UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXV

Advances in AGV perception for people and object detection in industrial warehouses

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutore:
Chiar.mo Prof. Jacopo Aleotti

Dottorando: Michela Zaccaria

Anni 2019/2022

To my parents,
with love and gratitude

Abstract

The advances in computer vision are meeting many market requirements in sectors such as automotive, security surveillance and industry 4.0. This thesis presents a research on applications of novel computer vision methods in automated warehouses. The idea is to design the next generation of Automated Guided Vehicles (AGVs), that should be able to perceive the surrounding environment to improve productivity and safety in an intelligent way. The main contributions are:

- the development of a multi-robot multiple camera system for people detection and tracking
- a comparison of deep learning models for 2D pallet detection
- a self-supervised method for category-level 6D pose estimation with geometric consistency using optical flow

The first contribution is the introduction of a multi-robot system for people detection and tracking in automated warehouses. Each Automated Guided Vehicle is equipped with multiple RGB cameras that can track the workers' current locations on the floor thanks to a neural network that provides human pose estimation. Based on the local perception of the environment, each AGV can exploit information about the tracked people for self-motion planning or collision avoidance. Additionally, data collected from each robot contribute to a global people detection and tracking system. A warehouse central management software fuses information received from all AGVs into a map of the current locations of workers. The estimated locations of workers are sent back to the AGVs to prevent potential collisions. The proposed method is based

on a two-level hierarchy of Kalman filters. Experiments performed in a real warehouse show the viability of the proposed approach.

A second contribution is related to the problem of automatic pallet detection in industrial environments using a single RGB camera. The problem is relevant in the context of Autonomous Guided Vehicle navigation, and for tasks like pallet storage and retrieval. In particular, an approach based on a convolutional neural network (CNN) followed by a decision-making step is presented. The convolutional neural network is trained to recognize two elements of a pallet, namely the pallet front side and its pockets. Also, a comparative study is carried out between three state-of-the-art CNNs: Faster R-CNN, SSD and YOLOv4. For training and evaluation, a dataset was collected in a warehouse. The dataset contains images of pallets in different configurations, either on the ground or on racks, and with arbitrary orientation. Overall, results indicate that Faster R-CNN and SSD perform better than YOLOv4.

Lastly, the problem of category-level 6D object pose estimation with deep learning was investigated using benchmark datasets. Category-level 6D object pose estimation aims at determining the pose of an object of a given category. Most current state-of-the-art methods require a significant amount of real training data to supervise their models. Moreover, annotating the 6D pose is very time consuming, error-prone, and it does not scale well to a large amount of object classes. Therefore, a handful of methods have recently been proposed to use unlabelled data to establish weak supervision. The proposed approach leverages the 2D optical flow as a proxy for supervising the 6D pose. To this purpose, the 2D optical flow between consecutive frames based on the 6D pose estimation is computed. Then the framework harnesses an off-the-shelf optical flow method to enable weak supervision using a 2D-3D optical flow based consistency loss. Experiments show that the proposed approach for self-supervised learning yields state-of-the-art performance on the NOCS benchmark, and it reaches comparable results with some fully-supervised approaches.

Contents

Chapter 1

Introduction

This Chapter introduces the key concepts of the PhD dissertation. Firstly, the main motivations behind this thesis are explained. Subsequently, the main contributions and an outline of the thesis structure are summarized.

Motivation

The computer vision field had an enormous expansion recently. Thanks to more powerful hardware and the introduction of new algorithms, much progress has been done in several tasks. Also, progress in deep learning had a strong impact on the evolution of computer vision in the last decade. In fact, nowadays for many computer vision tasks the state of the art involves convolutional neural networks, that seem to reach or outperform methods not based on machine learning.

However, the gap between research and applications in real environments is not completely filled. For some applications, such as autonomous driving, there are several studies about the use of computer vision techniques to solve application-oriented problems [1] [2] [3]. It is not possible to say the same for industrial applications. Recently, there have been studies such as ILIAD [4], a project founded by the European Union's Horizon 2020 with the more generic aim to improve Intra-Logistics with Integrated Automatic Deployment. However, the research field still suffers from

the lack of available datasets and benchmarks. Hence, the motivation for this work was the application of computer vision in an industrial environment (automated warehouse) to solve or help in performing tasks that can improve productivity and safety. An automated warehouse is an industrial environment where the process of goods picking, storage and dispatching is completely automatized. Automated warehouses include Automated Guided Vehicles (AGVs). In an automated warehouse, there are a few to hundreds of AGVs moving in the workspace. The tasks carried out by such vehicles are various as they need to move in the environment coordinating their paths, avoiding obstacles and people, loading and unloading goods in structured or unstructured areas. To accomplish these tasks the vehicle collects data from sensors and processes them to understand the surrounding environment. In this way, AGVs can safely handle heavy loads or goods that are complex to transport, and they can maintain high standards of performance. Currently, most of the tasks are carried out by exploiting onboard planar laser sensors. However, although laser scanners provide high-accuracy measurements, laser-based sensors have some drawbacks. For instance, they are often used together with static reflective markers, that need maintenance. Also, they have higher costs with respect to cameras and perform better only for short distances. Furthermore, they can not work with all materials, for example with transparent films that are commonly used for packaging. Hence, it is important to investigate the potential of computer vision approaches to face some of the problems mentioned before. Cameras are sensors that come at a lower cost and that can provide color information of the surrounding environment, that makes them more suitable for task such as pattern detection. The purpose of this dissertation was to develop a system integrated with the existing AGVs of E80Group S.p.A. that can perceive the surrounding environment, eventually making decisions to interact with it, in the vision of industry 4.0 (Figure 1.1).

This work was carried out in collaboration with E80 Group S.p.A. which is a leader company in the field of automation logistics. E80 Group S.p.A. was founded in 1980 in Viano, Italy and currently it is located in Australia, Brazil, Chile, China, the United Arab Emirates, France, Japan, Mexico, Spain, Sweden, the United Kingdom, Poland, Russia, Thailand and USA. E80 Group S.p.A counts more than 300 inte-

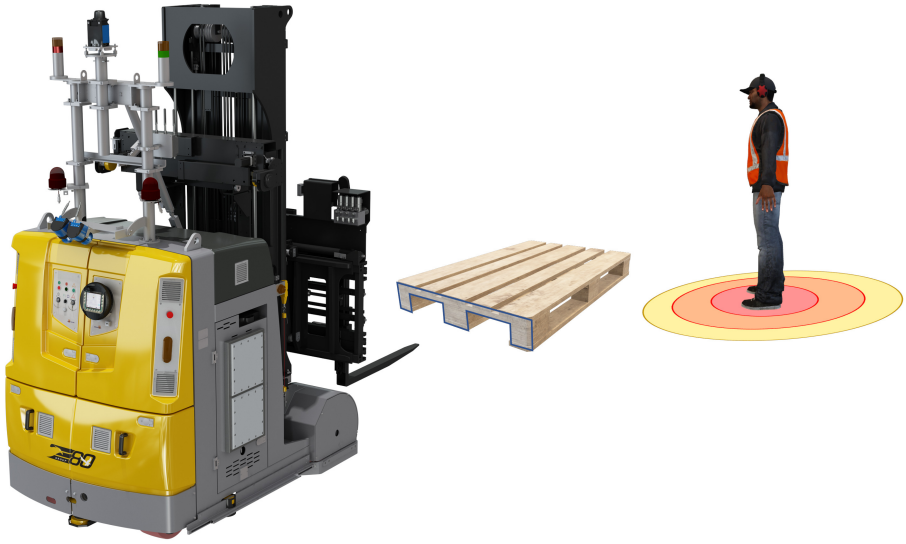

Figure 1.1: An intelligent AGV that can recognize and interact with the surrounding enviroment to improve performance and safety.

grated factories, with over 2000 robotic systems, 5000 Automated Guided Vehicles and 35 automated high-density warehouses. The main sectors are beverage, tissue and food. The products portfolio involves many solutions like palletizing and depalletizing robotic systems, high-speed stretch wrappers (Silkworm), robotic labelers, automated picking solutions, AVS/RS SmartStore and AS/RS CraneStore. The main products on which this work focuses are the Automated and laser-guided vehicles. In E80 Group there are different AGVs models that can adapt to several requirements and customizations (Figure 1.2).

Contribution and outline

For the reasons presented in the previous section, my research faced some relevant problems of a typical industrial environment. A typical need of the industry is human

Figure 1.2: E80 Group S.p.A. AGV models.

safety. Each AGV is equipped with a Proximity Laser Scanner (PLS) to meet the standard quality requirement defined by the International Organization for Standardization (ISO 3691-4 : 2020). However, to react to potentially dangerous situations, computer vision can help to prevent accidents. Chapter 2 presents a multi-robot multiple camera people detection and tracking system. The system makes each AGV able to detect and track people through multiple cameras. Furthermore, sensor information from each AGV is sent to a warehouse central management software, that exploits it to prevent potential collisions.

Another considered scenario is the loading of a pallet of goods. Sensors like 2D laser scanners are not suitable for pallet detection when there is a protective film used for wrapping. Furthermore, laser scanners are usually limited to work on a single plane at a fixed height. In Chapter 3, it is presented a method for automatic pallet detection based on a monocular vision system. The approach is based on a convolutional neural network (CNN) followed by a decision-making step. It is able to detect pallets in the environment even in presence of several layers of transparent plastic film wrapping the pallets (Figure 1.3). The presented method could be useful to count pallets on racks to update storage information or to perceive the presence of a pallet during pallet loading task. The pallet loading task requires inserting the forks of the

Figure 1.3: Example of failure cases for the planar laser scanner.

Figure 1.4: Example of different pallet sizes and dimensions.

vehicle in the pallet pockets and then lifting the pallet. Therefore, a potential application would be to use visual servoing to control the forking operation based on the feedback from the pallet detection on the 2D image.

To accomplish the forking task another approach would be to estimate the 3D pallet location. Currently, to accomplish a pallet loading tasks the environment needs to be structured and the location of the pallet must be known in advance. Moreover, even if there are some standard pallet dimensions, they are not unique. Figure 1.4 shows some different pallet types from E80 Group warehouses. The International Organization for Standardization defines six types of pallets (ISO 6780:2003), often used as references. However, the European Pallet Association (EPAL) also defines some standards (EUR, EUR1, EUR2, EUR3, EUR6) widely used, especially in Europe. Furthermore, based on different industrial needs for handling goods, the pallet

dimensions can change between North America and Australia for example, and often some custom solutions are adopted. Given the spread of industrial warehouses all around the globe, it would be difficult to know the pallet dimensions in each case. Moreover, in each industry plant, several different pallet types can be adopted.

Hence, two solutions were explored to estimate the pallet 3D pose without the assumption that the pallet position and dimensions are known in advance. The first solution is to use a stereo camera to achieve a 3D reconstruction. Chapter 4 presents a pipeline for pallet detection and localization, with several experiments to assess the accuracy and precision. The method does not require previous knowledge about the location and dimensions of the pallet.

A second solution is the use of deep learning to directly infer the 6D pose, which involves 6 parameters for the translation and rotation, and the object shape reconstruction. The main limitation of pose estimation neural networks is the lack of datasets, as well as the difficulty to create a new dataset due to the extensive work needed for labelling poses. Moreover, instance-level pose estimation requires the 3D CAD model of the object and this is not acceptable if the pallet model is not known in advance. For this reason, a self-supervision framework for category-level 6D pose estimation and shape reconstruction is introduced in Chapter 5. The proposed framework leverages the optical flow between consequent frames to self-supervise the training of 6D pose estimation on a new unseen and unlabeled dataset. This research was conducted in collaboration with the Technical University of Munich.

Lastly, conclusions and future works are discussed. The list of publications based on this thesis work at time of preparation of this final version is here reported.

International Conferences

- **M. Zaccaria**, R. Monica and J. Aleotti, “A Comparison of Deep Learning Models for Pallet Detection in Industrial Warehouses," *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2020, pp. 417-422, doi: 10.1109/ICCP51029.2020.9266168.

- **M. Zaccaria**, M. Giorgini, R. Monica and J. Aleotti, “Multi-Robot Multiple Camera People Detection and Tracking in Automated Warehouses," *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, Palma de Mallorca, Spain, 2021, pp. 1-6,
doi: 10.1109/INDIN45523.2021.9557363.

International Journals

- **M. Zaccaria**, F. Manhardt, Y. Di, F. Tombari, J. Aleotti, M. Giorgini, “Self-Supervised Category-level 6D Object Pose Estimation With Optical Flow Consistency”, under review

.

Chapter 2

Multi-Robot Multiple Camera People Detection and Tracking

2.1 Introduction

In automated warehouses, where multiple Automated Guided Vehicles (AGVs) operate, improving worker safety is crucial. For example, knowing the locations of workers in real-time within the warehouse can help to prevent accidents due to critical situations, like occlusions.

This Chapter presents a people detection and tracking approach for multi-AGVs systems, where each industrial vehicle is equipped with multiple RGB cameras and embedded hardware for video processing. The proposed method is based on a neural network for human pose estimation and a two-level hierarchy of Kalman filters for people tracking.

Computer vision approaches for people (or object) detection and tracking often rely on monocular vision [5, 6, 7, 8, 9]. In particular, the work by Agarwal et al. [5] proposed a real-time multiple object tracker for a single camera using Faster RCNN for detection, as well as a deep regression network for correspondence tracking. Cosma et al. [7] developed a computer vision system for pedestrian location estimation that was evaluated using two fixed cameras and embedded devices. A simple

online and real-time tracking (SORT) method was presented in [9] that exploited a Convolutional Neural Network (CNN) for detection and tracking like in the proposed approach. However, in [9] tracking was performed in the 2D image space, while the approach proposed in this thesis works in 3D space. Therefore, the SORT approach cannot be easily extended to a multi-camera setup with multiple mobile robots.

Other multi-camera approaches have been presented in [10, 11, 12]. The approach in [10] was based on an Extended Kalman Filter that enabled tracking of a single person. Moreover, it did not support body pose detection. In [11] a motion capture framework was presented that performed sensor fusion of multiple static cameras and IMU sensors attached to human bodies. Chen et al. [12] proposed a method for tracking the pose of multiple humans using multiple fixed cameras in real-time.

A review of multiple object tracking (MOT) techniques was presented in [13]. In general, it can be observed that embedded devices constraints, which are important in industrial applications, have been rarely considered in research studies dealing with people tracking.

Several authors investigated the problem of tracking people by multiple mobile robots equipped with planar laser scanners or RGB-D cameras. A disadvantage of laser scanners is that such sensors are more expensive than RGB cameras, and they do not provide enough information to detect the human pose. In particular, Tsokas [14, 15] applied a multiple hypothesis tracking algorithm [16] together with a Kalman filter. Similarly, in [17, 18] people tracking was conducted in outdoor environments by multiple robots equipped with laser scanners and GPS sensors. In [19] people tracking (without pose estimation) was performed using an RGB-D camera on a single mobile robot. However, RGB-D cameras have a limited range.

2.2 Method

The proposed approach for people detection and tracking is based on a two-level hierarchy. At the base level of the hierarchy, a local perception system is executed on each AGV. The local perception system of an AGV estimates the position and the velocity of the people around the vehicle in the global reference frame of the warehouse. In

particular, a people detection algorithm is executed for each camera mounted on the vehicle, and low-level Kalman filters fuse data from each onboard camera for people tracking. The position of each person, expressed in the local reference frame of the vehicle, is converted to the global reference frame of the warehouse given the pose of the AGV in the global reference frame of the warehouse, which is always available thanks to an AGV localization system based on reflectors installed over the walls of the building. At the top level of the hierarchy, a warehouse central management system runs high-level Kalman filters that track all workers' positions by fusing data received from the AGVs. The estimated locations of workers are sent back to the AGVs to prevent potential collisions.

2.2.1 AGV local perception: camera calibration

Extrinsic calibration of each onboard RGB camera was performed in order to determine the pose of the sensor with respect to the AGV local reference frame. Given the camera extrinsic calibration matrix and the pose of the AGV with respect to the warehouse reference frame, the current pose of the camera with respect to the warehouse reference frame can also be determined. The warehouse world reference frame is a right-handed coordinate system with the x-axis and y-axis lying on the floor and the z-axis pointing upwards.

In this work, I developed a fast and simple method for extrinsic calibration of onboard cameras that does not require the use of calibration markers. The approach takes advantage of the availability of a detailed and highly accurate 3D scan (point cloud) of the warehouse, acquired by a terrestrial laser scanner that defines the warehouse reference frame (Figure 2.1). In particular, a set of salient feature points in the building (e.g. points on the floor or corners) are picked in the 3D point cloud, using a custom visualization tool, as well as the corresponding 2D points on the camera image to solve the calibration problem (Figure 2.2). This proposed technique for camera calibration is quite efficient, considering that the total number of AGVs and cameras that must be calibrated in a large warehouse can be significantly high.

Figure 2.1: On the left, a terrestrial laser scanner on a tripod acquiring 3D scan of the warehouse. On the right, an example of 3D point cloud acquired.

2.2.2 AGV local perception: people detection

The pipeline of the proposed AGV local onboard perception system for people detection and tracking is shown in Figure 2.3. Incoming image frames of each camera $i \in 1 \ldots N$ are used as input for a human pose estimation neural network. In this work, the open-source Nvidia TRTPose model was adopted [20]. The TRTPose model contains a neural network architecture introduced in [21, 22]. TRTPose is based on the TensorRT framework that provides efficient low-level code, optimized for embedded systems like the Jetson platform. The output of the neural network is a set of visible keypoints for each detected person in the image frame. TRTPose detects up to 18 keypoints associated to the following body parts: nose, eyes, ears, shoulders, elbows, wrists, hips, knees, ankles, and neck (Figure 2.4). In particular, for each detected human individual the set of 2D keypoints on the image is defined as follows:

$$P_S = \{u_1, v_1, vis_1, u_2, v_2, vis_2, ..., u_T, v_T, vis_T\} \tag{2.1}$$

where (u_i, v_i) are the image coordinates of keypoint i, vis_i are flags indicating whether keypoint i is visible or not, and $T = 18$ is the number of detectable keypoints. Let M be the number or visible 2D keypoints for the person being tracked, with $M \leq T$. The M 2D keypoints P_S are used to compute a set of M 3D keypoints $p_i = (x_i, y_i, z_i)$ and the 2D (x_χ, y_χ) person's position on the floor for each detected person, as explained below.

The proposed people detection system makes two basic hypotheses. First, it is

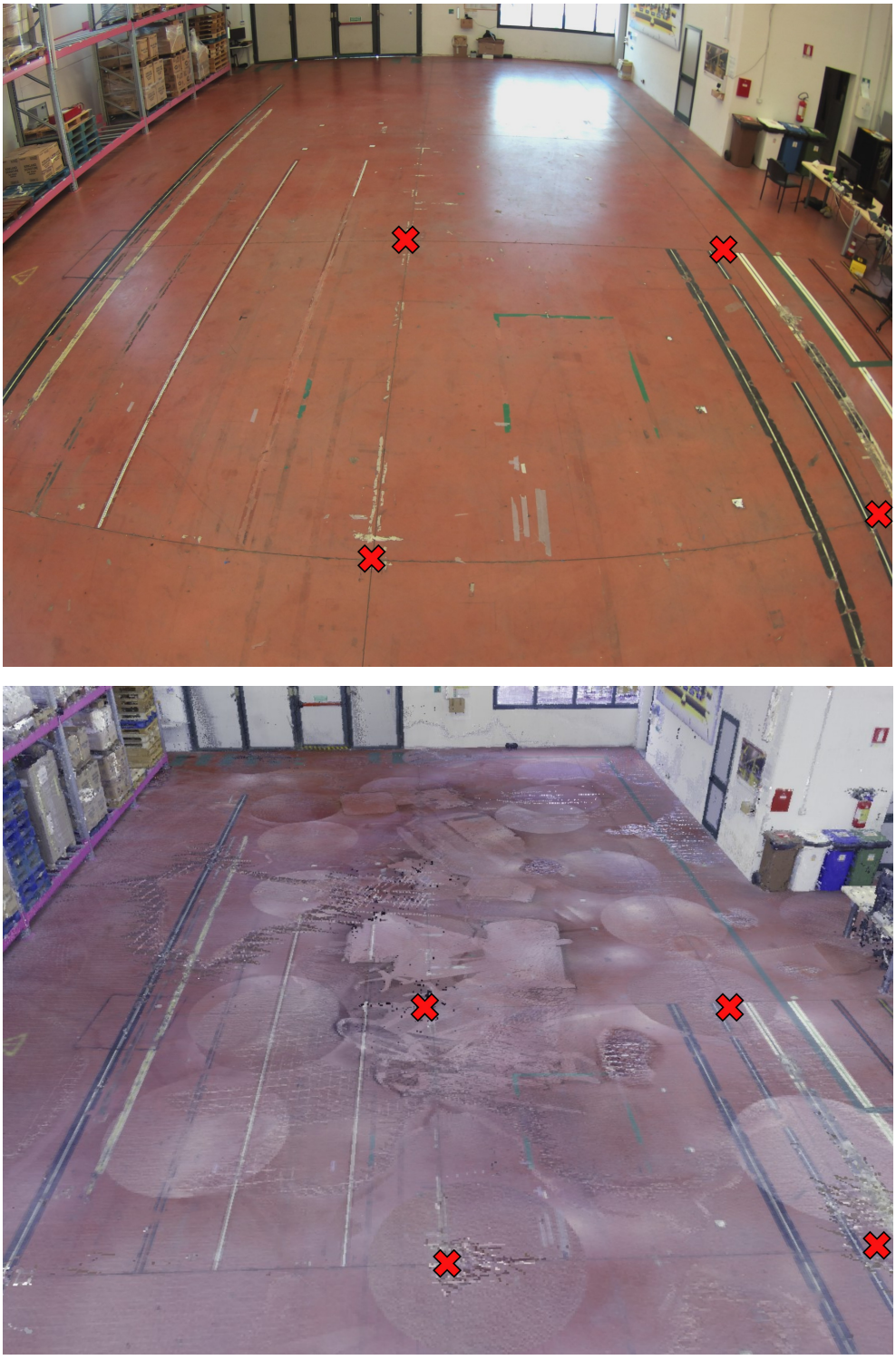

Figure 2.2: Extrinsic camera calibration from 2D image and 3D scan of the environment. Red marks are the selected feature points used for calibration (top: RGB image, bottom: 3D scan).

Figure 2.3: Proposed pipeline for people detection and tracking.

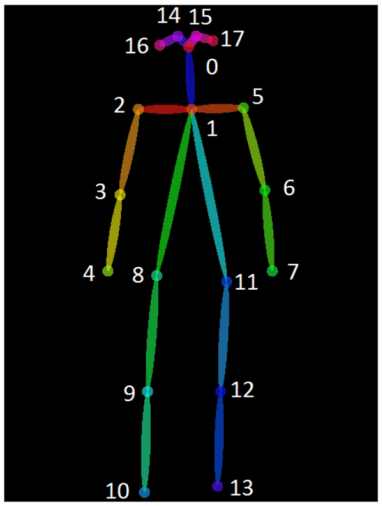

Figure 2.4: Persons' detectable keypoints.

assumed that people are either standing or walking on the floor of the warehouse. Second, it is assumed that the position of a person can be estimated by projecting the midpoint of the ankles on the floor. This second hypothesis is justified by the fact that ankles are closer to the floor with respect to any other body parts detected by TRTPose. Hence, the projection on the floor of the keypoints associated to the ankles should introduce a smaller error. In particular, it is assumed that human ankles are located at a fixed height of 13 cm above the floor. Based on the previous hypotheses in the proposed algorithm a human detection is considered valid only if both ankles are visible. The algorithm determines the position of the person in the world reference frame of the warehouse by computing the 3D coordinates of the mean point $\chi = (x_\chi, y_\chi, z_\chi)$ between the two ankle keypoints. In order to determine the 3D coordinates of χ in the world reference frame, both the extrinsic calibration and the position of the AGV in the world reference frame are used.

Firstly, for each person the midpoint (u_χ, v_χ) of the ankles in 2D image coordinates is computed. The height of point χ in the world reference frame is set at the constant value $z_\chi = 13$ cm. Then, x_χ and y_χ are computed from the standard pinhole

model of the camera:

$$s \begin{bmatrix} u_\chi \\ v_\chi \\ 1 \end{bmatrix} = AR \begin{bmatrix} x_\chi \\ y_\chi \\ z_\chi \end{bmatrix} + At \tag{2.2}$$

where s is the scale factor, A is the camera intrinsic matrix, R and t are the rotation and translation components for the conversion from world coordinates to camera coordinates. The scale factor s is given by the third row in (2.2). Coordinate values x_χ and y_χ can, therefore, be computed from:

$$\begin{bmatrix} x_\chi \\ y_\chi \\ z_\chi \end{bmatrix} = R^{-1}(sA^{-1} \begin{bmatrix} u_\chi \\ v_\chi \\ 1 \end{bmatrix} - t) \tag{2.3}$$

After the estimation of χ, the 3D person's keypoints (x_i, y_i, z_i) in the world reference frame are obtained from the re-projection of 2D keypoints (u_i, v_i). The following approximation is made: all 3D person's keypoints lie on a plane facing the camera and orthogonal to the floor. To construct this plane I first operate on the 2D floor plane. On this plane, two points P_1 and P_2 are defined in 2D homogeneous coordinates. Point P_1 is the projection on the floor of the midpoint χ of the person's ankles. Point P_2 is the projection on the floor of the camera (Figure 2.5). The homogeneous coordinates (l_a, l_b, l_c) of the line l passing through P_1 and P_2 are obtained as $l = P_1 \times P_2$. It is defined m as the line orthogonal to l passing through P_1, that is:

$$m = P_1 \times \begin{bmatrix} l_a \\ l_b \\ 0 \end{bmatrix} = \begin{bmatrix} m_a \\ m_b \\ m_c \end{bmatrix} \tag{2.4}$$

In 3D the keypoints coordinates of the human body are then constrained to lie on the vertical plane containing m, that is:

$$m_a x_i + m_b y_i + m_c = 0 \tag{2.5}$$

The plane equation (2.5) together with the camera equation

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = AR \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + At \tag{2.6}$$

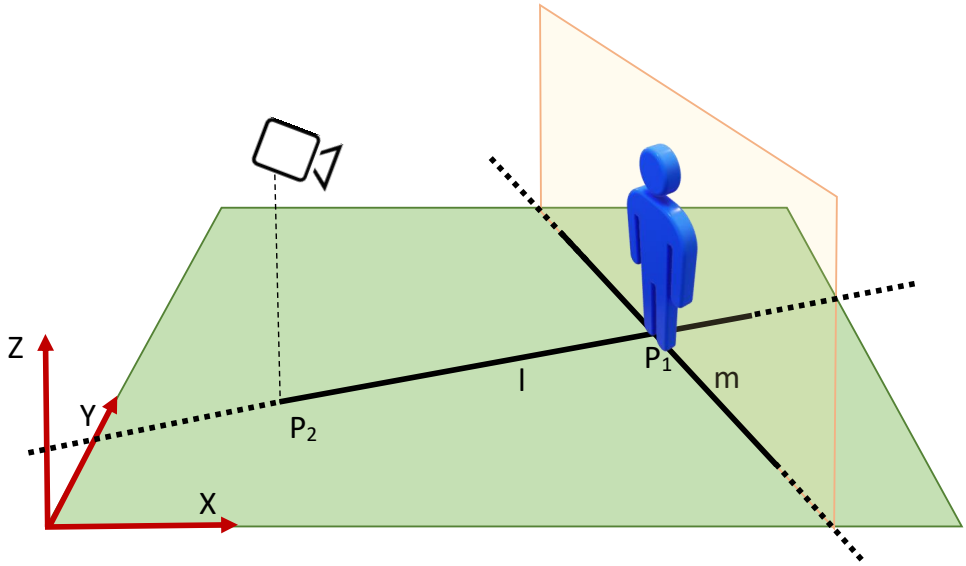

Figure 2.5: Point P_2 is the projection of the camera on the floor (in green). Point P_1 denotes the projection on the floor of the midpoint χ of the person's ankles. Line l passes through P_1 and P_2, while line m passes through P_1 and it is perpendicular to l. The person's 3D keypoints are constrained on the vertical plane containing m.

provide 4 equations in 4 unknown variables: x_i, y_i, z_i, and s_i. A solution can be expressed as a function F_P from 2D keypoints (u_i, v_i) to 3D keypoints p_i:

$$p_i = (x_i, y_i, z_i) = F_P(u_i, v_i, m_a, m_b, m_c, A, R, t) \tag{2.7}$$

2.2.3 AGV local perception: people tracking

A tracker based on the Extended Kalman Filter (EKF) is maintained for each tracked person. Each tracker holds the EKF state and the most recent 3D position of the person's keypoints. The EKF state X_K is:

$$X_K = \left[x_\chi^K, y_\chi^K, v_x^K, v_y^K\right] \tag{2.8}$$

where x_χ^K and y_χ^K are the coordinates of the person on the floor, while v_x^K and v_y^K are the velocities along the two axes. The most recent person's keypoints are expressed as a set of M 3D points $P_K = \left\{p_i^K\right\}$.

In the prediction step, after time step Δt, the new EKF state X'_K is computed by using a standard constant velocity model:

$$X'_K = \left[x_\chi^K + v_x^K \Delta t,\ y_\chi^K + v_y^K \Delta t,\ v_x^K,\ v_y^K\right] \tag{2.9}$$

The same translation is applied to the last detected keypoints P_K, to predict their new 3D position. The observation for the update phase of the EKF is:

$$Z = \left(u_\chi, v_\chi\right) \tag{2.10}$$

i.e., the midpoint of the person's ankles in the image. The observation model is given by the camera pinhole model (2.2). When a new observation occurs, Kalman state X'_K is updated. Observations are associated with trackers according to the Hungarian algorithm. The association is carried out independently for each camera, as multiple observations of the same person may occur, one for each camera. The EKF update is repeated for each associated observation.

The cost function for the Hungarian algorithm is the mean Euclidean distance of the currently observed keypoints $p_i = (x_i, y_i, z_i)$ from the last observed keypoints p_i^K, i.e.:

$$C = \frac{1}{M'} \sum_i \| p_i^K - p_i \| \tag{2.11}$$

Only the M' keypoints that are in common between the last detected keypoints and the currently observed keypoints are considered. If the association cost of an observation is greater than a cost threshold $C_{threshold}$, the association is rejected and a new tracker is created. The EKF state is initialized using the person's position on the floor and an initial velocity equal to zero. The cost threshold is computed as:

$$\begin{aligned} C_{threshold} &= C_{const} + \gamma \\ \gamma &= min\left(\alpha\sqrt{(v_x^K)^2 + (v_y^K)^2}, \beta\right) \end{aligned} \tag{2.12}$$

$C_{threshold}$ is the sum of a constant C_{const} and a variable term γ. The variable term is based on velocities v_x^K and v_y^K, and it depends on two parameters α and β, so that $C_{threshold}$ is higher when the person is moving, to deal with the larger position uncertainty.

A tracker is removed if no association is present for more than T_{lost} seconds. However, in order to reduce the number of false positives, a new tracker undergoes an initial period of T_{init} frames in which it can be removed immediately if it is not associated with any detection. Moreover, the maximum association cost $C_{threshold}$ is fixed to C_{init}, as there is a greater uncertainty in the EKF state.

2.2.4 Warehouse central management system

Periodically each AGV sends information about the tracked people to the central management system, shown in Figure 2.6. The central management system processes information received from each AGV sequentially. Each message contains the current AGV pose and information about all tracked people that were detected by at least one camera, including position, speed and the set of 3D human body keypoints. In the central management system for each person a top level Kalman filter is executed to merge data coming from all the AGVs. The internal state of the top level Kalman is:

$$X = [x^{K^\star}, y^{K^\star}, v_x^{K^\star}, v_y^{K^\star}] \tag{2.13}$$

Data association is performed again using the Hungarian algorithm. In this case, the association cost is the Euclidean distance between the $x^{K^\star}$,$y^{K^\star}$ and x^K,y^K, i.e. the

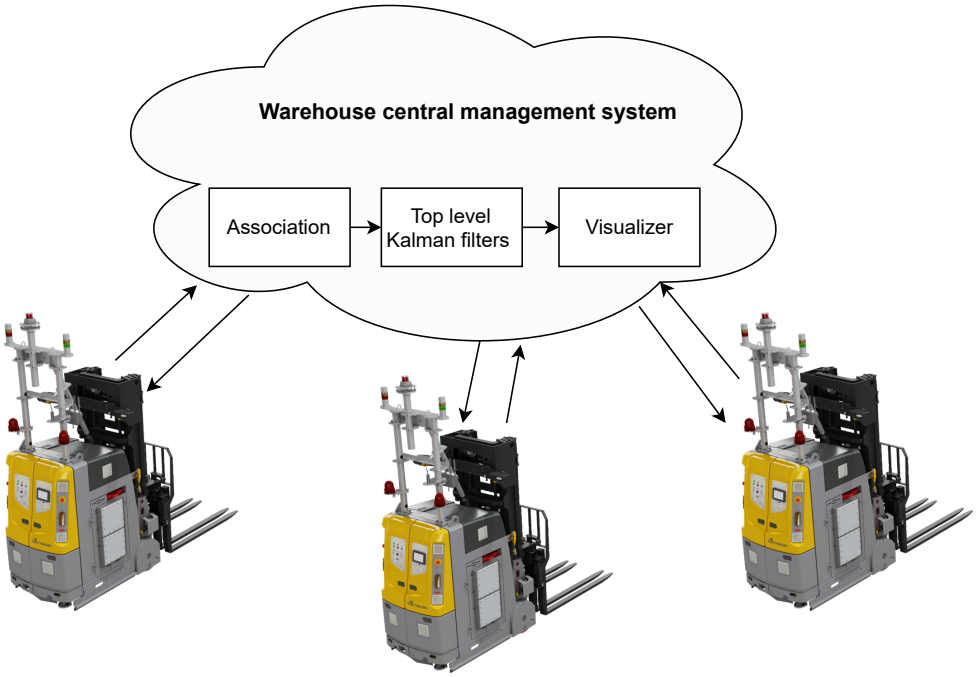

Figure 2.6: Warehouse central management system and AGVs.

positions in the top-level Kalman filters and in the low-level Kalman filters. The cost threshold is computed as in (2.12).

The central management system uses information in the top-level Kalman filters to prevent potential collisions between AGVs and people. A safety area is defined around each AGV. The size of the safety area increases linearly with the speed of the AGV. Using the Kalman filter state, the trajectory of each tracked person is predicted. If the trajectory intersects the safety area of an AGV, the central management system sends back to the AGV the predicted trajectory of that person.

Then, the AGV can take an appropriate action like stopping, reducing the speed or planning an alternative route to avoid collision. The central management system also integrates a 3D visualizer [23] that displays all tracked human skeletons.

2.3 Hardware: Jetson embedded system

To perform the experiments on a real AGV an embedded system for data processing was adopted. After a market availability analysis, the Jetson products offered by Nvidia were selected. The first Jetson platform was the TK1 released in 2014, followed by TX1 and TX2. More recently, the Jetson Xavier and Jetson Nano were released (Figure 2.7). In particular, the Jetson Nano has the lower price (less than 100 $). With its NVIDIA Maxwell 128 Core GPU, a CPU ARM A57 quad-core, and low power consumption (5 watt), it is a good solution for prototyping AI products. The Jetson Nano was used in an early stage of this study.

The Jetson Xavier, in the AGX version, has a GPU Nvidia Volta (512 core) and a CPU ARM v8.2 (8 core). It is slightly more expensive (around 4 times the Jetson Nano) but it is more powerful. Since AGVs have a high cost (about tens of thousands of dollars), the cost of a Jetson Xavier AGX can be considered negligible. Also, this choice allows to have a single device for each AGV on which artificial vision and deep learning processing of various kinds can be performed, not only related to pedestrian detection and tracking. The device is installed on the vehicle and in wired communication with it.

Figure 2.7: Jetson Nvidia production. On the left the Nvidia Jetson Nano, on the right the Nvidia Jetson AGX

Table 2.1: Experimental parameters

Name	Value
T_{lost}	4 s
T_{init}	5

Name	Value
C_{init}	3 m
C_{const}	1 m

Name	Value
α	1.6
β	2 m

2.4 Experiments and results

Experiments have been performed using AGVs developed by Elettric80 SpA. Each AGV carries a Jetson AGX Xavier board, equipped with a set of RGB cameras. The image resolution is 640×480. The Jetson board is mounted at the top of the AGV and the RGB cameras are arranged to cover a large working volume around the vehicle. In particular, three cameras face forward, horizontally spaced 30° apart. A fourth camera faces backward. The top view of the camera placement is in Figure 2.10. All cameras are tilted downward. The onboard computer vision system is shown in Figure 2.8. In the following, the experiments that have been carried out using up to two AGVs are reported. In order to simplify the experimental setup the second AGV was replaced with a fixed computer vision station, comprising only the Jetson board with multiple cameras. Parameter values are reported in Table 2.1. The AGV local perception system works at 7 fps. The working range of the proposed people detection and tracking pipeline is between 1.5 m and 10 m. Example images of the experiments are shown in Figure 2.9. Experiments were repeated across multiple

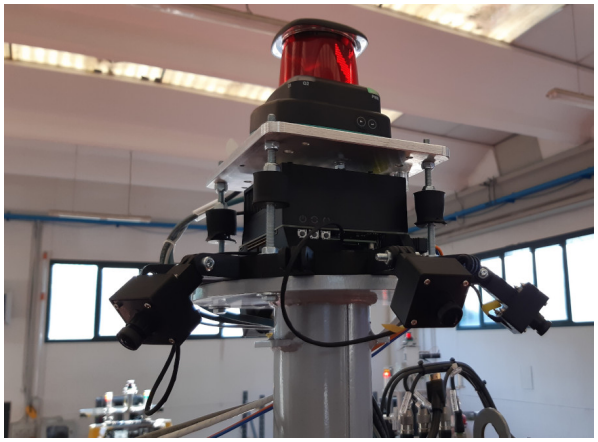

Figure 2.8: Onboard computer vision system on the AGV including a Jetson AGX Xavier board and multiple RGB cameras.

trials for different AGV and person's positions. In the first four experiments the AGV local perception system was evaluated.

In the first experiment, a single AGV is stationary and a person is standing on the floor. The localization error (Euclidean distance between ground truth and tracking result) is evaluated at different positions. Ground truth positions of the person being tracked were obtained from the 3D scan of the environment. In the second experiment, a single AGV is stationary and a person moves along a known segment (back and forth multiple times). The position error is measured as the distance on the floor between the person and the segment. In the third experiment, a single AGV is moving and a person is standing on the floor. The ground truth is again given by a set of known positions of the person on the floor. The AGV moves along both straight and curved paths. In the fourth experiment, a single AGV is moving along a straight or curved path and a person is moving along a straight known segment (back and forth multiple times). In the fifth experiment, the complete pipeline of the proposed system, comprising the two-level hierarchy of Kalman filters, was evaluated: a person moves along a known segment (back and forth multiple times) and the performances of the system are assessed using either a single local perception system (one moving

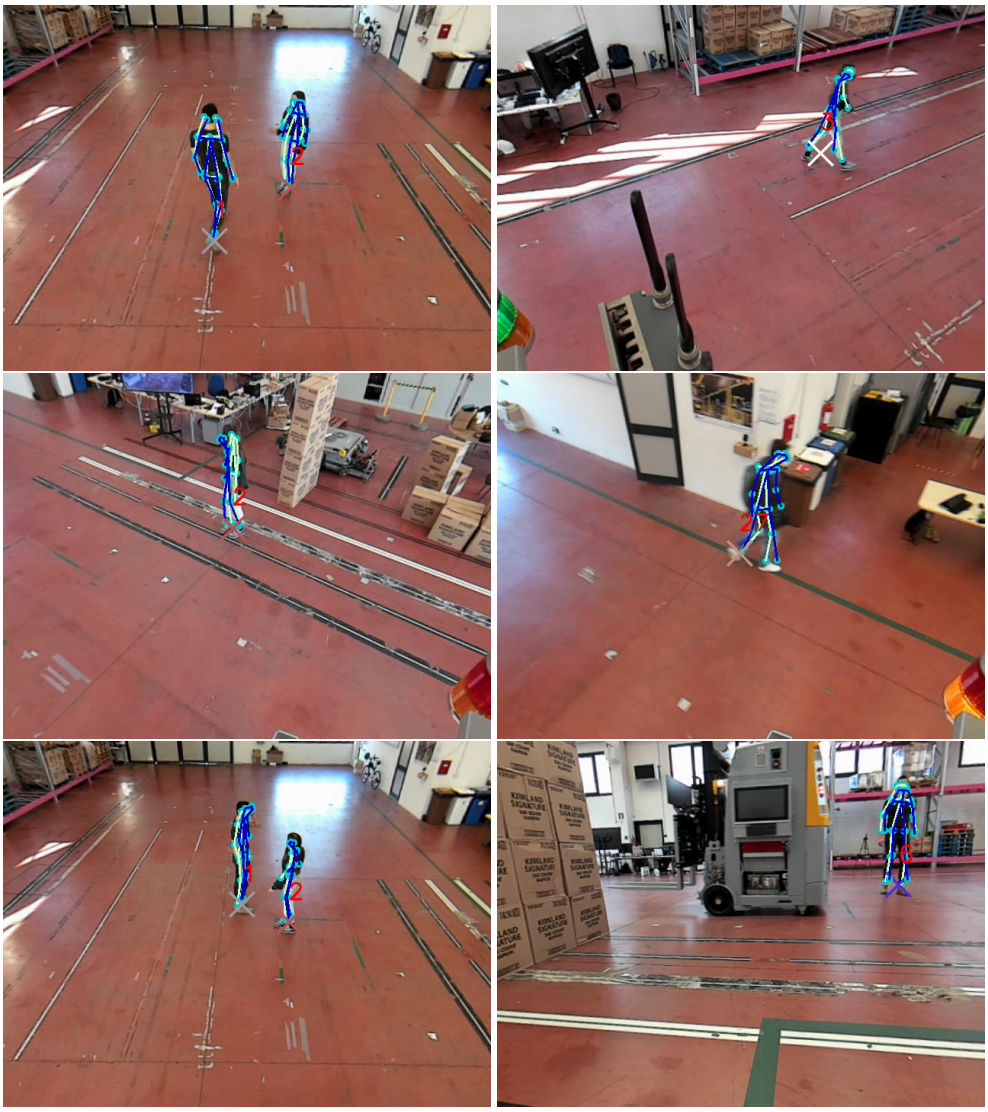

Figure 2.9: Example results of people detection and tracking. 3D human skeletons are displayed for detected people.

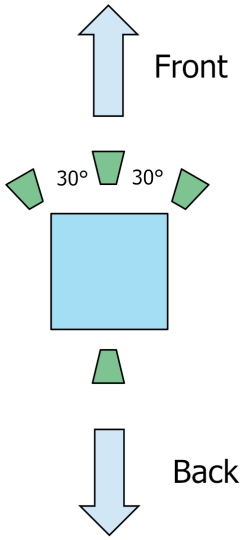

Figure 2.10: Top view of the vehicle (light blue square) showing the placement of the cameras (green polygons).

AGV, experiment 5.a) or two local perception systems (one moving AGV and a fixed computer vision station, experiment 5.b).

The position error (Table 2.2) is about a few tens of centimeters, which is more than acceptable for safe AGV navigation. The average error when the AGV is stationary (experiments 1 and 2) is lower than when the AGV is moving (experiments 3 and 4), probably due to communication delays of the AGV pose or to the reduced performance of TRTPose detection when the camera is moving. In experiment 5, it can be seen that the average position error and standard deviation, provided by warehouse central management system, is lower when two local perception systems are employed (experiment 5.b). Results of experiments 5.a and 5.b are plotted in Figure 2.11.

An additional experiment was carried out to evaluate tracking of more than one person. The performance was assessed by computing the Multiple Object Tracking Accuracy (MOTA)[24] on a video sequence with two people in the scene and a stationary AGV. The MOTA value is 0.96, with 2 identity swaps and 52 false negatives

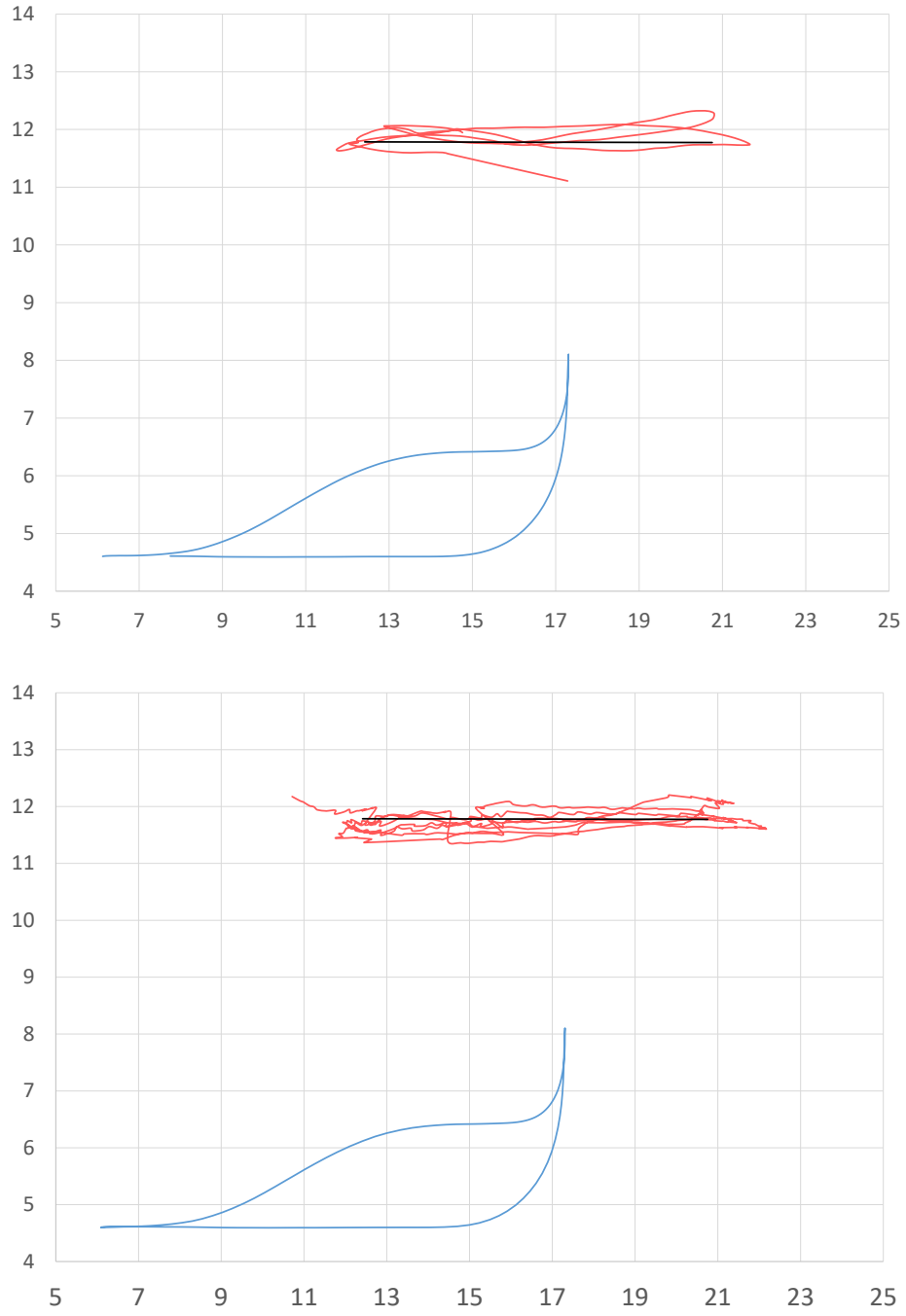

Figure 2.11: Experiment 5.a (top) and experiment 5.b (bottom): AGV path (blue line), person path (red line), ground truth of the person's path (black line). Units are in meters.

Table 2.2: Average position error and standard deviation of people detection and tracking (units are in meters)

	Trials	**Average error**	**Std. Dev.**
Experiment 1	10	0.160	0.010
Experiment 2	6	0.114	0.084
Experiment 3	2	0.333	0.258
Experiment 4	2	0.225	0.201
Experiment 5.a	1	0.206	0.160
Experiment 5.b	1	0.133	0.091

over 1525 frames.

Finally, the system was tested in a potentially dangerous situation (Figure 2.12). A person who is about to cross the path of an AGV is not visible due to an occlusion. However, the person is visible to a second fixed computer vision station nearby. Since the second AGV sends information about the tracked people to the central system (which redirects it to the first AGV), the first AGV stops to avoid the collision.

2.5 Discussion

In this Chapter, a multi-robot system for people detection and tracking in automated warehouses was presented. In the proposed system AGVs are equipped with multiple RGB sensors. Experiments have shown that the current setup can detect people at a distance up to 10 meters from the cameras. It is assumed that people are on the floor of the building.

Figure 2.12: Top: use case scenario of a dangerous situation (the fixed computer vision station is visible inside the yellow circle). Bottom left: the view from the moving AGV (the person is not visible). Bottom right: the view from a fixed computer vision station located behind the person.

Chapter 3

Deep Learning Models for Pallet Detection

3.1 Introduction

In the last decade, deep learning has dramatically improved both speed and accuracy in object detection against conventional computer vision methods. This Chapter addresses the problem of automatic detection of pallets in industrial environments, by using convolutional neural networks (CNNs) and a monocular RGB camera. A pallet is a wooden structure used to move or store goods. Each pallet has two pockets, i.e. two holes where the forks of a forklift truck must be inserted. The purpose of the task is to identify all pallets in a given image frame. Identification of pallets is important in several industrial use cases that involve Autonomous Guided Vehicles (AGVs), and that require scene understanding in unstructured environments. Example applications are collision avoidance of robotic forklifts, picking a pallet from a rack or storing a pallet in a given location.

The proposed pipeline first detects the bounding boxes of relevant elements of a pallet by using a CNN, namely the pallet front side and its pockets. After that, a decision block is applied that exploits such elements to select the final pallet proposals. In particular, to accept a detection of a pallet close to the sensor it is required

that both the pallet front side and its two pockets must be visible. This conservative approach is aimed at achieving safe and reliable AGV tasks. The 2D detection based on monocular vision obtained can be used to verify the presence of pallets during loading task or to count pallets to update warehouse storage information. Also, it is possible to perform pallet loading task with a visual servoing approach. Moreover, I present a comparative study evaluating the performance of three state of the art deep neural networks: Faster R-CNN, SSD and YOLOv4.

One main drawback of Convolutional Neural Networks is the need for many annotated samples (ground truth). Usually, in research papers CNNs are trained and evaluated on standard benchmark datasets. However, industrial computer vision applications have specific needs, such as detection of objects that are not available in standard datasets (e.g. pallets). Also, industrial computer vision applications should adapt to different environment conditions. Therefore, it is also presented a new dataset of RGB images that was acquired from an industrial setting (warehouse). Images of the dataset contain pallets in various configurations, i.e. images may contain multiple pallets at different heights, either on the ground or on racks. Moreover, pallets may have an arbitrary orientation. Furthermore, pallets can be partially covered by a transparent plastic wrap film.

The Chapter is organized as follows. Section 3.2 reviews the state of the art in automatic detection of pallets in industrial environments. Section 3.3 describes the proposed method, while Section 3.4 describes the dataset. Section 3.5 presents the experimental results. Section 3.6 concludes the work.

3.2 Related work

Only a few previous works investigated the application of deep learning models for automatic pallet detection. The closest work related to my research is by Li et al. [25, 26], where the Single Shot MultiBox Detector (SSD) was adopted. Instead of focusing on a single model, I present a comparison among three state of the art deep learning networks. Moreover, in the work presented in this Chapter the neural networks were trained to detect two object classes, i.e. the pallet front side and the pallet

Figure 3.1: Proposed pipeline for automatic pallet detection (from left to right). Each input RGB image (left image) is first processed by a CNN (center image) to detect the front sides of pallets (yellow boxes) and the pallet pockets (red boxes). Then, a decision block is applied to select the final pallet proposals (right image, green color).

pocket, while in [25, 26] the authors considered the pallet as a single object category. Several authors investigated conventional computer vision approaches for automatic pallet detection based on stereo vision and frontal views of the object [27, 28, 29, 30]. Conventional image processing techniques were considered in other approaches using monocular vision with the additional assumption of having a single pallet located on the ground with two visible sides [31, 32, 33, 34]. Molter et al. [35] investigated the use of a time-of-flight camera with pallets placed on the ground.

In [36, 37] two methods were presented that are not robust to lighting conditions as they exploit color information to separate the pallet from the background. In [38] a monocular vision system was developed for autonomous forklift vehicles that detects pallets by looking for rectangular features.

The problem of automatic pallet detection was also investigated by using other sensor types, or sensor data fusion. In [39, 40] pallet detection was achieved by exploiting planar laser scanners. Kita et al. [41] presented a method based on a wide-angle camera to detect pallets on shelves, assuming a frontal orientation of the objects. Baglivo et al. [42] proposed an approach to detect pallets on the floor based on sensor data fusion between a laser and a camera.

3.3 Proposed method

The pipeline of the proposed method for automatic detection of pallets is shown in Figure 3.1. Each input image is first processed by a CNN that detects two object categories, namely the 2D bounding box of the pallet front side and the 2D bounding box of the pallet pocket. The center of the pocket bounding box is a valuable information that can be used to determine the target configuration of the two forklift forks for a pallet retrieval task, that is why the pocket category is needed. Afterward, a decision block is executed that applies heuristic rules to select the final pallet proposals.

3.3.1 Convolutional neural network architectures

To address the pallet detection task I compared three convolutional neural network architectures: Faster R-CNN [43], Single Shot Multi-box Detector (SSD) [44] and You Only Look Once v4 (YOLOv4) [45].

Faster R-CNN has a two-steps approach, also called region-based, i.e. the first part of the network searches for objects within the image regardless of the class, and then it classifies them in the second part. In particular, Faster R-CNN is based on a region proposal concept, that was first introduced with R-CNN [46]. In R-CNN a selective search algorithm was adopted to detect region proposals from the input image. Afterwards, in 2015, Girshick introduced Fast R-CNN [47] that achieved a better performance in terms of computational time, as well as a better accuracy, by generating region proposals directly on a feature map computed on the whole image. Hence, it was no longer necessary for each identified bounding box on the image to go through a feature extraction CNN. However, Fast R-CNN still used the selective search algorithm, which was the actual bottleneck. In 2017, Ren et al. introduced the Faster R-CNN [43], where the selective search algorithm was replaced by the Region Proposal Network (RPN). The RPN is a CNN to detect region proposals.

In summary, Faster R-CNN uses a fully convolutional network for feature extraction that outputs a feature map of the input image. The feature extraction network can be of different types (VGG, ResNet, Inception, etc.). In this work, the backbone for Faster R-CNN is ResNet50. The feature map is used by the Region Proposal Network

(RPN) to predict region proposals. After ROI Pooling, which helps to standardize the proposals, a classifier determines the class of the object.

In this work, Faster R-CNN was trained for 10000 iterations, with a batch size of 2. The learning rate was set to 0.0025, with an initial warm up of 200 iterations. The input images were resized, keeping the aspect ratio, so that the shortest side of each image had a random value among $(640, 672, 704, 736, 768, 800)$ pixels. Moreover, the maximum side size was 1333. The training time took about 12 hours.

While Faster R-CNN is based on a two-steps approach, SSD and YOLO propose a one-step object detection network. In particular, SSD does not require region proposals. In each location of the feature map, a certain number of bounding boxes of different size and aspect ratio are generated to be evaluated. Moreover, SSD generates feature maps at different resolutions to facilitate prediction at various scales. The network architecture of SSD consists of a first convolutional extraction network of the feature map, followed by some convolutional layers that obtain multi-scale feature maps, and a final part that generates the estimated offset and the confidence for each class. The original paper [44] uses VGG16 as feature extraction network, but in this work it was replaced by HarDNet85 [48]. In this work, SSD training went over 150 epochs. Each epoch goes over all the training set once. The images of the dataset were initially resized to a fixed resolution of 512×512 pixels. The batch size was set to 8 and the learning rate was 0.004, with a decay factor of 0.0001 and a momentum of 0.9. The training took about 8 hours.

The third convolutional neural network used in the proposed comparison is YOLOv4. YOLO was introduced in its first version in 2016 by Redmon et al. [49]. In YOLO each input image is divided into a grid and for each cell a certain number of bounding boxes is detected. Despite fast detection, the localization error in YOLO was not negligible. YOLOv2 [50], also called YOLO9000, improved the detection accuracy. It was based on Darknet-19. Darknet is a fast neural network framework written in C and CUDA. However, one of the main weaknesses was the detection of small objects. Then, YOLOv3 [51] was introduced as an incremental improvement. It was based on Darknet-53. Recently, Bochkovskiy et al. introduced YOLOv4 [45], that improves both accuracy and speed with respect to YOLOv3. This version of

YOLO aims at achieving an accurate and fast network that can be used for training on a single and conventional GPU. The authors made several experiments to optimize the components of the CNN architecture. Furthermore, they made additional improvements as the introduction of the Mosaic and Self-Adversarial Training (SAT) data augmentation methods, and the Cross mini-Batch Normalization (CmBN). The backbone is CSPDarknet53. In this work, the input images for YOLOv4 were initially resized to 608×608. Training went over 24000 iterations, with a batch size of 16. It took about 23 hours. The learning rate was set to 0.001, with a decay of 0.0005 and 0.949 momentum.

3.3.2 Decision block

After CNN evaluation the detected front sides of pallets and pallet pockets are used to select the final pallet proposals by applying a decision block. The heuristic rules used by the decision-making step are as follows. If a detected pallet front side is greater than an area threshold, a pallet proposal is created only if the pallet front side contains exactly two pockets, otherwise it is discarded. In this case, the pallet proposal consists of the pallet front side and its two pockets. The area threshold was fixed to 150×10^3 pixels. This conservative decision rule is motivated by the fact that to perform safe pallet forking operations, for close and approximately frontal pallets, it is essential to recognize all the pallet components. Indeed, it is more advisable to not accept a pallet even if it is actually present (false negative), rather than generating a false positive that can lead to dangerous situations, like collisions between the AGV and the environment.

If a detected pallet front side is smaller than the area threshold, the decision block always accepts it as a pallet whether or not it contains the two pockets. Any pocket contained in the pallet front side is included in the pallet proposal. This second decision rule is motivated by the fact that for AGV navigation it is important to model all potential pallets. A pocket is considered inside a pallet front side if the intersection between their bounding boxes is at least 80% of the pocket bounding box area.

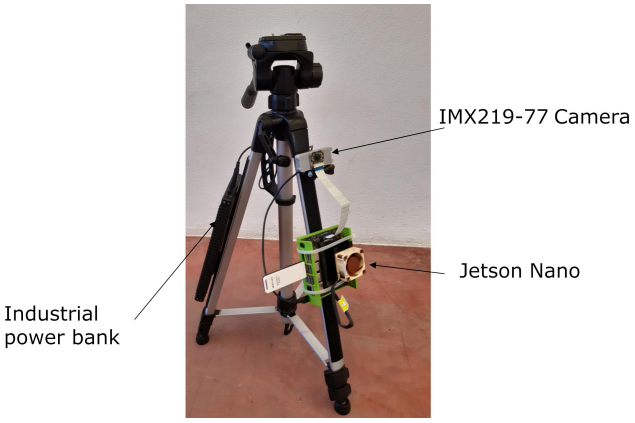

Figure 3.2: Data acquisition system.

3.4 Dataset

A dataset was collected in the warehouse of a mineral water company. The goods on the pallets were bottles and raw materials. A monocular RGB camera (3280×2464 resolution) was adopted, connected to a Jetson Nano embedded system (introduced in Section 2.3) mounted on a tripod. Figure 3.2 shows the set up for data acquisition, Figure 3.3 shows the acquisition software and the tablet used to control the acquisition. The camera was placed on top of a height-adjustable tripod. The acquired dataset contains 1344 images, and it was split into a training set (991 images) and a test set (353 images). Images were taken at various distances from pallets. Moreover, pallets were observed in different configurations: stacked, on the floor, on racks and from the point of view of a forklift that performs a forking operation.

The dataset was generated under different lighting conditions, i.e. some images were taken during the morning under natural light, while other images were taken during the evening when the warehouse artificial light was predominant. In addition, a flashlight was used in some cases to illuminate the pallets close to the camera. The dataset was manually annotated to collect information about two classes: pallet front side and pallet pocket. Table 3.1 reports the number of annotated objects for each class. Furthermore, Figure 3.4 shows example images of the dataset.

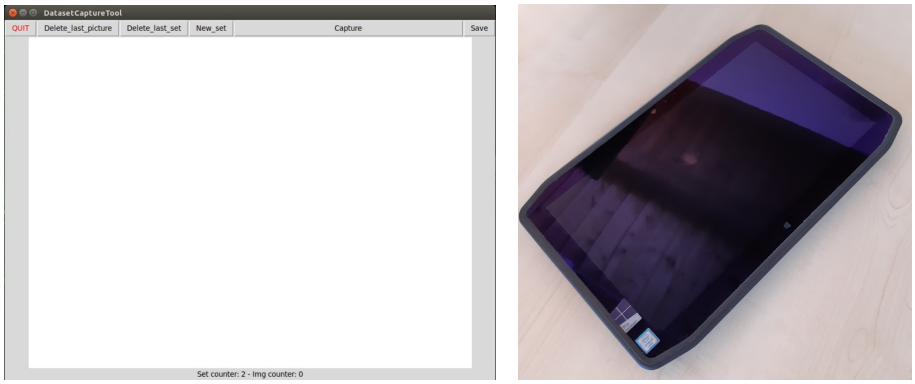

Figure 3.3: Data acquisition software (left) and tablet to control image acquisition (right).

Multiple images were acquired from the same camera pose, under different lighting conditions. A subgroup is defined as a set of images taken at the same camera pose. Since the pallets configuration did not change within each subgroup, the total number of images that had to be annotated was drastically reduced, i.e. one image from each group. Some images within the same subgroup were taken by dazzling the camera using the flashlight to make the scene darker, other images were taken by pointing the camera to the ground floor. Finally, some images were collected by illuminating the transparent plastic film wrapping the pallets. Figure 3.5 shows example images taken from the same subgroup. Information about subgroups of images is reported in Table 3.2. Subgroups have been organized into groups. A group contains all subgroups that have a similar distance of the camera to the closest pallet in the image, as well as a similar height of the camera from the ground. Group 6 is a special case as it is composed of a single subgroup, where the camera pose and the distance from pallets were changed across images.

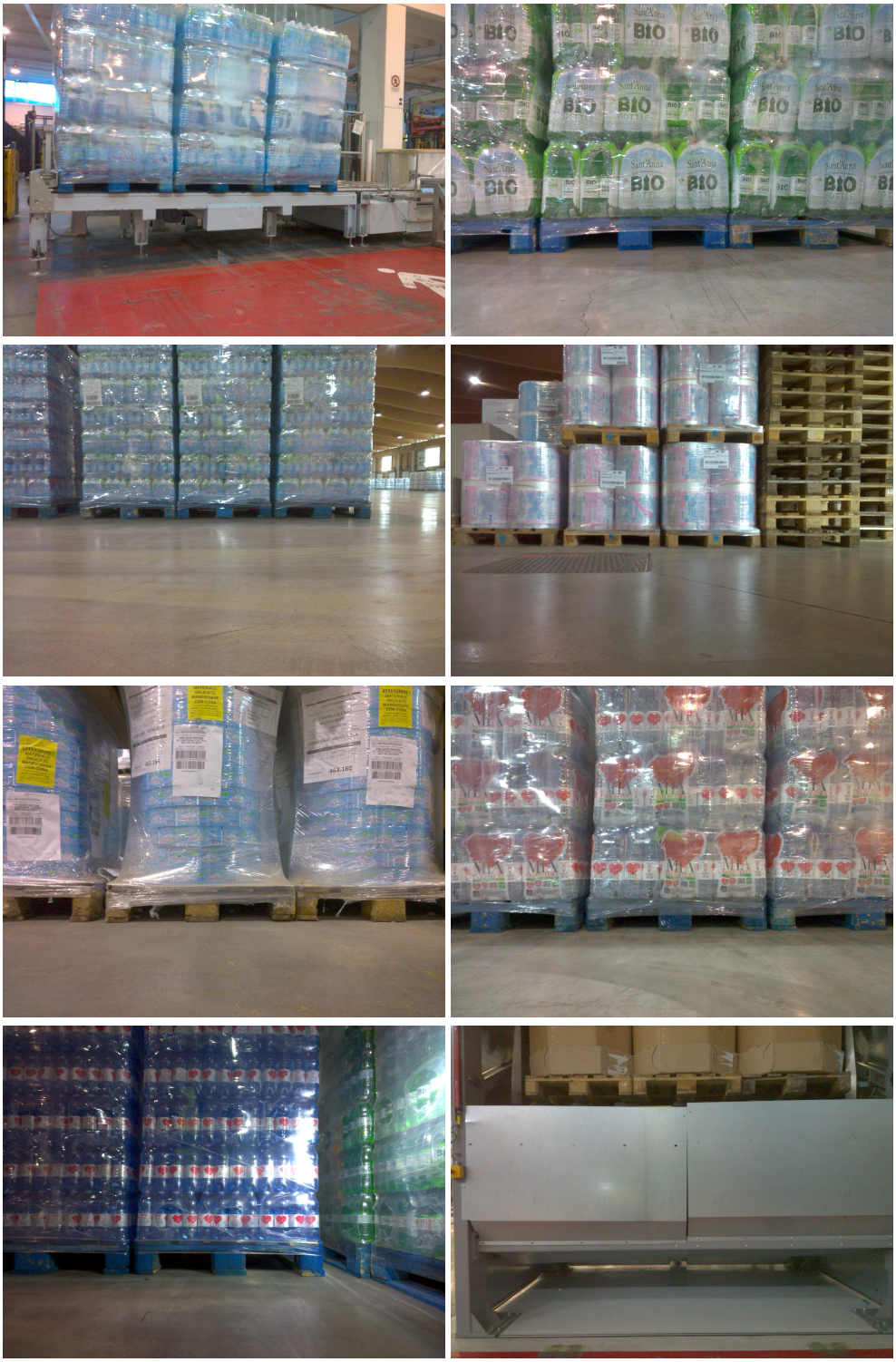

Figure 3.4: Example images of the dataset in an industrial warehouse.

Figure 3.5: Sample images within the same subgroup (taken at fixed camera pose) under different lighting conditions and time of day.

Table 3.1: Number of annotated objects for each class.

	Training set	**Test set**
Pallet front side	5110	1800
Pocket	10200	3600
Total	15300	5400

Table 3.2: Dataset organization in term of groups and subgroups.

Group name	**Distance**	**Height**	**Number of subgroups**
Group 1	150 cm	27.5 cm	68
Group 2	300 cm	27.5 cm	35
Group 3	250 cm	93 cm	24
Group 4	300 cm	151 cm	12
Group 5	300 cm	27.5 cm	12
Group 6	variable	95 cm	1

3.5 Experiments

3.5.1 CNNs detection results

The acquired dataset was used to train the three network architectures. The dataset was augmented using traditional techniques to improve generalization and robustness to many possible situations in a warehouse. The applied techniques for data augmentation involve horizontal flip, random crop, random rotation, random brightness change. A transfer learning method was applied. In particular, training started from a model pre-trained on the MS COCO dataset [52]. Also, some hyperparameters were fine-tuned to improve the performance as much as possible. For the training, a Nvidia GeForce GTX 1080 Ti GPU was used. Results have been collected and analyzed using the COCO metrics.

As shown in Table 3.3, for the intermediate results of the proposed pipeline, the CNN networks that achieve the highest average precision (AP) values are Faster R-CNN and SSD. Nevertheless, YOLOv4 achieved a better detection of objects observed at large distance. In the COCO metrics objects are considered as *small* if their bounding box area is lower than 32×32 pixels. The test set contains 46 small objects (only pockets) observed at large distance, i.e. less than 1% of the total test instances. The Faster R-CNN network was not able to detect small objects as both AR_S (average recall of small objects) and AP_S (average precision of small objects) are equal to 0. In general, it can be noticed that the AP is higher for the pallet front side category than for the pallet pocket category. This result can be explained by the fact that the front side of a pallet has more peculiar features than pallet pockets. Indeed, pallet pockets usually appear as dark regions, but sometimes they are less dark when the camera sees the background.

In terms of computation time Faster-RCNN can process images at 15 fps, while SSD works at 35.44 fps, and YOLOv4 at 27.8 fps. However, in industrial AGV applications the need for increased efficiency in image processing is not as important as the need for accuracy in pallet forking operations, and safety in robot navigation.

Some detection results obtained using Faster R-CNN are shown in Figure 3.6. The axis oriented bounding boxes enclose the detected objects. It can be seen that

the front sides of pallets close to the camera are properly detected. Moreover, pocket detection is robust even in the presence of wrapped pallets.

Table 3.3: COCO metrics for CNNs detection of the front sides of pallets and pallet pockets.

	Faster R-CNN	**SSD**	**YOLOv4**
AP	75.4	75.8	69.1
AP_{50}	93.8	92.0	86.3
AP_{75}	89.7	88.0	82.1
AP_S	0.0	6.5	14.3
AP_M	37.7	40.5	18.3
AP_L	78.8	78.3	73.6
AR_{max1}	12.6	12.6	11.7
AR_{max10}	53.3	53.6	49.4
AR_{max100}	82.1	80.4	78.0
AR_S	0.0	6.3	18.4
AR_M	53.4	50.9	31.1
AR_L	84.8	82.7	82.1
Pallet front side AP	80.1	81.5	69.5
Pocket AP	70.8	70.1	68.7

Figure 3.6: Examples of detected front sides of pallets and pallet pockets using Faster R-CNN.

Table 3.4: COCO metrics for detection of pallets after the decision block.

	Faster R-CNN	**SSD**	**YOLOv4**
AP	78.4	77.9	72.0
AP_{50}	93.2	90.5	87.7
AP_{75}	92.0	89.8	86.1
AP_S	–	–	–
AP_M	28.7	17.9	14.0
AP_L	79.1	78.9	72.7
AR_{max1}	25.0	23.7	23.1
AR_{max10}	74.1	70.5	68.4
AR_{max100}	85.3	81.6	79.7
AR_S	–	–	–
AR_M	33.4	20.2	19.6
AR_L	86.1	82.5	80.7

3.5.2 Decision block

Table 3.4 reports detection results of pallets after the execution of the decision block, at the end of the proposed pipeline. In the proposed task the AP_{75} score (average precision for objects with Intersection over Union threshold of 0.75) is particularly relevant because it is a strict metric that is useful when high accuracy is demanded. In general, experiments confirm that Faster R-CNN and SSD perform better that YOLOv4. Moreover, Faster R-CNN achieves slightly better results than SSD. Also, it can be noticed that the average precision is significantly higher for COCO large objects (AP_L metric) than for COCO medium objects (AP_M metric), for all three CNNs. The average precision and recall of COCO small objects (AP_S and AR_S metrics) are not available because all small objects were removed by the decision block.

Figure 3.7 shows examples of detected front sides of pallets and pallet pockets before the decision block, and the pallet proposals after the decision making step. The decision block clearly helps to filter out the results. There is a case (top row) where a front side pallet larger than the area threshold contains only one pocket, and therefore it is discarded. Moreover, Figure 3.7 shows examples of pockets that are not contained in any pallet (middle and bottom row).

3.6 Discussion

In this Chapter, a method for automatic pallet detection based on a monocular vision system was proposed. A conservative approach was pursued taking into account recognition of both the pallet front side and its two pockets. Three convolutional neural networks were compared on a novel dataset acquired in a warehouse. Experimental results indicate that Faster R-CNN and SSD perform better than YOLOv4. The proposed pipeline allows to perceive the presence of a pallet during pallet loading and automatically count pallets on racks to update storage information. Also, it can also be used to accomplish the pallet forking and loading task using a visual servoing approach. Another approach for pallet forking could be to first estimate the 6D pallet position with respect to the sensor and then with respect to the vehicle. This problem is addressed in the following Chapters.

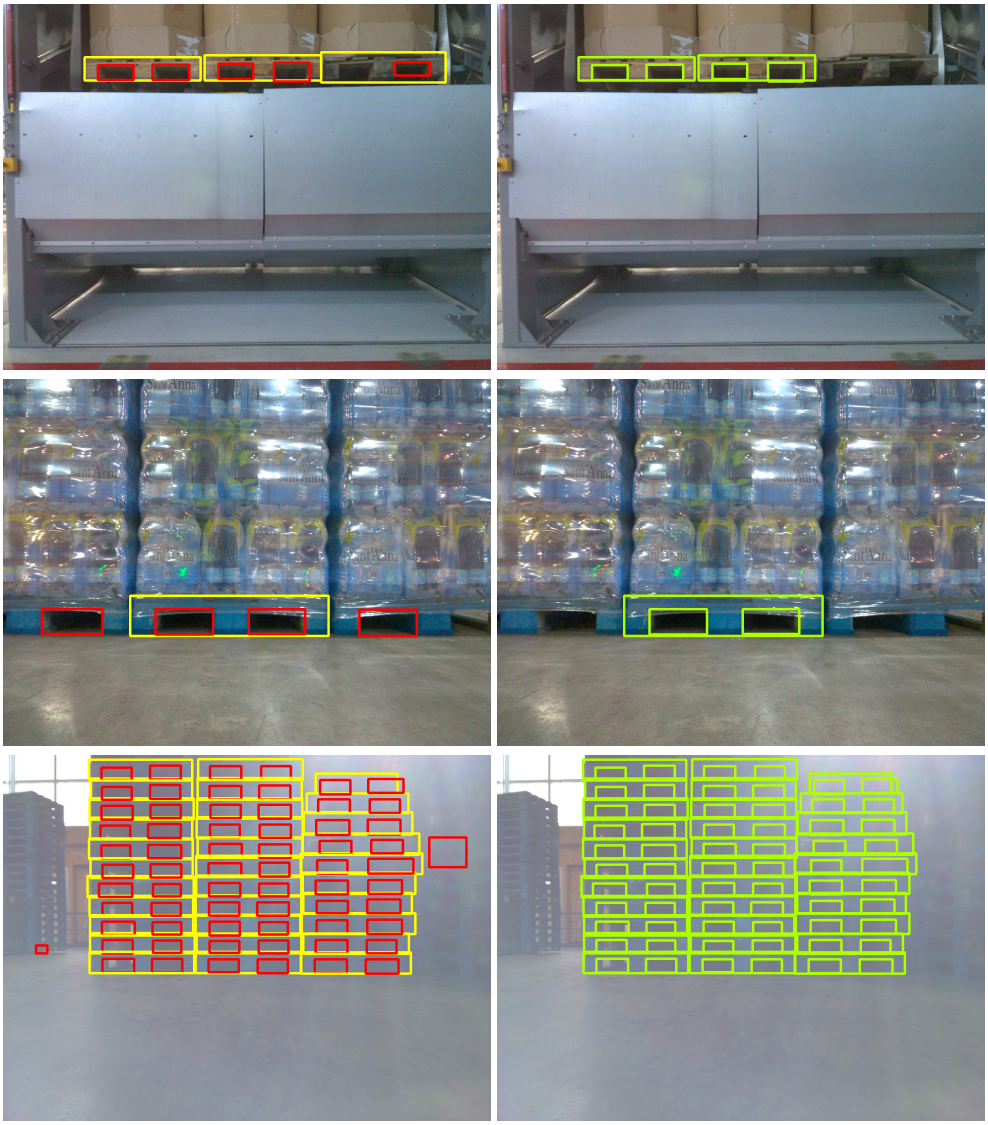

Figure 3.7: Left column: examples of detected front sides of pallets (yellow bounding boxes) and pallet pockets (red bounding boxes) before the decision block. Right column: pallet proposals (green) after the decision-making step. Pockets inside the detected pallets are also displayed in green.

Chapter 4

Recognition and Pose Estimation of Pallets with a Stereo Camera

4.1 Introduction

As introduced in Chapter 1, one of the main tasks of AGVs in an automated warehouse is pallet forking and loading. Chapter 3 presented a method for the detection of pallets and pockets that, among other purposes, can be used to automatically load a pallet through a visual servoing approach. However, another approach is to directly estimate the 3D pallet pose and size. The solution presented in this Chapter is to use a pair of cameras as a stereo head mounted on the vehicle. A stereo camera allows to reconstruct the 3D point cloud of the scene by positioning the two cameras close to each other, at a fixed distance called baseline. However, the information obtained from the 3D reconstruction must be post-processed in order to make it usable in practice. This Chapter presents a system for the detection and pose estimation of pallets based on a stereo camera, without any assumption on the pallet pose or size.

The Chapter is structured as follows. Section 4.2 presents some existing works, while Section 4.3 presents the pallet detection and localization system proposed. Experiments are reported in Section 4.4, and Section 4.5 contains a final discussion.

4.2 Related work

In literature, there are several works about pallet handling. Some works are based on laser sensors. For instance, in [53] the authors present a detection, localization and tracking system based on 2D range data.

Tsiogas et al. [54] present a complete system for pallet loading based on a monocular camera and a planar laser scanner that leverages the detection of the legs of the pallet to extract the pose. However, the width and height of the pallet have to be known in advance to cluster the legs of the pallet into an instance of a pallet object and in order to estimate the pose. Molter et al. [35] present a method based on a time-of-flight camera for detecting and localising pallets rotated up to 90° with respect to the camera plane. The point cloud is processed and filtered to determine the vertical plane of the pallet front side. Then, geometric knowledge of the pallet structure (euro pallets) is exploited to compute the pose. Hence, also in this case, the pallet size must be known in advance.

There are also some works based on stereo cameras. Varga et al. [30] introduced a method for pallet handling based on a stereo camera. The detection is performed on a monocular image with a sliding window approach and then a plane is fitted on the point cloud obtained from the stereo reconstruction. The work was then improved in [29, 55]. However, some a priori information, such as pallet sizes, was exploited. Furthermore, the detection is performed with a traditional computer vision method.

The method proposed in this Chapter is a flexible approach that leverages deep learning for 2D detection on RGB images and it uses the point cloud reconstructed from the stereo camera to define the pose of the pallet without the assumption that the pose, dimensions or shape of the pallet are known.

4.3 Method

A system for recognition and pose estimation of pallets based on a stereo camera and a neural network to perform pallet detection is proposed. An overall scheme is reported in Figure 4.1. The method works also for multiple pallets. The system is

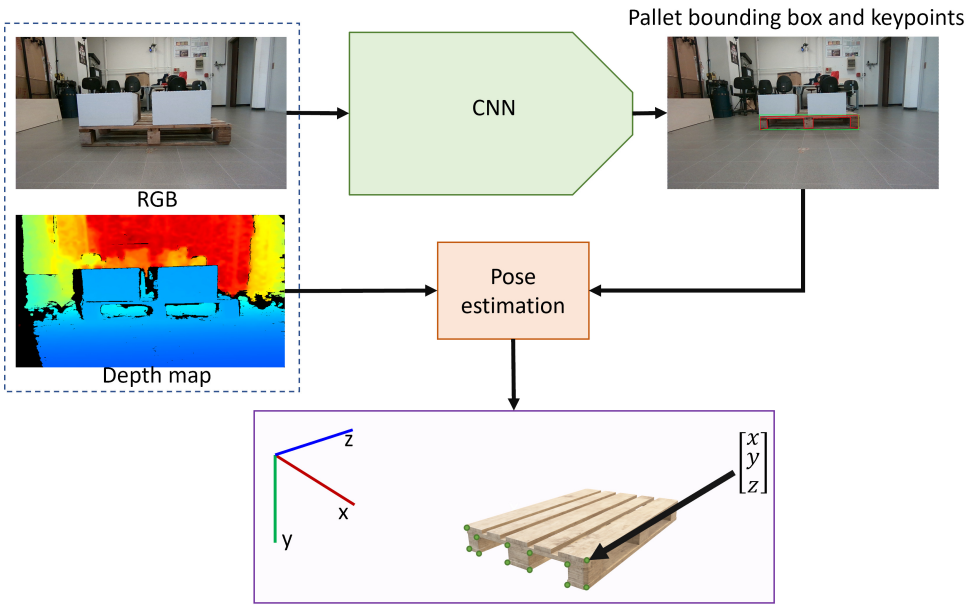

Figure 4.1: Overall schema of the 3D pallet detection and pose estimation system.

deployed to work on a Jetsons Xavier AGX.

The pipeline of the system is composed of the following steps: image acquisition, 3D stereo reconstruction, pallet detection and pallet pose estimation.

4.3.1 Image acquisition and stereo reconstruction

The stereo camera used is a D455 produced by Intel (Figure 4.2). The size of the camera is 124 mm $\times$ 26 mm $\times$ 29 mm. It has 2 grayscale cameras to compute the depth and a central global shutter RGB camera. The central RGB camera can work up to 30 fps with a maximum resolution of 1280×800. The two grayscale cameras can compute the disparity map with a framerate of 60 fps and a maximum resolution of 1280×720. The optimal working range goes from 60 cm to 6 m. The camera acquires RGB images and it has a baseline of 95 mm. The image resolution was set to 1280×720. The camera comes with an SDK that implements several functions.

Figure 4.2: Intel D455 stereo camera

Initially, the calibration information must be extracted. The intrinsic parameters are directly obtained through the Intel SDK. Instead, to determine the position of the camera in the world reference frame, the extrinsic parameters are obtained with the same calibration procedure presented in Chapter 3. To obtain the point cloud of the scene the steps are: stereo image rectification, stereo matching, and stereo reconstruction. All these steps are performed directly through the Intel SDK. The RGB image and the 3D point cloud in the camera reference frame are obtained.

4.3.2 Pallet Detection

The detection is performed on the RGB image with a convolutional neural network. In particular, unlike the bounding box pallet detection presented in Chapter 3, the trained neural network detects both pallets bounding boxes and their relevant keypoints. The 12 corners of each pallet were considered as keypoints, as shown in Figure 4.3. The neural network architecture presented in [56] is leveraged for the bounding box and keypoints detection. It is a Mask R-CNN variant, that adds a final branch for keypoints detection. Training was done on an extended version of the dataset presented in Chapter 3, which includes images acquired in a different warehouse. The dataset was acquired in a frozen food company warehouse and it contains 869 images with a 4000×3000 resolution. The final dataset is composed of 2213 images, 1642 used for training and 571 for testing. During the annotation phase, in each scene for each pallet, all the positions of the 12 keypoints were determined and saved in COCO format. Example images of the extended dataset are in Figure 4.4. In particular, for each

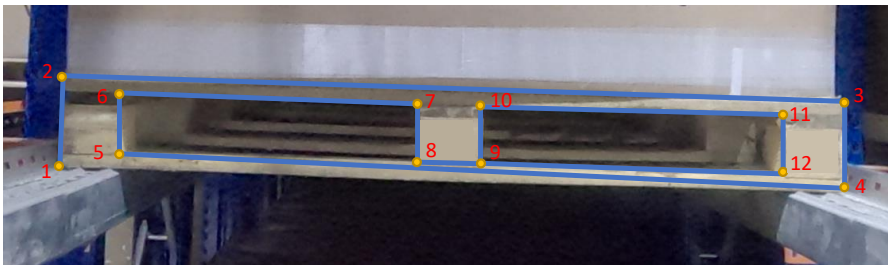

Figure 4.3: Pallet keypoints (in yellow).

detected pallet, the set of the 2D keypoints on the image is defined as follows:

$$K_{2D} = \{u_1, v_1, vis_1, u_2, v_2, vis_2, ..., u_{12}, v_{12}, vis_{12}\} \tag{4.1}$$

where (u_i, v_i) are the image coordinates of the keypoint i and vis_i is a flag indicating whether the keypoint i is visible or not. In this work, a pallet is marked as detected only if all 12 keypoints are visible.

4.3.3 Pose estimation

Thanks to the stereo reconstruction there is correspondence between each 2D pixel and its 3D coordinates:

$$P(\begin{bmatrix} u \\ v \end{bmatrix}) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = p \tag{4.2}$$

As sometimes the disparity map has some holes due to missing or wrong matching between the two images, the value could not be available for some pixels.

After the 12 keypoints of the pallet are detected on the 2D image, it is not possible to directly use the corresponding 3D coordinates of that pixels to compute the pallet pose. In fact, because of errors and missing information on the depth map, the approach would be unpredictable. Instead, as usually done in other works as [29, 35], a plane on the point cloud is fitted. However, unlike other works, the detection on the image provides not only a bounding box of the pallet, but also complete information on the structure of the pallet. This is exploited to create a binary mask that considers

Figure 4.4: Example images of the dataset. Some brands have been blurred.

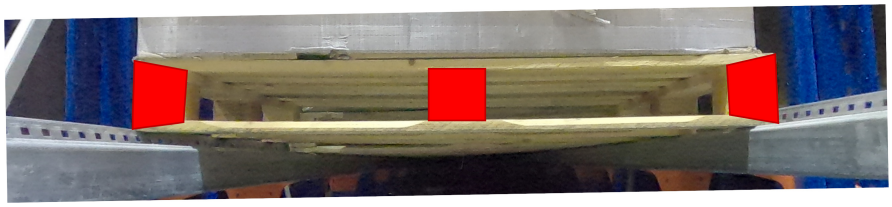

Figure 4.5: Mask on the legs of the pallet.

only the points on the legs of the pallet. The reason for this choice is that, usually, the points on the top of the front side are bad in the point cloud, while the bottom is not always present. The mask is shown in Figure 4.5 and it considers the rectangle between the keypoints $(1,2,6,5)$, $(8,7,9,10)$ and $(12,11,3,4)$. The mask is defined as a set of N points in the image space:

$$M = \{(u_j, v_j)\}_{j=1}^{N} \tag{4.3}$$

The mask is used to filter 3D points from the point cloud, to obtain a set of Q points in 3D:

$$P(M) = T = \{(x_j, y_j, z_j)\}_{j=1}^{Q} \tag{4.4}$$

where $Q \leqslant N$. Subsequently, a RANSAC algorithm is used to fit a plane on the set of points T. To further exploit the keypoints information, the 2D keypoints are back-projected on the 3D world assuming that they lie on the computed plane. The plane equation and the pinhole camera equation are:

$$\begin{cases} ax_i + by_i + cz_i + d = 0 \\ s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = AR \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + At \end{cases} \tag{4.5}$$

where A is the intrinsics matrix, R and t are the rotation and translation, and s_i is the scale value. Solving 4.5, each 2D keypoint (u_i, v_i) can be backprojected in 3D. The 3D coordinates of salient points of a pallet are obtained as follows:

$$p_i = (x_i, y_i, z_i) = F_P(u_i, v_i, a, b, c, A, R, t) \tag{4.6}$$

Table 4.1: COCO metrics on the test set.

Bounding box	
AP	78.20
AP_{50}	96.26
AP_{75}	92.43
Keypoints	
AP	96.16
AP_{50}	96.26
AP_{75}	96.26

where $i \in \{1, ..., 12\}$. A 3D visualizer based on the Point Cloud Library(PCL) is also used.

4.4 Experiments

The CNN uses as backbone a ResNet101. The model was trained for 5000 iterations with a batch size of 4. The resulting COCO metrics for the keypoints evaluation are in Table 4.1. The detection and pose estimation system was tested in two different environments. Firstly the experiments were performed in a laboratory environment, leveraging an OptiTrack motion capture system. Then, further experiments were carried out in a warehouse.

4.4.1 Experiments in a laboratory environment

The system was tested in a laboratory of the RIMLab, Robotics and Intelligent Machines Laboratory, of the University of Parma. The ground truth for the pallet localization was obtained with an OptiTrack, a motion capture and 3D tracking system. Several IR cameras were placed all around the room on the walls towards the center. The system was calibrated and the world coordinate system was placed in the middle

Figure 4.6: Rigid body markers on the back of the camera case.

of the room on the ground. Sphere markers can be used to get the position of specific points in the world reference frame. Hence, the extrinsic calibration is performed by placing four markers on the ground and four markers on the back of the camera. The markers on the camera, shown in Figure 4.6, define a rigid body with a reference frame placed usually in the middle of the markers. The Optitrack software provides rotation and translation of the rigid body. To obtain the translation and rotation from the rigid body reference frame to the camera reference frame, during the calibration procedure the camera is placed toward the ground where the four markers are placed. The procedure exploits the correspondences between the 3D coordinates obtained from the motion capture system and the 2D image pixel coordinates of that points on the image. The calibration set up is shown in Figure 4.7.

The position of the pallet keypoints is extracted by placing four sphere markers on the pallet. In particular, only four more external keypoints positions were evaluated, as shown in Figure 4.8. Some results of the detection are in Figure 4.9. Figure 4.10 shows the results of the pose estimation in the PCL visualizer.

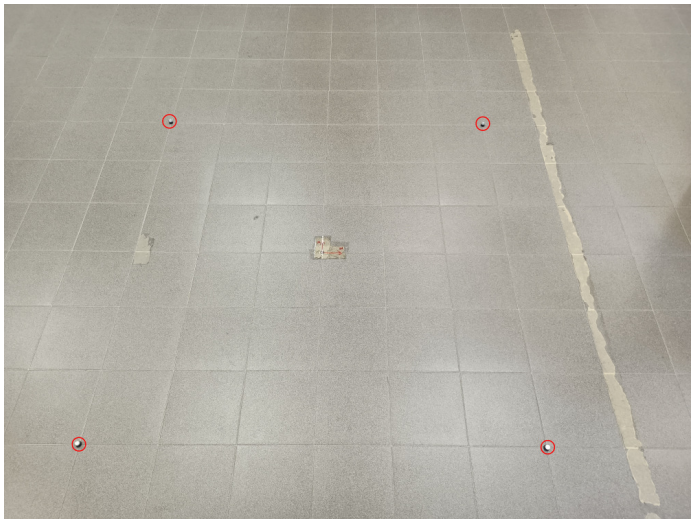

Figure 4.7: Calibration set up. Four sphere markers placed on the ground (highlighted with red circles).

Figure 4.8: OptiTrack marker placed on the pallet.

Figure 4.9: Examples of detections. The green rectangle is the bounding box. The red lines connect the keypoints.

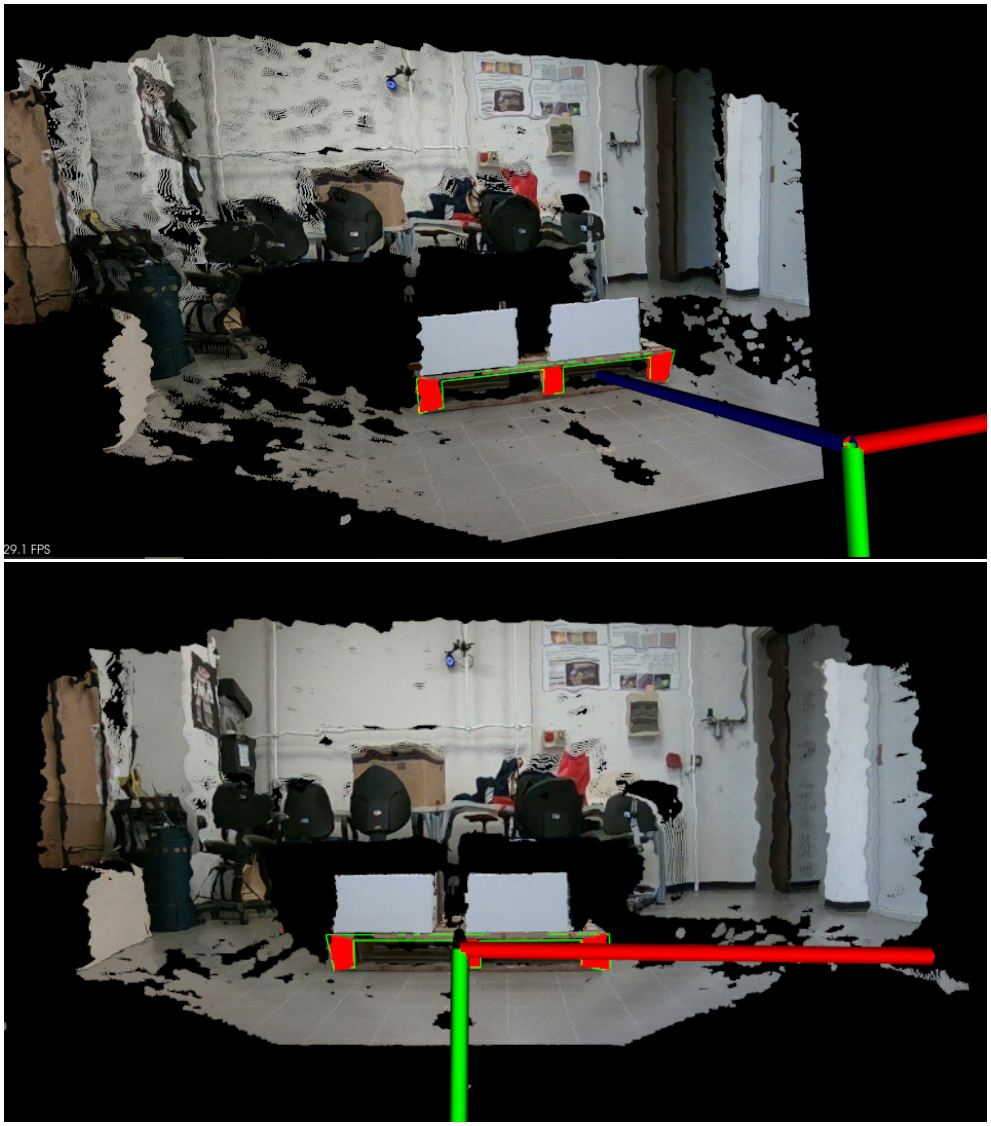

Figure 4.10: Pallet detection and pose estimation in the PCL visualizer.

Table 4.2: Keypoints localization error (m) in a laboratory environment.

Distance	Mean	Standard deviation	Max	Min
1.5 m	0.031	0.007	0.040	0.024
2 m	0.035	0.010	0.046	0.024
2.5 m	0.043	0.011	0.058	0.032
3 m	0.058	0.011	0.075	0.051
Pose 1	0.057	0.022	0.077	0.030
Pose 2	0.042	0.007	0.052	0.035

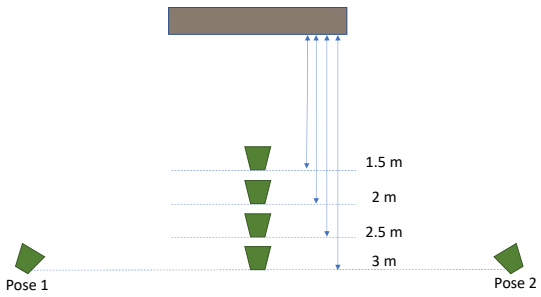

Figure 4.11: Camera configurations for the experiment in the laboratory environment.

The quantitative results of the experiments are in Table 4.2. For each keypoint the Euclidean distance between the estimated position and the ground truth position is computed. The Table presents mean and standard deviation. Also, the min error and max error are reported. The distance from the pallet placed toward the camera was changed from 1.5 m to 3 m. Furthermore, two configurations called "Pose 1" and "Pose 2" are considered. Figure 4.11 shows the configuration of the experiments, where the brown rectangle is the pallet and the green trapezoid is the camera. The error increases for higher distances, and it goes from 3.1 cm at 1.5 m to 5.8 cm at 3 m.

4.4.2 Experiments in warehouse

The system was tested in an E80 Group S.p.A. warehouse, located in Viano, Reggio Emilia, Italy, on a real AGV. The camera was connected to an onboard Jetson Xavier AGX and placed as shown in Figure 4.12. One or more pallets were positioned as if the vehicle should perform a loading task. The extrinsic calibration, to extract the rotation and translation of the camera in the vehicle reference frame, was obtained as explained in Section 2.2.1. The position of the AGV in the world reference frame was assumed to be known. An example of pallet detection is in Figure 4.13. An example of pose estimation with two stacked pallets in the PCL visualizer is in Figure 4.14. Instead, Figure 4.15 shows the same results with a visualizer provided by E80Group S.p.A..

Table 4.3: Keypoints localization error (m) in a warehouse.

Distance	Mean	Standard deviation	Max	Min
2.2 m	0.036	0.010	0.055	0.024
2.6 m	0.024	0.006	0.038	0.015
3 m	0.031	0.019	0.064	0.006
3.4 m	0.047	0.006	0.060	0.040
3.8 m	0.058	0.021	0.095	0.029
4 m	0.043	0.017	0.076	0.024
4.2 m	0.104	0.068	0.233	0.024
4.4 m	0.059	0.027	0.118	0.025
4.6 m	0.090	0.043	0.165	0.024

Table 4.3 shows quantitative result of the keypoints localization. In these experiments, the ground truth is obtained from a 3D point cloud acquired with a terrestrial laser scanner. The distance between the pallet and the camera varies from 2.2 m to 4.6 m. The error increases as the distance increases. For each distance configuration two acquisitions were tested. The results show an error of 2-3 cm at distances up to 3

m. The error increases as the distance increases, reaching 9 cm at 4.6 m.

Figure 4.12: Set up for experiments in an E80 Group S.p.A. warehouse. The camera is in the red rectangle of the top image.

Figure 4.13: Example of detection. The green rectangle is the bounding box. The red lines connect the keypoints.

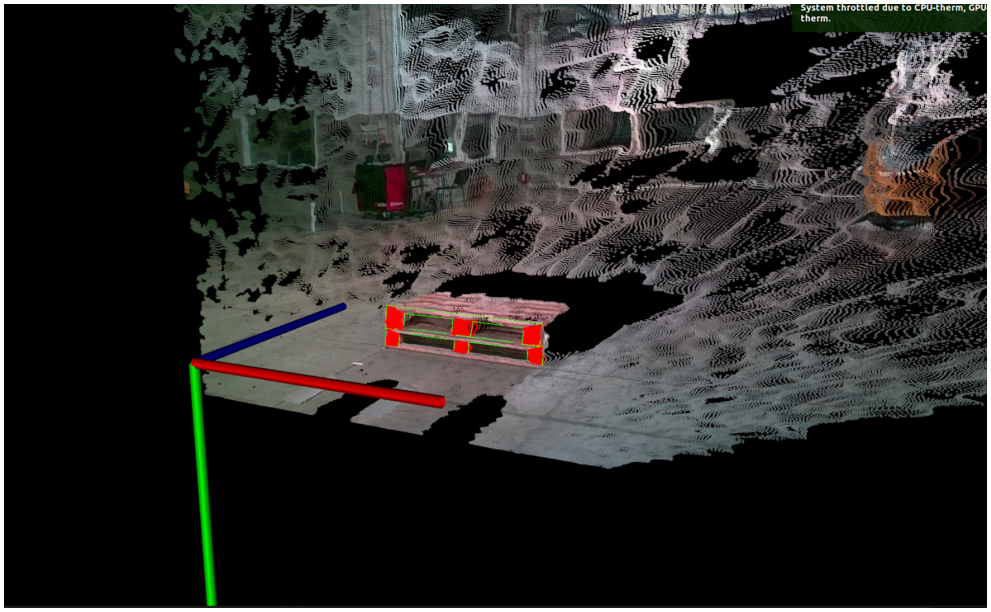

Figure 4.14: Pallet detection and pose estimation in the PCL visualizer

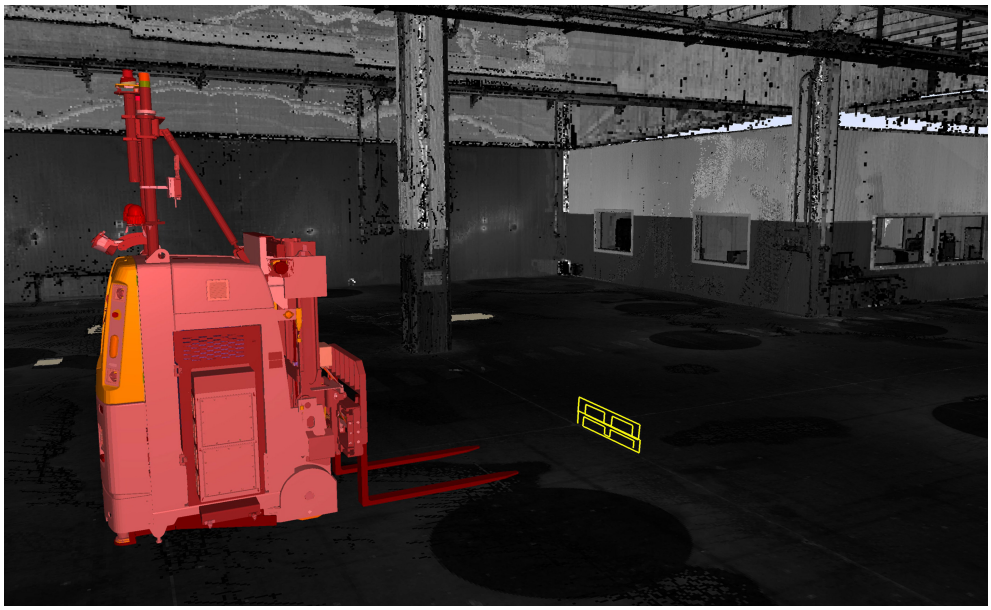

Figure 4.15: Pallet detection and pose estimation in the visualizer provided by E80 Group S.p.A.

The feasibility of the pallet loading task depends on: the size of the vehicle forks, the size of the pallet pockets, the pallet position and rotation error about the vertical axis. In the experiments presented in this Section, the vehicle forks are 1200 mm long, 120 mm wide and 45 mm thick and the pallet (picked up on the longest side) pockets are 797 mm deep, 455 mm wide and 120 mm high. The estimated maximum admissible error for a successful loading operation is 11 degrees for the angle about the vertical axis, 167.5 mm for the horizontal offset of the forks with respect to the center of the pocket, and 37.5 mm for the vertical offset. As the horizontal offset error is about 20 mm, the vertical offset error is about 8 mm and the rotation error is less than 1 degree, the forking task should be feasible using the stereo vision system.

4.5 Discussion

This Chapter presented a system for pallet detection and localization with a stereo camera. The system can be installed on a real AGV. The detection is performed with a CNN that extracts bounding box and 12 keypoints of the pallet in the scene. Then, the pallet pose is estimated. Several experiments to evaluate the performance of the localization were carried out. The system works even with multiple pallets, it recognises both pallet sides and it does not consider any assumption on the pallet shape or pose. The pallet side must not be perpendicular to the ground. The system is flexible as it allows to detect and estimate the pose in variable situations. The light conditions could be a limitation as the pallet must be visible from the camera. An assumption on the perpendicularity of the estimated plane to the ground plane could be done to improve the detection accuracy. Moreover, constraints on the keypoints relative position could improve the detection.

Chapter 5

Self-Supervised 6D Object Pose Estimation

5.1 Introduction

Being able to robustly estimate the 6D pose of an object, *i.e.* the object's 3D orientation and 3D location, is a long-standing problem in computer vision, yet, also a crucial task for many applications, including autonomous driving [57] and robotic grasping [58]. With the advent of deep learning, the accuracy of 6D object pose estimation has recently made a huge leap forward [59]. Nevertheless, there are still many challenges left to be solved. In particular, most current state-of-the-art methods can only handle a handful of objects [60, 61, 62]. To circumvent this issue, category level-based pose estimation has been recently revamped [63]. This is a significantly more challenging problem as it requires to solve for 3 additional unknown parameters describing the 3D size of the object. However, current methods leveraging RGB-D data have proven to be able to robustly estimate the category-level 6D pose, even under large intra-class variations [64, 65].

Despite such great progress, a large amount of real labelled data is typically required to train these models [66]. In addition, data labelling is very time-consuming and labour intensive [59]. Therefore, some methods fully rely on synthetic data [60,

67], though, these approaches still lack far behind the state of the art which uses also real data during training [68]. Interestingly, a new line of work has recently emerged, proposing to leverage self-supervision together with unlabeled real data to circumvent this problem [69, 70]. Thereby, these self-supervised techniques often rely on additional information such as depth data [69] to supervise their approach by means of seeking a misalignment in 3D [71, 72], following an ICP-like paradigm. While these approaches are promising they do not consider multi-frame consistency.

In this dissertation, it is presented a novel self supervised method that ensures consistency between consecutive frames as a mean to obtain better 6D object poses. The proposed method self-supervise the learning of the 6D object pose using the 2D optical flow of two consecutive frames.

To this end, after obtaining an initial pose estimation, the method harness 3D correspondences and differentiable rendering to obtain the corresponding optical flow, as shown in Figure 5.1. Finally, it is compared with the weak ground truth optical flow as obtained by an off-the-shelf optical flow method [73] to enforce the optical flow consistency.

The choice for employing the optical flow to self-supervise the category-level object pose estimation enables working in a more constrained 2D image space, rather than a large 3D euclidean space. Moreover, for many problems in 2D computer vision, including optical flow estimation, there are already many highly elaborate methods available, generalizing well to novel domains [73] [74]

In summary, this Chapter presents the following contributions:

1. A novel self-supervised method for category-level 6D object pose estimation, which leverages the much simpler task of optical flow as proxy.

2. It demonstrates how differentiable rendering can be harnessed to elegantly obtain the optical flow from two 6D pose observations.

3. The method for self-supervised learning yields state-of-the-art performance on the common NOCS benchmark, and it can even reach comparable results with some fully-supervised approaches.

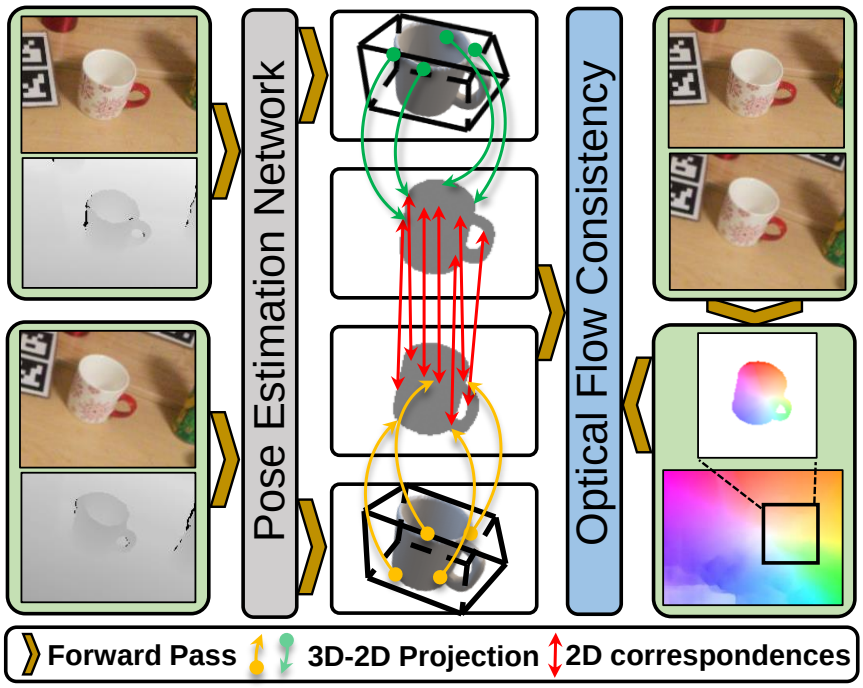

Figure 5.1: Using the 6D estimated pose of an object between two consecutive frames, the correspondences in the 3D space are obtained and then projected onto the 2D image plane. From the projected 2D correspondences it is possible to compute the corresponding optical flow using differentiable rendering and enforce consistency with the optical flow obtained from an off-the-shelf method to better guide training.

5.2 Related work

5.2.1 Instance-level pose estimation

Instance-level 6D pose estimation refers to estimating rotation and translation of a known set of objects with associated CAD models. Recently, many networks have been proposed for pose estimation from RGB images or RGB-D data. As for RGB-only methods, some approaches [75, 76, 77, 78, 79, 60, 80] regress 6D pose directly, while others [81, 82, 62, 83, 67, 84] employ 2D-3D correspondences via keypoint detection or dense coordinate regression to solve for the pose, leveraging the well-established PnP/RANSAC algorithm. Noteworthy, a small group of methods [85, 86, 87] resort to pose-sensitive latent embedding for subsequent pose estimation, demonstrating promising performance. As for RGB-D methods, similarly, some methods [88, 89, 90, 91] regress pose via end-to-end learning, others [92, 93] harness latent embedding, while [94] combines real-time optical flow with a low frame rate CNN. In spite of significant progress in recent years, instance-level methods suffer from inherent disadvantages, i.e. they can typically handle only a handful of objects and require explicit CAD models.

5.2.2 Category-level pose estimation

Category-level pose estimation deals with the task of estimating the full configuration (6D pose) of unseen objects from known categories, including of translation and rotation. The setting is more challenging due to the large intra-class shape and texture variations. Under the fully-supervised setting, Wang *et al.* [63] designed the *Normalized Object Coordinate Space* (NOCS) as a unified representation across different categories. Then Umeyama algorithm [95] was adopted to recover all parameters. Subsequent works [96, 64, 66, 97, 98] followed the NOCS representation and incorporated decoupled rotation representation [96, 64], explicit and implicit pose prediction [99, 64], geometric object properties [100, 64, 101], or shape prior [99, 66, 97, 102] to enhance the overall performance. 6-PACK [103] tracks the object's pose leveraging semantic keypoints, while CAPTRA [104] instead combines

coordinate prediction with direct regression. CASS [65] derives a learned canonical shape space to replace NOCS, and SGPA [97] uses transformer to extract global features.

Recently, self-supervised category-level pose estimation has received wide research interest. Li *et al.* [105] proposed to utilize SE(3) equivariant point cloud networks to predict canonical reconstruction and pose simultaneously. He *et al.* [106] leveraged differentiable renderer to self-supervise pose estimation via render-and-compare. UDA-COPE [107] exploits a teacher-student scheme to narrow domain gap between synthetic and real-world data. In addition, UDA-COPE looks for consistency in 3D space to improve the pseudo labels for enhancing the teacher's robustness. While most state-of-the-art methods for self-supervised category-level pose estimation solely enforce depth consistency in 3D space, the method presented in this dissertation leverages the 2D optical flow as an additional supervision signal in order to capture finer differences in object pose.

5.3 Method

This section introduces the method for tackling the challenging problem of estimating the 6D object pose (i.e. 3D rotation as well as 3D translation) plus scale, without requiring any 3D annotations. It leverages the optical flow in 2D, predicted by an off-the-shelf optical flow estimation model, as a mean for supervising the 6D pose. To this purpose, the differentiable rendering is harnessed to obtain the 2D optical flow from two individual pose estimates.

Since rendering requires having the correct pose together with the corresponding object mesh, SSC-6D [72] is adopted as backbone as it produces state-of-the-art results on the joint pose and shape estimation task. The proposed pipeline is shown in Figure 5.2. Given two masked RGB-D images from subsequent frames of a video as input, the 6D pose and geometry (in the form of an SDF volume) of the object is first estimated in each frame individually. Then, the optical flow between the two frames is computed by projecting the mesh vertices onto the image plane and measuring their movement in 2D image space. Both frames are also fed to an off-the-shelf optical

flow estimator [73] to obtain the corresponding weak ground truth flow. The optical flow is then leveraged to guide the optimization to improve the initial pose estimator. Since only when estimating the correct 6D poses in both frames one can obtain the right optical flow and since the directly estimated flow is very reliable in practice, the 2D proxy task can act as a very strong signal for optimizing the actual 3D objective.

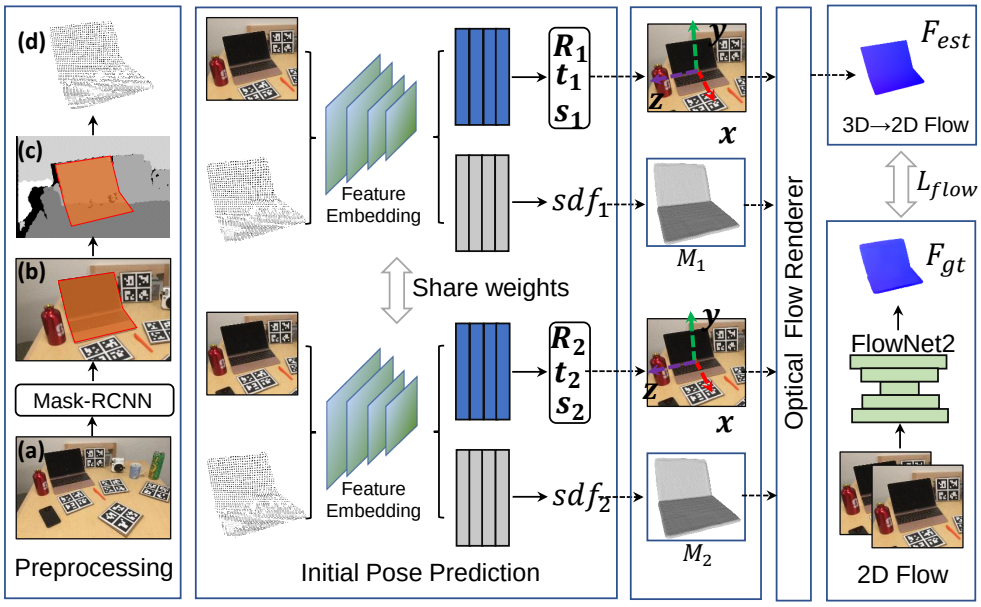

Figure 5.2: Schematic overview of the proposed framework. It first detects each object and computes their object masks with Mask R-CNN. Then, these masks are used to crop out the objects from the RGB image and the 3D point cloud. Given two subsequent frames as input, the model estimates the 6D poses and reconstructed shapes of the objects in each frame separately. Afterwards, the estimated objects are projected onto the 2D image space, which in turn allows to measure their 2D optical flow between the two images. This optical flow is then leveraged to establish a consistency loss with the ground truth optical flow, which is obtained with FlowNet2.

5.3.1 6D pose and shape estimation

The input consists of two separate RGB-D frames, showing the same object from different viewpoints. Then, the object is localized in each frame $i \in \{1,2\}$ using an off-the-shelf Mask R-CNN detector [108]. Using the estimated mask, it is possible to crop the object from the RGB and depth image, which is further backprojected to get the corresponding 3D object point cloud. Each resulting RGB image $I_i \in \mathbb{R}^{W \times H \times 3}$ and point cloud $\mathscr{P}_i \in \mathbb{R}^{Q \times 3}$ are fed together to the backbone feature extractor [72] E, before diverging into two individual branches for estimation of the object pose $(R_i, t_i, s_i) = P(E(I_i, \mathscr{P}_i))$ and a low-dimensional shape latent vector $\mathbf{e}_i = S(E(I_i, \mathscr{P}_i))$. Thereby, the shape is encoded using the learned shape embedding $\mathbf{e}_i \in \mathbb{R}^k$ of dimensionality $k = 16$ with a pretrained DeepSDF decoder D obtaining $sdf_i = D(\mathbf{e}_i, x)$ where $x \in \mathbb{R}^3$ is a 3D position, while the object pose is represented as rotation $R_i \in SO(3)$, translation $t_i \in \mathbb{R}^3$, and scaling factor $s_i \in \mathbb{R}$. Notice that the sdf_i value of a 3D point x describes the signed distance towards the object surface as encoded by $\mathbf{e}_i$. Thus, the underlying 3D geometry $\mathscr{M}_i$ can be easily extracted seeking the 0-level set $D(\mathbf{e}_i, x) = 0$ of the underlying function, which can be achieved using, for example, marching cubes.

5.3.2 Differentiable rendering for optical flow

As aforementioned, in this thesis it is proposed to improve the prediction of the 6D pose (R_i, t_i, s_i) of an object in an image I_i using the optical flow in 2D. Note that given two subsequent frames I_1 and I_2 the optical flow $F \in \mathbb{R}^{W \times H \times 2}$ describes the per-pixel displacement, which aligns the first frame with the second, i.e. at pixel (x,y) there is $I_1(x,y) = I_2(x + F(x,y)_x, y + F(x,y)_y)$.

In recent years estimating the optical flow from two images of the same scene has made tremendous improvements, thanks to deep learning and the availability of large datasets [74]. Consequently, for two rather similar views the optical flow can be extracted in a very robust and accurate way. In particular, the proposed framework leverages FlowNet2 [73] to obtain the pseudo ground truth flow F_{gt} from the two input frames I_1 and I_2. Examples of optical flow obtained with FlowNet2 are presented in

Figure 5.3: Example of subsequent frames I_1 (left column) and I_2 (center column) from NOCS REAL dataset. Optical flow by FlowNet2 (right column).

Figure 5.3. The color code used is in Figure 5.4. The color of each pixel represents the direction and the intensity is the magnitude.

In order to supervise the pose estimator using F_{gt}, it is needed to compute the corresponding flow from the predicted poses. Remember that the optical flow describes the per-pixel displacement of each pixel between the two frames. Hence, to accomplish this the predicted shape encoding $\mathbf{e}_i$ is considered and the underlying mesh $\mathscr{M}_i = (V_i, T_i)$ is extracted using a differentiable variant of marching cubes, where $V_i \in \mathbb{R}^{N\times 3}$ denote the 3D vertices of the mesh and $T_i \in \mathbb{N}^{M\times 3}$ denote its 3D triangles. Then the mesh is transformed into the 3D scene leveraging the predicted pose with:

$$\widehat{V}_i = s_i R_i V_i + t_i. \tag{5.1}$$

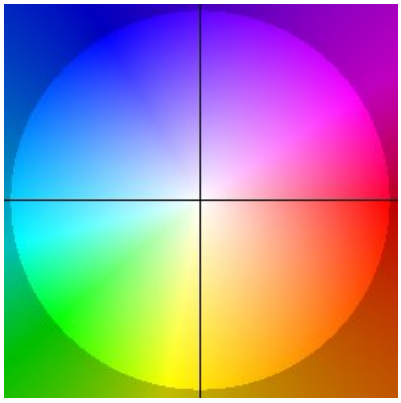

Figure 5.4: Optical flow visualization color code.

Finally, the optical flow can be easily obtained by projecting each vertex

$$\widehat{v}_i = K\widehat{V}_i, \tag{5.2}$$

onto the image plane, where K denotes the camera intrinsics, and subtracting the resulting pixel locations $\widehat{v}_1$ of the first view from the second view $\widehat{v}_2$. In other words, the optical flow F at the projected location $\widehat{v}_1$ amounts to

$$F'_{est}(\widehat{v}_1) = \widehat{v}_2 - \widehat{v}_1 \tag{5.3}$$

While this allows to obtain the flow with respect to each vertex of the mesh, it still suffers from the fact that multiple vertices can end up on the same pixel location, and that the output image would be only defined at a sparse set of points. To compensate for this issue, it is possible to use classical rendering, with

$$I = \mathscr{R}(R,t,s,V,T,\mathscr{C}), \tag{5.4}$$

where $\mathscr{C} \in \mathbb{R}^{N\times q}$ indicates the vertex colors and $I \in \mathbb{R}^{W\times H\times q}$ represents the rendered output image, to cull the hidden points and to produce a complete rendering of the flow. Unfortunately, rendering is known to be non-differentiable. Nonetheless, a handful of methods have recently been proposed to re-establish the gradient flow in the rendering process [109]. This is commonly achieved either by approximating the

gradient [110] or instead by approximating the rendering itself [111]. In particular, in the proposed framework Pytorch 3D differentiable render [112] is used. Finally, the vertex colors are set to the computed vertex displacements after projection and the flow is rendered with

$$F_{est} = \mathscr{R}(R_1, t_1, s_1, V_1, T_1, \widehat{v_2} - \widehat{v_1}). \tag{5.5}$$

Figure 5.5 shows examples of estimated optical flow F_{est} and masked optical flow F_{gt} used for self-supervision.

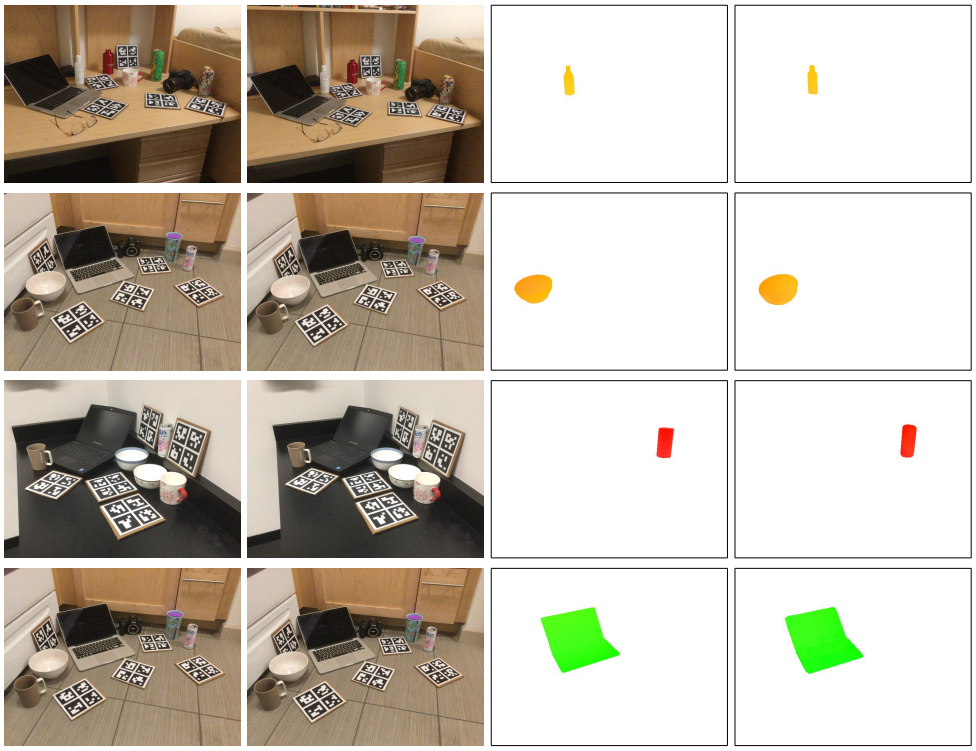

Figure 5.5: Examples of optical flows for some object instances. From left to right: first frame I_1, second frame I_2, optical flow estimated from the 6D poses and reconstructed shape (F_{est}), and the optical flow by FlowNet2 (F_{gt}).

5.3.3 Self-supervising the 6DoF pose

To obtain accurate object poses, it is not possible to directly train the model using a loss on the optical flow itself. Therefore, the appoarch presented in [69, 72] is followed and the model is trained in two steps. First, to get good initial estimates synthetic data are leveraged to train the model fully supervised in simulation. Subsequently, the model is finetuned on unlabelled real data in an effort to close the synthetic-to-real domain gap. The model is trained separately for each category. Also, an initial training of DeepSDF decoder on the synthetic CAD models is needed, one for each category.

The first fully-supervised training is done on synthetic data considering a single RGB-D frame as input. The training loss, as in [72], is L_{sup}, defined as $L_{sup} = L_{pose} + L_e$ where:

$$\mathscr{L}_{pose} = \frac{1}{N}\sum_{i=1}^{N} \|\widehat{V} - \widehat{V}_{gt}\|_2 \tag{5.6}$$

and

$$\mathscr{L}_e = \|\mathbf{e} - \mathbf{e}_{gt}\|_1 \tag{5.7}$$

$\widehat{V}_{gt}$ is computed as Equation 5.1, but using the ground truth pose, while $\mathbf{e}_{gt}$ is the ground truth shape latent vector obtained during the initial DeepSDF training step. For symmetric objects, the rotations are transformed into canonical rotations as proposed in [113].

To self-supervise the model on real data the loss functions from SSC-6D are adopted and then the loss for flow estimation is added in order to better guide the optimization. During the self-supervision, two subsequent frames are given as input for the model. The geometric alignment is guided by $\mathscr{L}_{chamfer}$, which computes the Chamfer loss between the point cloud observed $\mathscr{P}$ and the visible surface of the reconstructed model $\mathscr{P}_{rec}$. The visible surface of the reconstructed model in the object coordinates reference frame is obtained by applying the sphere tracing algorithm in [114] on the SDF representation of the object. Then, the estimated rotation, translation and scale are used to obtain $\mathscr{P}_{rec}$ in the camera coordinates reference frame.

The loss is defined as:

$$\begin{aligned}\mathscr{L}_{chamfer} = &\frac{1}{|\mathscr{P}_{rec}|}\sum_{p_i\in\mathscr{P}_{rec}}\min_{p_j\in\mathscr{P}}\|p_i-p_j\|_2+\\ &\frac{1}{|\mathscr{P}|}\sum_{p_j\in\mathscr{P}}\min_{p_i\in\mathscr{P}_{rec}}\|p_j-p_i\|_2\end{aligned} \tag{5.8}$$

The backpropagation is allowed only on the rotation, translation and scale. As a consequence (5.8) improves only the pose estimation. In order to improve shape reconstruction as well, an additional loss is used. By observing that the point cloud is located on the surface of the object, where the sdf value is zero, the additional loss is:

$$\mathscr{L}_{sdf} = \frac{1}{|\mathscr{P}^o|}\sum_{p_j\in\mathscr{P}^o}|\mathscr{D}(p_j,\mathbf{e})|\cdot\mathbb{1}(\|p_j\|_2<1) \tag{5.9}$$

where $\mathscr{P}^o$ is obtained by applying to the observed point cloud $\mathscr{P}$ the rotation, translation and scale. The resulting point cloud $\mathscr{P}^o$ is the observed point cloud in the coordinates reference frame of the object. The indicator function $\mathbb{1}(\cdot)$ avoids outliers, and it is equal to 1 only if p_j is inside the unit sphere. A regularization loss is also considered, defined as:

$$\mathscr{L}_{reg} = e^{\beta\|e\|_2}-1 \tag{5.10}$$

where β is a category-dependent hyper parameter which is set as in [72].

Hence, the self-supervision loss proposed in [72] is:

$$\mathscr{L}^i = \lambda_{chamfer}\mathscr{L}_{chamfer}+\lambda_{sdf}\mathscr{L}_{sdf}+\lambda_{reg}\mathscr{L}_{reg} \tag{5.11}$$

where $\lambda_{chamfer}=100$, $\lambda_{sdf}=1$, and $\lambda_{reg}=0.01$ and i is the frame number. This self supervision loss (5.11) is applied separately to each of the two input frames. Finally, the flow consistency loss exploits the weak ground truth flow and the flow obtained from the estimated poses, to supervise the training:

$$\mathscr{L}_{flow} = \lambda\, smoothL_1(F_{gt},F_{est}). \tag{5.12}$$

In the end, the proposed self-supervised loss is:

$$\mathscr{L}_{self} = \frac{1}{2}\mathscr{L}^1+\frac{1}{2}\mathscr{L}^2+\mathscr{L}_{flow} \tag{5.13}$$

It considers the self-supervision loss presented in [72] for each frame, and the geometric optical flow consistency. To avoid overfitting on the new dataset and to maintain the learning information gained on synthetic data, the training on the real data is based on a joint learning approach. Each batch combines J synthetic data and G pairs of real data. The final loss is:

$$\mathscr{L} = \frac{1}{J}\sum_{i=1}^{J}\mathscr{L}_{sup}^{i} + \frac{1}{G}\sum_{j=1}^{G}\mathscr{L}_{self}^{j} \tag{5.14}$$

5.4 Experiments

5.4.1 Experimental setup

The approach presented in the previous section is evaluated on the common NOCS benchmark [63], which consists of a synthetic dataset (NOCS CAMERA) and a real-world dataset (NOCS REAL) and it exhibits six different classes with various different object instances (bottle, bowl, camera, can, laptop, mug). The NOCS REAL dataset further contains 18 different scenes, of which 7 are for training, 6 for testing and 5 for validation. As in [72], the NOCS CAMERA dataset is used to train an initial model for 10 epochs in simulation. In addition, for the subsequent fine-tuning phase the training split from the NOCS REAL dataset is instead leveraged, however, all accompanying 6D pose labels are ignored. The model is trained for additional 11 epochs. Following common practice, a separate model is trained for each category. For all experiments, the weight λ for the flow loss was set to 0.1. For joint learning the batch size was set to 14, with $J = 12$ synthetic samples and $G = 2$ pairs of real samples.

Notice that each sequence from the NOCS REAL dataset is an actual video sequence, which allows to easily extract subsequent frames for flow estimation. In general this approach is beneficial as a video sequence of real data is fairly easy to obtain, and the only requirement is to have two frames capturing almost the same scene from different points of view. Note that to associate the same object in two different frames, the knowledge that there is always a single instance of a certain class in each sequence is simply exploited.

The metrics used for evaluation are, as in [63], the 3D Intersection Over Union (IoU) considering 0.25, 0.50 and 0.75 as thresholds. To further compare rotation and translation error, the 5°2*cm*, 5°5*cm*, 10°2*cm*, 10°5*cm* and 10°10*cm* metrics are considered. A pose is considered correct if the rotation and translation errors are respectively below the thresholds. For symmetric objects, the rotation around the vertical axis is not considered.

5.4.2 Comparison with the state-of-the-art

Results of the experiments in terms of mean Average Precision (mAP) are in Table 5.1. The results obtained with the proposed self-supervised framework are compared with some fully supervised methods (NOCS [63], CASS [65], DeformNet [66], SGPA [97], GPV-Pose [64]) on the NOCS REAL test set as reported in [72] and a state of the art self-supervised method [107]. Furthermore, the Table reports results with and without a final ICP refinement step as done in [72]. Some of the fully-supervised methods like SGPA and GPV-Pose achieve higher performance, however they require a labeled dataset. The proposed method without ICP refinement slightly improves the IoU with respect to SSC-6D, as IoU_{50} increases by 0.43, while it greatly improves metrics that take into account the rotation error. For example, the 5°2*cm* metric increases by 7.55. With ICP refinement, the IoU_{50} increases by 1.55 compared with SSC-6D, while the 5°2*cm* metric is 6.31 higher. The proposed method paired with ICP also exceeds UDA-COPE for 5°2*cm* and 5°5*cm*, which are two metrics that are more strict on error in rotation, while UDA-COPE achieves better results on the other metrics. Note that the 3D IoU is measured on the axis aligned bounding boxes, thus it gives more weight to translation error. Figure 5.6 reports the average precision for each object category. Table 5.2 reports the results on the test set of the NOCS CAMERA and NOCS REAL datasets. Each column contains from left to right: the model pretrained only on the NOCS CAMERA dataset (w/o SS), SSC-6D [72], and the proposed approach. Before self-supervision, when training is performed only on the NOCS CAMERA dataset in a supervised manner, the results are good on the NOCS CAMERA dataset and the model performs poorly on the unseen dataset NOCS REAL. With the SSC-6D self-supervision framework the results on the NOCS

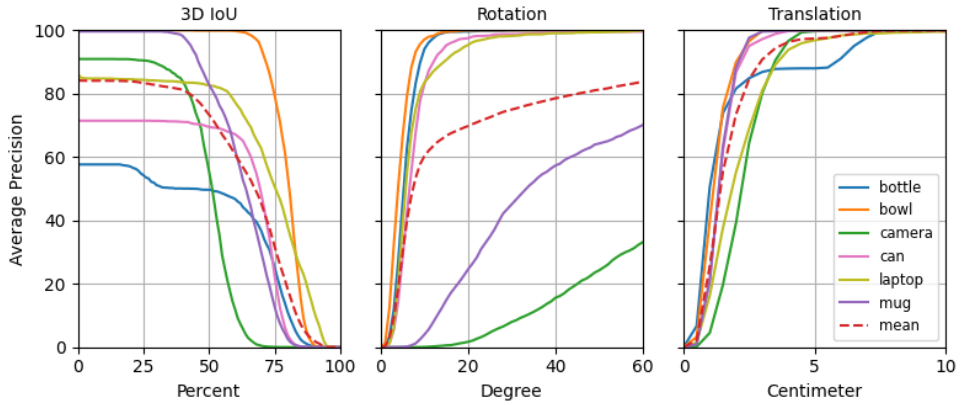

Figure 5.6: Average precision of IoU, rotation and translation error for each object category on the NOCS REAL dataset.

Table 5.1: Comparison of the proposed approach with other methods on the NOCS REAL dataset (in bold the maximum values for each group of columns)

	Fully supervised					Self-Supervised				
							SSC-6D		Proposed	
mAP(%)	NOCS	CASS	DeformNet	SGPA	GPV-Pose	UDA-COPE	w/o ICP	w/ ICP	w/o ICP	w/ ICP
IoU_{25}	78.72	68.09	80.22	-	**84.2**	-	83.16	83.11	**83.36**	83.12
IoU_{50}	47.23	25.95	68.73	80.1	**83.0**	**82.6**	72.95	72.67	73.38	74.22
5°2cm	7.26	19.66	19.11	**35.9**	32.0	30.4	16.76	28.6	24.31	**34.91**
5°5cm	9.86	23.60	21.15	39.6	**42.9**	34.8	19.62	33.39	27.09	**39.50**
10°2cm	14.04	51.61	43.39	**61.3**	-	**56.9**	44.12	51.82	49.37	53.13
10°5cm	24.25	59.02	54.08	70.7	**73.3**	**66.0**	54.53	62.93	59.25	63.79
10°10cm	24.51	59.25	56.00	-	**74.6**	-	56.24	65.07	59.98	**65.88**

REAL improve. However, with the proposed self-supervision pipeline, the results further improve by reaching an IoU_{50} of 73.38, and a 5°2cm metric of 24.31. Moreover, the Table also shows that the pose estimation on the NOCS CAMERA dataset benefits from the self-supervision, with a IoU_{50} that goes from 78.19 to 86.05 and 5°2cm that goes from 46.28 to 50.63.

Some qualitative results are shown in Figure 5.7. The highest pose estimation er-

Table 5.2: Comparison without self-supervision (w/o SS), SSC-6D, and the proposed approach

	NOCS CAMERA			NOCS REAL		
mAP(%)	**w/o SS**	**SSC-6D**	**Proposed**	**w/o SS**	**SSC-6D**	**Proposed**
IoU_{25}	88.54	92.29	**92.72**	66.83	83.16	**83.36**
IoU_{50}	78.19	85.03	**86.05**	28.77	72.95	**73.38**
5°2cm	46.28	50.40	**50.63**	1.84	16.76	**24.31**
5°5cm	49.81	54.16	**54.68**	2.40	19.62	**27.09**
10°2cm	62.22	65.57	**67.02**	4.16	44.12	**49.37**
10°5cm	68.26	71.73	**73.27**	8.66	54.53	**59.25**
10°10cm	70.06	73.28	**74.93**	10.84	56.24	**59.98**

ror occurs for the rotation estimation of objects in the *camera* category. As mentioned in [72], the lower performance on the *camera* class are probably due to the large difference in shape and size between models in the training and test sets. Results before and after self-supervision with optical flow are in Figure 5.8.

5.4.3 Ablation study

To evaluate the influence of the loss weight λ an ablation study was conducted. The parameter value was varied from 0 to 1. The results are in Table 5.3, and show that $\lambda = 0.1$ is the best value. Further experiments were conducted to highlight the influence of the data augmentation during the training. The data augmentation consisted of vertical image flipping. As shown in Table 5.4, data augmentation improves the Average Precision. The higher increment is related to the IoU_{75} metric, going from 28.47 to 31.93.

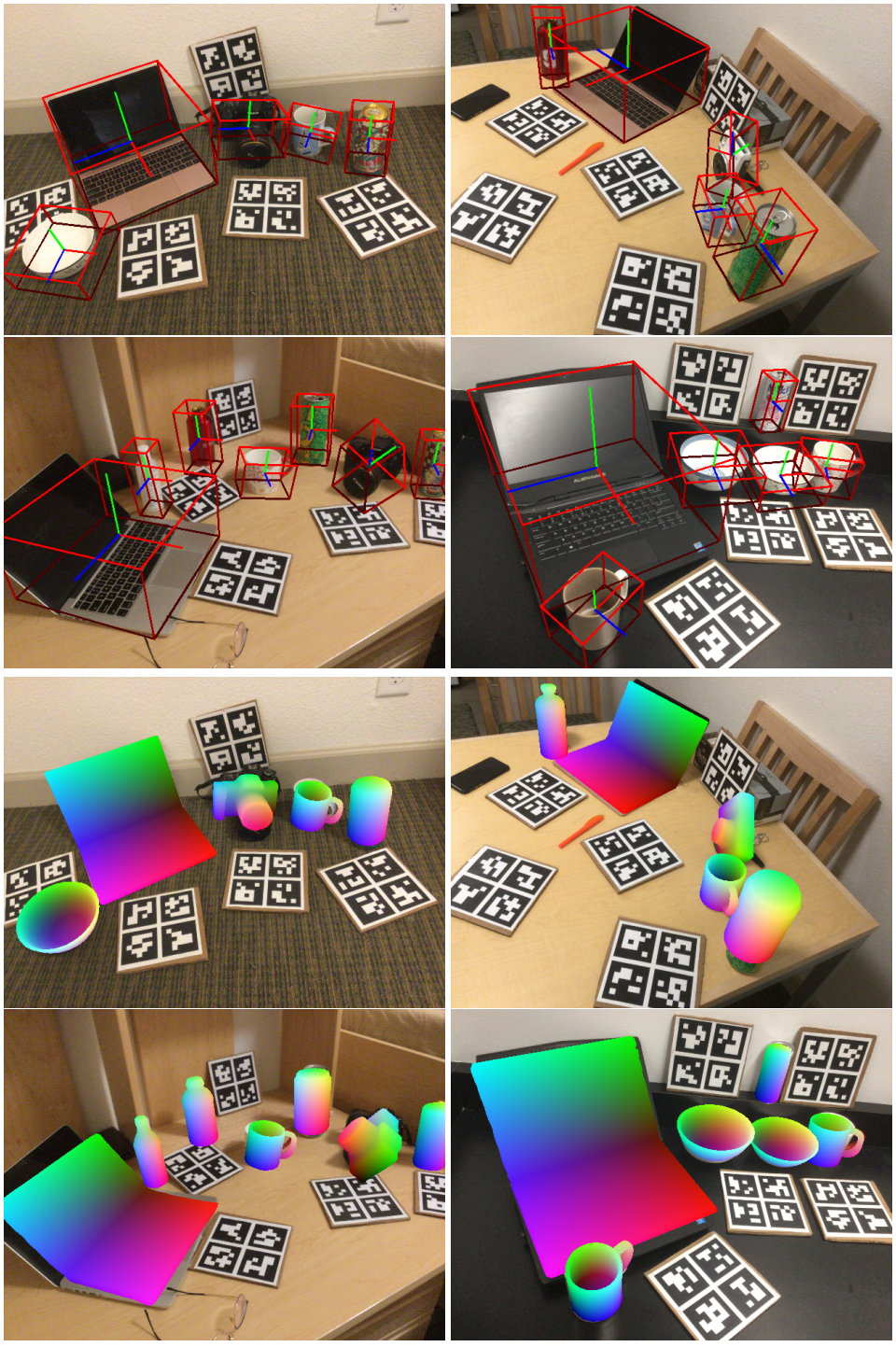

Figure 5.7: Example output images of the proposed method from NOCS REAL test set with bounding boxes superimposed to the detected objects (top). Color-coded images in a normalized object coordinate space (NOCS) representation [63] (bottom).

Table 5.3: Ablation study on λ parameter

	λ				
mAP(%)	0	0.1	0.3	0.5	1
IoU_{25}	83.21	83.36	**83.42**	83.30	83.30
IoU_{50}	72.92	73.38	73.01	**75.15**	73.74
IoU_{75}	28.68	**31.93**	29.11	28.58	27.46
5°2cm	18.51	**24.31**	23.67	19.87	18.67
5°5cm	21.32	**27.09**	27.01	22.62	21.63
10°2cm	43.26	**49.37**	47.77	45.46	48.10
10°5cm	51.98	**59.25**	56.74	53.28	57.71
10°10cm	53.35	**59.98**	57.98	54.99	58.93

Table 5.4: Ablation study on data augmentation

	Data augmentation	
mAP(%)	No	Yes
IoU_{25}	83.33	83.36
IoU_{50}	73.01	73.38
IoU_{75}	28.47	31.93
5°2cm	20.95	24.31
5°5cm	24.27	27.09
10°2cm	47.01	49.37
10°5cm	57.24	59.25
10°10cm	58.25	59.98

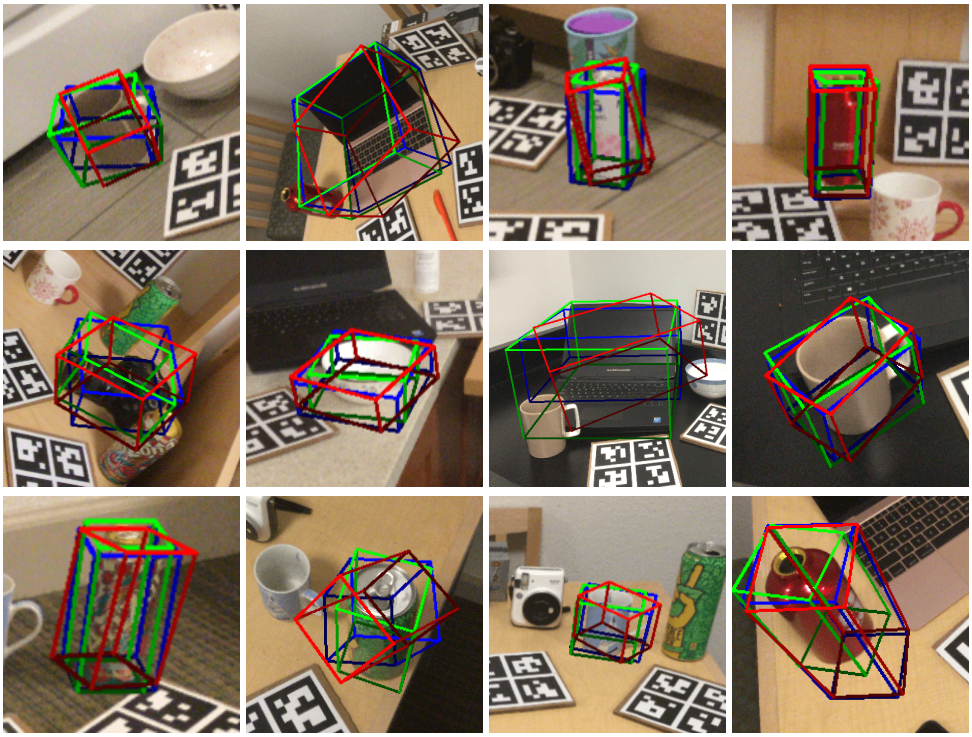

Figure 5.8: Comparison of pose estimation: ground truth in green, the proposed method in blue and SSC-6D in red.

5.5 Discussion

This Chapter introduced a self-supervision framework that leverages the 3D correspondences between two subsequent frames to improve the 6D pose estimation of an object. An optical flow estimation neural network is used to supervise the training. The method allows to perform training on a previously unseen dataset by only recording a video sequence in order to pair subsequent frames. Results of 6D pose estimation are comparable or better than other self-supervised methods on the REAL NOCS dataset. Also the results are comparable with some fully-supervised methods. Future works include investigating how to improve the robustness of the category-based approach in cases where there is a greater difference in objects shape and size

between the training set and the test set. Moreover, future works will investigate the application of the proposed method to solve the pallet detection and localization problems.

Chapter 6

Conclusions

This PhD dissertation presented a study and novel approaches to exploit traditional and innovative computer vision methods to improve Automated Guided Vehicles perception in industrial automated warehouses. The workers' safety problem was faced with the implementation of a people detection and tracking system. The system was extensively tested on real vehicles also in potentially dangerous conditions. Furthermore, the problem of pallet detection was addressed, without prior knowledge about the pallet shape and pose, by equipping the AGV with a vision system. The deployed system was able to detect the pallet in variable situations. Datasets from real scenarios were acquired, labelled and organized. The resulting AGV equipped with onboard cameras was able to perceive people in the surroundings, share people localization information with other AGVs, and automatically detect pallets without a priori knowledge about shape and location. These achievements may improve both safety and productivity. Also, a research on how to directly obtain the 6D pose estimation of an object with deep learning was conducted. As the main limitation of supervised approaches is the dataset labelling procedure, a self-supervised training procedure that requires only to take unlabeled video sequences of the real objects was exploited.

The proposed people detection and tracking system is strongly based on the assumption that the person's ankles are observable. In some cases, this assumption does not hold, for example, when the legs of a person are occluded. In future works,

it would be interesting to investigate more advanced approaches for people detection. Also, the deployed system will be installed and tested in complex scenarios in larger warehouses where several AGVs operate.

Moreover, even if the pallet detection system shows good detection results in different environments, it can be further improved by acquiring and labelling more datasets for the training, in order to improve generalization capabilities. In fact, if the warehouse presents peculiar types of goods, pallets, or lighting conditions this can have a strong impact on the detection results. Tests should also be performed on pallet forking tasks with the method introduced in Chapter 4, starting from different initial positions of the AGV with respect to the pallet to load.

Furthermore, future works will involve the application of the method developed in Chapter 5 to solve the pallet detection and localization problems.

In conclusion, it can be argued that since future AGVs will require more and more flexibility, accuracy and high reliability, automated vehicles will benefit from multiple sensors to maintain high performance. Therefore, the use of onboard computer vision systems will play an important role in the development of automated warehouses.

Bibliography

[1] Jieru Mei, Alex Zihao Zhu, Xinchen Yan, Hang Yan, Siyuan Qiao, Liang-Chieh Chen, and Henrik Kretzschmar. Waymo open dataset: Panoramic video panoptic segmentation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 53–72, Cham, 2022. Springer Nature Switzerland.

[2] Jun Cheng, Liyan Zhang, Qihong Chen, Xinrong Hu, and Jingcao Cai. A review of visual slam methods for autonomous driving vehicles. *Engineering Applications of Artificial Intelligence*, 114:104992, 2022.

[3] Éloi Zablocki, Hédi Ben-Younes, Patrick Pérez, and Matthieu Cord. Explainability of deep vision-based autonomous driving systems: Review and challenges. *International Journal of Computer Vision*, 130(10):2425–2452, Oct 2022.

[4] ILIAD project. `http://iliad-project.eu/`.

[5] A. Agarwal and S. Suryavanshi. Real-time* multiple object tracking (MOT) for autonomous navigation. Technical report, Stanford University, 2017.

[6] M. Andriluka, S. Roth, and B. Schiele. Monocular 3D pose estimation and tracking by detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 623–630, 2010.

[7] Adrian Cosma, Ion Emilian Radoi, and Valentin Radu. CamLoc: Pedestrian location estimation through body pose estimation on smart cameras. In *In-*

ternational Conference on Indoor Positioning and Indoor Navigation (IPIN), pages 1–8, 2019.

[8] M. Zaccaria, R. Monica, and J. Aleotti. A comparison of deep learning models for pallet detection in industrial warehouses. In *IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 417–422, 2020.

[9] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017.

[10] B Merven, F Nicolls, and G De Jager. Multi-camera person tracking using an extended Kalman filter. In *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*, volume 19, 2003.

[11] Charles Malleson, John Collomosse, and Adrian Hilton. Real-time multi-person motion capture from multi-view video and IMUs. *International Journal of Computer Vision*, 128(6):1594–1611, 2020.

[12] L. Chen, H. Ai, R. Chen, Z. Zhuang, and S. Liu. Cross-View Tracking for Multi-Human 3D Pose Estimation at Over 100 FPS. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3276–3285, 2020.

[13] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[14] N. A. Tsokas and K. J. Kyriakopoulos. Multi-robot multiple hypothesis tracking for pedestrian tracking with detection uncertainty. In *19th Mediterranean Conference on Control Automation (MED)*, pages 315–320, 2011.

[15] Nicolas A. Tsokas and Kostas J. Kyriakopoulos. Multi-robot multiple hypothesis tracking for pedestrian tracking. *Autonomous Robots*, 32(1):63–79, 2012.

[16] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004.

[17] Masataka Ozaki, Kei Kakimuma, Masafumi Hashimoto, and Kazuhiko Takahashi. Laser-based pedestrian tracking in outdoor environments by multiple mobile robots. *Sensors*, 12(11):14489–14507, 2012.

[18] K. Kakinuma, M. Hashimoto, and K. Takahashi. Outdoor pedestrian tracking by multiple mobile robots based on SLAM and GPS fusion. In *IEEE/SICE Intl Symposium on System Integration (SII)*, pages 422–427, 2012.

[19] Hengli Liu, Jun Luo, Peng Wu, Shaorong Xie, and Hengyu Li. People detection and tracking using RGB-D cameras for mobile robots. *International Journal of Advanced Robotic Systems*, 13(5), 2016.

[20] TRTPose. `https://github.com/NVIDIA-AI-IOT/trt_pose`.

[21] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, 2017.

[22] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 472–487, Cham, 2018. Springer International Publishing.

[23] M. Giorgini and J. Aleotti. Visualization of AGV in virtual reality and collision detection with large scale point clouds. In *IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 905–910, 2018.

[24] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, May 2008.

[25] T. Li, Q. Jin, B. Huang, C. Li, and M. Huang. Cargo pallets real-time 3D positioning method based on computer vision. *The Journal of Engineering*, 2019(23):8551–8555, 2019.

[26] T. Li, B. Huang, C. Li, and M. Huang. Application of convolution neural network object detection algorithm in logistics warehouse. *The Journal of Engineering*, 2019(23):9053–9058, 2019.

[27] L. Sabattini, M. Aikio, P. Beinschob, M. Boehning, E. Cardarelli, V. Digani, A. Krengel, M. Magnani, S. Mandici, F. Oleari, C. Reinke, D. Ronzoni, C. Stimming, R. Varga, A. Vatavu, S. Castells Lopez, C. Fantuzzi, A. Mäyrä, S. Nedevschi, C. Secchi, and K. Fuerstenberg. The PAN-Robots Project: Advanced Automated Guided Vehicle Systems for Industrial Logistics. *IEEE Robotics Automation Magazine*, 25(1):55–64, 2018.

[28] Robert Varga and Sergiu Nedevschi. Robust Pallet Detection for Automated Logistics Operations. In *11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPH)*, pages 470–477, 2016.

[29] R. Varga, A. Costea, and S. Nedevschi. Improved autonomous load handling with stereo cameras. In *International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 251–256, 2015.

[30] R. Varga and S. Nedevschi. Vision-based autonomous load handling for automated guided vehicles. In *10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 239–244, 2014.

[31] M. Seelinger and J.-D. Yoder. Automatic Pallet Engagment by a Vision Guided Forklift. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4068–4073, 2005.

[32] Jordi Pagès, Xavier Armangué, J. Salvi, Jordi Freixenet, and Jakub Marti. A Computer Vision System for Autonomous Forklift Vehicles in Industrial En-

vironments. In *9th Mediterranean Conference on Control and Automation (MEDS)*, 2001.

[33] G. Garibotto, S. Masciangelo, M. Ilic, and P. Bassino. ROBOLIFT: a vision guided autonomous fork-lift for pallet handling. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, 1996.

[34] Young Hun Song, Jee Hun Park, Suk Lee, and Kyung Chang Lee. Implementation of distributed architecture based on CAN networks for unmanned forklift. In *37th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pages 2595–2599, 2011.

[35] B. Molter and J. Fottner. Real-time Pallet Localization with 3D Camera Technology for Forklifts in Logistic Environments. In *IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 297–302, 2018.

[36] G. Chen, R. Peng, Z. Wang, and W. Zhao. Pallet Recognition and Localization Method for Vision Guided Forklift. In *8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCom)*, pages 1–4, 2012.

[37] Guang-zhao Cui, Lin-sha Lu, Zhen-dong He, Li-na Yao, Cun-xiang Yang, Buyi Huang, and Zhi-hong Hu. A robust autonomous mobile forklift pallet recognition. In *2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR)*, volume 3, pages 286–290, 2010.

[38] Jia-Liang Syu, Hsin-Ting Li, Jen-Shiun Chiang, Chih-Hsien Hsia, Po-Han Wu, Chi-Fang Hsieh, and Shih-An Li. A computer vision assisted system for autonomous forklift vehicles in real factory environment. *Multimedia Tools and Applications*, 76, 11 2016.

[39] Z. He, X. Wang, J. Liu, J. Sun, and G. Cui. Feature-to-Feature Based Laser Scan Matching for Pallet Recognition. In *International Conference on Mea-*

suring Technology and Mechatronics Automation, volume 2, pages 260–263, 2010.

[40] M. R. Walter, S. Karaman, E. Frazzoli, and S. Teller. Closed-loop pallet manipulation in unstructured environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5119–5126, 2010.

[41] Y. Kita, R. Takase, T. Komuro, N. Kato, and N. Kita. Detection and localization of pallets on shelves using a wide-angle camera. In *19th International Conference on Advanced Robotics (ICAR)*, pages 785–792, 2019.

[42] L Baglivo, N Biasi, F Biral, N Bellomo, E Bertolazzi, M Da Lio, and M De Cecco. Autonomous pallet localization and picking for industrial forklifts: a robust range and look method. *Measurement Science and Technology*, 22(8), 2011.

[43] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[44] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision (ECCV)*, 2016.

[45] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *ArXiv*, abs/2004.10934, 2020.

[46] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation (CVPR). In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[47] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[48] P. Chao, C. Kao, Y. Ruan, C. Huang, and Y. Lin. Hardnet: A low memory traffic network. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3551–3560, 2019.

[49] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[50] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[51] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[52] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[53] Ihab S. Mohamed, Alessio Capitanelli, Fulvio Mastrogiovanni, Stefano Rovetta, and Renato Zaccaria. Detection, localisation and tracking of pallets using machine learning techniques and 2d range data. *Neural Computing and Applications*, 32(13):8811–8828, Jul 2020.

[54] Efthimios Tsiogas, Ioannis Kleitsiotis, Ioannis Kostavelis, Andreas Kargakos, Dimitris Giakoumis, Marc Bosch-Jorge, Raquel Julia Ros, Rafa López Tarazón, Spyridon Likothanassis, and Dimitrios Tzovaras. Pallet detection and docking strategy for autonomous pallet truck agv operation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3444–3451, 2021.

[55] Robert Varga. and Sergiu Nedevschi. Robust pallet detection for automated logistics operations. In *Proceedings of the 11th Joint Conference on Computer*

Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, (VISIGRAPP 2016), pages 470–477. INSTICC, SciTePress, 2016.

[56] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the InternationalConference on Computer Vision (ICCV)*, 2017.

[57] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. ROI-10D: Monocular Lifting of 2D Detection to 6D Pose and Metric Shape. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2064–2073, 2019.

[58] Pengyuan Wang, Fabian Manhardt, Luca Minciullo, Lorenzo Garattoni, Sven Meier, Nassir Navab, and Benjamin Busam. Demograsp: Few-shot learning for robotic grasping with human demonstration. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5733–5740. IEEE, 2021.

[59] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent-Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. BOP: Benchmark for 6D object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[60] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2017.

[61] Mahdi Rad and Vincent Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *ICCV*, pages 3828–3836, 2017.

[62] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In *2019 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 4556–4565, 2019.

[63] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[64] Yan Di, Ruida Zhang, Zhiqiang Lou, Fabian Manhardt, Xiangyang Ji, Nassir Navab, and Federico Tombari. GPV-Pose: Category-level Object Pose Estimation via Geometry-guided Point-wise Voting. *arXiv preprint*, 2022.

[65] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning Canonical Shape Space for Category-Level 6D Object Pose and Size Estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11970–11979, 2020.

[66] Meng Tian, Marcelo H Ang, and Gim Hee Lee. Shape prior deformation for categorical 6D object pose and size estimation. In *European Conference on Computer Vision*, pages 530–546. Springer, 2020.

[67] Tomáš Hodaň, Dániel Baráth, and Jiří Matas. EPOS: Estimating 6D Pose of Objects With Symmetries. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11700–11709, 2020.

[68] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects. In *ICCVW*, pages 2767–2776, 2019.

[69] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6D: Self-supervised Monocular 6D Object Pose Estimation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 108–125, Cham, 2020. Springer International Publishing.

[70] Juil Sock, Guillermo Garcia-Hernando, Anil Armagan, and Tae-Kyun Kim. Introducing pose consistency and warp-alignment for self-supervised 6D object pose estimation in color images. In *2020 International Conference on 3D Vision (3DV)*, pages 291–300. IEEE, 2020.

[71] Gu Wang, Fabian Manhardt, Xingyu Liu, Xiangyang Ji, and Federico Tombari. Occlusion-Aware Self-Supervised Monocular 6D Object Pose Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[72] Wanli Peng, Jianhang Yan, Hongtao Wen, and Yi Sun. Self-Supervised Category-Level 6D Object Pose Estimation with Deep Implicit Shape Representation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(2):2082–2090, Jun. 2022.

[73] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017.

[74] Mingliang Zhai, Xuezhi Xiang, Ning Lv, and Xiangdong Kong. Optical flow and scene flow estimation: A survey. *Pattern Recognition*, 114:107861, 2021.

[75] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *RSS*, 2018.

[76] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6D pose refinement in RGB. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 833–849, 2018.

[77] Fabian Manhardt, Diego Martin Arroyo, Christian Rupprecht, Benjamin Busam, Tolga Birdal, Nassir Navab, and Federico Tombari. Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data. In *Proceedings*

of the IEEE International Conference on Computer Vision, pages 6841–6850, 2019.

[78] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6D pose estimation. *IJCV*, pages 1–22, 2019.

[79] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6D pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020.

[80] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16606–16616, 2021.

[81] Zhigang Li, Gu Wang, and Xiangyang Ji. CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6-DoF Object Pose Estimation. In *ICCV*, pages 7678–7687, 2019.

[82] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6D object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 431–440, 2020.

[83] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. DPOD: 6D Pose Object Detector and Refiner. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1941–1950, 2019.

[84] Kiru Park, Timothy Patten, and Markus Vincze. Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7667–7676, 2019.

[85] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3D Orientation Learning for 6D Object Detection from RGB Images. In *European Conference on Computer Vision (ECCV)*, 2018.

[86] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, Narunas Vaskevicius, Kai O Arras, and Rudolph Triebel. Multi-path learning for object pose estimation across domains. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13913–13922, 2020.

[87] Rasmus Laurvig Haugaard and Anders Glent Buch. SurfEmb: Dense and Continuous Correspondence Distributions for Object Pose Estimation with Learnt Surface Embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6749–6758, 2022.

[88] Chi Li, Jin Bai, and Gregory D Hager. A unified framework for multi-view multi-class object pose estimation. In *Proceedings of the european conference on computer vision (eccv)*, pages 254–269, 2018.

[89] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. In *CVPR*, pages 3343–3352, 2019.

[90] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. PVN3D: A deep point-wise 3D keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020.

[91] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. FFB6D: A Full Flow Bidirectional Fusion Network for 6D Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3003–3013, June 2021.

[92] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3D pose estimation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3109–3118, 2015.

[93] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep Learning of Local RGB-D Patches for 3D Object Detection and

6D Pose Estimation. In *European Conference on Computer Vision (ECCV)*, 2016.

[94] Nicola A. Piga, Yuriy Onyshchuk, Giulia Pasquale, Ugo Pattacini, and Lorenzo Natale. ROFT: Real-Time Optical Flow-Aided 6D Object Pose and Velocity Tracking. *IEEE Robotics and Automation Letters*, 7(1):159–166, 2022.

[95] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991.

[96] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Shen Linlin, and Ales Leonardis. FS-Net: Fast Shape-based Network for Category-Level 6D Object Pose Estimation with Decoupled Rotation Mechanism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1581–1590, June 2021.

[97] Kai Chen and Qi Dou. SGPA: Structure-Guided Prior Adaptation for Category-Level 6D Object Pose Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2773–2782, 2021.

[98] Jiaze Wang, Kai Chen, and Qi Dou. Category-Level 6D Object Pose Estimation via Cascaded Relation and Recurrent Reconstruction Networks. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[99] Jiehong Lin, Zewei Wei, Zhihao Li, Songcen Xu, Kui Jia, and Yuanqing Li. DualPoseNet: Category-level 6D Object Pose and Size Estimation Using Dual Pose Network with Refined Learning of Pose Consistency. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3540–3549, 2021.

[100] Haitao Lin, Zichang Liu, Chilam Cheang, Lingwei Zhang, Yanwei Fu, and Xiangyang Xue. DONet: Learning Category-Level 6D Object Pose and Size Estimation from Depth Observation. *arXiv preprint arXiv:2106.14193*, 2021.

[101] Ruida Zhang, Yan Di, Zhiqiang Lou, Fabian Manhardt, Nassir Navab, Federico Tombari, and Xiangyang Ji. RBP-Pose: Residual Bounding Box Projection for Category-Level Pose Estimation. *arXiv preprint arXiv:2208.00237*, 2022.

[102] Zhaoxin Fan, Zhengbo Song, Jian Xu, Zhicheng Wang, Kejian Wu, Hongyan Liu, and Jun He. ACR-Pose: Adversarial Canonical Representation Reconstruction Network for Category Level 6D Object Pose Estimation. *arXiv preprint arXiv:2111.10524*, 2021.

[103] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-PACK: Category-level 6D pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020.

[104] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J. Guibas. CAPTRA: CAtegory-level Pose Tracking for Rigid and Articulated Objects from Point Clouds. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13189–13198, 2021.

[105] Xiaolong Li, Yijia Weng, Li Yi, Leonidas J Guibas, A Abbott, Shuran Song, and He Wang. Leveraging SE (3) Equivariance for Self-supervised Category-Level Object Pose Estimation from Point Clouds. *Advances in Neural Information Processing Systems*, 34:15370–15381, 2021.

[106] Yisheng He, Haoqiang Fan, Haibin Huang, Qifeng Chen, and Jian Sun. Towards self-supervised category-level object pose and size estimation. *arXiv preprint arXiv:2203.02884*, 2022.

[107] Taeyeop Lee, Byeong-Uk Lee, Inkyu Shin, Jaesung Choe, Ukcheol Shin, In So Kweon, and Kuk-Jin Yoon. UDA-COPE: Unsupervised Domain Adaptation for Category-level Object Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14891–14900, 2022.

[108] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[109] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020.

[110] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[111] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *ICCV*, pages 7708–7717, 2019.

[112] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D Deep Learning with PyTorch3D, 2020.

[113] Giorgia Pitteri, Michaël Ramamonjisoa, Slobodan Ilic, and Vincent Lepetit. On object symmetries and 6d pose estimation from images. In *2019 International Conference on 3D Vision (3DV)*, pages 614–622, 2019.

[114] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering Deep Implicit Signed Distance Function With Differentiable Sphere Tracing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2016–2025, 2020.