Augmented Reality For Indoor Navigation In A Warehouse Setting

Examining Supervised Learning-Based Pretrained CNNs for Feature Extraction in Visual Odometry

MSc Thesis: Vision, Graphics, and Interactive Systems

1043
Aalborg University
Electronics and IT

AALBORG UNIVERSITY
STUDENT REPORT

Electronics and IT
Aalborg University
http://www.aau.dk

Title:
Augmented Reality For Indoor Navigation In A Warehouse Setting

Theme:
Examining Supervised Learning-Based Pretrained CNNs for Feature Extraction in Visual Odometry

Project Period:
Spring Semester 2023

Project Group:
1043

Participant(s):
Bastian Starup Petersson

Supervisor(s):
Kamal Nasrollahi

Copies: 1

Page Numbers: 53

Date of Completion:
June 1, 2023

Abstract:

This report examines the potential use of Augmented Reality (AR) technology in improving warehouse management processes, specifically in tasks such as picking, receiving, and relocating items. The report explores the technical aspects of AR and the challenges involved in developing an AR solution for warehouse navigation, including the use of Simultaneous Localization and Mapping (SLAM) algorithms for accurate positional tracking. The report proposes a deep learning-based feature extractor for visual odometry in large industrial environments, which combines the FAST key point extractor with pre-trained CNNs to generate feature descriptors and evaluates its performance against traditional feature extraction techniques such as ORB. The test results show that the ORB algorithm demonstrated superior efficiency and accuracy compared to the deep learning-based methods. Future work includes investigating the impact of extracting features from different layers within pre-trained networks and exploring the benefits of leveraging GPUs for computation.

Contents

Preface

Aalborg University, June 1, 2023

[illegible]

Bastian Starup Petersson
bpeter18@student.aau.dk

Chapter 1

Introduction

1.1 Finding Bins In A Warehouse

In recent years, companies have increasingly adopted new technologies to transform their operations and enhance productivity. Technology integration has become a critical focus for many organizations seeking a competitive advantage, from robotics and machine learning for automation to implementing the Internet of Things (IoT). One emerging technology that is gaining traction in industrial settings is Augmented Reality (AR). AR technology seamlessly combines the physical and digital worlds by overlaying digital components onto a real-life environment.

This report will explore the possibility of using AR in a warehouse setting, where it might offer significant benefits for warehouse employees. Specifically, the use of AR to help warehouse employees find a location in a warehouse will be examined. Currently, many warehouses use a numbering system to organize their physical locations, referred to as "bins", with each bin having a unique "bin code". Typically, the bin codes are designed to reflect physical locations in the warehouse, with specific digits representing the aisle and shelf of the bin. This allows warehouse employees to more easily find the location of a bin based on its unique bin code. While this system aims to simplify location interpretation for warehouse employees, it can still pose challenges, especially for inexperienced employees. In a warehouse with many locations, the system can become confusing and time-consuming to navigate. [1]

As argued by (Schwerdtfeger et al., 2008), finding a bin in a warehouse can roughly be divided into two main stages. Firstly the warehouse employee must find the bin's general area, typically by finding the correct aisle. Once the correct aisle is found, they have to locate the exact bin within that aisle and the correct item on that bin. Performing these different levels of long and short-distance navigation takes time and can be cognitively straining if done many times. This is the core motivation for this report since AR might be used to reduce this problem. [20]

Given the potential challenges in bin finding, understanding the typical tasks carried out in a warehouse becomes crucial. The following section will delve into the main activities of warehouse employees and explore how AR might offer assistance.

1.2 Main Warehouse Tasks

To develop an effective AR navigation solution for a warehouse, it is essential to understand the primary tasks carried out by warehouse employees. In a typical warehouse, the three main tasks employees spend the most time on are picking, receiving, and relocating items. In all of these tasks, the warehouse employee needs to find specific bins in the warehouse. Consequently, AR navigation could be useful for accomplishing all these tasks more efficiently.

Picking is the process of finding and collecting items in the warehouse according to a specific sales order. To perform this task, a warehouse employee will typically have a list of all the items to be picked, containing the item number, quantity, and bin code to pick from. Additional information might also be present, such as a description or an image of the items. AR can potentially help with this task by displaying the aforementioned information, along with guiding the employee to specific bins. If implemented on a pair of AR Glasses, this could be particularly useful, as the warehouse employee would free up a hand that would otherwise be used to hold a phone or hand-held scanner.

Receiving items are, to a large extent, the opposite of picking. In this task, the employee is presented with a list of items, quantities, and bin codes where they should be placed. Similar to picking, this information could be displayed through AR glasses, enabling the employee to use both hands while performing the task.

Relocating items involves moving items from one bin position to another. AR might also be helpful in this task, guiding the user from the original bin to the new bin location, thus streamlining the process.

In conclusion, each of these key warehouse tasks could greatly benefit from the integration of AR technology. However, for AR to truly revolutionize these tasks, the implementation of precise and reliable AR navigation is essential. The following section will discuss the technical aspects of AR navigation and its challenges.

1.3 AR Navigation

To effectively utilize AR for guiding a user to specific bins in a warehouse, the AR-enabled device needs to continuously update and track its physical position, as well as the positions of the bins. This involves advanced sensors and algorithms to achieve real-time tracking. Once the positions of the device and the bins are known, an arrow can be displayed to guide the user to the correct bin.

Feature extraction is a crucial aspect of AR navigation, as it helps the device recognize and understand its surroundings. The extracted features are then used by algorithms, such as SLAM (Simultaneous Localization and Mapping), to maintain a consistent mapping of the environment while also tracking the device's position.

As explained by (Feigl et al., 2020), there are several prebuilt AR software frameworks available that allow developers to build advanced AR applications without focusing much on the underlying algorithms. Three of the largest AR frameworks are ARKit for iPhone and iPad, ARCore for Android devices, and Microsoft HoloLens. While it is not publicly known which specific algorithms these frameworks use for positional tracking, they likely employ some variation of a SLAM algorithm using Visual-inertial odometry. However, as found by (Feigl et al., 2020), these frameworks have limitations in large industrial environments, particularly due to an accumulation of errors over time. This issue is exacerbated in environments that change over time, such as those with objects being moved or varying lighting conditions. [7]

This report examines the underlying principles behind state-of-the-art SLAM algorithms, with a specific focus on a feature-based SLAM algorithm called ORB-SLAM and its feature extractor, ORB (Oriented FAST and Rotated BRIEF). Good features are essential for SLAM algorithms to work effectively, so the report also investigates other state-of-the-art deep learning-based feature extractors. It is found that deep learning-based feature extractors can be challenging to train due to the difficulty in creating a labeled dataset of features, as the lack of semantic information makes it challenging and unreliable for humans to create ground truth data.

The report proposes to use features learned by a CNN pre-trained on a classification task to create feature descriptors. This approach is implemented into a proposed algorithm that draws inspiration from ORB features by using FAST (Features from Accelerated Segment Test) to find key points and CNNs to create feature descriptors. Finally, this proposed feature extractor is tested using a VO (Visual Odometry) algorithm, with its performance compared to the performance of ORB features.

Chapter 2

Problem Analysis

2.1 Warehouse Management Systems

For any company with a warehouse, the warehouse employees must be able to perform their job efficiently and without making many errors. Some of the main tasks in a warehouse are picking, restocking, and moving items around. A record must be kept to track the warehouse stock to carry out these tasks efficiently. In the past, a record of items was kept using a clipboard. However, with time, different technologies have become important tools for warehouse work. Today, most warehouses store a digital record of all items, their bin locations, and quantities on each bin. The warehouse employees will typically carry a hand-held computer with a barcode or RFID scanner to identify the different items and bins in the warehouse. When performing the different tasks in the warehouse, the employees can see where the items are located, among other useful information. This information can be pulled directly from a company's warehouse management system (WMS), ensuring that the information is always up to date. [5]

As new technologies emerge, they might present an interesting use case within the context of warehouse management. One such technology is AR, which might be used for indoor navigation in a warehouse. Examining some of the tools and methods used in warehouses today might be advantageous to enable AR navigation in a warehouse. It is in the interest of any company with a warehouse that their warehouse operations run smoothly and without mistakes being made. To do so, the warehouse must be well organized, and an accurate and up-to-date record must be kept, containing information about the items in storage, how many there are, and on what bins they are located. Before computers became widely available, such records were kept on paper. However, today most companies have adopted the use of computer databases to store these records. Typically, such a database will be part of a WMS, a software tool for organizing the warehouse. WMSs are often integrated into the company's ERP (Enterprise Resource Planning) system to easily share information across different departments and ensure accurate and efficient record keeping. One of the most popular ERP systems is Microsoft Dynamics 365 Business Central (BC). BC incorporates a WMS enabling a corporation to manage its warehouse.

2.2 Warehouse Hardware

In the realm of warehouse management, hardware plays a critical role in complementing the capabilities of WMS and ERP systems, particularly in terms of real-time tracking and inventory management. Hand-held devices are widely used by warehouse employees, providing them with a portable interface to access the WMS database from anywhere in the warehouse. Having an up-to-date record of the warehouse inventory is crucial since it allows warehouse employees to easily find items in the warehouse when picking a sales order or performing other warehouse activities. Accessing information about the warehouse directly from the WMS using a stationary computer or laptop is possible. However, a warehouse employee will typically obtain the necessary information using a hand-held device connected to the internet. This device provides the employee with an up-to-date record of the state of the warehouse and enables the employee to access the WMS database from anywhere in the warehouse. Typically these devices can be used both to pick sales orders and put purchase orders in stock. A variety of hand-held devices exist for this purpose. One of the well-established companies that make such devices is called Zebra. [25]

Zebra is a technology company that provides enterprises with solutions to manage their logistics operations. The company provides various products and services, including barcode scanners and mobile computers. Warehouse employees often use Zebras products to perform common warehouse tasks. One of their newer and more popular products is the Zebra TC21/TC26 Touch Computer. The device is lightweight and has a 5-inch high-definition touchscreen that is easy to view both indoors and outdoors. The device runs on Android and is built to withstand harsh environments. It supports both wifi and a cellular 4G internet connection, enabling it to connect to a company's WMS. The device has a 13-megapixel rear camera, a removable battery, and the option for a hands-free wearable solution with an optional wrist-mount accessory. [25]

The research presented in this report has partly been conducted in collaboration with a company called Selek. Thus it might be helpful to examine the tools and methods used by Selek. Selek is a B2B company located in Aalborg, Denmark, specializing in distributing accessories for the IT industry. Selek has a warehouse where they store products before shipping them to customers. The ERP system used at selek is BC, which is also used as their WMS. To perform warehouse operations at Selek, the employees use iPhones as hand-held devices. Similar to the Zebra devices, the iPhones provide both wifi and a cellular internet connection and the ability to scan barcodes using the inbuilt camera. Furthermore, Selek is still using pen and paper sometimes when picking orders.

As technology advances, warehouse management hardware has become more sophisticated, with various companies like Selek incorporating modern technologies into their operations. However, these tools' usefulness largely depends on the software that supports them.

2.3 WMS software

To fully leverage the potential of these innovative hardware solutions in warehouse management, robust and reliable software solutions are required. These software applications serve as the backbone of operations, translating the data collected by hardware devices into actionable insights.

If a company uses Zebra scanners, iPhones, or other devices for warehouse management, they need some software for them to function. Depending on the hardware used, a variety of software solutions are available. One solution comes from a company called Tasklet. They provide a software solution for Android-based devices, such as the Zebra devices, which can be used to perform common warehouse operations. [23]

However, at Selek, they use a custom-made app that runs on iPhones. This app is used to pick, restock and move items in the warehouse. It works with BC, which is used at selek, providing the warehouse employees with information about new sales, purchase orders, and other useful information.

Nevertheless, whether these software applications are functioning on a Zebra scanner, an iPhone, or any other handheld device, their efficiency is reliant on the clarity of the displayed information. This brings us to an innovative way to visualize data—Augmented Reality (AR)—that could potentially enhance the warehouse management process by providing an immersive experience to users.

2.4 Literature review on AR in a warehouse setting

2.4.1 Hand-held Devices vs. AR Glasses

AR is emerging as a promising solution to overcome the limitations of traditional display methods in warehouse management. Its ability to overlay digital information onto the real world in real-time can create more intuitive and engaging experiences, enhancing the efficiency and accuracy of warehouse operations. As a rapidly developing technology, AR has been the subject of various studies assessing its applications and potential benefits in a warehouse setting.

Given that hand-held devices are already being used widely in many warehouses today, it might make sense to use these devices to display AR content to the user. The AR content shown to the employees could be an arrow pointing the employee to a specific bin location in the warehouse or a square around the items they have to pick. Although this could be achieved using a hand-held device, it might be even more useful if implemented on a pair of AR glasses, as done by (Stoltz et al., 2017), who experimented with using Google Glass as their AR platform [22]. This way, the employees could free up a hand that would otherwise be used to hold the device. Furthermore, relevant information could be displayed to the employee constantly, making them more efficient.

2.4.2 AR Benefits and Challenges

A few studies have been conducted exploring the use of AR in a warehouse setting. One such study was made by (Stoltz et al., 2017), who showed that AR has the potential to increase efficiency and reduce errors in the warehouse. The paper highlights the importance of warehouse efficiency for the overall performance of supply chains and the potential benefits of using AR to improve warehouse operations. The study investigates the opportunities and barriers to implementing AR in warehouses through practitioner interviews and an experiment using Google Glass. The study's results indicate that although the technology is not yet mature, its potential benefits make it promising for the near future. The paper identifies several advantages of using AR in warehouse operations, including improved accuracy in task execution, increased efficiency, better visualization, and enhanced safety. AR can display relevant information about incoming goods, storage locations, and picking routes, reducing the likelihood of human error. It can provide real-time information about the status of an operator's task, reducing downtime and improving productivity. AR can also display images and other details of items to be picked, making it easier for operators to identify and locate them. However, the paper also highlights several limitations and challenges associated with using AR in warehouse operations. For example, AR devices may only be designed for short periods of continuous use and can cause comfort problems. The battery life is also not designed to last a full working day. The total cost of ownership for AR solutions is still quite high, and alternative IT solutions for warehouse management can be significantly cheaper. Some users may not be willing to wear AR devices with cameras and microphones due to privacy concerns. Overall, the paper suggests that AR technology is promising to improve warehouse operations, but careful consideration of its limitations and challenges is necessary to ensure successful implementation. [22]

2.4.3 AR Headsets Enhance Order Picking

Another paper investigating the use of AR in a warehouse setting examines how order picking can be improved by using an AR headset. This study was conducted using an experimental warehouse setup, and the AR headset used tracking markers to track the headset's position accurately. Two experiments were conducted in the paper, showing that using AR for order picking can improve efficiency and error rates. Furthermore, it was concluded that the AR headset used in this study did not cause higher user strain than traditional picking methods, such as pen and paper. Overall, the authors suggest that AR can significantly improve logistics and warehousing operations, but more research and development are needed to realize this potential fully. [19]

2.4.4 AR Visualization for Item Picking

Another study has been conducted using the same experimental setup. This study examines the effectiveness of three different visualization approaches (Arrow, Frame, and Tunnel) for picking items from a shelf. An experiment was conducted involving 34 subjects who were asked to perform a picking task using the three different visualization approaches. The study indicated that different visualization techniques were useful depending on the user's situation. If the item to be picked is in the user's field of view, it would be suitable to display a frame or tunnel visualization. However, an arrow was helpful for situations where the desired item was not in the field of view. The process of finding a specific item can be divided into two steps. Firstly, the employee has to find the correct aisle and shelf. Afterward, they have to locate the exact item to pick. These two types of navigation require different visualization techniques. In the former, the user needs to know the general direction and distance to the item. In the latter, more precision is required to pinpoint the item to pick. [20]

2.4.5 AR Drift Error in Large Warehouses

Some warehouses are very large, presenting a significant challenge for the use of AR. Rather using a hand-held device such as a phone, or a pair of AR glasses, the device needs to know its position at all times if it is to guide the user to a specific bin. This becomes increasingly challenging for large environments since errors accumulate over large distances. AR devices typically rely on a class of algorithms called SLAM to constantly track their position. In addition to the challenge of large environments, SLAM algorithms also work best for static environments, where features do not change over time. This presents further complications in the case of a warehouse since items are constantly in circulation, altering potentially useful features tracked by a SLAM algorithm. Furthermore, SLAM works best for well-lid environments where the lighting conditions do not change over time. (Feigl et al., 2020) investigates the use of AR in large industrial environments, using three of the largest and most well-established AR platforms [7]. These AR platforms are ARKit for Apple products, ARCore for Android devices, and Microsoft HoloLens. The study by (Feigl et al., 2020) found that the reliability of SLAM algorithms used on the aforementioned AR platforms could perform better in large industrial environments of around $1,600m^2$. They show that the tested AR systems accumulate an error of around 17m per 120m. The study also revealed that adding natural features improved tracking reliability but not enough to be considered useful for the industrial context. In addition, AR systems perform best in static environments and when only the user is moving. Good position accuracy requires many identifiable features, which can be challenging in an industrial context where lighting and the presence of moving objects can interfere with tracking accuracy. However, the study found that Microsoft HoloLens worked best among the three platforms tested, followed by ARKit, and then ARCore. Despite the limitations identified, the study concludes that AR has great potential in the industrial context if further research is conducted to address the challenges identified. Although this study is useful, research on AR systems in industrial settings is limited. [7]

2.4.6 AR in Warehouses: Summary

In summary, the key findings from the studies on AR in warehouse settings indicate that AR has the potential to increase efficiency and reduce errors in warehouses [22]. (Stoltz et al., 2017) found that AR can improve accuracy in task execution, enhance visualization, and provide real-time information to employees. (Schwerdtfeger et al., 2009) demonstrated that using AR for order picking could improve efficiency and error rates and did not cause higher user strain than traditional methods [19]. (Schwerdtfeger et al., 2008) also investigated the effectiveness of different visualization approaches for picking items, concluding that various techniques were suitable depending on the user's situation [20]. However, a significant challenge for AR in warehouses is the drift error in large environments. (Feigl et al., 2020) found that the reliability of SLAM algorithms used in AR platforms did not perform well in large industrial environments, accumulating significant errors. While improvements can be made by adding natural features, further research is needed to address these challenges and fully realize the potential of AR in the industrial context [7].

2.5 Initial Problem Formulation

As we have identified the potential of using AR technology to guide warehouse employees at Selek, it is crucial to understand the suitability of existing SLAM algorithms in this context. Evaluating their limitations and exploring possible improvements can pave the way for a more effective AR-based navigation system in large industrial warehouses. Therefore, the following initial problem formulation has been created:

Which state-of-the-art SLAM algorithms are suitable for an indoor AR navigation system, what are their limitations, and how can they be improved?

Chapter 3

Technical Analysis

3.1 Introduction

In the previous chapter, we identified the key problems faced in large-scale warehouse operations concerning inventory localization and the potential of Augmented Reality to address these problems. This chapter aims to explore the technical aspects of AR in more depth, with a particular focus on the core algorithms that power this technology. A significant component of an AR system is its ability to accurately determine and track its position in real time, a capability largely provided by SLAM algorithms. Thus, understanding SLAM and its variants forms an essential part of this analysis. The initial problem formulation should be answered at the end of this chapter. Finally, a final problem statement will be presented based on the findings.

Augmented reality is a technology that overlays computer-generated content on top of the real world. This is accomplished using an AR-enabled device, such as a smartphone or AR glasses, that displays real-world and digital information while tracking the user's location and real-time movements. The information shown in AR can be text, images, video, or 3D objects. In the specific case of guiding a user to a location in a warehouse, an arrow might be shown along with a text indicating the distance to the location.

However, developing such an AR solution presents several technical challenges, including accurate tracking and localization of the AR-enabled device, accurately displaying the AR content, and ensuring a seamless and natural user experience. To overcome these challenges, a thorough understanding of the current state and latest advancements in AR technology must be gained. [24]

A critical component of AR technology is positional tracking, which allows the AR-enabled device to accurately determine its pose in real-time. Positional tracking is vital since knowing the pose of the AR device allows it to accurately display AR content placed at a fixed position in the real world. The pose can be described with 6 degrees of freedom (6DoF), where three DoFs are dedicated to describing the position, and the other three DoFs describe the orientation. The pose should be updated in real time while the AR-enabled device moves through space. Estimating an accurate real-time pose of the device is crucial for displaying properly positioned and aligned AR content. Doing so requires a combination of advanced sensors and software.

(Schwerdtfeger et al., 2009) tracks the AR headset using visual markers and cameras [19]. This is not an ideal solution since the visual markers are inconvenient to wear, and it requires a large setup with multiple cameras, which does not easily scale to a large warehouse. Fortunately, other solutions exist for positional tracking. GPS and compass are two of the most common technologies for tracking position and orientation. However, these technologies can not alone obtain a 6DoF pose.

Furthermore, they need to be more accurate for AR, especially in indoor environments. Therefore, other, more accurate systems are often used for AR experiences. One such common system is a SLAM algorithm. This is especially useful if a phone is used to show AR content since visual markers would be very inconvenient on a phone. [19]

3.2 Simultaneous Localization and Mapping

3.2.1 What is SLAM?

SLAM is a class of algorithms commonly used to solve the problem of positional tracking in augmented reality. It is an algorithmic approach that allows an AR-enabled device to build a map of its surroundings while simultaneously estimating its pose within that map. SLAM uses sensors, such as cameras and accelerometers, to detect and track features in the environment.

One of the key benefits of SLAM is its ability to work in environments where GPS and compass-based tracking systems are not reliable or accurate enough, such as indoor environments or areas with a limited GPS signal. This makes SLAM a popular choice for AR applications that require accurate positional tracking.

3.2.2 ORB-SLAM

One of the state-of-the-art SLAM algorithms is called ORB-SLAM3, which is a succeeding version of the former ORB-SLAM2 and ORB-SLAM algorithms [11, 12, 4]. The original ORB-SLAM algorithm was published in 2015, by (Raul et al., 2015). It presents a feature-based SLAM algorithm that uses ORB features to perform both tracking and relocalization. ORB-SLAM works with a monocular camera and uses a bundle adjustment algorithm to estimate the camera pose.

One of the main contributions of ORB-SLAM is that it uses the same ORB features to perform both tracking, mapping, relocalization, and loop closing, which improves efficiency and reliability. It also uses a covisibility graph to limit the scope of tracking and mapping to a local area, which improves the system's efficiency and robustness. This is especially beneficial for real-time tracking in large environments since tracking becomes independent of the size of the map. Furthermore, ORB-SLAM implements a new loop-closing algorithm and real-time camera relocalization. [11]

There are a few reasons why (Raul et al., 2015) use ORB features, one of which is that they are rotation-invariant, meaning they can match features even if the camera is rotated. This is important because the camera's orientation can change as it moves through the environment, and the system needs to be able to track its position and orientation accurately to create an accurate map of the environment. Furthermore, ORB features are scale-invariant, which means they can accurately match features regardless of their size in the image. This makes them particularly useful for ORB-SLAM, as it allows the system to detect and track features at different distances from the camera. Additionally, a SLAM system should be able to run in real time. Thus, ORB-SLAM needs to run fast, and ORB features are computationally efficient, which makes them well-suited for real-time applications such as running AR on a phone. [17]

Using a monocular camera, as done in the original ORB-SLAM algorithm, shows promising results. Using a monocular camera, depth and movement can be estimated using visual odometry techniques. This is done by observing the scene from two different viewpoints, finding matching features, and minimizing a reprojection error. This can be achieved with a monocular camera, as in ORB-SLAM, by moving the camera around. However, estimating depth can be done more accurately with stereo vision, where two fixed cameras are used to estimate the depth using stereo triangulation.

Furthermore, depth could also be estimated using other sensors, such as LiDAR. This is the main contribution of the next iteration of ORB-SLAM, namely ORB-SLAM2. As described by (Raul et al., 2016), ORB-SLAM2 can use either monocular, stereo, or RGB-D cameras. As a result of using stereo vision or RGB-D cameras, a more accurate depth estimation of the ORB features can be obtained, thus improving the system's overall accuracy. A pre-processing step is incorporated to make the algorithm operate independently of which sensor is used, which extracts ORB features along with their depth. [12]

In 2020 ORB-SLAM3 was released by (Carlos et al., 2020), introducing further improvements to the algorithm. The two main contributions of ORB-SLAM3 are the use of inertial information, making it a visual-inertial SLAM system, and the ability to perform multi-map SLAM. By combining inertial information obtained from an IMU with the previously explained feature-based method, a Maximum-a-Posteriori approach can be used to perform even more accurate tracking and mapping. (Carlos et al., 2020) demonstrates a two to ten-times improvement by incorporating inertial information. [4]

The second improvement of ORB-SLAM3 is the use of multiple maps. If the system cannot locate its position with a previous map, it initializes a new one. And if features from two maps start to overlap, they are simply merged, forming one new map. This makes the algorithm more robust since it only breaks down if tracking is lost for a short period. [4]

3.2.3 Loop Closing

Visual SLAM systems are very closely related to visual odometry systems since they both use visual information to predict changes in the position of a camera. One of the main components distinguishing a SLAM system from an odometry system is the ability to perform loop-closing.

When obtaining data from sensors, rather it is encoders, cameras, or IMUs, there will always be some noise and other inaccuracies present. These errors should be minimized to the extent possible to maximize the accuracy of odometry algorithms. However, minor errors will always be present, causing odometry algorithms to drift over time. This means that the estimated position becomes more and more inaccurate the longer the algorithm is running. ORB-SLAM attempts to correct this error by using loop-closing.

The loop-closing algorithm used in ORB-SLAM is based on an algorithm developed by (Mur-Artal et al., 2014), which also uses ORB features to detect loops. The idea of loop-closing is to detect when the camera detects the same features as in a previous keyframe. Once this happens, the system can measure the discrepancy between the previous and new keyframes. By knowing the error between the two keyframes, it can be corrected for in all the keyframes in between, under the assumption that the accumulated error is somewhat consistent. [13]

3.3 Visual Odometry

Visual odometry (VO) is the process of estimating the motion of a camera by analyzing the changes in visual information captured in consecutive frames. It is an essential component of SLAM systems, which aim to build a map of an unknown environment while simultaneously estimating the camera's pose within that environment. VO is different from a full SLAM system as it only focuses on estimating the local motion of the camera, while SLAM also involves mapping the environment and estimating a global pose using techniques such as loop closing.

In literature, there are two main categories of visual odometry: geometric and non-geometric. Geometric methods estimate the camera's motion based on the geometry of the scene and are further divided into feature-based, appearance-based, and hybrid methods. Feature-based methods involve detecting and matching distinctive features across image frames. Popular feature detectors include SURF, ORB, BRISK, and more. On the other hand, appearance-based methods estimate the camera's motion based on the changes in the intensity values of the entire image or optical flow. Region-based matching and optical flow-based methods are two types of appearance-based methods. Hybrid methods combine both feature-based and appearance-based methods, using monocular, stereo, or RGB-D sensors. [15]

In a feature-based VO algorithm, it is essential to have well-defined and easily distinguishable features. These features allow the algorithm to match them across consecutive frames reliably, enabling accurate estimation of the camera's motion. The quality of the features can significantly impact the performance of the algorithm, as poorly defined or ambiguous features may lead to incorrect matches and ultimately result in inaccurate pose estimation.

Non-geometric visual odometry methods are another recent paradigm shift in the field. They use learning-based techniques to train regression models that estimate the camera's motion from image sequences without requiring prior knowledge of the camera's intrinsic parameters. These methods have gained popularity in recent years thanks to the evolution of machine learning techniques. [15]

One of the challenges of visual odometry is drift, which occurs when the estimates of the camera's motion accumulate errors over time, leading to inaccurate pose estimates. To mitigate this, SLAM systems often use loop closure and Bundle Adjustment (BA), which attempts to correct the errors and refine the estimated poses over longer time intervals. To mitigate drift, it can be helpful to incorporate an initialization step, as it initializes the camera's position and orientation, which is necessary for estimating the camera's motion. The initialization step can be done using various methods, including using a priori information, such as GPS or IMU data, or by detecting and matching distinctive features across the first few frames of the video sequence. The latter approach is commonly used in feature-based methods, where the first few frames are used to extract features and match them across the frames to initialize the camera's pose. [15]

In summary, visual odometry is a crucial component of SLAM systems, which estimate the local motion of a camera in unknown environments. Geometric methods, such as feature-based, appearance-based, and hybrid methods, estimate the camera's motion based on the scene's geometry. In contrast, non-geometric methods use learning-based techniques to estimate the motion directly from image sequences. However, drift is a major challenge in visual odometry, which requires loop closure and Bundle Adjustment techniques to mitigate the errors and refine the estimated poses. Furthermore, accurate initialization of the system can help to reduce drift and is thus an important component of VO. In the case of feature-based methods, the quality of the features is particularly important, as well-defined and easily distinguishable features allow for more accurate motion estimation and, ultimately, better performance.

3.4 Feature Detection

3.4.1 What Is Feature Detection?

One of the critical components of a feature-based odometry algorithm is its feature detection algorithm. In computer vision and image processing, a feature refers to a distinct and recognizable pattern or structure within an image, such as corners, edges, or textures. A feature detection algorithm aims to identify and extract these features from an image, providing valuable information for subsequent tasks such as object recognition, tracking, and localization. By reliably detecting and matching features across different images or frames, an odometry algorithm can estimate the relative motion and position of the camera or objects within the scene, enabling accurate navigation and mapping for applications such as AR.

Various feature detectors exist and can be divided into handcrafted and learned features. Handcrafted feature detectors, such as SIFT, SURF, and ORB, serve the purpose of extracting key points, which describe pixel positions in the image, as well as feature descriptors, which might be vectors that represent or encode information about the specific features [10, 2, 17]. On the other hand, learned features can be generated using deep learning techniques, such as CNNs (convolutional neural networks). These techniques have gained significant popularity in recent years due to their ability to automatically learn patterns and representations, including key points and feature descriptors, from large amounts of data [6].

One of the important aspects to consider for feature extractors is their ability to be scale and rotation invariant. Scale invariance refers to the ability of an algorithm to recognize and match features across images with varying scales or resolutions. This is particularly important in scenarios where objects may appear larger or smaller due to changes in distance or perspective. Rotation invariance, on the other hand, enables the algorithm to identify and match features regardless of their orientation in the image. This is critical when dealing with images captured from different angles or in situations where the camera or objects within the scene undergo rotation.

3.4.2 ORB features

As mentioned, ORB-SLAM uses the ORB feature detector [17]. The ORB feature detector is a rotation and scale invariant method for detecting and describing local features in images, combining the strengths of the FAST (Features from Accelerated Segment Test) keypoint detector and the BRIEF (Binary Robust Independent Elementary Features) descriptor. [17]

The FAST keypoint detector

FAST compares the brightness of a specific pixel in an image with 16 of its surrounding pixels. If eight or more of the 16 surrounding pixels are either darker or brighter than the center pixel, it is selected as a key point. By applying this algorithm to each pixel in an image, points containing useful features can be identified. To make FAST scale-invariant, ORB uses an image pyramid, enabling FAST to find useful key points at different scales of the image. [16]

The ORB feature detector achieves rotation invariance by computing the orientation of each key point using the intensity centroid method. This method calculates a patch's intensity-weighted center of gravity surrounding the key point, effectively determining its orientation. By rotating the image patches according to their orientations before applying the BRIEF algorithm, ORB obtains rotation invariance.

BRIEF

After finding a bunch of scale and rotation-invariant features in an image, BRIEF is then used to convert their associated image patches to a binary feature vector. To prevent the feature vectors from being sensitive to high-frequency noise, BRIEF starts by smoothening the image using a Gaussian kernel. The feature vector is constructed hereafter by finding pairs of random points in the image patch. For each pair of points, their intensities are compared. If the first point of a pair is the brightest, then a value of 1 is added to the binary feature vector. Otherwise, a value of 0 is added. The random pairs of points are drawn from a Gaussian distribution, and each binary value in the feature vector has its random point pair. [3]

3.4.3 Learned Feature Detectors

As mentioned earlier, learned feature detectors leverage deep learning techniques, such as convolutional neural networks (CNNs), to automatically learn patterns and representations from large amounts of data. These methods have gained significant popularity in recent years due to their ability to generalize well to various images and feature types, outperforming handcrafted feature detectors in many cases.

(DeTone et al., 2018) proposed a deep learning-based feature detection algorithm called SuperPoint [6]. This algorithm uses a neural network architecture that enables it to perform one forward pass for both finding key points and feature descriptors. This method differs from other common techniques that first find relevant key points, followed by computing the feature descriptors. The SuperPoint architecture uses a VGG-style neural network as an encoder, followed by two separate networks for computing the key points and the feature descriptors. [6]

Training a CNN, such as SuperPoint, to detect key points and feature descriptors presents a problem. Since semantic labels do not easily define key points, a human cannot reliably label such a dataset. Thus, no large dataset exists with key point labels. To overcome this problem, the authors rely on a synthetic dataset and self-supervised learning to train the network. They generate a synthetic dataset of 3D shapes and extract key point labels at the corners of the 3D shapes. [6]

The authors start by training a network they call MagicPoint on the synthetic dataset. Hereafter, they train the SuperPoint network on the MS-COCO 2014 dataset, where key point labels have been generated using the MagicPoint network and a technique they call Homographic Adaptation. This approach avoids human annotation and allows the SuperPoint network to be trained in a self-supervised manner. [6]

3.5 ARKit In Selek's Warehouse

3.5.1 Understanding Selek's Warehouse and its Challenges

As previously explained, one of the main problems with SLAM algorithms is the drift that accumulates over time. This has been demonstrated by (Feigl et al., 2020) [7]. However, it might be insightful to test this problem in the context of Seleks warehouse since this is where to motivation for this project stems from. To do so, it might be useful to understand the layout of Selek's warehouse and its bin positions.

Selek's warehouse consists of two warehouses connected by a small hub where orders are packed and shipped from. Both warehouses contain several aisles that are further divided into subsections. Each subsection has some shelves, and each shelf is subdivided into individual bins containing a specific item. In some cases, a bin contains multiple different items. When a warehouse employee needs to pick an item, they are presented with a bin code describing which bin the item is located on. A bin code at Selek includes three numbers describing the aisle, section, and shelf, respectively, as shown in Figure 3.1. The bin code is shown at each physical bin in the warehouse, along with a barcode containing the bin code. This system is meant to make it easy for warehouse employees to navigate the warehouse. However, it takes time for new employees to learn the system, presenting the opportunity to use AR to guide the employee. [1]

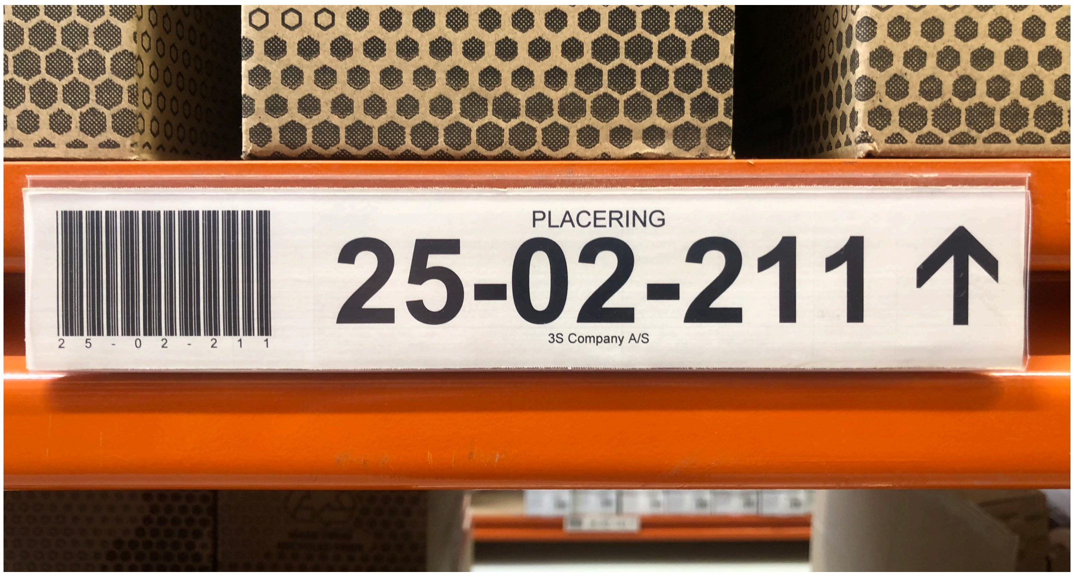


Figure 3.1: Example of bin code in the warehouse at Selek with its associated barcode.

At Selek, they use a custom-made app at the warehouse when picking, restocking, and moving items around. Among other features, this app incorporates a small camera view for scanning barcodes. This camera view could also be used to display AR content to the user. The app at Selek runs on iPhone, and thus ARKit might be used to achieve this. However, as shown by (Feigl et al., 2020), the accuracy of ARKit is inadequate when used in large industrial environments such as the warehouse at Selek. Thus, it might be advantageous to explore alternative solutions. [7]

If AR should be used to guide warehouse employees to a specific bin, the system should also know the physical positions of the bins. Thus, each bin should be represented with a point in 3D space with respect to the same coordinate system. Additionally, when a user opens the app, it does not initially know its position within this coordinate system, and therefore, it has to relocalize itself somehow. Solving these problems is essential if AR is to guide a user to a specific bin.

Having gained an insight into the layout, options, and tools used in Seleks warehouse sets the stage for the creation of a test app for iOS that uses ARKit. As explained in section 2.4.5, ARKit experiences drift over time when used in large industrial environments. The rest of this section explores the capabilities of ARKit and presents a test app that is used to test the accumulated drift when used in Selek's warehouse.

3.5.2 ARKit Tracking

As discussed in Chapter 1, this report is motivated by the idea of AR navigation in a warehouse. Specifically, a goal is to integrate an AR solution into Selek's existing iOS app used for warehouse management. While other frameworks exist for different devices, such as ARCore for Android and the HoloLens framework for Microsoft's HoloLens, Apple has developed ARKit specifically for iOS. As Selek utilizes an iOS app, this section will exclusively focus on ARKit. [7]

Although Apple has not published detailed information regarding the algorithm used for tracking, an overview of the main concepts is presented in a video from their annual developer conference (WWDC) in 2018 [9]. These concepts closely resemble those implemented in ORB-SLAM3. Like ORB-SLAM3, ARKit applies a feature-based visual-inertial odometry algorithm. Although the type of features utilized by Apple is unknown, it is likely that they use something computationally inexpensive yet robust to different scales and orientations, such as ORB features. Additionally, ARKit's tracking algorithm also employs loop closure. [9]

When using ARKit, it is possible to save a tracking state by saving an object known as an ARWorldMap. This allows users to return to the AR experience after closing and reopening the app or experiencing other interruptions. An ARWorldMap includes all the feature points tracked by the system and can be loaded back into ARKit when the user wishes to return to the AR experience. This feature could be particularly useful for AR navigation in a warehouse, as warehouse employees should be able to exit and re-enter the AR experience easily. [9]

3.5.3 Test App

To evaluate the usefulness of ARKit and uncover some of its potential shortcomings, a test app has been developed that can be used to test the accuracy of the system. To guide users to bin positions within a warehouse, it is essential to first record the position of the bins. At Selek, each bin in the warehouse is associated with a barcode, as seen in Figure 3.1. A possible approach to mapping these bin positions could involve scanning the barcodes while the ARKit's tracking algorithm is active, thereby recording the position of the barcodes.

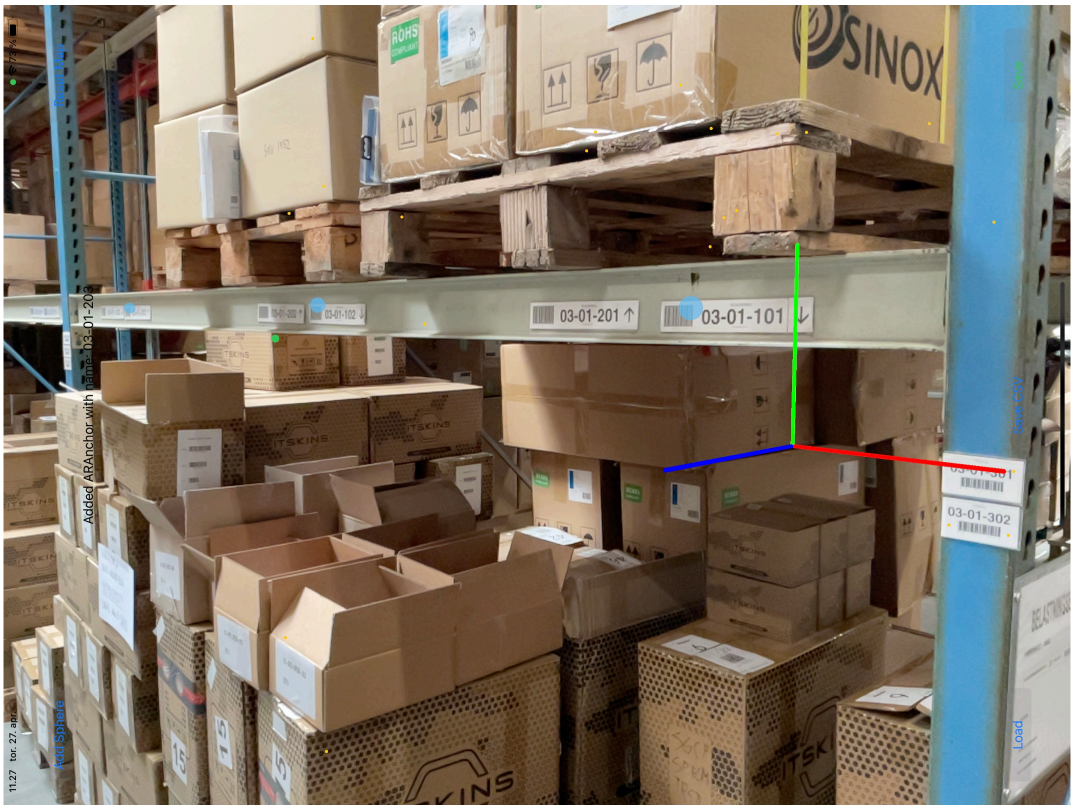


Figure 3.2: A screenshot of the test app showing blue spheres placed at the barcodes. Additionally, yellow dots can be seen, representing ARKit's features for tracking.

To accurately determine the positions of the barcodes, a ray-tracing algorithm was employed, measuring the distance between the barcodes and the camera. This method enabled the app to store the 3D positions of the barcodes, which are represented as blue spheres in Figures 3.2 and 3.3.

A total of four tests were conducted to evaluate the system's performance, using six barcodes for each test. In each test, the process involved scanning each barcode once and then scanning all of them again. The error distance between the two scans of the same barcode was then measured. Two scenarios were considered in each test: scanning all barcodes without closing the app between the sets of scans and scanning barcodes with the app closed between the two sets of scans. For the latter scenario, an ARWorldMap was utilized to relocalize the camera position upon reopening the app. These tests were carried out at short and long distances to assess the system's performance under different conditions.

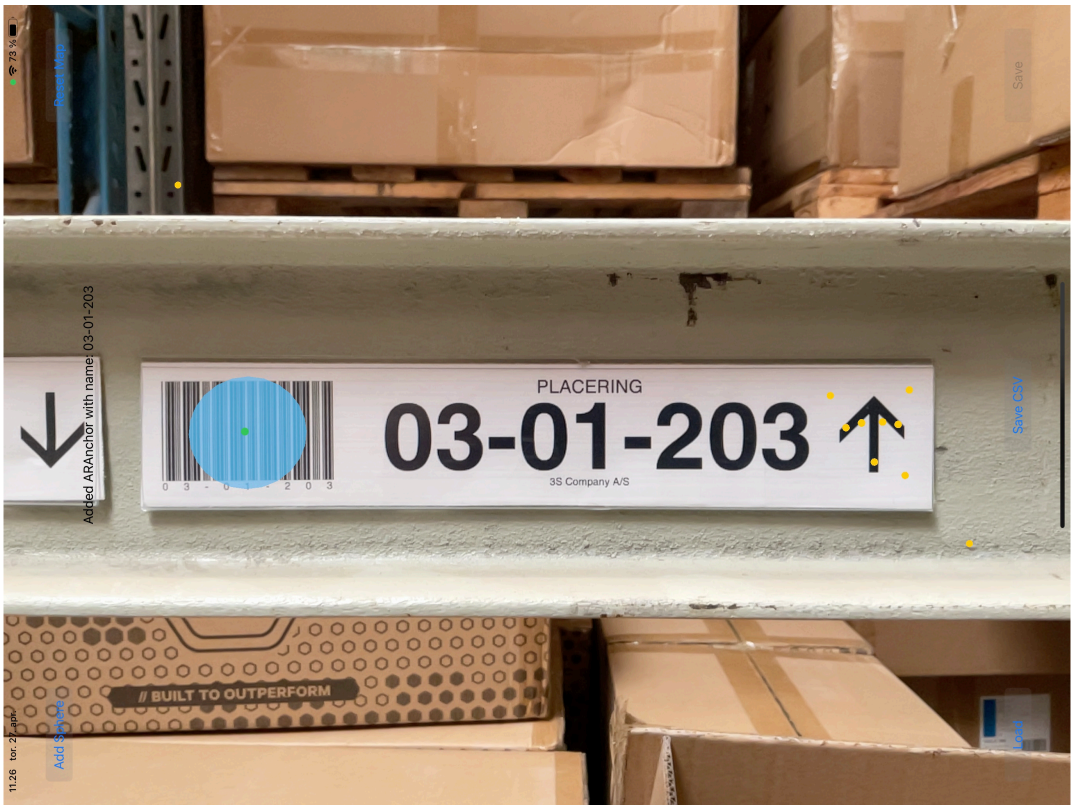


Figure 3.3: A screenshot of the test app showing blue spheres placed at the barcodes. Additionally, yellow dots can be seen, representing ARKit's features for tracking.

For short-distance testing, barcodes within a single aisle were scanned, resulting in a distance of $\sim$ 13 meters between the barcodes. In contrast, long-distance testing involved scanning three barcodes at one end of the warehouse and three more at the opposite end, creating a distance of $\sim$ 42 meters between the scanned barcodes. These tests aimed to examine ARKit's performance in a warehouse environment and to investigate the impact of distance on system performance. The test results, shown in Figures 3.4 and 3.5, reveal that long-distance tracking experiences significantly more drift than shorter distances.

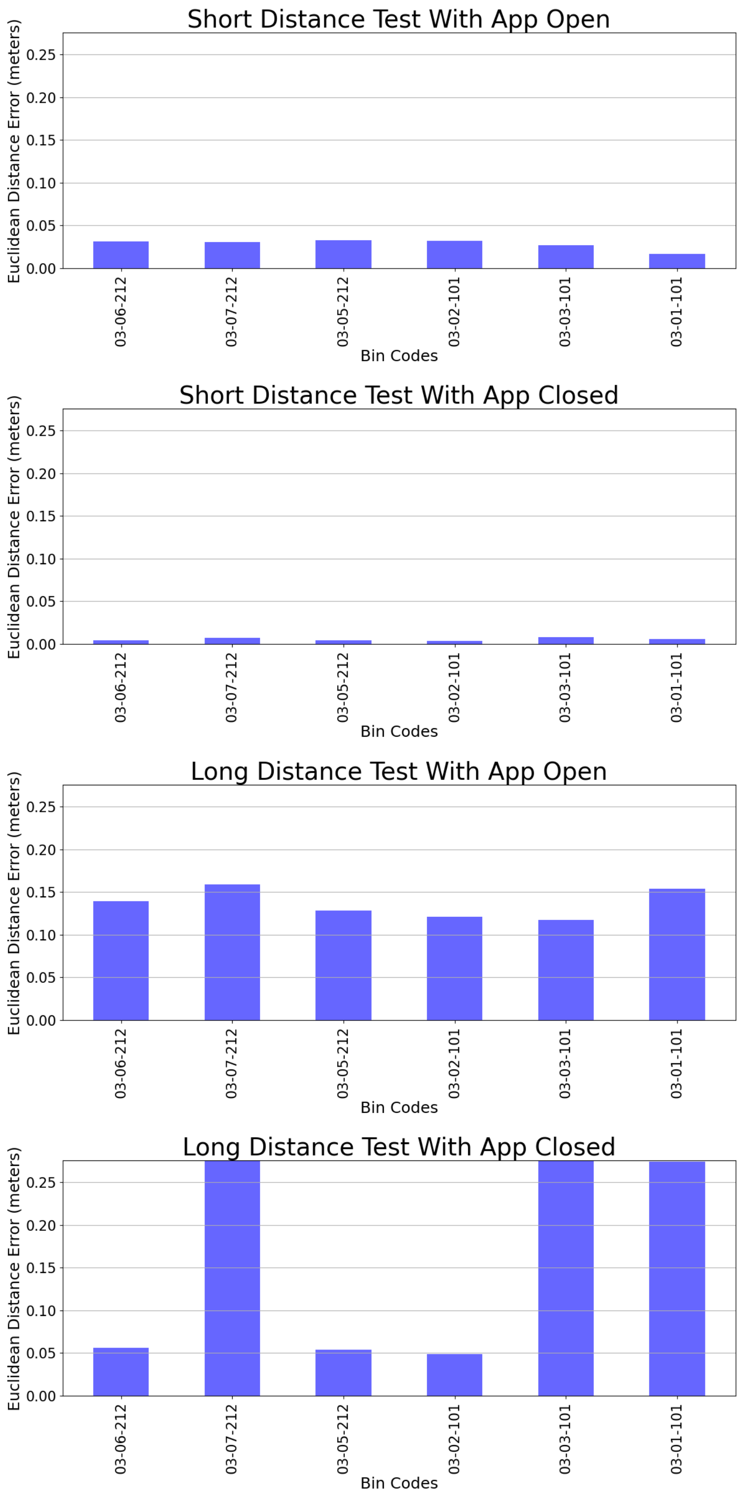


Figure 3.4: Euclidean Distance Error per Bin for Each Test

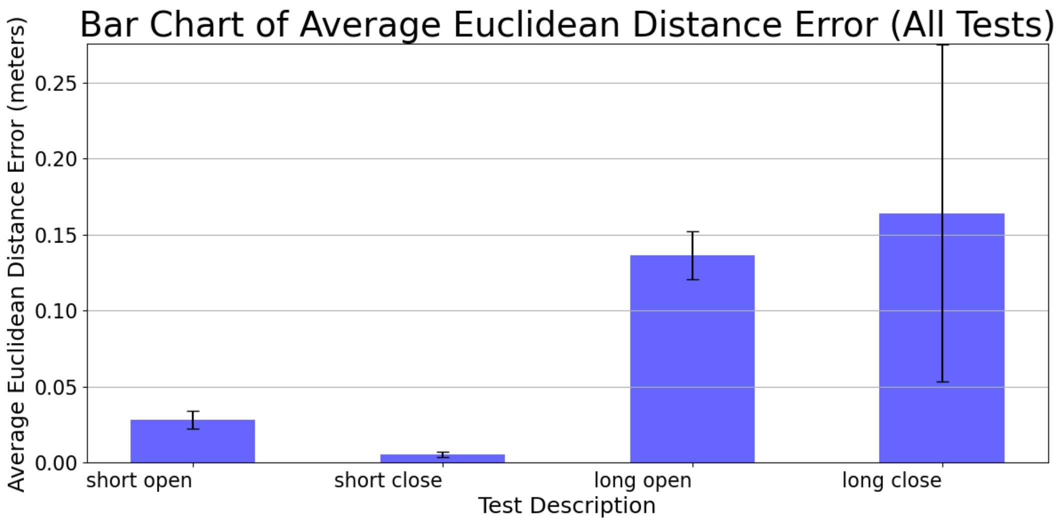


Figure 3.5: Average Euclidean Distance Error (All Tests)

As shown in the bar plot of average errors for each of the four tests (Figure 3.5), the error increases significantly for longer distance tests. Interestingly, relocalizing the camera position using ARWorldMap after reopening the app appears to have a lesser impact on the error. It might be assumed that as more bins are added to the system, the error may increase further due to the increased number of objects the system has to keep track of. These results align with the findings of (Feigl et al., 2020), as described in Section 2.4.5. The conducted tests have confirmed their observations, reinforcing the understanding of the impact of drift error in large-scale warehouse environments.

3.6 Conclusion

In conclusion, the drift issue in feature-based visual-inertial odometry algorithms, such as ARKit and ORB-SLAM3, poses a significant challenge for large-scale industrial environments like warehouses. The ability to accurately identify and track distinguishable features is crucial for the performance of feature-based SLAM algorithms. Recent studies on deep learning-based feature extractors have shown promising results in this context.

However, the creation of labeled datasets containing key point features for these deep learning-based feature extractors is labor-intensive and semantically ill-defined. This makes it difficult for humans to create high-quality labels reliably. It is worth considering using convolutional neural networks (CNNs) pre-trained on other tasks to address this challenge.

3.7 Final Problem Formulation

Having discussed the state-of-the-art SLAM algorithms and identified the key challenges, it is clear that improvements are necessary for their application in large industrial settings like warehouses. Given the complexity of these algorithms, this report will now focus on developing a simpler visual odometry solution using a monocular camera. To address the drift problem, the next chapters will explore using a pre-trained deep learning-based feature extractor.

Based on the insights gained from this chapter, the next chapter will present a proposed feature extractor inspired by ORB, which leverages the FAST algorithm to detect key points. Additionally, pre-trained CNNs will be utilized to generate feature descriptors for each key point. Several delimitations have been made to maintain a focused scope, including using monocular grayscale visual odometry for testing the proposed feature extractor instead of using a more complicated algorithm such as ORB-SLAM.

Thus, the final problem formulation is as follows:

How can a pre-trained deep learning-based feature extractor be designed and implemented, and how might it reduce drift in a VO system, a crucial component of SLAM algorithms, for AR applications in warehouse management?

Chapter 4

Implementation of proposed feature extractor

4.1 Introduction

The previous chapter examined SLAM algorithms with a special emphasis on ORB-SLAM and its successors. It was shown that although these algorithms have undergone massive improvements in the past years, they are still not suited for AR in large industrial environments, such as in a warehouse. This is due to the significant drift that these algorithms accumulate over time.

As previously explained, many variations of SLAM algorithms exist, all with many different subcomponents that make them up. The purpose of this chapter is to attempt to improve one of these sub-components, namely the feature extractor. To better develop and test the proposed feature extractor, a relatively simple visual odometry algorithm has been used [14]. This algorithm originally uses ORB features for feature extraction, making it a feature-based visual odometry algorithm. This chapter proposes a deep learning-based feature extractor, drawing inspiration from the ORB feature extractor. In the following sections, the VO algorithm used to test the proposed feature extractor is presented, followed by a description of the proposed feature extractor.

The source code for the proposed feature extractor is implemented in the 'deep-Features.py' Python script, which can be found in the "VisualOdometry" directory of the GitHub repository: `https://github.com/BSPetersson/Master_Thesis.git`

4.2 Monocular Visual Odometry

The algorithm discussed in this section is a feature-based visual odometry algorithm designed to work with monocular grayscale images. Implemented as a Python script by (Nielsen, 2022), its primary objective is to estimate the 3D trajectory of a monocular camera using a sequence of monocular grayscale image frames [14]. The algorithm locates ORB features in the images to match key points between consecutive frames. Subsequently, it computes the essential matrix, which aids in determining the relative motion between the frames. As a result, the camera's estimated trajectory in 3D space is obtained. This trajectory can then be compared to a ground truth trajectory, allowing for an evaluation of the algorithm's performance. In figure 4.1, this algorithm is illustrated as a block diagram, showing its main components. The first block in "Feature Matching" represents a specific feature extractor. Initially, the algorithm used ORB, but in the next section, another proposed feature extractor is introduced, using either VGG-16 or MobileNetV2. Thus, changing this specific part of the VO algorithm has been done to test the different feature extractors.

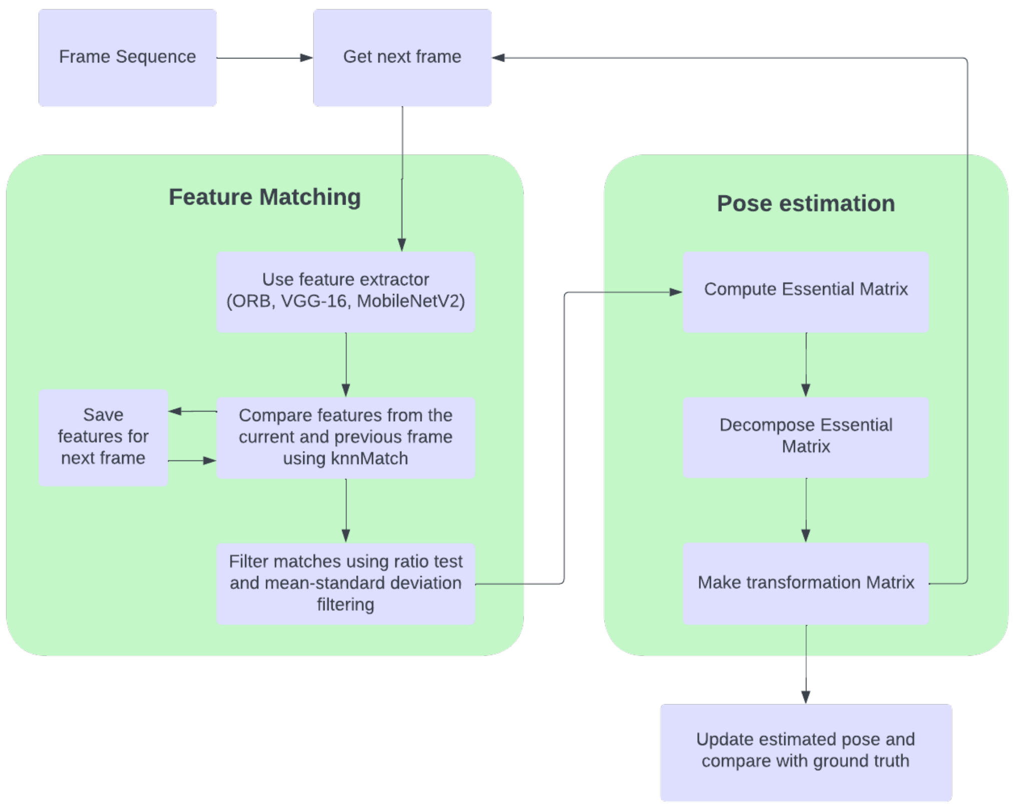

Frame Sequence
Get next frame
Feature Matching
Use feature extractor (ORB, VGG-16, MobileNetV2)
Save features for next frame
Compare features from the current and previous frame using knnMatch
Filter matches using ratio test and mean-standard deviation filtering
Pose estimation
Compute Essential Matrix
Decompose Essential Matrix
Make transformation Matrix
Update estimated pose and compare with ground truth

Figure 4.1: This block diagram shows the flow of the visual odometry algorithm used to test the proposed feature extractor.

As mentioned, the algorithm finds ORB features in each of the frames. Each ORB feature is described with a key point position in the image and a feature descriptor in the form of an array with numbers. The goal is to find similar feature descriptors in consecutive frames, hopefully corresponding to the same real-world features, also referred to as feature matching. To do this, a k-nearest neighbors matching algorithm is used. This algorithm takes an array of feature descriptors from two consecutive frames. It then computes the Euclidian distance between the feature descriptors in the two frames. For each feature descriptor in the first frame, it returns the two closest feature descriptors in the second frame. The reason for returning two matches is to apply a ratio test, which helps filter out ambiguous matches and retain only high-quality matches. The ratio test compares the distances of the two closest matches. If the ratio of the distances is less than a predefined threshold, it is considered a good match. This is because a low ratio indicates that the closest match is significantly closer than the second-closest match, suggesting that the match is unambiguous and more likely to be a true correspondence between the two frames. By filtering out matches that are too close or ambiguous, the algorithm ensures that only the most reliable matches are used for further processing.

The algorithm refines the set of matches further after applying the ratio test by implementing a mean-standard deviation filtering technique. This method evaluates the Euclidean distances, in pixel values, between all pairs of matched key points in consecutive frames. By calculating the mean distance and standard deviation of these distances, the algorithm gains insight into the general distribution of the matches. A threshold is then established based on the mean distance, and a multiple of the standard deviation and matches with distances greater than this threshold are discarded as potential outliers. This approach helps to eliminate matches that may have passed the ratio test but still show significant pixel-wise distance discrepancies between the corresponding key points, which could indicate that they are not genuine correspondences. By filtering out such matches, the algorithm enhances the overall quality and reliability of the feature-matching process. The mean-standard deviation filtering technique is particularly useful for removing matches that might have resulted from noise, repetitive patterns, or other factors leading to ambiguous correspondences. Consequently, the algorithm ensures that only the most reliable and accurate matches are used for further processing, ultimately improving the performance and precision of the system.

Having found matching features for two consecutive frames, the goal is to estimate the relative movement between them. To achieve this, the algorithm finds an essential matrix using the matched key points and the camera's intrinsic parameters. The essential matrix is a 3x3 matrix that encodes information about both the relative translation and rotation, and it can be decomposed into a translation vector and a rotation matrix. However, decomposing the essential matrix results in four solutions, and the algorithm must pick the correct one. This is done by projecting the 3D points reconstructed from the matched key points onto both camera views and checking the number of points with positive depth values (z coordinate) in both views. The correct solution is the one that results in the highest number of points with positive depth values in both camera views, as this implies that the points are in front of both cameras, which is consistent with the scene geometry. Having found the correct translation vector and rotation matrix, the algorithm constructs a transformation matrix. Obtaining a transformation matrix for each frame enables the algorithm to calculate a global estimate of the new camera pose by multiplying them.

Now that the visual odometry algorithm has been presented, the following section will introduce the proposed deep learning-based feature extractor that aims to improve the feature extraction process.

4.3 Proposed Deep Learning Based Feature Extractor

4.3.1 Introduction To Proposed Feature Extractor

In recent years, deep learning techniques have gained significant popularity and have been successfully applied to various computer vision tasks. As mentioned in Section 3.4.3, one such task is feature extraction, where a significant challenge lies in obtaining a labeled dataset of key point features due to the difficulty for humans to identify interesting key points within an image consistently [6]. This chapter proposes a novel approach that combines the FAST key point extractor with deep neural networks, drawing inspiration from the ORB algorithm and leveraging pre-trained CNNs like VGG-16 and MobileNetV2. The proposed feature extractor maintains scale invariance by constructing an image pyramid and applying the FAST algorithm at each scale. The hypothesis for rotation invariance is that the CNNs can detect the same features regardless of the rotation. By utilizing the pre-trained CNNs to generate feature descriptors, this approach aims to eliminate the need for a labeled key point dataset with corresponding feature descriptors and improve the feature extraction process. Figure 4.2 illustrates this algorithm with a block diagram, showing the main components of the algorithm.

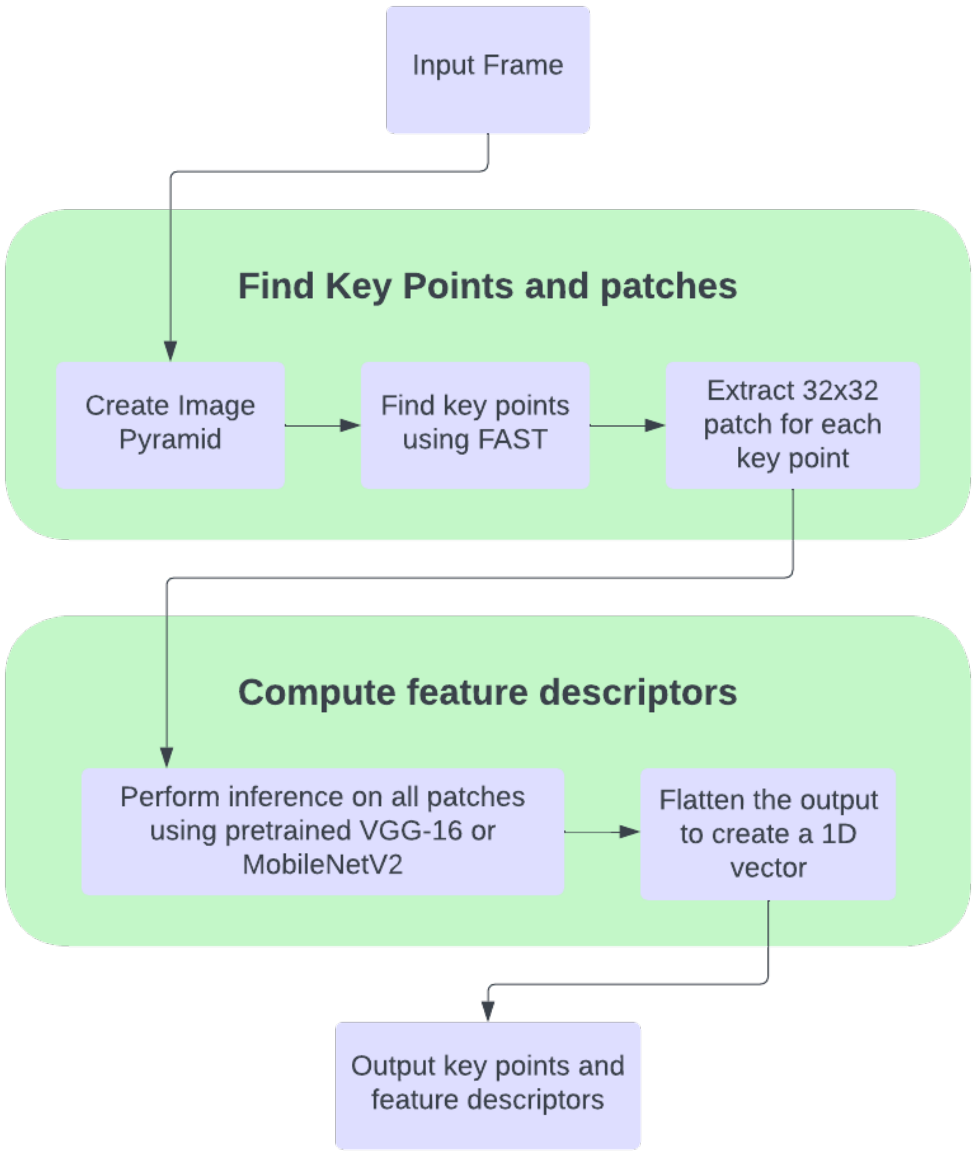

Input Frame
Find Key Points and patches
Create Image Pyramid
Find key points using FAST
Extract 32x32 patch for each key point
Compute feature descriptors
Perform inference on all patches using pretrained VGG-16 or MobileNetV2
Flatten the output to create a 1D vector
Output key points and feature descriptors

Figure 4.2: Block diagram representing the proposed feature extractor using either MobileNetV2 or VGG-16 to compute feature descriptors.

4.3.2 Using FAST for Key Point Extraction

The proposed feature extraction approach employs the FAST algorithm, an efficient and effective method for identifying key points within images. Like ORB features, the FAST algorithm detects key points while maintaining the advantages of deep learning techniques. An image pyramid is constructed to enhance scale invariance, and the FAST algorithm is applied to each scale of the image pyramid. This method ensures that key points are detected at different scales, making the feature extraction process more robust to variations in scale. The key points detected by the FAST algorithm will then be utilized as input for the pre-trained CNNs to generate the feature descriptors, which will be described in the following sections.

4.3.3 VGG-16

A feature descriptor must be constructed upon identifying interesting features within an image. Ideally, this feature descriptor should exhibit a certain degree of scale and rotation invariance. To compute these feature descriptors, a CNN is utilized. An image patch surrounding each key point is extracted and used as input for the CNN. The output of the CNN is a feature descriptor in the form of a vector.

In this project, two different CNNs are implemented and tested as a part of the proposed feature extractor, one of which is VGG-16. VGG-16 is a deep convolutional neural network originally used to classify images. In 2014, VGG-16 won the ImageNet challenge with a 92.7% accuracy. VGG-16 consists of multiple convolutional layers with pooling layers in between for downscaling and a fully connected layer at the end. A total of 13 convolutional layers are used in conjunction with five pooling layers, followed by three fully connected layers. The convolutional layers use a 3x3 filter and a stride of 1, and the pooling layers use a 2x2 filter with a stride of 2. The convolutional layers increase the number of filters from 64 in the first layer to 512 in the last layer. The two first fully connected layers have 4096 channels, and the last one has 1000 for classifying 1000 different objects. All the layers use ReLu as their activation function except for the final layer, which uses a soft-max function to produce the final output prediction. [21]

Although VGG-16 was originally developed to classify objects in images, it should be possible to adapt it to produce a feature vector that can be used in the proposed algorithm. To achieve this, the network has been pretrained on the ImageNet dataset, thereby teaching the network to recognize features. Hereafter, the last three dense layers are removed to output a feature map. By passing an image of the size 224x224 to this network, the resulting output will be a feature map of size 7x7x512. However, only small image patches around the key points found by FAST should be used in the proposed algorithm. Therefore, image patches of size 32x32x3 are passed through the network, consequently resulting in an output size of 1x1x512. Thus, this can be considered a feature descriptor with 512 features.

[21]

4.3.4 MobileNetV2

Another network proposed by (Sandler et al., 2021) that might be used to compute feature descriptors is MobileNetV2 [18]. This network uses two types of computational blocks. One computational block uses something called a residual connection. The idea behind a residual connection is to use skip connections, essentially bypassing one or more layers of data. This kind of architecture allows the network to learn more complex features and helps with the issues of vanishing gradients. The residual blocks output a feature map with the same spatial dimension as the input. In MobileNetV2, another computational block is used to downscale the spatial resolution of the feature maps by using a 3x3 convolution with a stride of 2. As activation functions, a modified version of ReLu is used called ReLu6. Similarly to VGG-16, MobileNetV2 is pretrained on the ImageNet dataset and a feature map of size 1x1x320 can be obtained by removing the last layers of the network and inputting an image patch of size 32x32x3. [18]

4.4 Conclusion

This section has introduced the idea of combining the FAST key point extractor with deep neural networks to obtain key points with attached feature descriptors. Two CNNs have been described and implemented into the proposed algorithm to produce the feature descriptors. Furthermore, a simple odometry algorithm was introduced [14], which can be used to test the proposed feature extractor and compare it to the ORB feature extractor, which is the purpose of the next chapter.

Chapter 5

Tests

5.1 Introduction

The previous chapter described the proposed feature extractor algorithm and a simple VO algorithm that can be used to test it. The purpose of this chapter is to describe how the tests were conducted, the specific parameters used, and the dataset used. As explained in the previous chapter, the proposed feature extractor has been implemented with two different CNNs, which are VGG-16 and MobileNetV2. Both of these implementations are tested and compared to the performance of ORB features. Both the VO algorithm and the feature extractors have some parameters associated with them, which exact values will also be given in this chapter. The results of the tests will be provided in the next chapter, along with a discussion of them.

5.2 The Dataset

Although this project mainly focuses on indoor AR navigation, the KITTI dataset is utilized to test the feature extractors due to its robustness, diversity, and widespread use in benchmarking VO algorithms [8]. The KITTI dataset, captured by a car-mounted camera, consists of 22 image sequences numbered from 00 to 21. While primarily intended for autonomous driving applications, its diverse and complex real-world scenarios make it a reliable testing platform for evaluating VO algorithms, essential for precise and stable AR experiences. In addition, the KITTI dataset can provide insights into the performance of the proposed feature extractor under various conditions, potentially offering valuable information for further development and adaptation for indoor industrial environments.

The KITTI dataset includes both RGB and grayscale images in monocular or stereo formats. For this project, monocular grayscale images from sequence 01 are chosen, using only the first 50 images as a constraint to avoid an unreasonably lengthy algorithm execution. Ground truth files containing transformation matrices describing the camera's true pose and a calibration file containing the camera's intrinsic parameters are provided. Figure 5.1 showcases two example images from the selected image sequence.

It is important to note that despite the outdoor nature of the KITTI dataset, many features present in these images, such as edges, corners, and textures, can also be found in indoor environments. Thus, the performance of the proposed feature extractor on the KITTI dataset can still provide valuable insights into its effectiveness and potential applicability for indoor AR navigation. Moreover, demonstrating strong performance on this challenging dataset indicates that the algorithm may also perform well in indoor settings.

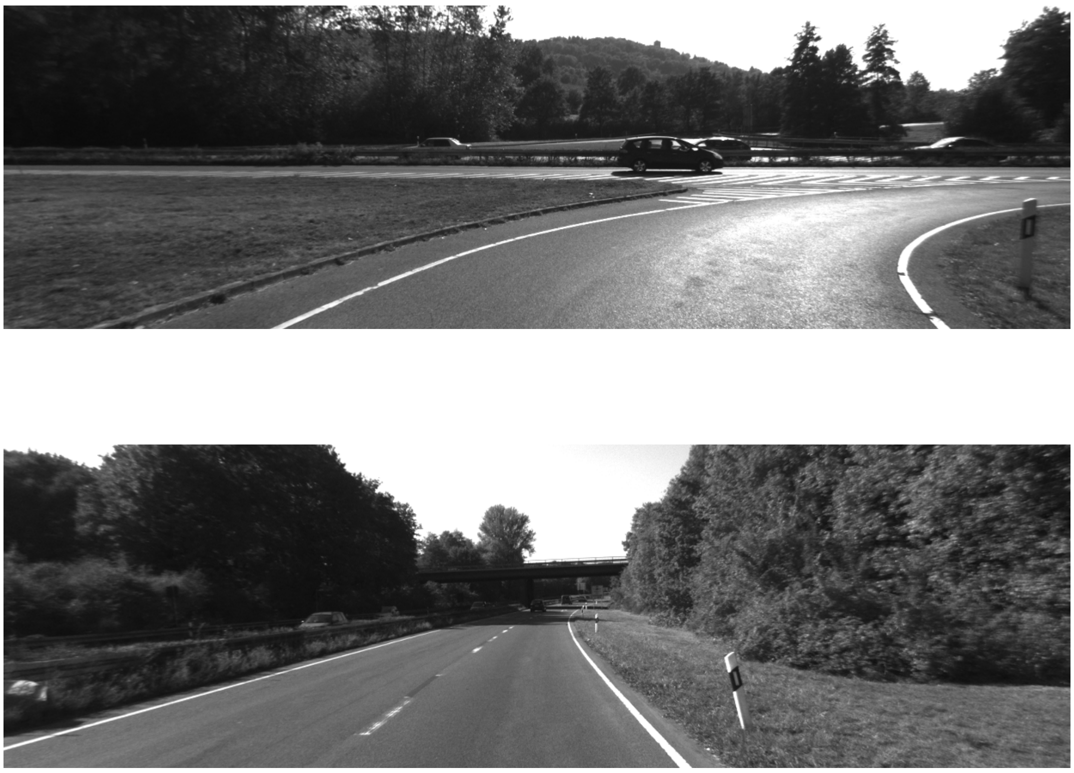


Figure 5.1: Two example images from the KITTI sequence used to test the VO algorithm

5.3 Adjustable Parameters

The VO algorithm used to test the feature extractors and the feature extractors themselves have some associated parameters that influence the algorithm's performance. To find an effective combination of parameters, a selective grid search is performed, testing specific chosen parameter combinations. This section gives a short description of each parameter, followed by a table of the various values tested.

5.3.1 Ratio Test Threshold

When performing feature matching, the ratio test compares the distances between the best-matching feature and the second-best matching feature. If the ratio of the best match's distance to the second-best match's distance is smaller than the threshold, the match is considered valid. By comparing the best and second-best match, the ratio test aims to ensure that a match is not only the best but also significantly better than the next best match. This helps filter out false matches that might have occurred due to noise or repetitive patterns in the images. Adjusting this threshold affects the strictness of filtering, thereby impacting the number of matches.

5.3.2 Mean Based Filtering

The ratio test compares the distance in feature space. However, there might be features that are still wrongly matched. Therefore, another filter is applied, which measures the pixel distance between two matched key points. It calculates the mean value of the distances and creates a threshold based on a multiple of the standard deviation. The multiple of the standard deviation can be adjusted.

5.3.3 Max Number of Features

Both the ORB features and the proposed deep learning-based feature extractor have a parameter limiting the number of found features. This is done by selecting the n best features based on their response values. Selecting the features with the best response values ensures that the most useful features are used while reducing compute time of the algorithm.

5.3.4 Image Pyramid

Image pyramids are used in various computer vision tasks, including feature detection and description. They help capture features at different scales, which is useful when dealing with images with varying scales and resolutions. An image pyramid has two associated parameters.

- **Num scales:** The number of scales in the image pyramid determines the number of downsampled versions of the input image.
- **Scale factor:** The scale factor is a multiplicative factor that determines the size of each successive level in the image pyramid. It is used to resize the image at each level. Adjusting the scale factor impacts the granularity of the detected features at different scales.

Parameter	**Values Tested**
Feature Extractor	[ORB, MobileNetV2, VGG-16]
Max Number of Features	[3000, 6000]
Mean Based Filtering	[False, True]
Standard Deviation Multiplier	[1, 2]

Table 5.1: Parameter values used in the selective grid search.

Chapter 6

Results and Discussion

6.1 Test Results

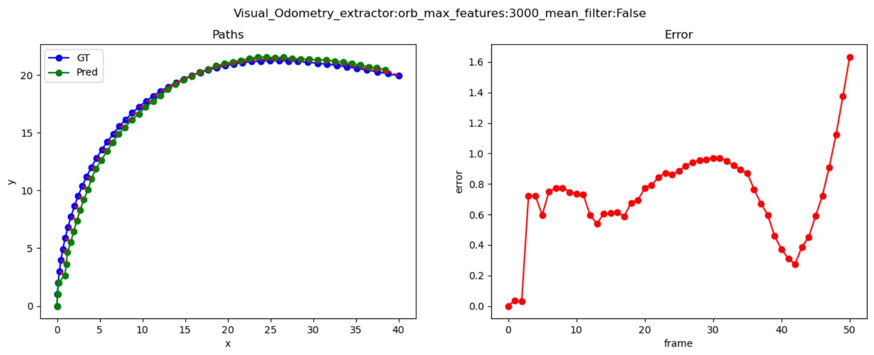
This chapter presents the results of the tests conducted on three different feature extractors, including the original ORB feature extractor and the proposed feature extractor, tested with two different neural networks. A grid search was performed to test different combinations of three parameters for each feature extractor, resulting in 18 tests. The final drift error for each test is displayed in Table 6.1. This error was calculated with equation 6.1, using the final predicted and ground truth x and y positions. The results indicate that, unfortunately, the proposed feature extractor performed worse than the ORB feature extractor, regardless of whether VGG-16 or MobileNetV2 was used. Notably, VGG-16 outperformed MobileNetV2, and one of the tests using MobileNetV2 failed due to insufficient features in one of the frames, resulting in an error when calculating the essential matrix.

			Final Error		
				Proposed Algorithm	
nFeatures	Use Mean Filter	Std Multiplier	ORB	MobileNetV2	VGG-16
3000	False		1.6349	40.9820	24.7457
	True	1	2.5108	36.9791	22.4421
	True	2	2.7650	NaN	22.2870
6000	False		2.2378	32.8022	23.9021
	True	1	2.7200	34.2734	25.0889
	True	2	2.1383	49.7098	27.0332

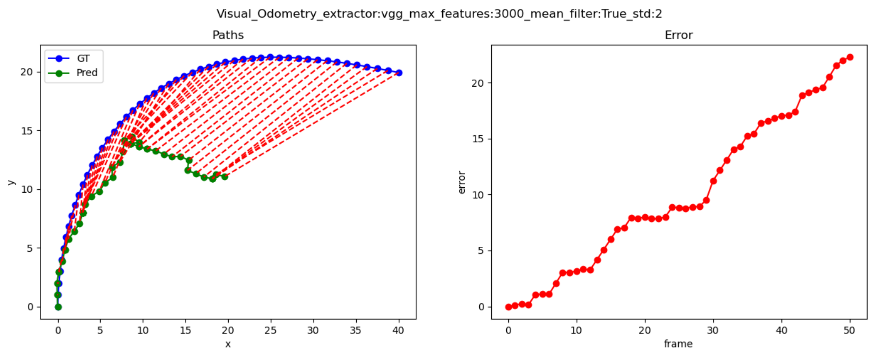
Table 6.1: Final drift error for all combinations of the tested feature extractors and adjustable parameters

To provide a visual representation of the findings, Figure 6.1 shows three graphs, each depicting the predicted and true paths in x and y coordinates as well as the drift error over time. These graphs display the three best tests using the highest-performing parameter combinations for each feature extractor. The graphs clearly show how the drift error accumulates over time. It can be seen that when using ORB features, the predicted path follows the ground truth path much more consistently. Using VGG-16, the predicted path seems to deviate from the true path quickly. However, when using MobileNetV2 it does not seem to follow the true path at all.

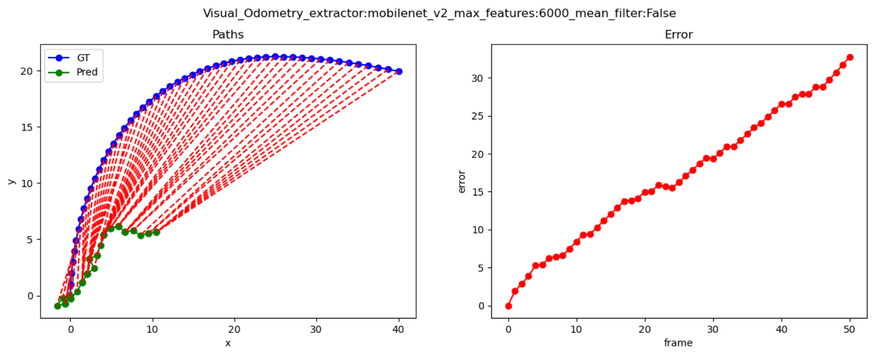
$$\text{Error} = \sqrt{(x_{\text{gt, final}} - x_{\text{pred, final}})^2 + (y_{\text{gt, final}} - y_{\text{pred, final}})^2} \tag{6.1}$$



(a) Using ORB feature extractor



(b) Using VGG-16 feature extractor



(c) Using MobileNetV2 feature extractor

Figure 6.1: Showing three graphs, each depicting the predicted and ground truth paths in x and y coordinates as well as the drift error over time. These graphs display the three best tests using the highest-performing parameters for each feature extractor.

6.2 Discussion

6.2.1 Underperformance of Proposed Feature Extractors

The main findings of this study indicate that the proposed feature extractors did not outperform the original ORB feature extractor in the visual odometry task. Both VGG-16 and MobileNetV2 were tested, with VGG-16 achieving better results compared to MobileNetV2. However, even with the better-performing VGG-16, the proposed feature extractors still fell short of the performance attained by the ORB feature extractor. The primary focus of this discussion is to delve into the possible reasons why the proposed feature extractors did not achieve superior results compared to the ORB feature extractor and explore potential improvements.

6.2.2 Training Data Discrepancy

A notable aspect to consider is the discrepancy between the data on which the neural networks were trained and the data used in the visual odometry task. VGG-16 and MobileNetV2 were pre-trained on the ImageNet dataset, a large-scale dataset comprising varied images ranging from objects and animals to scenes. Nevertheless, the characteristics and features inherent in the ImageNet dataset might not necessarily align with those present in the data utilized for testing the visual odometry algorithm.

This misalignment could result in the extraction of features that, while relevant within the context of ImageNet, might not provide adequate or essential information for our visual odometry task. As a result, the accuracy of camera motion estimation may be negatively affected, contributing to the poor performance of the proposed feature extractors compared to the ORB feature extractor.

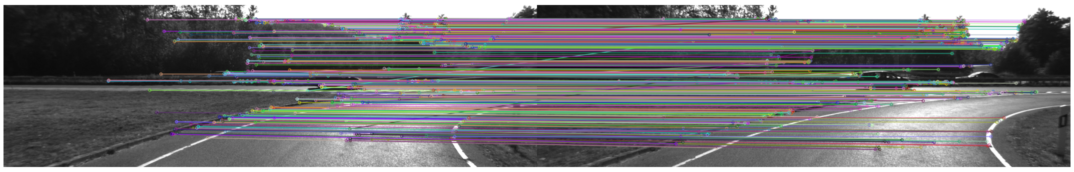
This deviation from expected performance motivates a reassessment of the process, suggesting the potential advantage of training these neural networks on data more reflective of the environments in which the visual odometry algorithm is expected to operate, thus aligning the feature extraction more closely with the task at hand.

6.2.3 Feature Dimensionality

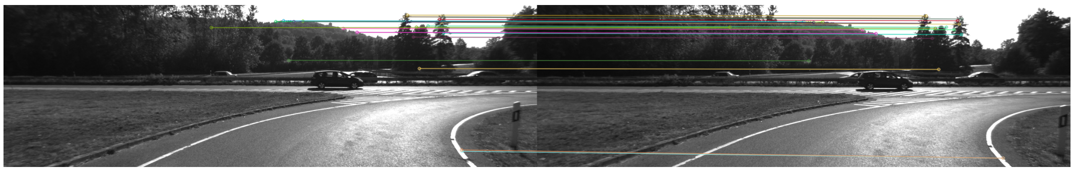
Another factor that might have contributed to the increased drift error when using the proposed feature extractors is the higher dimensionality of the feature vectors generated by VGG-16 and MobileNetV2. These models produce 512-dimensional and 1024-dimensional feature vectors, respectively, compared to the ORB's binary 32-dimensional feature vector.

While the richer representations could potentially capture more nuanced information about the images, the higher dimensionality also increases the complexity of the feature-matching process. Theoretically, it might lead to more false matches due to the 'curse of dimensionality', where the distance between points in a high-dimensional space becomes less meaningful. This could result in more matching errors and hence an increased drift over time.

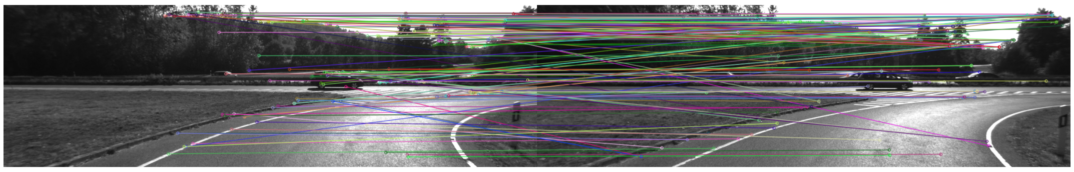
Moreover, the higher dimensionality could potentially make the system more sensitive to noise and variations in image quality, further degrading the performance. Thus, a balance must be struck between the richness of the representation and the robustness and reliability of the matches.



(a) Matched features using ORB feature extraction algorithm.



(b) Matched features using VGG-16 feature extraction algorithm.

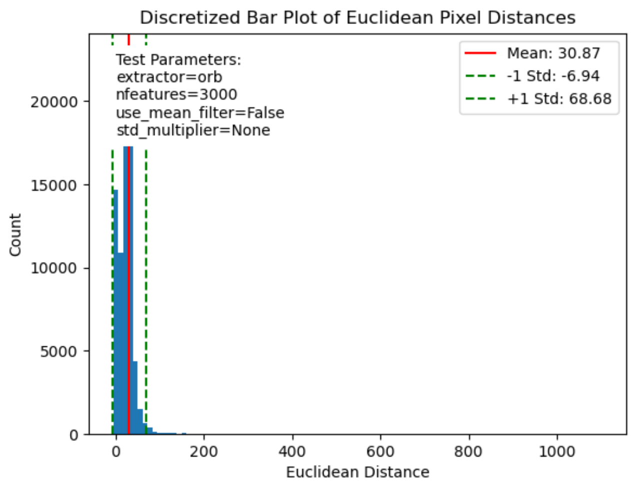


(c) Matched features using MobileNetV2 feature extraction algorithm.

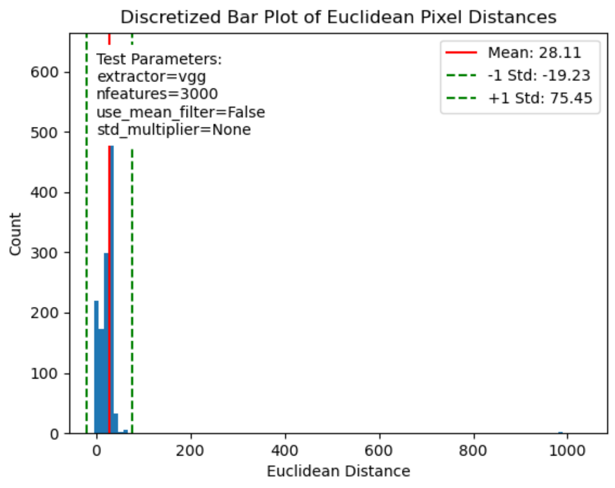
Figure 6.2: Matched features between consecutive frames using different feature extraction algorithms. (a) ORB, (b) VGG-16 with mean filtering, and (c) MobileNetV2. The VGG-16 and MobileNetV2 results are obtained using the proposed algorithm.

6.2.4 Feature Matching Analysis and Filtering

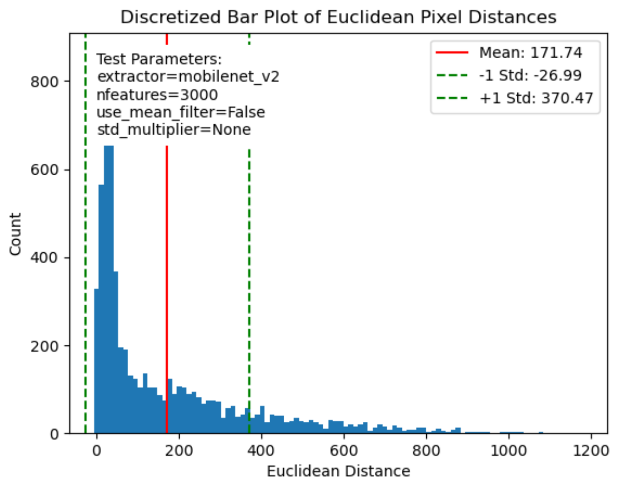
A reasonable assumption is that the matched features do not significantly change their pixel-wise positions from frame to frame. Of course, features far from the camera in real-world coordinates will move more in pixel coordinates than closer features. However, large discrepancies in the matched features' position might indicate wrongly matched features. To gain better insight into this, Figure 6.3 shows a discretized bar chart with the pixel-wise euclidean distance between matched features. The three plots in Figure 6.3 represents a test where no mean filter has been applied yet and with $nfeatures = 3000$. These plots show that using MobileNetV2 results in a much larger standard deviation, which could indicate many incorrectly matched features. The matched ORB and VGG-16 features have a much smaller standard deviation and, thus, fewer incorrectly matched features. Although, the matched ORB features seem to have a slightly smaller standard deviation than the matched VGG-16 features.



(a) Discretized bar plot showing the distribution of the pixel-wise distances between matched key points using ORB features.



(b) Discretized bar plot showing the distribution of the pixel-wise distances between matched key points using VGG-16 features.



(c) Discretized bar plot showing the distribution of the pixel-wise distances between matched key points using MobileNetV2 features.

Figure 6.3: Discretized bar plot showing the distribution of the pixel-wise distances between matched key points.

The idea with the mean filter, described in Section 4.2, is to remove some of these outliers. However, looking at the results in table 6.1, this does not seem to have any significant impact. Similarly, it seems hard to conclude that the number of features significantly impacts the performance since using 6000 features instead of 3000 only improves the performance sometimes, whereas other times, it worsens.

6.2.5 Compute Time

Another essential aspect to consider is the computation required to run the algorithms. The algorithm was tested using a 2016 MacBook Pro running on the CPU, resulting in an average ~0.194-second compute time per frame if using ORB features. However, using MobileNetV2 and VGG-16 took an average of ~92 and ~48 seconds per frame, respectively. The significant increase in compute time is mainly due to two factors. Firstly, performing inference on neural networks takes a significant amount of time. Secondly, MobileNetV2 and VGG-16 produce a feature vector that is 1024 and 512, respectively, compared to ORB, which produces only 32 features. More features result in the knn algorithm taking much longer to find matching features. Although running the algorithms on the aforementioned hardware takes a long time, it might be possible to speed it up using a GPU.

6.2.6 Relation to AR

As stated earlier in the report, VO is an integral part of SLAM algorithms, which is a crucial component in many AR applications. Especially if digital objects need to be placed in fixed real-world positions, as seen in figure 3.3, where a blue sphere is placed at a bin code position in the Selek warehouse.

It is important to note that the dataset used in this study was recorded from a car. While the dataset's origin might seem unrelated to an industrial environment like a warehouse, the findings can still apply to other scenarios, such as warehouse AR applications. This is because the fundamental principles of VO remain the same across different environments, as the goal is to estimate the camera's motion based on visual information.

Several factors contribute to the generalizability of the results to other scenarios. A good feature extractor should be able to recognize and track important features in various environments, regardless of the specific scene or context. The performance comparison of different feature extractors in the car dataset can still provide insights into their suitability for industrial environments.

Both car and warehouse environments may present challenges, such as occlusions, lighting variations, and dynamic objects, which can affect the performance of VO algorithms. Therefore, improvements made to address these challenges in one scenario can also apply to other scenarios.

In conclusion, although the dataset in this study was recorded from a car, the findings and insights gained can still be valuable for other scenarios, such as AR applications in industrial environments. By leveraging the knowledge obtained from this study, researchers and practitioners can develop more effective and robust VO systems for a wide range of AR applications, including those in warehouses and other industrial settings.

Chapter 7

Conclusion

7.1 Summary of Findings

In this thesis, various feature extraction techniques were examined and compared within the context of visual odometry. The objective was to evaluate the performance of the proposed deep learning-based feature extractors, using VGG-16 and MobileNetV2, in contrast to the traditional ORB feature extractor. The experiments were designed to assess the accuracy and robustness of these feature extraction techniques when estimating camera motion. The findings revealed that the ORB feature extractor surpassed both VGG-16 and MobileNet-V2 in the visual odometry task. The ORB algorithm demonstrated superior efficiency and accuracy in estimating camera motion compared to the deep learning-based methods. Furthermore, VGG-16 outperformed MobileNetV2, indicating that the VGG-16 network is better suited for feature extraction in visual odometry tasks, even though it did not exceed the performance of the ORB technique.

In section 3.4.3, an alternative deep learning-based feature extractor was explored, which utilizes synthetic data for key points and feature descriptors. This approach was chosen due to the challenges faced by humans in reliably labeling key points, given their semantically ill-defined nature. The proposed feature extractor harnesses pre-trained convolutional neural networks trained on the ImageNet dataset and extracts features from the last convolutional layer as feature descriptors. For key point detection, the FAST algorithm was implemented. Unfortunately, the results showed that this approach did not meet the expected performance levels. Nonetheless, there is room for further research to enhance these outcomes. In the following sections, potential improvements and future work will be discussed to address the limitations of this deep learning-based feature extractor and to investigate new possibilities for advancing visual odometry techniques.

7.2 Future Work and Improvements

7.2.1 Fine-Tuning Neural Networks

Although this study primarily used pre-trained networks as fixed feature extractors, one potential area for future research would be to fine-tune these networks. Fine-tuning would involve slightly adjusting the weights of the pre-trained network during the training process on the task-specific dataset. This could lead to more robust feature extraction and improved performance. As indicated in the discussion, the VGG-16 and MobileNetV2 networks used in this study were not explicitly trained for visual odometry tasks. Fine-tuning these networks on a visual odometry-specific dataset may improve their performance relative to the ORB extractor.

7.2.2 Dimensionality Reduction Techniques

While the discussion 6.2 pointed out that high-dimensionality of deep learning-based features may pose challenges for subsequent steps, future research could apply dimensionality reduction methods, like Principal Component Analysis (PCA), to the feature descriptors extracted by the neural networks. PCA, which linearly transforms the original data into a set of uncorrelated variables, could help to reduce computational complexity and enhance the efficiency and performance of the feature matching and filtering algorithms.

7.2.3 Exploring Features from Different Network Layers

As indicated in the discussion 6.2, different layers in CNNs learn distinct feature representations. Therefore, future research might benefit from extracting features from various layers within the pre-trained networks and examining the performance of visual odometry algorithms using these features. This could potentially yield a more diverse and comprehensive set of features to be utilized, which could enhance the accuracy and robustness of the visual odometry estimates.

7.2.4 Leveraging GPUs for Computation

Given the computational intensity of deep learning techniques, especially CNNs, future work could explore the benefits of utilizing GPUs for computation. This could significantly reduce computational time, leading to improved efficiency and real-time performance of the feature extraction and visual odometry algorithms. The discussion 6.2 mentioned the higher computational requirements of the deep learning-based methods compared to ORB, so harnessing the computational power of GPUs could make these methods more feasible for real-time applications.

7.2.5 Using a More Relevant Dataset

Lastly, considering the feedback from the discussion about the dataset used in this study, it might be beneficial to evaluate the accuracy of the proposed algorithms using a dataset more closely related to the primary motivation of this study. Using a dataset that accurately represents the unique features and complexities of indoor environments would offer a more accurate assessment of the proposed feature extraction techniques for indoor AR navigation.

7.3 Closing Remarks

In conclusion, this thesis investigated feature extraction techniques in visual odometry and proposed a deep learning-based feature extractor for use in large industrial environments. The experiments demonstrated that traditional feature extraction techniques, such as ORB, surpassed the deep learning-based methods in terms of accuracy and efficiency in estimating camera motion. Nonetheless, the proposed algorithm offers a promising avenue for further research and development. Potential improvements were suggested, such as exploring different layers in pre-trained networks, utilizing GPUs for computation, and evaluating the accuracy of the algorithms using datasets closely related to indoor AR navigation scenarios. As AR technology continues to evolve, this research could contribute to the development of more effective and robust visual odometry algorithms that improve warehouse management processes and other applications in indoor AR navigation.

Bibliography

[1] SELEK DANMARK ApS. *Visit to Selek*. Personal visit. Visited on: 2023-02-08. 2023.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[3] Michael Calonder et al. "BRIEF: Binary Robust Independent Elementary Features". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15561-1.

[4] Carlos Campos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM". In: *CoRR* abs/2007.11898 (2020). arXiv: `2007.11898`. URL: `https://arxiv.org/abs/2007.11898`.

[5] Christine Connolly. "Warehouse management technologies". In: *Sensor Review* 28.2 (2008), pp. 108–114. DOI: `10.1108/02602280810856660`.

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "SuperPoint: Self-Supervised Interest Point Detection and Description". In: *CoRR* abs/1712.07629 (2017). arXiv: `1712.07629`. URL: `http://arxiv.org/abs/1712.07629`.

[7] Tobias Feigl et al. "Localization Limitations of ARCore, ARKit, and Hololens in Dynamic Large-scale Industry Environments". In: Jan. 2020, pp. 307–318. DOI: `10.5220/0008989903070318`.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[9] Apple Inc. *Understanding ARKit Tracking and Detection*. Video recording. Cupertino, CA: Apple Inc., 2018. URL: `https://developer.apple.com/videos/play/wwdc2018/610/`.

[10] Tony Lindeberg. "Scale Invariant Feature Transform". In: vol. 7. May 2012. DOI: `10.4249/scholarpedia.10491`.

[11] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. "ORB-SLAM: a Versatile and Accurate Monocular SLAM System". In: *CoRR* abs/1502.00956 (2015). arXiv: 1502.00956. URL: http://arxiv.org/abs/1502.00956.

[12] Raul Mur-Artal and Juan D. Tardós. "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". In: *CoRR* abs/1610.06475 (2016). arXiv: 1610.06475. URL: http://arxiv.org/abs/1610.06475.

[13] Raul Mur-Artal and Juan D. Tardós. "Fast relocalisation and loop closing in keyframe-based SLAM". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 846–853. DOI: 10.1109/ICRA.2014.6906953.

[14] Nicolai Høirup Nielsen. *Computer Vision: Visual Odometry*. https://github.com/niconielsen32/ComputerVision/tree/master/VisualOdometry. 2022.

[15] Shashi Poddar, Rahul Kottath, and Vinod Karar. "Evolution of Visual Odometry Techniques". In: *CoRR* abs/1804.11142 (2018). arXiv: 1804.11142. URL: http://arxiv.org/abs/1804.11142.

[16] Edward Rosten and Tom Drummond. "Machine Learning for High-Speed Corner Detection". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443. ISBN: 978-3-540-33833-8.

[17] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[18] Mark Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". In: *CoRR* abs/1801.04381 (2018). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[19] Bjo rn Schwerdtfeger et al. "Pick-by-Vision: A First Stress Test". In: IEEE, 2009. DOI: 10.1109/ISMAR.2009.5336484.

[20] Björn Schwerdtfeger and Gudrun Klinker. "Supporting Order Picking with Augmented Reality". In: *IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2008. DOI: 10.1109/ISMAR.2008.4637331.

[21] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). URL: http://arxiv.org/abs/1409.1556.

[22] Marie-H'el'ene Stoltz et al. "Augmented Reality in Warehouse Operations: Opportunities and Barriers". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 12979–12984.

[23] *Tasklet Factory*. https://taskletfactory.com. Accessed March 6, 2023.

[24] Chen Yunqiang et al. “An overview of augmented reality technology”. In: Journal of Physics: Conference Series, 2019. DOI: 10.1088/1742-6596/1237/2/022082.

[25] Zebra Technologies Corporation. *PRODUCT SPEC SHEET: TC21/TC26 Touch Computer*. Online. Accessed: March 6, 2023. 2023. URL: https://www.zebra.com/content/dam/zebra_dam/en/spec-sheets/tc21-tc26-spec-sheet-en-us.pdf.