advent of large foundation models that can be trained with completely unlabeled images. In this work, we propose to leverage such task-agnostic image features to enable few-shot panoptic segmentation by presenting Segmenting Panoptic Information with Nearly 0 labels (SPINO). In detail, our method combines a DINOv2 backbone with lightweight network heads for semantic segmentation and boundary estimation. We show that our approach, albeit being trained with only ten annotated images, predicts high-quality pseudo-labels that can be used with any existing panoptic segmentation method. Notably, we demonstrate that SPINO achieves competitive results compared to fully supervised baselines while using less than 0.3% of the ground truth labels, paving the way for learning complex visual recognition tasks leveraging foundation models. To illustrate its general applicability, we further deploy SPINO on real-world robotic vision systems for both outdoor and indoor environments. To foster future research, we make the code and trained models publicly available at `http://spino.cs.uni-freiburg.de`.**

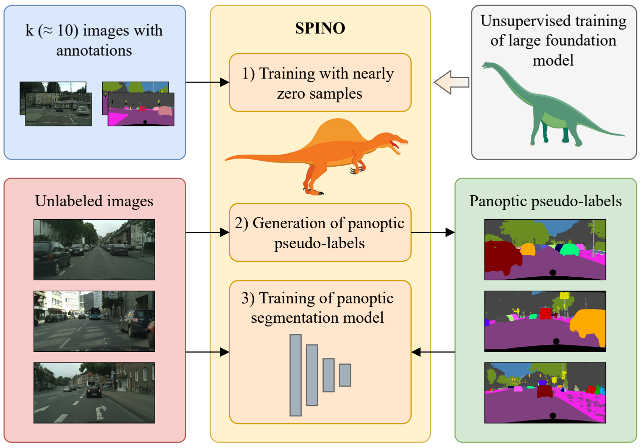
k (≈ 10) images with annotations
SPINO
Unsupervised training of large foundation model
1) Training with nearly zero samples
Unlabeled images
2) Generation of panoptic pseudo-labels
Panoptic pseudo-labels
3) Training of panoptic segmentation model

Fig. 1. SPINO enables few-shot panoptic segmentation by exploiting descriptive image features from unsupervised task-agnostic pretraining. We generate panoptic pseudo-labels by learning from only $k \approx 10$ annotated images in an offline manner. We can then leverage these pseudo-labels to train any panoptic segmentation model enabling online deployment.

I. Introduction

Panoptic segmentation [1] poses an important contribution to holistic scene understanding by enabling robots to assign semantic meaning to their environment while delineating individual objects. However, most previous methods addressing panoptic segmentation rely on supervised training [2], [3], hence requiring a large amount of ground truth labels. This hinders their widespread adoption as generating panoptic annotations is both expensive and time-consuming, e.g., manually labeling a single high-resolution image of urban scenarios takes approximately $1.5\,\mathrm{h}$ [4]. Therefore, it is paramount to reduce the number of required labels [5], e.g., by advancing weakly- and unsupervised methods or by leveraging task-agnostic pretraining strategies [6].

Facing similar issues, the domain of natural language processing (NLP) has recently seen a rise of large foundation models [7]. This paradigm shift in NLP also inspired the vision community to propose similar methods such as CLIP [8] or Segment Anything [9]. While both still require some supervision signal, e.g., from image captions or coarse object masks, DINO [10] learns visual representation in a fully unsupervised manner allowing to significantly extend the amount of usable resources. Prior works have shown that one can bootstrap such general representations for several tasks including depth estimation [11], semantic segmentation [11], [12], and object detection [13]. Based on these findings, we argue that it is time for a fundamental paradigm switch for vision tasks that exploit task-agnostic foundation models to enable few-shot training. In contrast to unsupervised techniques [12], [14], we show that such an approach can yield results competitive with fully supervised learning methods.

In this work, we present a method for *Segmenting Panoptic Information with Nearly 0 labels* (SPINO). As illustrated in Fig. 1, we first leverage a frozen DINOv2 [11] backbone to extract visual features. We subsequently train two task-specific heads for semantic segmentation and boundary estimation with as few as ten annotated images to perform few-shot panoptic segmentation. To enable real-time inference and to further boost the quality of our predictions, we generate panoptic pseudo-labels in an offline manner for a larger bag of raw images that can then be used to train any existing panoptic segmentation model. We perform extensive evaluations on several public [4], [15] and in-house datasets that demonstrate that our SPINO approach yields results that are highly competitive with fully supervised learning models. In particular, our extensive evaluations suggest that few-shot panoptic segmentation provides the means to soon become on par with supervised state-of-the-art methods.

To summarize, the main contributions are as follows:

1) We propose the first method for few-shot panoptic

[*] Equal contribution.
[1] Department of Computer Science, University of Freiburg, Germany.
[2] Department of Eng., University of Technology Nuremberg, Germany.
This work was funded by the German Research Foundation (DFG) Emmy Noether Program grant No 468878300 and the European Union's Horizon 2020 research and innovation program grant No 871449-OpenDR.

segmentation based on unsupervised foundation models.
2) We present a novel pseudo-label generation scheme that can be trained with as few as ten annotated images.
3) We show that SPINO yields results that are competitive to supervised training with ground truth labels.
4) In extensive evaluations, we illustrate the effect of various architectural design choices and apply our method to real-world robotic vision platforms.
5) We make the code and trained models publicly available at `http://spino.cs.uni-freiburg.de`.

II. Related Work

In this section, we present an overview of panoptic segmentation, visual representation learning, and both unsupervised and weakly-supervised image segmentation techniques.

Panoptic Segmentation: Panoptic segmentation [1] combines semantic and instance segmentation into a single task with two categories of scene elements. The static background comprises the so-called "stuff" classes such as *buildings*, whereas dynamic objects such as *cars* belong to the "thing" category. While "stuff" classes only receive a semantic label, "thing" classes are further separated on an instance level. Since the introduction of this task, several deep learning-based methods [2], [16]–[19] have been proposed requiring a large amount of data for training. Recently, the focus has shifted towards more challenging variants, e.g., open-vocabulary methods such as from Ding *et al.* [20] leveraging insights from foundation models [8]. Removing the need for labels, CoDEPS [21] addresses unsupervised domain adaptation from a source to a previously unseen target domain. In this work, we propose a method for few-shot panoptic segmentation requiring as few as ten annotated images.

Visual Representation Learning: Breakthroughs in natural language processing (NLP) [7] have shown that task-agnostic pretraining can yield feature representations that, fine-tuned to specific applications, become competitive with prior state-of-the-art methods [22]. A common approach to obtaining similar representations in the visual domain is contrastive learning [23]. However, although not using human annotations, the choice of the dataset still introduces a significant bias on the learned representation that can be mitigated by extensive data augmentation [24]. Masked autoencoders (MAE) [25] represent another type of self-supervised learners that learn to reconstruct areas in an image that have been masked. After pretraining, MAEs can be fine-tuned for various downstream tasks. More recently, the usage of foundation models in NLP has also started to influence computer vision. For instance, CLIP [8] leverages insights from constrastive learning by exploiting textual supervision to guide the learning of visual features. However, this text-guided supervision strategy limits the choice of training data. SAM [9] removes the need for captions and relies on a self-iterative training scheme starting from coarse object masks. While showing impressive zero-shot performance for semantic segmentation on unseen domains, it lacks the ability to assign class labels to the segments. Finally, DINO [10] represents a new family of foundation models that can be trained only from raw images. In particular, DINO demonstrates that such unsupervised pretraining can achieve even more explicit features for semantic segmentation than their supervised counterparts. Further advances have been shown by DINOv2 [11] that combines several prior insights with training on a curated dataset. In this work, we exploit descriptive image features from a DINOv2 backbone to generate panoptic pseudo-labels.

Unsupervised and Weakly-Supervised Segmentation: Since pixel-wise annotations for supervised training of image segmentation tasks are expensive to obtain, research in the last few years has shifted towards reducing the number of human annotations. Recent methods build on the observation that features from unsupervised pretraining are semantically consistent across images from differing domains [12]. For instance, LOST [26] uses DINO [10] features for bounding box extraction to bootstrap supervised training of an object detector. Objects can be assigned to the same class via k-means clustering in the feature space. Similarly, TokenCut [27] relies on Normalized Cut (NCut) [28] to group self-similar image regions based on DINO features. While these previous methods work well for foreground/background segmentation, FreeSOLO [29] addresses multi-object detection by enhancing coarse masks via one-stage self-training in a weakly supervised manner. However, requiring in-domain data results in a lack of generalization. In contrast, CutLER [13] achieves impressive zero-shot performance leveraging DINO features to generate coarse masks followed by weakly supervised training of a separate instance segmentation network. Although applicable to multi-object scenarios, relying on iterative NCut requires specifying the number of expected objects.

With respect to semantic segmentation, MaskContrast [30] and PiCIE [14] are notable methods from before the advent of large pretraining models. While MaskContrast contrasts learned features within and across saliency masks, PiCIE searches for descriptive image features guided by photometric invariance and geometric equivariance. Recently, both MaskDistill [31] and STEGO [12] leverage features from a frozen DINO [10] backbone. To further refine the pretrained features, STEGO adds a task-specific segmentation head followed by clustering. Other examples of exploiting foundation models include CLIP-ES [32], which relies on contrastive language-image pretraining [8], and SEPL [33] that combines the class-agnostic masks from SAM [9] with class activation maps for class assignment. To the best of our knowledge, our proposed SPINO constitutes the first attempt to directly exploit fully unsupervised representation pretraining for panoptic segmentation.

III. Technical Approach

In this section, we present our proposed approach SPINO for few-shot panoptic segmentation. As illustrated in Fig. 2, we leverage the recent foundation model DINOv2 [11] to extract descriptive image features for both semantic segmentation and boundary estimation. In particular, we propose a novel pseudo-label generation scheme that separates

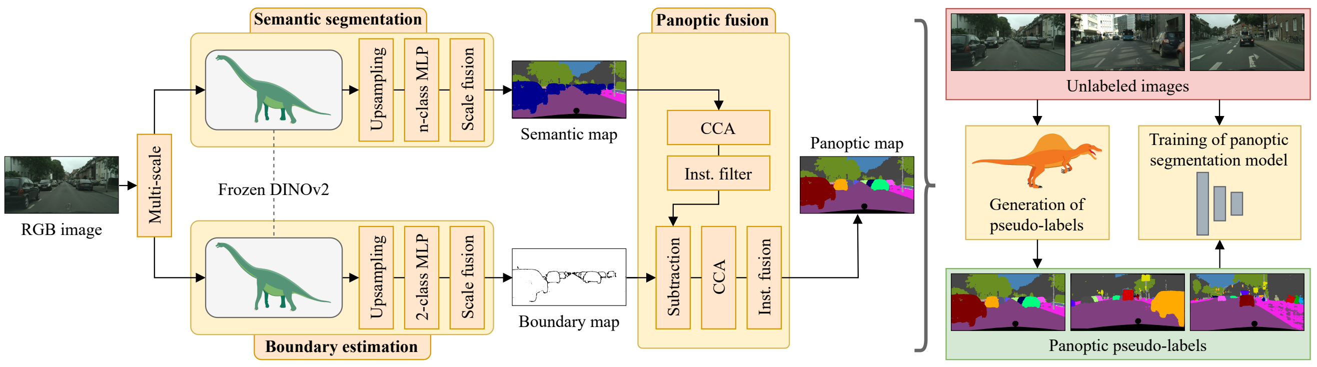

Fig. 2. Overview of our proposed SPINO approach for few-shot panoptic segmentation. SPINO consists of two learning-based modules for semantic segmentation and boundary estimation that leverage features from the recent foundation model DINOv2 [11]. A panoptic fusion scheme combines their outputs using connected component analysis (CCA) and multiple small instance filtering steps. SPINO creates pseudo-labels for a large number of unlabeled images using only $k \approx 10$ images with ground truth annotations. These pseudo-labels can then be utilized to train any panoptic segmentation model.

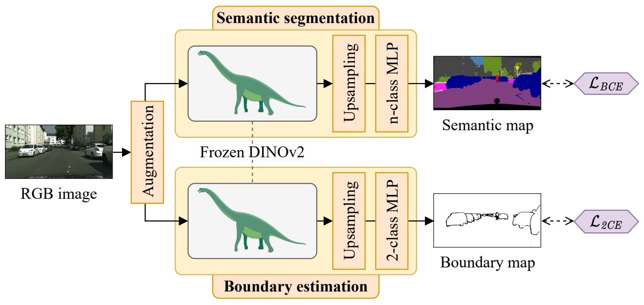

Fig. 3. Our proposed pseudo-label generator comprises two learnable modules for semantic segmentation and boundary estimation that exploit descriptive image features from the recent DINOv2 [11] foundation model, enabling training with only $k \approx 10$ ground truth panoptic annotations.

semantic regions of "thing" classes into individual instances by predicting object boundaries. With this approach, SPINO can bootstrap very few ground truth annotations for generating high-quality panoptic pseudo-labels. To enable real-time inference and to further boost the quality of our panoptic predictions, we train a panoptic segmentation model using the generated pseudo-labels.

A. *Few-Shot Pseudo-Label Generation*

We propose a novel panoptic segmentation scheme to generate panoptic pseudo-labels in an offline manner while requiring very few ground truth annotations for training. Our label generator consists of three main building blocks shown in Fig. 2, namely learnable modules for semantic segmentation and boundary estimation as well as a static component to fuse their predictions. The semantic segmentation module is comprised of a frozen DINOv2 [11] backend, a bilinear 14x-upsampling layer, and a final n-class MLP with 4 layers. Here, n denotes the number of semantic classes as specified in Sec. IV-A. In detail, we use the DINOv2 weights of the ViT-B/14 variant provided by the authors. For the boundary estimation module, we employ a similar design but use 4x-upsampling and set $n = 2$ for binary classification.

Training the Label Generator: A key idea of SPINO is to train our proposed pseudo-label generator with only k ground truth annotations, where k denotes numbers as small as 10. Notably, the unsupervised training procedure of DINOv2 does not further increase this number even when considering the pretraining. We illustrate the training of our pseudo-label generator in Fig. 3. First, to stabilize the training with such few samples, we employ various data augmentation techniques on the input RGB image including random cropping, horizontal flipping, and color jitter. Subsequently, we feed the augmented image to the two task-specific heads and compute the respective loss functions.

We supervise the semantic segmentation head with the bootstrapped cross-entropy loss function $\mathcal{L}_{BCE}$ [34] to account for rare classes.

$$\mathcal{L}_{BCE} = -\frac{1}{K}\sum_{i=1}^{N} \mathbb{1}\left[p_{i,y_i} < t_K\right] \cdot \log(p_{i,y_i}), \qquad (1)$$

where p_{i,y_i} denotes the posterior probability of pixel i for its ground truth class $y_i \in \{1, ..., c\}$ with c being the number of classes. The indicator function $\mathbb{1}(\cdot)$ is 1 if p_{i,y_i} is below a threshold t_K and 0 otherwise. We set $t_K = 0.2$ such that only those pixels with top-K highest losses contribute to $\mathcal{L}_{BCE}$. In order to train the boundary estimation module, we generate ground truth boundary maps as follows: If the instance ID of a pixel is different from any of its eight neighbors, we assign 1 to this pixel. Otherwise, we set the value of the center pixel to 0. During training, we compute the binary cross entropy loss $\mathcal{L}_{2CE}$ as the supervision signal.

$$\mathcal{L}_{2CE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p_{i,y_i}) + (1-y_i) \cdot \log(1-p_{i,y_i}), \qquad (2)$$

where $y_i \in \{0, 1\}$ is the binary boundary label of pixel i and p_{i,y_i} denotes the posterior probability.

Employing the Label Generator: In the next step, we leverage the aforementioned trained modules for semantic segmentation and boundary estimation to generate panoptic pseudo-labels for a large number of unlabeled images. In the following, we describe the procedure as depicted in Fig. 2. Inspired by ensemble learning, we use multi-scale test-time augmentation for both semantic segmentation and boundary estimation. For instance, for scale $s = 2$, we divide the image into four equally sized regions, upsample each region to the size of the $s = 1$ image, and obtain their softmax features. In the scale fusion block, we downsample these feature maps to the original size of the region, join the features of all regions in a single $s = 1$ map, and compute the mean across the considered scales. In detail, we use scales $\{1, 2, 3\}$ for the semantic head and scales $\{3, 4, 5\}$ for the boundary estimation head. Next, we feed the predicted semantic map and the estimated object boundary map to our panoptic fusion module. First, for each "thing" class, we perform connected component analysis (CCA) yielding disconnected blobs. If a blob consists of fewer pixels than a threshold, we assign the semantic *void* class to its pixels. Otherwise, we subtract the predicted border for this blob from the semantic map followed by CCA to detect separate instances within a blob. If the number of pixels of an instance is below another threshold, we add it to its nearest neighbor which fulfills the minimum size requirement. If all instances of a blob are below this threshold, we combine them into a single instance. Finally, due to the top-down approach, the inferred instance maps already contain semantic information leading to the desired pseudo-labels for panoptic segmentation.

B. Training a Panoptic Segmentation Model

After creating pseudo-labels for a large set of unlabeled images, we train a panoptic segmentation model as illustrated in Fig. 2. In contrast to the offline label generator, such a model allows for online panoptic segmentation while further enhancing the overall performance. Although this approach is generally applicable to any panoptic segmentation model, in this work, we follow the spirit of our pseudo-label generator. In detail, our bottom-up panoptic segmentation network consists of a frozen DINOv2 [11] backbone with an adapter module [35] and three task-specific heads [2] for semantic segmentation, instance center prediction, and pixel offset regression, respectively. In Fig. 4, we visualize this architecture. The semantic head predicts a semantic class for each pixel and is trained with the bootstrapped cross-entropy loss with hard pixel mining [2].

$$\mathcal{L}_{BCEH} = -\frac{1}{K}\sum_{i=1}^{N} w_i \cdot \mathbb{1}\left[p_{i,y_i} < t_K\right] \cdot \log(p_{i,y_i}) , \quad (3)$$

which builds upon Eq. (1) but adds weights $w_i > 1$ for pixels that belong to small instances. For other instances and "stuff" classes, the pixel weight remains at $w_i = 1$. Addressing instance segmentation, the center head generates a probability map with high values for instance centers and the offset head estimates the 2D offset of a pixel to the nearest instance center. To train these heads, we utilize the MSE loss $\mathcal{L}_{MSE}$ for the center head and the L1 loss $\mathcal{L}_{L1}$ for the offset head. Consequently, we compute the total loss as a weighted sum:

$$\mathcal{L}_{PAN} = \lambda_{sem}\mathcal{L}_{BCEH} + \lambda_{cen}\mathcal{L}_{MSE} + \lambda_{off}\mathcal{L}_{L1} \quad (4)$$

To increase the learning speed, we propose to further exploit the k annotated images, which were used to train the pseudo-label generator, also when training the panoptic segmentation model. In particular, we construct batches that contain both pseudo-labels and one ground truth sample. Formally, a batch $\mathbf{b}$ of size n is given by

$$\mathbf{b} = \{\hat{\mathbf{I}}_\mathbf{1}, \dots, \hat{\mathbf{I}}_{\mathbf{n-1}}, \mathbf{I}_{GT}\} , \quad (5)$$

where $\hat{\mathbf{I}}_\mathbf{i}$ denote pseudo-labeled images and $\mathbf{I}_{GT}$ is from the set of k images with ground truth labels. We further apply data augmentation via color jitter and horizontal flipping.

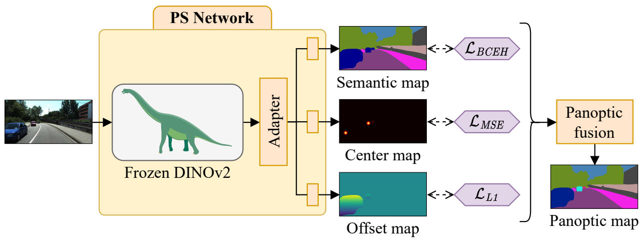

Fig. 4. To enable online predictions and to further boost the performance compared to the pseudo-label generator, we train a bottom-up panoptic segmentation model using our generated pseudo-labels. The network consists of a frozen DINOv2 [11] backbone with an adapter [35] and three task-specific heads, whose output is merged by a panoptic fusion module [2].

During test-time, a panoptic fusion module [2] predicts the final panoptic segmentation map from the output of the individual heads, shown in Fig. 4. In detail, it assigns a semantic label to the class-agnostic instance predictions using majority voting over the semantic predictions of all pixels within an instance.

IV. Experimental Evaluation

In this section, we demonstrate that our proposed SPINO outperforms unsupervised methods for semantic segmentation and yields competitive results compared to fully supervised setups for panoptic segmentation that require a huge number of ground truth annotations. We provide both quantitative and qualitative results on multiple public and in-house datasets. Finally, we extensively evaluate several design choices for our pseudo-label generator.

A. Datasets

We present results on various datasets including the public Cityscapes [4] and KITTI-360 [15] as well as our in-house data for automated driving and from an indoor office environment.

Cityscapes: The Cityscapes dataset [4] contains RGB images and fine panoptic annotations for automated driving in 50 cities across Germany and bordering regions. We select k images from the *train* split to train our label generator and generate pseudo-labels for the remaining images. In a separate experiment, we also generate pseudo-labels on the entire *train_extra* split. To evaluate the performance, we report metrics on the *val* split. When creating the pseudo-labels, we mask out the hood of the ego car as it remains static and hence can be inferred from the k annotated images [5]. We report metrics using 19 classes as per the official Cityscapes evaluation protocol.

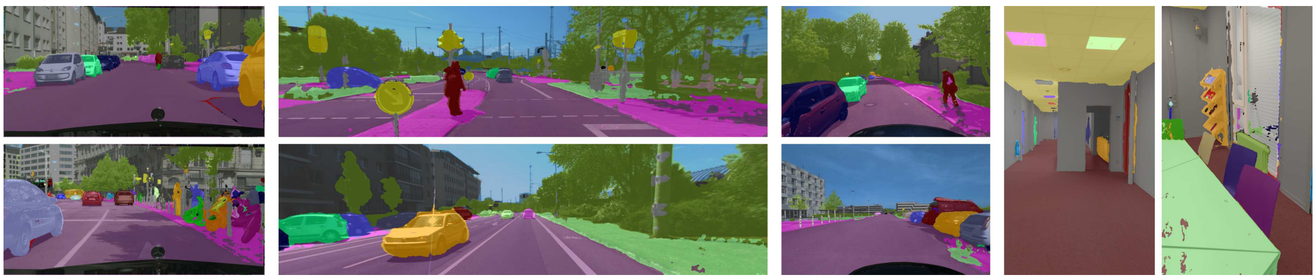

Fig. 5. Qualitative performance of our pseudo-label generator in four diverse domains from both public and in-house data sources. From left to right, we show Cityscapes [4], KITTI-360 [15], in-house automated driving, and an in-house office environment.

TABLE I
PANOPTIC/SEMANTIC SEGMENTATION ON CITYSCAPES

Method	**Train. data**	Acc	mIoU	PQ	SQ	RQ
Fully supervised						
DINOv2 + Adapt. + PH	GT	91.9	77.0	51.4	78.9	63.1
Unsupervised						
Modified DC [38]	n/a	35.3	6.8	–	–	–
PiCIE [14]	n/a	72.7	13.8	–	–	–
STEGO [12]	n/a	89.1	38.0	–	–	–
Few-shot supervision						
ResNet-50 + PH	10 GT	74.9	32.1	16.8	45.6	20.8
DINOv2 + PH	10 GT	81.6	49.4	20.6	49.9	25.8
DINOv2 + Adapt. + PH	10 GT	82.8	52.5	22.0	60.9	27.0
Pseudo-labels (*ours*)	10 GT	86.0	61.5	35.9	73.7	45.9
SPINO (*ours*)	PL	86.3	60.6	36.4	73.5	46.7
+ Mixed-batch	PL	86.6	61.2	36.5	74.8	46.3
SPINO (*ours*)	PL++	86.6	61.8	37.2	74.5	47.5

PH refers to the panoptic heads as shown in Fig. 4. GT and PL indicate training with ground truth annotations and pseudo-labels, where the "PL++" marks pseudo-labels on the *train_extra* split. The architecture of SPINO corresponds to "DINOv2 + Adapt. + PH".

KITTI-360: The KITTI-360 dataset [15] was recorded in Karlsruhe, Germany, and provides RGB images and panoptic annotations for sequential data. Following prior works [36], [37], we use sequence 10 for evaluation and the remaining sequences for the pseudo-label generation. We report results using 14 classes as detailed by Vödisch *et al.* [21].

In-House: To illustrate the main benefit of SPINO, i.e., enabling panoptic segmentation on different vision system with very few reference annotations, we employ our method on two in-house data sources. First, following the spirit of the public datasets, we use an automated driving perception car navigating in Freiburg, Germany. Second, to demonstrate general applicability, we record indoor data in our office environment. For both domains, we prepare annotations for ten images to train the pseudo-label generator.

B. Panoptic Segmentation

To evaluate the performance of SPINO, we measure the pixel accuracy (Acc) and the mean IoU (mIoU) for semantic segmentation as well as the panoptic quality (PQ), the segmentation quality (SQ), and the recognition quality (RQ) for panoptic segmentation. Based on the ablation studies in Sec. IV-C, we train our pseudo-label generator on $k = 10$ human-selected, labeled images with a batch size $b = 1$ and a learning rate $lr = 0.001$.

Few-Shot Training: First, we illustrate the efficacy of our pseudo-label generation scheme. As shown by the metrics in Tab. I, training Panoptic-DeepLab [2] (with a ResNet-50 backbone) on only ten images yields poor results that can be improved by replacing the backbone with a frozen DINOv2 [11]. Following the common methodology for dense prediction tasks, we also add an adapter module [35] to further increase the performance. However, the results remain significantly inferior to the quality of our pseudo-labels with respect to both semantic and panoptic segmentation. Notably, our pseudo-label generator comprises a much simpler design, e.g., estimating object boundaries instead of predicting instance centers and pixel offsets. For the overall SPINO approach, we adopt the network design of DINOv2 plus an adapter module. Naive training on the generated pseudo-labels already yields highly competitive results compared to training with ground truth labels considering that we use less than $0.29\,\%$ of the labels. We further show how the proposed mixed-batch strategy that closely incorporates the ten ground truth labels increases all three semantic metrics.

Next, we also generate pseudo-labels for the unlabeled *train_extra* split of Cityscapes, increasing the amount of training data for the panoptic segmentation model. The results in Tab. I indicate that our approach opens up an avenue for exploiting unlabeled large-scale data recordings for the training of existing panoptic segmentation methods.

Comparison with Unsupervised Segmentation: Second, we compare SPINO to the state-of-the-art for unsupervised semantic segmentation. As we follow the official Cityscapes evaluation protocol, we retrain PiCIE [14] and their modified DeepCluster [14], [38] using the released code on 19 classes. For STEGO [12], we use the provided network weights but reevaluate on 19 classes. Note that, for both PiCIE and STEGO, reducing the number of classes leads to higher metrics than reported by the authors. As SPINO significantly outperforms these baselines, we argue that requiring ten instead of zero annotated images is well justified.

Generalizability: Finally, we extend the evaluation to multiple datasets. In Tab. II, we report quantitative results on both Cityscapes [4] and KITTI-360 [15]. In detail, we compare supervised training with ground truth annotations to our few-shot approach. Similar to Tab. I, we report results for three backbones, namely ResNet-50 [39], DINOv2 [11], and

TABLE II

PANOPTIC SEGMENTATION ON CITYSCAPES AND KITTI-360

Method	Train. data	Cityscapes Acc	mIoU	PQ	SQ	RQ	KITTI-360 Acc	mIoU	PQ	SQ	RQ
Pseudo-labels	10 GT	86.0	61.5	35.9	73.7	45.9	75.8	54.7	32.5	70.7	42.1
ResNet-50 + PH	GT	89.4	64.9	44.2	75.3	56.1	83.0	64.1	41.0	76.5	50.5
DINOv2 + PH	GT	89.4	71.4	41.0	74.4	51.7	83.5	62.8	39.3	70.5	48.7
DINOv2 + Adapt. + PH	GT	91.9	77.0	51.4	78.9	63.1	86.0	65.6	42.5	72.9	51.2
ResNet-50 + PH	PL	85.4	57.3	33.0	67.8	42.3	76.2	52.1	32.2	67.6	41.0
DINOv2 + PH	PL	84.5	57.1	31.4	70.9	40.3	76.4	54.6	32.7	71.7	42.0
DINOv2 + Adapt. + PH	PL	86.3	60.6	36.4	73.5	46.7	76.6	55.5	33.3	71.9	42.8

PH refers to the panoptic heads shown in Fig. 4. GT and PL indicate ground truth annotations and pseudo-labels. The gray row corresponds to SPINO without mixed-batch training.

TABLE III

ABLATION STUDY: NETWORK ARCHITECTURE

Method	A: k-NN	B: Lin. Layer	C: CNN	D: MLP	E: Upsamling	Acc	mIoU	PQ	SQ	RQ
A	✓					78.7	51.7	26.1	68.5	35.0
B		✓				84.3	60.0	32.6	71.3	42.5
B + E		✓			✓	84.3	60.0	33.6	71.7	43.8
C + E			✓		✓	82.9	55.1	29.7	70.9	38.4
D + E				✓	✓	**86.0**	**61.5**	**35.9**	**73.7**	**45.9**

Due to the high computational complexity, the k-NN is evaluated without training data augmentation.

TABLE IV

ABLATION STUDY: DATA AUGMENTATION

Method	Acc	mIoU	PQ	SQ	RQ
Base	83.2	55.8	29.5	70.8	38.0
Training time					
+ Random flip	83.3	56.1	29.5	70.8	38.0
+ Random crop	83.0	57.2	30.0	70.7	39.1
+ Color jitter	83.1	57.3	30.1	70.9	39.1
Test time					
+ Multi-scale ensemble	**86.0**	**61.5**	**35.9**	**73.7**	**45.9**

TABLE V

ABLATION STUDY: BATCH SIZE

Batch size	Acc	mIoU	PQ	SQ	RQ
1	**86.0**	**61.5**	**35.9**	**73.7**	**45.9**
2	84.9	59.8	34.0	72.6	43.7
4	85.3	59.4	33.6	72.3	43.3
8	84.5	56.8	31.3	71.4	39.9

TABLE VI

ABLATION STUDY: NUMBER OF LABELS

Label count	Acc	mIoU	PQ	SQ	RQ
1	69.8	37.1	19.8	55.4	27.2
3	81.8	49.3	30.3	64.3	38.8
5	82.8	55.0	32.1	65.5	41.3
10	86.0	61.5	35.9	73.7	45.9
25	88.5	66.9	39.6	74.9	50.1
50	89.4	69.1	40.9	74.8	51.6
100	90.3	71.3	42.9	76.3	53.8

DINOv2 with an adapter [35]. Considering that our pseudo-labels are generated based on only ten images, the few-shot methods yield impressive results across the board. Note that ten images correspond to $0.29\,\%$ and $0.02\,\%$ of the utilized ground truth labels for Cityscapes and KITTI-360, respectively. Finally, we provide qualitative visualizations of our pseudo-labels in Fig. 5 for both public datasets as well as our in-house data including outdoor urban and indoor office environments. Further examples are shown in the supplementary video on the project website.

C. Ablation Studies of Pseudo-Label Generation

We extensively evaluate the architectural design of our pseudo-label generator and demonstrate its efficacy in contrast to several alternatives. In Tabs. III, IV, V and VI, we highlight the utilized variant in gray.

Network Architecture: In Tab. III, we compare the architectural design of our pseudo-label generator using MLPs to other network architectures. Similar to other methods [10], [26], we use a k-NN classifier on the DINOv2 feature patches with $k = 5$. Due to the high computational complexity of this approach, we omit training data augmentation for the k-NN. Next, we utilize a linear layer with and without prior upsampling. Compared to the k-NN, these learnable methods yield a significant improvement but remain inferior to the MLPs. Finally, we demonstrate that our design also outperforms a 4-layer CNN with 3×3 convolutions.

Data Augmentation: Next, we gradually activate the data augmentation techniques and list the results in Tab. IV. Utilizing data augmentation during the training enhances mIoU, PQ, and RQ, whereas the accuracy and SQ remain stable. Additionally, our employed test-time augmentation based on multi-scale ensemble prediction vastly improves the metrics across the board.

Batch Size: In Tab. V, provide results for various batch sizes. Note that we scale the learning rate proportionally to the batch size and keep the number of epochs constant. Due to leading to the highest quality of the pseudo-labels, we select a batch size $b = 1$.

Number of Ground Truth Labels: Finally, we investigate the effect of the label count on the quality of the pseudo-labels. In Tab. VI, we report results for increasing k from one-shot to $k = 100$. Note that for up to $k = 10$, we manually select the samples used for training. For $k > 10$, we randomly add further data. We observe a continuous improvement for greater k. Notably, for $k = 100$, our pseudo-label generator is almost on par with Panoptic-DeepLab while using $2.9\,\%$ of the annoations (see ResNet-50 backbone in Tab. II).

V. CONCLUSION

In this work, we introduced SPINO for few-shot panoptic segmentation by exploiting descriptive image representations from the unsupervised foundation model DINOv2. We demonstrated that SPINO can generate high-qualitative pseudo-labels after being trained on as little as ten annotated images. These pseudo-labels can then be used to train any existing panoptic segmentation method yielding results that are highly competitive to fully supervised learning approaches relying on human annotations. Finally, we extensively evaluated several design choices for the proposed pseudo-label generator. To facilitate further research, we made our code publicly available. In future research, we will refine the boundary estimation and employ SPINO in additional domains.

References

[1] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 9396–9405.

[2] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 12 472–12 482.

[3] R. Mohan and A. Valada, "Perceiving the invisible: Proposal-free amodal panoptic segmentation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9302–9309, 2022.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[5] L.-C. Chen, R. G. Lopes, B. Cheng, M. D. Collins, E. D. Cubuk, B. Zoph, H. Adam, and J. Shlens, "Naive-Student: Leveraging semi-supervised learning in video sequences for urban scene segmentation," in *Europ. Conf. on Computer Vision*, 2020, pp. 695–714.

[6] C. Lang, A. Braun, L. Schillingmann, K. Haug, and A. Valada, "Self-supervised representation learning from temporal ordering of automated driving sequences," *arXiv preprint arXiv:2302.09043*, 2023.

[7] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Conf. on Robot Learning*, vol. 139, 2021, pp. 8748–8763.

[9] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[10] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Int. Conf. on Computer Vision*, 2021, pp. 9630–9640.

[11] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, *et al.*, "DINOv2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[12] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman, "Unsupervised semantic segmentation by distilling feature correspondences," in *Int. Conf. on Learning Representations*, 2022.

[13] X. Wang, R. Girdhar, S. X. Yu, and I. Misra, "Cut and learn for unsupervised object detection and instance segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 3124–3134.

[14] J. Hyun Cho, U. Mall, K. Bala, and B. Hariharan, "PiCIE: Unsupervised semantic segmentation using invariance and equivariance in clustering," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021, pp. 16 789–16 799.

[15] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3292–3310, 2023.

[16] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "UPSNet: A unified panoptic segmentation network," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 8818–8826.

[17] L. Porzi, S. R. Bulo, A. Colovic, and P. Kontschieder, "Seamless scene segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 8277–8286.

[18] R. Mohan and A. Valada, "EfficientPS: Efficient panoptic segmentation," *Int. Journal of Computer Vision*, vol. 129, pp. 1551 – 1579, 2020.

[19] H. Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen, "Axial-DeepLab: Stand-alone axial-attention for panoptic segmentation," in *Europ. Conf. on Computer Vision*, 2020, pp. 108–126.

[20] Z. T. Zheng Ding, Jieke Wang, "Open-vocabulary universal image segmentation with maskclip," in *Int. Conf. on Machine Learning*, 2023.

[21] N. Vödisch, K. Petek, W. Burgard, and A. Valada, "CoDEPS: Online continual learning for depth estimation and panoptic segmentation," *Robotics: Science and Systems*, 2023.

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

[23] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 9726–9735.

[24] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, and L. V. Gool, "Revisiting contrastive methods for unsupervised learning of visual representations," in *Advances in neural information processing systems*, vol. 34, 2021, pp. 16 238–16 250.

[25] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 15 979–15 988.

[26] O. Siméoni, G. Puy, H. V. Vo, S. Roburin, S. Gidaris, A. Bursuc, P. Pérez, R. Marlet, and J. Ponce, "Localizing objects with self-supervised transformers and no labels," *British Mac. Vision Conf.*, 2021.

[27] Y. Wang, X. Shen, Y. Yuan, Y. Du, M. Li, S. X. Hu, J. L. Crowley, and D. Vaufreydaz, "TokenCut: Segmenting objects in images and videos with self-supervised transformer and normalized cut," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 1–13, 2023.

[28] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[29] X. Wang, Z. Yu, S. De Mello, J. Kautz, A. Anandkumar, C. Shen, and J. M. Alvarez, "FreeSOLO: Learning to segment objects without annotations," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 14 156–14 166.

[30] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, and L. Van Gool, "Unsupervised semantic segmentation by contrasting object mask proposals," in *Int. Conf. on Computer Vision*, 2021, pp. 10 032–10 042.

[31] W. V. Gansbeke, S. Vandenhende, and L. V. Gool, "Discovering object masks with transformers for unsupervised semantic segmentation," *arXiv preprint arXiv:2206.06363*, 2022.

[32] Y. Lin, M. Chen, W. Wang, B. Wu, K. Li, B. Lin, H. Liu, and X. He, "CLIP is also an efficient segmenter: A text-driven approach for weakly supervised semantic segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 15 305–15 314.

[33] T. Chen, Z. Mai, R. Li, and W. lun Chao, "Segment anything model (sam) enhanced pseudo labels for weakly supervised semantic segmentation," *arXiv preprint arXiv:2305.05803*, 2023.

[34] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 3309–3318.

[35] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision transformer adapter for dense predictions," in *Int. Conf. on Learning Representations*, 2023.

[36] R. Mohan and A. Valada, "Amodal panoptic segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 20 991–21 000.

[37] N. Gosala, K. Petek, P. L. Drews-Jr, W. Burgard, and A. Valada, "SkyEye: Self-supervised bird's-eye-view semantic mapping using monocular frontal view images," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 14 901–14 910.

[38] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Europ. Conf. on Computer Vision*, 2018, pp. 132–149.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

7.3 Collaborative Dynamic 3D Scene Graphs for Automated Driving

The appended paper [7] follows.

Collaborative Dynamic 3D Scene Graphs for Automated Driving

Elias Greve[1*], Martin Büchner[1*], Niclas Vödisch[1*], Wolfram Burgard[2], and Abhinav Valada[1]

***Abstract*— Maps have played an indispensable role in enabling safe and automated driving. Although there have been many advances on different fronts ranging from SLAM to semantics, building an actionable hierarchical semantic representation of urban dynamic scenes from multiple agents is still a challenging problem. In this work, we present Collaborative URBan Scene Graphs (CURB-SG) that enable higher-order reasoning and efficient querying for many functions of automated driving. CURB-SG leverages panoptic LiDAR data from multiple agents to build large-scale maps using an effective graph-based collaborative SLAM approach that detects inter-agent loop closures. To semantically decompose the obtained 3D map, we build a lane graph from the paths of ego agents and their panoptic observations of other vehicles. Based on the connectivity of the lane graph, we segregate the environment into intersecting and non-intersecting road areas. Subsequently, we construct a multi-layered scene graph that includes lane information, the position of static landmarks and their assignment to certain map sections, other vehicles observed by the ego agents, and the pose graph from SLAM including 3D panoptic point clouds. We extensively evaluate CURB-SG in urban scenarios using a photorealistic simulator. We release our code at `http://curb.cs.uni-freiburg.de`.**

I. Introduction

Spatial and semantic understanding of the environment is crucial for the safe and autonomous navigation of mobile robots and self-driving cars. Recent autonomy systems leverage high-definition (HD) map information as effective priors for several downstream tasks in automated driving (AD) including perception [1], localization [2], planning [3], and control [4]. HD maps are often constructed and maintained in a top-down manner [5], i.e., relying on traffic authorities or via arduous labeling efforts. In contrast, automatic bottom-up AD mapping approaches show high accuracy [6], [7] while being limited to occupancy or semantic mapping using, e.g., dense voxel grid manifolds. With respect to AD, map representations should ideally fulfill the following requirements [8]: 1) completeness and accuracy while scaling to large areas; 2) frequent updates to capture structural changes; 3) higher-level topological information grounded in rich sensor data; 4) efficient access and information querying. Given these requirements, typical SLAM maps only enable classical spatial or point-level semantic querying. We envision that modern AD mapping approaches should provide the means to process vision and language queries, e.g., from foundation models [9]. Enabling such demands can only become feasible by abstracting from given maps using sparse representations.

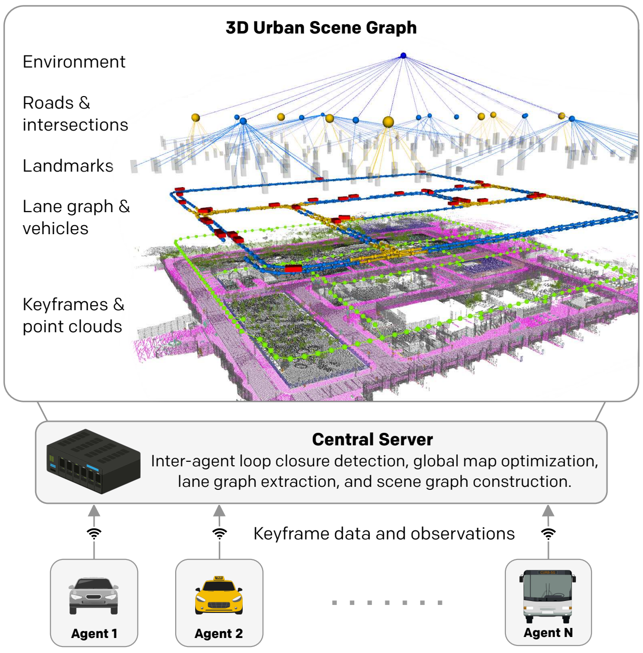
3D Urban Scene Graph
Environment
Roads & intersections
Landmarks
Lane graph & vehicles
Keyframes & point clouds
Central Server
Inter-agent loop closure detection, global map optimization, lane graph extraction, and scene graph construction.
Keyframe data and observations
Agent 1
Agent 2
Agent N

Fig. 1. For our proposed collaborative urban scene graphs (CURB-SG), multiple agents send keyframe packages with their local odometry estimates and panoptic LiDAR scans to a central server that performs global graph optimization. We subsequently partition the environment based on a lane graph from agent paths and other detected cars. Together with the 3D map, the lane graph forms the base of the large-scale hierarchical scene graph.

In this work, we propose *Collaborative URBan Scene Graphs* (CURB-SG) that effectively address the aforementioned requirements by constructing a hierarchical graph structure of the environment as shown in Fig. 1. 3D scene graphs enable efficient data storage of large environments while being queryable and preserving spatial information. Previous works on 3D scene graphs [10]–[12] focus on indoor environments, whose taxonomy cannot be directly transferred to large-scale urban domains. To close this gap, we introduce the following analogy to indoor variants: Cities (buildings) can be separated into intersections and roads (rooms), which contain static landmarks such as traffic signs (furniture) as well as dynamic objects such as vehicles (humans). We enable this partitioning by generating an online lane graph that serves as a common link among multiple graph layers. Addressing frequent updates and multi-agent cooperation, our method leverages a centralized collaborative SLAM approach that combines panoptic LiDAR data and local odometry estimates into a single 3D map while optimizing a global pose graph that

[*] Equal contribution.
[1] Department of Computer Science, University of Freiburg, Germany.
[2] Department of Eng., University of Technology Nuremberg, Germany.
The supplementary material can be found at https://arxiv.org/abs/2309.06635.
This work was funded by the European Union's Horizon 2020 research and innovation program grant No 871449-OpenDR and the German Research Foundation (DFG) Emmy Noether Program grant No 468878300.

benefits from inter-agent loop closures. Following the spirit of previous works on scene graphs [10]–[12], we extensively evaluate our proposed method on simulated data using the CARLA simulator [13].

To summarize, the main contributions are as follows:

1) We introduce a novel algorithm for representing urban driving environments as dynamic 3D scene graphs that are constructed from multi-agent observations to efficiently cover large areas.
2) We demonstrate an effective partitioning of urban environments using lane graphs constructed on the fly from panoptic LiDAR observations in a cooperative manner.
3) We present an efficient collaborative graph SLAM method to continuously update semantic maps while addressing scalability via edge contraction.
4) We provide extensive evaluations of the building blocks of our proposed framework.
5) We make our code and sample data publicly available at `http://curb.cs.uni-freiburg.de`.

II. Related Work

In this section, we first present a summary of LiDAR-based odometry and mapping, followed by an overview of multi-agent SLAM, and scene graphs in automated driving (AD).

LiDAR SLAM: LiDAR-based mapping has been pioneered by LOAM [14] that estimates robot motion from scan registration via ICP between subsequent point clouds. To address the full SLAM problem, HDL Graph SLAM [7] combines LiDAR odometry with local loop closure detection and performs joint pose graph optimization. Leveraging semantic segmentation, SUMA++ [6] masks dynamic classes during the mapping stage and proposes a semantic-aided variant of ICP. PADLoC [15] exploits panoptic segmentation during training to stabilize both loop closure detection and registration. In this work, we use panoptic point clouds to generate a large-scale semantic 3D map forming the base layer of our scene graph.

Collaborative SLAM: To cover large environments and to increase mapping speed, SLAM research begins to shift towards multi-agent methods [16]. Generally, collaborative SLAM can be realized in a centralized or distributed manner. Initial works such as C^{2}TAM [17] belong to the centralized category, performing global bundle adjustment on a server and localization on the clients. A similar paradigm is adopted by CVI-SLAM [18] and COVINS [19], proposing visual-inertial (VI) SLAM systems for a fleet of UAVs. While the robots run local VI odometry, a central server collects this information, searches for inter-agent loop closures to perform global optimization, and removes redundant data. With respect to LiDAR SLAM, LAMP 2.0 [20] allows collaboration between different types of robots to map large-scale underground environments. A similar use case is addressed by Swarm-SLAM [21], which supports further sensor modalities. Following a distributed paradigm, information is directly shared between the agents using peer-to-peer communication. Kimera-Multi [22] is a VI SLAM method that includes semantic information in the generated 3D mesh. For data fusion, it employs distributed pose graph optimization (PGO). Finally, DisCo-SLAM [23] proposes a LiDAR-based approach addressing the initially unknown relative position of the agents. For this, they use Scan Context [24] descriptors for global loop closure detection without spatial priors. In this work, we follow the centralized paradigm since we leverage collaborative SLAM to generate a single consistent scene graph that can be made available to other traffic participants to query information.

Scene Graphs for Automated Driving: 3D scene graphs constitute an effective interface unifying pose graphs from large-scale mapping and local information [25] such as frame-wise object detections [26], topological mapping [27]–[29], or semantic segmentation [30], [31]. Additionally, graphs enable the structural disassembly of large-scale scenes into objects and their relationships and facilitate higher-level reasoning, e.g., in the vision and language domain [32]. This further allows for efficient hierarchical abstraction in both spatial and semantic regimes [12], [33]. So far, 3D scene graphs for environment representation have only been applied in indoor domains. The first work in this field [12] proposes an offline, multi-layered hierarchical representation based on RGB images. Kim *et al.* [34] were the first to generate 3D scene graphs from RGB-D images for visual question answering [35] and task planning. Using a learning-based pipeline, Wald *et al.* [36] construct a 3D scene graph from an instance-segmented point cloud while predicting node and edge semantics in an offline manner. Rosinol *et al.* [33] present an offline framework capable of generating hierarchical scene graphs from dynamic indoor scenes that are divided into buildings, rooms, places, objects, and agents, as well as a metric-semantic mesh. Different from the aforementioned frameworks, Hydra [11], SceneGraphFusion [37], and S-Graphs [25] present real time-capable approaches. While Hydra does not tightly couple the optimized pose graph with the 3D scene graph, the non-hierarchical S-Graphs [25] close this gap. The follow-up work S-Graphs+ [10] also encodes hierarchies. In this work, we combine collaborative SLAM and 3D scene graphs to build hierarchical maps for AD. To the best of our knowledge, our work constitutes the first approach to 3D scene graph construction of urban driving scenes with a tightly coupled integration of inter-agent loop closures. Furthermore, we show how multi-agent cooperation facilitates frequent map updates and completeness.

III. Technical Approach

In this section, we present our CURB-SG approach for collaborative urban scene graphs. As illustrated in Fig. 2, CURB-SG is comprised of several components. In Sec. III-A, we describe our approach for collaborative SLAM to effectively combine panoptic information. Here, multiple agents transmit their onboard LiDAR odometry estimates along with panoptic point clouds to a central compute unit. This server combines the data by detecting intra- and inter-agent loop closures and performs pose graph optimization (PGO) to generate a globally consistent 3D map. In Sec. III-

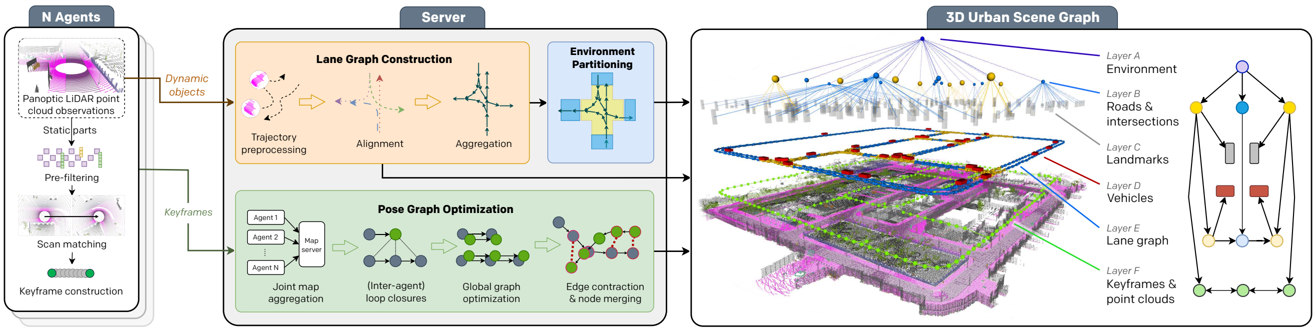

Fig. 2. Overview of CURB-SG: Multiple agents obtain panoptically segmented LiDAR data and provide an odometry estimate based on the static parts of the point cloud. A centralized server instance then performs pose graph optimization (PGO) including inter-agent loop closure detection and edge contraction based on the agents' inputs. Tightly coupled to the pose graph, we aggregate a lane graph from panoptic observations of other vehicles as well as the agent's trajectories. Next, the lane graph is partitioned to retrieve a topological separation that allows for the hierarchical abstraction of larger environments.

B, we propose to further aggregate the paths of the agents and other observed vehicles to extract an online lane graph allowing for partitioning the city into intersections and roads. Finally, the server registers dynamic traffic participants on the lane graph and generates a hierarchical scene graph by assigning static landmarks to the closest intersection or road.

A. Collaborative SLAM

We leverage collaborative LiDAR SLAM as the backend in our proposed CURB-SG. Due to its reliable performance and well-maintained code base, we build on top of HDL Graph SLAM [7] and extend it to a multi-agent scenario following a centralized approach as described in Sec. II. In this section, we describe the steps performed by each agent, followed by the centralized PGO as depicted in Fig. 2. Finally, we provide further details on how CURB-SG explicitly addresses both long-term and large-scale mapping.

Agents: Each agent is equipped with a LiDAR sensor to capture sparse 3D point clouds, which contain spatial information as well as point-wise panoptic segmentation labels. Initially, a point cloud is separated into its static and dynamic components following the conventional categorization of "stuff" and "thing" classes [38]. Similar to SUMA++ [6], we use only the static points for constructing the map. In contrast to HDL Graph SLAM [7], we utilize different voxel grid sizes for the various semantic classes. This approach retains more dense information where required, e.g., poles and traffic signs are being processed at a more fine-grained level than roads or buildings. Next, we perform point cloud registration via FAST-GICP [39] between subsequent LiDAR scans to estimate the motion of an agent. Following the common methodology and to reduce the required bandwidth between the agents and the server, we generate keyframes after a specified traveled distance based on LiDAR odometry. Each keyframe is sent to the server and contains an estimated pose and the static LiDAR point cloud with semantic labels, i.e., the "stuff" points. Since car instances contribute to the online construction of a lane graph (see Sec. III-B), the "thing" points from all the LiDAR scans are transformed relative to the pose of the previous keyframe and sent separately.

Server: The centralized server receives keyframes from all the agents and processes them in the following manner: First, upon receiving the first keyframe sent by an agent, the server registers this agent to the global pose graph. Second, the server searches for loop closure candidates between the added keyframe and the existing nodes in the pose graph to find both intra- and inter-agent loop closures. We rely on the original loop closure detection technique of HDL Graph SLAM [7], i.e., all nodes within a local search radius are considered to be candidates. If the fitness score of the ICP algorithm is below a threshold, a loop closure edge is added to the pose graph. Due to relying on an initial guess, we utilize the absolute ground truth value for the registration of a new agent. In practice, this could either be solved with GNSS measurements or by conducting an efficient global search for loop closure candidates leveraging point cloud descriptors [23]. Third, the server performs PGO using g^2o [40] to integrate the newly added keyframes and detected loop closures. To address scalability, we employ edge contraction as detailed in the following paragraph. Finally, we apply the same semantics-based voxelization to the entire 3D map as performed by the agents on their local LiDAR scans.

Long-Term and Large-Scale Mapping: If not handled explicitly, the pose graph would continue to grow while the mapping progresses. Since every keyframe contains a 3D point cloud, this not only significantly slows down the PGO but also increases memory consumption and disk storage. To address both problems, we remove the nodes and edges from the graph that carry redundant information. In Fig. 3, two agents have driven along the same road yielding multiple loop closures. Using a heuristic-driven approach, the loop closure edges that carry redundant information are being contracted by merging nodes. By redirecting the edges of the omitted to the remaining node, we ensure the legal connectivity of the pose graph. Notably, this is done after the PGO step. Consequently, the final pose graph becomes easier to maintain and more efficient to query when searching for new loop closures. The point cloud data associated with a removed node is combined with the data of the persisting node while omitting older data to guarantee up-to-date map information. In contrast, the dynamic observations linked to a node are completely

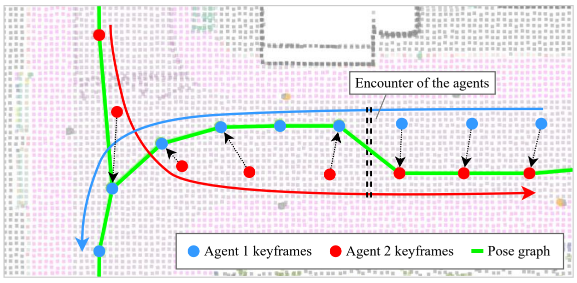

Fig. 3. In this example, two agents drive along the same road while passing each other at the dashed line. The detected loop closures yield additional edges in the pose graph. After optimization, the edges that carry redundant information are contracted by merging the older node into the more recently added node to update the map information.

transferred as they contribute towards the construction of the lane graph explained in Sec. III-B. For the same reason, each removed node is turned into a passive observation that stores the driven path of an ego agent.

B. Scene Graph Generation

The second key component of CURB-SG is a scalable environment representation of urban outdoor scenes for AD. Besides the aforementioned 3D semantic map, CURB-SG constructs a tightly-coupled hierarchical abstraction of the environment as shown in Fig. 2. By analogy with the separation of indoor scenes into buildings, rooms, and places [11], [12], [33], we decompose a constructed lane graph into intersecting and non-intersecting road areas allowing for spatial and semantic abstraction.

The root of our CURB-SG representation is given by *Layer A* that holds environment/city-level information. This environment is then spatially divided into intersections and their connecting roads (*Layer B*), which serve as the categorical counterparts to rooms and corridors in indoor scenes. Since the partitioning of our environment is based on a lane graph (presented in *Layer E*), the connectivity of *Layer B* is implicitly given by the connectivity of the lane graph (colored segments, Fig. 2). Next, we map static landmarks such as traffic signs and poles contained in *Layer C* including their bounding box to their corresponding spatial area defined by *Layer B*. These landmarks can serve as priors for localization or object detection. *Layer D* holds all currently observed dynamic vehicles. We map dynamic vehicles to their closest respective lane graph node, as defined in *Layer E*, to provide efficient access for downstream tasks, e.g., trajectory prediction. Central to this approach, *Layer E* is a directed lane graph to encode the low-level topology for vehicle navigation and is inferred from the paths of the ego agents as well as other perceived vehicles. We provide further details in the next paragraph. The lane graph defines the connectivity of the different spatial regions in the urban environment, comparable to edges among rooms in indoor scene graph variants. Finally, *Layer F* contains the pose graph from our SLAM backend and encodes LiDAR data in the form of semantic point clouds. As discussed in Sec. III-A, this layer is subject to continuous optimization and dynamic restructuring, e.g., due to loop closure detection and edge contraction. Based on the edges between the keyframes in this layer and spatial areas (*Layer B*), 3D map information is easily accessible given a rough road-level position estimate.

Lane Graph Generation: We generate a lane graph of the environment leveraging the trajectories of the ego agents as well as observations of surrounding vehicles. As the LiDAR point clouds of the agents contain instance IDs, we are able to differentiate between multiple observed vehicle instances in the agents' surroundings. For each observed vehicle, we extract the centroid of its partial point cloud. The position of a centroid is stored relative to the most recent keyframe. After transmitting the data to the server, the position of this dynamic observation can be retrieved given the link to its corresponding keyframe. Consequently, the positions of all the dynamic observations benefit from continuous keyframe updates due to PGO as depicted in Fig. 2. To evenly sample paths, we further filter the observations using both hand-crafted heuristics and DBSCAN [41] based on timestamps, angles, and relative displacements. This is particularly important for stationary and occluded objects as well as outliers caused by odometry noise. Following an iterative yaw-respective aggregation scheme [27], we convert all trajectories into directed graphs, apply Laplacian smoothing, and merge them to build a complete lane graph. Employing the same processing scheme, we add agent trajectories to this graph. Since CURB-SG maintains a connection between the lane graph and the keyframes used in SLAM, we can continuously propagate refinements from PGO to the lane graph.

Spatial Partitioning: Urban outdoor driving scenes exhibit a vastly different topology compared to indoor environments that have been represented using scene graphs so far. We found that classical methods such as wall dilation for retrieving disjoint environment graphs [11] are not directly applicable to urban environments. In our work, we propose to separate outdoor environments into intersecting and non-intersecting areas using the obtained lane graph (see above). Ultimately, this gives rise to the hierarchical environment abstraction introduced in CURB-SG enabling efficient querying for downstream tasks such as trajectory prediction. In particular, we detect intersections based on the following heuristics: First, we cluster high-degree lane graph nodes to find agglomerations of graph splits and merges. Second, we detect lane graph edges that intersect. These two approaches can be applied to various environments to efficiently handle challenging conditions such as multi-lane roads or non-trivial intersections. After identifying intersection nodes, the remaining disconnected sub-graphs fall into non-intersecting road areas. To assign components from other layers of the scene graph to the extracted partitions, we extend these areas beyond the lane node surroundings as illustrated in Fig. 2.

IV. Experimental Evaluation

In this section, we evaluate CURB-SG with respect to the collaborative SLAM backend, the constructed lane graph, and the proposed partitioning based on road intersections.

A. Experimental Setup

We evaluate CURB-SG on various urban driving scenarios using the CARLA simulator [13] due to a lack of real-world multi-agent datasets providing LiDAR scans. In particular, we perform experiments on a set of four diverse environments including *town01*, *town02*, *town07*, and *town10*. Following previous works [11], we use the panoptic annotations with temporally consistent instance IDs provided by the simulator. Where applicable, we demonstrate the efficacy of CURB-SG for one, two, and three agents and average results over ten randomly initialized runs. Due to the semantics-based voxelization on the server, the total number of map points of a fully explored town is relatively stable. As the path planning of the agents is randomized, it can take a long time until this number is reached. Therefore, we approximate full exploration by using 85 % as the termination criterion.

B. Collaborative SLAM

In this section, we evaluate the collaborative SLAM backend of our proposed CURB-SG with respect to both accuracy and cooperative gain in long-term scenarios.

Mapping and Localization: In Tab. I, we present the root mean squared errors (RMSE) of the agents' keyframes and the estimated position of the street signs to represent localization and mapping accuracy, respectively. We compute the position of a street sign as the geometric center of the corresponding bounding box that is inferred from the 3D map. We observe that both errors are reduced when more agents contribute towards the collaborative pose graph. Except for the case of two agents in *town07*, this holds true for the mean as well as the standard deviation across all environments. We further illustrate the robustness of our approach against noisy sensor data by imposing realistic metric Gaussian noise $\mathcal{N}(0, 0.02)$ on the LiDAR scans [42] of the agents in *town01* and *town02*. As shown in Tab. I, the noise does not significantly alter the errors indicating that downstream tasks such as lane graph estimation do not degrade either.

Long-Term Mapping: We demonstrate the efficacy of our proposed adaptions of HDL Graph SLAM [7] (see Sec. III-A) to address long-term mapping of large areas. In the rightmost column of Tab. I, we report the time required to map a town when using one, two, or three agents. Generally, the higher the number of contributing agents, the smaller the time required to explore the map. Similarly, in Fig. 4, we illustrate the mapping progress measured by the number of 3D points versus the simulation steps. While the results confirm the aforementioned general trend towards faster exploration in a multi-agent setup, the pure mapping speed will reach an upper bound above that additional agents will not further increase the speed. However, even afterward, these agents will keep sending measurements and vehicle observations contributing towards frequent map updates and enhancing the lane graph (see Sec. IV-C). We present further results for *town01* and *town10* in the suppl. material Sec. S.3.

Finally, we demonstrate that our proposed edge contraction successfully limits the number of nodes contained in the pose graph. In Fig. 5, we show the example of three agents operating in *town02* and compare the number of optimizable graph nodes with the total number of keyframes sent by the agents. We observe that without edge contraction, the pose graph continuously grows with the number of keyframes sent, rendering frequent optimization infeasible.

TABLE I
EVALUATION OF LOCALIZATION AND MAPPING PERFORMANCE

Environment	Agent count	RMSE (agents) [m]	RMSE (street signs) [m]	Exploration time [sim. steps]
town01	1	0.735 ± 0.492	0.865 ± 0.485	2502.50
	2	0.368 ± 0.343	0.480 ± 0.358	1267.70
	3	**0.132** ± **0.096**	**0.169** ± **0.096**	**1134.40**
+ *noise*	3	0.159 ± 0.093	0.225 ± 0.101	–
town02	1	0.306 ± 0.208	0.297 ± 0.194	2079.00
	2	0.249 ± 0.176	0.299 ± 0.238	943.80
	3	**0.126** ± **0.109**	**0.164** ± **0.148**	**597.60**
+ *noise*	3	0.119 ± 0.078	0.140 ± 0.064	–
town07	1	0.564 ± 0.406	–	3632.70
	2	0.234 ± **0.182**	–	1760.20
	3	**0.218** ± 0.198	–	**910.00**
town10	1	0.333 ± 0.185	–	923.70
	2	0.310 ± 0.185	–	724.00
	3	**0.116** ± **0.106**	–	**391.10**

Mean and standard deviation over ten runs of the RMSE of the agents' keyframes and the estimated position of the street signs representing localization and mapping accuracy, respectively. Note that the environments *town07* and *town10* do not contain street signs. The rightmost column lists the mean time required to map 85 % of the entire town measured in simulation steps.

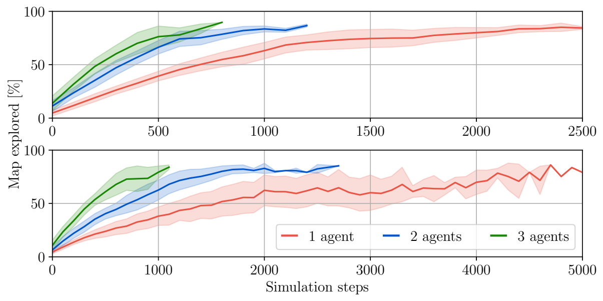

Fig. 4. The mapping progress in *town02* (top) and *town07* (bottom) for one, two, and three agents. Our collaborative SLAM method benefits from receiving inputs from multiple agents.

C. Lane Graph

We evaluate our proposed online lane graph generation approach from the paths of the ego agents and their observations of other vehicles (Sec. III-B). We present qualitative results in Fig. 6 for two scenarios simulated in *town02* with 30 additional non-agent vehicles: the left figure visualizes the lane graph in a single-agent scenario terminated as soon as the agent starts to repeatedly revisit intersections. Although the path of the agent, shown in blue, does not cover all the lanes, including the paths of the observed vehicles allows for a substantial extension of the lane graph. The right figure depicts a long-term scenario with three agents demonstrating that collaboration further boosts performance. Our method yields an almost complete lane graph even though several lanes have only been driven by the agents in the opposite direction.

We quantify these findings in Tab. II following previous

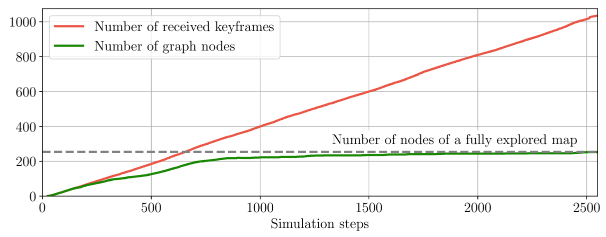

Fig. 5. Our proposed edge contraction mechanism effectively reduces the number of nodes in the pose graph to maintain the capability of frequent graph optimization. This plot shows three agents operating in *town02*.

TABLE II
LANE GRAPH EVALUATION

Ego	Obs.	TOPO P / R	GEO P / R	APLS	$SDA_{4.5}$	$SDA_{9.0}$	Graph IoU
1-agent scenario							
✓		**0.810** / 0.281	**0.923** / 0.415	0.658	0.000	0.042	0.386
✓	✓	0.678 / **0.562**	0.855 / **0.812**	**0.724**	**0.278**	**0.394**	**0.690**
3-agents scenario							
✓		**0.715** / 0.583	**0.874** / 0.762	**0.800**	0.188	0.357	0.658
✓	✓	0.574 / **0.712**	0.751 / **0.925**	0.756	**0.250**	**0.452**	**0.751**

Quantitative results obtained in *town02*. The two left columns indicate whether only the paths of the ego agents or also the estimated positions of other observed vehicles have been used. For the TOPO and GEO metrics, we provide both precision (P) and recall (R).

works on lane graphs: precision and recall of the TOPO and GEO metrics [29], APLS [43], SDA_R [27] with the subscript denoting the search radius in meters, and the graph IoU [27]. For more details, please refer to the respective reference. We observe that except for the TOPO/GEO precision and the APLS in the 3-agent scenario, all the metrics show an improvement when using not only the paths of the ego agents but also of the observed vehicles. We attribute the decrease in precision to the noise in the estimated position of the other vehicles. Since we approximate the center of a vehicle by the geometric mean of the respective 3D points, there is a bias towards the center line of a road for all oncoming cars. We further observe that increasing the number of agents does have a positive impact on all the metrics except for the TOPO/GEO precision and the $SDA_{4.5}$ demonstrating the efficacy of our method.

D. Environment Partitioning

We evaluate our approach for environment partitioning (Sec. III-B) by comparing it against the ground-truth intersection points of the underlying map. Throughout exploring the environment, the recall is normalized using the point cloud of the road surface obtained thus far. Our proposed lane graph-based method (LG) is compared against a morphological image skeletonization baseline (SK) that uses medial axes of the bird's-eye-view projected point cloud of the road surface. Kernelized smoothing and dilation followed by thresholding the obtained bird's-eye-view image helps in filtering false positive points and noise. In order to further increase precision, the SK baseline includes clustering culmination of intersection points in local areas that originate from artifacts in the skeleton graph. We report the precision and recall values across ten exploration runs on *town02* in Fig. 7. We observe

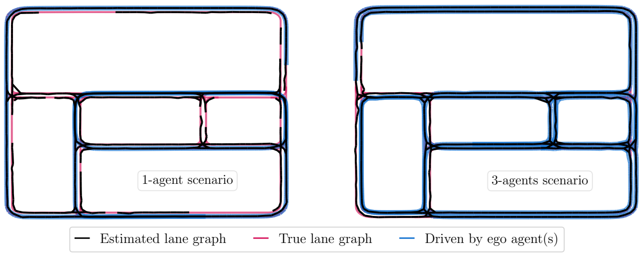

Fig. 6. Visualization of the constructed lane graph of *town02* when using one or three agents. Lanes marked in blue have been traversed by an ego agent. Others are reconstructed from observing surrounding vehicles.

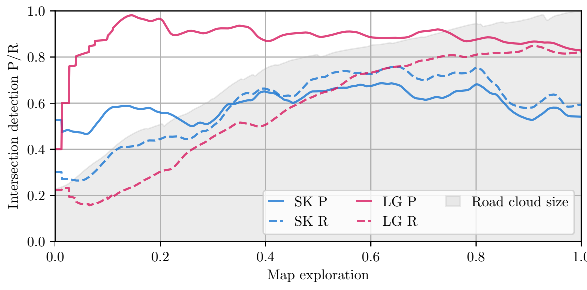

Fig. 7. Intersection detection quality of our lane graph-based detection of intersections (LG) and an image-based skeletonization baseline of the road surface (SK). Average precision (P) and recall (R) of both approaches across 10 runs with 3 agents and 40 vehicles on *town02* as well as the size of the investigated road surface point cloud are shown.

that our approach (LG) achieves at least 20 % greater precision while showing comparable or exceeding recall scores. As our approach relies on observed vehicle trajectories, we attribute the lower initial recall of the LG method to a small number of initially seen trajectories while the point cloud-based baseline already processes a larger extent of the surroundings at this stage. Nonetheless, we observe that the SK baseline yields vastly different partitioning solutions throughout exploration as it is not robust to artifacts such as occlusions due to vehicles or sparse LiDAR readings of distant road surfaces. We believe that a conservative, high-precision classifier is beneficial as over-segmentation increases the number of roads and intersections unnecessarily. Further explanations are provided in suppl. material Sec. S.4. Additionally, we observe that simply extracting intersections from the pose graph produces low recalls as every path has to be traversed by the agents instead of relying on more descriptive observations.

V. Conclusion

In this work, we introduced CURB-SG as a novel approach to building large-scale hierarchical dynamic 3D urban scene graphs from multi-agent observations. We furthermore demonstrated how our collaborative SLAM approach facilitates frequent map updates and rapid exploration while scaling to large environments. To foster further research in this direction, we made our code publicly available. In future work, we will address the reliance on simulated panoptic labels and known initial poses of the agents. Orthogonal to that, follow-up work could address a decentralized variant that operates under real-time constraints. Furthermore, we plan to include pedestrian information as well as additional topological elements such as road boundaries.

References

[1] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Conf. on Robot Learning*, 2018.

[2] D. Cattaneo, D. G. Sorrenti, and A. Valada, "CMRNet++: Map and camera agnostic monocular visual localization in LiDAR maps," *Int. Conf. on Robotics and Automation Workshop on Emerging Learning and Alg. Methods for Data Association in Robotics*, 2020.

[3] A. Diaz-Diaz, M. Ocaña, A. Llamazares, C. Gómez-Huélamo, P. Revenga, and L. M. Bergasa, "HD maps: Exploiting opendrive potential for path planning and map monitoring," in *IEEE Intelligent Vehicles Symposium*, 2022, pp. 1211–1217.

[4] R. Trumpp, M. Büchner, A. Valada, and M. Caccamo, "Efficient learning of urban driving policies using bird's-eye-view state representations," *Int. Conf. on Intelligent Transportation Systems*, 2023.

[5] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *Int. Conf. on Intelligent Transportation Systems*, 2018.

[6] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based semantic SLAM," in *Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 4530–4537.

[7] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LiDAR-based system for long-term and wide-area people behavior measurement," *Int. Journal of Adv. Rob. Systems*, vol. 16, no. 2, 2019.

[8] K. Wong, Y. Gu, and S. Kamijo, "Mapping for autonomous driving: Opportunities and challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 1, pp. 91–106, 2020.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *Int. Conf. on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[10] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "S-Graphs+: Real-time localization and mapping leveraging hierarchical representations," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4927–4934, 2023.

[11] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *Robotics: Science and Systems*, 2022.

[12] I. Armeni, Z.-Y. He, A. Zamir, J. Gwak, J. Malik, M. Fischer, and S. Savarese, "3D scene graph: A structure for unified semantics, 3D space, and camera," in *Int. Conf. on Computer Vision*, 2019, pp. 5663–5672.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conf. on Robot Learning*, 2017.

[14] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, 2014.

[15] J. Arce, N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada, "PADLoC: LiDAR-based deep loop closure detection and registration using panoptic attention," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1319–1326, 2023.

[16] D. Zou, P. Tan, and W. Yu, "Collaborative visual SLAM for multiple agents: A brief survey," *Virtual Reality and Intelligent Hardware*, vol. 1, no. 5, pp. 461–482, 2019.

[17] L. Riazuelo, J. Civera, and J. Montiel, "C^2TAM: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014.

[18] M. Karrer, P. Schmuck, and M. Chli, "CVI-SLAM — collaborative visual-inertial SLAM," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2762–2769, 2018.

[19] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, "COVINS: Visual-inertial SLAM for centralized collaboration," in *IEEE Int. Symp. on Mixed and Augmented Reality Adjunct*, 2021, pp. 171–176.

[20] Y. Chang, K. Ebadi, C. E. Denniston, M. F. Ginting, A. Rosinol, A. Reinke, *et al.*, "LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9175–9182, 2022.

[21] P.-Y. Lajoie and G. Beltrame, "Swarm-SLAM: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems," *arXiv preprint arXiv:2301.06230*, 2023.

[22] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-Multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Trans. on Robotics*, vol. 38, no. 4, pp. 2022–2038, 2022.

[23] Y. Huang, T. Shan, F. Chen, and B. Englot, "DiSCo-SLAM: Distributed scan context-enabled multi-robot LiDAR SLAM with two-stage global-local graph optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1150–1157, 2022.

[24] G. Kim and A. Kim, "Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 4802–4809.

[25] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9107–9114, 2022.

[26] C. Lang, A. Braun, and A. Valada, "Robust object detection using knowledge graph embeddings," in *DAGM German Conference on Pattern Recognition*, 2022, pp. 445–461.

[27] M. Büchner, J. Zürn, I.-G. Todoran, A. Valada, and W. Burgard, "Learning and aggregating lane graphs for urban automated driving," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 13 415–13 424.

[28] Liao, Bencheng and Chen, Shaoyu and Wang, Xinggang and Cheng, Tianheng, and Zhang, Qian and Liu, Wenyu and Huang, Chang, "MapTR: Structured modeling and learning for online vectorized HD map construction," in *Int. Conf. on Learning Representations*, 2023.

[29] S. He and H. Balakrishnan, "Lane-level street map extraction from aerial imagery," in *IEEE Winter Conference on Applications of Computer Vision*, January 2022, pp. 2080–2089.

[30] N. Gosala, K. Petek, P. L. Drews-Jr, W. Burgard, and A. Valada, "Skyeye: Self-supervised bird's-eye-view semantic mapping using monocular frontal view images," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 14 901–14 910.

[31] R. Mohan and A. Valada, "Amodal panoptic segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 21 023–21 032.

[32] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *Int. Journal of Computer Vision*, vol. 123, pp. 32–73, 2017.

[33] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *Robotics: Science and Systems*, 2020.

[34] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim, "3-D scene graph: A sparse and semantic representation of physical environments for intelligent agents," *IEEE Trans. on Cybernetics*, vol. 50, no. 12, pp. 4921–4933, 2020.

[35] J. Johnson, B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, "Inferring and executing programs for visual reasoning," in *Int. Conf. on Computer Vision*, 2017, pp. 2989–2998.

[36] J. Wald, H. Dhamo, N. Navab, and F. Tombari, "Learning 3d semantic scene graphs from 3d indoor reconstructions," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[37] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "SceneGraphFusion: Incremental 3D scene graph prediction from RGB-D sequences," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, June 2021, pp. 7515–7525.

[38] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada, "EfficientLPS: Efficient LiDAR panoptic segmentation," *IEEE Trans. on Robotics*, vol. 38, no. 3, pp. 1894–1914, 2022.

[39] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Int. Conf. on Robotics and Automation*, 2021, pp. 11 054–11 059.

[40] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^2o: A general framework for graph optimization," in *Int. Conf. on Robotics and Automation*, 2011, pp. 3607–3613.

[41] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, vol. 96, no. 34, 1996, pp. 226–231.

[42] G. Jozkow, P. Wieczorek, M. Karpina, A. Walicka, and A. Borkowski, "Performance evaluation of sUAS equipped with Velodyne HDL-32E LiDAR sensor," *The Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 171–177, 2017.

[43] A. Van, D. Lindenbaum, and T. M. Bacastow, "SpaceNet: A remote sensing dataset and challenge series," *arXiv preprint arXiv:1807.01232*, 2018.

7.4 Learning to estimate incipient slip with tactile sensing to gently grasp objects

The appended paper [1] follows.

Learning to estimate incipient slip with tactile sensing to gently grasp objects

Dirk-Jan Boonstra*
Dept. of Cognitive Robotics
TU Delft
The Netherlands
ORCiD 0000-0002-9827-0636

Laurence Willemet[1]*
Dept. of CSAIL
MIT
United States
lwilleme@mit.edu

Jelle Luijkx
Dept. of Cognitive Robotics
TU Delft
The Netherlands
j.d.luijkx@tudelft.nl

Michaël Wiertlewski
Dept. of Cognitive Robotics
TU Delft
The Netherlands
m.wiertlewski@tudelft.nl

***Abstract*—To gently grasp objects, robots need to generate enough friction without creating damage by applying the right amount of force. In practice, implementing this force regulation is challenging since it requires knowledge of the friction coefficient, which can vary from object to object and even from grasp to grasp. Fortunately, tactile sensing can provide information about friction notably by detecting the moment when the object slips away from the grasp. These tactile sensors capture distributed information about the deformation of the artificial skin in the normal and tangential direction, from which slippage can be detected. However, current approaches only react to slip, which leads to significant object movement. The movement can in turn induce a failure of the grasp and damage. In this study, we introduce a machine-learning method that anticipates slip by computing the so-called safety margin of grasp. It represents the margin of an additional lateral force that the frictional contact can bear. To find this value, we use a high-density camera-based tactile sensor to measure the 3D deformation of the surface over 82 points. We trained a Convolutional Neural Network (CNN) to obtain a frictional safety margin estimate from the tactile images. The safety margin offers a powerful metric for regulation and therefore a simple proportional controller was enough to robustly grasp a wide collection of objects. The results show that this control method outperforms slip detection methods, by reducing regrasp reaction times while decreasing the maximum applied grasping force to the low-value range of 1-3 N.**

***Index Terms*—tactile sensing, grasping, safety margin, robotic gripper**

INTRODUCTION

When grasping and manipulating, the contact between fingers and objects is constantly evolving. Forces are changing and the pressure and traction distribution evolve with friction and material properties. Consequently, it can be difficult to estimate and predict how the object will move within the grasp and whether or not the grasp will be stable. This prediction is crucial for grasping since the efforts at the contact determine if the object can rotate, pivot, slide, or stay in place. For example, if the information about the frictional resistance is missing, a simple controller cannot determine the optimal grasping force to apply. Therefore friction-agnostic approaches generally overestimate the grip force to avoid a catastrophic loss of grip [1, 2]. However, such large forces prevent possible damage by dropping objects and restraining manipulation flexibility [3].

Tactile sensing is a promising avenue for capturing the mechanical interaction at the interface between the environment and the fingers. Robotic tactile sensors capture the deformation of an artificial skin from which they can infer high-order information such as the material properties, such as compliance, texture, curvature, or the contact state such as distance to slip or effort. Tactile sensors work by discretizing the mechanical interaction, often using miniaturized high-resolution cameras pointed at the membrane [4, 5].

The images of the deformation field can be processed to estimate contact shape and force [6], or to detect slip from physics-based models [7]. More complex mechanical interactions can be captured using machine learning approaches [8–10]. However, when deployed for grasping regulation, the adjustment is based on slip detection, making them often too late to react and unable to regain stability after a slip [11–13].

At a mechanical level, the transition from stick to slip for a soft fingertip occurs gradually. When the tangential force increases from a fully stuck contact, the outer edge of the contact area begins slipping while the center remains stuck. The slip region grows until the entire contact area is in the slip state and the object fully slips [14, 15]. Some hypotheses postulate that humans use the ratio between the stick and the slip region inferred from the skin deformation to estimate the safety margin [16]. This estimation of the distance from the onset of slip is believed to be ultimately used to regulate their grip force [17, 18].

In this work, we measured the pattern of deformation of the artificial fingertip before the onset of slip with an iterated version of our ChromaTouch tactile sensor [19, 20]. We trained a convolutional neural network (CNN) to estimate the frictional strength using the safety margin. The model performance is evaluated against an unseen dataset, which showed an average prediction accuracy of 98.2% from the ground truth, when computing the MSE loss over the entire range of safety margin predictions. This estimation is accurate enough to use the safety margin as control input for gripper control. The speed and accuracy of the estimation and control make it suitable for real-time grasping applications on soft and complex objects such as fruits and vegetables.

* D. Boonstra and L. Willemet contributed equally to this paper.
[1] Corresponding author

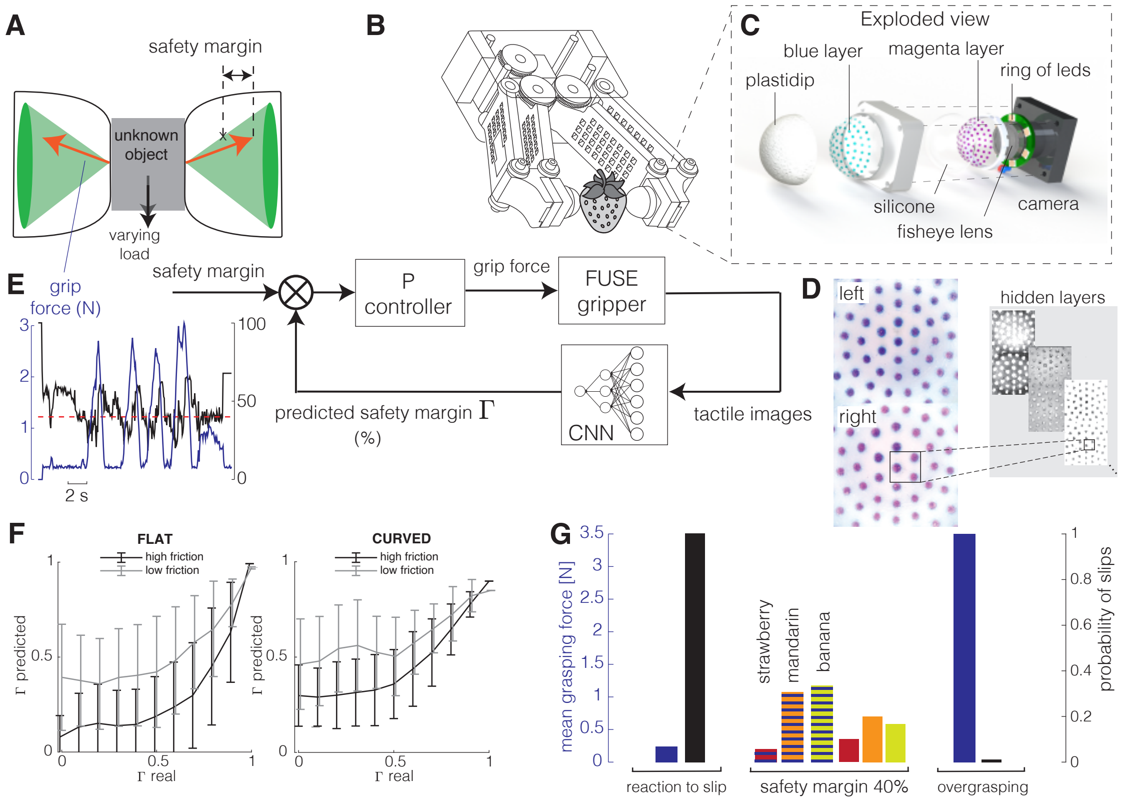

Fig. 1. **A**. Typical evolution of the interaction force when manipulating an object. The grip force is maintained with a safety margin Γ over the minimum required grip force defined by the friction cone. **B**. Render of the custom-made parallel FUSE gripper. **C**. Exploded view of the tactile sensor ChromaTouch. **D**. Hidden layers of the convolutional neural network are used to predict the safety margin. **E**. Grip force control to maintain a constant safety margin. **F**. Deviation of the prediction compared to the real safety margin. **G**. Mean grasping force and probability of slips for three control strategies (reaction to slip [13], constant safety margin of 40%, and overgrasping strategy with a fixed 3.5 N grasping force).

Our goal in this study is to create a tactile-enabled gripper that maintains a squeezing force on an arbitrary object so that the safety margin remains constant (Fig. 1A). To do so, we designed an impedance control gripper (Fig. 1B) which regulates its grasping force in real-time. The gripper has two soft tactile sensing fingertips able to capture the 3D deformation of a membrane using an embedded camera (Fig. 1C). The images of the interaction are fed to a CNN to estimate the frictional safety margin Γ (Fig. 1D,F). Γ is then used to adjust the grasping force in real-time (Fig. 1E), improving object manipulation and minimizing object slip (Fig. 1G).

Materials and Methods

Tactile sensing gripper

The FUSE tactile gripper consists of two main components: a set of Chromatouch tactile sensors and a custom-made robotic gripper (Fig. 2A). The Chromatouch tactile sensing mechanism relies on a color-mixing principle. Two layers of colored-markers are first 3D-printed with a Stratasys J735 PolyJet printer in flexible transparent AgilusClear with a Shore hardness of 30A. These layers are bonded together using a 1.2 mm-thick elastic silicone (Smooth-On SortaClear 12A), cast between the two marker layers. Three layers of white pigmented silicone (PlastiDip) are sprayed over the outside of the dome, to block light from external sources and to help diffuse internal light. A per-finger embedded USB camera (Basler Dart daA1920-160uc with a Basler Evetar M13B02118W fisheye lens attached) acquires 896 px $\times$ 896 px tactile images at 100 Hz with a surface resolution close to 22 px/mm. More information on the design of these tactile fingertips can be found in our previous work [20].

For this work, we custom-designed a parallel gripper where the force could be finely tuned and fit the need of delicate robot grasping. The design consists primarily of 3D-printed parts, with tactile sensors embedded at the fingertips. The design of this gripper, called FUSE, is made publicly available[1]. The fingers are driven using one servo-motor (Dynamixel XH430-W210-R), through a set of modified POM gears. The servo-motor enables our gripper with current control-control, which is calibrated to grasping force (N) by grasping a regular force sensor (ATI Nano43), see Fig. 2B. We assume a linear relation between motor current and grasping force.

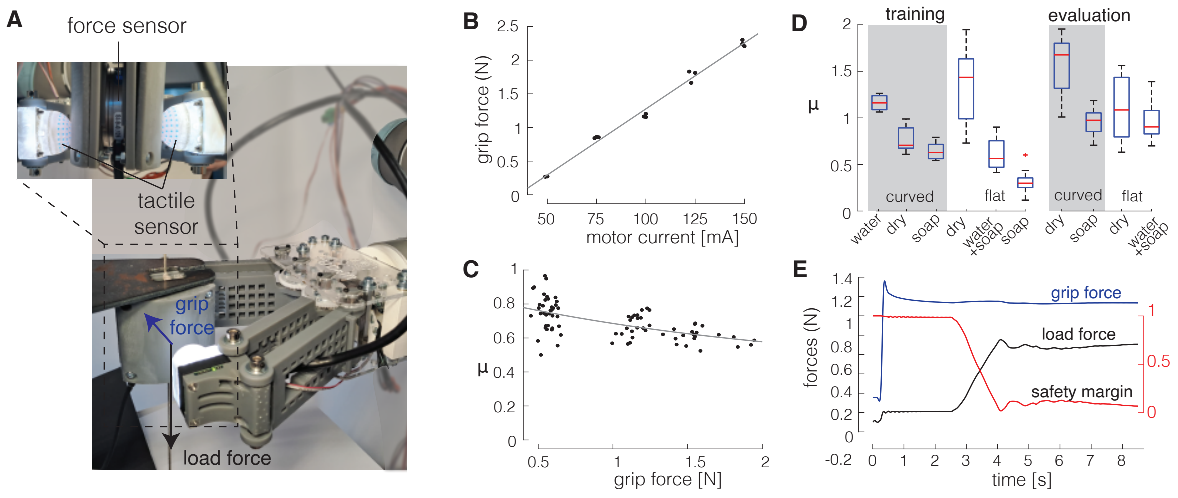

Fig. 2. **A**. Experimental setup for data collection. **B**. Calibration of the grasping force. **C**. Influence of the grip force on the friction coefficient. **D**. Mean and standard deviation of the friction coefficient estimated on the slipping trials for each dataset (training and evaluation). **E**. Grip force and load force evolution along one trial. The gripper first closes its arms on the load-compensated object, the DC-motor applies a pulling force after 2 s. The safety margin is computed using an approximation of the friction coefficient.

Experimental setup

We designed an experimental setup to impose arbitrary tangential and normal forces on the tactile sensors (Fig. 2A). During a trial, the tactile gripper first grasps, with a predefined grasping force, a stationary suspended object, whose weight is compensated by DC-motors and a capstan assembly. The apparent weight was programmed at 0g by constantly pulling the object against the direction of the gravitational field. A force sensor (ATI Nano43) is embedded inside this object, which measures the 3D force interaction at the gripper's fingertips. A second-long ramp-pulling force is applied to the suspended object two seconds after the object has been grasped, trying to pull it from the fingertips using two DC motors (Faulhaber, 2642 012 CR). A capstan transmission is used to deliver the resulting force of the DC motors to the object. The pulling force produced by the DC motors can be controlled using a linear servo amplifier (Maxon, LSC 30/2).

The position of the object was measured with an encoder (Baumer, BTIV 24S 16.24K 1024 G4 5) and the safety margin Γ was estimated using the force measurement and an approximation of the friction coefficient (Fig. 2E):

$$\Gamma(t) = \frac{f_f^* - f_f(t)}{f_f^*}, \tag{1}$$

with f_f the current frictional force, and f_f^* the maximum applicable force to overcome the frictional strength of the grasp, after which the object will start to slip. This frictional force was computed using an estimation of the friction coefficient as the average force ratio when the object is slipping (Fig. 2D). The influence of the grip force on the friction coefficient has been neglected (Fig. 2C).

[1]https://github.com/Dirrkk/fuse-gripper

During data collection, a randomized experimental plan containing the grasping force and the load force and its rate was followed. The grasping forces vary from 1.0 to 2.5 N in 0.5 N intervals, the load force was uniformly chosen between 2 and 3 N and the force rate was controlled at 2 levels (0.5 N/s and 1 N/s). The training dataset consisted of a flat and a curved object (radius of curvature $R = \infty$ and $R = 45$ mm) at 3 different frictional conditions (high, medium, low). These friction conditions were obtained by adding water or soap on the surface of the objects, the corresponding friction coefficients are reported Fig. 2D (left side). Every condition has been cycled 16 times using the experimental plan, resulting in a total of 96 trials, or 177.000 (left+right) images. We used 80% for training and 20% for testing. This procedure has been repeated 4 days later (the temperature was $6°C$ lower and the humidity decreased by 3%), to collect a validation dataset consisting of the same 2 objects, on 2 frictional conditions (high and low), resulting in 64 trials and 118.000 images. The data analysis is performed on this validation dataset.

Model training

The images from the tactile gripper and the above-computed safety margin, Γ are linked together using a CNN, ShuffleNetV2 [21]. This network is lightweight and mobile, and showed promising results in related tactile sensing studies [3]. Our application is suited for a lightweight network since the tactile images represent close-contact information with a limited pixel size, and do not need to contain complete environmental scenes. Furthermore, the lightweightness gives tactile grasping demonstrations in a portable setting.

Inputs of the network are the tactile images from both fingertips. The raw images are resized to 224 × 224 px, after which they are concatenated vertically and fed into the network, ensuring both images have the same timestamp.

ColorJitter and GaussianBlur data augmentation techniques are used to increase the generalization capabilities of the tactile model.

The output of the ShuffleNetV2 network is adapted to a single floating point value, which equals the predicted Γ for the input image. This reduction in output space is obtained by combining a set of linear layers with LeakyReLU activation functions and Batch Normalization (1D) layers, decreasing the output space from 1024 nodes to 1.

Training was done using PyTorch on an Ubuntu 20.04 machine with an Intel Xeon CPU and using CUDA on a Nvidia RTX3060Ti GPU with 8GB of video memory. We used 50 epochs with the batch size set to 64. The MSEloss function was used for backpropagation. For the optimizer, we made use of the Ranger21 framework [22], which is built around the AdamW optimizer, while also providing several techniques to further increase performance and prevent influences from local minima. We used Ranger21 default parameters and set the number of iterations to (number of epochs) x (length of the training dataset).

Controller design

The trained CNN described above, outputs one floating point value (Γ), based on a set of images from the left and right fingertips combined. A simple P controller is deployed on the gripper, which takes the estimated Γ value as input, and outputs the grip force required to maintain a target value for Γ. By adapting the target value, the distance to slip can be varied on a per-object basis. The K_P gain is set to 2, and the Γ difference is in the range 0-100%. The minimum applied grip force is set to 0.25 N, to keep the object in a force closure grasp. To prevent damage to the tactile sensor's soft silicone layer, the maximum applied grip force was limited to 3.2 N.

Working principle of the tactile sensor

To validate the working principle of the tactile sensor, we needed to calibrate the acquired images towards actual 3D deformation using an analytical model. We first acquired images during a normal indentation and a lateral slide (Fig. 3A). On Fig. 3B, images of the preload, and image differences of the initial contact, incipient slip and full slip are shown. We proceeded to find a robust transformation of the sub-image around each marker into the normal displacement u_z (Fig. 3C). To find this transformation, we used as ground-truth a filtered version of the Hertzian contact theory. This model predicts a parabolic displacement of the surface of the membrane. For further explanation about the method, refer to our previous work [20]. The lateral displacements u_x, u_y are computed using a marker tracking method (Fig. 3D). The effective 3D displacements are shown at different time courses during slip, for a high and a low friction condition in Fig. 3E. The output displacements have a sub-millimeter precision in the normal and tangential plane.

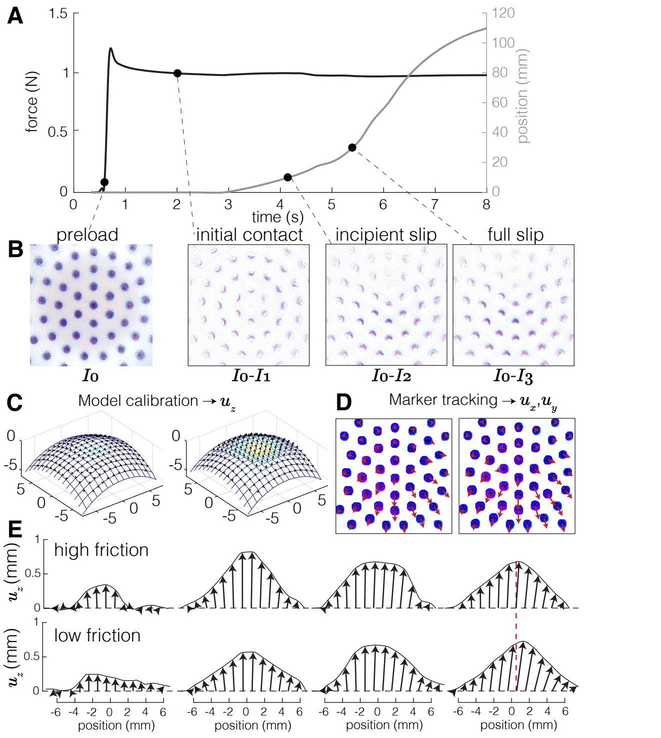

Fig. 3. **A**. Grasping force and vertical position of an object being gripped. **B**. Tactile images obtained during 4 phases. **C**. Interpolated normal displacement u_z of the grid of markers calibrated using a filtered version of Hertz contact model. **D**. Lateral displacement u_x and u_y obtained with markers centroid tracking. **E**. Reconstruction of the 3D displacement of each marker along the direction of slippage for a high and a low friction condition. The influence of friction is indicated by the vertical line.

RESULTS

Safety margin estimation

To show the accuracy of the trained CNN, we compare the model output, which only sees the tactile images, with ground truth data extracted from the force measurement collected during the experiments. We estimated the safety margin Γ on a validation dataset of unseen images when the gripper interacts with a flat or a curved object and with a high or a low friction coefficient. The ground-truth safety margin was recorded using a force sensor embedded into the object (see Materials & Methods for further details). To compare the predictions of Γ with the real values, we compute the Mean Squared Error (MSE) loss over the validation datasets, the trial average can be found in Table I. Taking the average over the four datasets yields a combined MSE loss of 0.01821, giving the total model an accuracy of 98.2% when predicting Γ.

The confusion matrices are shown in Figure 4A with the safety margin Γ divided in 10% bins to evaluate model performance on the full 0 to 100% range. The model showed similar performance on the flat and the curved object (Fig. 1F). Thus, to show the influence of friction on the safety margin estimation, the flat and curved object datasets are averaged in a single confusion matrix. From the diagonal trend in

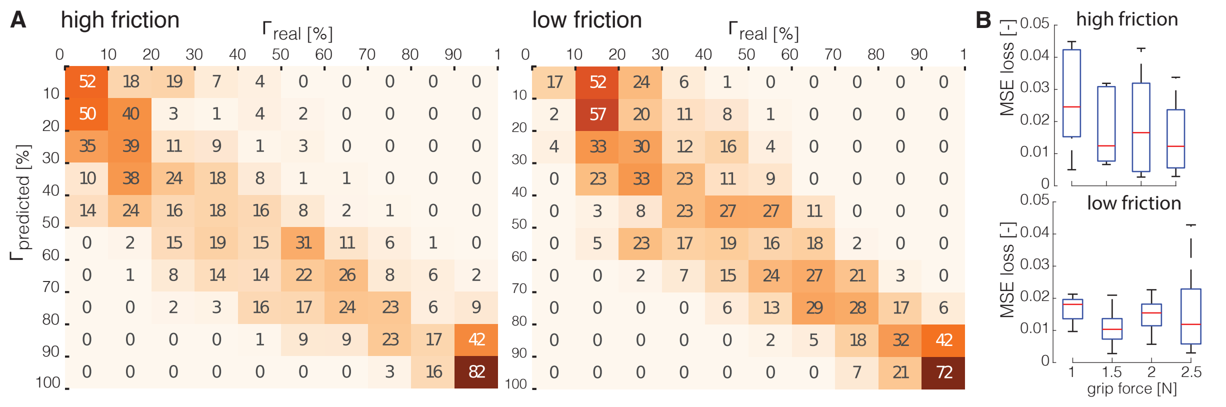

Fig. 4. **A**. Confusion matrix of Γ real versus Γ predicted, with experiments for both flat and curved objects combined. The experiment is done at high and low friction conditions. A perfectly trained model would result in only predictions on the matrix's diagonal. **B**. MSE loss (mean ± std) of the safety margin prediction for each applied grip force for high and low friction condition.

TABLE I
MEAN SQUARED ERROR (MSE) LOSS (MEAN ± STD) COMPUTED OVER THE 4 VALIDATION DATASETS: A FLAT AND A CURVED OBJECT EVALUATED ON BOTH HIGH AND LOW FRICTION CONDITIONS. THE MSE SHOWN IS THE COMBINED AVERAGE OVER ALL 16 TRIALS PER CURVATURE/FRICTION CONDITION.

MSE loss [-]	**Flat**	**Curved**
high friction	0.03005 ± 0.04232	0.01357 ± 0.02482
low friction	0.01256 ± 0.02480	0.01665 ± 0.01941

the figure, we can see that the predicted safety margin is positively correlated with the ground truth safety margin in both friction conditions. However, friction has a significant influence on the safety margin prediction accuracy (Anova, $F(1,63) = 4.27$, p= 0.043). We observe similar performance for both friction conditions when the safety margin is higher than 40%. However, when the safety margin drops below 10%, the model performs better in the high friction condition. In the low friction case, we can see that the prediction is underestimating the ground-truth with errors up to 0.4 when the real safety margin is equal to 0 (Fig. 1F).

Finally, the grip force applied has no significant influence on the MSE, although we can see that lower grip force results in higher errors in the high friction condition (Fig. 4B).

Controller validation

To evaluate the safety margin framework in a real time grasping task, the CNN model output is used with a Proportional controller to control the grip force of the gripper. Experiments are performed on a set of delicate soft fruits: a strawberry, a banana, and a mandarin. A 50 g weight was attached with a wire from the grasped object to induce a sudden increase in load force on the object. Fig. 5A shows the grip force required to hold the objects with a given target safety margin of 20%, 40%, and 60%. The boxplots represent the average and standard deviation over 5 trials. We can see a significant increase in grip force when increasing the safety margin target for all fruits (p< 0.001 and $F(2,31) = 189.7$, $F(2,33) = 19.6$, $F(2,29) = 80.74$ for the strawberry, the mandarin and the banana respectively). To measure the control reaction time, the attached weight is dropped, causing a sudden spike in load force. The gray zones from Fig. 5A show that, on average, this increase in load force resulted in an increased grip force, especially for the 60% safety margin target, which allows for higher grasp forces.

The probability of object slip decreases with the increase in safety margin, as shown in Fig. 5B. We can see that for the 20% target, the gripper managed to keep the relatively light strawberry between its fingertips for most cases, while the heavier banana and mandarin experienced more slips. Increasing the safety margin target to 40% and 60% caused fewer slips for the heavier objects. We can even see that for the 60% case, the strawberry experienced zero cases of full slip.

We also performed an experiment with slowly increasing load force, when grasping an empty cup which was slowly manually filled with rice. The results can be seen in Fig. 5C. At around 10 seconds, as weight is slowly added, we recorded an increase in grip force to maintain the set control target. We can see that over the whole range, a higher grip force is required to maintain the 60% target, while the 40% target only increases grip force to around 1 N when maintaining the grasp.

CONCLUSION AND DISCUSSION

We introduced a new method for controlling the grasping force of a gripper based on a new metric called the frictional safety margin. The frictional safety margin Γ was extracted from tactile sensor images using a convolutional neural network and its prediction shows an average accuracy of 98.2% compared to the safety margin found with force measurements. The trained network was evaluated on a validation dataset recorded on a different day. The robustness of the results shows the model capabilities in expanding to other friction conditions, as these vary on a daily basis and are highly sensitive to environmental conditions like humidity.

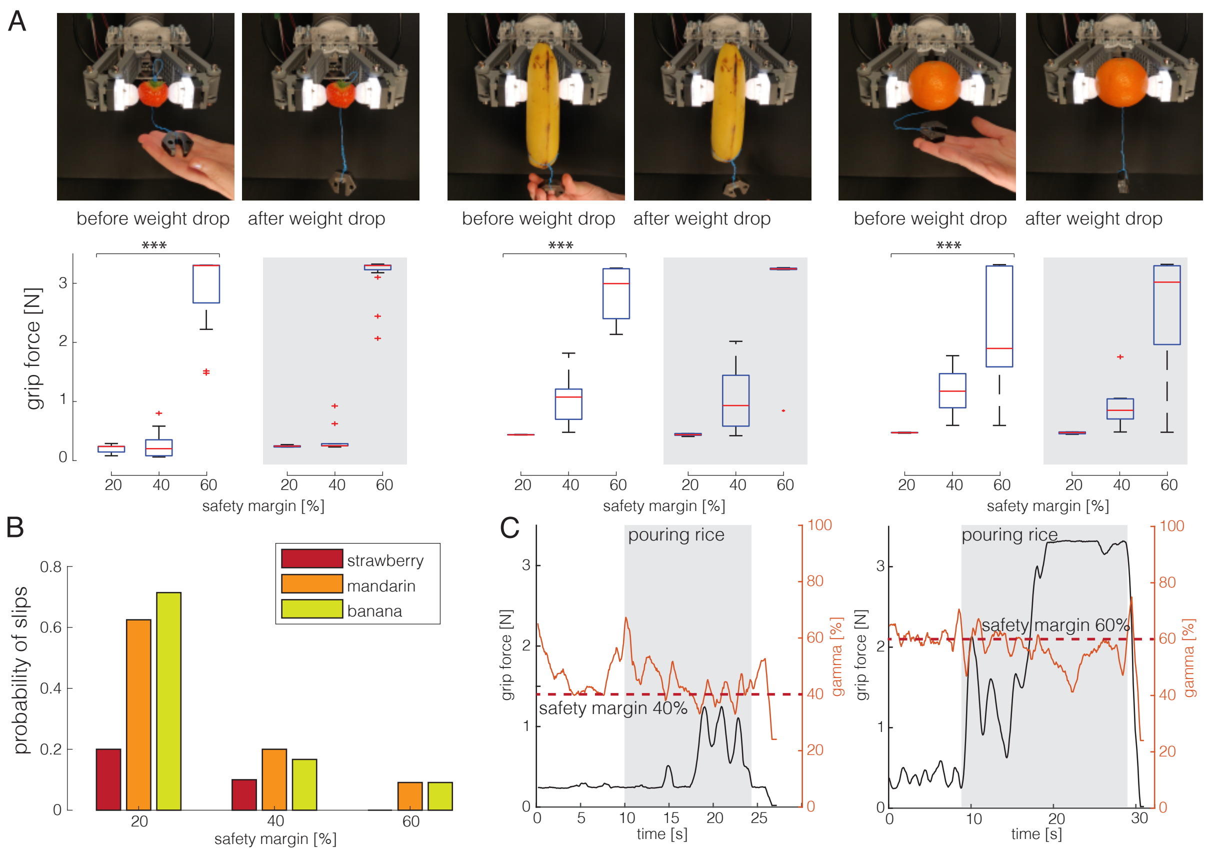

Fig. 5. **A**. Mean grasping force for 3 different fruits and 3 safety margin commands. The gray zone represents the period after the weight is dropped. **B**. Probabilities of slip as a function of the safety margin for the 3 fruits. **C**. Grip force and predicted safety margin when the FUSE gripper is grasping a cup being filled with rice. The safety margin was controlled at 40% and 60% respectively on the left and on the right.

Despite its excellent performance, the main limitation of the proposed approach is the prediction of Γ when the frictional contact is either small or slippery. To overcome the first issue, we decided to run our experiments within a limited range of grasping forces, between 0.25 and 3.2 N. Limiting the range kept results accurate while preventing damage of the tactile sensors. On slippery surfaces, we noticed that the accuracy of the estimate decreases for lower safety margins, especially in low friction conditions. The decrease in accuracy can be caused by the sparsity of data used for training in the lower range of safety margin. During data acquisition, the low safety margin, which corresponds to gross slippage, was not always reached because of limitations in the experimental setup in which maximum applicable load force was limited to 3 N. This low accuracy can result in difficulties in handling fragile and slippery objects, as an underestimation of the safety margin will result in a higher grasping force than necessary. However, the study shows that the optimal target safety margin is around 40% so low values of safety margin will be rarely reached. Furthermore, in cases where the predictions are less accurate, the flexibility of this control approach allows for increasing the safety margin. The error in the estimation is also illustrated by a large standard deviation of the predicted safety margin. This uncertainty can result in fluctuations in grasp force.

A second validation has been done by evaluating the grasping performance of fragile real-life objects. These real-life experiments show that, although the model has been trained on two object shapes, it can generalize to more complex scenarios. The frictional safety margin can also be used to increase grasping performance while reducing object damage. The trained network predicts Γ at 50 Hz on a desktop CPU, making it fast enough for real time control. The reaction time of the gripper after a weight drop has been measured at approximately 100 ms, which is in the same order of magnitude as the human reaction times to an external perturbation [23].

Acknowledgments

This work was supported by the 4TU Soft Robotics program. J.L. acknowledges the EU's H2020 OpenDR project (grant No 871449). We would like to thank Jens Kober for insightful comments, Mostafa Atalla, Marlies Popken and Max Polak for technical assistance.

References

[1] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, 2012.

[2] L. Zhang and J. C. Trinkle, "The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3805–3812.

[3] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5761–5768, 2021.

[4] N. F. Lepora, "Biomimetic active touch with fingertips and whiskers," *IEEE transactions on haptics*, vol. 9, no. 2, pp. 170–183, 2016.

[5] C. Sferrazza and R. D'Andrea, "Design, motivation and evaluation of a full-resolution optical tactile sensor," *Sensors*, vol. 19, no. 4, p. 928, 2019.

[6] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[7] P. Griffa, C. Sferrazza, and R. D'Andrea, "Leveraging distributed contact force measurements for slip detection: a physics-based approach enabled by a data-driven tactile sensor," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4826–4832.

[8] F. Visentin, F. Castellini, and R. Muradore, "A soft, sensorized gripper for delicate harvesting of small fruits," *Computers and Electronics in Agriculture*, vol. 213, p. 108202, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169923005902

[9] S. Dong, D. Ma, E. Donlon, and A. Rodriguez, "Maintaining grasps within slipping bounds by monitoring incipient slip," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3818–3824.

[10] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 297–303.

[11] J. Gao, Z. Huang, Z. Tang, H. Song, and W. Liang, "Visuo-tactile-based slip detection using a multi-scale temporal convolution network," 2023.

[12] J. W. James and N. F. Lepora, "Slip detection for grasp stabilization with a multifingered tactile robot hand," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 506–519, 2020.

[13] F. Veiga, B. Edin, and J. Peters, "Grip stabilization through independent finger tactile feedback control," *Sensors*, vol. 20, no. 6, p. 1748, 2020.

[14] B. Delhaye, A. Barrea, B. B. Edin, P. Lefevre, and J.-L. Thonnard, "Surface strain measurements of fingertip skin under shearing," *Journal of The Royal Society Interface*, vol. 13, no. 115, p. 20150874, 2016.

[15] M. Tada, "How does a fingertip slip?-visualizing partial slippage for modeling of contact mechanics," in *2006 Proceedings of Eurohaptics*, 2006, pp. 415–420.

[16] L. Willemet, N. Huloux, and M. Wiertlewski, "Efficient tactile encoding of object slippage," *Scientific Reports*, vol. 12, no. 13192, 2022.

[17] F. Schiltz, B. P. Delhaye, J.-L. Thonnard, and P. Lefèvre, "Grip force is adjusted at a level that maintains an upper bound on partial slip across friction conditions during object manipulation," *IEEE Transactions on Haptics*, vol. 15, no. 1, pp. 2–7, 2021.

[18] A. M. Hadjiosif and M. A. Smith, "Flexible control of safety margins for action based on environmental variability," *Journal of Neuroscience*, vol. 35, no. 24, pp. 9106–9121, 2015.

[19] X. Lin and M. Wiertlewski, "Sensing the frictional state of a robotic skin via subtractive color mixing," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2386–2392, 2019.

[20] R. B. Scharff, D.-J. Boonstra, L. Willemet, X. Lin, and M. Wiertlewski, "Rapid manufacturing of color-based hemispherical soft tactile fingertips," in *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*. IEEE, 2022, pp. 896–902.

[21] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[22] L. Wright and N. Demeure, "Ranger21: a synergistic deep learning optimizer," *arXiv preprint arXiv:2106.13731*, 2021.

[23] K. J. Cole and J. H. Abbs, "Grip force adjustments evoked by load force perturbations of a grasped object," *Journal of neurophysiology*, vol. 60, no. 4, pp. 1513–1522, 1988.